Performance Comparison between Different Decision Tree based Classification Algorithms and a Multi-layer Neural Network

Bhavya Vinnakota
20815072
M.Eng. ECE Department
University of Waterloo
bvinnako@uwaterloo.ca

Mayank Chandnani
20860563
M.Eng. ECE Department
University of Waterloo
m2chandn@uwaterloo.ca

Meet Mahajan
20868353
M.Eng. ECE Department
University of Waterloo
mnmahaja@uwaterloo.ca

***Abstract*—Data mining is one of the most sought out tool in this modern data driven world. Various algorithms and methods have been developed to achieve data mining. Decision tree based algorithms are one of the oldest algorithms used for data mining. Various algorithms like C4.5, CHAID, CART, ID3, etc. have been developed based on the decision tree logic over the years. Here, we implement three of the most famous decision tree algorithms, which are CART, C4.5 and ID3, and we compare them with a neural network based algorithm. We compare the time taken to train the models and the final prediction accuracy of the decision tree algorithms with a MLP neural network. The dataset we are using for our study is the Car Evaluation Dataset acquired from University of California, Irvine.**

***Index Terms*—MLP-NN, CART, C4.5, ID3, Decision Tree, Neural Network, Chefboost**

I. Introduction

Data Mining involves the extraction of certain, previously undiscovered and rationally important data from large blocks of information to gather significant patterns and trends. This gathered information is termed as knowledge. Data mining is always included in procedures for obtaining and defining structural patterns in data as a means for reviving that data and making predictions [1].

Data mining can mainly be segregated into five steps. First, extract factual data and load the transaction data. Second, collect and manage the data in a multidimensional database system. Third, Business analysts, administration teams and information technology specialists access this data and ascertain how to organize it. Fifth, present the data in a readable format, such as a graph or table [1]. Techniques for interpreting and modeling data can be broadly classified into three categories: "Supervised learning", "Unsupervised learning" and "Reinforced learning".

Supervised learning requires input data that has both independent (predictor) variables and a dependent (target) variable whose value is to be evaluated. By numerous means, the method "studies" how to model the value of the target variable depending on the predictor variables. Decision trees and neural networks are some of the algorithm patterns which come under the supervised learning technique. If the objective of learning is to identify the value of any variable in the given data, then supervised learning can be an ideal approach [2].

Unsupervised Learning is a training process, the target output is not given to the machine. Hence, the system learns of its own by recognizing and adjusting to structural attributes in the input [1].

Reinforced Learning is an output-based technique. In this, a mentor though available does not give the anticipated result but simply designates if the computed output is right or wrong. The information is given to assist the machine in the process [1].

Neural Networks were stated as generalizations of analytical models of human insight in biological neurons, it is considered as a data processing method that has specific performance features familiar with human neural biology. These features cover the capability of saving information and delivering it whenever necessary, the ability to distinguish patterns, despite the noise, tendency of recognizing preceding occurrences and create conclusions and perceptions about upcoming circumstances but Neural Networks are data-dependent and therefore, their performance improves with sample size. Whereas statistical methods, such as Regression work strongly for remarkably small sample data [3].

Our research objective was to train the dataset on the Neural network (MLP), C4.5, ID3 and CART algorithms to compare their resulting precision, accuracy and execution time for all the implementations. The rest of the research paper is ordered as: Section two presents the literature review, section three presents the implementation of the research model, section four is the results analysis and based on this analysis in Section five, conclusions were drawn.

II. Literature Review

A. Decision Tree

A decision tree has a flowchart-like tree structure, where every individual internal node signifies a test on an attribute, every branch signifies a consequence of this test, the class label is denoted by each leaf node (or terminal node). Given a tuple X, the attribute values of the tuple are examined upon

the decision tree. A path is determined from the root node to a leaf node which contains the class prediction for the tuple. Decision tree training employs a decision tree as a predictive model that outlines perceptions regarding a sample of data to conclude its target value [4].

Decision Trees are one of the predictive modelling approaches used in statistics, data mining and machine learning. These Tree models where the target variable can use a measurable set of data are termed as classification trees, in such structure, the leaves signify the class labels and their branches signify associations of features that drive to the class descriptions. A decision tree can be built comparatively fast compared to other algorithms in classification.

Decision tree classifiers achieve comparable or more dependable accuracy when compared with several other classification algorithms. Classification is one of the most frequently used techniques [4].

B. How to Build a Tree

Decision trees [5] can be built from the provided set of attributes. While few are higher in accuracy than others, determining the optimal tree is computationally speculative because of the exponential capacity of the research space. Nevertheless, several effective models have been generated to build a moderately correct, albeit sub-optimal, decision tree in a fair amount of time. The decision tree algorithms normally engage a greedy strategy that produces a decision tree by creating a group of regionally best decisions around which attribute use for partitioning the data.

C. Hunt's Algorithm

Hunt's algorithm [5] recursively grows a decision tree by partitioning the training data into successively more transparent subsets. Let D_t be the set of training data that reach a node t. The usual recursive method is defined as below:

- If D_t holds data that refer to the same class y_t, implies that t is a leaf node identified as y_t.
- If D_t is an empty set, then t is a leaf node identified by the default class, y_d
- If D_t holds records that relate to higher than one class, use an attribute test to divide the data into smaller subsets.

It recursively implements this method to every subset until all the data in the subset relates to one class. Hunt's algorithm considers that every combination of attribute sets has a different class label in the procedure. If all the records connected with D_t have the same attribute values besides the class label, then it is not feasible to break this data into any future subsets. In such a case, the node is indicated as a leaf node with the same class label as the majority class of training records connected with this node.

D. Determine the best attribute test Condition [5]

The decision tree [5] producing algorithm requires a method for defining the test requirements for various attribute types as well as an objective measure for assessing the virtue of every test condition.

First, the specification of an attribute test state and its consequences rely on the attribute types. We can take a two-way split or multi-way split, discretize or classify attribute values as required. The binary attributes drive to a two-way split test condition. For attributes that have multiple values, the test condition can be shown into a multi-way split on every distinguished value, or two-way split by asserting the attribute values into binary subsets.

Likewise, the ordinal attributes can also provide binary or multi-way splits as long as the grouping seems not to disrupt the symmetry characteristic of the attribute values. For continuous attributes, the test requirement can be expressed as a comparison test with two results or a range query. Or we can discretize the continuous value into a nominal attribute and then perform the two-way or multi-way split.

Considering there are many choices to define the test conditions from the given training set of data, we require to adopt measurements to determine the best way to split the records. The purpose of the most dependable test requirements is whether it results in a homogeneous class distribution in the nodes, which is the purity of the child nodes before and post splitting. The larger this degree of purity, the more reliable the class distribution.

To settle how strong a test condition presents, we need to analyze the degree of impurity of the parent before dividing with the degree of the impurity of the child nodes post splitting. The higher their difference, the better the test condition. The calculation of the degree of purity are:

- Gini Index
- Entropy
- Misclassification Error

E. ID3 Algorithm

Iterative Dichotomiser (ID) 3 is a decision tree training algorithm proposed by Quinlan Ross in 1986. It is serially executed and based on Hunt's algorithm. The essential purpose of the ID3 algorithm is to create the decision tree by engaging a top-down, greedy search within the provided sets to test every attribute at each tree node. In the decision tree method, the knowledge gain approach is commonly used to define a fitting property for every node of a produced decision tree [4]. Knowledge gain depends on the entropy of the dataset. Entropy of an attribute is a measurement of the uncertainty of that attribute. Entropy is a measure of the randomness in the information being processed. It basically measures the variance in the data. It provides the probability distribution of the target value in the data. For example, if for a particular attribute, the target value output is same for all instances of that attribute, the uncertainty or entropy of that attribute will be zero. If the instances of that attribute are divided evenly amongst all possible values of the target class, then the uncertainty in that attribute is the highest, i.e. 1.

$$Entropy(D) = -\sum_{d=1}^{|k|} p_k \log_2 p_k \tag{1}$$

here, *D* is the data set, *k* is the number of classes p_k is the probability of an attribute belonging to a particular class [7].

Once we have the entropy, we calculate the information gain in the dataset. Information gain is the decrease in uncertainty of the dataset. It is the metric of how much data is carried by an attribute.

$$Gain(D, a) = Entropy(D) - \sum_{v=1}^{V} \frac{|D^V|}{|D|} Entropy(D^V) \quad (2)$$

here, *D* represents the dataset, *a* is the attributes in a given set |D| and *V* is the number of values of an attribute [7].

Basically, we are subtracting the individual entropy of all the attributes of a particular feature from the entropy of the whole system. This way we come to know how much information gain each feature provides to the dataset. The feature that can provide us with the highest amount of information is chosen.

Consequently, we can pick the attribute with the greatest learning gain (entropy reduction is maximum) as the test attribute of the current node. In this direction, the knowledge required to analyze the training sample subset acquired from the following on partitioning will be the minutest. Hence this property is used for partitioning the sample set of data included in the current node to make a hybrid degree of diverse types for all produced sample subsets diminished to a minimum. Hence, the use of an information theory approach will efficiently decrease the required dividing amount of object classification [4].

F. C4.5 Algorithm

C4.5 [4] is a method used to produce a decision tree introduced by Ross Quinlan. C4.5 is an extension of Quinlan's earlier ID3 algorithm. The decision trees presented by C4.5 is usually referred to as a statistical classifier. It can take data with categorical or numerical values. The threshold is produced to manage continuous values and further divide attributes with values preceding the threshold and values equivalent to or beneath the threshold. C4.5 algorithm can readily manage missing values, as missing attribute values are not used in gain calculations by C4.5.

Gain ratio is used instead of only gain. Information gain tends to favor the attributes with higher number of values, instead gain ratio penalizes the attribute with higher number of values so that the bias in information gain can be reduced. It takes the number and size of branches into account.

$$Gainratio(D, T) = \frac{Gain(D, T)}{SplitInfo(D, T)} \quad (3)$$

$$SplitInfo(D, T) = -\sum_{i}^{n} \frac{|T_i|}{|T|} \log_2 \frac{|T_i|}{|T|} \quad (4)$$

here, gain is calculated the same way as in (2). T_i is the number of data points in the current subset and T is the total number of data points in the parent subset [8].

G. CART Algorithm[4]

CART stands for Classification And Regression Trees, proposed by Breiman in 1984. It produces both classifications and regression trees. The classification tree built by CART is based on binary splitting of the attributes. CART is also based on Hunt's algorithm and can be executed serially. Gini index is practiced as a splitting criterion in deciding the splitting attribute.

Gini index measures the probability that a chosen variable is classified incorrectly. Its value varies between 0 and 1. If all the variables belong to the same class and there is only one output class, Gini index will be 0 in that case. A Gini index of 1 means that the elements are distributed across various classes and it represents perfect inequality [9].

$$Gini = 1 - \sum_{i=1}^{n} p_i^2 \quad (5)$$

where, p_i is the probability of an object being classified to a particular class [9].

Although CART is distinctive from other Hunt's based algorithm because it is also used for regression analysis. The regression analysis feature is applied in intercepting a dependent variable provided a set of predictor variables over a period. CARTS encourages both continuous and nominal attribute data [4].

C4.5's tree-construction algorithm differs in several respects from CART, for instance [4],

- Tests in CART are constantly binary, but C4.5 enables two or more consequences.
- CART practices the Gini index to rank tests, whereas C4.5 exercises information-based models.
- CART prunes trees by a cost-complexity model whose parameters are determined by cross-validation, In C4.5, uses a single-pass algorithm obtained from binomial confidence limits
- CART views for surrogate tests that approximate the consequences when the tested attribute has an unknown value, on the other hand, C4.5 apportions the case probabilistic among the outcomes.

H. Artificial Neural Network (ANN) classifier

A multi-layered feed-forward Artificial Neural Network [6] is the extensively utilized algorithm and its layout includes one input layer, a minimum of one hidden layer and one output layer. Neural Network is an assuring method for various circumstances like non-normality, complicated feature spaces and multivariate data types, where conventional methods lose to provide reliable outcomes.

The most distinguishing feature of a neural network that drives its selection in this research is its robustness. The system utilizes conventional back-propagation for supervised learning. The number of hidden layers to practice and the decision between a logistic or hyperbolic activation function can be obtained. Training happens by altering the weights in the node

to reduce the distinction within the output node activation and the output.

The error is back-propagated by the network and weight adjustment is done utilizing a recursive method. The multi-layer perceptron algorithm including an error minimization back-propagation training was implemented in this research which is based on numerous optimal sets of structures and learning parameters.

III. Dataset Description

We have performed our analysis on the Car Evaluation Dataset. This dataset was donated by Marko Bohanec and Blaz Zupan to the UCI Machine learning repository in 1997 [10]. The dataset consists of 1728 instances which help in evaluating the acceptability of a car.

The dataset has three major concepts: price, tech and comfort. Six different features have been derived from these three concepts. The features derived are buying price and maintenance price from price; comfort is a subset of technical characteristics, the features it produces are number of doors, person capacity and luggage capacity of the boot. Apart, from comfort, the other subset and the final feature of the dataset is the estimated safety of the car. Table I shows the column names that these features are being referred to as in the data.set

Feature	Column Name
Buying Price	buying
Maintenance Price	maint
Number of Doors	doors
Person Capacity	persons
Luggage Capacity	lug_boot
Estimated Safety	safety
Acceptability	Decision

TABLE I: Column Names

All these features are used to decide the acceptability of the car. The acceptability of the car is decided by four attributes: unacceptable, acceptable, good and very good. These attributes act as the target value for the neural network. And for the decision trees, these values attributes act as the leaf node of the tree.

Furthermore, the input features are also divided into several attributes. The buying price and maintenance price are divided into four attributes each, namely, very high, high, medium and low. The number of doors can be divided into cars having 2 doors, 3 doors, 4 doors, or cars with 5 or more doors. Similarly, the people carrying capacity of the car can be divided into cars that can carry 2 persons, cars that can carry 4 persons and cars that can carry more than 4 persons. The boot size of the car could be classified as small, medium or big. And finally, the safety rating of the car was rated as low, medium or high.

Attribute	Abbreviation	Frequency
Very High	vhigh	432
High	high	432
Medium	med	432
Low	low	432

TABLE II: Buying Price Attributes

Attribute	Abbreviation	Frequency
Very High	vhigh	432
High	high	432
Medium	med	432
Low	low	432

TABLE III: Maintenance Price Attributes

Attribute	Abbreviation	Frequency
2	2	432
3	3	432
4	4	432
>=5	5more	432

TABLE IV: Number of Doors Attributes

Attribute	Abbreviation	Frequency
2	2	576
4	4	576
>4	more	576

TABLE V: Person Capacity Attributes

Attribute	Abbreviation	Frequency
Small	small	576
Medium	med	576
Big	big	576

TABLE VI: Luggage Capacity Attributes

Attribute	Abbreviation	Frequency
Low	low	576
Medium	med	576
High	high	576

TABLE VII: Safety Rating Attributes

Attribute	Abbreviation	Frequency
Unacceptable	unacc	1210
Acceptable	acc	384
Good	good	65
Very Good	vgood	69

TABLE VIII: Acceptability Attributes

Table II to table VII list all the input variables along with their attributes and abbreviations that are used for those attributes. The tables also mention the frequency of occurrence

# neurons	Training Time(s)	Training Accuracy(%)	Testing Accuracy(%)
16	6.44	71.14	72.60
32	7.29	74	71.76
64	6.75	75.46	72.45
128	9.64	77.24	76.62
256	9.84	80.79	78.24
512	11.29	84.18	82.41
1024	12.66	87.42	89. 58

TABLE IX: Single Layer Perceptron

# neurons	Training Time(s)	Training Accuracy(%)	Testing Accuracy(%)
16	7.88	74.85	71.99
32	9.16	72.45	68.75
64	8.81	82.64	82.64
128	11.03	86.27	87.73
256	17.18	92.05	91.67
512	33.15	96.76	94.68
1024	80.92	99.23	97.69

TABLE X: Two Layer Perceptron

# epochs	Training Time(s)	Training Accuracy(%)	Testing Accuracy(%)
10	3.98	71.45	67.82
20	6.9	79.78	76.85
30	9.88	84.80	85.88
40	12.62	88.35	89.35
50	15.26	92.67	91.52
60	19.76	94.60	92.18
70	21.74	96.06	93.29
80	24.32	96.76	94.91

TABLE XI: Epoch Selection

of a particular attribute in the dataset. Table VIII lists all the output variables of the dataset along with the abbreviation for the attributes and the frequency of occurrence of that attribute in the dataset.

IV. Models and Frameworks

As established earlier, the decision about the acceptability of the car will be taken through implementing concepts of neural networks and decision trees. The neural network being used is multi layer perceptron neural network. An MLP is appropriate here as the classification problem is not image based and the number of input features is just 6, which is not too high. The decision tree algorithms being implemented are ID3, CART and C4.5

A. Neural Network Details

The neural network being implemented is a multi-layer perceptron. The MLP has one input layer with 6 features, it has an output layer with four attributes and two hidden layers with 512 neurons each. The number of layers and number of neurons were calculated by trial and error.

Table IX covers the different number of neurons tried for a single layer of perceptron. The number of epochs for this experiment were 50.

Similarly, Table X covers the different number of neurons tried for two layers of perceptron. The number of epochs for this experiment were 50.

From the data of Table IX and Table X, it was observed that a two layer perceptron with 256 neurons gives high accuracy in low time. It was observed that for this network had the accuracy same as the decision tree algorithms.

Using 256 neurons and two layers we next checked the suitable number of epochs for getting the accuracy, and for 50 epochs the accuracies were similar to that of decision trees and the time taken was also not very high. The same have been tabulated Table XI.

Table XII is the final configuration of the multi layer perceptron neural network.

Input Layer	6 neurons
Hidden Layer 1	256 neurons
Hidden Layer 2	256 neurons
Output Layer	4 neurons
Training Epochs	50

TABLE XII: MLP-NN Configuration

B. Decision Tree Implementation

The decision tree is being implemented through a decision tree framework developed by Sefik Ilkin Serengil, called Chefboost [11]. Chefboost is a lightweight gradient boosting implementation of various decision tree algorithms. Various decision tree algorithms like ID3, CHAID, C4.5, CART and Regression Trees can be implemented through Chefboost just by using a few lines of code. Furthermore, Chefboost framework also provides bagging models like random forest.

As mentioned in Section II-E, Section II-F and Section II-G of this paper, the metrics used for deciding the nodes for ID3, C4.5 and CART algorithm are Gain, Gain ratio and Gini index respectively. A snippet from the Chefboost framework implementing these metrics has been represented in Fig. 1 [15].

The framework requires the user to select a target column in the dataset. In our case, we selected the decision column in the dataset as the target value. The decision column contains the acceptability attributes of the dataset. After a target column has been set, the whole training dataset, comprising of the input and output features is passed to the framework along with the algorithm that is to be implemented through the framework.

V. Implementation

For implementing the algorithms, the first step was to split the data in a ratio of 75% training set and 25% test dataset. After this split was made, different preprocessing was carried out for neural network implementation and for decision tree based implementation.

A. Preprocessing Data for Neural Networks

As the data available in the dataset is in the form of numbers like 2 and 3 for number of doors, and also in form of *vhigh*, *high*, *more*, *5more* for other attributes, it was necessary to convert all the data in a numerically uniform form.

To do so we used the *LabelEncoder()* feature available in the *scikit* library. *LabelEncoder()* lists out all the unique attributes of a feature and assigns them a numerical value starting from zero. The values are assigned in an alphabetical order.

Attribute	Enumeration
high	0
low	1
med	2
vhigh	3

TABLE XIII: Buying Price Encoding

Attribute	Enumeration
high	0
low	1
med	2
vhigh	3

TABLE XIV: Maintenance Price Encoding

Attribute	Enumeration
2	0
3	1
4	2
5more	3

TABLE XV: Number of Doors Encoding

Attribute	Enumeration
2	0
4	1
more	2

TABLE XVI: Person Capacity Encoding

Attribute	Enumeration
big	0
med	1
small	2

TABLE XVII: Luggage Capacity Encoding

Attribute	Enumeration
high	0
low	1
med	2

TABLE XVIII: Safety Rating Encoding

Attribute	Enumeration
acc	0
good	1
unacc	2
vgood	3

TABLE XIX: Acceptability Encoding

After all the attributes have been encoded, the training and testing sets are further divided based on input variables (*X_train*, *X_test*) and output variables (*y_train*, *y_test*). To obtain *X_train* and *X_test* from the datasets, the *Decision* column is dropped from both the training and testing datasets. To obtain *y_train* and *y_test* from the datasets, the *Decision* columns from both the training and testing datasets are added to empty arrays.

After obtaining *y_train* and *y_test*, the four numerical outputs of *Decision* are converted into one-hot encoding form. This is done because in label encoding there is a chance that

```
if algorithm == "C4.5":
        winner_one = subset_gainratios.index(max(subset_gainratios))
elif algorithm == "ID3": #actually, ID3 does not support for continuou
        winner_one = subset_gains.index(max(subset_gains))
elif algorithm == "CART":
        winner_one = subset_ginis.index(min(subset_ginis))
```

Fig. 1. Chefboost Algorithm

the algorithm assign some sort of hierarchy to the labels due to different number [12] [13] [14]. Thus, one hot encoding is needed to make the output uniform. Table XX shows the one-hot encoded mapping of the output data.

Attribute	Label Encoding	One-hot Encoding
acc	0	1 0 0 0
good	1	0 1 0 0
unacc	2	0 0 1 0
vgood	3	0 0 0 1

TABLE XX: One-hot Encoding

B. Neural Network Implementation

After data has been preprocessed, a model is built as per the configuration mentioned in Table XII.

X_train and *y_train* is passed to the *fit()* feature from the *keras* library. Once the model has been trained, *X_test* and *y_test* are passed to the *evaluate()* feature from the *keras* library to obtain the test accuracy.

C. Preprocessing for Decision Tree Algorithms

As mentioned in Section IV-B, Chefboost algorithm doesn't require the data to be preprocessed. Chefboost only requires a list of column names corresponding to the input features and a list containing the name of the target column.

A *features* list was defined containing the column names of input features as shown in Eq. 6,

$$features = [\text{"}buying\text{"}, \text{"}maint\text{"}, \text{"}doors\text{"}, \text{"}persons\text{"}, \text{"}lug\_boot\text{"}, \text{"}safety\text{"}] \quad (6)$$

Similarly, a *target* list was created to assign the column containing the target values. The equation is covered in Eq. 7,

$$target = [\text{"}Decision\text{"}] \quad (7)$$

For the test set, we need to divide in into *X_test* and *y_test*. Similar to neural networks, to obtain *X_test* from the datasets, the *Decision* column is dropped from both the testing dataset. And to obtain *y_test* from the dataset, the *Decision* columns from both the testing dataset is added to an empty array.

D. Decision Tree Implementation

To train the algorithms, the complete training dataset was passed to the *fit()* function using an object *chef* from Chefboost framework. The name of the algorithm to be implemented is also passed as a parameter to the *fit()* function.

After, training the algorithm, the testing inputs *X_test* are passed to the *predict()* function using an object *chef* from Chefboost framework. This returns a map of outputs classified according to the trained model.

```
def predict_all(model, data):
 # Method to predict for all the records
 def predict(instance):
    return chef.predict(model, instance)
 return list(map(predict, data.values))
```

Listing 1. predict_all function

A function was defined to predict outputs of all the models. The list obtained from this function was then compared with the original *y_test* using the *precision_score()* function of *scikit* library. This gives us the test accuracy.

After defining the function *predict_all*, the training and testing for all algorithms was done using the following codes.

```
model=chef.fit(train,{'algorithm':'ID3'})
predict = predict_all(model, X_test)
score = precision_score(y_test, predict,
average='weighted')
```

Listing 2. ID3 algorithm

```
model=chef.fit(train,{'algorithm':'C4.5'})
predict = predict_all(model, X_test)
score = precision_score(y_test, predict,
average='weighted')
```

Listing 3. C4.5 algorithm

```
model=chef.fit(train,{'algorithm':'CART'})
predict = predict_all(model, X_test)
score = precision_score(y_test, predict,
average='weighted')
```

Listing 4. CART algorithm

In the above mentioned code fragments, *train* is the complete training dataset. And the algorithm which is to be implemented is mentioned along with *train* in the *fit()* function.

VI. Results and Discussion

After implementing the four algorithms on the dataset, the test accuracy and training time for the networks were noted. Table XXI showcases the results that were observed.

Algorithm	Training Time (s)	Test Accuracy (%)
CART	3.71	93.15
ID3	5.28	92.05
C4.5	5.11	92.57
MLP-NN	16.41	91.67

TABLE XXI: Experiment Observations

Figure 2 and 3 plot the values obtained from implementation for easier comparison. It is clear from the figure that the training time taken by the neural network is the highest. Also, it is clear that the test accuracy of neural network is the smallest among all the algorithms.

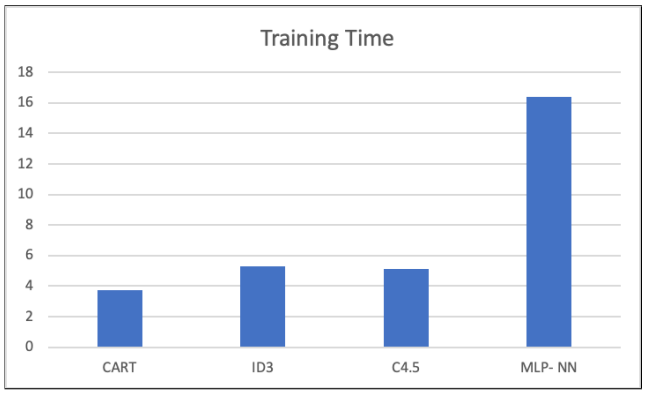

Fig. 2. Training Time

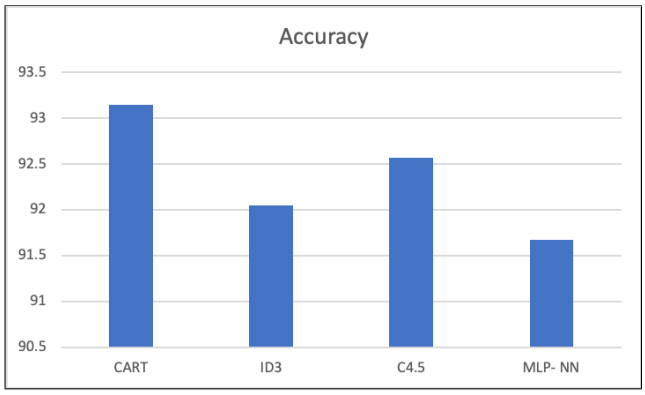

Fig. 3. Testing Accuracy

To compare the effect of train/test split on different algorithms, different train/test splits were implemented. Table XXII, Table XXIII, Table XXIV and Table XXV are tabulation of different training times and testing accuracies obtained by splitting training and testing sets in a ratio of 90/10, 80/20, 60/40 and 50/50 respectively.

Algorithm	Training Time (s)	Test Accuracy (%)
CART	4.72	83.93
ID3	5.9	84.54
C4.5	5.77	84.54
MLP-NN	18.51	89.02

TABLE XXII: 90/10 Split

Algorithm	Training Time (s)	Test Accuracy (%)
CART	3.79	94.09
ID3	5.35	94.95
C4.5	5.22	94.57
MLP-NN	16.72	91.91

TABLE XXIII: 80/20 Split

Algorithm	Training Time (s)	Test Accuracy (%)
CART	3.19	90.87
ID3	4.28	91.48
C4.5	4.28	92.25
MLP-NN	14.25	86.42

TABLE XXIV: 60/40 Split

Algorithm	Training Time (s)	Test Accuracy (%)
CART	2.89	90.95
ID3	4.12	91.44
C4.5	4.04	90.49
MLP-NN	10.84	86.46

TABLE XXV: 50/50 Split

Based on the values obtained from different splits, two graphs have been plotted to study the effect of train/test split on different algorithms.

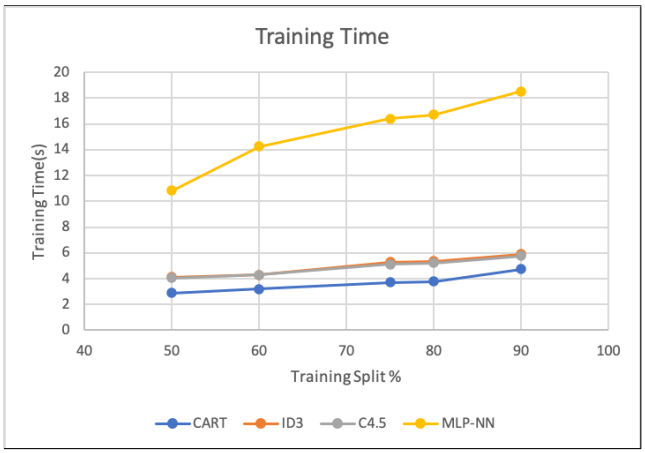

Fig. 4. Training Time

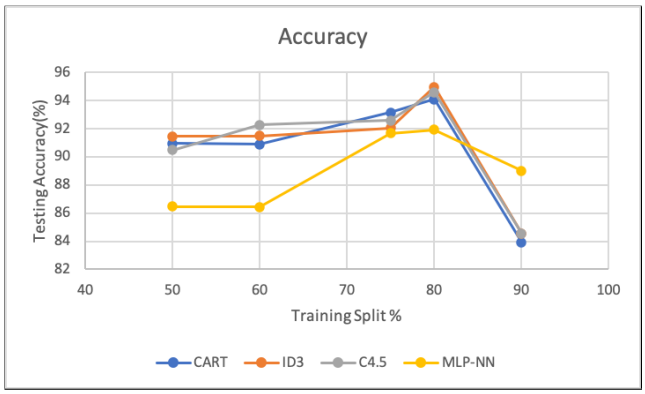

Fig. 5. Testing Accuracy

VII. Conclusion

From 4 in Section VI of the paper, it is observed that the time taken by CART algorithm for training is the least. It is also observed that the time taken by MLP-NN is the highest.

However, from Table IX and Table X it can be inferred that the training time of MLP-NN can be reduced if we decrease the number of neurons and number of layers. But it is achieved at the cost of accuracy.

Furthermore, From 5 in Section VI of the paper, it is observed that the accuracy of Decision Tree algorithms is better than the MLP-NN in most cases. When, the training set is much larger as compared to the testing set, the accuracy of the system drops. It was observed that the training accuracy at this time was 100% while the testing accuracy was in the range of 80% to 85%, this is an indication of possible over fitting in the models.

Again from Table IX and Table X it is clear that the testing accuracy of MLP-NN can be increased if we increase the number of neurons and number of layers. But as the number of layers and neurons increase, the training time also increases.

Also it is observed that the testing accuracy and training time are very similar for C4.5 and ID3. This is because C4.5 is developed on ID3 and was majorly developed to handle continuous data which wasn't possible in ID3. However, as our dataset doesn't have any continuous values, both the algorithms act very similarly.

In conclusion, for simple datasets like the one being explored in this paper, decision tree based algorithms are more accurate and much faster as compared to neural networks. However, for different datasets and different configurations of neural networks, higher accuracy can be achieved through neural network at the cost of training time.

References

[1] B.N. Patel, S.G. Prajapati and Dr.K.I. Lakhtaria, "Efficient Classification of Data Using Decision Tree," *Bonfring International Journal of Data Mining*, Vol. 2, No. 1, March 2012

[2] DTREG, predictive modeling company "Decision Trees Compared to Regression and Neural Networks" Aug.12, 2020.

[3] M.A. Razi, K. Athappilly, "A comparative predictive analysis of neural networks (NNs), nonlinear regression and classification and regression tree (CART) models," *Expert Systems with Applications 29 (2005) 65–74*

[4] H. Sharma, and S. Kumar, "A Survey on Decision Tree Algorithms of Classification in Data Mining," *International Journal of Science and Research (IJSR)*, ISSN (Online): 2319-7064 Index Copernicus Value (2013): 6.14 — Impact Factor (2015): 6.391

[5] Y.J. Bakos, "Decision Tree Classifier," Colorado School of Mines, Electrical Engineering & Computer Science, 2010, Fall, CSCI 568: Data Mining [Accessed: Aug. 12 2020].

[6] H. Z. Mohd Shafri , A. Suhaili and S.Mansor "The Performance of Maximum Likelihood, Spectral Angle Mapper, Neural Network and Decision Tree Classifiers in Hyperspectral Image Analysis," *Journal of Computer Science 3*, (6): 419-423, 2007 ISSN 1549-3636 © 2007 Science Publications

[7] Y. Wang, Y.Li, Y.Song, X.Rong, and S.Zhang, "Improvement of ID3 Algorithm Based on Simplified Information Entropy and Coordination Degree," Shandong University , Nov. 6, 2017

[8] V.B. Frank, "Classification Trees: C4.5," IRIDIA, Universit Libre de Bruxelles, July 7, 2003. [Online]. Available: http://www.applied-mathematics.net/classification/classifier.pdf.

[9] S. Tahsildar, "Gini Index For Decision Trees," April 18, 2019. [Online]. Available: https://blog.quantinsti.com/gini-index/

[10] M. Bohanec, and B. Zupan (1997). UCI Machine Learning Repository [https://archive.ics.uci.edu/ml/datasets/Car+Evaluation]. Irvine, CA: University of California, School of Information and Computer Science.

[11] Serengil, and S. Ilkin, "Lightweight Decision Trees Framework supporting Gradient Boosting (GBDT, GBRT, GBM), Random Forest and Adaboost w/categorical features support for Python," [https://github.com/serengil/chefboost], 2019.

[12] D. Yadav, "Categorical encoding using Label-Encoding and One-Hot-Encoder," Dec. 6, 2019. [Online]. Available: https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd

[13] S. Srinidhi, "Label Encoder vs. One Hot Encoder in Machine Learning," July 30, 2018. [Online]. Available: https://medium.com/@contactsunny/label-encoder-vs-one-hot-encoder-in-machine-learning-3fc273365621

[14] A. Sethi, "One-Hot Encoding vs. Label Encoding using Scikit-Learn," Mar. 6, 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/

[15] Serengil, and S. Ilkin, "Lightweight Decision Trees Framework supporting Gradient Boosting (GBDT, GBRT, GBM), Random Forest and Adaboost w/categorical features support for Python," [https://github.com/serengil/chefboost/blob/master/chefboost/training/Preprocess.py], 2019.