

1-1-2017

DTreeSim: A new approach to compute decision tree similarity using re-mining

GÖZDE BAKIRLI

DERYA BİRANT

Follow this and additional works at: <https://journals.tubitak.gov.tr/elektrik>



Part of the [Computer Engineering Commons](#), [Computer Sciences Commons](#), and the [Electrical and Computer Engineering Commons](#)

Recommended Citation

BAKIRLI, GÖZDE and BİRANT, DERYA (2017) "DTreeSim: A new approach to compute decision tree similarity using re-mining," *Turkish Journal of Electrical Engineering and Computer Sciences*: Vol. 25: No. 1, Article 9. <https://doi.org/10.3906/elk-1504-234>

Available at: <https://journals.tubitak.gov.tr/elektrik/vol25/iss1/9>

This Article is brought to you for free and open access by TÜBİTAK Academic Journals. It has been accepted for inclusion in Turkish Journal of Electrical Engineering and Computer Sciences by an authorized editor of TÜBİTAK Academic Journals. For more information, please contact academic.publications@tubitak.gov.tr.

DTreeSim: A new approach to compute decision tree similarity using re-mining

Gözde BAKIRLI*, Derya BİRANT

Department of Computer Engineering, Faculty of Engineering, Dokuz Eylül University, İzmir, Turkey

Received: 27.04.2015

Accepted/Published Online: 21.11.2015

Final Version: 24.01.2017

Abstract: A number of recent studies have used a decision tree approach as a data mining technique; some of them needed to evaluate the similarity of decision trees to compare the knowledge reflected in different trees or datasets. There have been multiple perspectives and multiple calculation techniques to measure the similarity of two decision trees, such as using a simple formula or an entropy measure. The main objective of this study is to compute the similarity of decision trees using data mining techniques. This study proposes DTreeSim, a new approach that applies multiple data mining techniques (classification, sequential pattern mining, and k-nearest neighbors) sequentially to identify similarities among decision trees. After the construction of decision trees from different data marts using a classification algorithm, sequential pattern mining was applied to the decision trees to obtain rules, and then the k-nearest neighbor algorithm was performed on these rules to compute similarities using two novel measures: general similarity and pieced similarity. Our experimental studies compared the results of these novel similarity measures and also compared our approach with existing approaches. Our comparisons indicate that our proposed approach performs better than existing approaches, because it takes into account the values of the branches in the trees through sequential pattern mining.

Key words: Data mining, classification, decision tree similarity, sequential pattern mining, k-nearest neighbors, re-mining

1. Introduction

Data mining is the process of analyzing data and extracting hidden useful patterns from large datasets to provide information about a related condition or estimation of a future similar condition. There are different types of data mining techniques, which serve different purposes. The main purposes are classifying new data based on training data, clustering similar instances, finding outliers, and discovering interesting patterns inherent in data. For example, association rule mining is used to find interesting relationships among a large set of data items. For example, outlier detection is used to detect anomalous observations (i.e. fraud) in a sample dataset while clustering is a data mining task to group similar objects together. Clustering techniques typically depend on the knowledge of cluster number. If cluster number is not known, then an algorithm without this input parameter (i.e. DBSCAN) can be applied or cluster validity indices can be utilized to determine the number of clusters in a dataset.

One of the major data mining tasks is classification, assigning items into target categories. Data mining algorithms for classifying include decision tree (DT), naive Bayes, neural network, and K-nearest neighbors (KNN) algorithms. DT algorithms build trees from a set of training data based on a heuristic to decide on the importance of the features. The C4.5 algorithm [1,2] was used as the DT algorithm in this study because it was

*Correspondence: gozdeb85@hotmail.com

ranked #1 in the top 10 algorithms for data mining in 2008 [3]. Another popular classification algorithm is KNN. In the KNN algorithm [4], an object is assigned to the class most common among its k nearest neighbors. While identifying the most similar k objects, commonly, the Euclidian distance function is used to calculate distances for continuous variables and the Jaccard distance is used for categorical variables. Since these functions are not suitable to compute DT similarity, a new distance measure was proposed in this study.

Another data mining task, sequential pattern mining (SPM), is applied to discover sequential and recurring structures or patterns. Applications of SPM include analysis of customer shopping sequences, medical treatments, natural disasters, weblog click streams, and DNA sequences. Unlike the previous studies, in this study, SPM was applied to DTs to determine compact rules for a DT to supply fast and correct comparisons for a similarity computation. First, sequence transactions were generated through paths from the top to the leaf of a tree, and then a SPM algorithm was applied to these sequence transactions to extract common patterns in the DT. These results (frequent sequences) were used to find similarity among DTs. The prefix-span algorithm [5] was selected as the SPM algorithm in this study. Since the main idea of the prefix-span algorithm is to use the least projections, quickly shrinking the sequence, it is one of the fastest SPM algorithms. It reduces the effort involved in candidate subsequence generation.

This study proposes a new approach, DTreeSim (decision tree similarity). The main goal of this study is to identify similarities among DTs by combining three data mining techniques, which are described above: 1) classification to construct DTs from a number of data marts, 2) SPM to determine the characteristics of these DTs, and 3) the KNN method to identify similar DTs, and consequently similar data marts. The evaluation of the similarity of DTs can be used for different purposes, e.g., to compare patterns related to different geographical regions or different periods. As an example, the similarity of DTs in time means that related data marts include similar observations in the time dimension. The problem of comparing two datasets can be reduced to finding similarity between two DTs corresponding to them. In addition, similarity measures can help us compare two models that have been built based on two datasets. Furthermore, the similarities among DTs are investigated for tree matching problems, ensemble learning, interpreting DTs, and change detection in classification models. A number of recent studies needed to evaluate the similarity of DTs for different purposes in different application areas such as for privacy preserving [6], agent-based modeling systems [7], the construction of a genetic algorithm tree (GATree) [8], and speech recognition [9].

Whereas data mining is used in many areas, such as sales forecasting [10], stock market analysis [11], medical treatment [12], earthquake prediction [13], and weather forecasting [14], it has not been extensively used in local municipalities. There are a few studies where data mining has been used for local municipalities [15,16]. This study addresses this lack by applying a novel approach to large datasets collected by a local municipality to investigate citizens' tax payment behaviors. In this study, citizen behavior analysis was carried out to predict future income by analyzing monthly-based tax returns for a local municipality in Turkey because the prediction of future tax payments may help local governments manage risk and budget status.

This study includes six main contributions. First, a link between DTs and SPM is introduced for the first time, with SPM applied to DTs. To achieve this goal, DTs are linearized and converted into sequences. This is a novel approach to DTs. The SPM provides compact rules for a DT to supply fast and correct comparisons. Second, a new distance function is needed for the KNN algorithm for sequential data, since existing distance functions like the Euclidian or the Manhattan do not meet our requirements. Thus, we defined and compared new distance measures to calculate the similarity of DTs: general similarity and pieced similarity. Third, data mining algorithms are applied sequentially, so a re-mining process is performed on re-mining results for the

first time. We have called this operation re²mining. Fourth, we apply a classification algorithm to multiple data marts. In this way, a forest is created from individual DTs for subject-, region-, or time-oriented analysis. Fifth, we apply our algorithm to novel experimental studies of local municipality data. Using our approach, tax payments can be analyzed and used to predict future tax revenue for the municipality. Last, DTreeSim, our new approach, is used for calculating similarity among DTs. Our comparisons indicate that our proposed approach performs better than existing approaches.

2. Related works

Works related to our study can be classified into four groups: 1) studies that investigated the similarity of DTs, 2) studies that combined SPM with other data mining techniques, 3) studies that proposed new distance formulas for KNN, and 4) studies that used re-mining.

The first group includes recent studies that have looked at the similarity of DTs from different perspectives. Ntoutsis et al. [17] presented a general framework for similarity estimation that includes, as a special case, the estimation of semantic similarity between DTs. Zhang and Jiang [18] developed splitting criteria based on similarity. Their approach supplies similarity based on the DT generation algorithm and a pruning methodology. Islam et al. [19] created another technique to find the similarity in DTs by comparing their accuracy rates. Syntactic similarity of DTs has been investigated using a graph algorithm for tree matching [20]. Another method identifies the number of same/different symbols for the same class, attribute, and attribute-value in DTs [21]. In some studies [6], the similarity of DTs was measured by the similarity of logic rules associated with the DTs.

We can classify previous studies that tried to find similarity between two different DTs into two subcategories: semantic and structural. While semantic similarity [17] measures the common feature subspace covered by two DTs for particular decision classes, structural similarity [22–24] compares two DTs structurally, focusing on their nodes, branches, total number of leaves, and tree depth. Dogra [7] provided a more complete similarity by taking into account both of these aspects.

In this study, we compare our approach (DTreeSim) with two existing methods: DTSim and Sim_{d1,d2}. The first one (DTSim) was proposed by Peahringer and Witten [22]. This function compares the set of all abstract paths, which are the multisets of attributes tested along a specific path from the root to leaf, along with the class assigned to that leaf. They ignored values of attributes and focused only on attribute names. The second method (Sim_{d1,d2}) was proposed by Perner [23,24]. He compared trees using decomposition rules and substructures. This method did not include leaf values or class labels.

The work done by Pekerskaya et al. [25] deals with the problem of detecting the difference between two datasets using DT comparisons. However, they developed a method to compare two cluster-embedded DTs. Other techniques measure the similarity of an original and a perturbed dataset by evaluating the similarity of DTs [6,26]. However, these studies focus on privacy preservation in data mining through noise addition.

Some studies (e.g., [8]) estimated the similarity of different DTs using a simple formula based only on the differences between the number of nodes and tree levels. For example, in [27], the similarity of DTs was estimated using a simple formula: $\text{tree_diff} = |(\text{levels_tree1} - \text{levels_tree2}) + (\text{nodes_tree1} - \text{nodes_tree2})|$. Another study that compared similarities between trees using a simple formula was proposed by Telaar and Fuhs [9]. They measured the fraction of joint entropy that is not mutual: $\text{DIFF}(T1, T2) = 1 - I(T2; T1)/H(T1, T2)$, where $H(T1; T2)$ is the joint entropy of the distribution over all classes in trees T1 and T2, and $I(T2; T1)$ corresponds to the mutual information on T1 and T2. In contrast to our work, they used an entropy measure

to find polyphone DT similarity. They also proposed using dissimilarity, instead of similarity. In this case, if $\text{DIFF}(T1, T2) = 0$, then the trees are identical, while, if $\text{DIFF}(T1; T2) = 1$, the trees are not correlated. In contrast to these studies, we chose not to use a simple formula to determine the similarity between two DTs; instead, we evaluate it using two data mining algorithms: the prefix-span and KNN. Some studies only considered the attributes of the trees during comparison; they ignored the actual values of attributes and their numerical/categorical cut-points. However, the approach proposed in this study considers all of these aspects.

The second group of related work includes the studies in which SPM has been combined with other data mining techniques. For example, Ma et al. [28] combined SPM with clustering to mine moving sequential patterns. Exarchos et al. [29] used SPM for the classification of proteins into folds. As another example, Tseng and Lee [30] integrated SPM with probabilistic induction to discover hidden patterns and classify data using re-mined sequential patterns. D’Silva and Vora [31] used SPM with clustering to create an intelligent recommendation system. Deng et al. [32] used association rules with decision trees and proposed a new tree model, the condition-based tree, and a new algorithm, the condition-based classifier. Unlike these previous studies, we applied SPM to DTs and we used it for a different purpose (to find similarity).

Another group of studies proposed new distance formulas for KNN, although the Euclidean distance formula is usually employed in the KNN method to measure similarity. For example, Ma and Kaban [33] explored a simple yet effective similarity definition within nearest neighbors for intrusion detection applications. Similarly, our study also proposes a new similarity measure for the KNN algorithm, but for a different purpose, to find the k-most similar DTs to a specific tree.

The last related group of work includes the studies that applied re-mining. Re-mining is a technique that applies two data mining techniques one after another. Liu and Yin [34] presented an efficient data re-mining technique for updating previously discovered association rules in the light of threshold changes. Demiriz et al. [35] proposed a new approach to mine price-, time-, and domain-related attributes through re-mining of association mining results. The underlying factors behind positive and negative relationships were characterized and described in a second data mining stage. Previous studies that include re-mining used this process for filtering results. In contrast to these studies, we use re-mining twice (re²mining), and we use it for a different purpose (for determining DT similarity).

3. DTreeSim

The main purpose of our new approach is to find the most similar trees among hundreds of DTs. The motivation for this is to analyze factors that make these trees similar and to make predictions for situations with similar factors. To achieve this goal, three data mining techniques, classification, SPM, and the KNN algorithm, are combined. Figure 1 shows the five main steps, involving two re-mining operations used in DTreeSim. First, data marts are created by partitioning a data warehouse for specific conditions. Second, a DT algorithm is applied to each data mart. Third, the paths of DTs are linearized into sequences. Fourth, a sequential pattern mining algorithm is applied to each tree sequence to find all frequent sequences. Last, the KNN algorithm is applied to the frequent sequence set to find the most similar trees.

3.1. Data marting

Data mining has traditionally operated on data in a data warehouse or data mart. A data warehouse (DW) is a collection of integrated data from one or more different sources, while a data mart (DM) is a subset of the DW created for a specific purpose. In other words, a DW supplies a general view of a problem, whereas a DM

provides a specific view for a particular problem. It is possible to create subject-, region-, or time-oriented DMs from a DW.

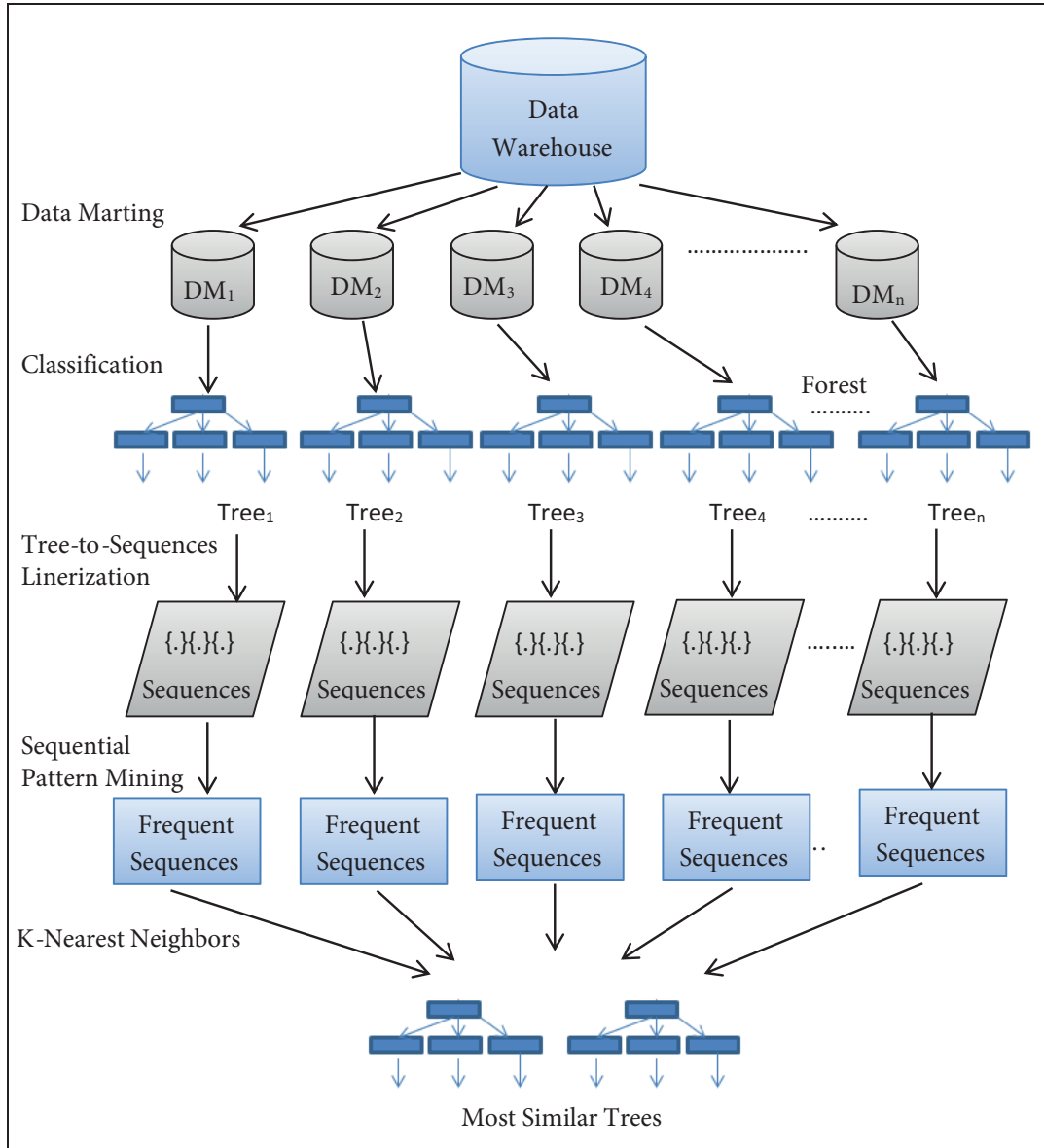


Figure 1. Schematic showing the DTreeSim approach.

DMs are created by filtering or partitioning a DW for specific conditions. Partitioning criteria, such as subjects, regions, or time intervals, should be determined to split a DW into smaller subsets.

Definition 1 Data marting is the process of partitioning a DW into smaller subsets to provide a specific view of the data. It has the following constraints:

$$DW = \bigcup_{i=1}^n DM_i \tag{1}$$

$$DM \subset DW \quad (2)$$

where n is the total number of DMs, and each DM is a subset of DW for a specific criterion.

3.2. Classification

The C4.5 decision tree algorithm in Weka is applied to each DM. At the end of this step, a forest (F) with n DTs has been created.

Definition 2 The forest (F) is defined as

$$F = \prod_{i=1}^n T_i \quad (3)$$

where n is the total number of DMs as well as the number of DTs; F is a set of n disjoint trees, and T is a tree.

3.3. Tree-to-sequence linearization

The paths of a DT are linearized and converted into sequences of branches. At the end of this step, branch sequences (BSs) are obtained from a DT and tree sequences (TSs) are acquired from various DTs.

Definition 3 A branch sequence is a sequence that is constructed along a specific path from the tree root to a given leaf, along with the class assigned to that leaf.

Definition 4 A tree sequence is the set of branch sequences related to a tree.

Definition 5 Tree-to-sequence linearization (TSL) proceeds as follows:

$$TS = \left\{ \begin{array}{l} BS_1 = \langle ([r] = [v_{r1}]) ([in_1] = [v_{11}]) ([in_2] = [v_{21}]) ([in_p] = [v_{p1}]) ([l_1]) \rangle \\ BS_2 = \langle ([r] = [v_{r2}]) ([in_1] = [v_{12}]) ([in_2] = [v_{22}]) ([in_p] = [v_{p2}]) ([l_2]) \rangle \\ \vdots \end{array} \right\}$$

where $[r]$ is the root of the decision tree, $[in_p]$ is the p th internal node in the related branch, $[l]$ is the leaf node (class), and $[v]$ is the value of the related node (a connection value between the related node and the next node). Each node and value pair $([in_1] = [v_{11}])$ is a branch, while each path of a tree is assigned as a sequence of branches. Thus, a TS is constructed from a set of BSs.

Each BS in the TS for related trees starts with $[r]$ and ends with a $[l]$. There should be only one $[r]$, but there can be multiple $[l]$ s. In this case, depth denotes the number of edges from the tree root node to the leaf node. Thus, there are Depth (1) – 1 internal nodes and value pairs in a BS.

Figure 2 gives an example of a tree structure. According to the example, the first BS is:

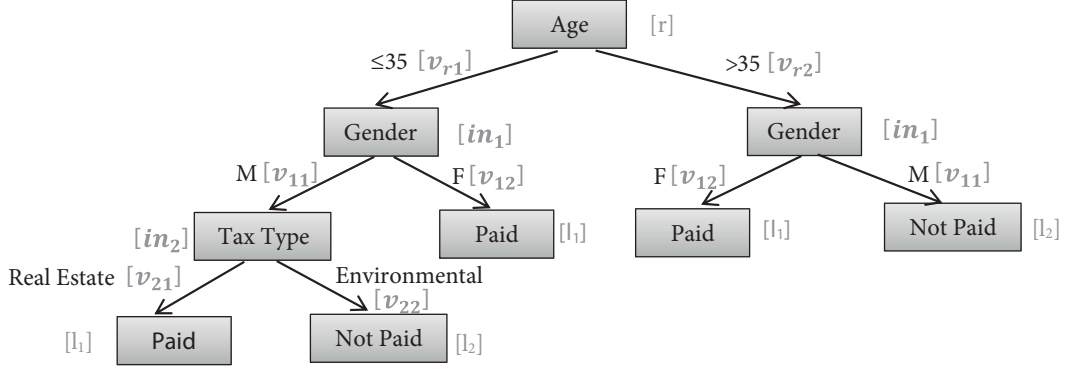


Figure 2. An example tree structure related to municipal data used in this study.

$$BS_1 = \langle ([r] = [v_{r1}]) ([in_1] = [v_{11}]) ([in_2] = [v_{21}]) ([l_1]) \rangle$$

where $[r] = \text{Age}$, $[v_{r1}] = [\leq 35]$, $[in_1] = [\text{Gender}]$, $[v_{11}] = [M]$, $[in_2] = [\text{Tax Type}]$, $[v_{21}] = [\text{Real Estate}]$, $[l_1] = [\text{Paid}]$. Depth $(l_1) = 3$ and hence there are two internal nodes in BS_1 : $[\text{Gender}]$ and $[\text{Tax Type}]$. There is one root $[\text{Age}]$ with two values (≤ 35 , > 35) and there are two leaves, class values at the same time: $[\text{Paid}]$, $[\text{Not Paid}]$. There is a total of 5 branches:

$$BS_1 = \langle ([\text{Age}] = [\leq 35]) ([\text{Gender}] = [M]) ([\text{Tax Type}] = [\text{Real Estate}]) ([\text{Paid}]) \rangle$$

$$BS_2 = \langle ([\text{Age}] = [\leq 35]) ([\text{Gender}] = [M]) ([\text{Tax Type}] = [\text{Envrn.}]) ([\text{Not Paid}]) \rangle$$

$$BS_3 = \langle ([\text{Age}] = [\leq 35]) ([\text{Gender}] = [F]) ([\text{Paid}]) \rangle$$

$$BS_4 = \langle ([\text{Age}] = [> 35]) ([\text{Gender}] = [F]) ([\text{Paid}]) \rangle$$

$$BS_5 = \langle ([\text{Age}] = [> 35]) ([\text{Gender}] = [M]) ([\text{Not Paid}]) \rangle$$

After this step, each attribute and value pair is coded with a unique integer value to apply the prefix-span algorithm in the SPM framework (SPMF) [36] to these data. For example, the pair $([r] = [v])$ is coded as “1” and $([in_1] = [v_1])$ is coded as “2”.

3.4. Sequential pattern mining

In this step, the prefix-span algorithm is applied to each TS to find all frequent sequences (FSs). SPM’s role is a little different in this approach compared to classical SPM tasks. This step provides a compact rule set, which represents the DT ideally. FSs are created for each TS by the SPM algorithm. Since the SPM is carried out on the results of the previous data mining process, this operation is clearly re-mining. Before applying the SPM, a preprocessing step involving TS clustering is necessary for accurate results.

Definition 6 *Tree sequence clustering is a preprocessing step before the SPM task, in which the sequences in the TS are grouped according to their class labels. This step is necessary, because the SPM algorithm may ignore class labels in the sequences, leading to errors in identifying similarity.*

For example, when there are two TSs:

$$TS_1 = \left\{ \begin{array}{l} BS_1 = \langle \langle [Age] = [\leq 35] \rangle \langle [Gender] = [M] \rangle \langle [Tax Type] = [Real Estate] \rangle \langle [Paid] \rangle \rangle \\ \dots \end{array} \right\}$$

$$TS_2 = \left\{ \begin{array}{l} BS_1 = \langle \langle [Gender] = [M] \rangle \langle [Tax Type] = [Real Estate] \rangle \langle [Not Paid] \rangle \rangle \\ \dots \end{array} \right\}$$

The SPM can create the following frequent sequences:

$$Frequent\ Sequence\ for\ TS_1 = \langle \langle [Gender] = [M] \rangle \langle [Tax Type] = [Real Estate] \rangle \rangle$$

$$Frequent\ Sequence\ for\ TS_2 = \langle \langle [Gender] = [M] \rangle \langle [Tax Type] = [Real Estate] \rangle \rangle$$

Hence, if there are multiple BSs in the TSs, which include $\langle [Gender] = [M] \rangle \langle [Tax Type] = [Real Estate] \rangle$ items with other distinct items like $\langle [Not Paid] \rangle$, $\langle [Paid] \rangle$, $\langle [Age] = [> 35] \rangle$, then the SPM eliminates other items and keeps only $\langle [Gender] = [M] \rangle \langle [Tax Type] = [Real Estate] \rangle$ as a FS. In this case, these two FSs are equal to each other and their similarity is 100%. However, the actual class labels for these sequences are completely different; one is labeled as $\langle [Paid] \rangle$, while the other one is $\langle [Not Paid] \rangle$. It is clear that class labels in the sequences affect the underlying similarity of DTs. In other words, class nodes in a tree have a decisive role, unlike internal nodes and the root. Thus, class labels should be protected during this SPM step. To do this, TSs related to a DT are grouped into m clusters as follows:

$$TS = \bigcup_{x=1}^m TS_x \quad (4)$$

where m is the number of classes in the dataset. For example, in the above case, there are two class labels: “Paid” and “Not Paid”. Each TS is subdivided into two groups: one of them (TS_{paid}) includes all BSs that end with $\langle [Paid] \rangle$, while the other (TS_{notpaid}) contains all sequences that end with $\langle [Not Paid] \rangle$. The SPM is applied to each sub-TS, creating sub-FSs. After this step, all sub-FSs should be compared with other sub-FSs with the same class label to calculate their similarity (pieced similarity).

Definition 7 *Pieced similarity is a similarity measure for two FSs that is calculated by adding sub-FS similarities:*

$$Sim(FS_1 + FS_2) = Sim(FS_{1c_1}, FS_{2c_1}) + Sim(FS_{1c_2}, FS_{2c_2}) + \dots + Sim(FS_{1c_m}, FS_{2c_m}) \quad (5)$$

where Sim stands for similarity and m is the number of classes, such as c_1, c_2, \dots, c_m .

3.5. Similarity calculation and K-nearest neighbor algorithm

The KNN algorithm is applied to the FS set to find the most similar k trees to a specific tree using the similarity measure given in Eq. (6). The KNN algorithm is executed many times to compare all trees with all others. This step also involves re² mining, because the KNN algorithm is applied to the results of the previous re-mining process.

Two main parameters affect the similarity of frequent sets: intersection and the length of the sets.

In pieced similarity:

$$intersection_{i,j} = \sum_{x=1}^m \#(FS_{ic_x} \cap FS_{jc_x}) \quad (6)$$

In general similarity:

$$intersection_{i,j} = \#(FS_i \cap FS_j) \quad (7)$$

where m is the number of classes. The intersection $_{i,j}$ gives the common frequent sets between two FSs (FS_i and FS_j).

At first glance, the intersection $_{i,j}$ seems to be adequate to identify similarity. However, the length of the FS sets affects similarity as well. Thus, the similarity of two sets is directly proportional to their intersection $_{i,j}$ and inversely proportional to the maximum set length.

Given FS_1 and FS_2 , where the intersection $_{1,2} = 25$, as well as FS_3 and FS_4 , where the intersection $_{3,4} = 15$, it may appear that FS_1 and FS_2 are more similar than FS_3 and FS_4 . However, if $\text{Max}(|FS_1|, |FS_2|) = 75$ and $\text{Max}(|FS_3|, |FS_4|) = 25$, it is clear that FS_3 and FS_4 are more similar. Thus, we define similarity as follows:

$$Sim(FS_i, FS_j) = 100 \times \frac{\sum_{x=1}^m \frac{\#(FS_{ic_x} \cap FS_{jc_x})}{\text{Max}(|FS_{ic_x}|, |FS_{jc_x}|)}}{m} \quad (8)$$

where FS_i is the i_{th} FS, FS_j is the j_{th} FS, and m is the number of classes. An example related to this equation is given in Section 4.3.

Figure 3 shows the pseudocode for our proposed algorithm, DTreeSim, consisting of the five steps described above. To ensure the correctness of the algorithm, we tested it on two identical DTs ($\text{DTreeSim}(DT_1, DT_1) = 100\%$) and two completely different DTs ($\text{DTreeSim}(DT_1, DT_2) = 0\%$). Our proposed method has a maximum time complexity of $O(n \times \log n)$ based on the complexities of the C4.5 ($n \times a \times \log n$), PrefixSpan ($O(n \times L)$), and the KNN with kdtree ($n \times \log n$) algorithms, where a is the number of attributes, n is the number of instances, and L is the maximum length of all transactions.

4. Experimental results

4.1. Dataset description

For the purpose of testing the accuracy of DTreeSim, the tax payments for citizens of a municipality in Turkey were analyzed over 10 years, between 2003 and 2013. Data preparation operations included data integration, reduction, cleaning, and discretization. These were applied to construct the DW from a huge database. In the data integration step, four different data tables were joined to form the citizens' tax payments data, which include nearly 720,000 instances, with nine attributes, as shown in Table 1.

Table 1. Basic dataset descriptors for local municipality data.

Attribute	Attribute values
Gender	male, female, indefinite
Marital status	single, married, widowed, indefinite
Region (on identity card)	Aegean, Central Anatolia, Marmara, Black Sea, Eastern Anatolia, Southern Anatolia, Mediterranean, Abroad
District	D1, D2, D3
Income type	environmental sanitation tax, real estate tax, water consumption, entertainment tax, etc.
Age	≤ 20 , (20–35], (35–50], (50–65], >65
Job	public, private sector, unemployed, retired
Number of children	0, 1, 2, 3, 4+
Class	paid, not paid

```

// Input: Data Warehouse (DW)
// Output: The most similar k tree pairs
Determine partitioning criteria (subject, region or time) for DW
// STEP1: Data Marting
For each transaction T in DW
    Add T to the related Data Mart (DM)
End for
// STEP2: Classification
For each DM in DW
    Apply C4.5 Classification, create a decision tree DT from DM
    Add DT to Forest F
End for
// STEP3: Tree-to-Sequence Linearization (TSL)
For each decision tree DT in F
    For each path P in DT
        Create a Branch Sequence (BS) from P
        Add BS to Tree Sequence (TS) of DT
    End for
End for
// STEP4: Sequential Pattern Mining
For each TS
    For each BS in TS
        Convert BS into SPMF form
    End for
    Apply Prefix-Span Algorithm to TS, create frequent sequences (FS)
    Add frequent sequences into FS set
End for
// STEP5: K-Nearest Neighbors
Determine k value
array[] maxSimilarityValues{};
array[,] similarPairs {{},{}}
For i = 1 to n // n is the number of trees
    For j = i+1 to n
        similarity = Calculate similarity between FSi, FSj
        If (maxSimilarityValues.Length < k)
            push similarity in maxSimilarityValues
            push {FSi,FSj} in similarPairs
        Else If (similarity > min(maxSimilarityValues) then
            maxSimilarityValues[index of min(maxSimilarityValues)] = similarity
            similarPairs[index of min(maxSimilarityValues)] = {FSi,FSj}
        End if
    End for
End for
Return similarPairs

```

Figure 3. Outline of the DTreeSim algorithm showing the five main steps.

In this experimental study, time-oriented DMs were created to investigate similarities through time. Citizens' tax payments for the period of 2003 to 2013 were partitioned by month, and each month was stored as a DM. After partitioning the DW, 12 DMs were created for each year, yielding 120 DMs for the 10-year study period. Thus, the DW stores data for all 10 years, while the DM contains only monthly data.

The purpose of this experimental study was to predict the potential income of the local municipality in advance. For example, if citizens' tax payment behaviors are similar for the months of January and July each year, then we can predict the income for next July by the end of this January. The same analysis can be applied

to region-oriented DMs. In this situation, we can find the most similar k regions for citizen behaviors and we can analyze factors that make these regions similar. For example, if citizens living in similar regions make their tax payments on time, the municipality can reward these regions by serving more services. If these regions are similar since the citizens in these regions do not pay their tax payments, then the municipality can campaign in these regions to encourage them to pay their tax payments on time.

4.2. General similarity of trees

Similarities between the trees in Forest F created at step 4.2 are calculated 7140 times ($1 + 2 + \dots + 199$) for 120 trees by applying step 3.5 (similarity calculation), after performing step 3.3 (TSL) and step 3.4 (SPM, with minimum support = 0.001).

Definition 8 Common frequent sequences (CFSs) are given by the cardinality of the intersection of two frequent sets. They are calculated with Eqs. (6) and (7) for pieced and general similarity, respectively.

The histogram of general tree similarity based on the CFS is shown in Figure 4, to illustrate their general similarity. While the y-axis of this graph shows the number of tree pairs, the x-axis includes CFSs between two trees. Each t_{th} tree is compared with all other $(120 - t)$ trees in F. The average CFS number was 15 for all tree pairs in F, while the maximum CFS number was nearly 80, for data between March 2003 and April 2003. Figure 5 displays the top 10 tree pairs that have maximum CFSs.

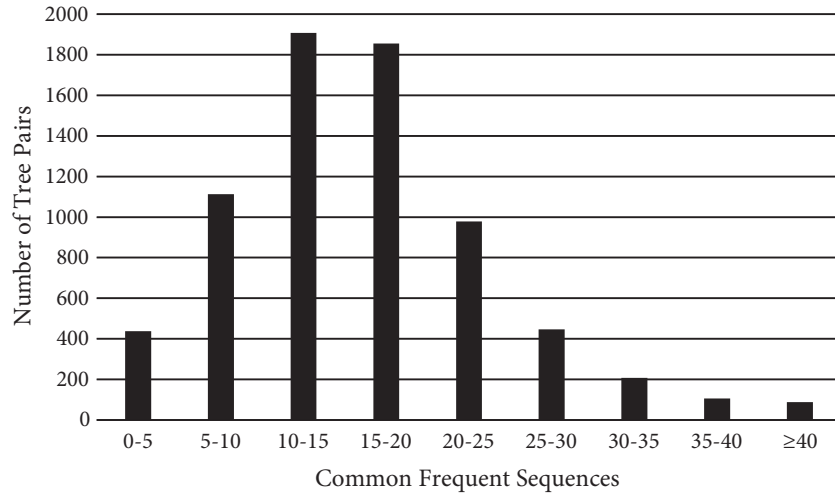


Figure 4. Histogram of general tree similarity based on the common frequent sequences (CFSs).

4.3. Pieced similarity of trees

Pieced similarity (Def. 7) was used to divide all 120 TSs into two sub-TSs (TS_{paid} , $TS_{notpaid}$), after the TSL step. The SPM step was applied to both sub-TSs to create two sub-FSs (FS_{paid} , $FS_{notpaid}$). Each sub-FS was compared with other sub-FSs that included the same class label. For example, January 2012 FS_{paid} was compared with February 2012 FS_{paid} , and January 2012 $FS_{notpaid}$ with February 2012 $FS_{notpaid}$. Similarity was calculated using the formulas given in Eq. (6). Figure 6 shows the histogram of pieced similarity given by the number of CFSs. Our results shown in Figure 7 show that the average CFS number was 40, while the maximum CFS was 200 (for March 2003 and April 2003). However, the maximum CFS number does not imply

the most similarity, since the maximum number of sequences in the pairs also affects similarity. Figure 8 displays the histogram of pieced similarity (%). When we considered the number of sequences, the most similar (64%) tree pairs were the DTs constructed for September 2010 and September 2011, as shown in Figure 9. According to the results, the DTs constructed for September and June were found similar for the 6 years. Thus, we can say that the local municipality can predict tax revenue for September at the end of June.

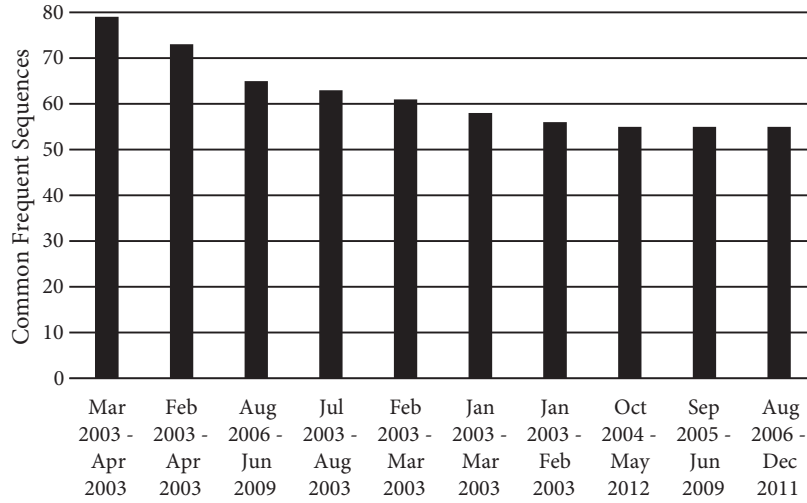


Figure 5. General tree similarity based on the common frequent sequences (CFSs) (top 10).

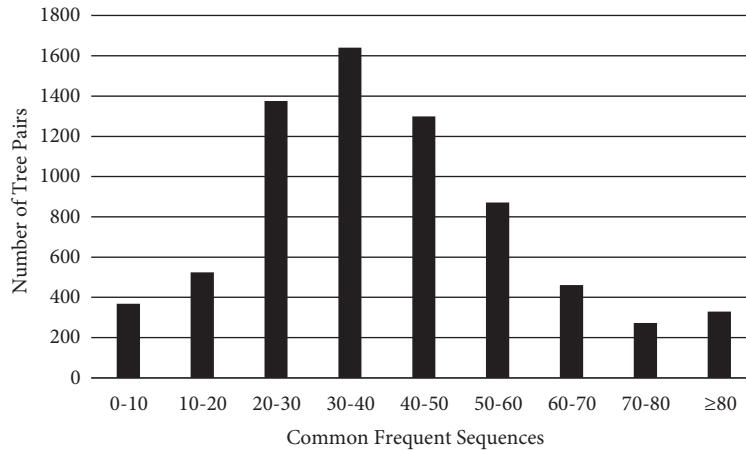


Figure 6. Histogram of pieced similarity given by the number of common frequent sequences (CFSs).

In another example, the DT with 62 sequences constructed for September 2010 was compared with the tree with 63 sequences for September 2011. The number of common sequences for these trees was 37, so the similarity for the *paid* class can be calculated as $37/\text{Max}\{62,63\}$, equal to 58%. When the same operation is repeated for the *notpaid* class, the result was 68%. The final similarity (64%) was calculated by averaging the results for each class.

$$\#(\text{September 2010 } FS_{\text{paid}}) = 62$$

$$\#(\text{September 2011 } FS_{\text{paid}}) = 63$$

$$\#(\text{September 2010 } FS_{\text{paid}} \cap \text{September 2011 } FS_{\text{paid}}) = 37$$

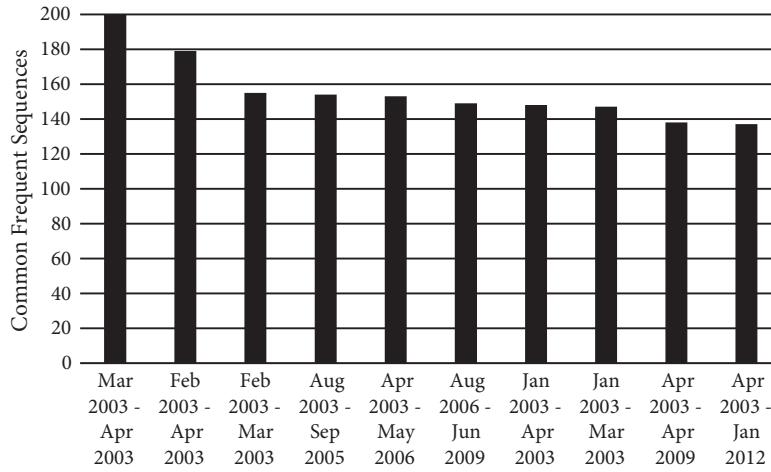


Figure 7. Pieced similarity given by the number of common frequent sequences (CFSs) (top 10).

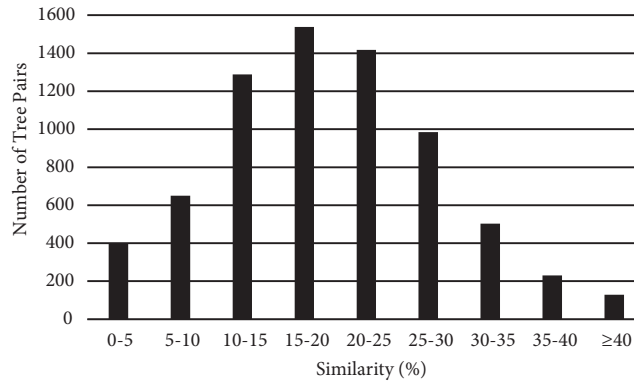


Figure 8. Histogram of pieced similarity given by percentage values.

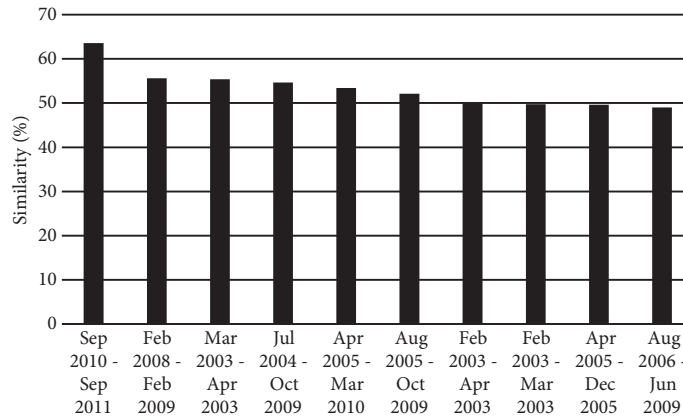


Figure 9. Pieced similarity given by percentage values (top 10 similarity (%)).

$$Sim (September 2010 FS_{paid}, September 2011 FS_{paid}) = \frac{37}{Max(62, 63)} \cong 58\%$$

$$\# (September 2010 FS_{notpaid}) = 54$$

$$\#(\text{September 2011 } FS_{\text{notpaid}}) = 57$$

$$\#(\text{September 2010 } FS_{\text{notpaid}} \cap \text{September 2011 } FS_{\text{notpaid}}) = 39$$

$$\text{Sim}(\text{September 2010 } FS_{\text{paid}}, \text{September 2011 } FS_{\text{paid}}) = \frac{39}{\text{Max}(54, 57)} \cong 68\%$$

$$\text{Sim}(\text{September 2010 } FS, \text{September 2011 } FS) = \frac{0.58 + 0.68}{2} \cong 64\%$$

4.4. Comparison of general similarity and pieced similarity

Definition 9 The average common frequent sequences (ACFSs) is given by the average cardinality of the intersection of two frequent sets for all trees:

$$ACFS = \frac{\sum_{x=1}^n CFS_x}{n} \quad (9)$$

where n is the total number of trees.

The ACFSs obtained using both similarity approaches are shown in Figure 10. According to our results, pieced similarity supplies 40 ACFSs, which is more than twice the ACFSs generated by general similarity.

Definition 10 Comparison number is the number of comparison operations carried out on trees to have the most similar trees.

Based on Figures 4 and 6, we can say that the correlation among the general similarity-based CFS results and pieced similarity-based CFS results is very high, since the sizes of the trees are smaller and more similar BSs are expected to be grouped under different class labels. However, the main disadvantage of using pieced similarity is the number of comparisons it generates, since the number of sequence sets doubles compared with general similarity, as shown in Figure 11.

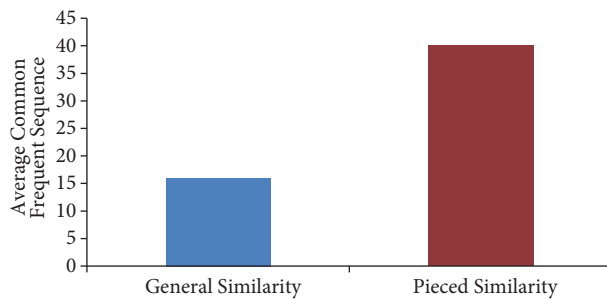


Figure 10. Average common frequent sequence (ACFS) numbers used to compare general similarity and pieced similarity techniques.

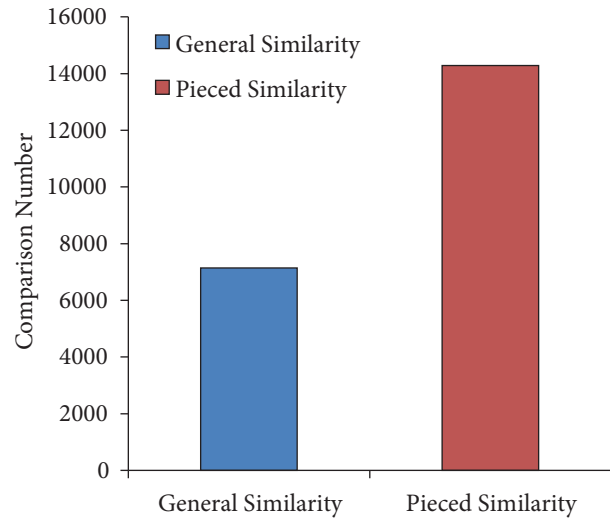


Figure 11. The number of comparison operations carried out on trees for general similarity and pieced similarity techniques.

4.5. Comparison of DTreeSim with existing approaches

DTreeSim was compared with two existing approaches: DTSim [22] and Sim_{d1,d2} [23,24]. Sample small trees used to compare similarity approaches are shown in Figures 12 and 13. The same trees, except with different attribute values (branch values), were used in [23,24]. We also needed to add the attribute values to calculate similarity using our approach.

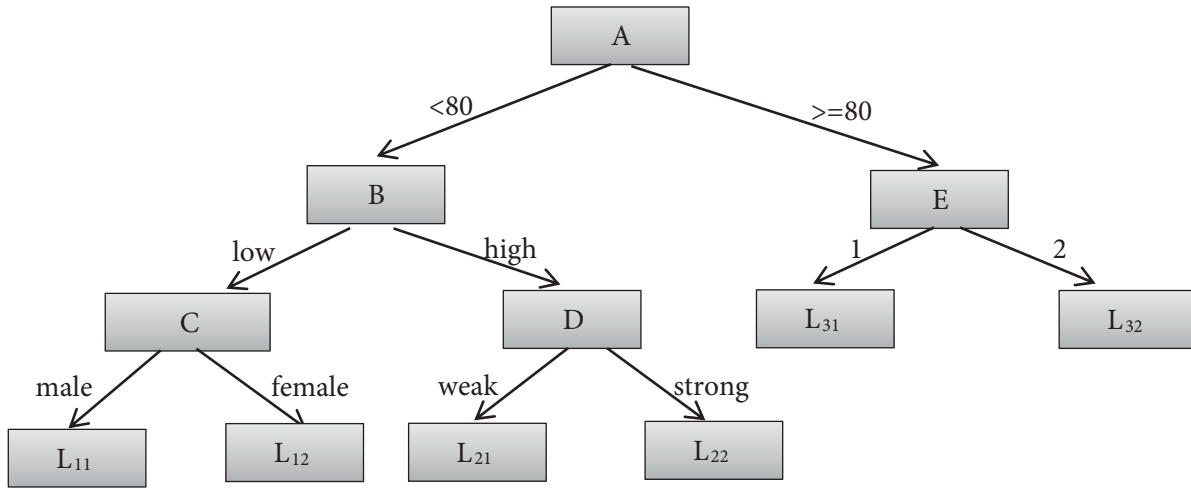


Figure 12. Structure for decision tree 1 [23,24].

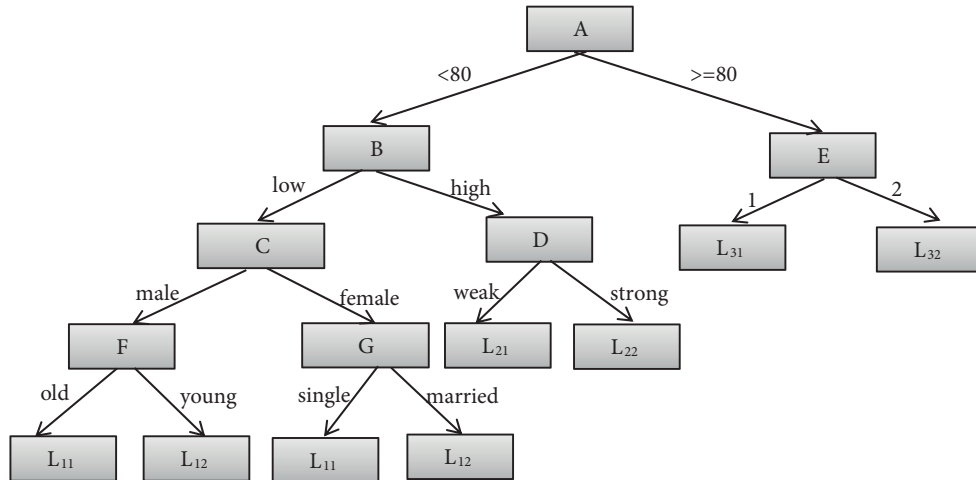


Figure 13. Structure for decision tree 2 [23,24].

It is possible to classify the degree of similarity in five types: very similar ($\geq 80\%$), similar ($< 80\%$ and $\geq 60\%$), neither similar nor dissimilar ($< 60\%$ and $\geq 40\%$), dissimilar ($< 40\%$ and $\geq 20\%$), and very dissimilar ($< 20\%$). Similarity values and degrees between Tree 1 and Tree 2 are shown in Table 2. Maximum similarity was calculated using Sim_{d1,d2} [23,24], giving 91.66%. This reflects the fact that Perner [23,24] ignored leaf (class) and attribute values. DTSim [22] had the minimum similarity value, with 66%. This is because DTSim compares branches one-to-one. In this algorithm, if there is one different node on a branch, then these branches are deemed different, even though the nodes are in the same order. Since our approach finds FSs for each

tree and compares these FSs rather than full branches, our similarity is greater than that of DTsim [22], but smaller than that of $Sim_{d1,d2}$ [23,24], because we consider both leaf (class values) and attribute values. Thus, the proposed approach produces more reliable results with respect to the existing approaches.

Table 2. Similarity values obtained using different approaches.

Approach	Similarity values (%)	The degree of similarity
DTreeSim	78.30	Similar
DTsim [22]	66	Similar
$Sim_{d1,d2}$ [23,24]	91.66	Very similar

4.6. The application of DTreeSim to benchmark data with bootstrapping

Since decision trees are known to be unstable and small perturbations in the dataset might yield significant differences in the resulting tree, we also used an ensemble learning technique [37]. Although there is a framework, inTrees (interpretable trees) [38], that creates a random forest with multiple trees, we applied bootstrapping [39] with the fraction $(1 - 1/e \approx 63.2\%)$. We constructed randomly sampled DMs for each partition, obtained a different DT for each sample, and measured the similarities among DTs. In this case, DTreeSim was applied to a dataset, “Car Evaluation”, from the UCI Machine Learning Repository (<http://archive.ics.uci.edu/ml/>). This dataset includes 1728 instances, with six attributes, as shown in Table 3. Bootstrapping was applied to this dataset to form five different DMs, and also different DTs. Using DTreeSim, we found that the 3rd and 5th trees were most similar. When we evaluated our result by checking classification accuracy for each tree, we saw that the 3rd and 5th trees were in fact closest to each other. This result supports the reliability of the proposed approach.

Table 3. Car evaluation dataset descriptors for data from the UCI Machine Learning Repository.

Attribute	Attribute values
Buying	vhigh, high, med, low
Maint	vhigh, high, med, low
Doors	2, 3, 4, 5, more
Persons	2, 4, more
Lug-Boot	small, med, big
Safety	low, med, high
Class	unacc, acc, good, vgood

5. Conclusions and future work

This study proposes DTreeSim as a new approach to evaluate the similarity of DTs and to find the k-most similar trees. To achieve this purpose, three data mining techniques (classification, SPM, and KNN), involving five steps, were proposed. In contrast to previous studies, SPM is used to find compact and pruned forms of frequent branches in DTs. With this new approach, the paths of DTs are considered as sequences, and their branches are considered as sequence items. In the final step, the frequent sequences extracted by the SPM are used by the KNN algorithm with a new distance measure to find similarities. While the SPM technique is applied to classified data, the KNN algorithm is carried out on the results of the SPM, such that a re-mining process is performed twice. We have called this re²mining.

The applicability of our new approach is shown using data that were collected on citizens’ tax payments by a local municipality in Turkey over a 10-year period. Two novel similarity techniques (general and pieced)

were applied to find the k-most similar trees. Experimental results show that our approach can be effectively used to find similarity in DTs. It could clearly be applied to different problems in different areas.

In the future, other SPM algorithms (e.g., SPADE, GSP) will be applied to find frequent sequences. In addition, an incremental approach will be developed to enable results to be updated as new data become available.

References

- [1] Quinlan JR. C4.5: Programs for Machine Learning. San Francisco, CA, USA: Morgan Kaufmann Publishers, 1993.
- [2] Quinlan JR. Improved use of continuous attributes in C4.5. *J Artif Intell Res* 1996; 4: 77-90.
- [3] Wu X, Kumar V, Quinlan JR, Ghosh J, Yang Q, Motoda H, McLachlan GJ, Ng A, Liu B, Yu PS et al. Top 10 algorithms in data mining. *Knowl Inf Syst* 2008; 14: 1-37.
- [4] Dudani SA. The distance-weighted k-nearest-neighbor rule. *IEEE T Syst Man Cyb* 1976; 6: 325-327.
- [5] Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu M. PrefixSpan: Mining sequential patterns by prefix-projected growth. In: 17th International Conference on Data Engineering; 2–6 April 2001; Heidelberg, Germany, New York, NY, USA: IEEE. pp. 215-224.
- [6] Fletcher S, Islam MZ. An anonymization technique using intersected decision trees. *Journal of King Saud University – Computer and Information Sciences* 2015; 27: 297-304.
- [7] Dogra IS. Improving and maintaining prediction accuracy in agent based modeling systems under dynamic environment. MSc, University of Windsor, Windsor, Canada, 2014.
- [8] Papagelis A, Kalles D. GATree: Genetically evolved decision trees. In: 12th IEEE International Conference on Tools with Artificial Intelligence; 13–15 November 2000; Vancouver, BC, Canada. New York, NY, USA: IEEE. pp. 203-206.
- [9] Telaar D, Fuhs MC. Accent- and speaker-specific polyphone decision trees for non-native speech recognition. In: Interspeech 2013, 14th Annual Conference of the International Speech Communication Association; 25–29 August 2013; Lyon, France. pp. 3313-3316.
- [10] Kumari A, Prasad U, Bala PK. Retail forecasting using neural network and data mining technique: a review and reflection. *Int J Emerg Trends Technol Comp Sci* 2013; 2: 266-269.
- [11] Fiol-Roig G, Miro-Julia M, Isern-Deya AP. Applying data mining techniques to stock market analysis. *Adv Intel Soft Comput* 2010; 71: 519-527.
- [12] Thangamani M, Thangaraj P, Bannari. Automatic medical disease treatment system using datamining. In: 2013 International Conference on Information Communication and Embedded Systems; 21–22 February 2013; Chennai, India. New York, NY, USA: IEEE. pp. 120-125.
- [13] Souza FT, Wang Z. A data mining approach to predict mass movements induced by seismic events in Sichuan, China. In: 2010 Sixth International Conference on Natural Computation; 10–12 August 2010; Yantai, China. New York, NY, USA: IEEE. pp. 1172-1177.
- [14] Olaiya F, Adeyomo AB. Application of data mining techniques in weather prediction and climate change studies. *Int J Inform Eng Electron Bus* 2012; 4: 51-59.
- [15] Solomon S, Nguyen H, Liebowitz J, Agresti W. Using data mining to improve traffic safety programs. *Ind Manage Data Syst* 2006; 106: 621-643.
- [16] Spielman SE, Thill J. Social area analysis, data mining and GIS. *Comput Environ Urban* 2007; 32: 110 - 122.
- [17] Ntoutsis I, Kalousis A, Theodoridis Y. A general framework for estimating similarity of datasets and decision trees: exploring semantic similarity of decision trees. In: SIAM International Conference on Data Mining; 24–26 April 2008; Atlanta, GA, USA. Philadelphia, PA, USA: SIAM. pp. 810-821.

- [18] Zhang X, Jiang S. A splitting criteria based on similarity in decision tree learning. *J Softw* 2012; 7: 1775-1782.
- [19] Islam MZ, Barnaghi PM, Brankovic L. Measuring data quality: predictive accuracy vs. similarity of decision trees. In: 6th International Conference on Computer & Information Technology; 19–21 December 2003; Dhaka, Bangladesh. pp. 457-462.
- [20] Last M, Maimon O, Minkov E. Improving stability of decision trees. *Int J Pattern Recogn* 2002; 16: 145-159.
- [21] Yoshida T. Term usage difference in a single dataset through collaborative registration. In: Columbus AM, editor. *Advances in Psychology Research*. Hauppauge, NY, USA: Nova Science Publishers, 2008. pp. 153-170.
- [22] Peahringer B, Witten IH. *Improving Bagging Performance By Increasing Decision Tree Diversity*. Boston, MA, USA: Kluwer Academic Publishers.
- [23] Perner P. How to compare and interpret two learnt decision trees from the same domain? In: 27th International Conference on Advanced Information Networking and Applications Workshops; 25–28 March 2013; Barcelona, Spain. New York, NY, USA: IEEE. pp. 318-322.
- [24] Perner P. How to interpret decision trees? *Lect Notes Comput Sc* 2011; 6870: 40-55.
- [25] Pekerskaya I, Pei J, Wang K. Mining changing regions from access-constrained snapshots: a cluster-embedded decision tree approach. *J Intell Inf Syst* 2006; 27: 215-242.
- [26] Islam MZ. Privacy preservation in data mining through noise addition. PhD, University of Newcastle, Newcastle, Australia, 2008.
- [27] Papagelis A, Kalles D. Breeding decision trees using evolutionary techniques. In: 18th International Conference on Machine Learning; 28 June–1 July 2001; Williamstown, MA, USA. Burlington, MA, USA: Morgan Kaufmann Publishers. pp. 393-400.
- [28] Ma S, Tang S, Yang D, Wang T, Han J. Combining clustering with moving sequential pattern mining: a novel and efficient technique. *Lect Notes Artif Int* 2004; 3056: 419-423.
- [29] Exarchos TP, Papaloukas C, Lampros C, Fotiadis DI. Mining sequential patterns for protein fold recognition. *J Biomed Inform* 2008; 41: 165-179.
- [30] Tseng VS, Lee CH. Effective temporal data classification by integrating sequential pattern mining and probabilistic induction. *Expert Syst Appl* 2009; 36: 9524-9532.
- [31] D'Silva MR, Vora D. Intelligent recommendation system using clustering and closed sequential pattern mining. *International Journal of Computer Science Engineering and Information Technology Research* 2014; 4: 133-140.
- [32] Deng H, Runger G, Tuv E, Bannister W. CBC: An associative classifier with a small number of rules. *Decis Support Syst* 2014; 59: 163-170.
- [33] Ma Z, Kaban A. K-Nearest neighbours with a novel similarity measure for intrusion detection. In: 13th UK Workshop on Computational Intelligence; 9–11 September 2013; Guildford, UK. New York, NY, USA: IEEE. pp. 266-271.
- [34] Liu J, Yin J. Towards efficient data re-mining (DRM). *Lect Notes Artif Int* 2001; 2035: 406-412.
- [35] Demiriz A, Ertek G, Atan T, Kula U. Re-mining item associations: methodology and a case study in apparel retailing. *Decis Support Syst* 2011; 52: 284-293.
- [36] Fournier-Viger P, Gomariz A, Gueniche T, Soltani A, Wu C, Tseng VS. SPMF: A Java open-source pattern mining library. *J Mach Learn Res* 2014; 15: 3389-3393.
- [37] Seni G, Elder JF. *Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions*. San Rafael, CA, USA: Morgan and Claypool Publishers, 2010.
- [38] Deng H. Interpreting Tree Ensembles with inTrees. Technical Report. arXiv:1408.5456, 2014.
- [39] Ishwaran H, Rao, J. Decision trees, advanced techniques in constructing. In: Kattan M, editor. *Encyclopedia of Medical Decision Making*. London, UK: SAGE Publishing, 2009. pp. 328-332.