Effective Decision Tree Learning

C.Sudarsana Reddy
Department of Computer Science and Engineering,
S.V.University College of Engineering, Tirupati.

V.Vasu, Ph.D
Department of Mathematics,
S.V.University, Tirupati

B.Kumara Swamy Achari
Department of Mathematics,
S.V.University, Tirupati

ABSTRACT
Classification is a data analysis technique. The decision tree is one of the most popular classification algorithms in current use for data mining because it is more interpretable. Training data sets are not error free due to measurement errors in the data collection process. Traditional decision tree classifiers are constructed without considering any errors in the values of attributes of the training data sets. We extend such classifiers to construct effective decision trees with error corrected training data sets. It is possible to build decision tree classifiers with higher accuracies especially when the measurement errors in the values of the attributes of the training data sets are corrected appropriately before using those training data sets in decision tree learning. Error corrected data sets can be used not only in decision tree learning but also in many data mining techniques.

In general, values of attributes in training datasets are always inherently associated with errors. Data errors can be properly handled by using appropriate error models or error correction techniques. Also, sometimes for preserving data privacy, attribute values in the original training data sets are modified so that modified data sets contain data values with some errors. Later on, these modified data sets are reconstructed before applying those tuples to data mining technique.

This paper introduces an effective decision tree (EDT) construction algorithm that uses a new error adjusting technique (NEAT) in constructing more accurate decision tree classifiers. The idea behind this new error adjusting technique is that 'many data sets with numerical attributes containing point data values have been collected via repeated measurements' and the process of repeated measurements is the common source of data errors in the training data sets. EDT describes an approach to correct the errors in the values of attributes of the training data sets and then error corrected attribute values of the data sets are used in decision tree learning.

Keywords: Decision tree, Classification, Data mining

1. INTRODUCTION
Data mining has many applications in research, science, engineering and business. Classification is an important technique in machine learning and data mining. Decision tree is the most commonly used data classification technique [1]. When decision trees are used for classification they are called classification trees [2].

Traditional decision trees are constructed by using training data sets as it is without considering inherent errors present in the training data sets. But in real life many training data sets contain errors in the values of the attributes of the training datasets. We introduce an effective decision tree (EDT) construction algorithm by using error corrected training data sets. In contrast to the traditional decision tree (TDT), our approach, effective decision tree (EDT) construction, produces more accurate decision tree classifiers by considering error adjusting technique. No data set is available without error. That is, there are no error free data sets particularly when the data sets containing numerical attributes.

EDT algorithm uses error correction or error adjustment technique based on the assumption that data sets are not always error free and it is likely that some sort of measurement errors are present in the collection process of data sets. Errors are inevitable in all training data sets particularly in the training data sets with numerical attributes containing point data.

2. PROBLEM STATEMENT
Traditional decision tree (TDT) classifiers are constructed by using training data sets directly without considering inherent data errors associated with values of attributes in the training data sets. Hence, traditional decision tree (TDT) classifiers produce incorrect or less accurate data mining results. As errors are associated with training data sets, it is important to develop effective, efficient and more accurate data mining techniques by taking error decreased attribute values of the training data sets.

Also, for preserving data privacy sometimes training data sets are modified or injected certain error values into the values of attributes in the training data sets in a systematic or controlled way. So, in such cases data sets contain errors with modified attribute values. Such modified data sets must be reconstructed by finding and removing errors in the modified training data sets before applying them in any data mining technique. So, usually errors are present in training data sets.

In traditional decision tree construction each tuple ti is associated with a feature vector, which is represented as

$$V_i = (v_{i,1}, v_{i,2}, v_{i,3}, \dots, v_{i,k}, class\_label)$$

where 'i' is the tuple number and 'k' is the number of attributes in the training data set. Decision tree is constructed using training data set and then the resulting classifier is tested using test data set. To find the class label of an unseen (new) test tuple $t_0 = (v_{0,1}, v_{0,2}, v_{0,3}, \dots, v_{0,k}, ?)$, it is required to traverse the decision tree from root node to a specific leaf node.

The present study proposes an algorithm to improve accuracy and performance of the traditional decision tree (TDT) algorithm. This new algorithm is called effective decision tree (EDT) construction algorithm. EDT assumes various percentages of errors in the attribute values and modifies values of attributes in the training data sets accordingly and then decision tree classifier is constructed using error corrected training data sets.

3. EXISTING ALGORITHM

3.1 Traditional Decision Tree (TDT) Algorithm Description

The traditional decision tree (TDT) algorithm constructs a decision tree splitting each node into left and right nodes. Initially, the root node contains all the training tuples. The process of partitioning the tuples in a node into two sets based on the value of an attribute and storing the resulting tuples in its left and right nodes is referred to as splitting. Whenever further split of a node is not required then it becomes a leaf node referred to as an external node. All other nodes except root node are referred as internal nodes. The splitting process at each internal node is carried out recursively until no further split is required. Further splitting of an internal node is stopped if one of the stopping criteria given hereunder is met.

1. All the tuples in an internal node have the same class label.
2. Splitting does not result nonempty left and right nodes.

In the first case, the probability for that class label is set to 1 whereas in the second case, the internal node becomes external node. The empirical probabilities are computed for all the class labels of that node. The best split pair comprising an attribute and its value is that associated with minimum entropy.

Entropy is a metric or function that is used to find the degree of dispersion of training data tuples in a node. In decision tree construction the goodness of a split is quantified by an impurity measure. One possible function to measure impurity is entropy. Entropy is an information based measure and it is based only on the proportions of tuples of each class in the training data set. Entropy is used for finding how much information content is there in a given data.

Entropy is taken as dispersion measure because it is predominantly used for constructing decision trees. In most of the cases, entropy finds the best split and balanced node sizes after split in such a way that both left and right nodes are as much pure as possible.

Accuracy and execution time of TDT algorithm for 9 data sets are shown in Table 5.2

Entropy is calculated using the formula

$$entropy(S) = \sum_{i=1}^{m} -p_i . log_2(p_i)$$

Where p_i = number of tuples belongs to the i^{th} class

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

$$H(z, A_j) = \frac{|L|}{|S|} \left(\sum_{c \in C} -\frac{p_c}{L} log_2 \left(\frac{p_c}{L} \right) \right) + \frac{|R|}{|S|} \left(\sum_{c \in C} -\frac{p_c}{R} log_2 \left(\frac{p_c}{R} \right) \right) \quad 3.1$$

$$H(z, A_j) = \frac{|L|}{|S|} \left(Entropy(L) \right) + \frac{|R|}{|S|} \left(Entropy(R) \right)$$

Where

A_j is the splitting attribute.

L is the total number of tuples to the left side of the split point z.

R is the total number of tuples to the right side of the split point z.

$\frac{p_c}{L}$ is the number of tuples belongs to the class label c to the left side of the split point z.

$\frac{p_c}{R}$ is the number of tuples belongs to the class label c to the right side of the split point z.

S is the total number of tuples in the node.

3.2 Pseudo code for Traditional Decision Tree (TDT) Algorithm

TRADITIONAL_DECISION_TREE (T)

1. If all the training tuples in the node T have the same class label then
2. set $p_T(c) = 1.0$
3. return(T)
4. If tuples in the node T have more than one class then
5. Find_Best_Split(T)
6. For i ← 1 to datasize[T] do
7. If split_atribute_value[t_i] <= split_point[T] then
8. Add tuple t_i to left[T]
9. Else
10. Add tuple t_i to right[T]
11. If left[T] = NIL or right[T] = NIL then
12. Create empirical probability distribution of the node T
13. return(T)
14. If left[T] != NIL and right[T] != NIL then
15. TRADITIONAL_DECISION_TREE(left[T])
16. TRADITIONAL_DECISION_TREE(right[T])
17. return(T)

4. PROPOSED ALGORITHM

4.1 Proposed Effective Decision Tree (EDT) Algorithm Description

The proposed algorithm called effective decision tree (EDT) algorithm constructs a decision tree classifier splitting each node into left and right nodes. Initially, the root node contains all the training data tuples with numerical attributes containing point data. The process of partitioning the tuples in a node into two sets based on the best split value of an attribute and storing the resulting tuples in its left and right nodes is referred to as splitting. Whenever further split of a node is not required then it becomes a leaf node referred to as an external node. All other nodes except root node are referred as internal nodes. The splitting process at each internal node is carried out recursively until no further split is required. Further splitting of an internal node is stopped if one of the stop stopping criteria given hereunder is met.

1. All the tuples in an internal node have the same class label.
2. Splitting does not result nonempty left and right nodes.

In the first case, the probability for that class label is set to 1.0 whereas in the second case, the internal becomes external (or leaf) node. The empirical probabilities are computed for all the class labels of that node. The best split pair comprising an attribute and its value is that associated with minimum entropy.

Entropy is computed for all values of all attributes of all the training data tuples in the current node. Value of an attribute is modified using new error adjusting technique (NEAT)

before computing entropy value. That is, entropy values are computed for modified values of attributes.

Entropy is a metric or function that is used to find the degree of dispersion of training data tuples in a node. In decision tree construction the goodness of a split is quantified by an impurity measure [2]. One possible function to measure impurity is entropy. Entropy is an information based measure and it is based only on the proportions of tuples of each class in the training data set. Entropy is used for finding how much information content is there in a given data.

Entropy is taken as dispersion measure because it is predominantly used for constructing decision trees. In most of the cases, entropy finds the best split and balanced node sizes after split in such a way that both left and right nodes are as much pure as possible.

For the proposed effective decision tree (EDT) algorithm accuracy and execution time for 9 data sets are shown in Table 5.5 and execution time comparisons for 9 datasets are shown Figure 5.3. Also execution time and accuracy comparisons of TDT and EDT are shown in Table 6.6 and charted in Figure 5.3 and Figure 5.4 respectively.

Classification accuracy improvements for 9 data sets by considering error adjustment technique in EDT are shown in Table 5.5 for different error values.

Entropy is calculated using the formula

$$entropy(S) = \sum_{i=1}^{m} -p_i . log_2(p_i)$$

Where p_i = number of tuples belongs to the i^{th} class

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

$$H(z, A_j) = \frac{|L|}{|S|} \left(\sum_{c \in C} -\frac{p_c}{L} log_2 \left(\frac{p_c}{L} \right) \right) + \frac{|R|}{|S|} \left(\sum_{c \in C} -\frac{p_c}{R} log_2 \left(\frac{p_c}{R} \right) \right) \quad 4.1$$

$$H(z, A_j) = \frac{|L|}{|S|} \left(Entropy(L) \right) + \frac{|R|}{|S|} \left(Entropy(R) \right)$$

Where

A_j is the splitting attribute.

L is the total number of tuples to the left side of the split point z.

R is the total number of tuples to the right side of the split point z.

$\frac{p_c}{L}$ is the number of tuples belongs to the class label c to the left side of the split point z.

$\frac{p_c}{R}$ is the number of tuples belongs to the class label c to the right side of the split point z.

S is the total number of tuples in the node.

Proposed effective decision tree (EDT) algorithm uses a special new error adjusting technique (NEAT) in constructing a decision tree classifier. The idea behind this new error adjusting technique is that 'many data sets with numerical attributes containing point data have been collected via repeated measurements' and the process of repeated measurements is the common source of data errors in the training data sets. No data set is available without error. That is, there are no error free data sets particularly the data sets containing numerical attributes. EDT algorithm uses error correction or error adjustment technique based on the assumption that data sets are not always error free and it is likely that some sort of measurement errors are present in the collection process of data sets. Errors are inevitable in all training data sets particularly in the training data sets with numerical attributes containing point data.

EDT gives more accurate results than the TDT method for constructing the decision tree classifier with data sets containing numerical attributes with point data. Error adjusting technique is applied for training data sets with numerical attributes containing point data for different error values and then decision tree classifiers are constructed. EDT is similar to the TDT algorithm in all respects except that EDT additionally uses error adjusting technique. Various error values are considered during effective decision tree (EDT) classifier construction. Entropy values are computed for each attribute value after modifying the attribute value using a specified error value.

Various error correction values are:

$$\pm 0.1, \ \pm 0.01, \ \pm 0.001, \ \pm 0.0001, \ \pm 0.00001, \ \pm 0.000001 \ etc$$

Before finding entropy value, attribute value is modified as follows:

value = value + 0.1; or value = value – 0.1; or value = value + error; or value = value – error ; or value = value + value * 0.1; or value = value – value * 0.1; or value = value + value * 0.1 * 0.1; or value = value – value * 0.1 * 0.1;

4.2 Pseudo code for Effective Decision Tree (EDT) Algorithm

EFFECTIVE_DECISION_TREE(T)

1. If all the training tuples in the node T have the same class label then
2. set $p_T(c) = 1.0$
3. return(T)
4. If tuples in the node T have more than one class label then
5. Apply error adjusting technique to each value of each attribute and find entropy
6. Find_Best_Split(T)
7. For i ← 1 to datasize[T] do
8. If split_atribute_value[t_i] <= split_point[T] then
9. Add tuple t_i to left[T]
10. Else
11. Add tuple t_i to right[T]
12. If left[T] = NIL or right[T] = NIL then
13. Create empirical probability distribution of the node T
14. return(T)
15. If left[T] != NIL and right[T] != NIL then
16. Effective_Decision_Tree (left[T])
17. Effective_Decision_Tree(right[T])
18. return(T)

5. EXPERIMENTS ON EFFICIENCY

A simulation model is developed for evaluating the performance of two algorithms: traditional decision tree (TDT) and effective decision tree (EDT) experimentally. The data sets shown in Table 5.1 from University of California (UCI) Machine Learning Repository are employed for evaluating the performance of the above said algorithms.

S No.	Data Set	Training Tuples	No.of Attributes	No of Classes	Test Tuples
1	Iris	150	4	3	10-
2	Glass	214	9	6	10-
3	IonoSphere	351	32	2	10-
4	BreastCancer	569	30	2	10-
5	Vehicle	846	18	4	10-
6	Segment	2310	14	7	10-
7	Satellite	4435	36	6	2000
8	PageBlock	5473	10	5	10-
10	Pen Digits	7494	16	10	3498

Table 5.1 Data Sets from the UCI Machine Learning Repository

10-fold cross-validation technique is used for test tuples for all training data sets with numerical attributes except Satellite and PenDigits training data sets. For Satellite and PenDigits training data sets with numerical attributes a separate test data set is used for testing.

The simulation model is implemented in Java 1.6 on a Personal Computer with 3.22 GHz Pentium Dual Core processor (CPU), and 2 GB of main memory (RAM). The performance measures, accuracy and execution time, for the above said algorithms are presented in Table 5.2 to Table 5.13 and Figure 5.1 to Figure 5.4.

No	DataSet Name	Total Training Tuples	Correctly classified tuples	Accuracy on training data	k-fold cross validation accuracy	Execution time (seconds)
1	Iris	150	147	98.0	97.333333	0.172
2	Glass	214	183	85.5140	90.952381	0.422
3	Ionosphere	351	292	83.1901	82.2857	0.875
4	Breast Cancer	569	559	98.24253	96.60714	2.234
5	Vehicle	846	673	79.55083	78.928571	8.109
6	Segment	2310	2255	97.619048	96.62338	49.969
7	Satellite	4435 (Test 2000)	1665	83.25	Not applied	269.218
8	Page block	5473	5472	99.981728	99.561243	61.203
9	Pen Digits	7494 (Test3498)	3215	91.91	Not applied	1220.406

Table 5.2 Accuracy and Execution Time of TDT Algorithm for 9 Data sets

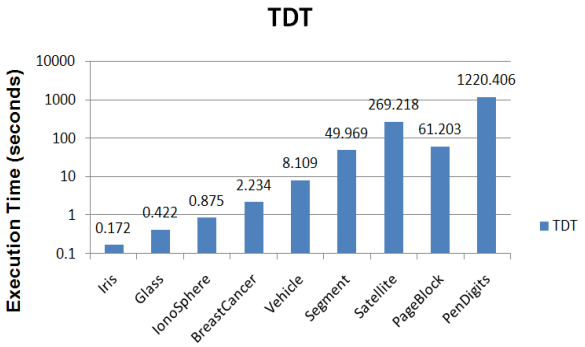

Figure 5.1 Execution Times for TDT Algorithm for 9 Data Sets.

No	Data Set Name	Total Training Tuples	Correctly classified tuples	Error	Accuracy on training data	k-fold cross validation accuracy	Execution time (seconds)
1	Iris	150	149	10^{-2}	99.333333	98.0	0.178
2	Glass	214	205	10^{-5}	95.79439	92.38095	0.422
3	Ionosphere	351	349	10^{-3}	99.4302	98.5714	1.844
4	BreastCancer	569	566	10^{-4}	99.472759	96.607142	2.359
5	Vehicle	846	780	10^{-4}	92.198582	89.642857	7.938
6	Segment	2310	2292	10^{-8}	99.220779	97.619048	51.719
7	Satellite	4435 Test	1693	10^{-3}	84.65	Not applied	267.485
8	Pageblock	5473	5470	10^{-5}	99.945185	99.542962	61.531
9	PenDigits	7494 Test	3185	10^{-2}	91.05203	Not applied	1221.156

Table 5.3 Accuracy and Execution Time of EDT Algorithm for 9 Data sets

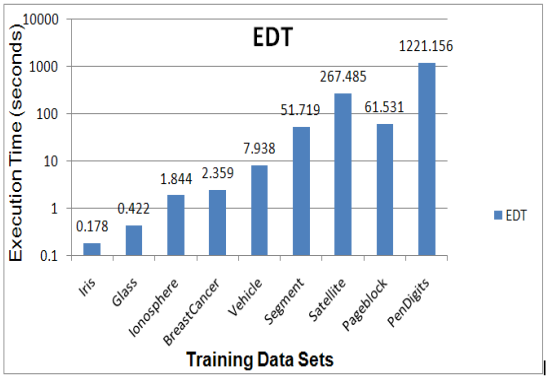

Figure 5.2 Execution Times for EDT Algorithm for 9 Data Sets.

S. No	Data Set Name	TDT Execution Time	EDT Execution Time	TDT Accuracy	EDT Accuracy
1	Iris	0.172	0.178	97.333333	98.0
2	Glass	0.422	0.422	90.952381	92.38095
3	IonoSphere	0.875	1.844	82.2857	98.5714
4	BreastCancer	2.234	2.359	96.60714	96.607142
5	Vehicle	8.109	7.938	78.928571	89.642857
6	Segment	49.969	51.719	96.62338	97.619048
7	Satellite	269.218	267.485	83.25	84.65
8	PageBlock	61.203	61.531	99.561243	99.542962
9	PenDigits	1220.406	1221.156	91.91	91.05203

Table 5.4 Comparison of Accuracy and Execution Time for TDT and EDT Algorithms for 9 Data Sets

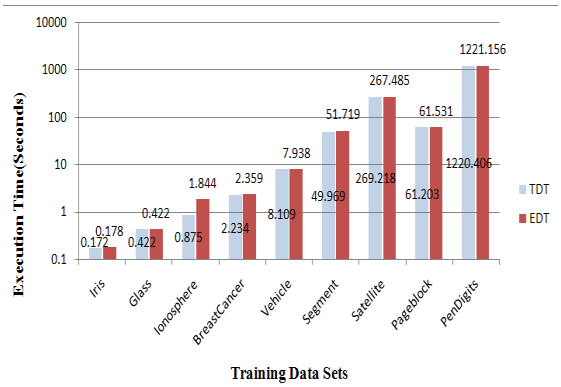

Figure 5.3 Comparisons of Execution Times for TDT and EDT Algorithms

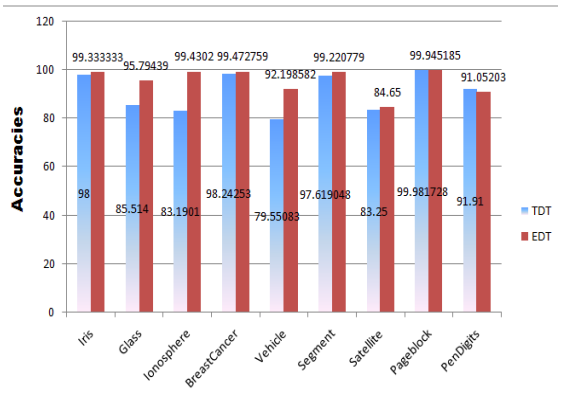

Figure 5.4 Comparisons of Accuracies for TDT and EDT Algorithms

Data Set	TDT	Best Case	Proposed Effective Decision Tree Algorithm (EDT)					
			0.1	0.01	0.001	0.0001	0.00001	0.000001
Iris	98.0	99.3333	94.6667	**99.3333**	99.3333	99.3333	99.3333	99.3333
Glass	85.5140	95.7944	59.813	71.028	74.299	91.589	**95.7944**	95.7944
Iono	83.1901	99.4302	86.8946	99.4302	**99.4302**	99.4302	99.4302	99.4302
Breast	98.2425	99.4728	94.0246	98.7698	99.4728	**99.4728**	99.4728	99.4728
Vehicle	79.5508	92.1986	59.4562	81.5603	**92.1986**	92.1986	92.1986	92.1986
Segment	97.6190	99.2208	82.2944	96.2771	98.4416	98.2251	98.2251	98.2251 98.4416 98.7446 **99.2208**
Satellite	83.25	84.65	69.45	84.55	**84.65**	84.65	84.65	84.65
PageBlock	99.9817	99.9452	98.52	99.6528	99.7259	99.7807	**99.9452**	99.9452
PenDigits	91.91	91.052	90.7947	**91.052**	91.052	91.052	91.052	91.052

Table 5.5 Accuracy of EDT Algorithm for Various Error Values

Proposed Effective Decision Tree (EDT) Algorithm with different error values is compared with Traditional Decision Tree (TDT) algorithm. EDT accuracies are calculated and shown in the Table 5.5. Bold values show highest classification accuracies when EDT is employed by using new error adjusting technique (NEAT). For Iris data set maximum error value is 0.01 and maximum classification accuracy is 98 when maximum 0.01 is removed from training data set.

SNo.	Total Training Tuples	Correctly Classified Tuples	Error	Accuracy on Training Data	k-fold Cross Validation Accuracy
1	150	142	10^{-1}	94.666666	94.666666
2	150	149	10^{-2}	99.333333	**98.0**
3	150	149	10^{-3}	99.333333	98.0
4	150	149	10^{-4}	99.333333	98.0
5	150	149	10^{-5}	99.333333	98.0

Table 5.6 Accuracy Details of EDT Algorithm for Iris Data Set for Different Error Values

SNO.	Total Training Tuples	Correctly Classified Tuples	Error	Accuracy on Training Data	k-fold Cross Validation Accuracy
1	214	128	10^{-1}	59.813	63.8095
2	214	152	10^{-2}	71.028	70.9524
3	214	159	10^{-3}	74.299	70.9524
4	214	196	10^{-4}	91.589	**92.38095**
5	214	205	10^{-5}	95.79439	92.38095
6	214	205	10^{-6}	95.79439	92.38095
7	214	205	10^{-7}	95.79439	92.38095
8	214	205	10^{-8}	95.79439	92.38095

Table 5.7 Accuracy Details of EDT Algorithm for Glass Data Set for Different Error Values

S.No.	Total Training Tuples	Correctly Classified Tuples	Error	Accuracy on Training Data	k-fold Cross Validation Accuracy
1	351	305	10^{-1}	86.8946	84.5714
2	351	349	10^{-2}	99.4302	98.285714
3	351	349	10^{-3}	99.4302	**98.5714**
4	351	349	10^{-4}	99.4302	98.5714
5	351	349	10^{-5}	99.4302	98.5714
6	351	349	10^{-6}	99.4302	98.5714
7	351	349	10^{-7}	99.4302	98.5714

Table 5.8 Accuracy Details of EDT Algorithm for IonoSphere Data Set for Different Error Values

S.No.	Total Training Tuples	Correctly Classified Tuples	Error	Accuracy on Training Data	k-fold Cross Validation Accuracy
1	569	535	10^{-1}	94.0246	93.571428
2	569	562	10^{-2}	98.7698	96.249999
3	569	566	10^{-3}	99.472759	96.428571
4	569	566	10^{-4}	99.472759	**96.607142**
5	569	566	10^{-5}	99.472759	96.607142
6	569	566	10^{-6}	99.472759	96.607142

Table 5.9 Accuracy Details of EDT Algorithm for BreastCancer Data Set for Different ErrorValues

S. No.	Total Training Tuples	Correctly Classified Tuples	Error	Accuracy on Training Data	k-fold Cross Validation Accuracy
1	846	503	10^{-1}	59.456265	54.7619
2	846	690	10^{-2}	81.56028	85.47619
3	846	780	10^{-3}	92.198582	**89.642857**
4	846	780	10^{-4}	92.198582	89.642857
5	846	780	10^{-5}	92.198582	89.642857

Table 5.10 Accuracy Details of EDT Algorithm for Vehicle Data Set for Different Error Values

S. No.	Total Training Tuples	Correctly Classified Tuples	Error	Accuracy on Training Data	k-fold Cross Validation Accuracy
1	2310	1901	10^{-1}	82.294372	81.125541
2	2310	2224	10^{-2}	96.277056	96.233766
3	2310	2274	10^{-3}	98.441558	97.099567
4	2310	2269	10^{-4}	98.225108	97.272727
5	2310	2269	10^{-5}	98.225108	97.229437
6	2310	2274	10^{-6}	98.441558	96.883117
7	2310	2281	10^{-7}	98.744589	97.489177
8	2310	2292	10^{-8}	99.220779	**97.619048**
9	2310	2292	10^{-9}	99.220779	97.619048
10	2310	2292	10^{-10}	99.220779	97.619048

Table 5.11 Accuracy Details of EDT Algorithm for Segment Data Set for Different ErrorValues

S. No.	Total Training Tuples	Correctly Classified Tuples	Error	Accuracy on Training Data
1	4435(test 2000)	1389	10^{-1}	69.45
2	4435(test 2000)	1691	10^{-2}	84.55
3	4435(test 2000)	1693	10^{-3}	**84.65**
4	4435(test 2000)	1693	10^{-4}	84.65
5	4435(test 2000)	1693	10^{-5}	84.65
6	4435(test 2000)	1693	10^{-6}	84.65

Table 5.12 Accuracy Details of EDT Algorithm for Satellite Data Set for Different ErrorValues

S. No.	Total Training Tuples	Correctly Classified Tuples	Error	Accuracy on Training Data	k-fold Cross Validation Accuracy
1	5473	5392	10^{-1}	98.520007	98.20841
2	5473	5454	10^{-2}	99.652841	99.15905
3	5473	5458	10^{-3}	99.725927	99.341865
4	5473	5461	10^{-4}	99.780742	99.396709
5	5473	5470	10^{-5}	99.945185	**99.542962**
6	5473	5470	10^{-6}	99.945185	99.542962
7	5473	5470	10^{-7}	99.945185	99.542962

Table 5.13 Accuracy Details of EDT Algorithm for PageBlock Data Set for Different ErrorValues

S. No.	Total Training Tuples	Correctly Classified Tuples	Error	Accuracy on Training Data
1	7494(test 3498)	3176	10^{-1}	90.79474
2	7494(test 3498)	3185	10^{-2}	**91.05203**
3	7494(test 3498)	3185	10^{-3}	91.05203
4	7494(test 3498)	3185	10^{-4}	91.05203

Table 5.14 Accuracy Details of EDT Algorithm

for PenDigits Data Set for Different ErrorValues
For all 9 training data sets estimated errors are shown in table 5.15

S. No	Data Set Name	Error value
1	Iris	0.01
2	Glass	0.0001
3	IonoSphere	0.001
4	BreastCancer	0.0001
5	Vehicle	0.001
6	Segment	0.00000001
7	Satellite	0.001
8	PageBlock	0.00001
9	PenDigits	0.01

Table 5.15 Estimated error values in the training data sets

6. CONCLUSIONS

6.1 Contributions

The performance of traditional decision tree (TDT) algorithm is verified experimentally. A new algorithm, Effective Decision Tree (EDT) is proposed and compared with traditional decision tree (TDT). It is found that the classification accuracy of EDT algorithm is better than TDT algorithm with almost same computational effort and same execution times.

6.2 Limitations

Construction of decision tree classifiers for large training data sets is less efficient and less scalable. Some privacy preserving techniques cause reduced utility of training data sets.

6.3 Suggestions for future work

Scalable and efficient techniques are needed for constructing decision tree classifiers. Special privacy preserving techniques are needed to maintain training data sets without loss of utility and accuracy when privacy preserving techniques are applied to training data sets. Also effective, efficient, and simple techniques are needed to reconstruct the modified training datasets before applying data mining techniques.

7. REFERENCES

[1] Jiawei Han, Micheline Kamber , Data Mining:Concepts and Techniques, Morgan Kaufmann, 2006.

[2] Introduction to Machine Learning EthemAlpaydin

[3] 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, 1996.

[4] R.E.Walpole and R.H. Myers, Probability and Statistics for Engineers and Scientists. Macmillan Publishing Company, 1993.

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

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

[7] J.R. Quinlan, "Improved Use of Continuous attributes in C4.5" , Journal of Artificial Intelligence Research, 4, pp. 77-90, 1996.