Optimizing the C4.5 Decision Tree Algorithm using MSD-Splitting

Patrick Rim[1]
California Institute of Technology
Pasadena, CA 91125, USA

Erin Liu[2]
Troy High School
Fullerton, CA 92831, USA

***Abstract*—We propose an optimization of Dr. Ross Quinlan's C4.5 decision tree algorithm, used for data mining and classification. We will show that by discretizing and binning a data set's continuous attributes into four groups using our novel technique called MSD-Splitting, we can significantly improve both the algorithm's accuracy and efficiency, especially when applied to large data sets. We applied both the standard C4.5 algorithm and our optimized C4.5 algorithm to two data sets obtained from UC Irvine's Machine Learning Repository: Census Income and Heart Disease. In our initial model, we discretized continuous attributes by splitting them into two groups at the point with the minimum expected information requirement, in accordance with the standard C4.5 algorithm. Using five-fold cross-validation, we calculated the average accuracy of our initial model for each data set. Our initial model yielded a 75.72% average accuracy across both data sets. The average execution time of our initial model was 1,541.57 s for the Census Income data set and 50.54 s for the Heart Disease data set. We then optimized our model by applying MSD-Splitting, which discretizes continuous attributes by splitting them into four groups using the mean and the two values one standard deviation away from the mean as split points. The accuracy of our model improved by an average of 5.11% across both data sets, while the average execution time reduced by an average of 96.72% for the larger Census Income data set and 46.38% for the Heart Disease data set.**

***Keywords*—C4.5 Algorithm; decision tree; data mining; machine learning; classification**

I. Introduction

In machine learning, classification is a type of supervised learning with many useful applications, from catching spam emails [1] to categorizing tumor scans [2]. Classification problems analyze data sets containing a collection of records that each have a set of attributes and a class label. The task is to create a model that maps each record's attribute set onto its class label. A classification model can be used for descriptive purposes by summarizing the attributes in a data set that correlate with a specific class label [3]. It can also be used for predictive purposes by classifying new records with unknown class labels [3].

There are many different ways to create a classification model based on a data set, but they all follow a similar approach. First, the data set must be split into a set of training data and a set of testing data. The training data is composed of records where the class label is included [4]. Each classification technique applies a different learning algorithm to the training data in order to build the classification model. Once the model is constructed, it is then applied to the testing data, which is composed of records where the class label is removed [4]. The accuracy of the model can then be calculated by comparing the class labels predicted by the model to the actual class labels of the testing data.

There are two general types of attributes in a data set. Discrete attributes are composed of values from a finite or countably infinite set, such as the set of natural numbers or non-numeric values [5]. Continuous attributes are composed of values from an uncountably infinite set, such as the set of real numbers, which includes all decimal values [5]. Many learning algorithms can only use discrete attributes [6]. Thus, these algorithms must employ different methods to discretize continuous attributes. One such algorithm, the C4.5 decision tree algorithm, splits continuous attributes into two groups at a split point that minimizes the expected information requirement [7]. In order to do this, the C4.5 algorithm calculates the expected information requirement for each possible split point. However, this method is inefficient and can consume large amounts of time, especially when applied to large data sets [8].

In this paper, we propose an optimization of the C4.5 decision tree algorithm that discretizes continuous attributes using our novel technique called MSD-Splitting. We will show that our optimization significantly improves both the accuracy and efficiency of the C4.5 algorithm. This paper is structured as follows: Section 2 describes relevant related work. Section 3 details the steps of the standard C4.5 decision tree algorithm. Section 4 describes MSD-Splitting and how it optimizes the C4.5 algorithm. In Section 5, we discuss and compare the results of the standard C4.5 algorithm and our optimized C4.5 algorithm when applied to two different data sets. In Section 6, we summarize our findings and present our conclusion.

II. Related Work

There has been an extensive amount of work done on the Census Income and Heart Disease data sets from UC Irvine's Machine Learning Repository using various approaches. One work done by Chakrabarty and Biswas [9] applies the Gradient Booster Classifier Model to the Census Income data set and calculates its accuracy. Another work done by Hedeshi and Abadeh [10] applies a fuzzy-boosting PSO approach to the Heart Disease data set to detect coronary artery disease.

There has also been an extensive amount of work done on the C4.5 decision tree algorithm, including the calculation of its accuracy and efficiency when applied to various data sets. For instance, a work done by Budiman et al. [11] calculates the accuracy of the C4.5 algorithm when applied to a student

data set. Another work done by Chauhan and Chauhan [12] calculates the accuracy of the C4.5 algorithm when applied to data sets of various sizes, as well as data sets containing noisy or missing data. Hssina, Merbouha, Ezzikouri, and Erritali [13] calculated the efficiency of the C4.5 algorithm when applied to a data set describing weather by measuring execution time. However, none of these works offer an optimization of the C4.5 algorithm.

There are works done that do offer an optimization to either the accuracy or the efficiency of the C4.5 algorithm. A work done by Muslim, Nurzahputra, and Prasetiyo [14] shows that the accuracy of the C4.5 algorithm can be improved by using the Split Feature Reduction Model and Bagging Ensemble. Agrawal and Gupta [8] showed that applying L'Hospital's Rule to the C4.5 algorithm improves its efficiency. Another work done by Yang and Chen [15] proposes a novel algorithm called Taiga that improves the efficiency of the C4.5 algorithm. However, these works do not offer an improvement to both the accuracy and efficiency of the C4.5 algorithm.

For this reason, we applied our novel technique of MSD-Splitting to the C4.5 algorithm in order to improve both the accuracy and efficiency of the algorithm. The efficiency of our optimized C4.5 algorithm improves from the efficiency of the standard C4.5 algorithm to a greater degree when applied to larger data sets. Since we created the technique of MSD-Splitting, there is currently no work in the literature that references it.

III. C4.5 Decision Tree Algorithm

Decision tree induction is a common classification technique used to build models [16]. The learning algorithms that decision trees use differ in the methods that they employ to select the attribute that is used to split the records at a given point in the tree. Dr. Ross Quinlan was instrumental in the development of decision tree learning algorithms, inventing the widespread ID3 and C4.5 algorithms [17][18]. These two algorithms both select attributes using a concept called information gain [17][18]. We will now describe the steps of the standard C4.5 algorithm.

A. Information Gain

To determine which attribute to use to split the records in a given node, which is a point on the decision tree, information gain is calculated for all of the attributes, and the one with the highest information gain is selected [19]. Information gain is the expected drop in entropy after the records in a node are split using a certain attribute [19]. In other words, it is a measure of how much information the model gains from splitting the records in a node using a certain attribute. Information gain is calculated using the following formula [19]:

$$\text{Gain}(A) = \text{Entropy}(D) - \text{Entropy}_A(D) \tag{1}$$

where A is a given attribute in a data set D.

Entropy, which is the expected information needed to classify a record in a given node, is calculated using the following formula [19]:

$$\text{Entropy}(D) = -\sum_{i=1}^{m} p_i \log_2(p_i) \tag{2}$$

where m is the total number of classes and p_i is the probability that any given record in the node belongs to the class i.

Entropy_A, which is the information needed to classify the records in a given node after it is partitioned using a certain attribute, is calculated using the following formula [19]:

$$\text{Entropy}_A(D) = \sum_{j=1}^{n} p_j \cdot \text{Entropy}(D_j) \tag{3}$$

where n is the number of partitions, p_j is the probability that any given record in the node is in partition j, and D_j is the subset of records that are in partition j.

B. Gain Ratio

In Quinlan's ID3 algorithm, information gain was used exclusively to select splitting attributes [17]. However, the attribute selection method of using the highest information gain has an inherent bias towards attributes that have a larger number of different values, because these attributes will produce a larger number of outcomes to be summed when chosen as the splitting attribute [18]. Due to this issue, Quinlan invented the C4.5 algorithm, a successor of the ID3 algorithm, which optimizes the information gain calculation process [18]. In the C4.5 algorithm, the information gain calculation is normalized to account for the number of outcomes a particular attribute will produce [18]. This normalized attribute selection measure is known as gain ratio. Gain ratio is the ratio of the information gain of a certain attribute to its split information [20]. It is calculated using the following formula [20]:

$$\text{GainRatio}(A) = \text{Gain}(A)/\text{SplitInfo}(A) \tag{4}$$

where A is a given attribute in a data set D.

SplitInfo, which is used to normalize the information gain calculation, is calculated using the following formula [20]:

$$\text{SplitInfo}(A) = -\sum_{j=1}^{n} p_j \log_2(p_j) \tag{5}$$

where n is the number of partitions and p_j is the probability that any given record in the node is in partition j. Since SplitInfo increases as the number of partitions increases, it normalizes the information gain of attributes with a large number of partitions.

C. Splitting Continuous Attributes

As mentioned previously, the C4.5 algorithm can only use discrete attributes to classify a data set [6]. The C4.5 algorithm divides records based on discrete attributes by creating branches on the decision tree for each distinct value [18]. However, this is difficult with continuous attributes as there may be too many distinct values. Creating a branch on the decision tree for each distinct value can lead to overfitting, an error where a model is too specific to a particular set of data, causing it to perform poorly when given new, unseen data [21]. The standard C4.5 algorithm addresses this problem by splitting continuous attributes into two groups at an ideal split point [18]. The set of possible split points is given by the set of the midpoints between any two adjacent values in the attribute. The ideal split point for a given continuous attribute is the point

with the minimum expected information requirement [7]. The expected information requirement calculation is equivalent to the Entropy$_A$ calculation [19].

D. Decision Tree Building

Building the decision tree is a recursive process, as shown in Fig. 1. First, all of the records in the training data are placed in the top node, or the root node, of the decision tree. The attribute with the highest gain ratio is chosen as the first splitting attribute [22]. Once the data set is split into different nodes based on the first attribute, each node is then split based on the attribute with the highest gain ratio when applied to the data in the node. Each node is given a majority label based on the class label of the majority of the records in the node [22]. This process is continued until one of the three stopping conditions is met: 1) all of the records in the node belong to the same class; 2) the node is empty; 3) none of the attributes provide any further information gain [22]. Once a stopping condition is met, the final node is considered a leaf and is given a class label [22]. Once every record in the data set is placed into a leaf, the decision tree building process is complete.

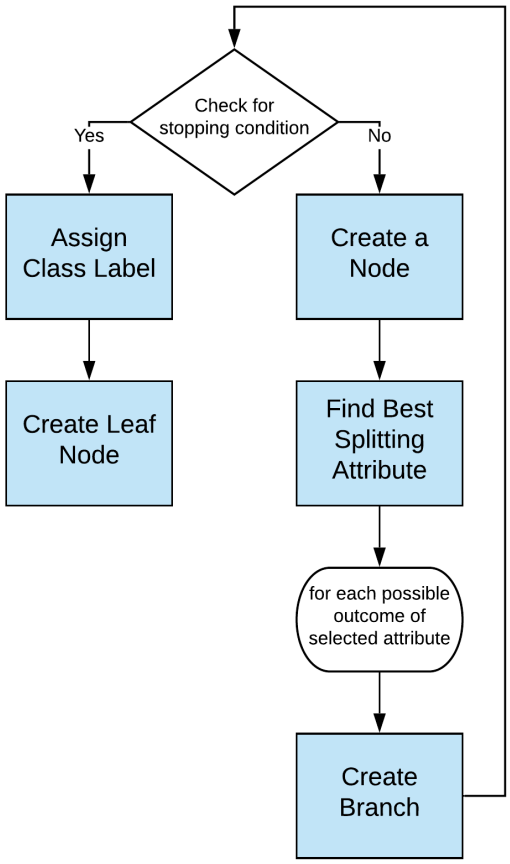

Fig. 1. Flowchart of the Recursive C4.5 Decision Tree Building Algorithm.

As an example, Fig. 2 shows the top two levels of our decision tree for the Heart Disease data set. Our model calculated that the 'Chest Pain' attribute yielded the highest gain ratio for the root node, which is why it was chosen as the first splitting attribute. Then, we can see that four new nodes were created based on the values of the 'Chest Pain' attribute. We can also see that for each node, our model chose different attributes as the next splitting attribute. These attributes were chosen because they yielded the highest gain ratio for their respective nodes.

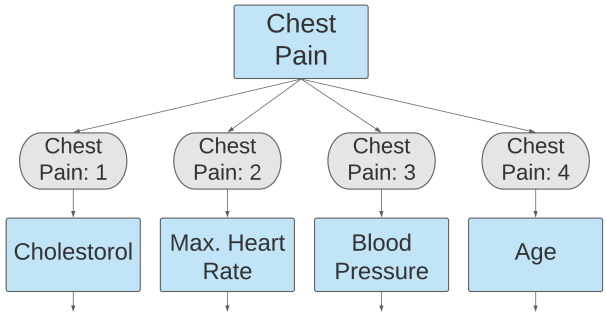

Fig. 2. Our Decision Tree for the Heart Disease Data Set.

Fig. 3 shows a segment of the bottom end of a generic decision tree. Leaves are created when one of the stopping conditions is met. They can be created on different levels of the decision tree.

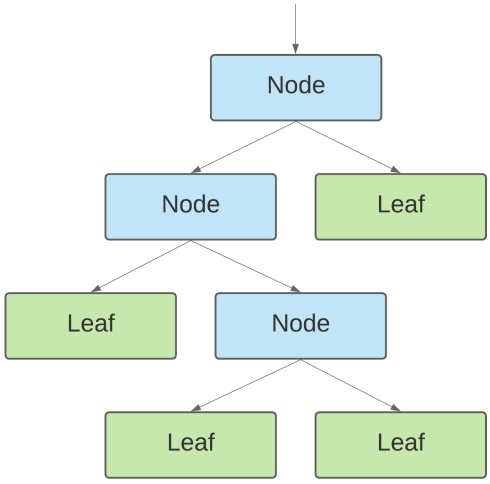

Fig. 3. Generic Diagram of Decision Tree Leaves.

In practice, the records in a leaf are not always guaranteed to have the same class label. For instance, the records in a leaf may still consist of different class labels while none of the attributes provide further information gain. In this case, the leaf is labeled with the class label of the majority of the records in the leaf [22].

E. Classifying New Data

Once the decision tree is built, it can be used to classify new, unclassified data [16]. Each new record is passed through the tree and branched at each node based on its attribute values until it is finally classified into a leaf [23]. The record is then labeled with the class label of the leaf. If the record reaches a

node that does not contain a branch for the record's attribute value, the record is labeled with the majority label of the node [22].

IV. MSD-Splitting

We will now describe MSD-Splitting and how it improves the accuracy and efficiency of the C4.5 algorithm. MSD-Splitting is short for Mean and Standard Deviation Splitting. The mean (μ) of an attribute is the average of its values. It is calculated using the following formula [24]:

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n} \tag{6}$$

where x_i is the i-th value and n is the total number of values in the attribute.

The standard deviation (σ) of an attribute is a measure of the dispersion of its values. It is calculated using the following formula [25]:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n}} \tag{7}$$

where x_i is the i-th value, μ is the mean, and n is the total number of values in the attribute.

Thus, we can write the two values that are one standard deviation away from the mean as

$$\mu - \sigma,\ \mu + \sigma$$

where μ is the mean and σ is the standard deviation of the attribute.

Standard deviation is important because it measures the typical dispersion of the values in the attribute [25]. Thus, values within one standard deviation away from the mean can be considered to have a lower than typical deviation, while values more than one standard deviation away from the mean can be considered to have a higher than typical deviation.

MSD-Splitting splits each continuous attribute into four groups, using its mean (μ) and the values one standard deviation away from the mean ($\mu - \sigma$, $\mu + \sigma$) as logical split points. A value x in the attribute is binned into one of four groups, which are defined as follows:

1) $x < \mu - \sigma$: below mean, high deviation
2) $\mu - \sigma \leq x < \mu$: below mean, low deviation
3) $\mu \leq x < \mu + \sigma$: above/at mean, low deviation
4) $x \geq \mu + \sigma$: above mean, high deviation

The values in each of these four groups are estimated to share a similar deviation and to be closely related to each other. The four groups are labeled on the graph of a normal distribution in Fig. 4.

A. Application to the C4.5 Algorithm

The standard C4.5 algorithm splits continuous attributes into two groups by choosing one split point with the minimum expected information requirement [7]. However, this process is often inaccurate and inefficient. The two groups will likely provide little information gain because the values in each of the two groups are not likely to be closely related to each other. Attributes with low information gain are detrimental to the overall accuracy of the model [26]. Furthermore, iterating through every possible split point in a large data set and calculating the expected information requirement for each point can consume large amounts of time [8].

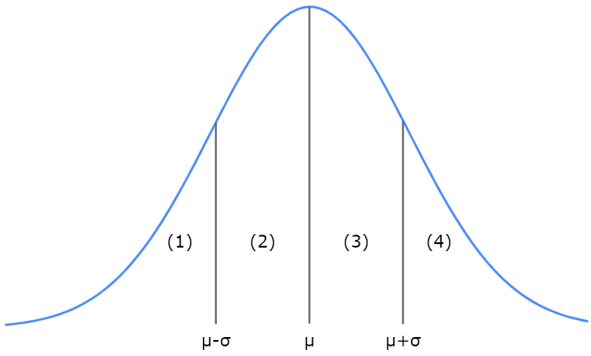

Fig. 4. Four Groups Created by MSD-Splitting on the Graph of a Normal Distribution.

Our optimized C4.5 algorithm instead splits continuous attributes using MSD-Splitting. By doing so, we improve both the accuracy and efficiency of the C4.5 algorithm.

B. Effect on Accuracy

Increasing the number of groups into which the values are binned means that the values in each group will generally be more closely related to each other [27]. Splitting continuous attributes using MSD-Splitting doubles the number of groups from two to four. The values in each of the four groups are likely to be more closely related to each other than the values in each of the two groups created by the standard C4.5 algorithm. It is then likely that the discretized attributes will provide higher information gain when it is discretized using MSD-Splitting rather than using the standard C4.5 algorithm's method. Attributes with higher information gain improve the overall accuracy of the classification model [26].

Furthermore, since the groups are created using logical split points that group values with low deviation together and values with high deviation together, we estimate that the values in each group will be even more closely related to each other. Thus, the discretized attributes are even more likely to provide higher information gain [28], improving the overall accuracy of the model.

C. Effect on Efficiency

Discretizing continuous attributes using MSD-Splitting can significantly improve the efficiency of the C4.5 algorithm, especially when it is applied to large data sets. The standard C4.5 algorithm iterates through each possible split point and calculates its expected information requirement. The expected information requirement calculation, which is equivalent to Equation 3, has a linear time complexity [29]. Then, running this calculation for every possible split point has a quadratic time complexity, or a time complexity of $O(n^2)$. On the other hand, finding the split points for the MSD-Splitting method only requires the calculation of the attribute's mean and standard deviation, which has a linear time complexity, or

TABLE I. ACCURACY OF INITIAL AND OPTIMIZED MODELS

Data Set Group	Accuracy of Initial Model (%)	Accuracy of Optimized Model (%)	% Change
Census Income_1234_test5	78.64	80.49	+2.35%
Census Income_1235_test4	78.46	81.28	+3.59%
Census Income_1245_test3	79.84	81.57	+2.17%
Census Income_1345_test2	78.13	81.20	+3.93%
Census Income_2345_test1	79.16	81.94	+3.51%
Census Income Average	**78.85**	**81.30**	**+3.11%**
Heart Disease_1234_test5	67.21	73.77	+9.76%
Heart Disease_1235_test4	73.77	78.69	+6.67%
Heart Disease_1245_test3	80.33	81.97	+2.04%
Heart Disease_1345_test2	63.33	66.67	+5.27%
Heart Disease_2345_test1	78.33	88.33	+12.77%
Heart Disease Average	**72.59**	**77.89**	**+7.29%**
Overall Average	**75.72**	**79.59**	**+5.11%**

TABLE II. EXECUTION TIME OF INITIAL AND OPTIMIZED MODELS

Data Set Group	Exec Time of Initial Model (s)	Exec Time of Optimized Model (s)	% Change
Census Income_1234_test5	1567.50	48.40	-96.91%
Census Income_1235_test4	1575.55	50.98	-96.76%
Census Income_1245_test3	1569.91	48.74	-96.90%
Census Income_1345_test2	1456.98	53.30	-96.34%
Census Income_2345_test1	1537.94	51.30	-96.66%
Census Income Average	**1541.57**	**50.54**	**-96.72%**
Heart Disease_1234_test5	0.04379	0.02660	-39.25%
Heart Disease_1235_test4	0.03903	0.01905	-51.20%
Heart Disease_1245_test3	0.04052	0.02186	-46.06%
Heart Disease_1345_test2	0.03790	0.01968	-48.09%
Heart Disease_2345_test1	0.04137	0.02146	-48.12%
Heart Disease Average	**0.04052**	**0.02173**	**-46.38%**

a time complexity of O(n). Since O(n) is more efficient than O(n^2), we can see that MSD-Splitting improves the efficiency of the C4.5 algorithm. The improvement in efficiency will be greater for larger data sets due to the quadratic time complexity of the standard C4.5 algorithm's method of finding the ideal split point, which grows at a faster rate as the number of values and possible split points increases [30].

V. EXPERIMENTAL RESULTS

We ran both the standard C4.5 algorithm and our optimized C4.5 algorithm with MSD-Splitting on two data sets obtained from UC Irvine's Machine Learning Repository: Census Income, used to predict whether a person's income exceeds $50,000 per year, and Heart Disease, used to predict whether a person has heart disease [31]. The Census Income data set contains 48,842 records and 14 attributes, while the Heart Disease data set contains 303 records and 76 attributes. We worked with the commonly used Cleveland subset of the Heart Disease data set with 14 attributes. In our initial model, we applied the standard C4.5 algorithm to both data sets, splitting continuous attributes by minimizing the expected information requirement. In our optimized model, we applied our optimized C4.5 algorithm with MSD-Splitting. We then calculated the accuracy and efficiency of our initial and optimized models.

A. Accuracy

To calculate the accuracy of our two models, we split both data sets into five groups and performed five-fold cross-validation for each data set where we ran both models on each data set five times, using one of the groups as the testing data and the other four groups as the training data for each trial [32]. The first trial used the first group of the data set as the testing data, the second trial used the second group as the testing data, and so on. We then averaged the results of each of the five trials to obtain the accuracy of our two models for each data set.

Table I displays the accuracy of our initial and optimized models applied to the Census Income and Heart Disease data sets. The table also displays the % change in accuracy between our two models for each data set.

B. Efficiency

To calculate the efficiency of our two models, we measured the average execution time of both models for each data set, where a low execution time equals high efficiency and a high execution time equals low efficiency. We calculated average execution time using the same five-fold cross-validation technique that we used to calculate the accuracy of our two models.

Table II displays the execution time of our initial and optimized models applied to the Census Income and Heart Disease data sets. The table also displays the % change in execution time between our two models for each data set.

Since the Census Income data set is significantly larger than the Heart Disease data set, the difference between the execution time of our initial model and our optimized model is significantly greater for the Census Income data set than for the Heart Disease data set. This is due to the behavior of the previously mentioned quadratic time complexity of our initial model [30]. In order to account for the different sizes of the data sets, we must compare the efficiency of our two models separately for each data set. Since the overall average execution time and the overall average % change in execution time between our two models do not provide any meaningful information, we chose not to display this information.

C. Discussion

We can see that the overall average accuracy of our initial model is 75.72%, while the overall average accuracy of our optimized model is 79.59%. We can calculate that our optimized model has a 5.11% increase in accuracy from our initial model. Since the only portion of the C4.5 algorithm that was changed between our two models was the method of splitting continuous attributes, we can conclude that splitting continuous attributes using MSD-Splitting rather than by minimizing the expected information requirement increases the accuracy of the C4.5 algorithm.

Our optimized model has a 96.72% decrease in execution time from our initial model when applied to the Census Income data set, compared to a smaller 46.38% decrease when applied to the Heart Disease data set.

Since the execution time of the standard C4.5 algorithm's method of finding the ideal split point increases quadratically with the number of values and possible split points, our initial model takes significantly longer to run when applied to data sets that have continuous attributes with many distinct values. Larger data sets tend to have more distinct values in their continuous attributes than smaller data sets. Thus, our initial model has a longer execution time when applied to larger data sets. While the execution time of the standard C4.5 algorithm's method of finding the ideal split point increases quadratically, the execution time of our optimized C4.5 algorithm's method of finding the split points only increases linearly. This explains why the efficiency of our optimized model improves from the efficiency of our initial model to a greater degree when applied to the larger Census Income data set than when it is applied to the smaller Heart Disease data set.

Again, since the only portion of the C4.5 algorithm that was changed between models was the method of splitting continuous attributes, we can conclude that splitting continuous attributes using MSD-Splitting rather than by minimizing the expected information requirement increases the efficiency of the C4.5 algorithm, especially when applied to larger data sets.

VI. Conclusion

In this paper, we described our novel technique of MSD-Splitting and how it improves the accuracy and efficiency of the C4.5 decision tree algorithm. In our initial model, we applied the standard C4.5 algorithm to two different data sets. Then, in our optimized model, we applied our optimized C4.5 algorithm with MSD-Splitting to the same two data sets. After calculating the accuracy and efficiency of our initial and optimized models, we can conclude that splitting continuous attributes using MSD-Splitting significantly improves the accuracy of the C4.5 algorithm. We can also conclude that MSD-Splitting significantly improves the efficiency of the C4.5 algorithm, especially when applied to large data sets. This is because the execution time of the standard C4.5 algorithm has a quadratic time complexity, while our optimized C4.5 algorithm with MSD-Splitting has a more efficient linear time complexity. Since the increase in execution time of the standard C4.5 algorithm grows faster as the size of the data set increases, our optimized C4.5 algorithm will have an even greater improvement in efficiency from the standard C4.5 algorithm when applied to data sets that are even larger than the ones that we used. For this reason, we believe that our optimized C4.5 algorithm with MSD-Splitting is ideal for classification tasks involving extremely large data sets.

Classifying data sets is critically important in the increasingly consequential fields of data mining and machine learning. As the amount of data that we create grows exponentially [33], we must be able to extract and interpret useful information from this data as accurately and efficiently as possible. By optimizing the widespread C4.5 decision tree algorithm, we refine and expedite the data classification process.

Acknowledgment

This research was conducted at California State University, Fullerton. We would like to thank our research supervisor Dr. Shawn X. Wang, professor of computer science at California State University, Fullerton, for providing invaluable guidance and support.

References

[1] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi, and O. E. Ajibuwa, "Machine learning for email spam filtering: review, approaches and open research problems," *Heliyon*, vol. 5, no. 6, Jun. 2019.

[2] T. T. Tang, J. A. Zawaski, K. N. Francis, A. A. Qutub, and M. W. Gaber, "Image-based Classification of Tumor Type and Growth Rate using Machine Learning: a preclinical study," *Scientific Reports*, vol. 9, no. 1, Aug. 2019.

[3] J. L. Rastrollo-Guerrero, J. A. Gómez-Pulido, and A. Durán-Domínguez, "Analyzing and Predicting Students' Performance by Means of Machine Learning: A Review," *Applied Sciences*, vol. 10, no. 3, p. 1042, Feb. 2020.

[4] M. A. Shafique and E. Hato, "Formation of Training and Testing Datasets, for Transportation Mode Identification," *Journal of Traffic and Logistics Engineering*, vol. 3, no. 1, Jun. 2015.

[5] J. Han, M. Kamber, and J. Pei, "2 - Getting to Know Your Data," in *Data Mining*, 3rd ed., Morgan Kaufmann, 2012, pp. 39–82.

[6] U. M. Fayyad and K. B. Irani, "On the handling of continuous-valued attributes in decision tree generation," *Machine Learning*, vol. 8, no. 1, pp. 87–102, 1992.

[7] J. R. Quinlan, "Improved Use of Continuous Attributes in C4.5," *Journal of Artificial Intelligence Research*, vol. 4, pp. 77–90, Mar. 1996.

[8] G. L. Agrawal and H. Gupta, "Optimization of C4.5 Decision Tree Algorithm for Data Mining Application," *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, no. 3, Mar. 2013.

[9] N. Chakrabarty and S. Biswas, "A Statistical Approach to Adult Census Income Level Prediction," *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, Oct. 2018.

[10] N. G. Hedeshi and M. S. Abadeh, "Coronary Artery Disease Detection Using a Fuzzy-Boosting PSO Approach," *Computational Intelligence and Neuroscience*, pp. 1–12, Apr. 2014.

[11] E. H. Budiman, H. H. Haviluddin, N. H. Dengan, A. H. Kridalaksana, M. H. Wati, and Purnawansyah, "Performance of Decision Tree C4.5 Algorithm in Student Academic Evaluation," *Lecture Notes in Electrical Engineering*, Feb. 2018.

[12] H. Chauhan and A. Chauhan, "Implementation of decision tree algorithm c4.5," *International Journal of Scientific and Research Publications*, vol. 3, no. 10, Oct. 2013.

[13] B. Hssina, A. Merbouha, H. Ezzikouri, and M. Erritali, "A comparative study of decision tree ID3 and C4.5," *International Journal of Advanced Computer Science and Applications (IJACSA), Special Issue on Advances in Vehicular Ad Hoc Networking and Applications*, Jun. 2014.

[14] M. A. Muslim, A. Nurzahputra, and B. Prasetiyo, "Improving accuracy of C4.5 algorithm using split feature reduction model and bagging ensemble for credit card risk prediction," *2018 International Conference on Information and Communications Technology (ICOIACT)*, Mar. 2018.

[15] Y. Yang and W. Chen, "Taiga: performance optimization of the C4.5 decision tree construction algorithm," *Tsinghua Science and Technology*, vol. 21, no. 4, pp. 415–425, Aug. 2016.

[16] R. H. A. Alsagheer, A. F. H. Alharan, and A. S. A. Al-Haboobi, "Popular Decision Tree Algorithms of Data Mining Techniques: A Review," *International Journal of Computer Science and Mobile Computing*, vol. 6, no. 6, pp. 133–142, Jun. 2017.

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

[18] S. L. Salzberg, "C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993," *Machine Learning*, vol. 16, no. 3, pp. 235–240, Sep. 1994.

[19] S. Singh and P. Gupta, "Comparative Study ID3, CART, and C4.5 Decision Tree Algorithm: A Survey," *International Journal of Advanced Information Science and Technology (IJAIST)*, vol. 27, Jul. 2014.

[20] A. Rizka, S. Efendi, and P. Sirait, "Gain ratio in weighting attributes on simple additive weighting," *IOP Conference Series: Materials Science and Engineering*, Oct. 2018.

[21] R. Jothikumar and B. R. V. Siva, "C4.5 classification algorithm with back-track pruning for accurate prediction of heart disease," *Biomedical Research*, 2016.

[22] P. Tan, M. Steinbach, A. Karpatne, and V. Kumar, "Chapter 4: Classification: Basic Concepts, Decision Trees, and Model Evaluation," in *Introduction to Data Mining*, Pearson Addison-Wesley, 2006.

[23] H. Sharma and S. Kumar, "A Survey on Decision Tree Algorithms of Classification in Data Mining," *International Journal of Science and Research (IJSR)*, vol. 5, no. 4, pp. 2094–2097, Apr. 2016.

[24] H. Hassani, M. Ghodsi, and G. Howell, "A note on standard deviation and standard error," *Teaching Mathematics and its Applications*, May 2010.

[25] P. J. Barde and M. P. Barde, "What to use to express the variability of data: Standard deviation or standard error of mean?," *Perspectives in Clinical Research*, vol. 3, no. 3, pp. 113–116, 2012.

[26] D. Rajeshingo and J. P. A. Jebamalar, "Accuracy Improvement of C4.5 using K means Clustering," *International Journal of Science and Research (IJSR)*, 2015, ISSN 2319-7064.

[27] D. G. Altman and P. G. Royston, "The cost of dichotomising continuous variables," *BMJ Statistics Notes*, May 2006.

[28] H. Dag, K. E. Sayin, I. Yenidogan, S. Albayrak, and C. Acar, "Comparison of feature selection algorithms for medical data," *2012 International Symposium on Innovations in Intelligent Systems and Applications*, Jul. 2012.

[29] J. Su and H. Zhang, "A Fast Decision Tree Learning Algorithm," *AAAI'06: Proceedings of the 21st national conference on Artificial intelligence*, vol. 1, pp. 500–505, Jul. 2006.

[30] E. A. Graf, J. H. Fife, H. Howell, and E. Marquez, "The Development of a Quadratic Functions Learning Progression and Associated Task Shells," *ETS Research Report Series*, vol. 2018, no. 1, pp. 1–28, Dec. 2018.

[31] B. Becker and R. Detrano, *UCI Machine Learning Repository*, 1996. [Online]. Available: http://archive.ics.uci.edu/ml.

[32] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," *IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence*, vol. 2, pp. 1137–1143, Aug. 1995.

[33] R. Devakunchari, "Analysis on big data over the years," *International Journal of Scientific and Research Publications*, vol. 4, no. 1, Jan. 2014.