MC4.5 decision tree algorithm: an improved use of continuous attributes

Anis Cherfi*, Kaouther Nouira and Ahmed Ferchichi

Université de Tunis,
ISGT,
LR99ES04 BESTMOD,
2000, Le Bardo, Tunisia
Email: cherfianis@gmail.com
Email: kaouther.nouira@planet.tn
Email: ahmad.ferchichi@gmail.com
*Corresponding author

Abstract: C4.5 is one of the top ten data mining algorithms; it is the most widely used decision trees construction techniques. Although effective, it suffers from the problem of complexity when it deals with continuous attributes. It also leads to a certain level of information loss. Therefore, minimising such loss and reducing the time complexity is one of the main goals in this paper. With the intention of alleviating these problems, this paper presents a novel algorithm namely MC4.5, which proposes the statistical mean as an alternative to the C4.5 threshold selection process. To demonstrate the effectiveness of the new algorithm, a complete evaluation was launched to prove that MC4.5 complies with the objectives previously mentioned. From the theoretical perspective, we develop an analysis of the complexity to compare algorithms. Empirically, we conduct an experimental study using 30 datasets to prove that, in most cases, the proposed algorithm leads to smaller decision trees with better accuracy comparing to the C4.5 algorithm.

Keywords: decision tree; modified C4.5; MC4.5; statistical mean; continuous attributes; classification; information gain; C4.5.

Reference to this paper should be made as follows: Cherfi, A., Nouira, K. and Ferchichi, A. (2020) 'MC4.5 decision tree algorithm: an improved use of continuous attributes', *Int. J. Computational Intelligence Studies*, Vol. 9, Nos. 1/2, pp.4–17.

Biographical notes: Anis Cherfi received his MSc in Statistical Machine Learning in 2015 from the Institut Supérieur de Gestion de Tunis. He is currently a PhD student at the Department of Computer Science at Institut Supérieur de Gestion de Tunis. His research interests include data mining, decision trees, data discretisation, big data and time series analysis.

Kaouther Nouira held the position of the IT Department Director at the Institut Supérieur de Gestion de Tunis during the period 2014–2017. She worked mainly on improving the quality of IT education. She is the Founder of Applied Business Intelligence Diplôma. She is the Head of the EHealth Research Team at Business and Economic Statistics Modeling (BESTMOD) Laboratory. His research has resulted in several high-quality publications and led to the implementation of several prototypes of intelligent medical surveillance systems.

Ahmed Ferchichi obtained his PhD from the Joseph Fourier University (Grenoble, France, 1983). He has been a Professor of Computer Science since 1980 at the University of Tunis, then at the University of Jendouba. His research interests are in the following areas: software engineering, pedagogy, curriculums, computational thinking, sustainable education, and dissemination of scientific culture in Arabic. In this context, he has written several articles and organised several international events.

This paper is a revised and expanded version of a paper entitled 'MC4.5 decision tree algorithm: an improved use of continuous attributes' presented at Doctoral Workshop on Computing, Management and Decision Making (CMDM), Tozeur, Tunisia, 20–22 February 2017.

1 Introduction

Decision trees are one of the most widely used classification techniques. Many variations of decision tree algorithm were proposed in the literature (Saqib et al., 2015). They include classification and regression tree (CART) (Breiman, 1984), iterative dichotomiser 3 (ID3) (Quinlan, 1986), chi-squared automatic interaction detector (CHAID) (Kass, 1980) and conditional inference trees (Hothorn et al., 2006). A decision tree is a classifier expressed as a recursive partition of the training instances. It is constructed in a top-down manner. In each iteration, the instance space is partitioned by choosing the best attribute to split them (Agrawal and Gupta, 2013; Patel and Singh, 2015).

An attributes in a learning problem may be nominal (categorical) or it may be continuous (numerical). Numerical attributes with a very large, even infinite domain, become an important challenge in areas of pattern recognition, machine learning and data mining. Mining data with numerical attributes requires discretisation before or throughout the process of model building (Garcia et al., 2013). A special kind of discretisation is performed through the decision tree construction process. Decision tree algorithm uses binarisation which splits the numerical values into two intervals (Yang and Chen, 2016).

C4.5 is one of the most widely used decision tree algorithms (Lu et al., 2015). Its accuracy level is high enough, independently of the data volume to be processed. One of the latest studies that compare decision trees and other learning algorithms shows that C4.5 has a very good combination of error rate and speed (Lim et al., 2000; Hssina et al., 2014). Various studies identify C4.5 algorithm as one of the top classifiers in data mining (Wu et al., 2008). The algorithm has the ability to handle incomplete training dataset (García-Laencina et al., 2015), prune the resulted tree in order to reduce its size and optimise the decision path. C4.5 has also the ability to deal with continuous attributes. It handles continuous attributes using binarisation process. Those attributes are replaced by discrete ones using threshold values which separate data into two intervals (Kotsiantis, 2013; Behera and Mohapatra, 2015; Perner, 2015).

Even for C4.5 and other algorithms that can directly deal with quantitative data, learning is often less efficient and less effective. Several researchers reported that C4.5 algorithm contains some weaknesses in domains with continuous attributes. They offer

evidence that it can benefit from continuous attributes by using a new discretisation method, or by developing some process in C4.5 binarisation method (Sumam et al., 2013; Quinlan, 1996; Chong et al., 2014; Ruggieri, 2002). Minimising information loss and reducing the time complexity of C4.5 is one of the main goals in this paper. We therefore propose statistical mean as an alternative to the threshold searching process in C4.5 algorithm. In this paper, a new algorithm called modified C4.5 (MC4.5) is proposed.

The present paper is organised as follows: Section 2 describes in details the C4.5 binarisation process. The new algorithm is stated in Section 3 with a mathematical formulation. Experimental investigations are drawn in Sections 4 and 5, where a set of tests are carried out to measure the efficiency of the MC4.5 applied on different databases in comparison with the initial C4.5 algorithm. Finally, Section 6 concludes the paper.

2 C4.5 decision tree construction algorithm

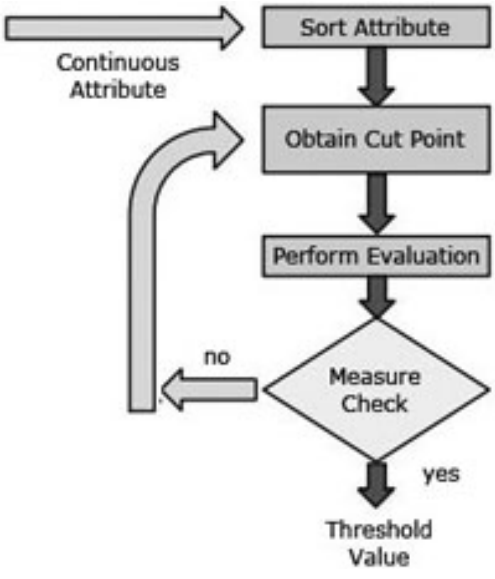
C4.5 decision tree algorithm constructs a decision tree starting from a training set ($D_{m,(n+1)} = [AC]$). Matrix A involves different types of attributes A_j (with $1 \leq j \leq n$), which may have continuous, discrete or unknown values. Classes vector C has discrete values. The training set D contains a set of training cases (A_i, C_i) with $1 \leq i \leq m$. At the training stage, the C4.5 uses the top down strategy based on the divide and conquers approach to construct the decision tree. First, C4.5 algorithm creates a root node that contains the whole training set D, with all training cases weight equal to 1.0. From the root node until arriving at a leaf node, if all training instances associated with the current node belong to one single class, the algorithm creates a leaf node labelled with the most frequent class C_i. Otherwise, it uses the information gain and entropy measures to decide which attribute will be selected for the test at the node. In each internal node, the attribute with the highest information gain is used to split the associated cases. For a discrete attribute A_j, the information gain is computed by splitting D in function of each distinct value of A_j. C4.5 performs test on continuous attributes by comparing the feature values against a certain threshold value, Z. The set of training cases D will be partitioned into two subsets: the first group contains cases with values for attribute A_j are less than or equal to Z and the second group contains cases whose values are greater than Z. So, the problem of dealing with continuous attributes could be addressed by choosing an appropriate threshold value that defines the interval borders (Dimitoglou et al., 2012; Witten et al., 2016; Quinlan, 2014; Elomaa, 2014).

2.1 Handling continuous attributes

As explained in Figure 1, for each continuous attribute A_j, C4.5 algorithm creates a binary split at value Z that yields the best information gain (or information gain ratio). As shown in Figure 1, C4.5 chooses the threshold value Z as follows: the training cases with known attribute values are first sorted in ascending order based on the values of attribute $A_j : \{A_j^{(1)}, A_j^{(2)}, \ldots, A_j^{(m)}\}$ to obtain the vector V: $\{V_1, V_2, \ldots, V_l\}$, where l is the number of distinct values in A_j. The largest value V_l is excluded because it splits off only one training case so there are $l - 1$ possible split on A_j, all of which should be used to compute the information gain considering the split above V_i and then determine the value

V_{best} that yields the best split (the maximal information gain as $gain_{\max}$). V_{best} is set to be the local threshold and $gain_{\max}$ is the information gain for the attribute A_j. It is common to choose the midpoint of each interval: $\frac{V_i + V_{i+1}}{2}$ as the local threshold value. C4.5 uses the entire cases to set the largest value of A_j that does not exceed the midpoint as the threshold value.

Figure 1 C4.5 threshold searching process



In 2013, Sumam et al. reported that the binarisation process in C4.5 is computationally intensive. To perform binarisation, the continuous attribute values are first sorted. This process is time consuming and it is not practical for large datasets. To sort the attribute values, C4.5 algorithm uses quick sort method with a complexity of $O(n\log(n))$. Despite this, several authors have showed that learning process may be dominated by sorting of continuous attribute values.

Generalisation limits in domains with continuous attributes is the most important problem in C4.5 algorithm. The selected threshold value cannot reflect and judge the generalisation capability of the continuous attribute. So, C4.5 algorithm may use continuous attributes with a low generalisation performance to split data. Deciding based on these attributes will increase the tree size and decrease the model accuracy. Liu and Setiono (1995) review and compare C4.5's performance with Chi2 global discretisation and conclude that Chi2 discretisation is effective at improving C4.5 performance. These authors found that Chi2 global discretisation performed as well as C4.5 local discretisation and occasionally improved its accuracy (Liu and Setiono, 1995). A recent research compares multiple scanning and C4.5 discretisation show that using a cut point that reflect the data distribution can improve the decision tree performance. Authors prove that using multiple scanning discretised data yields more accurate and less complex decision trees (Grzymala-Busse and Mroczek, 2015).

3 MC4.5 algorithm

Actually, the selected threshold value in C4.5 algorithm does not reflect data distribution. The MC4.5 algorithm uses the statistical mean as a threshold value. Statistical mean is used to guarantee that only attributes with high generalisation power will be selected to construct the decision tree.

3.1 MC4.5 binarisation process

C4.5*Stat algorithm deals with completely numerical datasets using statistical measures (Sumam et al., 2013), this algorithm uses statistical variance as an alternative to information gain and statistical mean as a splitting criterion, it avoids sorting data to calculate threshold value and so it avoids intensive computationally for threshold search process. C4.5*Stat reduces the modelling time by adopting the statistical mean as a threshold value but this solution cannot deal with discrete attributes. To overcome this problem, we propose the use of statistical mean as a threshold value as in C4.5*Stat, combined with the C4.5 information gain to cope with both continuous and discrete attributes.

Measures of central tendency may be useful in describing data, because they communicate information about fluctuation in data and provide information about the most typical or representative value in a distribution, so it can be useful to find a suitable decision boundary. The assumption here is that, if a block of data is not fluctuated, it has a low variance and then there is a chance that instances converge to a particular class. The mean is the most frequently used of the three measures of central tendency (mode, median and mean). In contrast to the mode or median, the mean is affected by every value. It is a general point that reflects the data fluctuations. In spite of the impact of extreme values, the presence of such values may affect the selected threshold and produce less effective decision tree. But in many cases, extreme values can represent a minority class Extreme values can represent a minority class. Using the mean as a cut point, we are not expecting to have the perfect threshold value, but to have a general value that allows us to use the attribute that perfectly describes data. In our case, we need a value that generalises data and that can be affected by extreme values, so the mean is the fulcrum that exactly balances all of the scores. The mean is the value that generalises and describes data in our case. Using the mean as threshold value will avoid the computational over head of information gain computation to find threshold value. Even in the case of class-imbalanced data, mean splitting method will reflect the strengths and weaknesses of data distribution. It will judge the capability of a given attribute to generalise data distribution. Only attributes that generalise data will be used to build the decision tree. Attributes that separate classes perfectly will not be used, because it may cause overfitting and cannot generalise the data distribution. So, we expect a slightly increase in the tree size.

3.2 MC4.5 algorithm descriptions

The proposed MC4.5 uses the same process as C4.5 to build a decision tree:

1 Let D be the set of training data.

2 Put all of D in a single tree node.

3 Stop if all items in D are in the same class.

4 For each attribute A, find the normalised information gain from splitting on attribute A.

5 Let $A - Best$ be the attribute with the highest normalised gain.

6 Create a decision node that splits items on attribute $A - Best$.

7 Stop splitting if one of the following conditions is met, otherwise continue with step for:

   a If this partition divides data into subsets that belong to single class and no other node needs splitting.

   b If there are no remaining attributes for further splitting.

To compute gain, this algorithm uses the same process as in C4.5 algorithm for attribute with discrete values. If attribute A has continuous numeric values, MC4.5 algorithm uses a different process to select the threshold value. The algorithm select the best attribute based on comparing the value of A against a threshold value Z: $A \leq Z$ and $A > Z$. Here, threshold value Z is the population mean:

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

Algorithm 1 describes the threshold selection process in MC4.5 algorithm. This alternative reduces the complexity in the threshold selection process, now we have no need to calculate information gain of $l - 1$ distinct values, only the mean is calculated and it will be the splitting criterion. Therefore, information gain will be calculated only one time to find the information gain of attribute. Also, if the attribute contains two separated subset, then the threshold value will yield a high information gain value that reflect the high generalisation of this attribute. Otherwise, the attribute will not be used to split data.

Algorithm 1 MC4.5 find threshold algorithm

Inputs: A: a continuous attribute-valued dataset;

Outputs: $A_{\max}$: the local threshold value;

$gain_{\max}$: information gain for attribute A;

Algorithm: FindThreshold(A)

$A_{\max}$ = Mean(A);

$gain_{\max}$ = gain ratio if we split on A_{best};

return A_{best};

3.3 Analysis of time complexity

As we discussed before, the complexity order of the threshold selection process may dominate the leaning process. It is mainly conditioned by the number of candidate cut points. Suppose that we have a dataset with *m* instances and *n* continuous attributes. Also, we suppose that for each feature A_j, we have *l* possible cut points. To select the best threshold value for each attribute, C4.5 algorithm applies the quick sort method to sort the values of the *n* features, the minimum complexity will be $\mathcal{O}(mn\log n)$. Then, constructing the list of possible cut points will lead to a complexity order of $\mathcal{O}(mn)$. To select the threshold value, C4.5 starts by computing the information gain considering the split above each possible cut point for each attribute, which leads to a complexity order of $\mathcal{O}(mnl)$. Finally, it selects the best cut point with a complexity order of $\mathcal{O}(ml)$. According to these steps, we can state that the complexity of C4.5 algorithm in this case is $\mathcal{O}(mn\log n + mn + mnl + ml)$ (Wang et al., 2015; Cherfi et al., 2018). For the same task, the proposed algorithm will avoid the sorting process complexity. Also, information gain will not be computed for *l* possible cut points, but only for the mean value. MC4.5 starts by computing the mean value, which leads to a complexity order of $\mathcal{O}(mn)$. Then, it calculates the information gain for the split above the mean with a complexity of $\mathcal{O}(mn)$. Finally, we avoid the complexity of selecting the optimal cut point while we have only one possible value. To summarise, the complexity of the proposed algorithm is $\mathcal{O}(2mn)$. Compared to C4.5 algorithm, we have reduced the time complexity by more than $\mathcal{O}(mn\log n + ml)$.

4 Experimental setup

We use various dataset to compare MC4.5 and C4.5 algorithms and then investigate the resulted trees. This section presents the experimental study, it specifies all the properties and issues related to datasets, validation procedure and the parameters of used algorithms.

We carried out experiments on 30 datasets taken from the UCI (2016) repository and KEEL dataset repository (Alcalá et al., 2010). Datasets with and without continuous attributes were used to train MC4.5 algorithm. Also, small and large datasets were used in order to investigate MC4.5's performance. Table 1 summarises the main properties of the data. For each dataset, it includes the name, number of instances, number of numeric and nominal attributes and number of classes. Due to the small dataset size, ten-fold cross-validation was used to evaluate the performance of MC4.5.

We used the J48 tree (the C4.5 decision tree implementation) of Weka software (Witten et al., 2016) to implement the new algorithm. MC4.5 program was coded using Java, by changing few Weka classes. Table 2 summarises the parameters of the two classifiers. The default parameter values have been established according to the author's™ criteria. For a complete description of experiments and source code, please refer to our GitHub R Repository (2018).

It is evident that there is a direct dependence between discretisers and the classifier used. Based on Garcia et al. (2013), we have stressed out the two best performing discretisers [ChiMerge (Kerber, 1992) and FUSINTER (Zighed et al., 1998)] in decision trees. The 30 datasets were discretised and the C4.5 classifier was used to establish the

performance measures based on the discretised data. To demonstrate the usefulness and performance of our decision tree algorithm, we use accuracy as a performance measure to compare the generalisation classification rate of MC4.5 against C4.5 algorithm with ant without prior discretisation. Nevertheless, in decision tree, other measures are required to investigate the model complexity. We refer to the tree size, number of leaves and time taken to build the model. Those performance measures will be adopted to measure the tree complexity. For all experiments, we have used the Wilcoxon (1992) test as a non-parametric statistical test. Wilcoxon test is simple, safe and robust test used for statistical comparisons of classifiers.

Table 1 Summary description of classification datasets

#	*Dataset*	*Ex.*	*Atts.*	*Num.*	*Nom.*	*Cl.*
1	Appendicits	106	7	7	0	2
2	Australian	690	14	8	6	2
3	Balance	625	4	4	0	3
4	Balance-scale	625	4	4	0	3
5	Bands	539	19	19	0	2
6	Breast-w	699	10	10	0	2
7	Clevland	303	13	13	0	5
8	Coil2000	9,822	85	85	0	2
9	Credit-c	690	15	6	9	2
10	Credit-g	1,000	20	7	13	2
11	Haberman	306	3	3	0	2
12	Heart Statlog	270	13	13	0	2
13	Heart-c	303	13	6	0	5
14	Labor	57	16	8	8	2
15	Leukemia	72	7,129	7,129	0	2
16	Lymphography	148	19	3	16	4
17	Mammographic	961	5	5	0	2
18	Marketing	6,876	13	13	0	9
19	Newthyroid	215	5	5	0	3
20	Ozone	2,536	73	73	0	2
21	Page blocks	5,472	10	10	0	5
22	Prostate	102	6,034	6,034	0	2
23	Satimage	6,435	36	36	0	7
24	Shuttle	57,999	9	9	0	7
25	Sonar	208	60	60	0	2
26	Spambase	4,597	57	57	0	2
27	Spectfheart	267	44	44	0	2
28	Vehicule	846	18	18	0	4
29	Wdbc	596	30	30	0	2
30	Wisconcon	699	9	9	0	2

Table 2 Parameters of the MC4.5 and C4.5 algorithms

Parameters	*Value*
Pruned tree	True
Confidence factor	0.25
Examples per leaf	2
Reduced-error pruning	3

5 Analysis and empirical results

The collection of training sets, described previously were applied to both MC4.5 and C4.5 algorithms. Table 3 presents the performance of MC4.5 algorithm compared to the C4.5 in term of correctly classified instances over the 30 datasets. Similarly, Table 3 summarises the results associated with tree size and number of leaves. Table 3 contains all detailed results for each dataset, with and without prior discretisation.

We use the test accuracies from Table 3 and run a Wilcoxon signed-rank test considering a level of significance equal to $\alpha = 0{:}05$. Table 4 shows Wilcoxon test results of all used measures. We carried out all possible comparisons using the performance measures in Table 3. Finally, Figure 2 shows the relationship between the accuracy level of each algorithm and the tree size. It summarises the difference in accuracy (MC4.5 accuracy – C4.5 accuracy), also it contains the difference in tree size (C4.5 tree size – MC4.5 tree size). So, positive values in Figure 2 describe cases where MC4.5 outperforms C4.5 algorithm and negative ones summarise cases where MC4.5 fails to improve C4.5 performance. It helps us to define which algorithm yields the best combination of accuracy and model complexity. While Figure 3 summarises the modelling time of the two algorithms, it shows that MC4.5 outperforms the C4.5 algorithm.

In Table 3, we have the classification accuracy results for both algorithms. According to these results, MC4.5 algorithm yields superior accuracy than C4.5 and C4.5 (ChiMerge). Over 30 datasets, it increases the C4.5 accuracy level with 1.65%. Comparing the two classifiers and investigating the impact of the type of attributes, we observe that when the number of continuous attributes increases, MC4.5 acts better than original C4.5 (lines 7, 14, 15 and 22 from Table 3). We can assert that using MC4.5 algorithm leads an improvement in classification accuracy, tree size and classification time. It is especially relevant in high-dimensional data (lines 15 and 22), where there is an improvement of 5% in accuracy. It is clear that only decision trees generated from data discretised by FUSINTER are more accurate than MC4.5. But, C4.5 (FUSINTER) yields more complex trees (lines 10, 26 and 28), also it is clear that discretisation causes serious loss of information in some cases (lines 4, 28 and 30). Results show that the decision trees we obtain can sometimes be less complex than those found by C4.5. Furthermore, our decision trees are competitive in terms of number of leaves and trees size. In average, MC4.5 trees are slightly smaller than C4.5. But the difference between the two algorithms increases when the dataset contains a large number of instances. For lines 18 and 20, there is a remarkable difference in tree size between the two algorithms in favour of MC4.5. And for those cases, MC4.5 algorithm provides more accurate trees than C4.5.

Table 3 Performances of C4.5 and MC4.5 classification algorithms

#	*MC4.5*			*C4.5*			*C4.5 (ChiMerge)*			*C4.5 (FUSINTER)*		
	Accuracy	*Size*	*N. leaves*	*Accuracy*	*Size*	*N. leaves*	*Accuracy*	*Size*	*N. leaves*	*Accuracy*	*Size*	*N. leaves*
1	*88.68*	7	4	85.85	5	3	87.74	5	3	*88.68*	5	4
2	85.07	37	20	84.06	58	31	86.67	28	16	*87.25*	38	25
3	*78.72*	145	73	78.24	103	52	78.08	55	37	74.24	77	56
4	*79.84*	145	73	76.64	103	52	78.40	55	37	74.72	85	64
5	67.67	81	41	63.29	67	34	*70.68*	33	17	69.32	91	75
6	94.85	29	15	94.56	27	14	*95.71*	23	12	95.28	12	9
7	54.21	85	43	48.48	91	46	55.56	61	44	*56.57*	59	35
8	*93.97*	17	9	93.95	17	9	93.92	1	1	94.03	1	1
9	85.80	15	9	86.09	42	30	86.23	25	18	*88.55*	39	29
10	*71.10*	92	62	70.50	140	103	69.30	155	112	70.01	141	121
11	74.51	17	9	71.57	5	3	*76.47*	5	3	75.16	4	3
12	79.26	31	16	76.67	35	18	81.11	32	22	*81.48*	36	23
13	79.54	54	31	77.56	51	30	76.24	15	8	*80.20*	19	11
14	78.95	5	3	73.68	5	3	80.70	8	5	*89.47*	6	4
15	87.50	9	5	81.94	5	3	90.28	5	3	*91.67*	5	3
16	78.38	30	19	77.03	34	21	79.73	30	20	*79.73*	30	19
17	*83.86*	7	4	83.98	9	5	84.70	7	4	83.37	21	16
18	31.14	2,275	1,138	31.06	2,359	1,180	31.65	2,205	1,784	*32.01*	1,857	1,305
19	*94.42*	13	7	92.09	17	9	93.02	22	15	93.95	13	10
20	96.92	1	1	96.33	49	25	96.92	1	1	*97.04*	21	11
21	96.89	111	56	*97.06*	87	44	96.77	111	89	96.29	98	76
22	87.25	11	6	82.35	9	5	84.31	11	6	*92.16*	9	6
23	86.00	649	325	86.28	635	318	*86.93*	1,002	859	86.26	631	386
24	99.96	61	31	*99.97*	47	24	99.92	134	115	99.91	71	45
25	73.56	47	24	72.12	37	19	*74.25*	35	18	74.04	41	33
26	93.21	221	111	92.93	233	117	*93.41*	163	82	92.95	262	174
27	76.40	47	24	74.91	41	21	77.65	35	18	*80.15*	31	22
28	72.70	193	97	72.58	195	98	68.56	157	118	67.21	290	243
29	*95.78*	25	13	94.02	23	12	94.20	19	10	93.85	23	19
30	*95.17*	15	8	*95.90*	21	11	91.17	23	12	93.56	22	16
Mean	82.04	149	76	80.39	152	78	82.01	149	116	82.64	135	95

Figure 2 Accuracy and tree size difference between C4.5 and MC4.5 (see online version for colours)

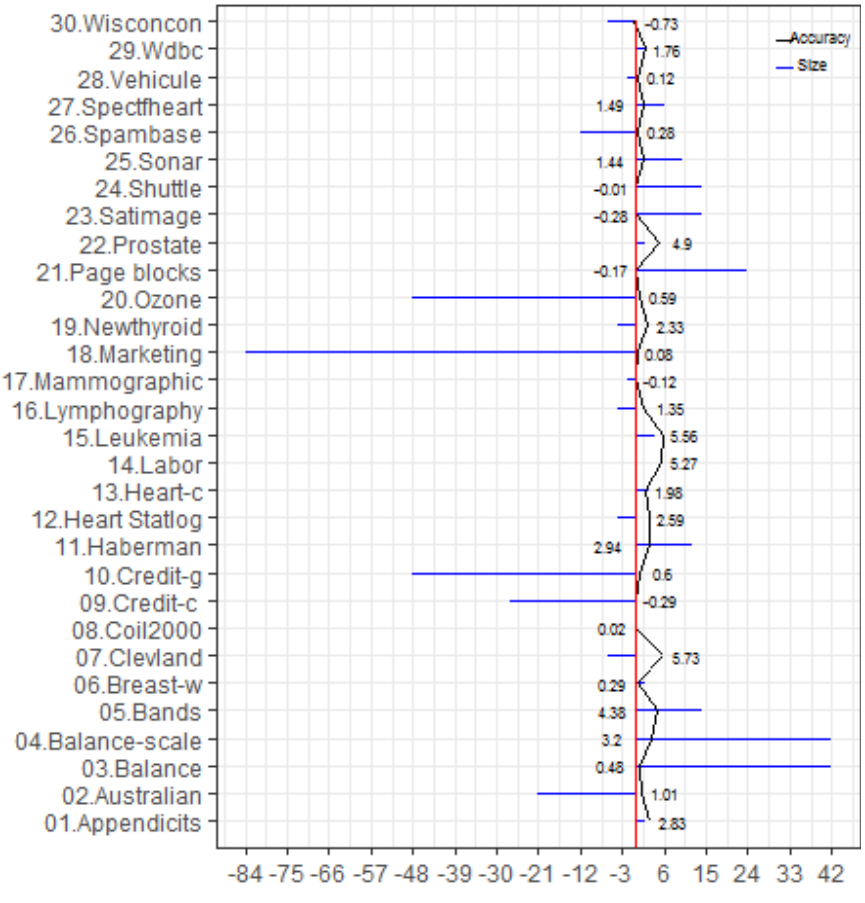
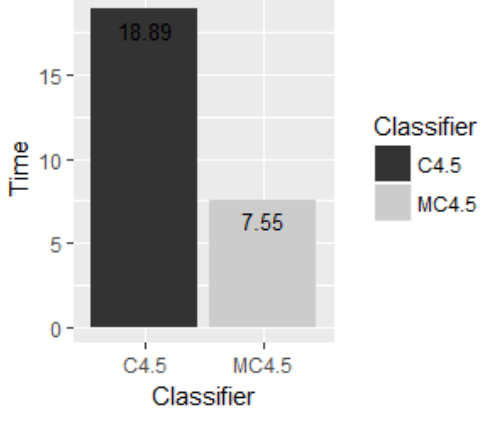


Figure 3 Modelling time



Note: C4.5 and MC4.5.

In addition to classification accuracy and size of trees, we compare the training times of the algorithms. It turns out that there are such large differences in execution times. Figure 3 provides strong evidence of significant differences between C4.5 and MC4.5 modelling time. At a glance, we can explicate that C4.5 algorithm is more complex than MC4.5. Our proposed algorithm takes less processing time than original C4.5 that is due to the new categorisation way proposed in this paper. The main advantage of this way is that it avoids sorting data and computing information gain for each distinct numeric value to deal with.

Table 4 Results obtained by the Wilcoxon test

VS	R^+	R^-	*Exact P-value*	*Confidence*
C4.5 vs. MC4.5[1]	422.0	43.0	2.366E–5	0.95028
C4.5 vs. MC4.5[2]	234.5	230.5	≥ 0.2	0.95028
C4.5 vs. MC4.5[3]	85.0	380.0	≥ 0.2	0.95028

Another interesting remark can be made about results in Table 3, the proposed algorithm decreases the tree accuracy in six cases. There is three datasets where the reduction in model accuracy is due to a remarkable decrease in the model complexity (lines 9, 17 and 30). The other subset of datasets (lines 21, 23 and 24), contains cases where there is a huge difference in the classes distribution (class-unbalanced data).

Results in Table 3 are assessed by means of Wilcoxon test (Table 4). As we can see, there is a significant difference in favour of the MC4.5 algorithm. These results support the conclusion that MC4.5 has drastically improved the model accuracy and the execution time. Also, it provides the same tree size as C4.5 algorithm.

Finally, the new MC4.5 algorithm uses only continuous attributes with high generalisation performances. Using statistical mean as a threshold value, it penalises continuous attributes with high specification level to avoid overfitting problems and to reduce the decision tree size. Also by improving the generalisation abilities in tree nodes, MC4.5 yields more accurate models than C4.5.

6 Conclusions

To conclude, MC4.5 yields the best tree accuracy for most of the used datasets. As discussed previously, MC4.5 gives a general description of data to avoid overfitting. We conclude that MC4.5 brings improvements in terms of accuracy and model complexity. In most cases, C4.5's performance is less than MC4.5 ones, in that it added more time to build the final model. Pruning technique should be adapted to the new binarisation method proposed on MC4.5 to guarantee more accurate and less complex decision trees.

References

Agrawal, G.L. and Gupta, H. (2013) 'Optimization of C4.5 decision tree algorithm for data mining application', *International Journal of Emerging Technology and Advanced Engineering*, Vol. 3, No. 3, pp.341–345.

Alcalá, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L. and Herrera, F. (2010) 'KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework', *Journal of Multiple-Valued Logic and Soft Computing*, Vol. 17, Nos. 2–3, pp.255–287.

Behera, H.S. and Mohapatra, D.P. (Eds.) (2015) 'Computational intelligence in data mining – Volume 1', *Proceedings of the International Conference on CIDM*, Vol. 410, Springer, 5–6 December.

Breiman, L. (1984) *Classification and Regression Trees*, Routledge, New York.

Cherfi, A., Nouira, K. and Ferchichi, A. (2018) 'Very fast C4.5 decision tree algorithm', *Applied Artificial Intelligence*, Vol. 32, No. 2, pp.119–137.

Chong, W.M., Goh, C.L., Bau, Y.T. and Lee, K.C. (2014) 'Fast numerical threshold search algorithm for C4.5', *2014 IIAI 3rd International Conference on Advanced Applied Informatics (IIAIAAI)*, IEEE, pp.930–935.

Dimitoglou, G., Adams, J.A. and Jim, C.M. (2012) *Comparison of the C4.5 and a Naïve Bayes Classifier for the Prediction of Lung Cancer Survivability*, arXiv preprint arXiv: 1206.1121.

Elomaa, T. (2014) 'In defense of C4.5: notes on learning one-level decision trees', *ML-94*, Vol. 254, p.62.

Garcia, S., Luengo, J., Sáez, J.A., Lopez, V. and Herrera, F. (2013) 'A survey of discretization techniques: taxonomy and empirical analysis in supervised learning', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 25, No. 4, pp.734–750.

García-Laencina, P.J., Abreu, P.H., Abreu, M.H. and Afonoso, N. (2015) 'Missing data imputation on the 5-year survival prediction of breast cancer patients with unknown discrete values', *Computers in Biology and Medicine*, Vol. 59, No. 2015, pp.125–133.

Github R Repository (2018) *Discretization* [online] https://github.com/Anis-cherfi/Discretization (accessed 10 July 2018).

Grzymala-Busse, J.W. and Mroczek, T. (2015) 'A comparison of two approaches to discretization: multiple scanning and C4.5', *in International Conference on Pattern Recognition and Machine Intelligence*, Springer International Publishing, pp.44–53.

Hothorn, T., Hornik, K. and Zeileis, A. (2006) 'Unbiased recursive partitioning: a conditional inference framework', *Journal of Computational and Graphical Statistics*, Vol. 15, No. 3, pp.651–674.

Hssina, B., Merbouha, A., Ezzikouri, H. and Erritali, M. (2014) 'A comparative study of decision tree ID3 and C4.5', *International Journal of Advanced Computer Science and Applications*, Vol. 4, No. 2, pp.13–19.

Kass, G.V. (1980) 'An exploratory technique for investigating large quantities of categorical data', *Applied Statistics*, Vol. 29, No. 2, pp.119–127.

Kerber, R. (1992) 'ChiMerge: discretization of numeric attributes', *in Proceedings of the tenth national conference on Artificial intelligence*, AAAI Press, July, pp.123–128.

Kotsiantis, S.B. (2013) 'Decision trees: a recent overview', *Artificial Intelligence Review*, Vol. 39, No. 4, pp.261–283.

Lim, T.S., Loh, W.Y. and Shih, Y.S. (2000) 'A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms', *Machine Learning*, Vol. 40, No. 3, pp.203–228.

Liu, H. and Setiono, R. (1995) 'Chi2: feature selection and discretization of numeric attributes', *in Proceedings of Seventh International Conference on Tools with Artificial Intelligence, 1995*, IEEE, pp.388–391.

Lu, Z., Wu, X. and Bongard, J.C. (2015) 'Active learning through adaptive heterogeneous ensembling', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 27, No. 2, pp.368–381.

Patel, N. and Singh, D. (2015) 'An algorithm to construct decision tree for machine learning based on similarity factor', *International Journal of Computer Applications*, Vol. 111, No. 10, pp.22–26.

Perner, P. (2015) 'Decision tree induction methods and their application to big data', in *Modeling and Processing for Next-Generation Big-Data Technologies*, pp.57–88, Springer, Cham.

Quinlan, J.R. (1986) 'Induction of decision trees', *Machine Learning*, Vol. 1, No. 1, pp.81–106.

Quinlan, J.R. (1996) 'Improved use of continuous attributes in C4.5', *Journal of Artificial Intelligence Research*, Vol. 4, No. 1996, pp.77–90.

Quinlan, J.R. (2014) *C4.5: Programs for Machine Learning*, Elsevier.

Ruggieri, S. (2002) 'Efficient C4.5 [classification algorithm]', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 14, No. 2, pp.438–444.

Saqib, F., Dutta, A., Plusquellic, J., Ortiz, P. and Pattichis, M.S. (2015) 'Pipelined decision tree classification accelerator implementation in FPGA (DT-CAIF)', *IEEE Transactions on Computers*, Vol. 64, No. 1, pp.280–285.

Sumam, M.I., Sudheep, E.M. and Joseph, A. (2013) *A Novel Decision Tree Algorithm for Numeric Datasets – C4.5*Stat*, Recent Science Publications.

UCI (2016) *UCI Machine Learning Repository* [online] https://archive.ics.uci.edu/ml/datasets.html (accessed 15 July 2016).

Wang, R., Kwong, S., Wang, X.Z. and Jiang, Q. (2015) 'Segment based decision tree induction with continuous valued attributes', *IEEE Transactions on Cybernetics*, Vol. 45, No. 7, pp.1262–1275.

Wilcoxon, F. (1992) 'Individual comparisons by ranking methods', in *Breakthroughs in Statistics*, pp.196–202, Springer, New York.

Witten, I.H., Frank, E., Hall, M.A. and Pal, C.J. (2016) *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann.

Wu, X., Kumar, V., Quinlan, J.R., Ghosh, J., Yang, Q., Motoda, H., Zhou, Z.H. et al. (2008) 'Top 10 algorithms in data mining', *Knowledge and Information Systems*, Vol. 14, No. 1, pp.1–37.

Yang, Y. and Chen, W. (2016) 'Taiga: performance optimization of the C4.5 decision tree construction algorithm', *Tsinghua Science and Technology*, Vol. 21, No. 4, pp.415–425.

Zighed, D.A., Rabaséda, S. and Rakotomalala, R. (1998) 'FUSINTER: a method for discretization of continuous attributes', *International Journal of Uncertainty, Fuzziness and Knowledge-based Systems*, Vol. 6, No. 3, pp.307–326.