




Article

Guided Decision Tree: A Tool to Interactively Create Decision Trees Through Visualization of Subsequent LDA Diagrams

Miguel A. Mohedano-Munoz ¹, Laura Raya ² and Alberto Sanchez ^{3,*}

¹ Estación Biológica de Doñana (CSIC), 41092 Seville, Spain; ma.mohedano@ebd.csic.es

² Faculty of Engineering, U-tad, 28290 Madrid, Spain; laura.raya@u-tad.com

³ Department of Computer Science & Statistics, Universidad Rey Juan Carlos, 28933 Madrid, Spain

* Correspondence: alberto.sanchez@urjc.es

Featured Application: The developed tool can be applied to motion recognition in individuals to accurately identify the exercises they are performing. This application is particularly useful in fields such as sports science, physical therapy, fitness tracking, and Virtual Reality, where understanding and monitoring exercise routines and movement are crucial.

Abstract: Decision trees are a widely used machine learning technique due to their ease of interpretation and construction. This method allows domain experts to learn from raw data, but they cannot include their prior knowledge in the analysis due to its automatic nature, which implies minimal human intervention in its computation. Conversely, interactive visualization methods have proven to be effective in gaining insights from data, as they incorporate the researcher's criteria into the analysis process. In an effort to combine both methodologies, we have developed a tool to manually build decision trees according to subsequent visualizations of data mapping after applying linear discriminant analysis in combination with Star Coordinates in order to analyze the importance of each feature in the separation. The nodes' information contains data about the features that can be used to split and their cut-off values, in order to select them in a guided manner. In this way, it is possible to produce simpler and more expertly driven decision trees than those obtained by automatic methods. The resulting decision trees reduces the tree size compared to those generated by automatic machine learning algorithms, obtaining a similar accuracy and therefore improving their understanding. The tool developed and presented here to manually create decision trees in a guided manner based on the subsequent visualizations of the data mapping facilitates the use of this method in real-world applications. The usefulness of this tool is demonstrated through a case study with a complex dataset used for motion recognition, where domain experts built their own decision trees by applying their prior knowledge and the visualizations provided by the tool in node construction. The resulting trees are more comprehensible and explainable, offering valuable insights into the data and confirming the relevance of upper body features and hand movements for motion recognition.

Keywords: multivariate visualization; decision trees; visual data mining; linear discriminant analysis; motion recognition



Citation: Mohedano-Munoz, M.A.; Raya, L.; Sanchez, A. Guided Decision Tree: A Tool to Interactively Create Decision Trees Through Visualization of Subsequent LDA Diagrams. *Appl. Sci.* **2024**, *14*, 10497. <https://doi.org/10.3390/app142210497>

Academic Editor: Antonio Fernández-Caballero

Received: 20 September 2024

Revised: 3 November 2024

Accepted: 9 November 2024

Published: 14 November 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In recent years, artificial intelligence (AI), and more specifically machine learning, has made great progress and awakened interest in different application fields alongside the increase in data recording [1]. Nevertheless, machine learning methods are sometimes understood as black box processes. This lack of transparency can be a problem for their application in some fields that involve high-stakes decisions [2], which pushes researchers to find explainable models to apply in these situations. Explainable models have the advantage of allowing domain experts to include their previous knowledge and their criteria in the model's construction.

A way to include domain experts in the model-building process is through interactive visualization. Interactive visualization has been proven an efficient way of improving the knowledge obtained by the domain experts in data analysis processes. User interaction can improve and modify the models obtained, increase researchers' confidence in the model, and give them a better understanding of the processes underlying the data.

Machine learning algorithms can be categorized as supervised learning and unsupervised learning, according to their target and data requirements. While clustering is an unsupervised technique used to group samples by a similarity criterion, classification tasks fall within supervised learning techniques, and their aim is to predict the categorical value of unlabeled elements from a group of labeled elements [3]. A classifier is then a machine learning model that, given a sample with different variables, predicts a label from the samples learnt in the training process from a specific pre-labeled, training dataset. One of the most well-known methods employed in classification tasks is decision trees. Classification trees successively divide the dataset according to the fulfillment of different criteria until they define the conditions that characterize the different classes present in the training dataset [4].

Specifically, decision trees define the division criteria according to automatic algorithms, which try to maximize the gaining of information on every subset construction or the value of the Gini impurity measure. Nevertheless, this kind of algorithm does not allow researchers to include their domain knowledge and makes it difficult for them to understand the model's inner workings [5,6]. Moreover, for complex datasets or some algorithms' options, the tree might be too big or complex to grasp at a glance how the classification is made [7]. In some contexts, this may limit real-world applications or completely hinder them [8].

For this reason, we previously created a method to manually build decision trees based on the Star Coordinates representation of the LDA mapping of clinicians' data [9]. In this paper, we present the development of an interactive visual tool, namely guided decision tree (GDT), that meets all the requirements for interactive decision tree building for general purposes. GDT enables domain experts to manually create decision trees in a simple, guided, and visual manner. As a main contribution, this approach not only enhances the interpretability of the decision trees but also integrates domain experts' insights into the model-building process. Specifically, in this case, we focus its usage on applications that have proven to be highly challenging and very diverse contexts in the field of Artificial Intelligence, including human pose estimation from videos [10,11]. In particular, we concentrate on motion recognition as a case study of GDT, since it can be applied in various fields, such as healthcare for rehabilitation and physical therapy [12], sports science for improving performance and preventing injuries [13], and human-computer interaction for developing intuitive interfaces for virtual reality and gaming [14].

The paper is structured as follows. Section 2 shows the state of the art of different methods of building decision trees and human-supervised machine learning methods. Section 4 details the software architecture of the tool. Section 5 shows a case study in which the tool is used by domain experts to build a classifier from a motion recognition dataset. Finally, Section 6 details the conclusions obtained and analyzes possible future works that may continue the tool's development.

2. Related Works

Visualization techniques allow users to intuitively extract data relationships based on their domain knowledge and understand the derived model. Combined with other non-visual techniques, they can increase the effectiveness of the knowledge extraction process [5,15]. In a machine learning pipeline, there are three main stages: before, during, and after model building [16]. Visualization plays a different role depending on which of these it is applied to. Before model building, its main aim is to allow the researcher to better understand the data for either preparation or feature extraction. During the building, it may allow for a better understanding of the model's inner workings and help

diagnose and/or steer the model. Finally, after the model is built, visualization allows final users to understand the results and promote trust in them [17]. In particular, there are several methods that have been used to represent decision trees, such as outline views, node link diagrams, and tree maps, among others [5]. Some tools have been developed that combine several visualization techniques with decision trees at different points of the model's training and selection.

In the tool [18] described by Nguyen et al., several visualization algorithms are applied to node link diagrams—standard, tightly-coupled, fish-eye and their own 2.5D tree visualization. Its main objective is to provide the user with a better way to inspect the models in an iterative process to thus determine the best one. It also allows for visualization during model development to select features to be used for branch pruning, although it only provides information about the model structure and not about the data themselves.

Neville et al. propose a system [19] with node link and tree map views with modifications to convey more information visually, along with feature importance and different possible models. This gives the analyst insight into the model and facilitates validation, as it portrays the number of samples and class purity in a way that can be quickly seen at a glance.

A first approach to the historical state of the art of interactive tree construction is the review written by Liu and Salvendy [5], which analyzed the main programs used to build and prune decision trees and make use of visualization techniques to improve knowledge extraction. From the software reviewed in it, only Alice, EM and Weka allow for interactive tree building. As a form of support for interactive building, Alice and EM supply the users with a tabular view of the predictive power of input attributes as algorithmic assistance. Data are displayed as support for domain experts in node construction. This representation differs according to the application. Alice shows the data distribution of each attribute corresponding to the decision node by class, EM shows the distribution of the selected split attribute by class, and Weka shows bar charts for each input attribute and a 2D scatter plot of two chosen split attributes. In [5] the development of decision tree tools, visualization techniques have continued and new tools have emerged. Stiglic et al. state that simpler trees generate greater understanding for domain experts in biologic science [20]. For this purpose, they promote the use of a pruning and tuning method based on visualization and branch replacement in order to simplify the tree. The premise of this form of tree construction is to fit the top-down representation of the tree into a single screen to maximize knowledge extraction from domain experts.

A thorough subsequent review on how visualization has been applied to rule-based classifiers, conducted by Strebb et al. [21], analyzed 152 publications on task-focused visualizations. This review presents many tools that enable interaction with visualizations for model building and evaluation. However, none of the tools in this review provide a direct application of the method in [9]. Below, we detail some interesting proposals for interactive decision tree construction in the literature.

BaobabView's proposal [22] guides the user in the non-binary tree construction through coordinated views of data distribution, data distribution per feature, a tree overview, and the representation of the confusion matrix on the training set. The user is provided with the opportunity to grow, optimize, prune, and analyze the resulting tree in every step with algorithmic support. The graphic structure of the tree is based on a top-down structure, with the nodes linked by bezier lines to mark the class separation in every step. It also has a specific mode in which to prune nodes where class separation is hard. This tool can also be used for exploratory data analysis and allows the growth of trees from defined nodes. Although we value aspects of this proposal, we consider that extracting information through data distributions is more difficult than extracting it from the visualization implemented in our tool.

PaintingClass [23] is a tool for interactive decision tree construction based on star coordinate representations. From the top-down overview of the tree, the user can split the records in subsets and define subsets from the drawn areas. The goal of this methodology is

not to use the trees as classifiers, but to use the tool as part of the exploratory data analysis process. This tool shares some similarities with our approach in its process, but it differs in its objectives, since we apply a dimensionality reduction method in the process that can be considered a linear classifier. These different goals imply that we require a labeled dataset to construct the tree, as opposed to PaintingClass and their exploratory data analysis.

3. Proposal Basics

The main reason we developed this software is to provide researchers with a specific tool to manually build decision trees from their own datasets. GDT relies on the visualization of the linear discriminant analysis (LDA) [24] mapping, according to a Star Coordinates system as support for decision tree construction.

Star Coordinates [25–27] (SC) is a multivariate visualization system based on radial axis onto which data points are linearly projected. It is focused on graphic representation more than numerical analysis. The system’s axes are created as follows. Initially, each feature present in the dataset is assigned an axis of equal size and distribution that begins at the origin. Then, the unit vectors are calculated by mapping the minimum value to the origin and the maximum to the end of the corresponding axis. To represent a high-dimensional data point in the new 2D system, the value of each feature is multiplied by the corresponding unit vectors, and the sum of the resulting vectors returns the linear projection in the mapping (see Figure 1). Users can then carry out single or multi-axis rotation and scaling to find views that better suit their visualization needs.

(0.5, 0.3, 0.3, 0.1, 0.2, 0.5, 0.5, 0.2)

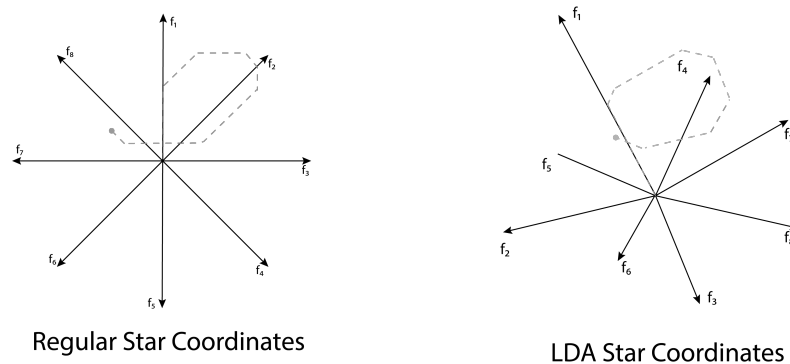


Figure 1. Use of Star Coordinates to represent records. The first case shows how to apply it when all the features have the same weight and arbitrary orientation. The second case shows the application after use of the transformation matrix obtained from the LDA algorithm to define the feature weight and orientation in the mapping.

This property can be used to represent an LDA algorithm, as shown in Figure 1, and place the points according to this mapping. LDA tries to maximize the separation between the elements belonging to different classes while minimizing intra-class dispersion. It can be used as a linear classifier or as a method of dimensional reduction, where the maximum number of dimensions to represent it is the minimum between the number of classes and the number of features.

The representation of the axis in Star Coordinates provides the researchers with information about the importance of a feature in the linear transformation through their length and orientation [28]. Therefore, the axes of the LDA mapping in Star Coordinates can be used as guide by the researchers to choose the split feature when they are creating a decision node in an iterative process. Figure 2 shows this process applied in every node.

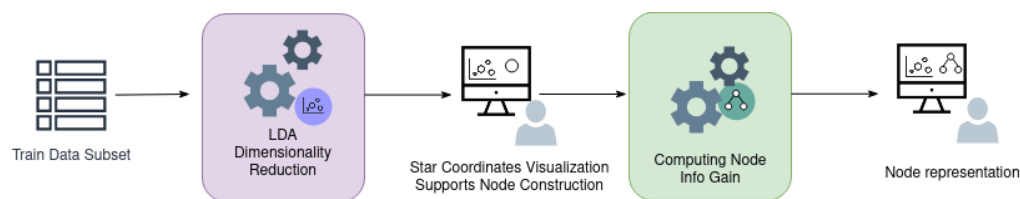


Figure 2. Schematic of user interaction for node building. The method proposes applying LDA in every target node subset, represent it on Star Coordinates, and decide the partitioning feature from that visualization.

4. Software Architecture

Domain experts and researchers defined the following requirements to interactively analyze data and manually create decision trees:

- It has to allow domain experts to make a feature selection before starting the analysis.
- It has to split the dataset into training and testing sets according the percentage supplied by the user.
- The tree viewer has to show at first sight the purity of the nodes, and it has to automatically reorganize the layout to display the tree structure as a way to maintain the decision tree in a comprehensive way in every step.
- The tree interaction has to enable the domain experts an to delete nodes, create decision nodes, or define nodes as leaf nodes.
- The decision support visualization has to include the data and a representation for the feature weight in the model, presenting the data details on demand.
- It must offer an appropriate way to represent the dimensionality reduction when there are only two classes left in the interest node, due to LDA limitations.
- The tool has to automatically determine the cutoff values according the entropy values, as described in the method.
- Lastly, the domain expert has to be able to compare with automatic trees and easily export the results.

To fulfill these requirements, we have developed GDT, a pure python visualization web tool. GDT is designed with a client–server architecture, as shown in Figure 3, and is structured in three layers: presentation, service, and logic. It employs the Python Plotly DashAPI to manage the presentation and the service layer. Plotly Dash is an open-source product, integrated with Python and R, which facilitates the creation of reactive web applications focused on data analysis and visualization. Dash applications have a Flask web server whose interface is represented by the use of the JavaScript library React.js. It is complemented with the Dash Bootstrap Components library to include Dash in the bootstrap style. The use of Dash allows users to create responsive web applications without needing knowledge of JavaScript programming. The web conception of these applications makes them multi-platform and ready for use on mobile devices.

Calculations and data management are performed in the logic layer. For this, we use pandas [29] for managing data in dataframe format, scikit-learn [30] for applying machine learning algorithms, and modifications to the Python C4.5 [31] algorithm for building the interactive tree. The use of the scikit-learn library is common when conducting machine learning analysis in the Python environment. For the creation of automatic decision trees, scikit-learn implements the Classification And Regression Trees (CARTs) algorithm using the Gini impurity or entropy as a split criterion. It creates a classifier using the features of the dataset and thresholds calculated by the algorithm and classifying criterion until all leaves are pure or no more splits are possible, unless a maximum depth or minimum number of samples is established. This approach could lead to overfitting [4], so it is possible to pass a value to perform cost–complexity pruning. In classification problems, it is common for the data to be imbalanced among classes. To address this issue, scikit-learn provides the ability to adjust the weights assigned to each class [32], ensuring a

more balanced and accurate classification. However, we must note that one of the biggest challenges of decision trees applied to real-world problems is that automatically generated trees may be too big and/or complex for users to understand [7].

The webtool code is composed of three modules; the first module deals with the user interaction, the second takes care of tasks related to the visualizations and dimensionality reduction, and the last is responsible for the tree construction and calculation of the information gain in every split.

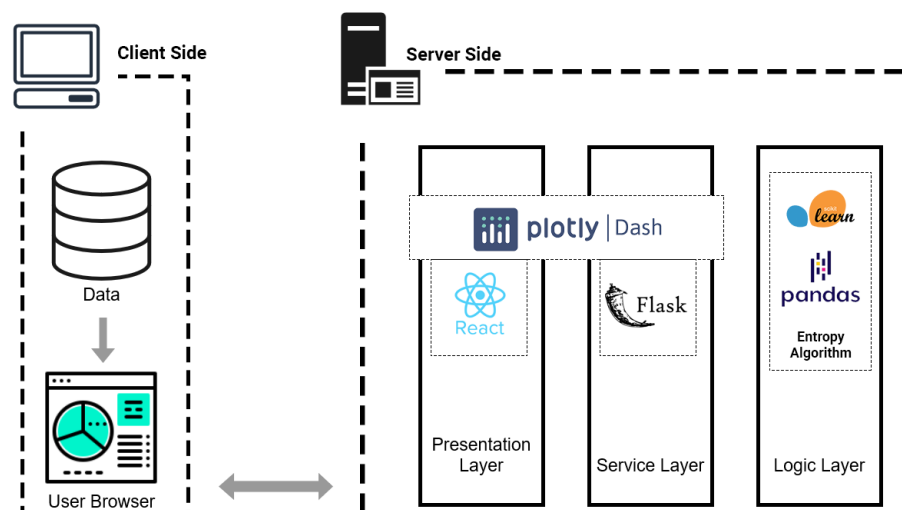


Figure 3. Software architecture diagram. GDT presents a client–server architecture, based on Python libraries. Plotly Dash manages the presentation and service layers. The logic layer includes the scikit-learn and pandas libraries and the modification of entropy algorithms.

The following subsections will detail how the tool enables the necessary interactions to fulfill the requirements outlined at the beginning of this section.

4.1. Data Management and Node Structure

The tool employs pandas objects to manage the datasets throughout all the stages, stored as tree-structured JSON objects. The original dataset remains unchanged across the whole exploratory data analysis. Data are normalized to the 0–1 range in node representation, so that visualizations of data and features can be compared with each other. The dataset is split into training and testing sets. In every node representation, the training dataset is queried according to the requirements of the node and the parent nodes in the path to build the corresponding subset. These subsets are used to build the node visualizations.

For management of the node construction, the information of the tree is stored in a JSON object. This dictionary includes an entry with the data used, the features presented in the analysis and the RGB values for each class, an entry with a summary of click interactions with the tree, and an entry to detail the nodes' construction. In the JSON object, at the 'node' entry, each node in the tree has a subentry. These subentries collect the following information about the node: the node ID, the father node ID, the path from the root node, the values of purity in the node, the partitioning feature and the threshold value, the size of the node subset regarding the training size, and if the node is considered a leaf, decision, or root. This information is employed throughout the whole construction process to build the active node subset in every interaction. It is also used to build the classifier to evaluate performance.

4.2. Interface

We have developed the graphic user interface presented in Figure 4. This interface allows domain experts to interact with the application and, through interaction, visually extract knowledge. Users can modify the parameters of the analysis, manually build

the tree, and select the automatic tree parameters. The interface presents four areas of interaction: (i) upload and tree builder, (ii) tree viewer, (iii) classifier parameters, and (iv) classifier performance summary.

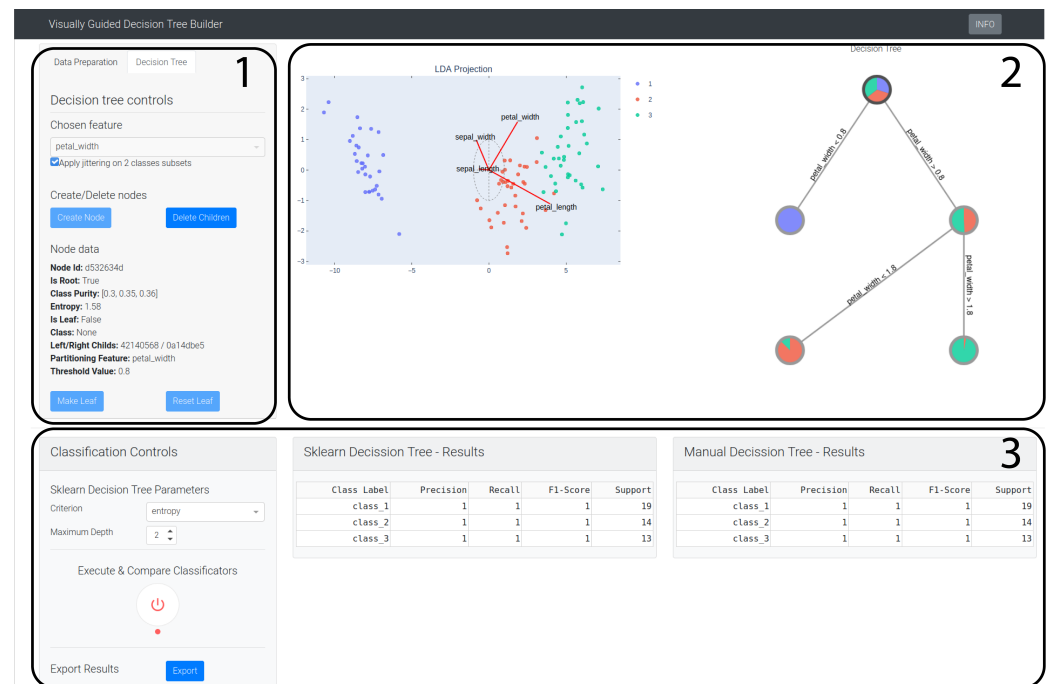


Figure 4. GUI. The work space is divided into three areas: (1) management of the user interaction in the data upload, feature selection, and the manual tree creation; (2) interactive tree and LDA mapping viewer; (3) automatic decision trees and comparing performance settings.

4.2.1. Upload and Tree Builder

The upload and tree building area is divided into two tabs: data preparation and decision tree control. We allow users to load .csv files and define the character used as a separator in the data file through the data preparation tab (Figure 5a). Users can also define the feature used as a label in the supervised learning and the dimensionality reduction method; they can also define the features used in the analysis. The size of the training set and the testing set is also defined in this area through a knob controller. Once all the fields are set, data are split into the training and testing set. Then, we represent the root node as the beginning of the tree's construction and its corresponding initial SC representation of the LDA mapping.

The decision tree control (see Figure 5b) allows users to choose the actions related to the selected node in the viewer. Every interaction in this menu updates the representation in the tree viewer, except the 'jittering in two classes' checklist, which only affects the support decision visualization. From this tab, the user can define the node as a decision node by choosing the feature to use as a splitting criterion in the input. Users can also consider the node a leaf node, according to their perception in the visualization and the ratio of cases in the node subset, and define it by clicking the button 'Make Leaf'. This can be undone later, in case that user decides to go deep into the tree by clicking the 'Reset Leaf' button. Users can also delete all children nodes related to the currently selected node in the tree viewer by clicking the 'Delete Children' button. In this tab, users also obtain details of the node, such as the purity of classes, the partitioning feature used, and the threshold value.

(a) Data preparation.

(b) Node construction.

Figure 5. Data interaction. From these menus, the user can (a) prepare the data for the analysis and build the train and test sets; and (b) create nodes and extract information from the partition in the node. In (a), analysts can define the separator used in the data file and the feature class for supervised learning and select features for the analysis and the percentage of the training–testing set. In (b), they can define the parameters for the node construction and deletion and define it as a leaf node.

4.2.2. Tree Viewer

This area shows the representation of the current state of the tree defined by the user (see Figure 6). We employed the Cytoscape component included in the Dash library to build this representation. We defined the following requirements for effectiveness. It has to be reactive to modifications in the tree and adapt the layout to better fit in the space as the tree expands. It also has to present information about node composition at first sight, the partitioning feature, and the corresponding threshold value. To represent the composition, each node is represented as a pie chart, showing the rate of each class. Using this Plotly module allows us to best fit the tree in a top-down grid that fits in the space assigned in the screen. When the node is considered a leaf node, the outline stroke for the pie chart is colored in green, and when the branch has not defined a leaf node, the outline stroke is colored in red. This also allows the user to obtain an overview of the tree or obtain details by zooming in on the area of interest or by clicking on the interest node.

4.2.3. LDA Representations

The method includes the use of LDA as a dimensionality reduction method as a form of support for the domain experts' decision. The LDA mapping is represented in a Star Coordinates representation system which includes axis vectors. The length and orientation of these vectors are connected with a greater influence in the SC visualization [26,28]. The representation is powered by a ScatterGl component present in the Plotly library. This allows users to support their decisions about the partitioning feature in every node and determine the most influential features in LDA mapping that have maximizing the class separation as a main goal (see Figure 7).

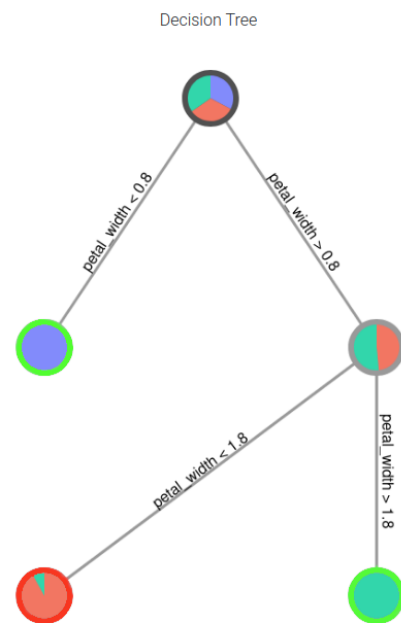


Figure 6. Users can extract knowledge about how the subset is split in every step of the classification. In this case, we show the steps to creating a tree in the Iris dataset [33]. Each node is represented as a pie chart with colors indicating the different categories, and the size representing their rate. If the node is defined as a leaf node, the outline stroke is colored in green. If the last node on a branch has not been defined as a leaf node, its outline stroke is colored in red.

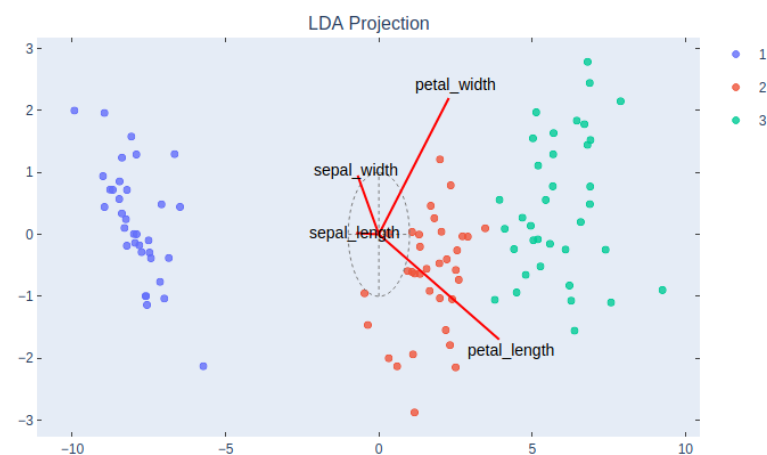


Figure 7. Support decision visualization based on LDA and Star Coordinates for the Iris dataset [33].

When the algorithm has to split a subset that contains only two classes, note that LDA can only reduce the dimensionality at a maximum of $n - 1$ dimensions, where n is the number of classes in the subset. Thus, when a node contains only two classes, users can choose between a Star Coordinates visualization with a principal component analysis-based jittering (see Figure 8a) or they can opt for a coordinated representation of the one-dimensional LDA mapping, the weight values of the features in the model, and a histogram showing the distribution of cases along the LDA representation (see Figure 8b).

The color map used maintains coherence between the different nodes' visualizations and the representation of the trees. The viewer is designed following the guidelines in the Visual Information-Seeking Mantra [34,35]. Among other things, this means that classes can be silenced from the visualization by clicking on the legend, and the application also enables users to retrieve details on demand from a HoverTool included in the viewer.

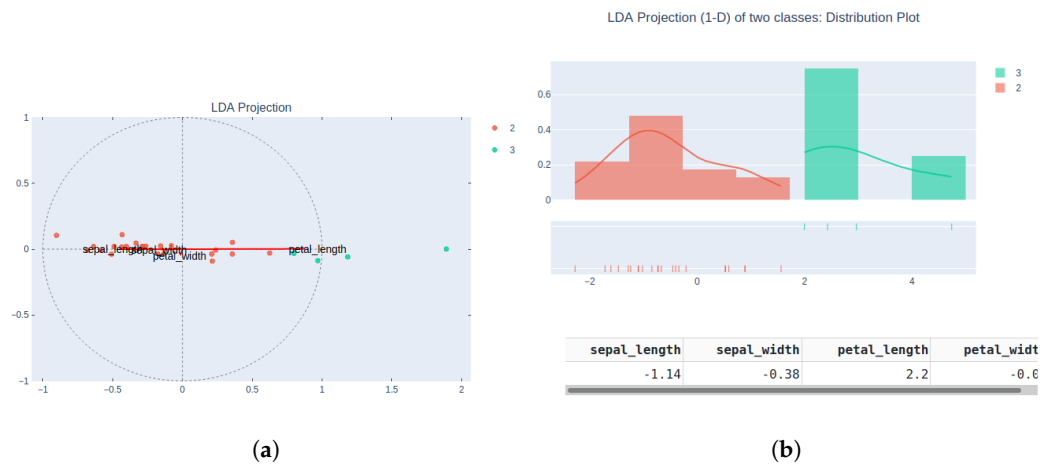


Figure 8. GDT allows the users to choose the representation for decision support in two classes of nodes. (a) SC representation of LDA after applying a jittering based on PCA in the same node. (b) Coordinated view of 1D LDA projection, distribution plot of samples, and values of features in linear transformation matrix. In this instance, the focus is on how it distinguishes between the versicolor (2, green) and virginica (3, red) classes within the iris dataset.

4.2.4. Classification Performance and Exporting Results

We have developed GDT to be able to compare between the built tree and an automatically generated one. Once the tree is complete, the users can interact with the classification controls. This area manages the parameters to build the trees implemented by the scikit-learn library. Users can define the split criterion by choosing between entropy or Gini and define the maximum depth from these automatically built trees.

Once the automatic tree has been executed, users can compare the classification performance between the sci-kit learn tree and the manually built tree. To show the performance of the trees, GDT includes the following indicators for each class present in the dataset: Precision, Recall, F1-Score, and the number of records used to test each class. The Precision value shows the number of correctly classified records of a class among the total number of observations classified as such. The Recall value shows the number of observations classified as one class among the total observations of that class. Lastly, the F1-Score is a general metric used to evaluate the classifier performance in each class, calculated as $F_1 = 2 \times (Precision \times Recall) / (Precision + Recall)$. These results are presented in a table, as shown in Figure 9.

Sklearn Decision Tree - Results					Manual Decision Tree - Results				
Class Label	Precision	Recall	F1-Score	Support	Class Label	Precision	Recall	F1-Score	Support
class_1	1	1	1	19	class_1	1	1	1	19
class_2	1	1	1	14	class_2	1	1	1	14
class_3	1	1	1	13	class_3	1	1	1	13

Figure 9. Performance obtained by the automatic and manually built trees in the Iris dataset [33]. Each class consists of 50 balanced records before training and testing sets with a 70:30 ratio.

Once the classifier is built and the user decides that the performance is good enough, both the scikit-learn and the manually built tree can be exported in PDF format by clicking the corresponding button on the interface. These representations employ the Graphviz library [36] to export the PDF. The scikit-learn tree generates a DOT visualization automatically. We color the leaf nodes according to the colormap used in all the visualizations. The manually built tree is exported from a custom function, and we maintain the color map associated with the classes for comparison with the automatically generated tree. The exported trees (see Figure 10) provide detailed information about every split feature, the cutoff value, entropy, the number of samples in relation to the initial training, and the purity values.

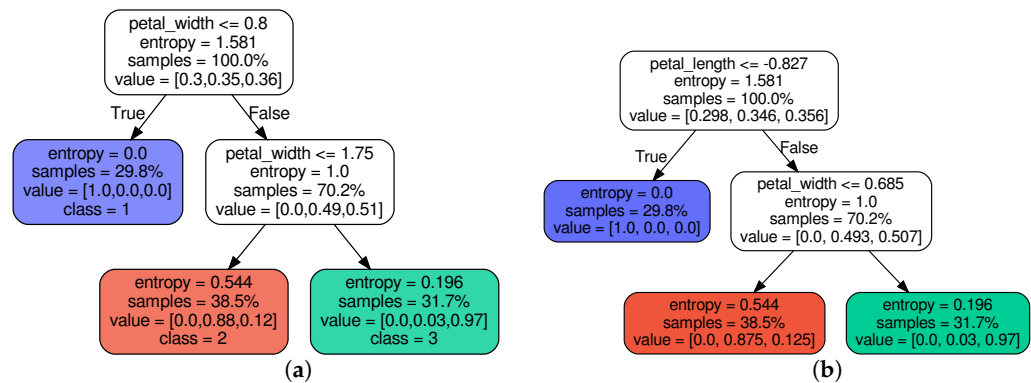


Figure 10. Example of exported trees both manually obtained and automatically generated by means of scikit-learn from the Iris dataset [33]. The similar format allows domain experts to make a comparison at first sight. (a) Guided interactively built tree and (b) scikit-learn tree.

5. Case Study

To demonstrate the effectiveness of using GDT on real data cases, we have provided it to domain experts in biomechanics specializing in human movement. Based on their prior knowledge and the visualizations provided by the tool, they built a classifier with which to compare its performance to machine learning methods, both in performance and complexity.

Note that sometimes the best decision tree is not the most efficient one, but the one whose partial decisions cause the least rejection in the user. This is easily observable in organic character animation issues, as the final positions of certain limbs (e.g., a hand, a walk cycle) may involve intermediate positions unfamiliar to a user. These anatomically correct but unusual positions provoke rejection in the users of graphic environments, enhancing the uncanny valley so feared in visual perception in this discipline.

The anatomical knowledge of the modeler and the animation expert is essential for a final result that is not only possible but also familiar. For this reason, we have tested our tool with a dataset of avatar positions. This dataset includes various poses and movements that are critical for creating realistic and acceptable animations. By incorporating expert knowledge into the decision-making process, we aim to reduce the uncanny valley effect and improve user acceptance of the final animated characters.

5.1. Data Description

The dataset utilized in this experiment originates from Dyna [37], which is derived from 4D scans of human bodies converted into meshes with generic features to ensure anonymity. Subsequent work generated produced skeletons from these meshes [38], and this is the dataset employed in this experiment. The dataset used to test the tool consists of 21,495 records obtained through skeleton extraction through SMPL models [39] of individuals doing exercises obtained from video frames and 14 labeled movements. This movements are as follows: 0—knee movement, 1—running on the spot, 2—chicken wings, 3—punching, 4—light stiff hopping, 5—shaking arms, 6—shaking hips, 7—bouncing on toes, 8—light loose hopping, 9—jumping jacks, 10—one-legged jump, 11—shaking shoulders, 12—moving hips, and 13—one-legged loose jump. The records in the dataset are obtained through the capture of the skeleton position in an angle axis system obtained from video frames.

For this analysis, we have created two scenarios from the original dataset. The original dataset consisted of 72 features (3 features for every joint, axis x , y and z), and after feature selection based on feature relevance, it was reduced to 15 features. This feature selection was based on their importance when representing LDA in two dimensions within a Star Coordinates system. Through this representation, a metric can be obtained for how relevant a variable is in achieving class separation, based on the magnitude and direction of its corresponding vector in the resulting linear transformation. Higher magnitudes are asso-

ciated with greater importance, and features with similar directions in the representation presumably imply a correlation [28]. This has been used to perform the initial feature selection of the dataset, as well as to guide the construction of the decision trees. Finally, the features included in the analysis are as follows: Left Foot 2, Spine 1, Left Hand 1, Left Foot 3, Left Collar 3, Left Collar 2, Right Hip 1, Left Hand 3, Right Ankle 3, Left Knee 1, Left Shoulder 3, Right Shoulder 3, Right Knee 1, Left Elbow 2 and Right Elbow 2. The set of variables used in both scenarios are displayed in Figure 11. Note that the original dataset showed a considerable imbalance between classes, so it was necessary to reduce the number of records in both scenarios through undersampling balancing classes. In both cases, the dataset scenarios were split into training and testing sets, with a 70:30 ratio. Table 1 provides a comparative summary of the number of records, features, and poses included in both scenarios.

Table 1. Comparison of datasets based on the number of records, features, and poses included.

Dataset	No. Records	No. Features	No. Poses
Original dataset	21,495	72	14
First scenario	6324	15	6
Second scenario	12,390	15	14

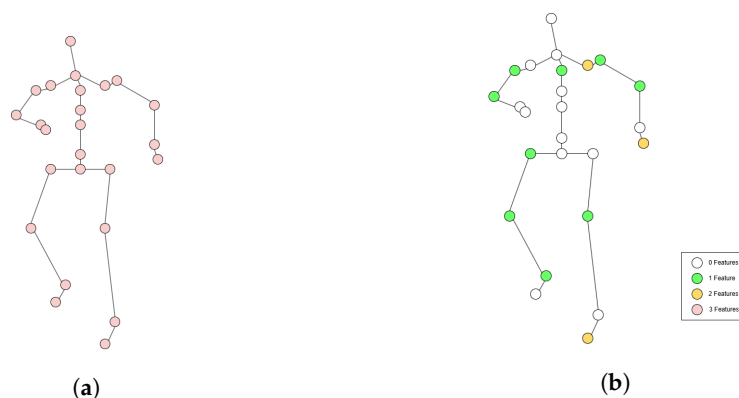


Figure 11. Representation of the variables in the original dataset (a) and in the two scenarios used for decision tree construction (b). The set of variables has been reduced based on feature importance metrics.

5.2. First Scenario

The goal in the first scenario is to build a classifier that can compete in performance with automatically built trees, using GDT and applying their previous knowledge. In this scenario, the dataset is reduced to six balanced classes, with 6324 records related with one of the following movements: 0—knee movement, 5—shaking arms, 7—bouncing on toes, 8—light loose hopping, 11—shaking shoulders, and 12—moving hips.

From this starting point, the domain experts represented the LDA corresponding to the training set and decided to create a node using the feature ‘59’ (Right Elbow 2) based on their perception and their prior knowledge. They persisted in this process of node construction until they considered the classifier to be built (see Figure 12a). Once the tree was built, they executed the scikit-learn algorithm (see Figure 12b) with the same maximum depth as the manual decision tree to compare their performance, (see Figure 13), observing that it is similar in both trees.

From the decision tree construction process, the domain experts could extract some insights from the raw data. The first insight found by the domain experts was that they were able to discriminate, with high performance, for 27% of the samples in only two steps, as shown in Figure 14. The second insight that they obtained from an exploratory data analysis was the predominance of the characteristics corresponding to the upper body when classifying movements. Specifically, they found upper body features to be the partitioning

feature on 56 occasions in the tree-building process. This corresponds to 69.13% of the total. Also, in this building process, the domain experts used left-sided features more frequently. They were used 53 times in this building process, compared to 28 for right-sided features and 13 for spine- and hip-related features.

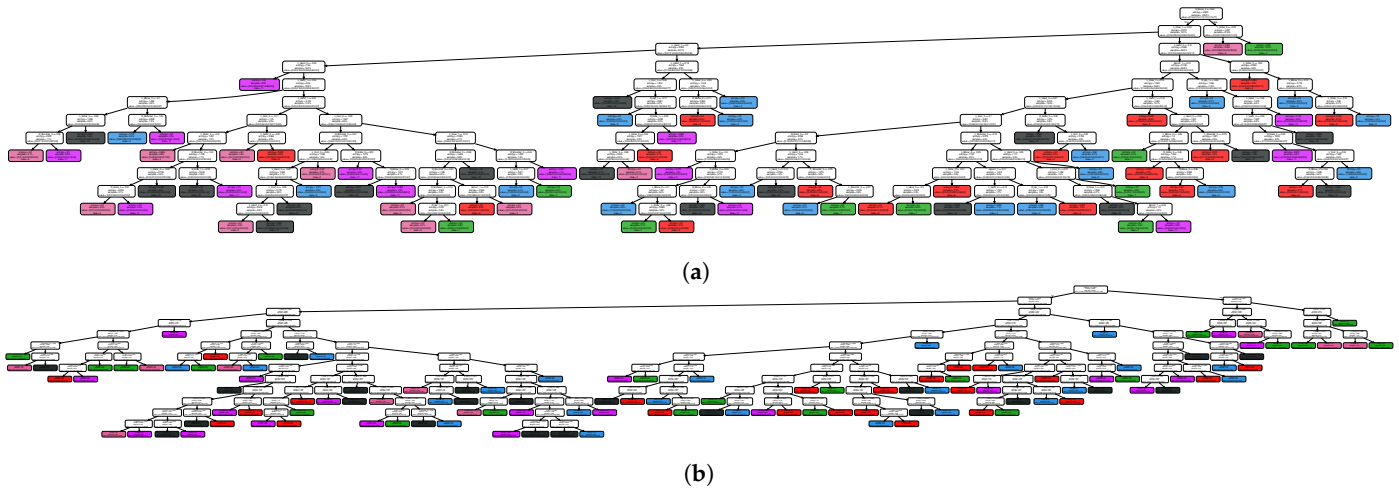


Figure 12. Resulting decision trees. (a) Decision tree built by domain experts by using GDT. (b) Decision tree built by using scikit-learn. The manually built tree (a) is simpler and easier to follow than the tree created by the algorithm (b).

Sklearn Decision Tree - Results					Manual Decision Tree - Results				
Class Label	Precision	Recall	F1-Score	Support	Class Label	Precision	Recall	F1-Score	Support
class_0	0.9905	0.9631	0.9766	325	class_0	0.941	0.8831	0.9111	325
class_5	0.9937	0.9874	0.9905	317	class_5	0.9625	0.8896	0.9246	317
class_7	0.978	0.9936	0.9857	313	class_7	0.9119	0.9585	0.9346	313
class_8	0.9788	0.9618	0.9702	288	class_8	0.8349	0.9479	0.8878	288
class_11	0.9534	0.9808	0.9669	313	class_11	0.9302	0.8946	0.9121	313
class_12	0.9563	0.9623	0.9592	318	class_12	0.9122	0.9151	0.9137	318

Figure 13. Performance comparison between the tree built by scikit-learn and manually, with six classes in the dataset. Each class starts balanced, with 1024 samples, and training and testing sets are created with a 70:30 ratio. The results of the manual tree are similar to those obtained with the automatic algorithm, but with a simpler tree built by a process that allows domain experts to participate in the process.

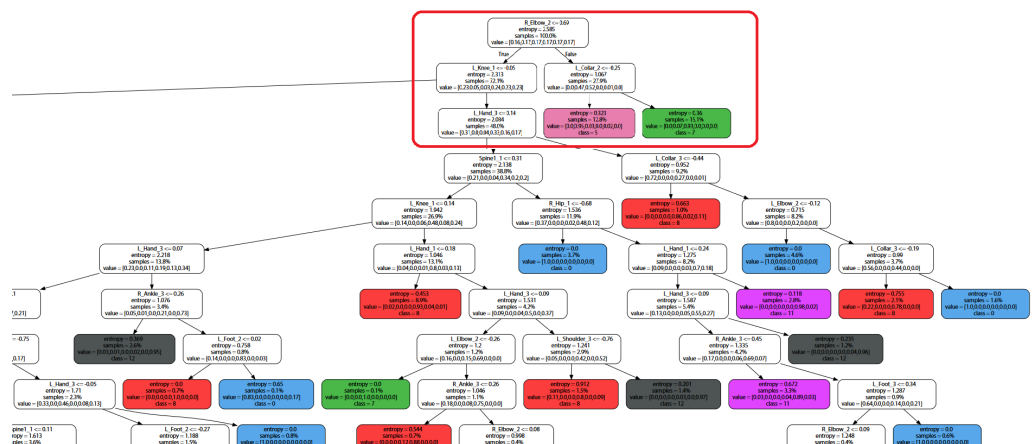


Figure 14. Detail from the decision tree built by the domain experts. Highlighted in a red box are the first two steps where leaf nodes (pink and green) are defined for more than 27% of the total samples.

5.3. Second Scenario

The goal in this second scenario is the same as seen in the first scenario, but the problem to tackle is more complex due to the higher number of samples (12,390) and labels (14 balanced) in the classification problem. Adding samples and classes makes visualizations more difficult for domain experts to follow and makes it more difficult for experts to apply their previous knowledge.

The domain experts, by using the same process as in the previous scenario, built a decision tree to compare it with the decision tree automatically generated by scikit-learn. The resulting trees are available in Figure 15 in low resolution due to format restrictions. For more details, check https://github.com/mohedano-munoz/decision_tree_builder/tree/main/decision_trees_movements (accessed on 8 November 2024) to access the complete trees in high resolution. The higher complexity of the automatic tree can be observed in the figure due to the different sizes of the trees and different numbers of nodes. In this case, the performance of the automatic tree is superior, as shown in Figure 16, although only slightly.

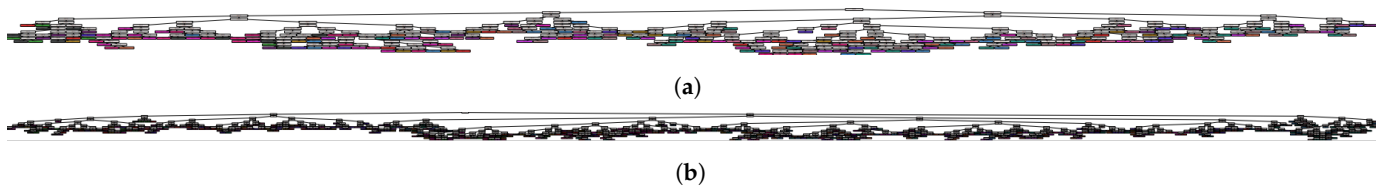


Figure 15. Resulting decision trees for the whole dataset with 14 classes. (a) Guided Decision tree. (b) Scikit-learn decision tree.

Sklearn Decision Tree - Results					Manual Decision Tree - Results				
Class Label	Precision	Recall	F1-Score	Support	Class Label	Precision	Recall	F1-Score	Support
class_0	0.927	0.9407	0.9338	270	class_0	0.7619	0.8296	0.7943	270
class_1	0.9492	0.9346	0.9419	260	class_1	0.9417	0.7462	0.8326	260
class_2	0.9766	0.9653	0.9709	259	class_2	0.9306	0.8803	0.9048	259
class_3	0.9458	0.9632	0.9545	272	class_3	0.793	0.7463	0.7689	272
class_4	0.9688	0.9118	0.9394	272	class_4	0.74	0.8162	0.7762	272
class_5	0.97	0.9557	0.9628	271	class_5	0.9423	0.9041	0.9228	271
class_6	0.9527	0.8942	0.9225	293	class_6	0.8242	0.7679	0.7951	293
class_7	0.932	0.951	0.9414	245	class_7	0.8074	0.8041	0.8057	245
class_8	0.8647	0.935	0.8984	246	class_8	0.7463	0.813	0.7782	246
class_9	0.945	0.945	0.945	291	class_9	0.7839	0.7354	0.7589	291
class_10	0.9044	0.847	0.8748	268	class_10	0.7269	0.7052	0.7159	268
class_11	0.9485	0.9214	0.9348	280	class_11	0.8364	0.8214	0.8288	280
class_12	0.9266	0.9796	0.9524	245	class_12	0.7766	0.8939	0.8311	245
class_13	0.8806	0.9593	0.9183	246	class_13	0.695	0.7967	0.7424	246

Figure 16. Performance comparison between the tree built by scikit-learn and manually, with 14 classes in the dataset. Each class starts balanced, with 885 samples, and training and testing sets are created with a 70:30 ratio. The results of the manual tree are slightly worse in this case, but the result is a simpler decision tree wherein the domain experts could apply their knowledge.

The difference in complexity between these trees is evident; the interactively constructed tree has 217 decision nodes, while the tree generated by scikit-learn has 504 nodes. In the initial stages of constructing the guided tree, the emphasis is placed on variables associated with the lower body’s position, such as the knees, and from these nodes, the tree is further developed. A key distinction is its emphasis on the hip, with the variable ‘R_Hip_1’ being decisive in 20 of nodes in the guided tree. This variable appears both in the early stages of the tree and in the nodes close to the leaf nodes (see Figure 17). Shoulder rotation is also significant, with ‘L_Shoulder_3’ and ‘R_Shoulder_3’ contributing to 35 decision nodes. Furthermore, hand movements play a crucial role in defining motion, as indicated by the variables used in the tree. A detailed count of the most frequently appearing variables in the decision nodes for classifying movements for both trees, for comparison purposes, is provided in Table 2.

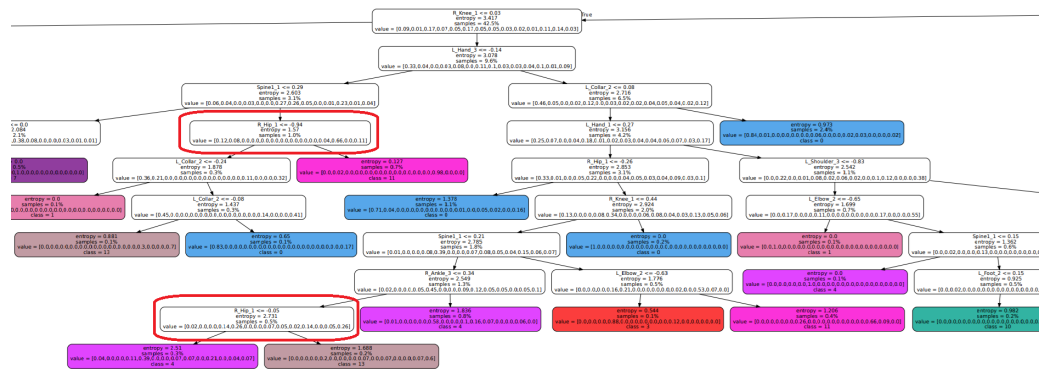


Figure 17. Zoomed-in view of Figure 15a, focusing on the early stages, where it can be seen, for instance in the nodes highlighted with the red boxes, that the variable ‘R_Hip_1’ is a decisive node.

Table 2. Top #15 variable occurrences in the decision nodes.

Scikit-Learn Tree		Guided Decision Tree	
Feature	Nodes	Feature	Nodes
R_Shoulder_3	47	R_Hip_1	20
R_Elbow_2	42	L_Hand_1	20
Spine1_1	40	L_Shoulder_3	19
L_Hand_3	40	L_Hand_3	19
L_Elbow_2	38	L_Collar_2	16
L_Foot_3	34	R_Shoulder_3	16
L_Collar_2	32	R_Knee_1	14
R_Hip_1	32	Spine1_1	14
L_Knee_1	32	L_Knee_1	13
R_Knee_1	32	L_Elbow_2	13
R_Ankle_3	29	L_Collar_3	13
L_Hand_1	29	R_Elbow_2	11
L_Collar_3	28	L_Foot_3	10
L_Shoulder_3	26	R_Ankle_3	10
L_Foot_2	23	L_Foot_2	9

6. Conclusions

The main objectives of this study were to highlight the substantial benefits of combining domain expertise with decision tree construction and to offer domain experts a practical tool for creating visually guided decision trees. We have developed an interactive webtool, named GDT, for data handling; visualization (using SC and LDA); guided decision tree creation; and finally, comparison of the main parameters of the classification accuracy with the DecisionTreeClassifier within the scikit-learn library of Python. From GDT, the users can split the dataset into training and testing sets, create nodes, and split the training set according to [9]. The specific tasks that the tool performs to meet the needs of the method are as follows: (i) splitting the dataset in train and test set, (ii) allowing the user to select features, (iii) creating and managing the subsets belonging to different nodes, and (iv) visualizing, managing, and editing the decision tree in a proper way. We have published GDT and its source code (in Python, using Dash, Plotly, scikit-learn, pandas, and other packages) online (https://github.com/mohedano-munoz/decision_tree_builder, accessed on 8 November 2024) so that analysts can use it and visualization experts can extend and adopt it to their needs.

To demonstrate the utility of the tool, we presented a case study on motion recognition. In this case study, the manually constructed decision trees achieved slightly lower accuracy than those generated by sci-kit learn, but they were simpler and more interpretable. The manual construction process enabled domain experts to build more straightforward decision trees, gain valuable insights from the data, and apply their prior knowledge in

the classifier construction. Our findings underscore the importance of a collaborative approach in machine learning, where human insight and computational efficiency work hand-in-hand to achieve better results. In this case, thanks to the GDT tool, domain experts confirmed that upper body features, particularly hip and shoulder rotation, along with hand movements, play a crucial role in defining motion and are more relevant within motion recognition.

In future work, we plan to use GDT in different areas that require a selection of the most important variables in explainable models or that benefit from understanding the underlying processes. Fields such as health, social welfare, and meta-human utilization require decision trees that are more understandable and closer to the end user to validate their versatility and effectiveness. Additionally, there is potential for further development of both the tool and the method. For instance, implementing the C5.0 algorithm could provide an alternative for automatic tree construction. Another valuable enhancement could be the inclusion of various metrics for node splitting based on the dataset, allowing analysts the flexibility to choose the most appropriate metric for their specific needs.

Author Contributions: Conceptualization, M.A.M.-M. and A.S.; Data curation, M.A.M.-M.; Funding acquisition, A.S.; Investigation, M.A.M.-M. and A.S.; Methodology, L.R.; Software, M.A.M.-M.; Supervision, L.R. and A.S.; Validation, L.R. and A.S.; Visualization, M.A.M.-M. and A.S.; Writing—original draft, M.A.M.-M.; Writing—review and editing, L.R. and A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Agencia Estatal de Investigación (Spanish Research Agency), grant number PID2021-122392OB-I00.

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable. The data obtained from Dyna are fully anonymous (there are no videos or images, only 3D meshes with generic faces) and are licensed for academic use (http://dyna.is.tue.mpg.de/data_license accessed on 15 September 2024).

Data Availability Statement: The dataset analyzed during the current study originally comes from Dyna (<http://dyna.is.tue.mpg.de> accessed on 1 July 2022) and is a dataset of 4D scans of human bodies modified afterwards in [38].

Acknowledgments: The authors would like to thank F. Blasco for constructive criticism and help in the development of the GDT webtool.

Conflicts of Interest: The authors declare no conflicts of interest. The founders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

GDT	Guided Decision Tree
LDA	Linear Discriminant Analysis
SC	Star Coordinates
CART	Classification And Regression Trees

References

1. Al-Jarrah, O.Y.; Yoo, P.D.; Muhaidat, S.; Karagiannidis, G.K.; Taha, K. Efficient Machine Learning for Big Data: A Review. *Big Data Res.* **2015**, *2*, 87–93. [CrossRef]
2. Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* **2019**, *1*, 206–215. [CrossRef]
3. Alpaydin, E. *Introduction to Machine Learning*; MIT Press: Cambridge, MA, USA, 2020.
4. Breiman, L.; Friedman, J.; Stone, C.J.; Olshen, R.A. *Classification and Regression Trees*; CRC Press: Boca Raton, FL, USA, 1984.
5. Liu, Y.; Salvendy, G. Visualization support to better comprehend and improve decision tree classification modelling process: A survey and appraisal. *Theor. Issues Ergon. Sci.* **2007**, *8*, 63–92. [CrossRef]

6. Fayyad, U.M.; Piatetsky-Shapiro, G.; Smyth, P.; Uthurusamy, R. (Eds.) *Advances in Knowledge Discovery and Data Mining*; American Association for Artificial Intelligence: Washington, DC, USA, 1996.
7. Breslow, L.A.; Aha, D.W. Simplifying decision trees: A survey. *Knowl. Eng. Rev.* **1997**, *12*, 1–40. [[CrossRef](#)]
8. Price, W.N. Big data and black-box medical algorithms. *Sci. Transl. Med.* **2018**, *10*, eaao5333. [[CrossRef](#)]
9. Soguero-Ruiz, C.; Mora-jimenez, I.; Mohedano-Munoz, M.A.; Rubio-Sanchez, M.; de Miguel-Bohoyo, P.; Sanchez, A. Visually guided classification trees for analyzing chronic patients. *BMC Bioinform.* **2020**, *21* (Suppl. S2), 92. [[CrossRef](#)]
10. Güler, R.A.; Neverova, N.; Kokkinos, I. DensePose: Dense Human Pose Estimation in the Wild. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7297–7306. [[CrossRef](#)]
11. Mehta, D.; Sridhar, S.; Sotnychenko, O.; Rhodin, H.; Shafiei, M.; Seidel, H.P.; Xu, W.; Casas, D.; Theobalt, C. VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera. *ACM Trans. Graph.* **2017**, *36*, 1–14. [[CrossRef](#)]
12. Su, D.; Hu, Z.; Wu, J.; Shang, P.; Luo, Z. Review of adaptive control for stroke lower limb exoskeleton rehabilitation robot based on motion intention recognition. *Front. Neurobot.* **2023**, *17*, 1186175. [[CrossRef](#)] [[PubMed](#)]
13. Wu, F.; Wang, Q.; Bian, J.; Ding, N.; Lu, F.; Cheng, J.; Dou, D.; Xiong, H. A Survey on Video Action Recognition in Sports: Datasets, Methods and Applications. *IEEE Trans. Multimed.* **2023**, *25*, 7943–7966. [[CrossRef](#)]
14. Lv, Z.; Poesi, F.; Dong, Q.; Lloret, J.; Song, H. Deep Learning for Intelligent Human–Computer Interaction. *Appl. Sci.* **2022**, *12*, 11457. [[CrossRef](#)]
15. Fayyad, U.M.; Wierse, A.; Grinstein, G.G. Introduction. In *Information Visualization in Data Mining and Knowledge Discovery*; Morgan Kaufmann: Burlington, MA, USA, 2002; pp. 1–20.
16. Marsland, S. *Machine Learning: An Algorithmic Perspective*; CRC Press: Boca Raton, FL, USA, 2014.
17. Yuan, J.; Chen, C.; Yang, W.; Liu, M.; Xia, J.; Liu, S. A Survey of Visual Analytics Techniques for Machine Learning. *arXiv* **2020**, arXiv:2008.09632. [[CrossRef](#)]
18. Nguyen, T.D.; Ho, T.B.; Shimodaira, H. Interactive Visualization in Mining Large Decision Trees. In Proceedings of the Knowledge Discovery and Data Mining. Current Issues and New Applications, Kyoto, Japan, 18–20 April 2000; Terano, T., Liu, H., Chen, A.L.P., Eds.; Springer: Berlin/Heidelberg, Germany, 2000; pp. 345–348. [[CrossRef](#)]
19. Neville, P.; Barlow, T. Case Study: Visualization for Decision Tree Analysis in Data Mining. In Proceedings of the Information Visualization, IEEE Symposium on, IEEE Computer Society, Los Alamitos, CA, USA, 22–23 October 2001; p. 149. [[CrossRef](#)]
20. Stiglic, G.; Kocbek, S.; Pernek, I.; Kokol, P. Comprehensive Decision Tree Models in Bioinformatics. *PLoS ONE* **2012**, *7*, e33812. [[CrossRef](#)] [[PubMed](#)]
21. Streeb, D.; Metz, Y.; Schlegel, U.; Schneider, B.; El-Assady, M.; Neth, H.; Chen, M.; Keim, D.A. Task-Based Visual Interactive Modeling: Decision Trees and Rule-Based Classifiers. *IEEE Trans. Vis. Comput. Graph.* **2021**, *28*, 3307–3323. [[CrossRef](#)] [[PubMed](#)]
22. Van Den Elzen, S.; Van Wijk, J.J. Baobabview: Interactive construction and analysis of decision trees. In Proceedings of the 2011 IEEE Conference on Visual Analytics Science and Technology (VAST), Providence, RI, USA, 23–28 October 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 151–160. [[CrossRef](#)]
23. Teoh, S.T.; Ma, K.L. PaintingClass: Interactive construction, visualization and exploration of decision trees. In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 24–27 August 2003; ACM: New York, NY, USA, 2003; pp. 667–672. [[CrossRef](#)]
24. Fisher, R.A. The Use of Multiple Measurements in Taxonomic Problems. *Ann. Eugen.* **1936**, *7*, 179–188. [[CrossRef](#)]
25. Kandogan, E. Star Coordinates: A Multi-dimensional Visualization Technique with Uniform Treatment of Dimensions. In Proceedings of the IEEE Information Visualization Symposium, Late Breaking Hot Topics, Salt Lake City, UT, USA, 9–10 October 2000; IEEE Computer Society: Washington, DC, USA, 2000; pp. 9–12.
26. Rubio-Sánchez, M.; Raya, L.; Díaz, F.; Sanchez, A. A comparative study between RadViz and Star Coordinates. *IEEE Trans. Vis. Comput. Graph.* **2016**, *22*, 619–628. [[CrossRef](#)]
27. Morales-Vega, J.; Raya, L.; Sánchez-Rubio, M.; Sanchez, A. A virtual reality data visualization tool for dimensionality reduction methods. *Virtual Real.* **2024**, *28*, 41. [[CrossRef](#)]
28. Sanchez, A.; Raya, L.; Mohedano-Munoz, M.A.; Rubio-Sánchez, M. Feature selection based on star coordinates plots associated with eigenvalue problems. *Vis. Comput.* **2020**, *14*, 203–216. [[CrossRef](#)]
29. McKinney, W. Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference, Austin, TX, USA, 28 June–3 July 2010; van der Walt, S., Millman, J., Eds.; pp. 56–61. [[CrossRef](#)]
30. Buitinck, L.; Louppe, G.; Blondel, M.; Pedregosa, F.; Mueller, A.; Grisel, O.; Niculae, V.; Prettenhofer, P.; Gramfort, A.; Grobler, J.; et al. API design for machine learning software: Experiences from the scikit-learn project. *arXiv* **2013**, arXiv:1309.02382. [[CrossRef](#)]
31. Quinlan, J. *C4. 5: Programs for Machine Learning*; Morgan Kaufmann: Burlington, MA, USA, 1993.
32. Decision Trees—Scikit-Learn Documentation. Available online: <https://scikit-learn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart> (accessed on 29 September 2023).
33. Anderson, E. The Species Problem in Iris. *Ann. Mo. Bot. Gard.* **1936**, *23*, 457–509. [[CrossRef](#)]
34. Shneiderman, B. The eyes have it: A task by data type taxonomy for information visualizations. In Proceedings of the Proceedings 1996 IEEE Symposium on Visual Languages, Boulder, CO, USA, 3–6 September 1996; IEEE: Piscataway, NJ, USA, 1996; pp. 336–343. [[CrossRef](#)]

35. Heer, J.; Shneiderman, B. Interactive Dynamics for Visual Analysis. *Commun. ACM* **2012**, *55*, 45–54. [[CrossRef](#)]
36. Ellson, J.; Gansner, E.R.; Koutsofios, E.; North, S.C.; Woodhull, G. Graphviz and dynagraph—Static and dynamic graph drawing tools. In *Proceedings of the Graph Drawing Software, Perugia, Italy, 21–24 September 2003*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 127–148. [[CrossRef](#)]
37. Pons-Moll, G.; Romero, J.; Mahmood, N.; Black, M.J. Dyna: A Model of Dynamic Human Shape in Motion. *ACM Trans. Graph.* **2015**, *34*, 1–14. [[CrossRef](#)]
38. Santesteban, I.; Garces, E.; Otaduy, M.A.; Casas, D. SoftSMPL: Data-driven Modeling of Nonlinear Soft-tissue Dynamics for Parametric Humans. *Comput. Graph. Forum* **2020**, *39*, 65–75. [[CrossRef](#)]
39. Loper, M.; Mahmood, N.; Romero, J.; Pons-Moll, G.; Black, M.J. SMPL: A skinned multi-person linear model. *ACM Trans. Graph.* **2015**, *34*, 1–16. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.