Vol. 9, No. 2, 2022, Pages 86-112

JIEMS
Journal of Industrial Engineering and Management Studies

journal homepage: www.jiems.icms.ac.ir

Machine learning decision tree based on regression in data mining to extract more knowledge

Zahra Jiryaei Sharahi *[1], Yahia Zare Mehrjerdi[1], Mohammad Saleh Owlia[1], Masoud Abessi[1]

[1] Department of Industrial Engineering, College of Engineering, University of Yazd, Yazd, Iran.

Received: Jan 2022-31 / Revised: Jul 2022-04/ Accepted: Aug 2022-09

Abstract
In a data-driven decision-making process, there are various types of data that should be thoroughly processed and analyzed. Data mining is a well-recognized method to obtain such information by analyzing data and transforming it into actionable insights for further use. Among the various data mining techniques such as classification, clustering, and association rules, this research focused on classification techniques and presented an innovative regression-based learning approach in the decision tree (DT) models. DT algorithms are easy-to-understood and can work with different data types including continuous, discrete, and non-numerical. Despite a large number of existing studies, which attempt to enhance the performance of the DT models, there is still a gap in accurately extracting knowledge from databases. In this research, this issue is addressed by exploiting regression and coefficient of determination (R2) methods in a DT. The proposed tree provides new insights in the following aspects: split criterion, handling continuous and discrete variables, labeling leaf node, pruning process by stopping criteria and tree evaluation. The superiority of the proposed algorithm is demonstrated using a real-world hospital database and a comparison with existing approaches is provided. The results showed that the proposed algorithm outperforms the existing methods in terms of higher accuracy and lower complexity.

Keywords: Data mining; classification; decision tree; split criterion; R square.

Paper Type: Original Research

1. Introduction

Data mining is the process of automatically discovering nontrivial, previously undiscovered, and possibly beneficial patterns in databases. Research has shown that data is expected to double every three years. Thus, data mining has become an important tool to transform such data into information. The datasets in data mining applications are often large, so new classification techniques have been developed to deal with millions of objects, each of them possessing perhaps dozens or even hundreds of attributes. Hence, classifying such datasets becomes an important concern in data mining (Wang et al., 2008). Classification is the process of automatically categorizing an object based on its attributes into one of several pre-defined categories. Classification is also known as supervised learning (Kotsiantis et al. 2007) in which a given set of data records is divided into training and test data sets. Neural Networks (Lippmann, 1987), Logistic Regression (Khoshgoftaar et al. 2000), and DTs (Salzberg, 1994) have been used as classification algorithms, to name a few. DT, as the most popular classification algorithm proposed in the data mining, provides a readily comprehensible modeling technique that simplifies the classification process.
In data mining, there are various methods for extracting knowledge from a large quantity of data. However, the primary objective of this study is to propose a new method that increases the amount of knowledge extracted from large databases. Also, a review of the literature on the topic of the DTs indicates that this technique needs to be further developed in the context of applying the concept of regression in the construction of branches, as well as the application of its rules in identifying predetermined patterns and new patterns of fraud and abuse especially in the field of health. In a nutshell, the following are the contributions of the current research:

*Corresponding Author: z.jiryaei@stu.yazd.ac.ir

DOI: 10.22116/jiems.2022.327172.1474

- Proposing a R2 based feature selection technique that is suitable for distinct and as well as continuous variables.

- Discretization of continuous variables using an efficient method.

- Comparison of our proposed model with existing models on a hospital dataset.

- A different approach to stopping tree growth and managing nodes.

Rest of the paper is organized as follows: in Section 2 various researches done in this area are investigated. Section 3 resents the problem definition and elaborately describes the proposed approach. Section 4 displays the outcome and analysis of the proposed approach. Section 5 contains the sensitivity analysis topic. Finally, conclusion and the future direction of the work are presented in section 6.

2. Related Work

2.1. Machine learning

Machine learning (ML) as a branch of computer science is a sub-field of Artificial Intelligence (AI) and has evolved from pattern recognition. ML is used to explore the structure of data and fit into the models, which can be understood and utilized by decision-makers (Wang et al., 2020; Cohen, 2021; Itani et al., 2020; Keswani et al., 2020; Shobha & Rangaswamy, 2018).
The fundamental purpose of ML is to build a model by taking input, and use a statistical formula to predict output when new data enters the system. ML examines sample data to find patterns and builds decision rules to create a predictive model that can be used to predict future data. These predictive models are capable enough of self-learning with minimal human intervention and take a decision based on certain circumstances. In most cases, large datasets are required for optimal learning. ML can broadly categorize into supervised, unsupervised and semi-supervised learning containing wide sets of algorithms in each category (Tao et al., 2021; Wang et al., 2020).
In supervised learning, the training set consists of data, which contains the target class. A supervised learning model has two main tasks to be performed, classification and regression. Classification is the prediction of the nominal class label while the regression is the prediction of the numeric value of the class label. The supervised learning algorithms covered include DTs, linear regression, logistic regression, ensemble of classifiers including random forest and gradient boosted trees, neural networks, support vector machines (SVM), k-nearest neighbor, naive Bayes, and Bayesian logistic regression. Unsupervised learning does not use target class, and is based upon clustering algorithms of various sorts. It is basically applied to extract a hidden pattern from the dataset when the dataset is not having labels. Semi-supervised Learning is the combination of supervised and unsupervised learning and is used when a smaller number of labeled data is identified for a particular application. The goal of semi-supervised learning is to classify some of the unlabeled data using the labeled information set (Le & Clarke, 2018; Wang et al., 2018; Sies & Van Mechelen, 2020;).

2.2. Data mining

The world is data-rich but information-poor. Data mining is searching knowledge in data and for interesting prototypes, (Sarker, 2018). Data mining is regarded as an emerging technology that has made radical change in the information world. The term "data mining" (often referred to as knowledge discovery) refers to the method of analyzing data from different perspectives and synthesizing it into valuable information. Thus, data mining includes the main functional elements that help to transform data into data warehouse, manage data in multidimensional database, facilitate data access for professionals or experts, analyze data using applied tools and techniques, and present data in ways that make sense to provide information (Panhalkar & Doye, 2021). Depending on the needs of the user, different data mining techniques are used, such as classification, clustering, regression, summarization, association and anomaly detection (Lu et al., 2015; Li et al., 2015).

2.3. Classification

Classification is the most commonly applied data mining technique, which employs a set of pre-classified examples to develop a model that can classify the population of records at large efficiently (Barsacchi et al., 2020; Wang et al., 2018). A classifier is generated by learning function over the training set. Then it performs on a new example in turn to predict the corresponding class label.

There are many classification methods including DTs (Kotsiantis, 2013), neural networks (Abpeikar et al., 2020), support vector machines (Baloochian & Ghaffary, 2019), Naive Bayes (Farid et al., 2014), Bayesian networks (Quadrianto & Ghahramani, 2015), k-nearest neighbor (Bobadilla et al., 2013), rough set theory (Zhang & Dai, 2015). Among them, the DT is found to be a simple, expressive, robust and efficient classifier (Kappelhof et al., 2021; Sarker et al., 2020), and it has been widely used in knowledge discovery and pattern recognition fields.

2.4. Decision tree

The DT algorithm is one of the most popular procedures which creates a knowledge representation structure (Pilz et al., 2018). By using this method, cases divide into separate groups or the values of a target variable can be predicted by the values of predictor variables (Ginde et al., 2009). A DT includes a single root node, some internal nodes and several leaf nodes. The root node is the start of DT. The internal nodes connect the root node and leaf nodes (Meng et al., 2020). DTs are built by recursively splitting a node into two or more sub-nodes to produce homogeneity for the target variable of each resulting sub-node (Wang et al. 2020). Therefore, a pathway from the root node to the leaf node corresponds to a series of attributes (questions) with their values (responses) (Piramuthu, 2008; Han et al., 2011).
The superiority of DT methods as the most popular data mining techniques is operating without hypothesis on the system, handling efficiently numerical and categorical, and performing implicitly variable feature selection. Also, the generalization, the effectiveness, the robustness and the data bruit resisting are among the prediction model features (Barsacchi et al., 2020) which leads to choose this method to perform improvement actions in knowledge extraction.
The primary part of the DT algorithms is the selection of the division feature. Splitting is a key process of separating nodes into one or more sub-nodes. In other words, the growing of the tree is based on splitting criteria which is applied recursively on the data set to find the accurate model that can provide the best decision to the instance's classification (Benkercha & Moulahoum, 2018). There are plenty of approaches that a DT may be constituted of a dataset, based upon which attributes to pick for every node, and what situations are to be used for splitting at that node (Fayyad & Irani, 1992; Quinlan, 1996; Saroj & Anand, 2021). A suitable attribute is the one, which splits the data such that each successor node is as pure as possible. Information gain, Gini index, and gain ratio are the popular methods by which the node attribute of a DT is decided (Shobha & Rangaswamy, 2018). Although there are many specific DT algorithms, the CART, C5.0, CHAID, Exhaustive CHAID, and QUEST algorithms are the well-known algorithm and the most commonly used ones (Kotsiantis, 2013; Luo et al., 2021). These algorithms are used to evaluate the performance of the proposed model and each of them is briefly described as in next sections.

2.5. CART (classification and regression tree)

CART (Breiman et al., 2017) was ranked in the top 10 algorithms for data mining (Wu et al., 2008) and hence we regarded CART and its performance as a satisfactory reflection of the expected performance of the DT algorithms that are currently often used in practice (Kappelhof et al., 2021). CART algorithm uses the Gini index as the impurity metric to determine attributes for classifying samples and constructs binary trees in a top-down manner (Rutkowski et al., 2014). This algorithm has been used in many studies (Handley et al., 2014; Bar-Hen et al., 2015). The main drawback of this algorithm is that it is not suitable to handle continuous attributes (Meng et al., 2020).

2.6. C5.0

C5.0 is an updated version of C4.5 (Quinlan, 1986). It extends the C4.5 classification algorithms described in (Salzberg, 1994). Based on Kuhn and Johnson (2013) the model has been advanced in terms of speed, memory usage, the size of the DT, boosting (that improves the accuracy of the trees), and weighing (that allows the user to weigh different cases and misclassification types). A comparison study of the top 10 algorithms of data mining (Wu et al., 2008) identified C4.5 as the most influential data mining algorithm.

C5.0 DT model is essentially a tree involving a set of decision nodes, among which the root and each internal node are labeled with a question (Pradhan, 2013). The arcs descend from each root node to the leaf nodes, where a solution to the related question is proposed. A split is created at each node by making a binary decision, which separates a class or more classes from the global dataset. The C5.0 is a kind of algorithm that calculates the best splits based on information gain ratio (IGR). The IGR is considered as a probability-based measure used to calculate the degree of uncertainty reduction. Generally, the DT grows down by calculating the split with the biggest IGR until the best solution is available (Rajeswari & Suthendran, 2019; Guo et al., 2021). The C5.0 algorithm can overfitting easily lead to form an excessive tree structure. Overfitting is a situation in which the resulting tree consists of the noise and random errors from the training set causing an excellent result for training and test set, but the practical application will produce a large error (Yu et al., 2018).

2.7. CHAID (Chi-Squared Automatic Interaction Detector)

CHAID proposed by Kass in 1980, CHAID decision trees support multiple splits, allowing each node to have multiple child nodes. Based on the type of target variable, to determine splitting rules and node formation, F-test is used when the dependent variable is continuous and the Chi-Squared test is used when the dependent variable is nominal. Each predictor at each split that provides the best prediction of the target variable is the one with the lowest p-value from the test (Bach et al., 2018). The most significant predictor of the CHAID decision tree would be the top node. Splitting continues until there are no significant relationships between the remaining predictors and the target variable (Perri, et al., 2012; Milanovic & Stamenkovic, 2016).

2.8. Exhaustive CHAID

The Exhaustive CHAID (Biggs et al., 1991), a modification to the basic CHAID algorithm, has the same splitting and stopping steps as CHAID but the merging step is more exhaustive than CHAID, by continuing to merge categories of the predictor variable until only two super categories are left. It then examines the series of merges for the predictor variable and finds the set of categories that offers the most powerful association with target variable. Thus, the Exhaustive CHAID can find the best split for each predictor variable (Sut and Simsek, 2011; Hothorn et al., 2006).

2.9. QUEST (quick unbiased and efficient statistical tree)

The Quick, Unbiased, Efficient Statistical Tree (QUEST) algorithm is a binary and single-variable tree algorithm for classification and data mining (Loh & Shih, 1997). It is similar to the CART algorithm (Breiman et al., 2017). However, there are some minor differences. For instance, QUEST uses an unbiased variable selection method, uses imputation to handle missing values instead of surrogate splits, and handles categorical variables in many categories. It deals with split selection and split-point selection, separately. The univariate split performs unbiased field selection, which means if all of the predictor fields are similarly informative concerning the target field; it chooses any of the predictor fields with equal probability. The QUEST uses chi-squared tests for unordered variables and analysis of variance (ANOVA) tests for ordered variables. While keeping the accuracy of estimation in the CART, the QUEST algorithm accelerates the process of making a classification tree (Chou, 2012).

2.10. Regression

Regression analysis is a predictive modelling technique for formulating the correlation between a set of dependent and independent variables. The purpose of the regression is to apply a mathematical function to the data that captures these changes. Linear regression as an essential regression algorithm attempts to model the relationship among two variables by fitting a linear equation to observed data (Yeturu, 2020; Gkioulekas & Papageorgiou, 2021). Unlike the linear regression model, which simplifies the relationships between the different variables, other regression models such as logistic, nonlinear can be named as the developed states of the linear regression model to deal with more complex scenarios. Logistic regression (LG) is almost similar to linear regression used for classification problem through assigning observations to a discrete set of classes (De Caigny et al., 2018; Rezapour et al., 2020; Saroj & Anand, 2021). To map different predictors to probabilities, sigmoid activation function can be used to transfer the data with any value to a value between 0 and 1. Dependent variable value '0' indicates non-occurrence of event and '1' indicates presence of the event
A suitable linear regression model offers valid statistical inferences on diverse applications with forecasting. The success of linear regression analysis lies in the adequacy of the fitted model in explaining the variations in the data set. A popular tool to determine the adequacy of the fitted model is the coefficient of determination and the adjusted version. The coefficient of determination is popularly known as R2. They are treated as summary measures for the goodness of fit of any linear regression model. The R2 is based on the proportion of variability of the study variable that can be explained through the knowledge of a given set of explanatory variables. It is the square of the multiple correlation coefficient between the study variable and all the explanatory variables present in the linear regression model (Sahani & Ghosh, 2021). However, to the best of our knowledge, no study has been reported to use R2 as a measure to create split in DT construction.

2.11. Healthcare dataset

In the literature, numerous studies have been undertaken to extract useful knowledge from various types of healthcare data. Healthcare data mining has proved to be useful in areas such as predictive medicine, patient relationship management, fraud and abuse detection, data analysis, healthcare monitoring and individual treatments' usefulness assessment (Chang & Chen, 2009; Pashaei et al., 2015). The advantages of integrating data mining into medical research are to improve diagnostic precision, cut costs and reduce human resources (Khajehei & Etemady, 2010). The efficacy of a fraud detection system largely depends on the efficiency of the used techniques and the quality of available databases (Bach et al., 2018).

In areas like healthcare, the model must understand the rationale behind the model output to use it when making a decision. For this reason, it is far not possible to apply black-box models in these scenarios, irrespective of their predictive performance. DTs as an interpretable model are examples of this kind of model (Sagi & Rokach, 2021). The focus of data mining work is the design and implementation of data mining algorithm. Therefore, enhancing the performance and accuracy of data mining algorithm has constantly been a remarkable issue. In this study, using a new criterion to identify splitting attributes in DTs construction is the best choice. To overcome the drawback of mentioned methods the contribution of this study are as follows:

- Here a R2 based technique is proposed to identify splitting variable in DTs.
- The proposed method works for any type of variable: discrete and continuous.
- Efficient discretization method is proposed for continuous variables.
- New approach is presented for converting nodes to leaves and stopping tree growth.
- Labeling leaf nodes is completed through a novel process.
- Extensive evaluation and analysis is performed on various models of the DT.
- The efficiency and effectiveness of the proposed method is investigated through an extensive sensitivity analysis.

3. Methodology

- The data classification process involves learning and classification. In learning, the training data are analyzed by classification algorithm. In classification, test data are used to estimate the accuracy of the classification rules (Baitharu & Pani, 2016). Hence, once the model was constructed utilizing the training samples, the test sub-dataset is used to assess the performance of the created model in representing/predicting the unseen data (Ghiasi et al., 2020). The procedure for building the proposed trees is explained in Figure 1. The main ideas of this procedure are explained in the following sections.

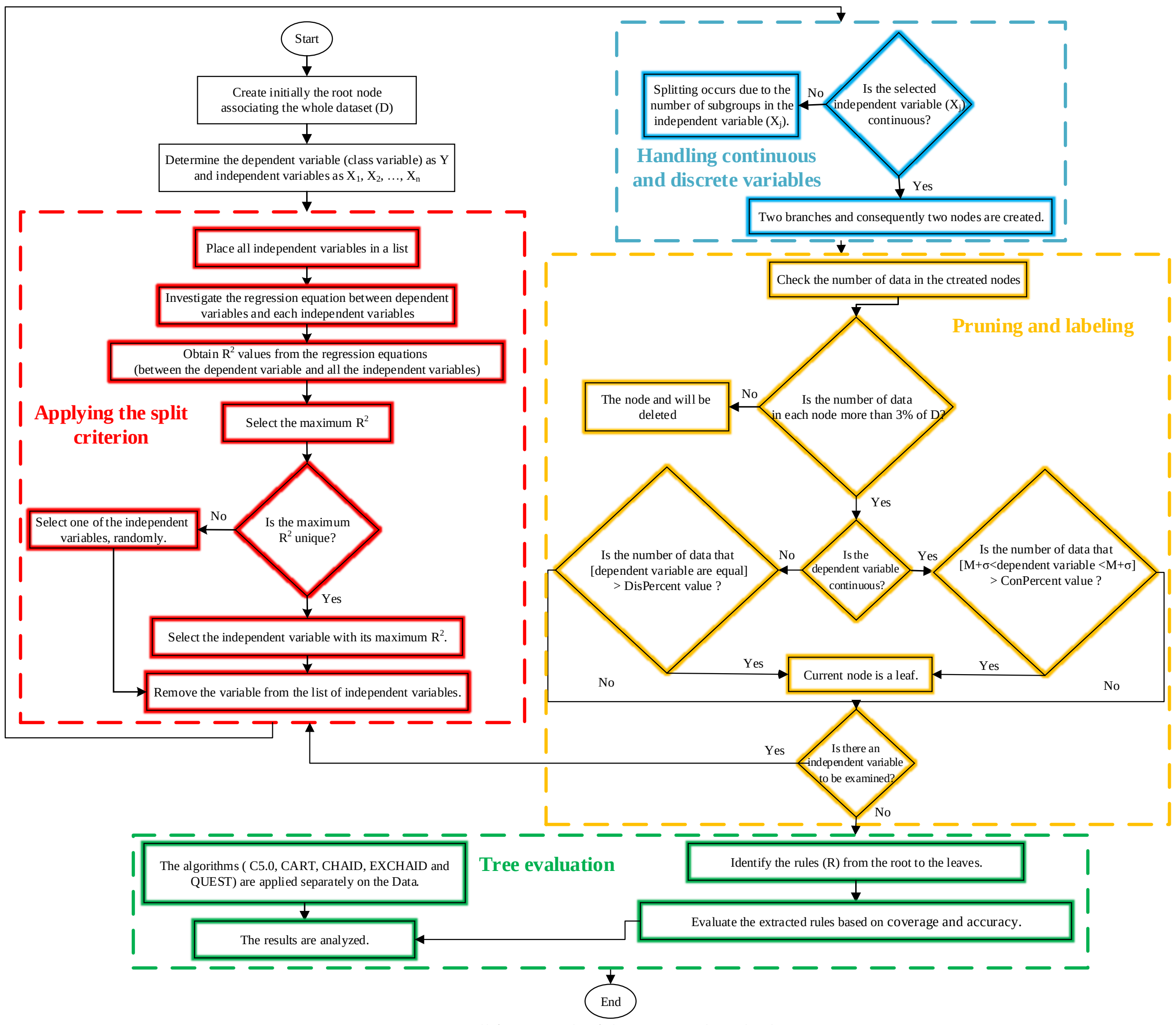
Start
Create initially the root node associating the whole dataset (D)
Determine the dependent variable (class variable) as Y and independent variables as X_1, X_2, …, X_n
Place all independent variables in a list
Investigate the regression equation between dependent variables and each independent variables
Obtain R^2 values from the regression equations (between the dependent variable and all the independent variables)
Applying the split criterion
Select the maximum R^2
Is the maximum R^2 unique?
No
Select one of the independent variables, randomly.
Yes
Select the independent variable with its maximum R^2.
Remove the variable from the list of independent variables.
Is the selected independent variable (X_j) continuous?
No
Splitting occurs due to the number of subgroups in the independent variable (X_j).
Handling continuous and discrete variables
Yes
Two branches and consequently two nodes are created.
Check the number of data in the ctreated nodes
Pruning and labeling
Is the number of data in each node more than 3% of D?
No
The node and will be deleted
Yes
Is the dependent variable continuous?
No
Is the number of data that [dependent variable are equal] > DisPercent value ?
Yes
Is the number of data that [M+σ<dependent variable <M+σ] > ConPercent value ?
Yes
Yes
Current node is a leaf.
No
No
Is there an independent variable to be examined?
Yes
No
The algorithms (C5.0, CART, CHAID, EXCHAID and QUEST) are applied separately on the Data.
Tree evaluation
Identify the rules (R) from the root to the leaves.
Evaluate the extracted rules based on coverage and accuracy.
The results are analyzed.
End

Figure 1. Overall framework of the proposed method

3.1. Split Criterion

Constructing a DT is often a recursive procedure, where it repeatedly optimizes a function and partitions the training data in the root and internal nodes until a termination condition is met. This function is usually referred as split criterion (Hamsa et al., 2016). Reviewing the literature indicates that several classical split criteria, including ID3 (Quinlan, 1986), C4.5 (Salzberg, 1994), classification and regression tree (CART) (Breiman et al., 2017) and

CHAID (Kass in 1980) have been proposed. Among the aforementioned criteria, ID3 and C4.5 are based on the information entropy, CHAID uses F-test and Chi-Squared test, while the CART adopts the Gini index. It should be indicated that some other split criteria are also introduced, but they are not classified as independent schemes (Mehta et al., 1996; Abellán & Moral, 2003; Lee, 2006; Mantas & Abellán, 2014; Wang et al., 2014; Mantas & Abellán, 2014b; Wu et al., 2016; Wang et al., 2017; Mantas et al., 2016; Wang et al., 2020; Chandra et al., 2010; Höppner, 2020).
In the proposed tree, R2-based criterion is employed to select the split variable at each branching node. In a similar way to some algorithms in DT construction, the variable with the highest R2 score will be selected. To do this, the regression equation between a dependent variable and each independent variable in the examined node is recognized. One of the following scenarios is likely to happen:

- The dependent variable is continuous and the independent variable is continuous: Calculate linear regression between the dependent variable and the independent variable and obtain R2.
- The dependent variable is continuous and the independent variable is discrete: Calculate the linear regression between the dependent variable and the independent variable. Consider the dummy variable for the independent variable and then obtain R2.
- The dependent variable is discrete and the independent variable is continuous: Calculate the logistic regression between the dependent variable and the independent variable by taking the dummy variable for the dependent variable and then obtain R2.
- The dependent variable is discrete and the independent variable is discrete: Calculate the logistic regression between the dependent variable and the independent variable by taking the dummy variable for the dependent and independent variables and then obtain R2.

After obtaining R2 values from the regression equations (between the dependent variable and all the independent variables), arrange them in a descending order to select the highest R2 value. Then choose the independent variable with a maximum of R2 as a split variable (if the variable is not unique, select one of the independent variables randomly).

3.2. Handling continuous and discrete variables

There have been many studies on how to deal with continuous variables (Fayyad & Irani, 1992; Quinlan, 1996; Wang et al., 2014; Tao et al., 2021). Here in node split step two scenarios will occur. When the independent variable (Xj) is continuous, two branches and consequently two nodes are created, where A = (max Xj - min Xj) and B = min Xj + (A/2). In sub-node 1 the values Xj >B and in sub-node 2 the values of Xj<B are located. When the independent variable (Xj) is discrete, splitting occurs due to the number of subgroups in the independent variable (Xj).

3.3. Labeling leaf node

Depending on whether the dependent variable is continuous or discrete, the examined node is converted to a leaf, which is selected for further investigation or deleted. In a continuous dependent variable, the mean (μ) and standard deviation (σ) of the dependent variable are calculated within the node data. If more than the ConPercent of the data has their dependent variable between +/- one standard deviation from the mean, the node will convert to a leaf. Also, the node is labeled with the mean of the dependent variable. Otherwise, depending on the number of instances in the node, that node will either be further examined or deleted. In a discrete dependent variable, if more than Dispercent of the instances is from a single class, the node will convert to a leaf and labeled with the most frequent class. Otherwise, depending on the number of instances in the node, that node will either be further examined or deleted.

3.4. Pruning Process by Stopping Criteria

A very important concern in building DTs would be deciding when the growth of the tree should be stopped. Also, improving the generalization ability of the DT, the pruning is needed to remove the over-subdivided leaf nodes caused by the effect of noise data, so that their parent nodes or higher nodes become leaf nodes. This process prevents overfitting of the training set (Oliver & Hand, 1996; Luo et al., 2021; Yu et al., 2018).

Pruning method is a search algorithm that reduces the size of DTs by removing sections of the tree that provide little power to classify instances (Manikandan et al., 2020). Pruning is one of the best mechanisms applied while constructing a DT (prepruning or forward pruning) or after the construction of a tree (postpruning or backward pruning) (Begon et al., 2017; Sim et al., 2017; Windeatt & Ardeshir, 2001). The main drawback of pruning is that it may increase the classification errors on the training data set, but it improves the accuracy of classification on unseen data points (Panhalkar & Doye, 2021). Since the pre-pruning approach is less computationally complex, it would probably be a better solution for dispersed data. In many articles, the advantages of pre-pruning have been confirmed based on data from various domains (Begon et al., 2017; Sim et al., 2017).

This study applies three pre-pruning methods; imposing a threshold on a measure (minimum number of instances) to stop tree growth, applying the Conpercent and Dispercent values and getting empty the list of independent variables. For minimum number of instances (3% of the total instances) pruning method, when a minimum number of objects in the leaves is increased, the size of the tree increases, while the accuracy of classification is generally unaffected (Przybyła-Kasperek, & Aning, 2021; Patel & Upadhyay, 2012). Based on the Conpercent and Dispercent values, a decision is made about when the leaf will become a node, remove or further review. On the other hand, the branching of a DT is also stopped when there are no more variable in the list of independent variables to select.

3.5. Tree evaluation

The purpose of building a DT is to predict the dependent variable based on the independent variables by learning decision rules (Myles et al., 2004; Pradhan, 2013). The paths from a root node to leaf nodes reveal the classification rules. The decision rules are generally of the form 'IF condition THEN conclusion' statements. The "IF" part of a rule is known as the rule antecedent or precondition. The "THEN" part is the rule consequent. In the rule antecedent, the condition consists of one or more variable tests that are logically ANDed. The rule's consequent contains a class prediction. If the condition in a rule antecedent holds true for a given set of data, the rule antecedent is satisfied and the rule covers the data (Witten et al., 2017; Meng et al., 2020; Wang et al., 2020; Khalili-Damghani et al., 2018; Yeo & Grant, 2018; Gkioulekas & Papageorgiou, 2021)

Once a tree has been constructed, performance evaluation is the most important task to be performed, for checking accuracies of learning methods, used to construct the tree. To evaluate the performance of the proposed method, various criteria used in previous research (specificity, accuracy, precision, recall, F-measure, Confusion Matrix) (Rajeswari & Suthendran, 2019; Kudła & Pawlak, 2018). In this study, rule assessment metrics such as coverage and accuracy with minor modification are applied. In the proposed tree, an extracted rule (R) is evaluated with the criteria of coverage and accuracy. Given a set of data (X) from a class labeled data set (D), let ncovers be the number of data covered by R (If the dependent variable (Y) is continuous, the value of dependent variable is in the interval of a standard deviation of the mean); ncorrects be the number of data correctly classified by R; |D| be the amount of data in D and K is the number of leaves. The coverage and accuracy of R can be defined as follows:

$$\text{Coverage (R)} = \text{ncovers}/|D| \tag{1}$$

$$\text{Accuracy (R)} = \text{ncorrect}/\ \text{ncovers} \tag{2}$$

A rule's coverage is the percentage of data that is covered by the rule (their variable values hold true for the rule's antecedent). For a rule's accuracy, it evaluates the data that is covered and percentage of them the rule can correctly classify. This description is shown in Table 1.

Table 1. Model evaluation on training data

Leaves	Rules	Rule antecedent (Precondition): IF --- & --- & ...			Rule consequent: THEN ---	Rule standard deviation *	Coverage	Accuracy
Leaf 1	Rule 1	Variable test1	Variable test2	Variable test...	Conclusion 1	$\partial 1$	C1	A1
Leaf 2	Rule 2	Variable test1	Variable test2	Variable test...	Conclusion 2	$\partial 2$	C2	A2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Leaf k	Rule k	Variable test1	Variable test2	Variable test...	Conclusion k	∂k	Ck	Ak

*** This column is calculated if the dependent variable (Y) is continuous.**

The model is evaluated using test data and for each test data, the dependent variable (Y) is predicted. Considering the total number of test data as Z, the model evaluation on the test data is shown in Table 2.

Table 2. Model evaluation on test data

Test data	Actual value of dependent variable	The rule number that covers the test data	Mean of dependent variable in the leaf	Rule's accuracy	If the dependent variable (Y) is discrete	If the dependent variable (Y) is continuous	
					Is the actual class equal to the predicted class? No=0 or Yes=1	standard deviation of the leaf	Is the value of dependent variable in the interval of a standard deviation of the mean? No=0 or Yes=1
test 1	AC1	Rule i	Conclusion rule i	Ai	0 or 1	∂i	0 or 1
test 2	AC2	Rule j	Conclusion rule j	Aj	0 or 1	∂j	0 or 1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
test z	ACz	Rule m	Conclusion rule m	Am	0 or 1	∂m	0 or 1

Finally, the accuracy of the model in the experimental data set is calculated as follows:

$$P_i = \begin{cases} 1 & if \quad conclusion \quad rule = AC_i \\ 0 & if \quad conclusion \quad rule \neq AC_i \end{cases} \tag{3}$$

$$Accuracy_i = \begin{cases} A_i & if \quad P_i = 1 \\ 1 - A_i & if \quad P_i = 0 \end{cases} \tag{4}$$

$$TotalAcuracy = \frac{\sum_{i=1}^{z} Accuracy_i}{z} \tag{5}$$

4. Experimental Results and Analysis

In this section, experiments are conducted to demonstrate the superiority of the proposed tree model. All experiments are run in MATLAB 9.2.0. For more details, the MATLAB code is given in Appendix. In this study, the dataset is divided into training and test sets in a ratio of approximately 5:1 (Delen et al., 2013; Tanyu et al., 2021).

4.1. Training phase

Data is collected from six health insurance organizations in 2020. In the training phase, 1000 samples with 15 variables were investigated. The brief description of the data is given in Table 3. In this research, the "diagnosis code" variable was considered as a dependent variable and the rest of the variables were considered as independent variables.

Table 3. Brief description of the data

Variable	Variable type	Description
Referral Month	discrete	January, February, March, April, May, June, July, August, September, October, November, December
Insurance Type	discrete	T, S, K, O, A, V
Hospitalization days	discrete	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Age	discrete	0-14, 14-21, 22-40, 41-60, 61-70, 71-100
Diagnosis Code	discrete	Z03.5, I51.9, I51.6, R69, I25.1, E66.8
Paraclinical tests	continuous	Min:0, Average:2713920, Max:2713920
Surgery	continuous	Min: 0, Average: 22602210 Max: 71139200
Radiology	continuous	Min: 0, Average: 16900 Max: 5589760
Drug	continuous	Min: 144980, Average: 3952923, Max:129358200

Other	continuous	Min: 40620, Average: 645380, Max: 6545697
Consumer goods	continuous	Min: 72258, Average: 36567329, Max: 139232238
Vascular interventions	continuous	Min:0, Average: 1260141, Max: 43245440
Hoteling	continuous	Min: 0, Average: 1705213, Max: 25650000
Visit	continuous	Min: 0, Average: 22743, Max: 2323200
All services	continuous	Min: 49887782, Average: 67063846, Max: 183682060

As presented above, among 54 diagnosis codes assigned to 1,000 instances, only the 6 mentioned diagnosis codes cover 895 data. Hence, in the processing phase, the number of data is reduced to this number. After running the model on the training data, a DT is created (Figure 2). It is observed that 7 levels are created which have a total of 49 nodes. Among the created nodes, there are 8 leaves, 14 nodes have been further examined based on variable selection and the remaining 27 nodes have been removed. Also, the description of the node color is shown in Table 4.

Table 4. The description of node color

Node color	Description
Green	The nodes turned into leaves
Red	The nodes have been deleted
White	The nodes examined further

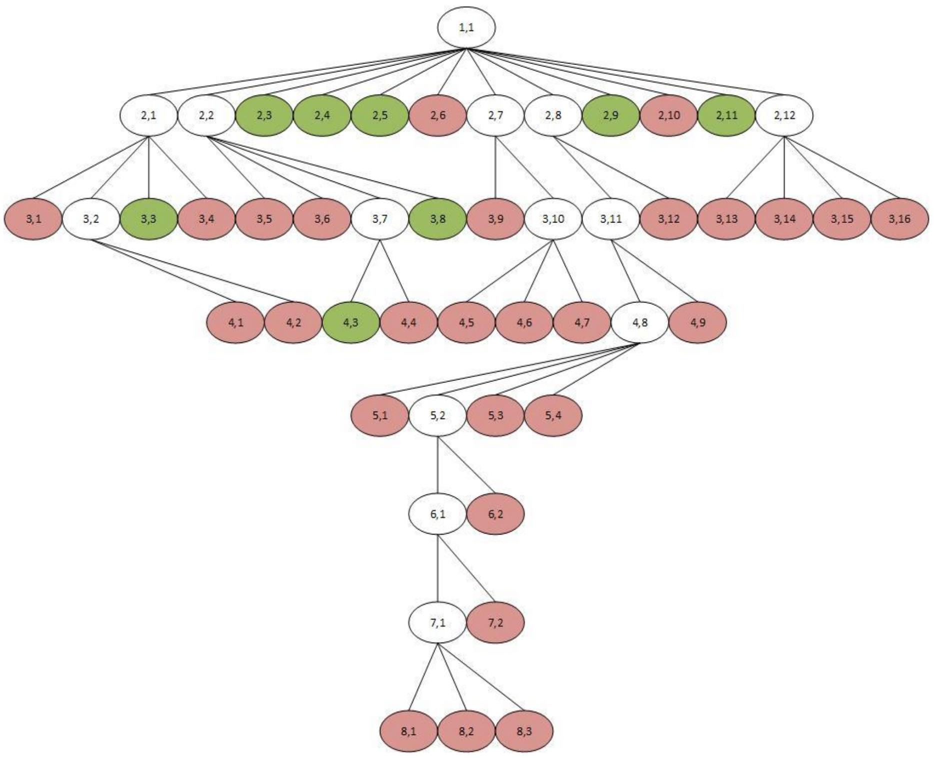

Figure. 2. The created tree by the implementation of the proposed model

Number of instances in each leaf are shown in Figure 3. Based on the pruning process by stopping criteria the nodes with less than 3% of the total instances have been deleted. Other nodes have also turned into leaves or branched from them.

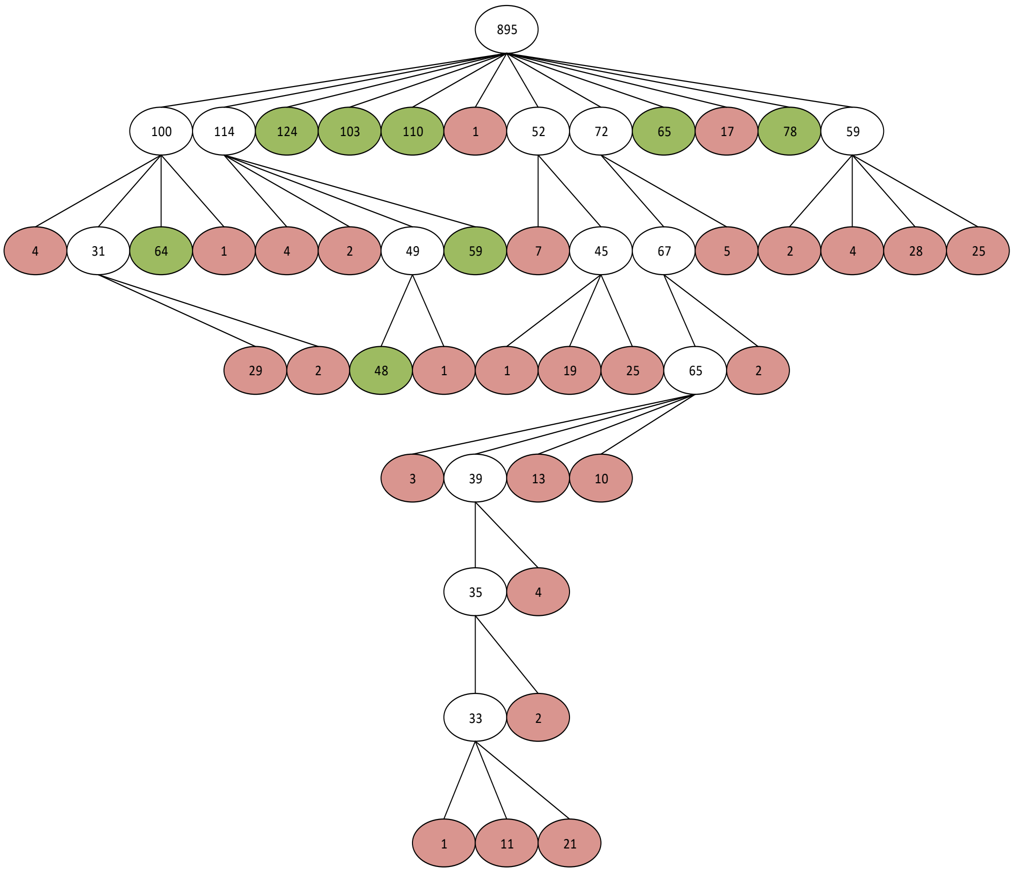

Figure 3. Number of instances in each node

The nodes examined further are shown in Figure 4. According to the split criterion, the nodes examined further with the selected independent variable and related values are shown in Figure 4. For example, at the root node, the independent variable "Referral Month" as the discrete variable has the highest R2 value, which makes it as the split variable. At the level one, nodes 1, 2, 7, 8, and 12 were qualified for further examination. In other words, the number of instances in them was more than 3% of the total data and the scattering of the dependent variable prevented them from turning into leaves. At the level two, nodes 2, 7, 10, and 11were qualified for further examination. Likewise, at the level three, node 8, at the level four, node 2, at the level five, node 1, and at the level six, node 1 were selected for further examination. Therefore, the depth of the constructed tree is 7.

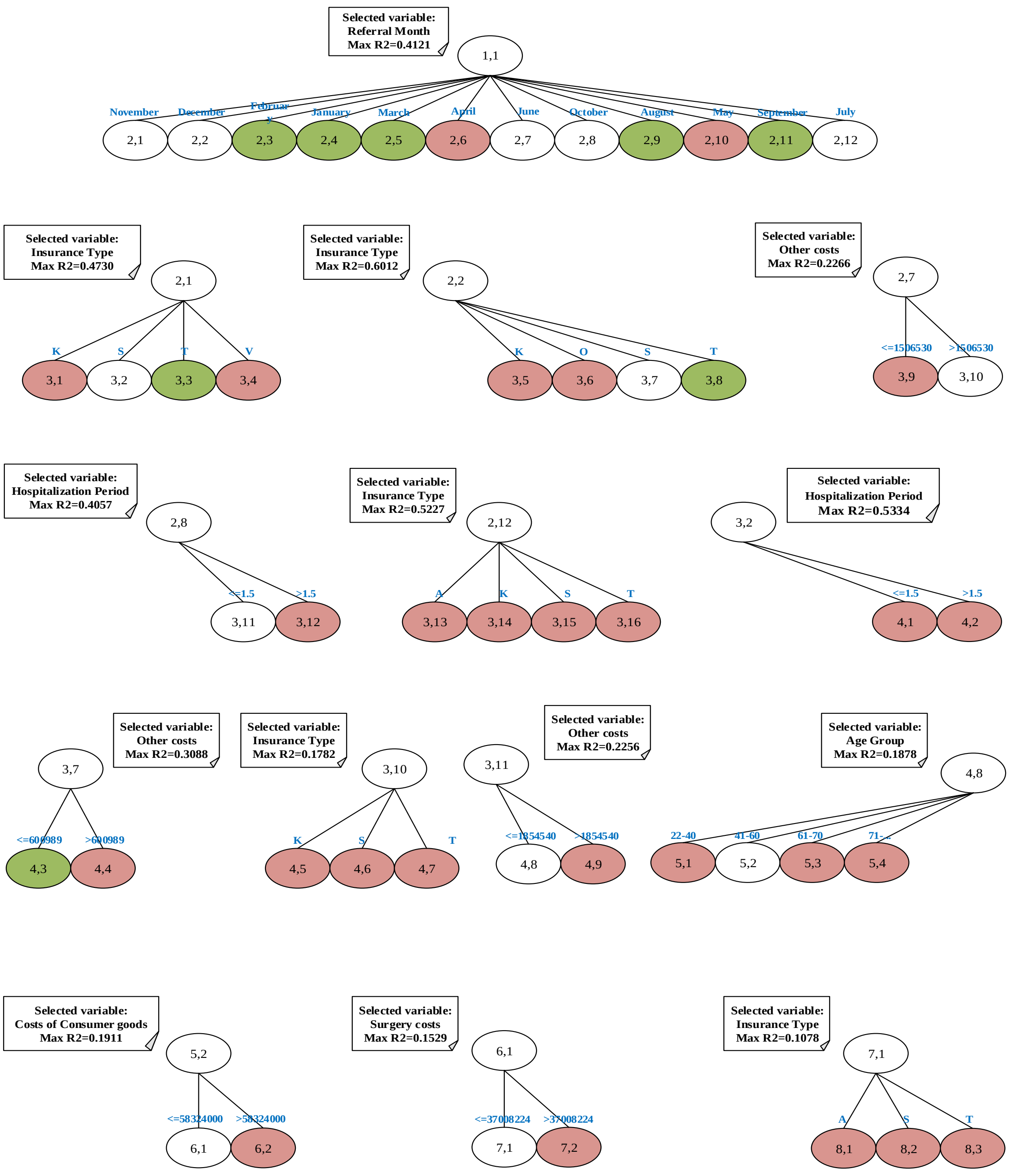
Selected variable: Referral Month Max R2=0.4121
1,1
November
December
February
January
March
April
June
October
August
May
September
July
2,1
2,2
2,3
2,4
2,5
2,6
2,7
2,8
2,9
2,10
2,11
2,12
Selected variable: Insurance Type Max R2=0.4730
K
S
T
V
3,1
3,2
3,3
3,4
Selected variable: Insurance Type Max R2=0.6012
K
O
S
T
3,5
3,6
3,7
3,8
Selected variable: Other costs Max R2=0.2266
<=1506530
>1506530
3,9
3,10
Selected variable: Hospitalization Period Max R2=0.4057
<=1.5
>1.5
3,11
3,12
Selected variable: Insurance Type Max R2=0.5227
A
K
S
T
3,13
3,14
3,15
3,16
Selected variable: Hospitalization Period Max R2=0.5334
3,2
<=1.5
>1.5
4,1
4,2
Selected variable: Other costs Max R2=0.3088
3,7
<=600989
>600989
4,3
4,4
Selected variable: Insurance Type Max R2=0.1782
3,10
K
S
T
4,5
4,6
4,7
Selected variable: Other costs Max R2=0.2256
3,11
<=1854540
>1854540
4,8
4,9
Selected variable: Age Group Max R2=0.1878
4,8
22-40
41-60
61-70
71-
5,1
5,2
5,3
5,4
Selected variable: Costs of Consumer goods Max R2=0.1911
5,2
<=58324000
>58324000
6,1
6,2
Selected variable: Surgery costs Max R2=0.1529
6,1
<=37008224
>37008224
7,1
7,2
Selected variable: Insurance Type Max R2=0.1078
7,1
A
S
T
8,1
8,2
8,3

Figure 4. The nodes examined in the created tree

In the created tree, eight nodes turned into leaves that are shown in Figure 5. As mentioned before, in nodes that have become leaves, the number of instances in them was more than 3% of the total data and the scattering of the dependent variable caused them to turn into leaves. With these conditions at level 1, five leaves, at level 2, two leaves, and at level 3, one leaf are identified.

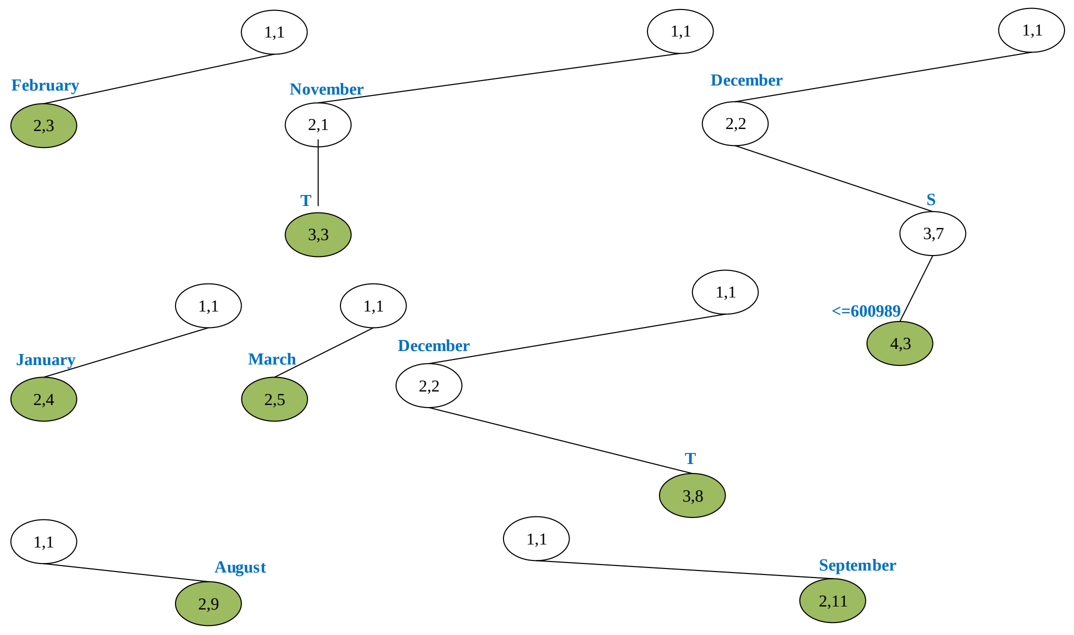

Figure 5. The nodes turned into leaves in the created tree

Each rule is created for each leaf from root to the leaf. Therefore, it is clear that the number of rules is equal to the number of leaves. The eight extracted rules with accuracy and coverage amounts are given in Table 5. As can be seen in the structure of the rules, “referral month”, “insurance type”, and “visit cost” have the greatest impact on the “diagnosis code” as dependent variables.

Table 5. Extracted rules

Leaves	Rules	Rule antecedent (Precondition)			Rule consequent (Diagnosis Code)	ncover	ncorrect	Coverage	Accuracy
Leaf 1	Rule 1	referral month "November"=	-	-	Z03.5	124	109	13.9%	87.9%
Leaf 2	Rule 2	insurance type = “T”	referral month "June"=	-	Z03.5	64	35	7.2%	54.7%
Leaf 3	Rule 3	<= visit cost 600989	insurance "S"type =	referral month "August"=	I51.9	48	24	5.4%	50.0%
Leaf 4	Rule 4	referral month "September"=	-	-	Z03.5	103	78	11.5%	75.7%
Leaf 5	Rule 5	referral month "December"=	-	-	Z03.5	110	102	12.3%	92.7%
Leaf 6	Rule 6	insurance type = “T”	referral month "August"=	-	Z03.5	59	44	6.6%	74.6%
Leaf 7	Rule 7	referral month "May"=	-	-	Z03.5	65	49	7.3%	75.4%
Leaf 8	Rule 8	referral month "June"=	-	-	Z03.5	78	72	8.7%	92.3%

As it can be seen in the above table, out of 651 instances in the leaves, 513 instances are classified correctly. Thus, according to the tree evaluation process, the accuracy of the proposed tree is 78.8%, which is a satisfactory value. In DT algorithms, the complexity of DT model is an important factor that needs to be considered. There is probably an enormous amount of leaf nodes in a fully-grown tree, and only a few training instances are in each leaf node. If the algorithm searches too long or concentrates an excessive amount on a few hard-to-learn instances, the problem of over fitting can occur. In addition, the performance of the DT might be also reduced when dealing with a noise dataset (Abellán et al., 2018; Czajkowski & Kretowski, 2019). On growing DTs, the tree complexity is measured by one of the following metrics: tree depth, total number of leaves, and number of variables used (Przybyła-Kasperek & Aning, 2021).

The effectiveness and efficiency of our proposed algorithm is evaluated against well-known algorithms such as C5.0, CART, CHAID, EXCHAID, and QUEST. The models are applied separately on the train data. Diagram of the models and their complexity are shown in Figure 6 and 7, respectively. Also, the accuracy of models is given in Table 6. As can be seen, the proposed model is in a satisfactory situation in terms of complexity compared to other models. In addition, the proposed model has the highest accuracy.

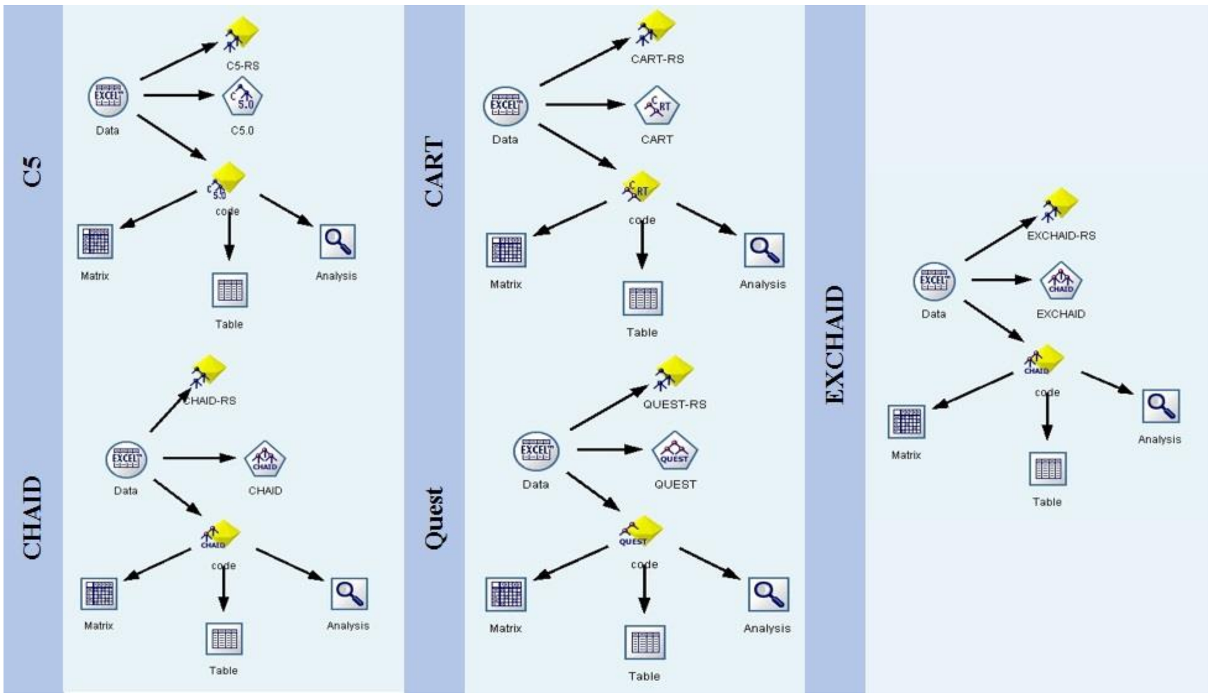

Figure 6. DT models in clementine environment

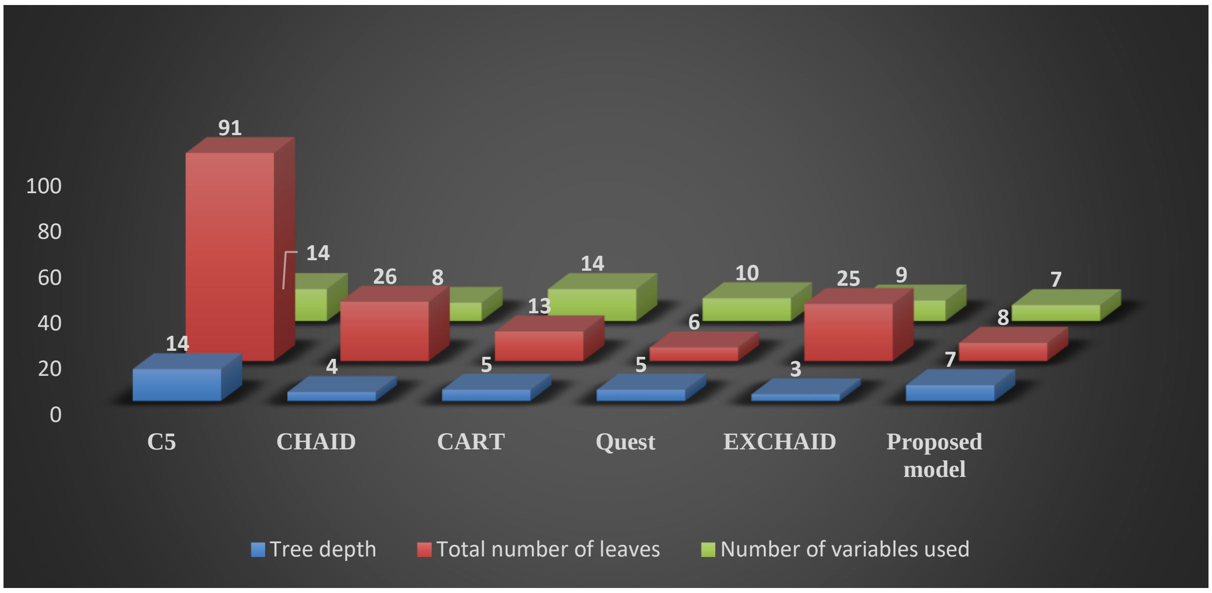

Figure 7. Comparison of model’s complexity

Table 6. Comparison models accuracy

C5	CHAID	CART	Quest	EXCHAID	Proposed model
77.80%	67.60%	66.40%	60.10%	66.40%	78.80%

Due to the high accuracy of the proposed model, it can be concluded that this model extracts useful knowledge (with more certainty and less error) from data. As a result, managers can trust the extracted rules and use them for fraud detection and decision-making.

4.2. Testing phase

In order to test the proposed algorithm, we apply it to 20% of our date (200 instances). The result of applying the proposed algorithm is shown in Table 7 as follows:

Table 7. Model evaluation in test data

Rules	Accuracy	Number of data covered by the rule	Number of data classified correctly by the rule i	Number of data not correctly classified by the rule i
1	87.90%	15	13	2
2	54.70%	14	14	0
3	50.00%	24	15	9
4	75.70%	33	14	19
5	92.70%	20	19	1
6	74.60%	15	4	11
7	75.40%	20	20	0
8	92.30%	20	20	0
Sum		161	119	42

The result shows that 39 records are not covered by any of the rules identified in the training phase (Rules 1 to 8). Among the 161 remaining records, the following results are obtained: based on Table 7 and the formulas 3, 4, and 5, the average accuracy of the proposed algorithm is 64.31% with regard to the test data covered by the rules (161 data).

5. Sensitivity analysis

Most studies have examined the cause and effect association between the target variable and the input variables (Davis, 1989). Measuring the importance of predictor variables is often recognized as sensitivity analysis, which is relative to the importance of each variable when making predictions (Delen, et al., 2013). Nonetheless, in this article sensitivity analysis is treated differently. To investigate the efficiency and effectiveness of our proposed algorithm, sensitivity analysis on the proposed algorithm is performed in three parts: the importance of Dispercent and Conpercent values, the type of dependent variable, and the amount of data.

5.1. Change in Dispercent and Conpercent values

The values of Dispercent and Conpercentare considered according to the type of dependent variable (discrete or continuous). In this study, the discrete dependent variable (detection code) is defined. Hence, after creating a split by an independent variable, in each created node if the amount of data with the same dependent variable is greater than Dispercent value, the node turns into a leaf. By default, in implementation of the proposed model, the Dispercent value was considered to be 0.5. Here, the Dispercent value is changed from 0.1 to 1 and the results are shown in Figure 8. As expected, the higher Dispercent value results in the lower number of leaves and the greater number of deleted nodes. In other words, with the increase of Dispercent value, the construction of the DT becomes stricter.

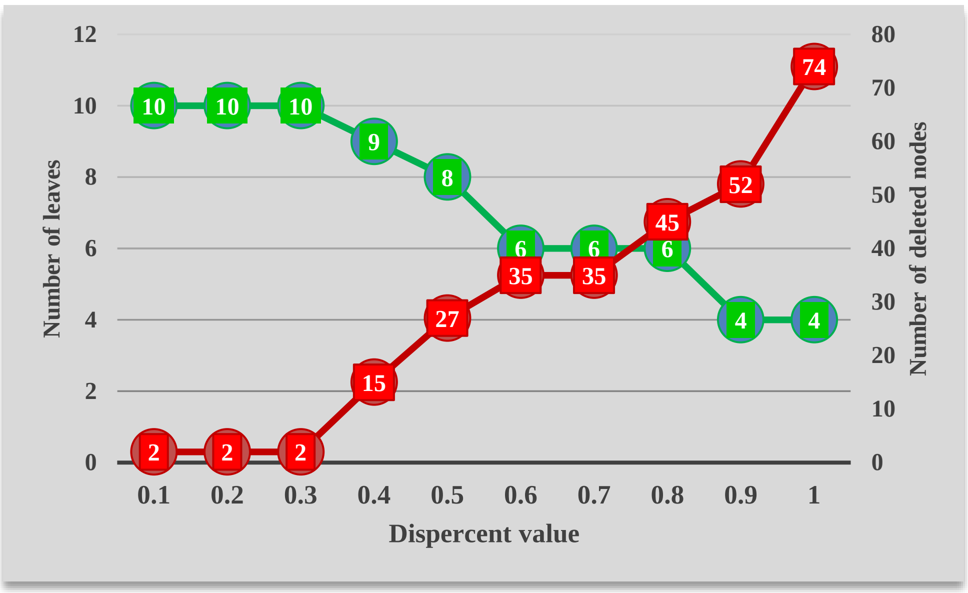

Figure 8. Number of leaves and deleted nodes in Dispersant value change conditions

As shown in the above figure, changing in the Dispercent value did not result in any significant change in the number of created leaves. Therefore, it can be said that the proposed model is almost robust under the conditions of changing Dispersant value. The mentioned mode is true for the case in which the dependent variable is continuous, with a Conpercent value.

5.2. Change in the type of dependent variable

In the first run of the proposed model, the discrete variable “diagnosis code” was considered as a dependent variable. In this part, to analyze the proposed model in different situations the continuous variable "drug costs" is considered as a dependent variable. A DT is created based on applying the model on the training data (1000 instances). As it can be seen in Figure 9, only one node (root node) in the tree has been examined and a branch has been created from it. The constructed tree has only one level with 54 nodes. Among the created nodes, there are 6 leaves and consequently 6 rules and 48 nodes have been removed. The number of instances in the 54 nodes created is shown in Figure 10. It has already been stated that according to the stop criteria, a node is deleted, examined or turned into a leaf.

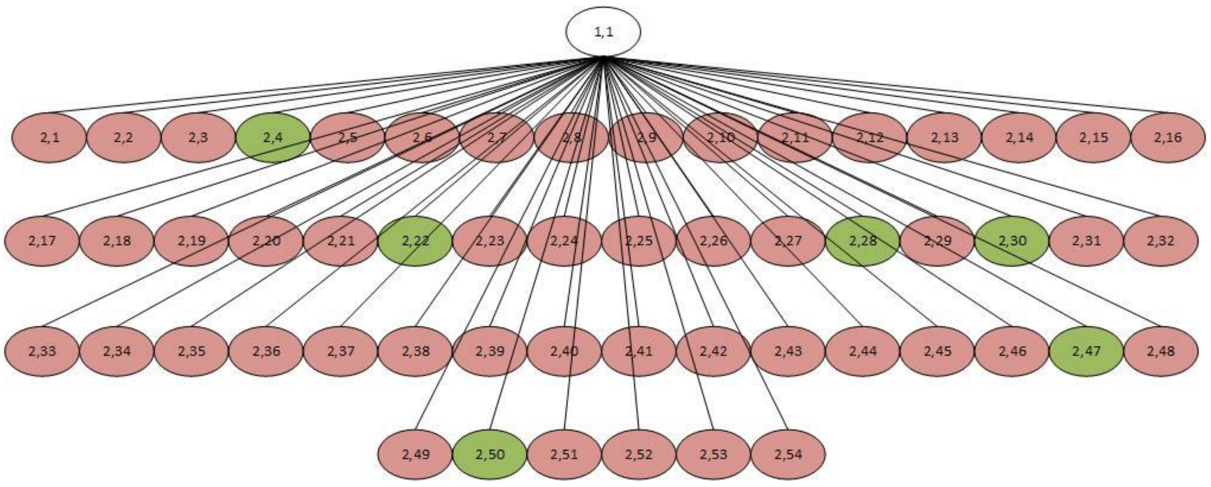

Figure 9. The created tree by the implementation of the proposed algorithm

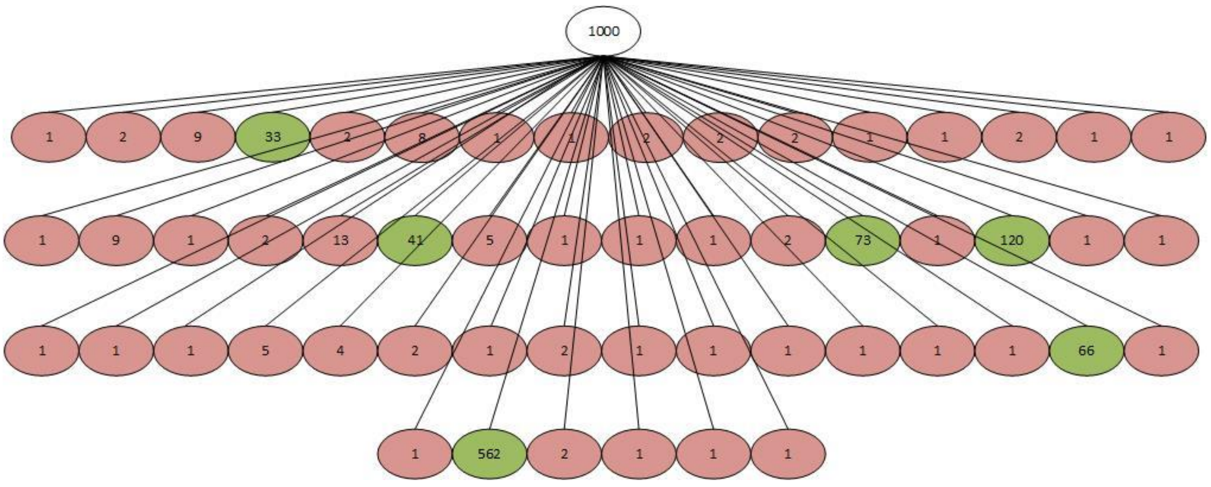

Figure 10. Number of instances in each node

In the root node, as the only node examined, the "Diagnosis code" variable with the highest value of R is selected as the split variable. Figure 11 shows the labeling of the created nodes based on selected independent variable.

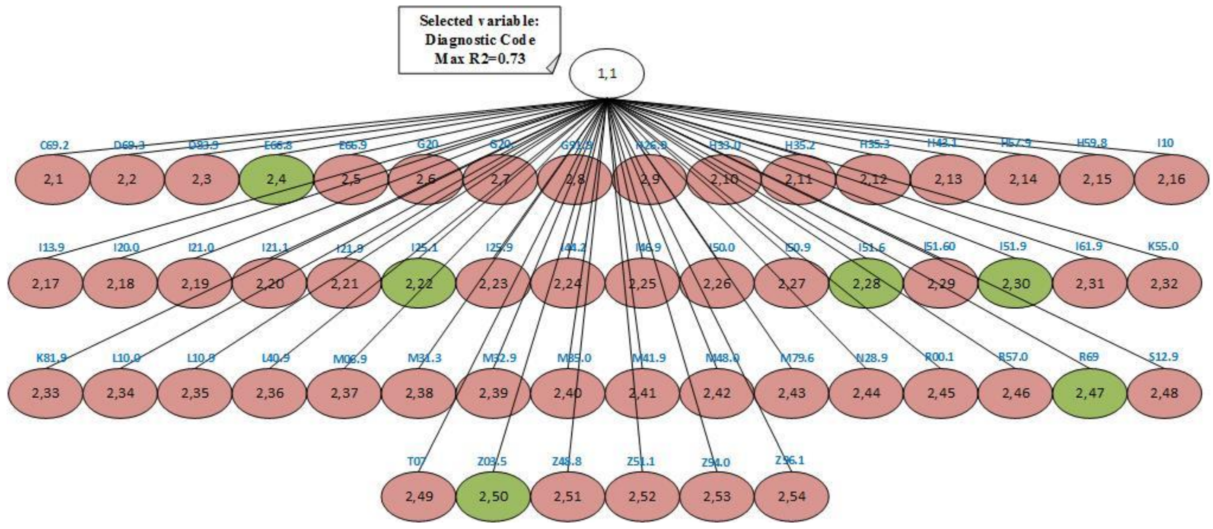

Figure 11. The nodes examined further in the created tree

The created tree has only one level and all 6 detected leaves in the tree are at this level. The labeling of the leaves can be clearly seen in Figure 12. Also, the rules assigned to each leaf are given in Table 8. According to the result, 743 out of 895 instances in the leaves are classified correctly. Therefore, the accuracy of the proposed tree is 83.02%, which is acceptable and it can be concluded that useful, pure and reliable knowledge is obtained. The extracted rules indicate that "diagnostic code" has the greatest impact on "drug cost" as a dependent variable, which is a perfectly valid and logical issue.

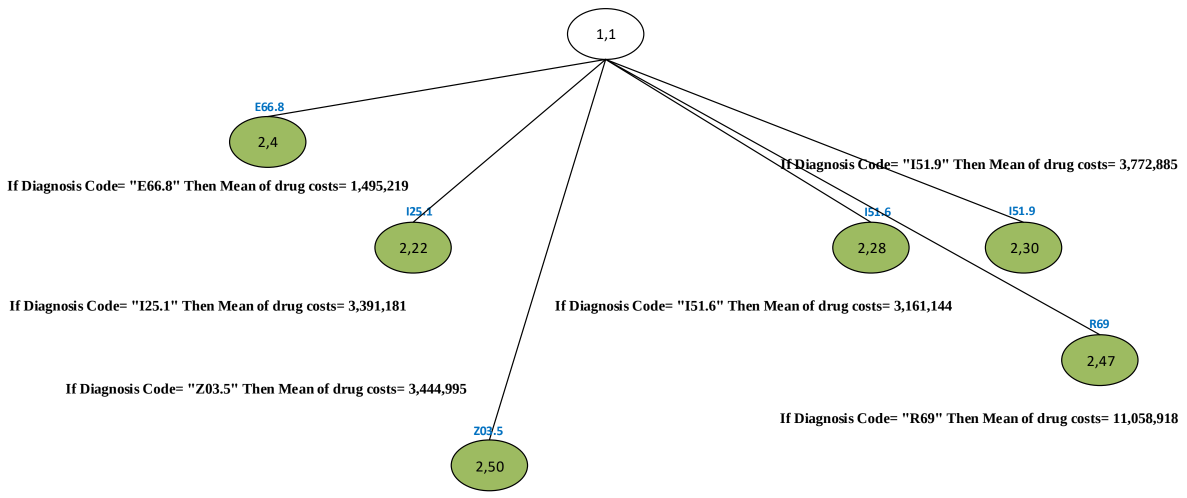
1,1
E66.8
2,4
If Diagnosis Code= "E66.8" Then Mean of drug costs= 1,495,219
I25.1
2,22
If Diagnosis Code= "I25.1" Then Mean of drug costs= 3,391,181
If Diagnosis Code= "Z03.5" Then Mean of drug costs= 3,444,995
Z03.5
2,50
If Diagnosis Code= "I51.9" Then Mean of drug costs= 3,772,885
I51.6
2,28
I51.9
2,30
If Diagnosis Code= "I51.6" Then Mean of drug costs= 3,161,144
R69
2,47
If Diagnosis Code= "R69" Then Mean of drug costs= 11,058,918

Figure 12. The nodes turned into leaves

Table 8. Extracted rules

Leaves	Rules	Rule antecedent (Precondition)	Rule consequent (Mean of drug costs)	ncover	ncorrect	Coverage	Accuracy
Leaf 1	Rule 1	Diagnosis Code= "E66.8"	1,495,219	33	27	3.30%	81.8%
Leaf 2	Rule 2	Diagnosis Code= "I25.1"	3,391,181	41	29	4.10%	70.7%
Leaf 3	Rule 3	Diagnosis Code= "I51.6"	3,161,144	73	60	7.30%	82.2%
Leaf 4	Rule 4	Diagnosis Code= "I51.9"	3,772,885	120	99	12.00%	82.5%
Leaf 5	Rule 5	Diagnosis Code= "R69"	11,058,918	66	59	6.60%	89.4%
Leaf 6	Rule 6	Diagnosis Code= "Z03.5"	3,444,995	562	469	56.20%	83.5%

Comparing the proposed model with other models (CART, CHAID and EXCHAID), we achieved that the proposed model has a proper performance and a slight degree of complexity. Therefore, the proposed model has the desired performance for both types of discrete and continuous dependent variables. The evidence is displayed in Figures 13 and 14, and Table 9.

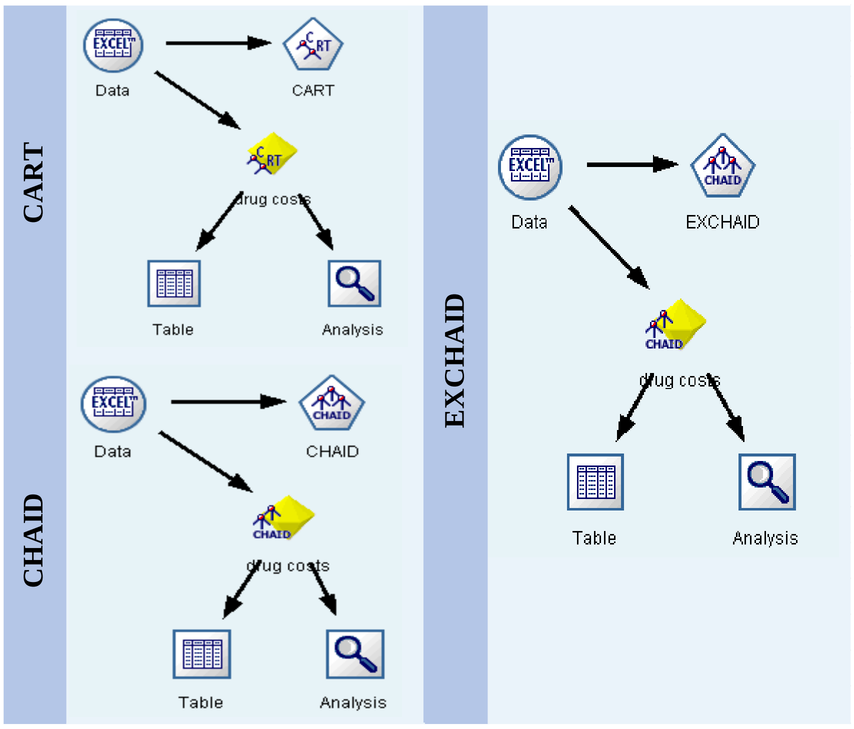

Figure 13. DT models in clementine environment

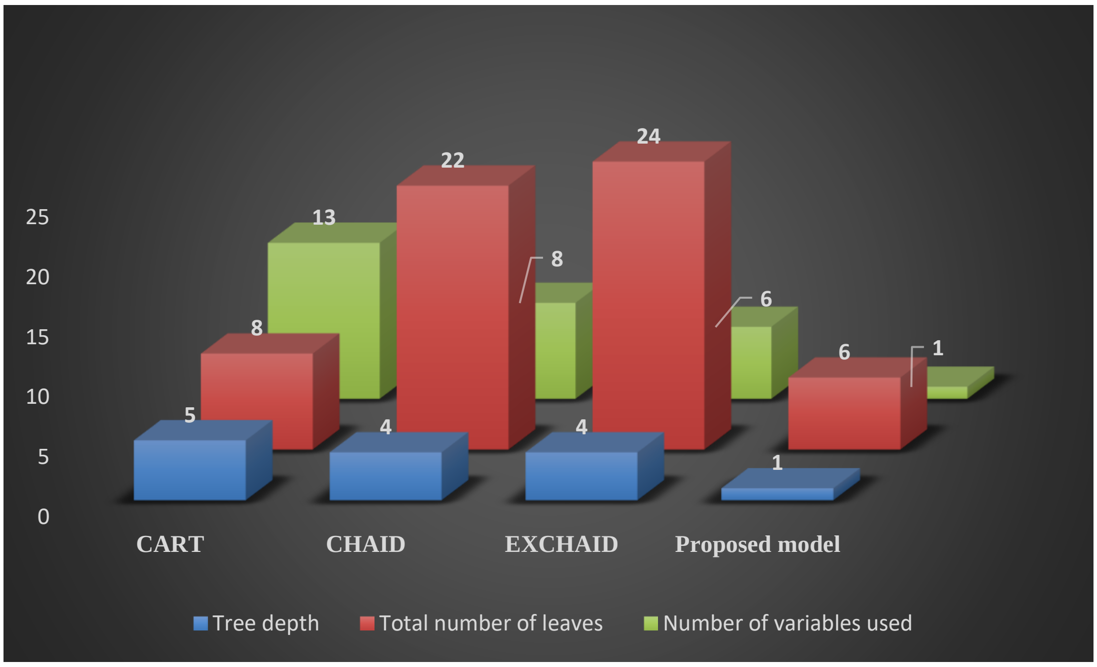

Figure 14. Comparison of model's complexity

Table 9. Comparison models accuracy

CART	CHAID	EXCHAID	Proposed model
75.47%	72.49%	71.17%	83.02%

6. Conclusion and Future research

This research presented a novel DT algorithm by incorporating the concept of regression and more specifically the coefficient of determination. The proposed model can be summarized in three steps: tree construction, pruning and evaluation. The developed DT showed promising results as compared to those of the existing algorithms such as C5.0, CART, CHAID, EXCHAID, and QUEST. The uniqueness of the proposed tree is to manage discrete and continuous variables as independent and dependent variables in DT construction by applying a coefficient of determination in node split. The results indicated the high accuracy and low complexity of the proposed tree compared to other models. The high accuracy of the proposed model indicated that the rules are extracted with greater purity and therefore useful knowledge is achievable. Also, performing sensitivity analysis on the type of dependent variable and changes in Conpercent and Dispercant values demonstrated strength and stability of the proposed tree compared to other models. In the future, several independent variables in each node may be considered during construction of DT (i.e., constructing an oblique DT).

References

Abellán, J., & Moral, S. (2003). Building classification trees using the total uncertainty criterion. International Journal of Intelligent Systems, 18 (12), 1215–1225.

Abellán, J., Mantas, C. J., & Castellano, J. G. (2018). Adaptative CC4. 5: Credal C4. 5 with a rough class noise estimator. Expert Systems with Applications, 92, 363-379.

Abpeikar, S., Ghatee, M., Foresti, G. L., & Micheloni, C. (2020). Adaptive neural tree exploiting expert nodes to classify high-dimensional data. Neural Networks, 124, 20-38.

Bach, M. P., Dumičić, K., Žmuk, B., Ćurlin, T., & Zoroja, J. (2018). Internal fraud in a project-based organization: CHAID decision tree analysis. Procedia computer science, 138, 680-687.

Baitharu, T. R., & Pani, S. K. (2016). Analysis of data mining techniques for healthcare decision support system using liver disorder dataset. Procedia Computer Science, 85, 862-870.

Baloochian, H., & Ghaffary, H. R. (2019). Multiclass Classification Based on Multi-criteria Decision-making. Journal of Classification, 36(1), 140-151.

Bar-Hen, A., Gey, S., & Poggi, J. M. (2015). Influence measures for CART classification trees. Journal of Classification, 32(1), 21-45.

Barsacchi, M., Bechini, A., & Marcelloni, F. (2020). An analysis of boosted ensembles of binary fuzzy decision trees. Expert Systems with Applications, 154, 113436.

Begon, J. M., Joly, A., & Geurts, P. (2017, July). Globally induced forest: A prepruning compression scheme. In International Conference on Machine Learning (pp. 420-428). PMLR.

Benkercha, R., & Moulahoum, S. (2018). Fault detection and diagnosis based on C4. 5 decision tree algorithms for grid connected PV system. Solar Energy, 173, 610-634.

Biggs, D., De Ville, B., & Suen, E. (1991). A method of choosing multiway partitions for classification and decision trees. Journal of applied statistics, 18(1), 49-62.

Bobadilla, J., Ortega, F., Hernando, A., & Glez-de-Rivera, G. (2013). A similarity metric designed to speed up, using hardware, the recommender systems k-nearest neighbors' algorithm. Knowledge-Based Systems, 51, 27-34.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (2017). Classification and regression trees. Routledge.

Chandra, B., Kothari, R., & Paul, P. (2010). A new node splitting measure for decision tree construction. Pattern Recognition, 43(8), 2725-2731.

Chang, C. L., & Chen, C. H. (2009). Applying decision tree and neural network to increase quality of dermatologic diagnosis. Expert Systems with Applications, 36(2), 4035-4041.

Chou, J. S. (2012). Comparison of multilabel classification models to forecast project dispute resolutions. Expert Systems with Applications, 39(11), 10202-10211.

Cohen, S. (2021). The basics of machine learning: strategies and techniques. In Artificial Intelligence and Deep Learning in Pathology (pp. 13-40). Elsevier.

Czajkowski, M., & Kretowski, M. (2019). Decision tree underfitting in mining of gene expression data. An evolutionary multi-test tree approach. Expert Systems with Applications, 137, 392-404.

De Caigny, A., Coussement, K., & De Bock, K. W. (2018). A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees. European Journal of Operational Research, 269(2), 760-772.

Delen, D., Kuzey, C., & Uyar, A. (2013). Measuring firm performance using financial ratios: A decision tree approach. Expert systems with applications, 40(10), 3970-3983.

Farid, D. M., Zhang, L., Rahman, C. M., Hossain, M. A., & Strachan, R. (2014). Hybrid decision tree and naïve Bayes classifiers for multi-class classification tasks. Expert Systems with Applications, 41(4), 1937-1946.

Fayyad, U. M., & Irani, K. B. (1992). On the handling of continuous-valued attributes in decision tree generation. Machine learning, 8(1), 87-102.

Ghiasi, M. M., Zendehboudi, S., & Mohsenipour, A. A. (2020). Decision tree-based diagnosis of coronary artery disease: CART model. Computer methods and programs in biomedicine, 192, 105400.

Ginde, A. A., Liu, M. C., & Camargo, C. A. (2009). Demographic differences and trends of vitamin D insufficiency in the US population, 1988-2004. Archives of internal medicine, 169(6), 626-632.

Gkioulekas, I., & Papageorgiou, L. G. (2021). Tree regression models using statistical testing and mixed integer programming. Computers & Industrial Engineering, 153, 107059.

Guo, Z., Shi, Y., Huang, F., Fan, X., & Huang, J. (2021). Landslide susceptibility zonation method based on C5. 0 decision tree and K-means cluster algorithms to improve the efficiency of risk management. Geoscience Frontiers, 101249.

Hamsa, H., Indiradevi, S., & Kizhakkethottam, J. J. (2016). Student academic performance prediction model using decision tree and fuzzy genetic algorithm. Procedia Technology, 25, 326–332.

Han, J., Pei, J., Kamber, M. (2011). Data Mining: Concepts and Techniques. Elsevier.

Handley, T. E., Hiles, S. A., Inder, K. J., Kay-Lambkin, F. J., Kelly, B. J., Lewin, T. J., ... & Attia, J. R. (2014). Predictors of suicidal ideation in older people: a decision tree analysis. The American Journal of Geriatric Psychiatry, 22(11), 1325-1335.

Höppner, F. (2020). Multidimensional Decision Tree Splits to Improve Interpretability. Procedia Computer Science, 176, 156-165.

Hothorn, T., Hornik, K., & Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. Journal of Computational and Graphical statistics, 15(3), 651-674.

Itani, S., Lecron, F., & Fortemps, P. (2020). A one-class classification decision tree based on kernel density estimation. Applied soft computing, 91, 106250.

Kappelhof, N., Ramos, L. A., Kappelhof, M., van Os, H. J. A., Chalos, V., van Kranendonk, K. R., ... & Marquering, H. A. (2021). Evolutionary algorithms and decision trees for predicting poor outcome after endovascular treatment for acute ischemic stroke. Computers in Biology and Medicine, 133, 104414.

Kass, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. Applied statistics, 119-127.

Keswani, B., Vijay, L., Keswani, P., Vijay, P., & Mohapatra, A. G. (2020). Amalgamation of Machine Learning and Artificial Intelligence for Breast Cancer Detection. In Terahertz Biomedical and Healthcare Technologies (pp. 177-193). Elsevier.

Khajehei, M., & Etemady, F. (2010, September). Data mining and medical research studies. In 2010 Second International Conference on computational intelligence, modelling and simulation (pp. 119-122). IEEE.

Khalili-Damghani, K., Abdi, F., & Abolmakarem, S. (2018). Hybrid soft computing approach based on clustering, rule mining, and decision tree analysis for customer segmentation problem: Real case of customer-centric industries. Applied Soft Computing, 73, 816-828.

Khoshgoftaar, T. M., Allen, E. B., Jones, W. D., & Hudepohl, J. P. (2000). Accuracy of software quality models over multiple releases. Annals of Software Engineering, 9(1-2), 103-116.

Kotsiantis, S. B. (2013). Decision trees: a recent overview. Artificial Intelligence Review, 39(4), 261-283.

Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering, 160, 3-24.

Kudła, P., & Pawlak, T. P. (2018). One-class synthesis of constraints for Mixed-Integer Linear Programming with C4. 5 decision trees. Applied Soft Computing, 68, 1-12.

Kuhn, M., & Johnson, K. (2013). Classification trees and rule-based models. In Applied predictive modeling (pp. 369-413). Springer, New York, NY.

Lee, S. K. (2006). On classification and regression trees for multiple responses and its application. Journal of Classification, 23(1), 123-141.

Le, T., & Clarke, B. (2018). On the interpretation of ensemble classifiers in terms of Bayes classifiers. Journal of Classification, 35(2), 198-229.

Li, X., Zhao, H., & Zhu, W. (2015). A cost sensitive decision tree algorithm with two adaptive mechanisms. Knowledge-Based Systems, 88, 24-33.

Lippmann, R. P. (1987). An introduction to computing with neural nets. IEEE Assp magazine, 4(2), 4-22.

Loh, W. Y. & Shih, Y. S. (1997). Split selection methods for classification trees. Statistica Sinica, 815-840.

Lu, J., Behbood, V., Hao, P., Zuo, H., Xue, S., & Zhang, G. (2015). Transfer learning using computational intelligence: A survey. Knowledge-Based Systems, 80, 14-23.

Luo, X., Xia, J., & Liu, Y. (2021). Extraction of dynamic operation strategy for standalone solar-based multi-energy systems: A method based on decision tree algorithm. Sustainable Cities and Society, 70, 102917.

Manikandan, R., Patan, R., Gandomi, A. H., Sivanesan, P., & Kalyanaraman, H. (2020). Hash polynomial two factor decision tree using IoT for smart health care scheduling. Expert Systems with Applications, 141, 112924.

Mantas, C. J., & Abellán, J. (2014). Analysis and extension of decision trees based on imprecise probabilities: Application on noisy data. Expert Systems with Applications, 41(5), 2514-2525.

Mantas, C. J., & Abellán, J. (2014b). Credal-C4. 5: Decision tree based on imprecise probabilities to classify noisy data. Expert Systems with Applications, 41(10), 4625-4637.

Mantas, C. J., Abellán, J., & Castellano, J. G. (2016). Analysis of Credal-C4. 5 for classification in noisy domains. Expert Systems with Applications, 61, 314-326.

Mehta, M., Agrawal, R., & Rissanen, J. (1996, March). SLIQ: A fast scalable classifier for data mining. In International conference on extending database technology (pp. 18-32). Springer, Berlin, Heidelberg.

Meng, X., Zhang, P., Xu, Y., & Xie, H. (2020). Construction of decision tree based on C4. 5 algorithm for online voltage stability assessment. International Journal of Electrical Power & Energy Systems, 118, 105793.

Milanović, M., & Stamenković, M. (2016). CHAID decision tree: Methodological frame and application. Economic Themes, 54(4), 563-586.

Myles, A. J., Feudale, R. N., Liu, Y., Woody, N. A., & Brown, S. D. (2004). An introduction to decision tree modeling. Journal of Chemometrics: A Journal of the Chemometrics Society, 18(6), 275-285.

Oliver, J. J., & Hand, D. (1996). Averaging over decision trees. Journal of Classification, 13(2), 281-297.

Panhalkar, A. R., & Doye, D. D. (2021). Optimization of decision trees using modified African buffalo algorithm. Journal of King Saud University-Computer and Information Sciences.

Pashaei, E., Ozen, M., & Aydin, N. (2015, August). Improving medical diagnosis reliability using Boosted C5. 0 decision tree empowered by Particle Swarm Optimization. In 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC) (pp. 7230-7233). IEEE.

Patel, N., & Upadhyay, S. (2012). Study of various decision tree pruning methods with their empirical comparison in WEKA. International journal of computer applications, 60(12).

Perri, P. F., & van der Heijden, P. G. (2012). A property of the CHAID partitioning method for dichotomous randomized response data and categorical predictors. Journal of classification, 29(1), 76-90.

Pilz, S., Trummer, C., Pandis, M., Schwetz, V., Aberer, F., Gruebler, M., ... & Maerz, W. (2018). Vitamin D: current guidelines and future outlook. Anticancer research, 38(2), 1145-1151.

Piramuthu, S. (2008). Input data for decision trees. Expert Systems with applications, 34(2), 1220-1226.

Pradhan, B. (2013). A comparative study on the predictive ability of the decision tree, support vector machine and neuro-fuzzy models in landslide susceptibility mapping using GIS. Computers & Geosciences, 51, 350-365.

Przybyła-Kasperek, M., & Aning, S. (2021). Stop Criterion in Building Decision Trees with Bagging Method for Dispersed Data. Procedia Computer Science, 192, 3560-3569.

Quadrianto, N., & Ghahramani, Z. (2014). A very simple safe-Bayesian random forest. IEEE transactions on pattern analysis and machine intelligence, 37(6), 1297-1303.

Quinlan, J. R. (1986). Induction of decision trees. Machine learning, 1(1), 81-106.

Quinlan, J. R. (1996). Improved use of continuous attributes in C4. 5. Journal of artificial intelligence research, 4, 77-90.

Rajeswari, S., & Suthendran, K. (2019). C5. 0: Advanced Decision Tree (ADT) classification model for agricultural data analysis on cloud. Computers and Electronics in Agriculture, 156, 530-539.

Rezapour, M., Molan, A. M., & Ksaibati, K. (2020). Analyzing injury severity of motorcycle at-fault crashes using machine learning techniques, decision tree and logistic regression models. International journal of transportation science and technology, 9(2), 89-99.

Rutkowski, L., Jaworski, M., Pietruczuk, L., & Duda, P. (2014). The CART decision tree for mining data streams. Information Sciences, 266, 1–15.

Sagi, O., & Rokach, L. (2021). Approximating XGBoost with an interpretable decision tree. Information Sciences, 572, 522-542.

Sahani, N., & Ghosh, T. (2021). GIS-based spatial prediction of recreational trail susceptibility in protected area of Sikkim Himalaya using logistic regression, decision tree and random forest model. Ecological Informatics, 101352.

Salzberg, S.L. (1994). C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers Inc, 1993. Mach Learn 16, 235–240. https://doi.org/10.1007/ BF00993309.

Sarker, I. H. (2018). Mobile data science: Towards understanding data-driven intelligent mobile applications. arXiv preprint arXiv:1811.02491.

Sarker, I. H., Colman, A., Han, J., Khan, A. I., Abushark, Y. B., & Salah, K. (2020). Behavdt: A behavioral decision tree learning to build user-centric context-aware predictive model. Mobile Networks and Applications, 25(3), 1151–1161.

Saroj, R. K., & Anand, M. (2021). Environmental factors prediction in preterm birth using comparison between logistic regression and decision tree methods: an exploratory analysis. Social Sciences & Humanities Open, 4(1), 100216.

Shobha, G., & Rangaswamy, S. (2018). Machine learning. In Handbook of statistics (Vol. 38, pp. 197-228). Elsevier.

Sies, A., & Van Mechelen, I. (2020). C443: a Methodology to See a Forest for the Trees. Journal of Classification, 37(3), 730-753.

Sim, D. Y. Y., Teh, C. S., & Ismail, A. I. (2017). Improved boosting algorithms by pre-pruning and associative rule mining on decision trees for predicting obstructive sleep apnea. Advanced Science Letters, 23(11), 11593-11598.

Sut, N., & Simsek, O. (2011). Comparison of regression tree data mining methods for prediction of mortality in head injury. Expert systems with applications, 38(12), 15534-15539.

Tanyu, B. F., Abbaspour, A., Alimohammadlou, Y., & Tecuci, G. (2021). Landslide susceptibility analyses using Random Forest, C4. 5, and C5. 0 with balanced and unbalanced datasets. CATENA, 203, 105355.

Tao, Q., Li, Z., Xu, J., Xie, N., Wang, S., & Suykens, J. A. (2021). Learning with continuous piecewise linear decision trees. Expert Systems with Applications, 168, 114214.

Wang, F., Wang, Q., Nie, F., Li, Z., Yu, W., & Ren, F. (2020). A linear multivariate binary decision tree classifier based on K-means splitting. Pattern Recognition, 107, 107521.

Wang, F., Wang, Q., Nie, F., Yu, W., & Wang, R. (2018). Efficient tree classifiers for large scale datasets. Neurocomputing, 284, 70-79.

Wang, G., Zhang, C., & Huang, L. (2008, August). A study of classification algorithm for data mining based on hybrid intelligent systems. In 2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (pp. 371-375). IEEE.

Wang, L., Li, Q., Yu, Y., & Liu, J. (2018). Region compatibility based stability assessment for decision trees. Expert Systems with Applications, 105, 112-128.

Wang, R., Kwong, S., Wang, X. Z., & Jiang, Q. (2014). Segment based decision tree induction with continuous valued attributes. IEEE transactions on cybernetics, 45(7), 1262-1275.

Wang, Y., Xia, S. T., & Wu, J. (2017). A less-greedy two-term Tsallis Entropy Information Metric approach for decision tree classification. Knowledge-Based Systems, 120, 34-42.

Wang, Y., Zhang, Y., Lu, Y., & Yu, X. (2020). A Comparative Assessment of Credit Risk Model Based on Machine Learning—— a case study of bank loan data. Procedia Computer Science, 174, 141-149.

Windeatt, T., & Ardeshir, G. (2001, September). An empirical comparison of pruning methods for ensemble classifiers. In International Symposium on Intelligent Data Analysis (pp. 208-217). Springer, Berlin, Heidelberg.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2017). Trees and rules. Data Mining, 209–242.

Wu, C. C., Chen, Y. L., Liu, Y. H., & Yang, X. Y. (2016). Decision tree induction with a constrained number of leaf nodes. Applied Intelligence, 45(3), 673-685.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., ... & Steinberg, D. (2008). Top 10 algorithms in Data mining‖, knowl inf Syst.

Yeo, B., & Grant, D. (2018). Predicting service industry performance using decision tree analysis. International Journal of Information Management, 38(1), 288-300.

Yeturu, K. (2020). Machine learning algorithms, applications, and practices in data science. In Handbook of Statistics (Vol. 43, pp. 81-206). Elsevier.

Yu, F., Li, G., Chen, H., Guo, Y., Yuan, Y., & Coulton, B. (2018). A VRF charge fault diagnosis method based on expert modification C5. 0 decision tree. International Journal of Refrigeration, 92, 106-112.

Zhang, X., Dai, J., & Yu, Y. (2015). On the union and intersection operations of rough sets based on various approximation spaces. Information Sciences, 292, 214-229.

Appendix. MATLAB codes

```
clc;clear;
rowNumber=895;
colNumber=14;
deVarType='D';
inVarType=['D';'D';'D';'C';'C';'C';'C';'C';'C';'C';'C';'C';'C'];
minDataNumber=30;
disPercent=0.5;
conPercent=0.5;
inVarNumber=colNumber-1;
[num,txt]=xlsread('data.xlsx',['A',num2str(1),':','A',num2str(rowNumber)]);
if deVarType=='D'
   deVarData=txt;
else
   deVarData=num;
end
inVarData=cell(1,colNumber-1);
for i=2:colNumber
   [num,txt]=xlsread('data.xlsx',[xlcolumnletter(i),num2str(1),':',xlcolumnletter(i),num2str(rowNumber)]);
   if inVarType(i-1)=='C'
      inVarData{i-1}=num;
   else
      inVarData{i-1}=txt;
   end
end
createdNodes(1,1)=1;
leafNodes(1,1)=0;
deletedNodes(1,1)=0;
investigatedNodes=[];
currentLevel=1;
currentNumber=1;
nodes(1,1).includedRow=ones(1,rowNumber);
nodes(1,1).selectedVar=[];
nodes(1,1).rSquare=[];
nodes(1,1).leaf=0;
nodes(1,1).deleted=0;
nodes(1,1).rootNumber=[];
nodes(1,1).direction=[];
nodes(1,1).Value={};
nodes(1,1).separation=[];
nodes(1,1).allowedVars=(1:inVarNumber);
lastNumber=zeros(1,inVarNumber+1);
lastNumber(1)=1;
counter=0;
tic;
warning('off','all')
while 1
   counter=counter+1;
   currentDataIndex=find(nodes(currentLevel,currentNumber).includedRow==1);
   currentDataNumber=size(currentDataIndex,1);
   allowedVars=nodes(currentLevel,currentNumber).allowedVars;
   if deVarType=='C'
      Rsq=zeros(1,length(allowedVars));
      for i=1:length(allowedVars)
         if inVarType(allowedVars(i))=='C'
            temp=[inVarData{allowedVars(i)}];
            X=[ones(size(currentDataIndex))',temp(currentDataIndex,:)];
            Y=deVarData(currentDataIndex,:);
            b=X\Y;
            Y_hat=X*b;
            Rsq(i)=1-sum((Y - Y_hat).^2)/sum((Y - mean(Y)).^2);
         elseif inVarType(allowedVars(i))=='D'
            temp=[inVarData{allowedVars(i)}];
            X=temp(currentDataIndex,:);
            X_dummy=dummyvar(nominal(X));
            X_dummy=X_dummy(:,1:size(X_dummy,2)-1);
            Y=deVarData(currentDataIndex,:);
            X_prim=[ones(size(currentDataIndex))',X_dummy];
            [~,~,~,~,stats]=regress(Y,X_prim);
            Rsq(i)=stats(1);
         end
      end
      sortedRsq=sort(Rsq,'descend');
      candidateIndex=find(Rsq==sortedRsq(1));
      selectedVar=datasample(allowedVars(candidateIndex),1,'replace',false);
      nodes(currentLevel,currentNumber).selectedVar=selectedVar;
      nodes(currentLevel,currentNumber).rSquare=Rsq(candidateIndex);
   elseif deVarType=='D'
      n=length(currentDataIndex);
      nagelkerke=zeros(1,length(allowedVars));
      for i=1:length(allowedVars)
         if inVarType(allowedVars(i))=='C'
            temp=[inVarData{allowedVars(i)}];
            X=temp(currentDataIndex,:);
            Y=deVarData(currentDataIndex,:);
            Y_dummy=dummyvar(nominal(Y));
            Y_dummy=Y_dummy(:,1:size(Y_dummy,2)-1);
            logLikePart1=zeros(size(currentDataIndex,2),size(Y_dummy,2));
            logLike0Part1=zeros(size(currentDataIndex,2),size(Y_dummy,2));
            logLikePart2=zeros(size(currentDataIndex,2),1);
            logLike0Part2=zeros(size(currentDataIndex,2),1);
            for j=1:size(Y_dummy,2)
               [b,~,~]=glmfit(X,Y_dummy(:,j),'binomial', 'link', 'logit');
               %[B,~,~]=mnrfit(X,categorical(Y));
               %b=B(:,j);
               lin=b(1)+b(2)*X;
               logLikePart1(:,j)=Y_dummy(:,j).*lin;
               logLikePart2(:,j)=exp(lin);
               X0=zeros(size(Y_dummy(:,j)));
               [b0,~,~] = glmfit(X0,Y_dummy(:,j),'binomial','link','logit');
               %[B0,~,~]=mnrfit(X0,categorical(Y));
               %b0=B0(:,j);
               logLike0Part1(:,j)=Y_dummy(:,j).*b0(1);
               logLike0Part2(:,j)=exp(ones(size(currentDataIndex,2),1)*b0(1));
            end
            logLike=sum(sum(logLikePart1,2)-log(1+sum(logLikePart2,2)));
            logLike0=sum(sum(logLike0Part1,2)-log(1+sum(logLike0Part2,2)));
            nagelkerke(i)=(1-(exp((logLike0-logLike)*(2/n))))/(1-exp(logLike0*(2/n)));
         elseif inVarType(allowedVars(i))=='D'
            temp=[inVarData{allowedVars(i)}];
            X=temp(currentDataIndex,:);
            X_dummy=dummyvar(nominal(X));
            X_dummy=X_dummy(:,1:size(X_dummy,2)-1);
            Y=deVarData(currentDataIndex,:);
            Y_dummy=dummyvar(nominal(Y));
            Y_dummy=Y_dummy(:,1:size(Y_dummy,2)-1);
            if isempty(X_dummy)
               X_dummy=zeros(size(Y_dummy,1),1);
            end
            logLikePart1=zeros(size(currentDataIndex,2),size(Y_dummy,2));
            logLike0Part1=zeros(size(currentDataIndex,2),size(Y_dummy,2));
            logLikePart2=zeros(size(currentDataIndex,2),1);
            logLike0Part2=zeros(size(currentDataIndex,2),1);
            for j=1:size(Y_dummy,2)
                [b,~,~]=glmfit(X_dummy,Y_dummy(:,j),'binomial', 'link', 'logit');
                %[B,~,~]=mnrfit(X_dummy,categorical(Y));
                %b=B(:,j);
                lin=b(1)+X_dummy*b(2:size(b,1));
                logLikePart1(:,j)=Y_dummy(:,j).*lin;
                logLikePart2(:,j)=exp(lin);
                X0=zeros(size(Y_dummy(:,j)));
                [b0,~,~] = glmfit(X0,Y_dummy(:,j),'binomial','link','logit');
                %[B0,~,~]=mnrfit(X0,categorical(Y));
                %b0=B0(:,j);
                logLike0Part1(:,j)=Y_dummy(:,j).*b0(1);
                logLike0Part2(:,j)=exp(ones(size(currentDataIndex,2),1)*b0(1));
            end
             logLike=sum(sum(logLikePart1,2)-log(1+sum(logLikePart2,2)));
             logLike0=sum(sum(logLike0Part1,2)-log(1+sum(logLike0Part2,2)));
             nagelkerke(i)=(1-(exp((logLike0-logLike)*(2/n))))/(1-exp(logLike0*(2/n)));
             end
            end
            sortedNag=sort(nagelkerke,'descend');
            candidateIndex=find(nagelkerke==sortedNag(1));
            selectedVar=datasample(allowedVars(candidateIndex),1,'replace',false);
            nodes(currentLevel,currentNumber).rSquare=nagelkerke(candidateIndex);
            nodes(currentLevel,currentNumber).selectedVar=selectedVar;
         end
         temp=[inVarData{selectedVar}];
         selectedInVarData=temp(currentDataIndex);
            if inVarType(selectedVar)=='C'
  createdNodes(currentLevel+1,lastNumber(currentLevel+1)+1:lastNumber(currentLevel+1)+2)=ones(1,2);
            nodes(currentLevel+1,lastNumber(currentLevel+1)+1).rootNumber=currentNumber;
            nodes(currentLevel+1,lastNumber(currentLevel+1)+2).rootNumber=currentNumber;
            nodes(currentLevel,currentNumber).separation=min(selectedInVarData)+(max(selectedInVarData)-min(selectedInVarData))/2;
            nodes(currentLevel+1,lastNumber(currentLevel+1)+1).direction='<=';
   nodes(currentLevel+1,lastNumber(currentLevel+1)+1).value=nodes(currentLevel,currentNumber).separation;
            nodes(currentLevel+1,lastNumber(currentLevel+1)+2).direction='>';
   nodes(currentLevel+1,lastNumber(currentLevel+1)+2).value=nodes(currentLevel,currentNumber).separation;
            index1= selectedInVarData<=nodes(currentLevel,currentNumber).separation;
            index1=currentDataIndex(index1);
            index2= selectedInVarData>nodes(currentLevel,currentNumber).separation;
            index2=currentDataIndex(index2);
            includedRow1=zeros(1,rowNumber);
            includedRow2=zeros(1,rowNumber);
            includedRow1(index1)=1;
            includedRow2(index2)=1;
            nodes(currentLevel+1,lastNumber(currentLevel+1)+1).includedRow=includedRow1;
   nodes(currentLevel+1,lastNumber(currentLevel+1)+1).allowedVars=allowedVars(allowedVars~=selectedVar);
   nodes(currentLevel+1,lastNumber(currentLevel+1)+2).allowedVars=allowedVars(allowedVars~=selectedVar);
            if sum(nodes(currentLevel+1,lastNumber(currentLevel+1)+1).includedRow)<minDataNumber
             nodes(currentLevel+1,lastNumber(currentLevel+1)+1).deleted=1;
             deletedNodes(currentLevel+1,lastNumber(currentLevel+1)+1)=1;
            else
             nodes(currentLevel+1,lastNumber(currentLevel+1)+1).deleted=0;
             deletedNodes(currentLevel+1,lastNumber(currentLevel+1)+1)=0;
            end
            nodes(currentLevel+1,lastNumber(currentLevel+1)+2).includedRow=includedRow2;
            if sum(nodes(currentLevel+1,lastNumber(currentLevel+1)+2).includedRow)<minDataNumber
             nodes(currentLevel+1,lastNumber(currentLevel+1)+2).deleted=1;
             deletedNodes(currentLevel+1,lastNumber(currentLevel+1)+2)=1;
            else
             nodes(currentLevel+1,lastNumber(currentLevel+1)+2).deleted=0;
             deletedNodes(currentLevel+1,lastNumber(currentLevel+1)+2)=0;
            end
            investigatedNodes(currentLevel+1,lastNumber(currentLevel+1)+1)=0;
            investigatedNodes(currentLevel+1,lastNumber(currentLevel+1)+2)=0;
            if deVarType=='D'
             uniqueCell1=uniquecell(deVarData(includedRow1==1));
             for i=1:length(uniqueCell1)
               classSize=length(find(strcmp(deVarData(includedRow1==1),uniqueCell1(i))==1));
               if classSize/length(find(includedRow1==1))>=disPercent
                 leafStatus=1;
                 break;
               else
                 leafStatus=0;
               end
             end
             if leafStatus==1 && deletedNodes(currentLevel+1,lastNumber(currentLevel+1)+1)==0
               nodes(currentLevel+1,lastNumber(currentLevel+1)+1).leaf=1;
               leafNodes(currentLevel+1,lastNumber(currentLevel+1)+1)=1;
             else
               nodes(currentLevel+1,lastNumber(currentLevel+1)+1).leaf=0;
               leafNodes(currentLevel+1,lastNumber(currentLevel+1)+1)=0;
             end
             uniqueCell2=uniquecell(deVarData(includedRow2==1));
             for i=1:length(uniqueCell2)
               classSize=length(find(strcmp(deVarData(includedRow2==1),uniqueCell2(i))==1));
               if length(classSize)/length(find(includedRow2==1))>=disPercent;
                 leafStatus=1;
                 break;
               else
                 leafStatus=0;
               end
             end
```

```
        if leafStatus==1 && deletedNodes(currentLevel+1,lastNumber(currentLevel+1)+2)==0
          nodes(currentLevel+1,lastNumber(currentLevel+1)+2).leaf=1;
          leafNodes(currentLevel+1,lastNumber(currentLevel+1)+2)=1;
        else
          nodes(currentLevel+1,lastNumber(currentLevel+1)+2).leaf=0;
          leafNodes(currentLevel+1,lastNumber(currentLevel+1)+2)=0;
        end
      elseif deVarType=='C'
        dataMean=mean(deVarData(includedRow1==1));
        dataStd=std(deVarData(includedRow1==1));
        status=(deVarData(includedRow1==1)<dataMean+dataStd)+(deVarData(includedRow1==1)>dataMean-dataStd);
        classSize=length(find(status==2));
        if classSize/length(find(includedRow1==1))>=conPercent && deleted-
Nodes(currentLevel+1,lastNumber(currentLevel+1)+1)==0
          nodes(currentLevel+1,lastNumber(currentLevel+1)+1).leaf=1;
          leafNodes(currentLevel+1,lastNumber(currentLevel+1)+1)=1;
        else
          nodes(currentLevel+1,lastNumber(currentLevel+1)+1).leaf=0;
          leafNodes(currentLevel+1,lastNumber(currentLevel+1)+1)=0;
        end
        dataMean=mean(deVarData(includedRow2==1));
        dataStd=std(deVarData(includedRow2==1));          sta-
tus=(deVarData(includedRow2==1)<dataMean+dataStd)+(deVarData(includedRow2==1)>dataMean-dataStd);
        classSize=length(find(status==2));
        if classSize/length(find(includedRow2==1))>=conPercent && deleted-
Nodes(currentLevel+1,lastNumber(currentLevel+1)+2)==0
          nodes(currentLevel+1,lastNumber(currentLevel+1)+2).leaf=1;
          leafNodes(currentLevel+1,lastNumber(currentLevel+1)+2)=1;
        else
          nodes(currentLevel+1,lastNumber(currentLevel+1)+2).leaf=0;
          leafNodes(currentLevel+1,lastNumber(currentLevel+1)+2)=0;
        end
      end
      if currentLevel==inVarNumber && deletedNodes(currentLevel+1,lastNumber(currentLevel+1)+1)==0
        nodes(currentLevel+1,lastNumber(currentLevel+1)+1).leaf=1;
        leafNodes(currentLevel+1,lastNumber(currentLevel+1)+1)=1;
      end
      if currentLevel==inVarNumber+1 && deletedNodes(currentLevel+1,lastNumber(currentLevel+1)+2)==0
        nodes(currentLevel+1,lastNumber(currentLevel+1)+2).leaf=1;
        leafNodes(currentLevel+1,lastNumber(currentLevel+1)+2)=1;
      end
      lastNumber(currentLevel+1)=lastNumber(currentLevel+1)+2;
    elseif inVarType(selectedVar)=='D'
      uniqueValues=unique(selectedInVarData);        created-
Nodes(currentLevel+1,lastNumber(currentLevel+1)+1:lastNumber(currentLevel+1)+length(uniqueValues))=ones(1,length(uniqueVal
ues));
      nodes(currentLevel,currentNumber).separation=uniqueValues;
      oneClass=0;
      for i=1:length(uniqueValues)
        nodes(currentLevel+1,lastNumber(currentLevel+1)+i).rootNumber=currentNumber;
        nodes(currentLevel+1,lastNumber(currentLevel+1)+i).direction='=';
        nodes(currentLevel+1,lastNumber(currentLevel+1)+i).value=uniqueValues(i);
        index= strcmp(selectedInVarData,uniqueValues(i))==1;
        index=currentDataIndex(index);
        includedRow=zeros(1,rowNumber);
        includedRow(index)=1;
        nodes(currentLevel+1,lastNumber(currentLevel+1)+i).includedRow=includedRow;
        if sum(nodes(currentLevel+1,lastNumber(currentLevel+1)+i).includedRow)<minDataNumber
           nodes(currentLevel+1,lastNumber(currentLevel+1)+i).deleted=1;
           deletedNodes(currentLevel+1,lastNumber(currentLevel+1)+i)=1;
        else
           nodes(currentLevel+1,lastNumber(currentLevel+1)+i).deleted=0;
           deletedNodes(currentLevel+1,lastNumber(currentLevel+1)+i)=0;
        end           nodes(currentLevel+1,lastNumber(currentLevel+1)+i).allowedVars=allowedVars(allowedVars~=selectedVar);
        investigatedNodes(currentLevel+1,lastNumber(currentLevel+1)+i)=0;
        if deVarType=='D'
          uniqueCelli=uniquecell(deVarData(includedRow==1));
          for j=1:length(uniqueCelli)
            classSize=length(find(strcmp(deVarData(includedRow==1),uniqueCelli(j))==1));
            if classSize/length(find(includedRow==1))>=disPercent
              leafStatus=1;
             break;
           else
             leafStatus=0;
            end
```

```
                end
                if leafStatus==1 && deletedNodes(currentLevel+1,lastNumber(currentLevel+1)+i)==0
                    nodes(currentLevel+1,lastNumber(currentLevel+1)+i).leaf=1;
                    leafNodes(currentLevel+1,lastNumber(currentLevel+1)+i)=1;
                 else
                    nodes(currentLevel+1,lastNumber(currentLevel+1)+i).leaf=0;
                    leafNodes(currentLevel+1,lastNumber(currentLevel+1)+i)=0;
                 end
             elseif deVarType=='C'
                  dataMean=mean(deVarData(includedRow==1));
                  dataStd=std(deVarData(includedRow==1));
                  status=(deVarData(includedRow==1)<dataMean+dataStd)+(deVarData(includedRow==1)>dataMean-dataStd);
                  classSize=length(find(status==2));
                  if classSize/length(find(includedRow==1))>=conPercent && deleted-
Nodes(currentLevel+1,lastNumber(currentLevel+1)+i)==0
                    nodes(currentLevel+1,lastNumber(currentLevel+1)+i).leaf=1;
                    leafNodes(currentLevel+1,lastNumber(currentLevel+1)+i)=1;
                 else
                    nodes(currentLevel+1,lastNumber(currentLevel+1)+i).leaf=0;
                    leafNodes(currentLevel+1,lastNumber(currentLevel+1)+i)=0;
                  end
             end
             if currentLevel==inVarNumber+1 && deletedNodes(currentLevel+1,lastNumber(currentLevel+1)+i)==0
                nodes(currentLevel+1,lastNumber(currentLevel+1)+i).leaf=1;
                leafNodes(currentLevel+1,lastNumber(currentLevel+1)+i)=1;
            end
        end
        lastNumber(currentLevel+1)=lastNumber(currentLevel+1)+length(uniqueValues);
    end
    investigatedNodes(currentLevel,currentNumber)=1;
    selectedLevel=0;
    for i=2:size(createdNodes,1)
        index=find(createdNodes(i,:)==1);
       if ~isempty(find(investigatedNodes(i,index)+leafNodes(i,index)+deletedNodes(i,index)==0, 1))
          selectedLevel=i;
          break;
       end
    end
    if selectedLevel~=0
        candidateNumbers=find(1-
createdNodes(selectedLevel,:)+leafNodes(selectedLevel,:)+investigatedNodes(selectedLevel,:)+deletedNodes(selectedLevel,:)==0);
        selectedNumber=candidateNumbers(1);
    else
        break;
    end
    currentLevel=selectedLevel;
    currentNumber=selectedNumber;
end
toc;
[cIndex1,cIndex2]=find(createdNodes==1);
[dIndex1,dIndex2]=find(deletedNodes==1);
[lIndex1,lIndex2]=find(leafNodes==1);
disp('Number of created nodes:');
disp(length(cIndex1));
disp('Number of leaf nodes:');
disp(length(lIndex1));
disp('Number of deleted nodes:');
disp(length(dIndex1));
disp('Writing results to excel...');
for i=1:length(lIndex1)
    level=lIndex1(i);
    number=lIndex2(i);
    rowNumbers=find(nodes(level,number).includedRow==1)';
    xlswrite('leaves.xlsx',rowNumbers,['leaf',num2str(i),'-','rows'],['A1',':','A',num2str(length(rowNumbers))]);
    xlswrite('leaves.xlsx',deVarData(rowNumbers),['leaf',num2str(i),'-','rows'],['B1',':','B',num2str(length(rowNumbers))]);
    count=0;
    while level>1
        count=count+1;
        rootNumber=nodes(level,number).rootNumber;
        xlswrite('leaves.xlsx',nodes(level-1,rootNumber).selectedVar,['leaf',num2str(i),'-
','rule'],['A',num2str(count),':','A',num2str(count)]);
        xlswrite('leaves.xlsx',nodes(level,number).direction,['leaf',num2str(i),'-','rule'],['B',num2str(count),':','B',num2str(count)]);
        xlswrite('leaves.xlsx',nodes(level,number).value,['leaf',num2str(i),'-','rule'],['C',num2str(count),':','C',num2str(count)]);
        number=nodes(level,number).rootNumber;
        level=level-1;
    end
end
for i=1:length(dIndex1)
    level=dIndex1(i);
    number=dIndex2(i);
    rowNumbers=find(nodes(level,number).includedRow==1)';
    if ~isempty(rowNumbers)
        xlswrite('deleted.xlsx',rowNumbers,['deleted',num2str(i),'-','rows'],['A1',':','A',num2str(length(rowNumbers))]);
        xlswrite('deleted.xlsx',deVarData(rowNumbers),['deleted',num2str(i),'-','rows'],['B1',':','B',num2str(length(rowNumbers))]);
    end
end
```

This article can be cited: Jiryaei Sharahi, Z., Zare Mehrjerdi, Y., Owlia, M.S., Abessi, M., (2022). Machine learning decision tree based on regression in data mining to extract more knowledge. Journal of Industrial Engineering and Management Studies, Vol. 9, No. 2, pp. 86-112.