Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Decision Trees Behavior Visualization and Analysis

Jan Krátký

Supervisor: Ing. Petr Vondrášek
Study program: Open Informatics
Specialisation: Artificial Intelligence and Computer Science
May 2023

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Krátký Jan** Personal ID number: **493779**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Decision Trees Behavior Visualization and Analysis

Bachelor's thesis title in Czech:

Vizualizace a analýza chování rozhodovacích stromů

Guidelines:

1) Study how Seznam.cz currently uses decision trees for evaluation of the search results relevance.
2) Explore and compare decision tree visualization libraries for Python.
3) Propose methods to extract useful information about the model from the available data.
4) Implement the proposed methods and asses its contributions.
5) Evaluate the benefits of the implemented solution and compare the results with the current solution used in Seznam.cz.

Bibliography / sources:

[1] A Visual Introduction to Machine Learning. www.r2d3.us/visual-intro-to-machine-learning-part-1.
[2] CatBoost - State-of-the-art Open-source Gradient Boosting Library With Categorical Features Support. https://catboost.ai/.
[3] Płoński, Piotr. Visualize a Decision Tree in 4 Ways With Scikit-Learn and Python. MLJAR, 22 June 2020, mljar.com/blog/visualize-decision-tree.
[4] How to visualize decision tree. How to visualize decision trees. (n.d.), https://explained.ai/decision-tree-viz/
[5] Tamara Munzner. Visualization Analysis and Design. A K Peters Visualization Series, CRC Press, 2014

Name and workplace of bachelor's thesis supervisor:

Ing. Petr Vondrášek Seznam.cz, a.s. Praha

Name and workplace of second bachelor's thesis supervisor or consultant:

Ing. Petr Pošík, Ph.D. Department of Cybernetics FEE

Date of bachelor's thesis assignment: **01.02.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

Ing. Petr Vondrášek Supervisor's signature	prof. Ing. Tomáš Svoboda, Ph.D. Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt Student's signature

Acknowledgements

I would like to express my gratitude to my supervisor, Mr. Petr Vondrášek, Ing., for his valuable advice and insights, and to Seznam.cz for providing the model and data used in this work.

I would also like to thank Mr. Ondřej Žára, RNDr., for his promptness and willingness to facilitate my collaboration with Seznam.cz as part of the project.

Last but not least, I want to thank my family for their continuous support and belief in me.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, May 26, 2023

Abstract

Companies utilize machine learning models for practical business decisions, and as the use of these models increases, so does the need for e ective visualization techniques. These techniques can help make machine learning models more accessible and intuitive, and they can also be utilized for future model improvements.

This thesis emphasizes the importance of visualizing decision trees, which serve as fundamental building blocks for more complex models. The thesis explores various approaches to decision tree visualization and proposes an interactive method suitable for a model used for evaluating the relevance of search results in Seznam.cz. This method captures the model's structure along with the data ows passing through it.

Keywords: Machine learning visualization, Decision trees, Gradient boosting, Search engine

Supervisor: Ing. Petr Vondrá²ek
Seznam.cz, a. s.,
Radlická 3294/10,
15000 Praha 5

Abstrakt

Spole£nosti vyuºívají modely strojového u£ení pro praktická obchodní rozhodnutí a s rostoucím vyuºitím t¥chto model· roste i pot°eba ú£inných technik jejich vizualizace. Tyto techniky mohou pomoci zp°ístupnit a intuitivním zp·sobem prezentovat modely strojového u£ení, mohou být také pouºity k jejich budoucímu vylep²ení.

Tato práce zd·raz¬uje význam vizualizace rozhodovacích strom·, které slouºí jako základní stavební kameny pro sloºit¥j²í modely. Práce zkoumá r·zné p°ístupy k vizualizaci rozhodovacích strom· a navrhuje interaktivní metodu vhodnou pro model, který se pouºívá k vyhodnocování relevance výsledk· vyhledávání ve spole£nosti Seznam.cz. Tato metoda zachycuje strukturu modelu spolu s datovými toky, které jím procházejí.

Klí£ová slova: Vizualizace strojového u£ení, Rozhodovací stromy, Gradient boosting, Vyhledáva£

P°eklad názvu: Vizualizace a analýza chování rozhodovacích strom·

Contents

Figures

Tables

Chapter 1

Introduction

Visualization plays crucial role in computer science, as it allows for complex data to be displayed in a way that is easily understandable to humans. Data visualization is used to represent information in the form of graphs, maps, diagrams or other visual formats, making it possible to gain insights and discover patterns that might be missed through numerical analysis alone. It has gained a special place in the eld of machine learning. Companies utilize machine learning models to make practical business decisions, and as the use for them continues to grow, so does the need for e ective visualization techniques. These techniques can help to make machine learning models more accessible and intuitive, and may even be used to improve models in the future.

This work highlights the signi cance of visualizing machine learning models, with focus on decision trees which are fundamental building blocks of gradient boosting machines particularly inspected in this work. It explores di erent approaches of the decision tree visualization and proposes a an interactive method that captures model behavior together with the structure of data ows that pass through the model.

First, we will examine the basic terminology of standard decision trees and one speci c type of tree construction in Chapter 2. This knowledge will then be used in the assembly of more complex models that utilize decision trees as fundamental building blocks (Chapter 3). The theoretical part is then contextualized within the domain of search engines in Chapter 4. After exploring currently used techniques for visualizing decision trees in Chapter 5, the proposed solution is described (Chapter 6).

Most of the content of the work revolves around a speci c machine learning model provided by the Seznam.cz company, which in practice uses it to evaluate the relevance of search results. This model uses the output of several standard deicison trees as part of the gradient boosting technique (detailed in 3.2). The survey and practical examples in this work relate to individual decision trees in this model and the validation data both provided by Seznam.cz.

Chapter 2

Overview of the Decision Tree Algorithm

Deicison trees are popular type of supervised machine learning algorithm which handles non-linear data sets e ectively and is used for both classi cation and regression tasks[1]. They are easy to understand and interpret, making them popular choice among practitioners and researchers alike. Decision trees have been widely used in a variety of elds such as healthcare, nance, marketing, and image recognition, among others.

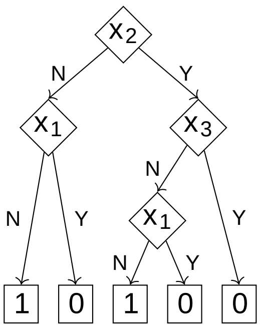

Figure 2.1: Basic structure of the decision tree

Altough modern advanced machine learning models, such as deep neural networks, typically perform better in many metrics due to the tendency of decision trees to over t[2], they are not nearly as interpretable. Therefore, decision trees are also used as so-called surrogate models [1] for these black-box classi ers. The idea is to get simpler model in the form of decision tree to replicate the black-box model predictions as best as possible, and at the same time, provide interpretability.

In the following sections we will explore the basic principles of standard decision tree algorithm in the context of later visualization techniques. Without going into unnecessary details, some machine learning methods for learning decision trees will be brie y presented.

[1] Classi cation tasks involve categorizing data into prede ned classes, while regression tasks involve predicting numerical values based on observed features.

[2] The model predicts unseen data with over-specialization on the training data.

2.1 Terminology

At high level, decision tree algorithm works by recursively partitioning the input space into smaller and smaller regions that should cluster parametrically similar data. The goal is to create tree-like structure where the leaf nodes of the tree represent the decision or the output of the classi er, while the branches represent the decision-making process. Each internal node of the tree corresponds to a split point. In it, based on the output of the given split function, the model decides which node to go to within the tree path.

The decision tree terminology further related to visualizations used in this work includes:

- Split node: An internal node in the tree that splits the data into two or more subsets based on certain split function.
    - Split function: A rule that determines how the data should be divided at each split node based on the value of selected split feature.
        - Split feature: A feature or an attribute of the input data that is used in the split function. It may be classi ed as categorical or continuous. Categorical feature is a variable that can take one of a limited, and usually xed number of possible values. Continuous features, on the other hand, can take on one of an in nite number of numeric or oat values.
        - Threshold: Boundary value for speci c continuous feature that is used to split the data in the split function.
        - Branch: An edge connecting two nodes that represents the output of the split function.
    - Root node: The top-most node of the tree that in the context of a data stream represents the entire input dataset.
- Leaf node: A node that does not split the data anymore and represents the nal outcome or decision. In the classi er type of decision tree the outcome is a class label, in regressor it is a numerical value.
- Depth: The length of the longest path from the root node to any leaf node, indicating the complexity or size of the tree.
- Level: A horizontal layer of nodes in the tree structure. The root node is considered to be at level 0, and subsequent levels are numbered sequentially.

2.1.1 Oblivious Decision Trees

In later sections, we will be interested in decision binary trees that only have continuous split features in their nodes. Moreover, the model that is the subject of visualization in this work uses so-called oblivious trees which are grown symmetrically. In addition to being perfectly binary, they also use the same split functions in nodes at the same level of the tree. So each oblivious decision tree outputs one of 2^d decisions, where d is the depth of the tree. And that is done by using d split feature-threshold combinations, which are essentially parameters of that tree [4].

Figure 2.2: Example of an oblivious tree structure [2]

The choice of oblivious trees has several advantages compared to the general ones. Among them is a simple tting scheme, e ciency of implementation on CPU and the ability to make very fast model appliers. It also works as a regularization, so it can provide quality bene ts for many tasks [2].

2.2 Learning

There are several machine learning strategies to build an appropriate decision tree structure before applying the model to unseen test data. As is common practice in machine learning, they use data pre-labeled with target values in an optimization process called training. During the training, the model is structured in a such way that these training data passing through it are labeled as precisely as possible by the model itself. In the case of decision trees, leaf nodes should contain predominantly data with the same or similar target values.

The essence of the strategy itself is reduced to the choice of the optimal split function based on the incoming subset of training data. The optimal one is the one that splits the data most e ciently. Due to the structure of the decision tree, this method can be applied recursively to each node at each level until the required data purity, the speci ed depth or some other criterion is reached. After this growth phase, which is often called a greedy algorithm [9], practices like pruning [3] can be applied to improve the tree performance in the sense of obtaining better generalization on unseen data.

One of the most commonly used metrics to asses the optimality or gain from each possible splitting option in classi cation tasks is the Gini impurity.

Based on the Gini-value in 2.1 we are first able to express the probability that two samples randomly chosen from dataset D have different class labels.

$$Gini(D) = 1 - \sum_{i=1}^{n} p_i^2 [10] \tag{2.1}$$

n is the number of class labels in dataset D and p_i is denoted as the probability that class i occurs in dataset D.

Using the Gini-value the definition of the Gini-impurity index in 2.2 to asses the purity of the subsets made by the concerned split function can be set.

$$Gini\_index(D;k) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} Gini(D^v) [10] \tag{2.2}$$

D^v represents one subset of dataset D classified based on split function k and V is the total number of subsets. The task of choosing the optimal split function is then expressed by minimization problem in 2.3 where the optimal k produces the least Gini-impurity index.

$$k^* = \arg\min_{k \in K} Gini\_index(D;k) [10] \tag{2.3}$$

Since in regression a continuous variable is predicted, the same process cannot be applied in these tasks. Instead, the mean square error metric (in 2.4) is often used in the regression. This measurement, according to which the split function with the greatest gain can be chosen, tells us how much the predictions deviate from the original target in individual subsets.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 [11] \tag{2.4}$$

n is the number of data samples in the given dataset, y is the actual target value of sample in training dataset and $\hat{y}$ is the prediction.

Chapter 3

Improving Decision Trees with Ensembles

The standard decision tree algorithm has been extended in various ways to improve its performance and applicability to di erent types of problems. One common extension is the use of ensemble methods, which combine multiple decision trees to improve their accuracy and reduce an excessive specialization to the training data. They di er in the approach of building the ensemble and addressing the weaknesses of standard decision trees. The most common such approaches are bagging and boosting [12] and their concrete model instances in the form of Random Decision Forest and Gradient Boosting on Decision Trees (GBDT), which we will look at in the following sections.

Figure 3.1: Methods of creating decision tree ensembles [15]

3.1 Random Decision Forest

One way to create and combine a set of decision trees to improve performance is the bootstrap aggregation, or bagging for short. The idea is to create several subsets of data from original training set chosen randomly with replacement. Each such collection is then used to grow single decision tree, together forming a tree set or forest. Predictions from this ensemble are then aggregated, usually done by averaging.

Random Forest is an extension over bagging [14]. It takes one extra step where in addition to taking the random subset of data, it also uses the random selection of features to grow trees, which helps to reduce the correlation between the trees in the forest and ultimately reduces the generalization error.

Decision trees, where the selection of split functions is a random variable, also have the advantage of greater independence from data in the training process. Tree structures can first be built independently and then, using the training data, the probability estimates of each class in individual leaves $P(class|leaf)$ are calculated. On the other hand, in a random decision tree, the random split functions do not give us the same credible insight into the data as with standard decision trees with thoughtfully designed splits.

3.2 Gradient Boosting on Decision Trees

Another way of combining individual decision trees is boosting. Boosting leverages the idea of learning from your mistakes. It generally works by iteratively training decision trees via an error-based data reweighting scheme. We take a single training dataset, and use it to grow a single decision tree. Then the training set is re-weighted so that records with incorrectly predicted targets receive more weight and this weighted data trains another tree. The procedure continues until the limit of the number of trees or the required accuracy is reached. Predictions of the final ensemble model is the weighted sum of the predictions made by individual trees.

Boosting itself, which is best exemplified by the Adaboost technique [13], identifies the shortcomings by using high weight data points. In a similar manner, gradient boosting achieves the same objective by employing the gradient descent algorithm [14]. This algorithm optimizes the differentiable loss function which is a measure indicating how good are model's coefficients at fitting the underlying data.

In the context of decision trees, we can consider the mentioned regression mean square error metric from 2.4 for single data point mulitplied by coefficient of one half[1] as a loss function. If we differentiate this expression (in 3.1) with respect to the prediction value, we simply get the difference between the predicted value and the actual target value of that data point.

$$\frac{\partial}{\partial \hat{y}} \left(\frac{1}{2}(y - \hat{y})^2 \right) = \hat{y} - y \tag{3.1}$$

Evaluated derivative is then used as a new target value of the data points during training a new tree. The algorithm subsequently calculates the new residual based on aggregating the predictions of the previous trees with the prediction of the new one. This process is repeated for each subsequent tree.

[1] The coefficient is often used for convenience and simplicity in the subsequent mathematical calculations, without fundamentally affecting the result.

Chapter 4

Search Engine Context

One industry where machine learning models derived from decision trees are used is in search engines. Within its search engine, Seznam.cz company uses the Gradient Boosting on Decision Trees model (inspected in 3.2) to evaluate the relevance of search results. The reason is not only the speed of the evaluation, but also the interpretability.

Seznam.cz divides search queries according to what user probably wanted to achieve by formulating the query. Several types of user intents are distinguished, the main ones include:

- Navigation: The user wanted to go to a speci c page and expects this result on the rst place.
- Company: The user is looking for a speci c company, institution or web service.
- Goods: The user wants to buy, order or get something.
- Learn more/advice: A wide range of information queries ranged from troubleshooting to using online tools like calculator.
- Queries about news or current information.

These groups of queries have their own relevance evaluation model trained on di erent data.

Each sample in these training datasets contain a query string and a speci c URL[1] address which could possibly appear on the search engine results page after entering this query. The job of the annotator in Seznam.cz is to label these pairs in the training data with the number 1 (positive label) when it is a relevant pair or with the 0 (negative label) otherwise[2].

Samples in datasets are then enriched with properties derived from the query-URL pair. An example of a less sophisticated property would be the number of letters that appear in both the query and the URL. These continuous features are then used for training the model.

[1] Uniform Resource Locator, a reference to a web resource that speci es its location on a computer network.

[2] As a target variable the "box" feature is used in validation set.

Figure 4.1: Search engine results page (SERP) with query string "libre o ce" in the search bar and relevant URL addresses cs.libreo ce.org and libreo ce.org

query	url	box	feature_0
libre o ce	https://cs.libreo ce.org/	1.0	255.0
lidl.cz	https://www.lidl.cz/	1.0	250.0
perfect clinic	https://www. rmy.cz/	0.0	255.0

Table 4.1: Simpli ed example of navigation query validation data

Although the data are more of a classi cation nature (having only two types of target variables in the training dataset), in practice, regression models are employed. This model, upon being trained and provided with a set of features derived from a new unseen test pair, generates a oat value indicating its perceived relevance. Generally, the higher this value is, the more relevant the pair is and the more likely the URL address is to appear on the search engine results page. The speci c threshold for inclusion of an URL address on the results pages is then set based on the previous analysis of machine learning metrics as precision and recall[3].

Seznam.cz uses GBMT models implemented by the CatBoost open-source library, which is developed by Yandex, a Russian multinational technology company. For practical reasons, CatBoost incorporates symmetric oblivious trees (described in 2.1.1) in its models.

This work uses a simpli ed CatBoost model stored in JSON format (4.2) and a validation data set containing navigation queries, both provided by Seznam.cz for the purposes of research and visualization design. As part of con dential company procedures, the original feature names in the validation dataset are replaced by universal names according to the feature indexes. So, instead of names that describe how the given feature was derived from the string query-URL pair, the names as feature_0, feature_1 and so on are used.

[3] https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/

Figure 4.2: Structure of the CatBoost model JSON le provided by Seznam.cz

Chapter 5

Exploring Contemporary Decision Tree Visualization Techniques

As machine learning models grow in complexity, gaining deeper insight into their behavior and performance becomes more important. One way to get such an understanding of model behavior is to track universal metrics common to all machine learning models. To gain deeper knowledge about the internal behavior of the model, the visualization of the model structure itself, which is speci c to each model type, can be explored.

Common general metrics that are often visualized in the analysis may include, for example, feature importance. It quanti es the in uence of each feature value on model predictions. By analyzing feature importance, we can identify the most signi cant factors driving the model's decision-making process.

Figure 5.1: Graph of feature importance

Although the calculation of feature importance can be performed for various machine learning models, the speci c method used may vary depending on the model type. In CatBoost models it is calculated by a measuring how much on average the prediction changes if the feature value changes [16]. Some other gradient boosting machine implementations assign feature importance based on the number of times a feature is selected for splitting across all the trees in the ensemble.

In the following sections, model-speci c visualizations methods related to decision trees will be explored. Some of these techniques go beyond just showing the tree structure but also provide additional information about how data ows through the model. Only methods available through python-based open-source libraries were considered.

Most of the following visualization libraries are not directly compatible with models created by the CatBoost library due to di erent implementations. In such cases, for the purpose of visualization demonstration, a model supported by these libraries was trained using validation data provided by Seznam.cz (in 5.1.2 and 5.2.1). In another case, for the visualization demonstration, a model of the supported type was trained using data available in the library's documentation (in 5.2.2). Given the classi cation nature of the data (described in 4), it was possible to train both regression and classi cation models from it. The properties of these visualizations were then utilized in the proposed solution.

5.1 Capturing Tree Structure and Node Details

One fundamental aspect of decision tree visualization is representing the hierarchical structure of the tree itself. Traditional methods often focus solely on displaying branching structure and target variable associated with each leaf node. The most common ones provide at least some understanding of the data distribution across the tree.

5.1.1 CatBoost

CatBoost o ers its own built-method for plotting individual trees from the ensemble. This visualization with split functions in internal nodes, branch descriptions and outcome values in leaf nodes can serve as a base-level solution.

Figure 5.2: Part of the decision tree structure visualization made by the Catboost library

5.1.2 scikit-learn

Popular machine learning library scikit-learn o ers several types of decision tree models and models derived from decision trees such as regression and classi cation type of standard decision tree, random decision forest or gradient boosting machines. In addition to their training and testing, it also provides their visualization. Apart from the primary structure, it also captures basic information about the data ow.

Visualization of the regressor (shown in 5.3) includes the counts of samples from the training dataset present in each node. Each node also contains the average target value of the samples present in it. Within the leaf nodes, these values serve as outcomes of the model.

Higher target values in the node are indicated by darker colors, intuitively conveying information about the distribution of data and the e ectiveness of given split functions. Squared error as a measurement of impurity (described in 2.2) that was used by the model during training can be also seen in the node content.

Figure 5.3: Visualization of the regressor decision tree structure by the scikit-learn library

Classi cation visualization (in 5.4) further divides the count of samples according to their target classes. Nodes are colored based on the class of the majority of samples present. The more is the color of the node satured the more samples of one speci c class have majority over others. Node Gini impurity (described in 2.2) is also provided.

5.2 Additional Information about Data Flow

While capturing the tree structure and node details is valuable, better understanding how the model processes and routes data through the decision tree is equally important. Contemporary decision tree visualization methods go beyond static representations and incorporate dynamic visualizations that showcase more detailed ow of validation data as it traverses the tree. Some of these techniques o er insights into the path taken by individual samples, ilustrating the decision-making process and highlighting the speci c split functions encountered along the way.

Figure 5.4: Visualization of the classi er decision tree structure by the scikit-learn library

5.2.1 dtreeviz

The dtreeviz library designed speci cally for decision tree visualization and model interpretation captures in detail the split feature versus target value distributions and the number of samples in individual nodes. As mentioned by the creators of the library, whose visualizations are enriched with multiple types of graphs, the goal is to determine how well the target values can be separated based on the split functions [6].

In the classi cation decision tree visualization (in 5.5) we can see how dtreeviz attempts to address the limitations of traditional scikit-learn visualizations from 5.1.2. Internal and leaf nodes are categorically distinguished using suitable types of graphs. Stacked histograms, with the x-axis representing the feature space and an arrow indicating the threshold, are used in internal nodes. Although the histograms clearly convey information about how far the values of split feature instances deviate from the thresholds, the information about sample counts in nodes is not as readable.

Leaf nodes are represented by pie charts that clearly show their purity.

In the regression variant (shown in 5.6), histograms and pie charts are replaced by scatter plots where y-axis corresponds with scale of target value. Horizontal dashed lines indicate the target mean for the left and right buckets and a vertical dashed line indicates the thresholds in feature space. The visualization of our model, or rather the data, is somewhat unfortunate in this case due to its nature (described in 4). The distribution of points in a scatter plot would be more useful in the case where the data has a target variable that can take more than two values.

Among other things, the library also o ers a look at how a speci c data sample is run down the tree to a leaf (shown in 5.7). Below the leaf node, to which the highlighted path of the sample leads, we can see a list of sample's split feature values that were used. This helps explain why a particular instance gets the prediction it does.

The dtreeviz library also allows visualizing the tree with di erent data than the one used in training, unlike the scikit-learn library. This enables

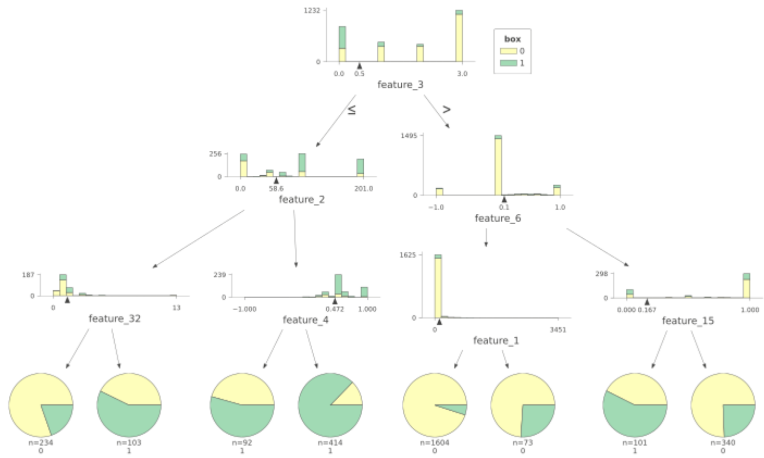

Figure 5.5: Visualization of the decision tree classifier structure by the dtreeviz library

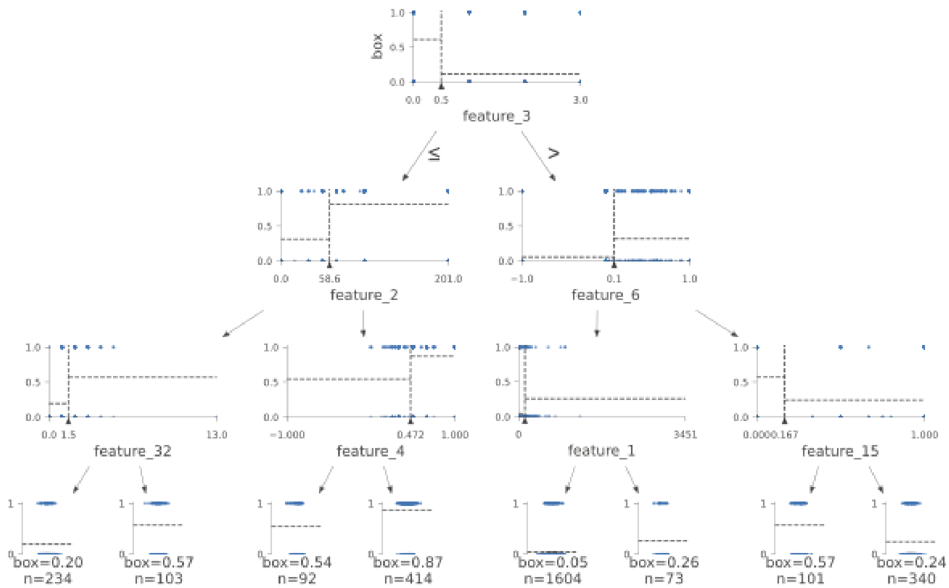

Figure 5.6: Visualization of the decision tree regressor structure by the dtreeviz library

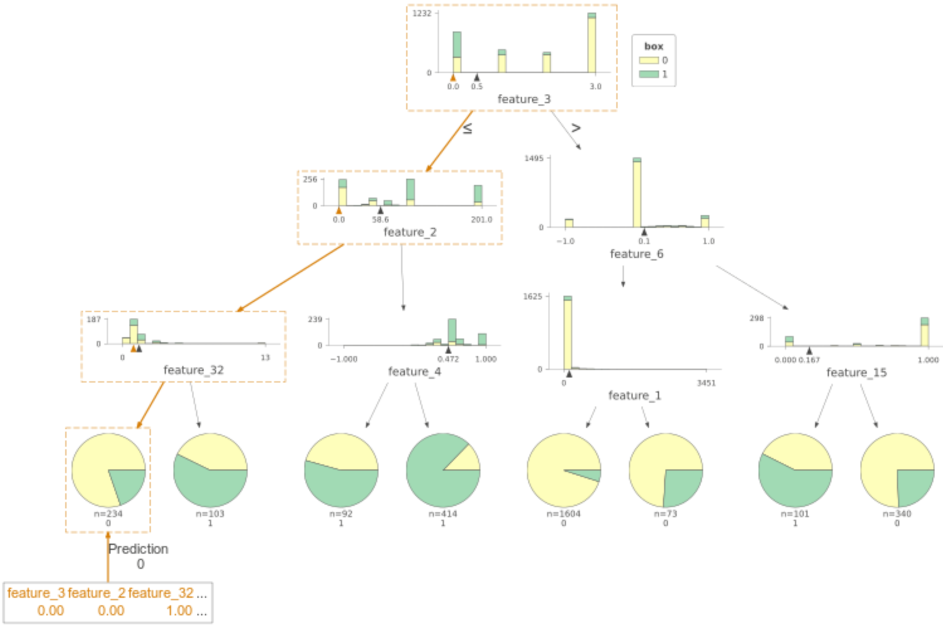

Figure 5.7: Visualization of the classification decision tree with the path of a specific instance

testing the tree structure on new validation data.

5.2.2 Pybaobabdt

Based on the study [17], the Pybaobabdt library from 2021 introduces a unique technique for visualizing classification decision trees. It replaces the classical node-link construction with a data-flow-based technique that seamlessly integrates tree visualization with data visualization. Each class is represented by a color, and the thickness of the stream corresponds to the number of instances present in a given branch or node.

This technique effectively and intuitively conveys certain information. It is clear which classes are easy to separate, which classes are similar or where does the main flow of items go. Additionally, this visualization is significantly more scalable than the previous ones.