International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 2 684 – 690

Comparative Analysis of Various Decision Tree Classification Algorithms using WEKA

Priyanka Sharma
Assistant Professor
Dept. of Computer Science and Applications
Chaudhary Devi Lal University
Sirsa, India
e-mail: pinki.sharma2912@gmail.com

***Abstract*—** Classification is a technique to construct a function or set of functions to predict the class of instances whose class label is not known. Discovered knowledge is usually presented in the form of high level, easy to understand classification rules. There is various classification techniques used to classify the data, one of which is decision tree algorithms. This paper presents a comparative analysis of various decision tree based classification algorithms. In experiments, the effectiveness of algorithms is evaluated by comparing the results on 5 datasets from the UCI and KEEL repository.

Keywords-*classification; weka; decision tree; data mining*

*****

I. INTRODUCTION

Databases are increasing day-by-day. To get any information the large databases is not only very difficult but also mind boggling. Classification is a technique which can be used to construct such a function or model which can be used to find class of unknowns. Broadly speaking, classification [1][2][3][4] is the process of finding a model which describes and distinguishes data classes or concepts, for the purposes of being able to use the model to predict the class of objects whose class labels are unknown. The derived model is based on the analysis of the set of training data (i.e., data objects whose class label is known) and predicts categorical class labels. There are various classification techniques which can be used for labeling of unknown instances. Decision tree based algorithms are most widely used techniques of classification. Decision tree is an acyclic graph like structure having root as well age internal nodes as test conditions and leaf nodes as class labels. An unknown class label for any tuple can be easily found by tracing the path from root node to a leaf node which holds the class for that tuple. WEKA is generally used for classifying datasets using decision tree algorithms. WEKA is an open source, platform independent and easy to use data mining tool issued under GNU General Public License. It contains tools for data pre-processing, classification, regression, clustering, association rules and visualization. WEKA provides many different algorithms for data mining and machine learning.

This paper presents comparative analysis of various decision tree classification algorithms using WEKA. In section 2, classification process is discussed. Section 3 presents various decision tree algorithms which will be compared. In section 4, WEKA has been discussed using which various decision tree algorithms for classification have been compared. Section 5 and 6 presents implementation and results of the analysis. Section 7 represents concluding remarks.

II. CLASSIFICATION

Wherever Classification[1] is probably the most widely used data mining technique. It predicts categorical class labels and classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it to classify unseen data. A model or classifier is constructed to predict categorical labels. Most decision making models are usually based upon classification methods. These techniques, also called classifiers, enable the categorization of data (or entities) into pre-defined classes. The use of classification algorithms involves a training set consisting of pre-classified examples. There are many algorithms that can be used for classification, such as decision trees, neural networks, logistic regression, etc.

Data classification is a two-step process:

i) Learning
ii) Classification.

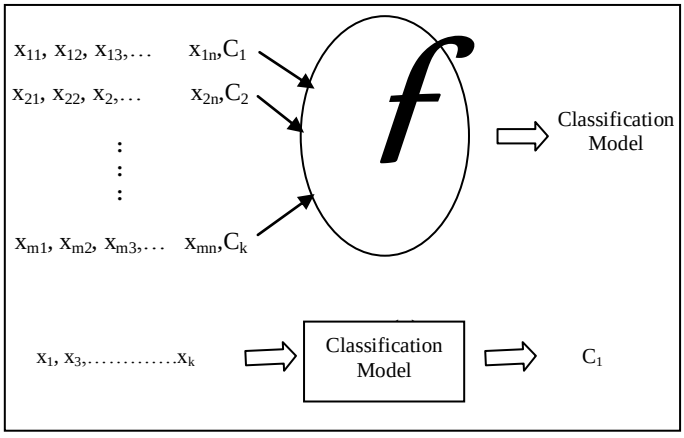

Figure 2. Classification process: (a) Learning (b) Classification Phase [3].

In the first step, a classifier is built describing a predetermined set of data classes or concepts. This is the learning step (or training phase), where a classification algorithm builds the classifier by analyzing or "learning from" a training set made up of database tuples and their associated class labels.

A tuple, X, is represented by an n-dimensional attribute vector, $X = (x_1, x_2,\ldots, x_n)$, depicting n measurements made on the tuple from n database attributes, respectively, A_1, A_2, …. , A_n. Each tuple, X, is assumed to belong to a predefined class as determined by another database attribute called the class label attribute. The class label attribute is

discrete-valued and unordered. It is categorical in that each value serves as a category or class. The individual tuples making up the training set are referred to as training tuples and are selected from the database under analysis. Because the class label of each training tuple is provided, this step is also known as supervised learning (i.e., learning of the classifier is "supervised" in that it is told to which class each training tuple belongs).

In the second step, the model is used for classification. First, the predictive accuracy of the classifier is estimated. If we were to use the training set to measure the accuracy of the classifier, this estimate would likely be optimistic, because the classifier tends to over fit the data (i.e., during learning it may incorporate some particular anomalies of the training data that are not present in the general data set overall). Therefore, a test set is used, made up of test tuples and their associated class labels. These tuples are randomly selected from the general data set. They are independent of the training tuples, meaning that they are not used to construct the classifier.

Data mining tool learns from examples or the data (data warehouses, databases etc) how to partition or classify certain object/data. As a result, data mining software formulates classification rules. The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier.

III. DECISION TREE ALGORITHMS

Decision Tree [5] is a flowchart like tree structure. The decision tree consists of three elements, root node, internal node and a leaf node. Top most element is the root node. Leaf node is the terminal element of the structure and the nodes in between is called the internal node. Each internal node denotes test on an attribute, each branch represents an outcome of the test, and each leaf node holds a class label. Various decision tree algorithms are:

A. ADTree Classification Algorithms

An alternating decision tree (ADTree) [6] is a machine learning method for classification. It generalizes decision trees. An alternating decision tree consists of decision nodes and prediction nodes. Decision nodes specify a predicate condition and prediction nodes contain a single number. ADTrees always have prediction nodes as both root and leaves. An instance is classified by an ADTree by following all paths for which all decision nodes are true and summing any prediction nodes that are traversed. It is different from other algorithms in the sense that in this algorithm, an instance follows only one path through the tree.

B. BFTree Classification Algorithm

BF (Best First) tree algorithms [7] are binary trees in which the "best" node is expanded at each point. The "best" node is the node whose split leads to maximum reduction of impurity (e.g. Gini index or information gain) among all nodes available for splitting. The resulting tree will be the same when fully grown; just the order in which it is built is different. The tree growing method attempts to maximize within-node homogeneity. The extent to which a node does not represent a homogenous subset of cases is an indication of impurity. A terminal node in which all cases have the same value for the dependent variable is a homogenous node that requires no further splitting because it is "pure".

C. Decision Stump Classification Algorithm

A decision stump [8] is a machine learning model consisting of a one-level decision tree i.e., it is a decision tree with one internal node (the root) which is immediately connected to the terminal nodes (its leaves). A decision stump makes a prediction based on the value of just a single input feature therefore they are also called 1-rules.

Depending on the type of the input feature, several variations are possible. For nominal features, one may build a stump which contains a leaf for each possible feature value or a stump with the two leaves, one of which corresponds to some chosen category, and the other leaf to all the other categories. For binary features these two schemes are identical. A missing value may be treated as a yet another category. For continuous features, usually, some threshold feature value is selected, and the stump contains two leaves — for values below and above the threshold. However, rarely, multiple thresholds may be chosen and the stump therefore contains three or more leaves.

D. FT Classification Algorithm

FT (Functional Tree) [9] combines a standard univariate decision tree, such as C4.5, with linear functions of the attributes by means of linear regressions. While a univariate DT uses simple value tests on single attributes in a node, FT can use linear combinations of different attributes in a node or in a leaf. In the constructive phase a function is built and mapped to new attributes. A model is built using the constructor function. The constructor function should be a classifier or a regression function depending on the type of the problem. In the former the number of new attributes is equal to the number of classes, in the latter the constructor function is mapped to one new attribute. Each new attribute is computed as the value predicted by the constructed function for each example. In the classification setting, each new attribute value is the probability that the example belongs to one class given by the constructed model. The merit of each new attribute is evaluated using the merit-function of the univariate tree, and in competition with the original attributes.

E. ID3 Classification Algorithm

ID3 (Iterative Dichotomiser 3) [10][11] is the precursor to the C4.5 algorithm, and is typically used in the machine learning and natural language processing domains. It generates decision tree from the dataset. The ID3 algorithm begins with the original dataset as the root node. On each iteration of the algorithm, it iterates through every unused attribute of the dataset and calculates the entropy (or information gain) of that attribute. Attribute which has the smallest entropy (or largest information gain) value is selected and the dataset is then split by the selected attribute (e.g. roll_no. < 50, 50 <= roll_no. < 100, roll_no. >= 100) to produce subsets of the data. Iterative recursion continues on each subset, considering only attributes never selected before until one of the following conditions occur:

- every element in the subset belongs to the same class, then the node is turned into a leaf and labeled with the class of the examples
- there are no more attributes to be selected, but the examples still do not belong to the same class, then the node is turned into a leaf and labeled with the most common class of the examples in the subset
- there are no examples in the subset, this happens when no example in the parent set was found to be matching a specific value of the selected attribute, for example

if there was no example with roll_no. >= 100. Then a leaf is created, and labeled with the most common class of the examples in the parent set.

Throughout the algorithm, the decision tree is constructed with each non-terminal node representing the selected attribute on which the data was split, and terminal nodes representing the class label of the final subset of this branch.

F. J48 Classification Algorithm

A J48 decision tree [12] is a predictive machine-learning model that decides the target value (dependent variable) of a new sample based on various attribute values of the available data. The internal nodes of a decision tree denote the different attributes; the branches between the nodes tell us the possible values that these attributes can have in the observed samples, while the terminal nodes tell us the final value (classification) of the dependent variable. The attribute that is to be predicted is known as the dependent variable, since its value depends upon, or is decided by, the values of all the other attributes. The other attributes, which help in predicting the value of the dependent variable, are known as the independent variables in the dataset. It examines information gain for choosing an attribute to split the data. To make the decision, the attribute with highest normalized information gain i.e. the attribute that discriminates the various instances most clearly is used.

G. LAD Tree Classification Algorithm

Logical Analysis of Data (LAD) tree [13] builds a classifier for binary target variable based on learning a logical expression that can distinguish between positive and negative samples in a data set. The basic assumption of LAD model is that a binary point covered by some positive patterns, but not covered by any negative pattern is positive, and similarly, a binary point covered by some negative patterns, but not covered by positive pattern is negative. The construction of LAD model for a given data set typically involves the generation of large set patterns and the selection of a subset of them that satisfies the above assumption such that each pattern in the model satisfies certain requirements in terms of prevalence and homogeneity.

H. LMT Classification Algorithm

A logistic model tree (LMT) [14][15] is a classification model with an associated supervised training algorithm that combines logistic regression (LR) and decision tree learning. Logistic model trees use a decision tree that has linear regression models at its leaves to provide a piecewise linear regression model (where ordinary decision trees with constants at their leaves would produce a piecewise constant model). In the logistic variant, the LogitBoost algorithm is used to produce an LR model at every node in the tree; the node is then split using the C4.5 criterion. Each LogitBoost invocation is warm-started from its results in the parent node. Finally, the tree is pruned. The basic LMT induction algorithm uses cross-validation to find a number of LogitBoost iterations that does not overfit the training data. A faster version has been proposed that uses the Akaike information criterion to control LogitBoost stopping.

I. Random Tree Classification Algorithm

A random tree [16] is a collection (ensemble) of tree predictors that is called *forest*. It can deal with both classification and regression problems. The classification works as follows: the random trees classifier takes the input feature vector, classifies it with every tree in the forest, and outputs the class label that received the majority of "votes". In case of a regression, the classifier response is the average of the responses over all the trees in the forest.

All the trees are trained with the same parameters but on different training sets. These sets are generated from the original training set using the bootstrap procedure: for each training set, you randomly select the same number of vectors as in the original set (=N). The vectors are chosen with replacement i.e., some vectors will occur more than once and some will be absent. At each node of each trained tree, not all the variables are used to find the best split, but a random subset of them. With each node a new subset is generated. However, its size is fixed for all the nodes and all the trees. None of the built trees are pruned.

In random trees there is no need for any accuracy estimation procedures, such as cross-validation or bootstrap, or a separate test set to get an estimate of the training error. The error is estimated internally during the training. Compute the classification error estimate as a ratio of the number of misclassified vectors to all the vectors in the original data.

J. Random Forest Classification Algorithm

Random forests [17][18] are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random forests correct for decision trees' habit of over fitting to their training set.

Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance of the final model.

K. Simple CART Classification Algorithm

Simple Cart (Classification and regression tree) [19] is a classification technique that generates the binary decision tree. Since output is binary tree, it generates only two children. Entropy is used to choose the best splitting attribute. Simple Cart handles the missing data by ignoring that record. This algorithm is best for the training data.

L. User Classifier Classification Algorithm

During the training step, User Classifier [20] at first mine tree patterns from the training dataset, then they build structural classification rule set. After constructing rules, they are sorted and undesirable rules are eliminated. The training phase of User-Classifier is done. In the testing phase, for each data point, User- Classifier choose a subset of rule set, called covering rule set, which are most related rules to the data point. Then based on the amount of similarity between these rules and the data point as well as the confidence degree of the selected rules, label of the data point is predicted.

IV. WEKA

WEKA (Waikato Environment for Knowledge Analysis) [21][22] is an open source, platform independent and easy to use data mining tool issued under GNU General Public License. It comes with Graphical User Interface (GUI) and contains collection of data preprocessing and modeling techniques. Tools for data pre-processing, classification,

regression, clustering, association rules and visualization as well as suited for new machine learning schemes are provided in the package. The algorithms can either be applied directly to a dataset or called from your own Java code. It also provides flexible facilities for scripting experiments. It is portable since it is fully implemented in the Java programming language and thus runs on almost any modern computing platform.

Weka provides access to SQL databases using Java Database Connectivity and can process the result returned by a database query. It is not capable of multi-relational data mining, but there is separate software for converting a collection of linked database tables into a single table that is suitable for processing using Weka.

User interfaces

Weka's main user interface is the Explorer, but essentially the same functionality can be accessed through the component-based Knowledge Flow as well as the command line interface (CLI). There is also the Experimenter, which allows the systematic comparison of the predictive performance of Weka's machine learning algorithms on a collection of datasets. The

- The *Cluster* panel gives access to the clustering techniques in Weka, e.g., the simple k-means, cobweb, DBSCAN, CLOPE algorithm to provide different kind of clustering's for different situations and usage of their results.
- The *Select attributes* panel provides algorithms for identifying the most predictive attributes in a dataset.
- The *Visualize* panel shows a scatter plot matrix, where individual scatter plots can be selected and enlarged, and analyzed further using various selection operators.

Extension packages

In version 3.7.2 of weka, a package manager was added to allow the easier installation of extension packages. Much functionality has come in weka through continuous extension and updates to make it more sophisticated.

V. METHODOLOGY AND PERFORMANCE MEASURES

Decision tree algorithms discussed in section 3 have been compared with the help of WEKA. Steps followed in the analysis are:

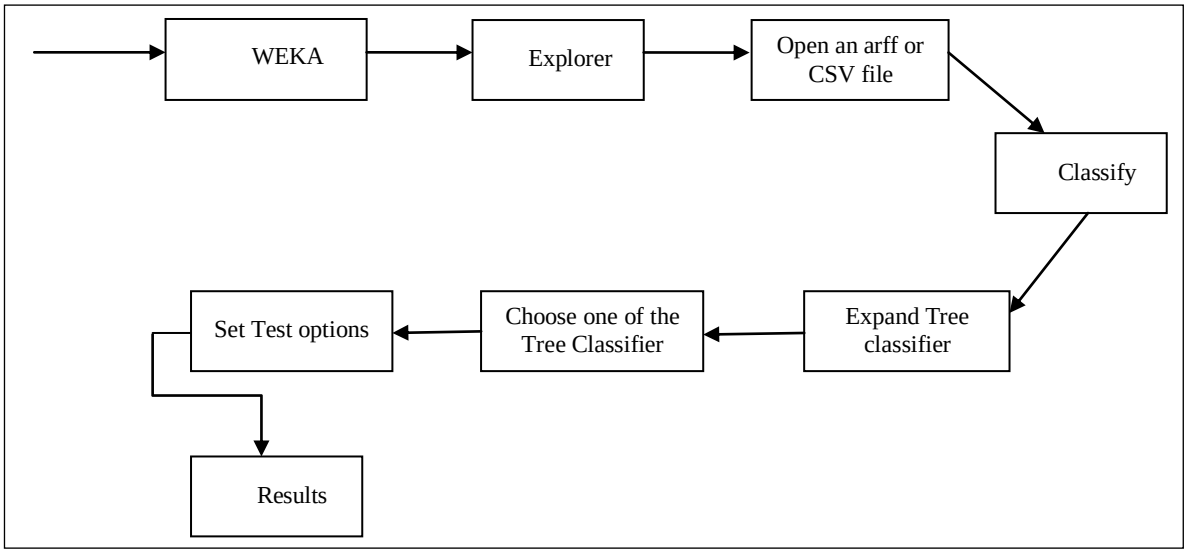

Figure 2. Decision Tree Classification process using WEKA.

Explorer interface features several panels providing access to the main components of the workbench:

- The *Preprocess* panel has facilities for importing data from a database, a csv or an arff file, etc., and for preprocessing this data using a so-called filtering algorithm. These filters can be used to transform the data from numeric to discrete, to remove missing instances, to appropriately choose missing values and converting csv file to arff and vice versa.
- The Classify panel enables the user to apply classification and regression algorithms to the resulting dataset, to estimate the accuracy of the resulting predictive model, and to visualize errors. There are various type of classification algorithms like rule based, decision tree, naïve Bayesian, lazy, mi, misc etc. This paper makes use of decision tree classification algorithms.
- The Associate panel attempts to identify all important interrelationships between attributes in the data with the help of association learners like apriori, filtered associator, predictive apriori etc.

As shown in Fig2, data file supplied should be in arff or CSV form. Data File should not contain unique id attribute like names, roll nos., remove these attribute either before supplying it for classification or untick these attribute before classification in WEKA. Various performance measures have been used to classify the data. Before knowing these measures, a little about some important terms which are used in these measures is presented. These are:-

- True Positive (TP) – total number of positive classes that were correctly predicted.
- True Negative (TN) – total number of negative classes that were correctly predicted.
- True Negative (FP) – total number of negative classes that were wrongly predicted as positive.
- True Negative (FN) – total number of positive classes that were wrongly predicted as negative.
- True Positive Rate (TPR) – positives correctly classified/total positives.
- False Positive Rate (FPR) – negatives in correctly classified/total negatives.
- N – Total number of classified instances.

Accuracy: It determines the proportion of the total number of instance predictions which are correctly predicted.

$$Accuracy = \frac{TP + TN}{N}$$

Precision: It is a measure of exactness. It is the ration of the predicted positive cases that were correct to the total number of predicted positive cases.

$$Precision = \frac{TP}{TP + FP}$$

Recall: Recall is a measure of completeness. It is the proportion of positive cases that were correctly identified to the total number of positive cases. It is also known as sensitivity or true positive rate (TPR).

$$Recall = \frac{TP}{TP + FN}$$

F-Measure: It is the harmonic mean of precision and recall. It is an important measure as it gives equal importance to precision and recall.

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Receiver Operating Characteristic (ROC) Curve: It is a graphical approach for displaying the tradeoff between true positive rate (TPR) and false positive rate (FPR) of a classifier. TPR is plotted along the y axis and FPR is plotted along the x axis. Performance of each classifier represented as a point on the ROC curve.

VI. Experimental Setup and Results

A comparative analysis of various decision tree classification algorithms has been made using five datasets taken from the KEEL [23] (a software tool to assess evolutionary algorithms in data mining problems) and UCI [24] machine learning repository. All the datasets are summarized in Table I.

TABLE I. DATASETS USED IN EXPERIMENTS.

Datasets	#Instances	#Attributes	#Classes	Test Method
Mushroom	5644	23	2	10-CV
Car	1728	7	4	10-V
Iris	150	5	3	2-CV
BreastCancer	277	10	2	2-CV
Nursery	12960	9	5	10-CV

We have used twofold cross validation (2-CV) for the datasets that contains less than 700 hundred instances and a tenfold cross validation (10-CV) for the datasets that contain more than 700 hundred instances. Various performance measures for all the datasets mentioned in Table I are shown in the Table II, III, IV, V and VI.

TABLE II. COMPARISON OF VARIOUS TREE CLASSIFICATION ALGORITHMS FOR MUSHROOM DATASET.

Decision Tree	Accuracy (%) ± Mean absolute Error	Precision	Recall	F-Measure	ROC Curve Area	Time Taken (in secs.)
ADTree	100±0.01	1	1	1	1	0.34
BFTree	100±0.0	1	1	1	1	4.4
Decision Stump	89.87±0.17	0.913	0.899	0.895	0.859	0.05
FT	100±0.01	1	1	1	1	8.36
ID3	100±0.0	1	1	1	1	0.13
J48	100±0.0	1	1	1	1	0.08
LAD	100±0.01	1	1	1	1	12.45
LMT	100±0.03	1	1	1	1	152.21
Random Tree	100±0.00	1	1	1	1	0.05
Random Forest	100±0.0	1	1	1	1	0.3
Simple Cart	100±0.0	1	1	1	1	3.56
User Classifier	61.80±0.47	0.382	0.618	0.472	0.499	21.36

TABLE III. COMPARISON OF VARIOUS TREE CLASSIFICATION ALGORITHMS FOR CAR DATASET.

Decision Tree	Accuracy (%) ± Mean absolute Error	Precision	Recall	F-Measure	ROC Curve Area	Time Taken (in secs.)
ADTree	93.05±0.1	0.93	0.931	0.93	0.99	0.2
BFTree	97.05±0.02	0.971	0.97	0.97	0.994	1.02
Decision Stump	70.02±0.2	0.49	0.7	0.577	0.709	0.01
FT	95.08±0.03	0.951	0.951	0.951	0.962	1.96
ID3	89.35±0.02	0.964	0.962	0.963	0.946	0.06
J48	92.30±0.04	0.924	0.924	0.924	0.976	0.06
LAD	90.68±0.1	0.916	0.907	0.911	0.981	0.59
LMT	98.79±0.01	0.988	0.988	0.988	0.998	57.56

Random Tree	85.88±0.07	0.854	0.859	0.856	0.935	0.04
Random Forest	93.06±0.08	0.93	0.931	0.93	0.99	0.14
Simple Cart	97.11±0.02	0.972	0.971	0.971	0.993	0.85
User Classifier	70.02±0.23	0.49	0.7	0.577	0.497	6.8

TABLE IV. COMPARISON OF VARIOUS TREE CLASSIFICATION ALGORITHMS FOR IRIS DATASET.

Decision Tree	Accuracy (%) ± Mean absolute Error	Precision	Recall	F-Measure	ROC Curve Area	Time Taken (in secs.)
ADTree	94.00±0.05	0.94	0.94	0.94	0.985	0.1
BFTree	93.33±0.04	0.933	0.933	0.933	0.986	0.12
Decision Stump	66.67±0.22	0.5	0.667	0.556	0.833	0.0001
FT	95.33±0.04	0.954	0.953	0.953	0.985	0.31
ID3	94.67±0.04	0.948	0.947	0.947	0.977	0.01
J48	94.67±0.05	0.947	0.947	0.947	0.976	0.01
LAD	94.67±0.04	0.947	0.947	0.947	0.985	0.04
LMT	94.67±0.08	0.947	0.947	0.947	0.982	0.46
Random Tree	94.67±0.04	0.948	0.947	0.947	0.977	0.0001
Random Forest	94.00±0.04	0.94	0.94	0.94	0.985	0.08
Simple Cart	94.67±0.05	0.947	0.947	0.947	0.976	0.13
User Classifier	33.33±0.44	0.111	0.333	0.167	0.5	3.84

TABLE V. COMPARISON OF VARIOUS TREE CLASSIFICATION ALGORITHMS FOR BREAST CANCER DATASET.

Decision Tree	Accuracy (%) ± Mean absolute Error	Precision	Recall	F-Measure	ROC Curve Area	Time Taken (in secs.)
ADTree	73.29±0.38	0.708	0.733	0.688	0.647	0.04
BFTree	69.68±0.36	0.665	0.697	0.671	0.591	0.3
Decision Stump	71.48±0.37	0.708	0.715	0.711	0.636	0.02
FT	66.07±0.37	0.648	0.661	0.654	0.599	0.46
ID3	60.29±0.33	0.661	0.673	0.666	0.575	0.03
J48	70.76±0.38	0.666	0.708	0.66	0.547	0.02
LAD	71.84±0.36	0.689	0.718	0.688	0.621	0.22
LMT	73.65±0.35	0.713	0.736	0.699	0.695	1.65
Random Tree	61.37±0.40	0.598	0.614	0.605	0.532	0.04
Random Forest	69.68±0.36	0.671	0.697	0.677	0.638	0.08
Simple Cart	72.56±0.36	0.696	0.726	0.683	0.624	0.26
User Classifier	70.76±0.41	0.501	0.708	0.586	0.497	4.55

TABLE VI. COMPARISON OF VARIOUS TREE CLASSIFICATION ALGORITHMS FOR NURSERY DATASET.

Decision Tree	Accuracy (%) ± Mean absolute Error	Precision	Recall	F-Measure	ROC Curve Area	Time Taken (in secs.)
ADTree	96.09±0.05	0.98	0.981	0.981	0.999	0.5
BFTree	99.49±0.01	0.995	0.995	0.995	0.999	12.19
Decision Stump	66.25±0.14	0.496	0.663	0.551	0.828	0.07
FT	96.25±0.02	0.962	0.963	0.962	0.984	25.7
ID3	98.19±0.002	0.995	0.996	0.995	0.991	0.19
J48	97.05±0.02	0.97	0.971	0.97	0.995	0.15
LAD	91.75±0.065	0.896	0.919	0.907	0.98	13.3
LMT	98.99±0.01	0.99	0.99	0.99	0.999	453.57
Random Tree	94.62±0.02	0.947	0.946	0.947	0.976	0.05
Random Forest	98.09±0.04	0.98	0.981	0.981	0.999	0.49
Simple Cart	99.58±0.02	0.996	0.996	0.996	0.999	9.76
User Classifier	33.33±0.27	0.111	0.333	0.167	0.5	2.46

In the analysis, six different measures have been used for comparing various classification decision tree algorithms. All the measures play a measure role in making any classification decision. From the results obtained in the Tables I, II, III, IV and V, it can be seen that Decision stump classification algorithm takes minimum time to classify data but gives less accuracy i.e. accurately classified instances are comparatively smaller in number. Id3 as well as J48 have quite good accuracy with a little increase in time used for classification. LMT gives maximum accuracy, but time taken to build classification model is much higher than other classifiers or we can say maximum in all the classifiers in most of cases. User classifier gives minimum accuracy and also has minimum precision, recall, F-measure and ROC curve area value among all the classifiers in most of the cases and also time taken for classification is more as compared to other classifiers. Rest of the models also lies in between the best and worst ones.

VII. CONCLUSIONS

Decision tree algorithms is one of the most widely used techniques in classification. Comparative analysis of various tree classification algorithms has been made. The results have been validated using 5 datasets taken from UCI and KEEL repository and noticed that successfully discovers valid rule sets with a good accuracy. Few of the classifiers have better accuracy, others take less time, and many others have a trade-off between accuracy and time taken. Most appropriate classifier can be used according to their usage.

REFERENCES

[1] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2001.

[2] P. Sharma and S. Ratnoo, "A review on Discovery of Classification Rules using Pittsburgh Approach," *Int. J. Artif. Intell. Knowl. Discov.*, vol. 4, no. 3, pp. 6–16, 2014.

[3] P. Sharma and S. Ratnoo, "Bottom-up Pittsburgh approach for discovery of classification rules," in *International Conference on Contemporary Computing and Informatics (IC3I)*, IEEE, 2014, pp. 31–37.

[4] P. Sharma, "Discovery of Classification Rules using Genetic Algorithm with non-random Population initialization," *Int. J. Artif. Intell. Knowl. Discov.*, vol. 4, no. 3, pp. 24–29, 2014.

[5] "Decision tree learning," *Wikipedia, the free encyclopedia*. 08-Feb-2015.

[6] Y. Freund and L. Mason, "The alternating decision tree learning algorithm," in *icml*, 1999, vol. 99, pp. 124–133.

[7] H. Shi, "Best-first decision tree learning," Citeseer, 2007.

[8] Wikipedia contributors, "Decision_stump," *Wikipedia, The Free Encyclopedia*. Wikimedia Foundation, 23-Aug-2012.

[9] J. Gama, "Functional trees for classification," in *International Conference on Data Mining*, IEEE, 2001, pp. 147–154.

[10] C. Jin, De-lin Luo, and M. Fen-xiang, "An improved ID3 decision tree algorithm," in *4th International Conference on Computer Science & Education*, 2009, pp. 127–130.

[11] Wikipedia contributors, "ID3_algorithm," *Wikipedia, The Free Encyclopedia*. Wikimedia Foundation, 20-Dec-2014.

[12] Wikipedia contributors, "C4.5_algorithm," *Wikipedia, The Free Encyclopedia*. Wikimedia Foundation, 28-Jan-2015.

[13] E. Boros, P. L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, and I. Muchnik, "An implementation of logical analysis of data," *IEEE Trans. On Knowl. Data Eng*, vol. 12, no. 2, pp. 292–306, 2000.

[14] N. Landwehr, M. Hall, and E. Frank, "Logistic model trees," *Mach. Learn.*, vol. 59, no. 1–2, pp. 161–205, 2005.

[15] Wikipedia contributors, "Logistic_model_tree," *Wikipedia, The Free Encyclopedia*. Wikimedia Foundation, 03-Apr-2014.

[16] Wikipedia contributors, "Random_tree," *Wikipedia, The Free Encyclopedia*. Wikimedia Foundation, 13-Jul-2014.

[17] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[18] "Random forest," *Wikipedia, the free encyclopedia*. 05-Feb-2015.

[19] D. Steinberg and P. Colla, "CART: classification and regression trees," *Top Ten Algorithms Data Min.*, vol. 9, p. 179, 2009.

[20] M. Ware, E. Frank, G. Holmes, M. Hall, and I. H. Witten, "Interactive machine learning: letting users build classifiers," *Int. J. Hum.-Comput. Stud.*, vol. 55, no. 3, pp. 281–292, 2001.

[21] E. Frank, M. Hall, G. Holmes, R. Kirkby, B. Pfahringer, I. H. Witten, and L. Trigg, "Weka," in *Data Mining and Knowledge Discovery Handbook*, Springer, 2005, pp. 1305–1314.

[22] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009.

[23] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework.," *J. Mult.-Valued Log. Soft Comput.*, vol. 17, 2011.

[24] A. Asuncion and D. Newman, *UCI machine learning repository*. Irvine, 2007.