Heinrich Heine Universität Düsseldorf

logicDT: a procedure for identifying response-associated interactions between binary predictors

Michael Lau, Tamara Schikowski & Holger Schwender

Article - Version of Record

Suggested Citation:
Lau, M., Schikowski, T., & Schwender, H. (2023). logicDT: a procedure for identifying response-associated interactions between binary predictors. Machine Learning, 113(2), 933–992.
https://doi.org/10.1007/s10994-023-06488-6

Wissen, wo das Wissen ist.

UNIVERSITÄTS- UND LANDESBIBLIOTHEK DÜSSELDORF

This version is available at:

URN: https://nbn-resolving.org/urn:nbn:de:hbz:061-20250307-113819-1

Terms of Use:

This work is licensed under the Creative Commons Attribution 4.0 International License.

For more information see: https://creativecommons.org/licenses/by/4.0

Machine Learning (2024) 113:933–992
https://doi.org/10.1007/s10994-023-06488-6

logicDT: a procedure for identifying response-associated interactions between binary predictors

Michael Lau[1,2] · Tamara Schikowski[2] · Holger Schwender[1]

Received: 3 November 2022 / Revised: 13 October 2023 / Accepted: 16 November 2023 /
Published online: 22 December 2023
© The Author(s) 2023

Abstract
Interactions between predictors play an important role in many applications. Popular and successful tree-based supervised learning methods such as random forests or logic regression can incorporate interactions associated with the considered outcome without specifying which variables might interact. Nonetheless, these algorithms suffer from certain drawbacks such as limited interpretability of model predictions and difficulties with negligible marginal effects in the case of random forests or not being able to incorporate interactions with continuous variables, being restricted to additive structures between Boolean terms, and not directly considering conjunctions that reveal the interactions in the case of logic regression. We, therefore, propose a novel method called logic decision trees (logicDT) that is specifically tailored to binary input data and helps to overcome the drawbacks of existing methods. The main idea consists of considering sets of Boolean conjunctions, using these terms as input variables for decision trees, and searching for the best performing model. logicDT is also accompanied by a framework for estimating the importance of identified terms, i.e., input variables and interactions between input variables. This new method is compared to other popular statistical learning algorithms in simulations and real data applications. As these evaluations show, logicDT is able to yield high prediction performances while maintaining interpretability.

Keywords Decision trees · Interpretable machine learning · Regression procedure · Variable importance measures · Importance of interactions · Polygenic risk scores

Editor: Johannes Fürnkranz.

✉ Michael Lau
michael.lau@hhu.de

Tamara Schikowski
tamara.schikowski@iuf-duesseldorf.de

Holger Schwender
holger.schwender@hhu.de

[1] Mathematical Institute, Heinrich Heine University, Universitätsstrasse 1, 40225 Düsseldorf, Germany

[2] IUF - Leibniz Research Institute for Environmental Medicine, Auf'm Hennekamp 50, 40225 Düsseldorf, Germany

1 Introduction

In many practically relevant applications, a proper coverage of interactions between predictors is key for constructing strong predictive models. One particularly important example is the analysis of genetic or environmental risk factors in epidemiological and medical studies for, e.g., constructing genetic/polygenic risk scores (Che & Motsinger-Reif, 2013; Ho et al., 2019) that can be viewed as a function $\varphi : \mathcal{X} \to \mathcal{Y}$ from the p-dimensional space $\mathcal{X} = \{0, 1, 2\}^p$ of p SNPs (single nucleotide polymorphisms), i.e., single base-pair substitutions in the DNA, to the response space $\mathcal{Y}$ assigning a risk estimate. For example, for a binary outcome such as a binary disease status, a probability estimate $\hat{P}(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x}) \in [0, 1]$ of developing this disease might be a proper risk estimate. Since SNPs are variables with three possible outcomes counting the number of minor allele occurrences with respect to both chromosomes, i.e., how often the less frequent variant occurs in an individual, they can be easily (and biologically meaningful) divided into two binary variables each, i.e., in $\text{SNP}_D = \mathbb{1}(\text{SNP} \neq 0)$ and $\text{SNP}_R = \mathbb{1}(\text{SNP} = 2)$, coding for a dominant and a recessive effect, respectively. It is well-known that in the analysis of genetic features such as SNPs, interactions, e.g., gene-gene interactions (Che & Motsinger-Reif, 2013) and gene-environment interactions (Ottman, 1996), play a crucial role. Especially in this setting, not only a high predictive ability of the resulting models, but also a high interpretability for understanding which and how genetic variants influence the risk of disease is desirable.

Tree-based statistical learning methods such as decision trees, random forests, or logic regression are very popular and versatile in recognizing underlying data structures. These methods have been already applied to analyze SNP data (e.g., Bureau et al., 2005; Winham et al., 2012; Ruczinski et al., 2004). However, these methods typically fail at simultaneously achieving a reliable predictive strength and a high interpretability of how exactly predictions are composed.

In this article, we propose the tree-based supervised learning procedure *logicDT* (logic decision trees) which is specifically tailored for properly incorporating interactions between binary predictors. Continuous relationships of additional covariates and interactions of these covariates with the binary variables can also be covered by this procedure. logicDT is designed for yielding highly interpretable prediction models, while maintaining a high predictive ability. For measuring the influence of predictors and their interactions, a novel variable importance measure framework is proposed which, in principle, can be used in conjunction with any other learning procedure.

We start with briefly discussing similar methods and efforts on enhancing existing algorithms in Sect. 2. Then, logicDT and its extensions are presented in detail in Sect. 3. We additionally prove that logicDT is consistent. In Sect. 4, the novel variable importance measuring framework for estimating the influence of input variables and their interactions is proposed. Empirical studies on simulated data as well as on real data follow in Sect. 5 illustrating logicDT's properties in practice and comparing logicDT to other procedures. Sections 6 and 7 contain discussions and concluding remarks.

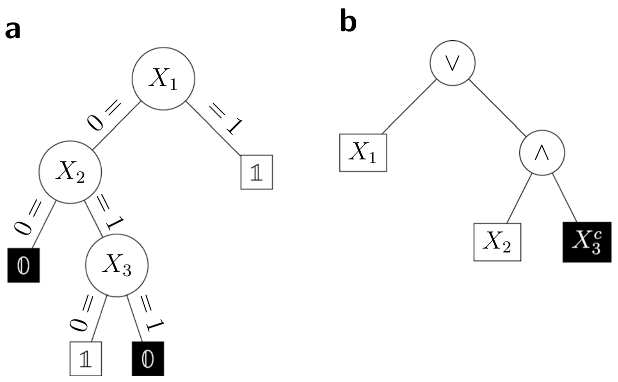


Fig. 1 Exemplary tree models for three binary input variables X_1, X_2 and X_3 predicting two different classes 0/false and 1/true. In **a**, a classification tree is shown. **b** depicts a logic tree describing the Boolean expression $X_1 \vee (X_2 \wedge X_3^c)$, where negations are denoted by ^{c} in this article. For the logic tree, terminal nodes containing negated predictors are depicted as black squares containing white text. Vice versa, non-negated predictors are depicted as white squares containing black text. Both trees are equivalent, i.e., they perform the same predictions for each predictor setting. Adapted from Lau et al. (2022)

2 Background and related work

In the following, we briefly discuss tree-based supervised learning procedures and their extensions.

2.1 Decision trees and random forests

One very popular and powerful statistical learning method are decision trees. Important implementations include classification and regression trees (CART) (Breiman et al., 1984) and C4.5 (Quinlan, 1993). Decision trees recursively partition the predictor space $\mathcal{X}$ considering one predictor per split into disjoint patches, to which individually a prediction value will be assigned. For predicting new outcomes, one starts at the root node and follows the edges corresponding to the specific predictor setting until a leaf is reached. Figure 1a illustrates an exemplary decision tree consisting of three binary predictors in a binary classification scenario.

Algorithm 1: Decision Tree Fitting

```
function fitDecisionTree(Training data $\mathcal{D}$):
    Create an empty decision tree $T$ with root node $t_0$
    Initialize an empty stack where each element is a tuple $(t, \mathcal{D}_t)$
    stack.push($(t_0, \mathcal{D})$)
    while $|stack| > 0$ do
        $(t, \mathcal{D}_t)$ = stack.pop()
        if Stopping criterion is met then
            $\widehat{Y}_t = \frac{1}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} y$
        else
            splits = Initialize empty list
            for every input variable $X_j$ do
                splits.append(Best split on $X_j$)
            end
            $s^*$ = Best split in splits
            Split the inner node $t$ in $T$ on $s^* = (\mathcal{X}_{t_L}, \mathcal{X}_{t_R})$
            stack.push($(t_L, \mathcal{D}_{t_L})$)
            stack.push($(t_R, \mathcal{D}_{t_R})$)
        end
    end
    return $T$
end
```

Similar to Louppe (2014), Algorithm 1 summarizes the fitting process of decision trees. In Lines 11 through 14, the locally best split, i.e., the predictor and the splitting point which maximize the node homogeneity after splitting is identified and used for further splitting the tree into two subnodes. For measuring the homogeneity, an impurity measure i is used which assigns a node an estimate of its heterogeneity. For evaluating the strength of a split s partitioning the node t into two child nodes t_L and t_R, the impurity reduction

$$\Delta i(s,t) \; := \; i(t) - \frac{n_{t_L}}{n_t} i(t_L) - \frac{n_{t_R}}{n_t} i(t_R) \; \geq \; 0 \tag{1}$$

for the number of training observations n_t falling into node t is maximized. For regression purposes, the impurity measure of the mean squared error

$$i_{\text{Regression}}(t) \; := \; \frac{1}{n_t} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_t} (y - \hat{y}_t)^2 \tag{2}$$

is used as the impurity measure considering the subset $\mathcal{D}_t$ of the training data set $\mathcal{D}$ to node t and the predicted outcome $\hat{y}_t$ in node t. For classification or risk estimation, the Gini impurity

$$i_{\text{Gini}}(t) \; := \; \sum_{c\in\mathcal{Y}} \frac{n_{c,t}}{n_t} \left(1 - \frac{n_{c,t}}{n_t}\right) \tag{3}$$

is used for classes $c \in \mathcal{Y}$ and their corresponding frequency $n_{c,t}$ in node t. An alternative popular impurity measure for classification tasks is the information gain

$$i_{\text{Entropy}}(t) := -\sum_{c \in \mathcal{Y}} \frac{n_{c,t}}{n_t} \log_2 \left(\frac{n_{c,t}}{n_t} \right)$$

that is based on the Shannon entropy (e.g., Louppe, 2014).

The partitioning of a tree branch locally stops when the training data cannot be further divided, i.e., if for all $(\boldsymbol{x}, y), (\boldsymbol{x}', y') \in \mathcal{D}_t$, it either holds $\boldsymbol{x} = \boldsymbol{x}'$ or $y = y'$ (see Line 7 of Algorithm 1). Usually, to prevent overfitting, additional stopping criteria are used such as the minimum node size, i.e., the minimum number of training observations falling into a leaf, or a minimum impurity reduction which has to be achieved in order to split the node. However, these additional stopping criteria yield hyperparameters which, thus, require proper tuning. Finally, the last important step is the assignment of a predicted value to a leaf (Line 8 of Algorithm 1). Although theoretically, this predicted value is already used for evaluating the splits. The prediction values are obtained by empirical risk minimization yielding the arithmetic mean for regression tasks. For binary risk estimation, also the arithmetic mean of the outcome Y given the predictor values $\boldsymbol{x}$ is used if Y is coded as 0 or 1. If pure classifications are considered, the class with the lowest risk estimate is chosen.

A particularly popular and successful extension of decision trees are random forests which build ensembles of randomized decision trees yielding even higher predictive performance at the cost of losing interpretability of the fitted models (Breiman, 2001). The randomization is performed by employing bagging (Breiman, 1996), which is described in more detail in Sect. 3.9, and by considering random predictor subsets for splitting at each node. Random forests can substantially outperform single decision trees due to the instability issue of decision trees, which states that small noise-like changes of the training data set can lead to large modifications of the fitted model. This instability issue is mainly caused by the greedy fashion of choosing splits (Li & Belford, 2002; Murthy & Salzberg, 1995).

If deep trees are grown, both single decision trees and random forests can overfit (Hastie et al., 2009; Tang et al., 2018). For certain, not necessarily realistic scenarios (e.g., no subsampling combined with totally randomized trees in which the splits are chosen independent of the outcome or too extreme subsampling in which the subsample size remains constant, but the sample size approaches infinity), Tang et al. (2018) proved that random forests with deeply grown trees are inconsistent.

If shallow trees are grown, fruitful splits might be left out. Furthermore, decision trees and random forests struggle uncovering interactions effects, if the interacting variables only exhibit negligible marginal effects (Wright et al., 2016). Moreover, due to the prediction values of the leaves being constant for finitely many predictor scenarios in conventional decision trees and random forests, continuous function relationships can only be approximated by step functions. However, for example, in the analysis of genetic and environmental risk factors of certain diseases, in which random forests are frequently used (Winham et al., 2012; Bellinger et al., 2017), a continuous influence of an environmental factor on the disease risk is reasonable.

There are a variety of modifications to decision trees and random forests which try to overcome the issues mentioned above. These methods, however, address individual issues. In the following section, we will discuss some of these modifications.

2.2 Extensions of decision trees and random forests

For improving the ability on detecting interactions, one well-known approach is the usage of multivariate splits, i.e., splits based on multiple variables at once, e.g., by using linear combinations of the predictors. Exemplary methods of this class are oblique decision trees (Murthy et al., 1994) and oblique random forests (Menze et al., 2011), where a particular implementation of the latter is, e.g., SPORF (Sparse Projection Oblique Randomer Forests; Tomita et al., 2020). For binary predictors as considered in this article, these multivariate linear splits can be used for creating Boolean conjunctions of predictors, thus, potentially splitting on an interaction. However, methods that try to linearly separate the current feature space based on the (binary) class label in each splitting node (such as the method proposed by Menze et al., 2011) are only suited to classification tasks. Another recent modification is interaction forests (Hornung & Boulesteix, 2022) which directly searches for interaction splits at each node. An overview over such interaction-focused modifications of decision trees and random forests is, e.g., given by Hornung and Boulesteix (2022).

The greedy search algorithm employed in classic decision tree fitting procedures (such as in CART) is fast and scales to high-dimensional problems. However, as the greedy search conducts local searches for splits, it requires detectable marginal effects to identify interaction effects. For example, if X_1 and X_2 interact with each other, X_1 or X_2 have to be individually identified first as splitting variables. Due to increasing computational capabilities, optimal decision trees have been proposed by Nijssen and Fromont (2010) and Bertsimas and Dunn (2017) to perform a global optimization. In the former method, namely DL8 (decision trees from lattices), dynamic programming is utilized to fit decision trees. In the latter method, namely OCT (optimal classification trees), the decision tree fitting problem is phrased as a mixed-integer optimization problem. More recently, alternative optimal decision tree algorithms that utilize dynamic programming such as DL8.5 (Aglin et al., 2020a) and MurTree (Demirović et al., 2022) and optimal decision tree fitting procedures that incorporate multivariate splits such as WODT (Yang et al., 2019) and SVM1-ODT (Zhu et al., 2020) have been proposed. A review of optimal decision tree fitting procedures is, e.g., given by Carrizosa et al. (2021).

Blockeel and De Raedt (1998) proposed combining decision trees with logic programming. Their method is called TILDE (top-down induction of logical decision trees). At each inner node, a Boolean conjunction is responsible for further partitioning the input data. Model fitting is performed in a greedy fashion very similar as in C4.5 (Quinlan, 1993). However, the space of eligible splits, over which the greedy search is applied, has to be defined by the user by utilizing background knowledge and, e.g., specifying which variables may be part of the same conjunction. Another important difference between TILDE and other decision tree algorithms is that TILDE uses logic programs for specifying data examples. This is in contrast to the statistical learning setup considered in this article. We consider the standard setting, in which data are given in a tabular format and relevant background knowledge about the relationships of certain variables is not available.

Rule extraction methods aim at increasing the interpretability of tree ensemble methods while keeping their predictive strength. They start by fitting a tree ensemble such as random forests and try to extract the most important prediction rules from the individual decision tree paths. These prediction rules are then gathered in rule lists yielding the final model, in which predictions are made according to which rules hold true. One of the first and most established rule extraction methods is RuleFit (Friedman & Popescu, 2008), which fits a boosted ensemble of decision trees and selects the most important rules using the lasso (Tibshirani, 1996). Alternative rule extraction methods include node harvest (Meinshausen, 2010) and SIRUS (Stable and Interpretable Rule Set, Bénard et al., 2021), which both fit random forests for generating the models from which the rules are to be extracted.

For modeling continuous regression models in the leaves, typically, GLMs are employed such as in MOB (model-based recursive partitioning, Zeileis et al., 2008). An overview on several GLM-based approaches is, e.g., given by Rusch and Zeileis (2013). However, the right parametric model might not be known prior to fitting models so that a more flexible non-linear regression model might be preferable. Moreover, these methods do not lay a focus on properly handling interactions between the splitting variables.

2.3 Logic regression

Logic regression (Ruczinski et al., 2003) is another tree-based supervised learning method. It has been specifically developed for analyzing SNP data and is, therefore, frequently used in such analyses (e.g., Ruczinski et al., 2004; Zhi et al., 2015). Logic regression is focussed on binary predictors and tries to identify Boolean combinations of the predictors that shall explain the variation in the outcome. These Boolean expressions can also be presented as logic trees, i.e., trees holding predictors (or their negations) in their leaves and recursively combining them with the Boolean AND-operator (denoted by $\wedge$ in the following) or the Boolean OR-operator (denoted by $\vee$ in the following) using inner nodes. Figure 1b illustrates an exemplary logic tree corresponding to the Boolean expression $X_1 \vee (X_2 \wedge X_3^c)$. If a true logic tree is identified with class 1 and a false logic tree is identified with class 0, this tree is equivalent to the classification tree from Fig. 1a.

To generalize the usage of logic regression to regression purposes, logic trees are embedded in GLMs, i.e., a model of the form

$$g(\mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}]) = \beta_0 + \beta_1 L_1(\boldsymbol{x}) + \cdots + \beta_m L_m(\boldsymbol{x})$$

is considered for a link function g and logic trees $L_1, \dots, L_m$. In general, every possible logic regression model can be transformed into an equivalent decision tree, and vice versa (Ruczinski et al., 2003). However, logic trees tend to be more sparse, i.e., by using Boolean logic, logic trees can describe the same prediction model with fewer nodes than decision trees in certain scenarios. For example, even in the simple prediction model depicted in Fig. 1, the logic tree consists of five nodes, whereas seven nodes are required in the CART tree to represent the Boolean expression. Note that this tree sparsity property holds true for binary classification scenarios in which a hard classification task instead of a more general class probability estimation task is considered.

The fitting procedure in logic regression is performed by a global stochastic search over all possible models, i.e., logic trees $L_1, \dots, L_m$ and their GLM coefficients $\beta_0, \dots, \beta_m$, where these GLM coefficients are determined by fitting a GLM using the considered logic trees as predictors in each step of the global stochastic search. In particular, simulated annealing (Kirkpatrick et al., 1983) is employed using simple modifications of the current model/state, i.e., adding or removing branches, exchanging variables or operators, and splitting or removing variables. Alternatively, a greedy local search always moving to the best neighbor state can be employed. However, this faster search comes without any guarantees of finding a globally optimal state. For evaluating the current state, a score function such as the mean squared error for linear regression or the deviance for logistic regression is used. For a detailed description and discussion of logic regression, see Ruczinski et al. (2003).

Single logic regression models tend to be unstable, if the signal is weak or if many predictors are actually predictive. One approach to tackle this problem is to apply bagging to logic regression models (Schwender & Ickstadt, 2007). However, similar to random forests, these models are no longer easily interpretable.

Even single logic regression models can be hard to interpret due to possibly complex logic tree structures. Typically, one is interested in the statistical interaction of predictors, which can be defined as the effect of the presence of certain predictor settings at once, i.e., using Boolean conjunctions, since conjunctions of input variables directly reveal the specific type of interaction that is considered (Chen et al., 2011). By De Morgan's laws, if a Boolean disjunction needs to be represented, the negation of the conjunction containing the negations of the input terms can be used, i.e., making disjunctions obsolete if all negations are available.

Logic regression can only take quantitative covariables additively into account by adding them to the linear predictor of the GLM containing the logic trees as single terms. Thus, no interactions between the binary predictors and quantitative predictors can be included. Similarly, interactions between logic trees themselves can also not be captured, thus, relying on the additive structure of the individual terms. If, for example, the scale of an underlying linear predictor is unknown, being able to also model interactions between the terms can be beneficial. Consider, e.g., the regression function

$$\begin{aligned}\mathbb{E}[Y \mid \boldsymbol{X}] &= \left[\alpha \cdot \mathbb{1}(X_1) + \beta \cdot \mathbb{1}(X_2 \wedge X_3^c)\right]^2 \\ &= \alpha^2 \cdot \mathbb{1}(X_1) + 2 \cdot \alpha \cdot \beta \cdot \mathbb{1}(X_1 \wedge X_2 \wedge X_3^c) + \beta^2 \cdot \mathbb{1}(X_2 \wedge X_3^c).\end{aligned}$$

On the squared scale, the terms X_1 and $X_2 \wedge X_3^c$ do not interact. However, on the original scale, if both terms are true at once, the linear predictor is adjusted by an additional $2\alpha\beta$.

3 Logic decision trees

To overcome the issues mentioned in the last section, we propose a novel method, called *logicDT (logic decision trees)*, which combines decision trees and an improved version of the Boolean term search of logic regression.

We define logic decision trees to be decision trees that can use Boolean conjunctions of input variables as splitting variables, which is in contrast to standard decision tree procedures. Logic decision trees may be used for regression purposes, in which—similar to regression trees—each leaf holds a direct estimate of the outcome, or for classification purposes, in which—similar to probability estimation trees (Provost & Domingos, 2003; Malley et al., 2012)—each leaf holds an estimate of the class membership probability. As discussed in Sect. 3.5, logic decision trees may also contain regression models in their leaves for modeling continuous relationships.

Allowing Boolean conjunctions of input variables as splitting variables, firstly, simplifies the resulting decision tree. If we, e.g., consider an outcome that is only altered if $X_1^c \wedge X_2$ holds, then creating a tree stump (i.e., a decision tree consisting of only one split) splitting on $X_1^c \wedge X_2$ would be sufficient when using logicDT, whereas a common decision

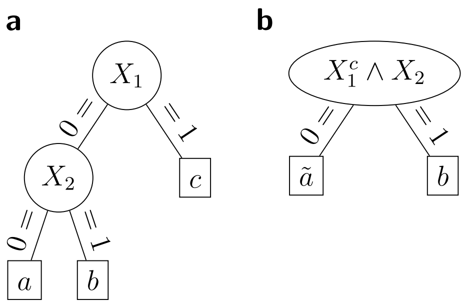


Fig. 2 Decision trees for splitting on $X_1^c \wedge X_2$. In **a**, a standard decision tree splitting on single input variables is shown. In **b**, a Boolean conjunction is used for splitting

tree only using single input variables for splitting would require a split on X_1 and another split on X_2 in the branch in which $X_1 = 0$ holds (see Fig. 2).

Secondly, this makes the prediction values in some leaves more robust. In our example, the common decision tree in Fig. 2a would further distinct between $X_1 = 1$ and $X_1^c \wedge X_2^c = 1$, while the tree in Fig. 2b uses one shared prediction, thus, utilizing more observations for creating the prediction value. Thirdly, due to the greedy search employed in standard decision tree splitting approaches, the interaction might not be found due to potentially negligible marginal effects of, in our example, X_1 or X_2 leading to splitting on other variables or not splitting at all, if a stopping criterion is triggered.

In the following subsections, logicDT is presented in detail.

3.1 Preliminaries

Let $\boldsymbol{X} = (X_1, \dots, X_p)$ be a p-dimensional random vector of binary input variables taking values in the p-dimensional space $\mathcal{X} = \{0, 1\}^p$ and let Y be a target random variable taking values in the space $\mathcal{Y}$. Let $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\}$ be a training data set with independent and identically distributed observations from the joint probability distribution of $(\boldsymbol{X}, Y)$. Then the corresponding statistical learning task can be formulated as estimating the true regressor $\mathbb{E}_{(\boldsymbol{X},Y)}[Y \mid \boldsymbol{X} = \cdot\,]$ by a function $\varphi : \mathcal{X} \to \mathcal{Y}$ using the training data set $\mathcal{D}$ (e.g., Hastie et al., 2009).

In this article, Boolean conjunctions between binary input variables are denoted using the Boolean $\wedge$ (AND) and negations of binary input variables are denoted using a superscript ^{c} (complement), i.e., $X_j^c = 1 - X_j$.

logicDT is aimed at identifying response-associated interactions, where two input variables X_i and X_j are defined to interact with each other with respect to the outcome Y, if the effect of one input variable (i.e., the partial derivative/finite differences of $\mathbb{E}[Y \mid \boldsymbol{X}]$ with respect to one input variable) depends on the other input variable (Sorokina et al., 2008). Therefore, if there is no interaction between X_i and X_j, the regression function $\mu(\boldsymbol{X}) = \mathbb{E}[Y \mid \boldsymbol{X}]$ can be decomposed into a sum $\mu(\boldsymbol{X}) = \mu_{\setminus i}(\boldsymbol{X}_{\setminus i}) + \mu_{\setminus j}(\boldsymbol{X}_{\setminus j})$, where $_{\setminus i}$ denotes leaving out the ith entry of the vector of input variables (Friedman & Popescu, 2008). This definition can be directly generalized to (statistical) interactions of arbitrary order. If there is no interaction between $X_{(1)}, \dots, X_{(k)}$, μ can be decomposed into a sum of functions, in which no summand is a function of all considered variables $X_{(1)}, \dots, X_{(k)}$ simultaneously.

In this article, we mainly focus on binary input variables. Therefore, every function $\varphi : \mathcal{X} \to \mathcal{Y}$ mapping from a p-dimensional space of binary input variables to a real number can be expressed as a sum of the form

$$\varphi(X) = \beta_0 + \sum_{j=1}^{m} \beta_j \cdot \mathbb{1}\left(X_{k_{j,1}}^{(c)} \wedge \cdots \wedge X_{k_{j,p_j}}^{(c)}\right),$$

where $^{(c)}$ denotes potentially negating the considered variable and $k_{j,i}$ is the index of the ith variable in the jth summand. Hence, binary input variables $X_{(1)}, \dots, X_{(k)}$ interact with each other (with respect to Y), if μ cannot be decomposed without using a Boolean conjunction that simultaneously includes $X_{(1)}, \dots, X_{(k)}$. Boolean disjunctions are not considered in logicDT, since, by De Morgan's laws, Boolean disjunctions can be expressed using Boolean conjunctions and negations.

3.2 Core methodology of logicDT

The aim of logicDT is to identify important input variables and Boolean conjunctions of input variables to perform accurate predictions of the outcome. An input variable or a Boolean conjunction of input variables will be in the following referred to as a *term*. A set of terms will be referred to as a *state*. Examples of possible states would be

$$\{\{X_{73}\}\} \quad \text{or} \quad \{\{X_1^c \wedge X_2\}, \{X_5\}, \{X_9 \wedge X_{14}^c \wedge X_{42}^c\}\}.$$

In logicDT, states are obtained by a global stochastic search procedure that is introduced later in this section.

Logic decision trees are induced by identifying a state and exclusively using the terms contained in this state as input variables for fitting a conventional decision tree. For example, the three terms $X_1^c \wedge X_2$, X_5, and $X_9 \wedge X_{14}^c \wedge X_{42}^c$ are used as input variables to induce a decision tree, if the corresponding state $\{\{X_1^c \wedge X_2\}, \{X_5\}, \{X_9 \wedge X_{14}^c \wedge X_{42}^c\}\}$ is considered. Hence, creating a logic decision tree based upon a state is a two-stage procedure. First, the original training data set is transformed into a *tree training data set* using the terms of the considered state. Next, using this tree training data set, a decision tree is fitted.

For a set consisting of m terms

$$\left\{\left\{X_{k_{1,1}}^{(c)}, \dots, X_{k_{1,p_1}}^{(c)}\right\}, \dots, \left\{X_{k_{m,1}}^{(c)}, \dots, X_{k_{m,p_m}}^{(c)}\right\}\right\},$$

the original training data set is transformed into a tree training data set by constructing a $n \times (m+1)$ data matrix containing the m different predictors or conjunctions and the outcome. For example, if a training data set is given by

$$\mathcal{D} = \left[\begin{array}{c|c} \boldsymbol{x}_1 & y_1 \\ \boldsymbol{x}_2 & y_2 \\ \boldsymbol{x}_3 & y_3 \\ \boldsymbol{x}_4 & y_4 \end{array}\right] = \begin{array}{c} \begin{array}{cccc} X_1 & X_2 & X_3 & Y \end{array} \\ \left[\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \end{array}$$

and the state $s = \{\{X_1\}, \{X_2 \wedge X_3^c\}\}$ is identified by the global stochastic search, the tree training data set, which is directly used for fitting the decision tree, is given by

$$\mathcal{D}_s = \begin{array}{c} \begin{array}{ccc} X_1 & X_2 \wedge X_3^c & Y \end{array} \\ \left[\begin{array}{cc|c} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right] \end{array}. \tag{4}$$

Since each term is a binary variable itself, there is only one possible split of the data based on this term. Thus, the tree fitting procedure only needs to consider one split per input term, which makes the identification of the best local split particularly fast. For evaluating potential node splits and selecting the split, the conventional node impurity splitting criterion from Eq. (1) is used. For regression tasks, the MSE (mean squared error) impurity (see Eq. (2)) is used, and for classification tasks, the Gini impurity (see Eq. (3)) is used.

After the tree corresponding to the current state has been fitted, its performance on the training data is evaluated by passing all observations through the tree and calculating a *score* that measures the training data error, where the score is chosen so that a smaller value of the score corresponds to a better fit. For regression purposes, the MSE is employed. For risk estimation/classification purposes, probability estimation trees (Provost & Domingos, 2003; Malley et al., 2012) are grown that directly hold class probability estimates in their leaves by using empirical probabilities, i.e., using proportions of class occurrences. Thus, for scoring a state in the risk estimation/classification setting, the deviance is used, which is also known as the cross entropy or the negative binomial log-likelihood.

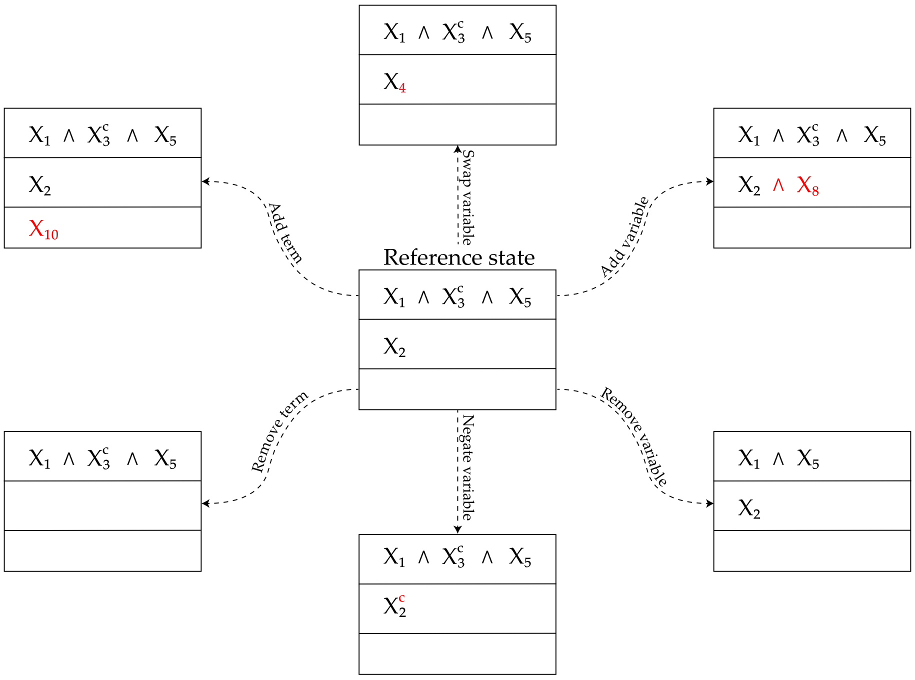
Alternatively, the negative area under the curve with respect to the receiver operating characteristic (AUC) might be used. However, the AUC does not capture the magnitude of the risk estimate in contrast to the deviance. Another alternative is the Brier score, which is the mean squared error between the risk estimate and the actual outcome.

For identifying an ideal state, logicDT performs a global search over all eligible states. The search is performed by using the current state to construct a decision tree, evaluating the performance of this tree, modifying the current state, and repeating this procedure. Modifications of logicDT states are called *neighbors* and are implicitly defined by slightly altering a given state. Figure 3 illustrates the possible state modifications/neighbor states using exemplary states. In the center of this figure, the current state is depicted. The possible state changes include

- exchanging or negating single variables (see, e.g., the replacement of X_2 by X_4 in the top and the negation of X_2 in the bottom of Fig. 3),
- adding or removing single variables from a term (see, e.g., the addition of X_8 in the top right and the removal of X_3^c in the bottom right of Fig. 3),
- adding or removing logic terms consisting of exactly one variable (see, e.g., the addition of X_{10} in the top left and the removal of X_2 in the bottom left of Fig. 3).

To avoid tautologies and uninformative terms, some specific alterations are prohibited. More precisely, the same variable should not occur more than once in a single term and the same term should not occur more than once in the proposed state.

The search is initialized by finding the single input variable that minimizes the score function, e.g., $\{\{X_{73}\}\}$. Using this initial state, a global optimization procedure employing simulated annealing (Kirkpatrick et al., 1983) is carried out for finding the state that minimizes the score function, i.e., now permitting all possible states potentially consisting of more than one term.



$X_1 \wedge X_3^c \wedge X_5$
X_4
Swap variable
$X_1 \wedge X_3^c \wedge X_5$
X_2
X_{10}
Add term
$X_1 \wedge X_3^c \wedge X_5$
$X_2 \wedge X_8$
Add variable
Reference state
$X_1 \wedge X_3^c \wedge X_5$
X_2
Remove term
$X_1 \wedge X_3^c \wedge X_5$
Negate variable
$X_1 \wedge X_3^c \wedge X_5$
X_2^c
Remove variable
$X_1 \wedge X_5$
X_2

Fig. 3 Exemplary state modifications of the reference state $\{\{X_1, X_3^c, X_5\}, \{X_2\}\}$ depicted in the center

Simulated annealing is a stochastic optimization algorithm that, given a current state, randomly selects one of its neighbor states, evaluates its score, and uses the score difference between these two states for determining the probability of transitioning to the proposed neighbor state. For a state s and a proposed neighbor state s', the score function ϵ, and the current *temperature* t, this state acceptance probability is given by

$$\gamma(\epsilon(s), \epsilon(s'), t) \; := \; \min\left\{1, \exp\left(\frac{\epsilon(s) - \epsilon(s')}{t}\right)\right\}. \tag{5}$$

Thus, if a state with a better score is proposed, the transition is carried out with probability 1. However, worse states may also be accepted with the acceptance probability $\in (0, 1)$ to avoid getting stuck in local minima. The main idea of simulated annealing is slowly lowering the temperature t such that the acceptance probability of worse states tends to 0 and in the end, the globally optimal state is identified.

In logicDT, a fully automatic simulated annealing schedule governing the temperature lowering is employed. If desired, the cooling schedule can be changed, e.g., by decreasing or increasing the parameter λ, that controls the magnitude of the temperature decreases, for performing a finer or coarser stochastic search. The number of search iterations is, thus, (implicitly) controlled by λ and stopping criteria for terminating the search procedure. Alternatively, a fixed geometric cooling schedule can also be employed in logicDT. However, we recommend using the adaptive cooling schedule for fitting logicDT models. More details on the simulated-annealing-based search in logicDT are given in Appendix 1.

The proposed state modifications ensure that the modifications lead to a Markov chain that fulfills aperiodicity and irreducibility when performing a global search via simulated annealing. These properties ensure that simulated annealing asymptotically leads with probability 1 to a globally optimal state (Van Laarhoven & Aarts, 1987). More details on these Markov properties are given in Appendix 2.

3.3 The logicDT algorithm

In Algorithm 2, the logicDT procedure is presented.

Algorithm 2: logicDT Fitting

```
1  function logicDT(Training data $\mathcal{D}$):
2      $s$ = Initialize state/set of terms
3      $\mathcal{D}_s$ = Apply $s$ to $\mathcal{D}$
4      $T$ = fitDecisionTree($\mathcal{D}_s$)
5      $\text{Score}_{\text{min}} = \text{Score}(T)$
6      while Global search is not finished do
7          $s'$ = Modify current state $s$
8          $\mathcal{D}_{s'}$ = Apply $s'$ to $\mathcal{D}$
9          $T'$ = fitDecisionTree($\mathcal{D}_{s'}$)
10         $\text{Score}_{\text{new}} = \text{Score}(T')$
11         if State $s'$ is accepted based on $\text{Score}_{\text{min}}$ and $\text{Score}_{\text{new}}$ then
12             $s = s'$
13             $T = T'$
14             $\text{Score}_{\text{min}} = \text{Score}_{\text{new}}$
15         end
16     end
17     return $(s, T)$
18 end
```

In Line 2, the initial state is obtained by choosing the single input variable that minimizes the score. That is, for each input variable, a decision tree using only this input variable, i.e., a decision tree stump, is fitted and evaluated. The input variable X_j that leads to the minimum score is chosen as the initial state $\{\{X_j\}\}$. Alternatively, a random state or an empty state could also be used as the initial state.

In Lines 3 and 8, the current state is used for transforming the original training data set $\mathcal{D}$ into a tree training data set that can be directly used by a learning procedure using the identified terms as input variables. See Eq. (4) for an example on how a tree training data set is obtained from the original data set consisting of the values of the input variables.

If no leaf regression models for continuous covariables shall be fitted, the decision trees are constructed using Algorithm 1 (see Lines 4 and 9 of Algorithm 2). If leaf regression models are to be fitted (see Sect. 3.5 for more details), the splitting criterion from Sect. 3.5.2 is used in place of the impurity reduction criterion and the corresponding regression models are fitted in each leaf in contrast to single prediction values.

In Lines 5 and 10 of Algorithm 2, the training data score is calculated by passing all training observations through the fitted decision tree, performing predictions using leaf regression models if these were fitted, and comparing the predictions with the true outcomes.

In Line 7, the current state is modified by randomly performing one of the state modifications proposed in Sect. 3.2, where the state modification is randomly drawn from a uniform distribution over all possible state modifications of the current state.

This proposed modified state is then evaluated in Line 11, i.e., it is randomly accepted with the acceptance probability from Eq. (5).

The global search is carried out until a stopping criterion is true. More details on the search algorithm itself are discussed in Appendix 1.

logicDT is implemented in the R package `logicDT` (Lau, 2023) available on CRAN.

3.4 Controlling the complexity of logicDT models

For restricting the complexity of logicDT models and regularizing them, the maximum number `max_conj` of terms and the total maximum number `max_vars` of variables contained in a state should in practice be properly tuned to avoid overfitting or underfitting. Since some (potentially very long) conjunctions might correspond to no or very few observations, similar to the stopping criterion in decision trees, a *minimum conjunction size*, defining the minimum number of observations falling into this conjunction and its negation, can be specified in logicDT to exclude practically useless terms. Furthermore, one may prohibit the removal (and the addition) of whole terms in order to guarantee a certain number of terms. This might, e.g., be useful if a pure variable selection should be performed so that the maximum number of total variables is set to the maximum number of terms. In this case, the initial state should be chosen such that it already includes the desired number of terms.

logicDT aims to identify the optimal set of predictors and conjunctions with regard to the predictive ability. Thus, post-pruning of the fitted decision trees is not necessary, since the model complexity is already covered by the model size hyperparameters and the ideal splitting terms are already identified by the global search, which is similar to logic regression and in contrast to standard decision trees. However, the following two stopping criteria for locally terminating the splitting of a branch are used to filter out completely unnecessary splits.

One of the stopping criteria is the minimum number of observations in the respective leaves. If a split would lead to child nodes from which at least one of the children contains less than the prespecified number of observations, this split is prohibited. This criterion is particularly useful for regression and risk estimation purposes, where a stable estimate needs a certain amount of observations.

As second stopping criterion, the minimum (scaled) impurity reduction is considered. A split is discarded, if it does not reach the required impurity reduction, i.e., if

$$\frac{n_t}{n} \cdot \Delta i(s,t) \leq cp,$$

holds for the impurity reduction $\Delta i(s,t)$ defined in Eq. (1) and the complexity parameter $cp \geq 0$. For continuous outcomes, cp will be scaled by the empirical variance s_Y^2 of the outcome Y to ensure the right scaling, i.e., $cp \leftarrow cp \cdot s_Y^2$. Since the impurity measure for continuous outcomes is the mean squared error, this can be interpreted as controlling the minimum reduction of the normalized mean squared error (NRMSE—normalized root mean squared error—to the power of two).

The hyperparameter optimization in logicDT is discussed in more detail in Sect. 3.6.

3.5 Quantitative covariables

Decision trees are particularly suitable models for binary input data, since there is only a finite number of possible predictor scenarios in this case, i.e., every possible prediction function (including the true regression function $\mathbb{E}[Y \mid X]$) can be expressed using a decision tree. Quantitative predictors often induce a continuous relationship to the outcome that cannot be properly expressed with piecewise constant functions such as decision trees or random forests. In standard decision-tree-based methods, continuous variables are included as possible splitting candidates in the decision tree fitting process. This approach is very intuitive for merely considering all available data. However, as mentioned above, this does not allow to cover continuous relationships.

3.5.1 Leaf regression models

For properly including quantitative covariables in logicDT models, we propose, similar to MOB (model-based recursive partitioning, Zeileis et al., 2008), to fit regression models in the leaves that result from splits exclusively using the binary terms. This approach allows to fit individual curves for each binary term setting, thus, also covering interactions between the binary predictors and the quantitative covariable.

In principle, any kind of regression model such as linear or non-linear regression models could be fitted in the leaves depending on the application. Moreover, multiple regression models could also be fitted, if multiple covariables need to be considered.

For properly evaluating logicDT states, regression models need to be fitted in each decision tree and used to generate the training data predictions for computing the score, i.e., the regression models should be fitted in each iteration of the search procedure of logicDT. If, however, the computational burden is too high for, e.g., fitting non-linear regression models in each leaf of each decision tree, we recommend using linear models for the search and non-linear regression models for the final fit. In this case, the functional relationship is still taken into account in the search process and the final model utilizes the desired type of regression model. For a fast model fitting with a binary outcome, logistic regression curves through LDA (linear discriminant analysis) might be fitted that have a closed-form solution (Hastie, Tibshirani, and Friedman, 2009), and therefore, do not require an iterative optimization procedure such as standard logistic regression.

3.5.2 Splitting criterion

If regression models should be fitted in each leaf, functional trends have to be analyzed instead of simple leaf means. Therefore, we propose evaluating splits based on a likelihood-ratio test for comparing nested models as an alternative to the conventional node impurity splitting criterion specified in Eq. (1). More precisely, linear regression or LDA models, which can be determined particularly quickly, are fitted for each eligible split and resulting child node. Since we consider simple regression models, each model consists of two parameters (offset and slope) such that the difference in parameters of two submodels versus one joint model is given by $2 \cdot 2 - 2 = 2$. Thus, the likelihood-ratio test statistic

$$-2\log(\Lambda) := -2\log\left(\frac{L_{\text{reduced}}}{L_{\text{full}}}\right) \tag{6}$$

is—under the null hypothesis of equal model parameters in both subnodes—asymptotically χ^2-distributed with 2 degrees of freedom following Wilks' theorem (Wilks, 1938). Here, L_{reduced} denotes the maximized likelihood of the reduced model (i.e., the fitted joint regression model using one node) and L_{full} denotes the maximized likelihood of the full model (i.e., the model consisting of two individually fitted sub-regression models resulting in two nodes).

With the test statistic from Eq. (6), we, hence, test

$$\begin{aligned} & H_0 : \mathbb{E}[Y \mid \boldsymbol{X}_{(t)} = \boldsymbol{x}_{(t)}, X_s, E] = \mathbb{E}[Y \mid \boldsymbol{X}_{(t)} = \boldsymbol{x}_{(t)}, E] \\ \text{vs.} \quad & H_1 : \mathbb{E}[Y \mid \boldsymbol{X}_{(t)} = \boldsymbol{x}_{(t)}, X_s, E] \neq \mathbb{E}[Y \mid \boldsymbol{X}_{(t)} = \boldsymbol{x}_{(t)}, E], \end{aligned}$$

where t is the node that shall be splitted, $\boldsymbol{X}_{(t)}$ is the subvector of input variables that are used as splitting variables in ancestor nodes of t, $\boldsymbol{x}_{(t)}$ is the corresponding binary vector containing the predictor setting at node t, X_s is the binary predictor that shall be evaluated for splitting the node, and E is (are) the continuous covariable(s). We, thus, test with this likelihood-ratio test whether the split on X_s leads to different prediction models in the current tree branch. E.g., for one continuous covariable, the model

$$g(\mathbb{E}[Y \mid \boldsymbol{X}_{(t)} = \boldsymbol{x}_{(t)}, X_s, E]) = \beta_0 + \beta_1 \cdot E + \gamma_0 \cdot \mathbb{1}(X_s) + \gamma_1 \cdot \mathbb{1}(X_s) \cdot E$$

is used for testing the null hypothesis $H_0 : \gamma_0 = \gamma_1 = 0$, which is equivalent to the above null hypothesis, using the identity as link function g for a continuous outcome and the logit function as link function g for a binary outcome.

Using this new splitting criterion, likelihood-ratio tests for all eligible splits at a certain node are performed to appropriately rank eligible splits and to interpretably quantify the strength of a split. The split that achieves the lowest p-value is used, if this p-value is below a prespecified significance threshold such as $\alpha = 50\%$. Here, we propose to use a very liberal (high) threshold to avoid to miss fruitful splits. If no split can provide such a p-value, the node in question is declared as a terminal node so that this splitting criterion can also act as a stopping criterion.

Figure 4 illustrates an exemplary logicDT model with two terms and three variables in total. The current set of terms on the left induces the decision tree on the right by fitting a decision tree using the terms as potential splitting variables. The quantitative covariable E is used for evaluating the splits in likelihood-ratio tests and for fitting the regression models in the leaves. Therefore, in the root node, the terms SNP3D^c $\wedge$ SNP2D and SNP1D are both evaluated as splitting candidates by fitting regression models using E as the predictor. Since SNP3D^c $\wedge$ SNP2D yields a lower p-value than SNP1D in the likelihood-ratio test splitting criterion, the term SNP3D^c $\wedge$ SNP2D is used for splitting the root node. The fitted tree is then evaluated as a whole using a score function (see Sect. 3.2). Afterwards, the state is slightly modified using the modifications proposed in Sect. 3.2 and the procedure is repeated.

3.6 Hyperparameter optimization

For maximizing the performance of logicDT, it is necessary to optimize the model complexity parameters that act as regularization parameters. These parameters are

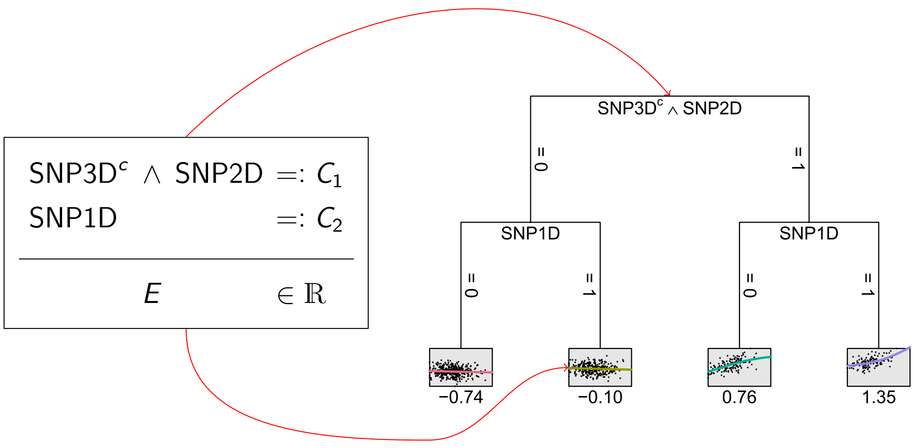


Fig. 4 An exemplary logicDT model/state. On the left hand side, the set of terms is depicted with an additional quantitative covariable which is excluded from the search over the set of terms. On the right hand side, the resulting decision trees which uses the binary predictors and identified conjunctions as input/splitting variables. Since in this case also a quantitative variable is supplied, the leaves are continuous functions instead of single point estimates

- `max_vars`—the total maximum number of variables contained in the model,
- `max_conj`—the maximum number of conjunctions/terms in the model,
- `nodesize`—the minimum number of observations per leaf in the resulting decision tree,
- `conjsize`—the minimum of observations contained in a conjunction and its negation.

In general, `max_vars` $\geq$ `max_conj` has to be fulfilled. Furthermore, we recommend imposing `max_vars` $\leq 2 \cdot$ `max_conj` in cases in which marginal effects still seem to be dominant and it is not justifiable that only high-order interaction terms compose the main influence on the outcome. This restriction is useful due to the standard learning issue that more complex models usually fit the training data better. Moreover, it reduces the set of eligible hyperparameter configurations to be evaluated speeding up the hyperparameter tuning process.

Specifically for fitting single logicDT models (via simulated annealing), it is advisable to remove the ability of removing whole conjunctions from the model in the search procedure. This ensures that the final model consists of exactly `max_conj` terms and that no extensively complex conjunctions make up the model. This also allows for a simple variable selection of marginal effects by additionally restricting `max_vars` = `max_conj`.

The purpose of `nodesize` is to ensure that each leaf contains enough observations for concluding meaningful models, i.e., stable means, or if a continuous covariable is included, regression models. A proper value for `conjsize` avoids evaluating models with uninformative conjunctions, i.e., conjunctions for which a split does not imply meaningful information due to a low number of observations. Note that for the observed values, it holds $\texttt{nodesize}_{\text{obs}} \leq \texttt{conjsize}_{\text{obs}}$, since the decision tree can further split the space. Thus, in practice, `nodesize` and `conjsize` can be set to the same value. Similar to Malley et al. (2012) who regarded probability estimation trees, we recommend a value between 1% and 10% of the total number of training observations for obtaining stable leaf estimates.

Using these parameter restrictions, a grid search evaluating all possible parameter combinations is then carried out (based on validation data) in order to identify the best setting. In Sect. 5, hyperparameter optimization following this scheme is performed.

3.7 Consistency of logicDT

In this section, we now study theoretical properties of logicDT, more precisely, the consistency of logicDT. For this purpose, we consider the core logicDT methodology, i.e., only permitting binary predictors. Without loss of generality, we assume a continuous outcome. Binary risk estimation/binary classification can be viewed as a special case using the Brier score as score function in an empirical risk minimization framework. The following theorem states that logicDT is strongly consistent. The proof of this theorem is given in Appendix 2.

Theorem 1 (Consistency of logicDT) *Suppose $\mu : \{0,1\}^p \to \mathcal{Y}$ is a p-dimensional regression function and that the outcome Y with*

$$\mathbb{E}[Y \mid \boldsymbol{X}] = \mu(\boldsymbol{X})$$

is bounded. Then, logicDT fitted via simulated annealing is strongly consistent, i.e., almost sure convergence

$$\mathbb{E}_{(\boldsymbol{X},Y)}\left[(\mu(\boldsymbol{X}) - T_n(\boldsymbol{X}))^2\right] \xrightarrow[n\to\infty]{\text{a.s.}} 0$$

holds for fitted logicDT models T_n to training data sets $\mathcal{D}_n = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\}$.

The following remark provides an application of Theorem 1 to hard classifications, in which the misclassification rate is evaluated.

Remark 1 For the binary classification/risk estimation case, alternatively to considering the Brier score, the excess misclassification rate is bounded by

$$\begin{aligned} 0 \;&\leq\; \mathbb{P}_{(\boldsymbol{X},Y)}(\hat{\varphi}_{T_n}(\boldsymbol{X}) \neq Y) \;-\; \mathbb{P}_{(\boldsymbol{X},Y)}(\varphi^*(\boldsymbol{X}) \neq Y) \\ &\leq\; 2\sqrt{\mathbb{E}_{(\boldsymbol{X},Y)}\left[(\mu(\boldsymbol{X}) - T_n(\boldsymbol{X}))^2\right]} \;\xrightarrow[n\to\infty]{\text{a.s.}}\; 0 \end{aligned}$$

for the classifiers $\hat{\varphi}_{T_n}(\boldsymbol{x}) = \mathbb{1}(T_n(\boldsymbol{x}) \geq 0.5)$ and the Bayes classifier φ^* (see, e.g., Theorem 1.1, Györfi et al., 2002).

Thus, the misclassification rate of the best possible classifier φ^* will be asymptotically almost surely attained by logicDT.

Note that Theorem 1 holds as long as the proposed hyperparameters are properly chosen so that the true underlying model satisfies the chosen hyperparameters. More precisely, `max_vars` and `max_conj` need to be sufficiently big and `nodesize` and `conjsize` need to be sufficiently small.

3.8 Computational complexity of logicDT

In this section, we study the computational complexity of logicDT, which is mainly controlled by the complexities of conducting a simulated-annealing-based search and fitting decision trees. A guarantee for obtaining a globally optimal model is only given if infinite iterations (or iterations in the magnitude of the size of the complete search space) are carried out in the simulated-annealing-based search (Van Laarhoven & Aarts, 1987). In practice, this is because of the size of the search space, typically, infeasible. Therefore, this asymptotic search is in practical applications approximated using a finite number of iterations (for more details on the search process, see Appendix 1). Therefore, we assume that the number of search steps is given by a finite number M.

Using the complexities of simulated annealing, decision tree fitting, and tree training data set transformation and using Algorithm 2, the computational complexity of logicDT is given in the following theorem. The proof of this theorem is given in Appendix 3.

Theorem 2 (Computational complexity of logicDT) *Suppose M is the number of search steps performed, n training observations are given, and the hyperparameters* `max_vars`, `max_conj`, `nodesize` *are fixed. Then, the computational complexity of logicDT is given by*

$$\mathcal{O}\left(Mn\left[\texttt{max\_vars} + \texttt{max\_conj}\frac{n}{\texttt{nodesize}}\right]\right).$$

Using Theorem 2, results about appropriate numbers M of search iterations based on the Markov chain length (i.e., the number of search iterations for a fixed temperature), and assumptions on the hyperparameter choices, the following corollary states that the computational complexity of logicDT is polynomial in p. The corresponding proof is, again, provided in Appendix 3.

Corollary 1 (Polynomial complexity of logicDT) *Assume that the parameters* `max_vars` *and* `max_conj` *both scale linearly with p and that the parameter* `nodesize` *is constant (with respect to n which is the worst-case scenario in which the logic decision tree may be arbitrarily deep). Further assume that the Markov chain length is fixed. Then, the computational complexity of logicDT is given by*

$$\mathcal{O}\left(n^2p^2\log(p)\right).$$

If instead the Markov chain length is chosen in the magnitude of the number of neighbor states per state (as suggested by Aarts & Van Laarhoven, 1985), the computational complexity of logicDT is given by

$$\mathcal{O}\left(n^2p^4\log(p)\right).$$

3.9 Bagged logicDT

If a single model consisting of relatively few variables cannot explain the whole variation in the outcome from the whole set of predictors or if the predictive power is of higher interest than the interpretability of the model, ensemble models consisting of several simpler models might be a preferable choice.

A particularly simple, yet effective approach is bagging (Breiman, 1996), in which for a given number of bagging iterations (e.g., 500), a single model is fitted on a random subset of the original training data set. The random subsets are typically generated via bootstrapping, i.e., performing random draws from the original training data with replacement n times. The resulting model is the ensemble of all models. Predictions are performed by averaging the predictions of the individual models. The number of iterations should, as in random forests, be chosen such that more iterations cannot reduce the generalization error substantially anymore.

Since sufficient bagging iterations are also desired in logicDT, simulated annealing with a proper amount of iterations itself might just be too slow. Moreover, the main issue of greedy search approaches, i.e., that a globally optimal state could be missed due to being stuck in a local optimum, might be diminished through considering different subsets of the training data set and stabilizing the model over them. In other words, the variance stabilizing property of bagging might be sufficient to account for the drawbacks of a greedy search (Murthy & Salzberg, 1995).

For the usage of logicDT in an ensemble framework, we, therefore, propose a greedy search for fitting individual logicDT models. In this greedy search, the same state modifications as in the simulated-annealing-based search are used (see Sect. 3.2). In contrast to simulated annealing, the greedy search deterministically chooses the best neighbor in each iteration. Thus, for each current state, all its neighbors are evaluated and the neighbor with the lowest score amongst all neighbors is chosen as new state. Note that for increasing numbers of predictors and increasing numbers of allowed terms and total variables, the number of eligible neighbors per state increases quadratically, thus, slowing down the greedy search. For handling higher-dimensional data, a randomization of the greedy search might be a solution which we, however, did not consider in this article.

Another very useful property of bagging is that in the fitting of an individual model not all observations from the training data are employed. The not considered observations called oob (out-of-bag) observations can, therefore, be used to estimate the generalization error, similar to using independent test data. This estimate is called the oob error and is obtained by only using models that were not built using the considered observation. More precisely, the oob error is calculated by averaging over the oob errors of the observations, where the oob error of an observation can be computed by only choosing the models which did not use this observation for training and by temporarily constructing an ensemble from this subset of models for predicting the outcome of this observation. In particular, for the estimation of variable importance measures (VIMs), bagging and oob observations are very beneficial. As discussed in the following section, we, therefore, also use them in the construction of the VIM considered in logicDT.

4 Variable importance measures

In many applications, it is useful to measure the influence of the input variables or their interactions on the prediction of an outcome. Variable importance measures (VIMs) directly try to quantify this influence. Typically, this influence is estimated by comparing two models, namely

- the original full model containing the term of interest and

- a kind of informatively reduced model, in which the term of interest no longer plays an informative role.

Then, the difference between the prediction errors of these two models is computed and is taken as an estimate of how the prediction based on the model improves if the term is properly included, where the prediction errors are, e.g., given by the mean squared error in regression tasks or $1 - \text{AUC}$ in binary risk estimation tasks.

4.1 Computation of VIMs

Let $\epsilon(\tilde{\boldsymbol{X}})$ be a prediction error measure capturing the performance of a fitted model informatively using only the input variables in $\tilde{\boldsymbol{X}} \subseteq \boldsymbol{X}$, interpreting the random vector of input variables $\boldsymbol{X} = (X_1, \dots, X_p)$ as a set $\boldsymbol{X} = \{X_1, \dots, X_p\}$. Then, the importance of an input variable X_i is given by

$$\text{VIM}(X_i) = \epsilon(\boldsymbol{X} \setminus X_i) - \epsilon(\boldsymbol{X}). \tag{7}$$

Here, $\epsilon(\boldsymbol{X} \setminus X_i)$ describes the prediction error of the reduced model informatively excluding the variable X_i and $\epsilon(\boldsymbol{X})$ describes the prediction error of the original full model.

Bagging allows the unbiased estimation of VIMs on the full training data set by performing oob predictions. Moreover, bagging also has the advantage that multiple potentially different models are explored stabilizing the VIMs themselves. Thus, for estimating VIMs in logicDT, bagging is used and the discussed VIMs are computed on the oob observations.

4.1.1 Permutation VIM and removal VIM

One particularly popular approach for estimating the reduced model is the *permutation VIM* used in random forests (Breiman, 2001). In this approach, for estimating the importance of a certain input variable, its corresponding observations are randomly permuted and predictions with this random permutation are performed. Typically, the VIM data set is permuted multiple times in the specific predictor and the average prediction error of these permutations is compared against the original error.

As an alternative, the reduced model can also be directly fitted using a reduced training data set from which the predictor of interest was removed (Mentch & Hooker, 2016). In the following, we call this approach the *removal VIM*.

4.1.2 Logic VIM

For binary predictors, we additionally propose a specific third procedure for computing VIMs. The idea of this *logic VIM* is based on considering each possible predictor setting of the input variable of interest equally, i.e., for a binary predictor $X_1 \in \{0, 1\}$, the error of the reduced model is estimated by performing predictions fixing $X_1 = 0$, performing predictions fixing $X_1 = 1$ and averaging these predictions before computing the error. Thus, for each observation, the prediction of the reduced model considers both possible decision tree paths, one for $X_1 = 0$ and one for $X_1 = 1$, equally and is generated without knowledge about X_1.

4.2 Adjustment for interactions

In standard VIM procedures such as the permutation VIM in random forests, only importances of single input variables are considered. In the context of logicDT, we measure the importance of terms, i.e., of identified single input variables or conjunctions of several input variables. For instance, if the resulting model consists of $\{\{X_1\}, \{X_2 \land X_3^c\}\}$, we are interested in specifying the importance of X_1 as well as the importance of the term $X_2 \land X_3^c$. This is achieved by considering terms such as $X_2 \land X_3^c$ as single input variables, i.e., by directly considering a tree training data set as in Eq. (4).

Since decision trees can handle interactions themselves, it might be possible that, e.g., X_1 as well as the interaction $X_1 \land X_2^c$ exhibit strong effects on the outcome. However, due to the strong marginal effect, only the single predictors X_1 and X_2 might be included in the logicDT model, complicating the estimation of the importance of the interaction.

Hence, we propose a novel VIM adjustment procedure for interactions that quantifies the importance of interactions that were not identified by a supervised learner such as logicDT. This VIM adjustment approach presented in the following does not depend on logicDT, but enables logicDT to appropriately estimate interaction importances. Therefore, they could, in principle, be applied to all black-box models for estimating interaction importances.

The idea behind the VIM adjustment procedure is based on considering several predictors at once, i.e., the reduced model results from reducing multiple variables in one step. Comparing the performances of this reduced model and the original model yields a joint VIM of the set of predictors (Bureau et al., 2005). Analogously to Eq. (7), the joint VIM is obtained by

$$\mathrm{VIM}(X_{i_1}, \dots, X_{i_k}) = \epsilon(\boldsymbol{X} \setminus \{X_{i_1}, \dots, X_{i_k}\}) - \epsilon(\boldsymbol{X}). \tag{8}$$

Since this joint VIM still includes the marginal effects of the individual predictors and their sub-interactions of an order lower than the order of the actual interaction influencing the outcome, we propose the *interaction VIM* that corrects for any effects contained in the regarded interaction. This interaction VIM of $X_{i_1} \land \cdots \land X_{i_k}$ is given by

$$\begin{aligned}\mathrm{VIM}(X_{i_1} \land \cdots \land X_{i_k}) &= \mathrm{VIM}(X_{i_1}, \dots, X_{i_k} \mid \boldsymbol{X} \setminus \boldsymbol{Z}) \\ &\quad - \sum_{\{j_1,\dots,j_l\} \subsetneq \{i_1,\dots,i_k\}} \mathrm{VIM}(X_{j_1} \land \cdots \land X_{j_l} \mid \boldsymbol{X} \setminus \boldsymbol{Z}),\end{aligned} \tag{9}$$

where $\boldsymbol{Z} := \{X_{i_1}, \dots, X_{i_k}\}$ is the set of input variables in the considered interaction. In our notation, $\land$ denotes the interaction importance, while commas represent the joint importance. By $\mathrm{VIM}(\boldsymbol{A} \mid \boldsymbol{X} \setminus \boldsymbol{Z})$, the VIM of $\boldsymbol{A}$ considering the predictor set excluding the variables in $\boldsymbol{Z}$, i.e., the improvement of additionally considering $\boldsymbol{A}$, while regarding only the predictors in $\boldsymbol{X} \setminus \boldsymbol{Z}$, is denoted. The interaction importance captures the importance of a general meaning of interaction, i.e., it considers whether some variables do interact in any way and quantifies the effect of the joint presence of these variables adjusted for single occurrences. For a predictor set $\tilde{\boldsymbol{A}} := \{X_{j_1}, \dots, X_{j_l}\} \subseteq \boldsymbol{Z}$, the restricted joint VIM, i.e., the VIM of $\tilde{\boldsymbol{A}}$ considering only the predictors $\boldsymbol{X} \setminus \boldsymbol{Z}$ in the reduced model, is, following Eq. (8), given by

$$\mathrm{VIM}(\tilde{\boldsymbol{A}} \mid \boldsymbol{X} \setminus \boldsymbol{Z}) = \epsilon(\boldsymbol{X} \setminus \boldsymbol{Z}) - \epsilon(\tilde{\boldsymbol{A}} \cup (\boldsymbol{X} \setminus \boldsymbol{Z})). \tag{10}$$

Excluding all variables in $\boldsymbol{Z}$ composing the interaction in the respective reference models is crucial for isolating the effects that should be adjusted for. If, e.g., a two-way interaction $X_1 \land X_2$ is studied, its interaction VIM (9) is given by

$$\begin{aligned}\mathrm{VIM}(X_1 \wedge X_2) &= \mathrm{VIM}(X_1, X_2 \mid \boldsymbol{X} \setminus \{X_1, X_2\}) \\ &\quad - \mathrm{VIM}(X_1 \mid \boldsymbol{X} \setminus \{X_1, X_2\}) - \mathrm{VIM}(X_2 \mid \boldsymbol{X} \setminus \{X_1, X_2\}).\end{aligned} \tag{11}$$

If, e.g., $\mathrm{VIM}(X_1 \mid \boldsymbol{X} \setminus X_1) \overset{(7),(10)}{=} \mathrm{VIM}(X_1)$ would be used instead of $\mathrm{VIM}(X_1 \mid \boldsymbol{X} \setminus \{X_1, X_2\})$ in Eq. (11), the whole importance of X_1, that also contains the interaction with X_2, would be subtracted from the joint importance not isolating the interaction importance that should be estimated.

Recursively applying Eq. (11) to the general case in Eq. (9) yields

$$\mathrm{VIM}(X_{i_1} \wedge \cdots \wedge X_{i_k}) = \sum_{\{j_1,\dots,j_l\} \subseteq \{i_1,\dots,i_k\}} (-1)^{k-l} \cdot \mathrm{VIM}(X_{j_1}, \dots, X_{j_l} \mid \boldsymbol{X} \setminus \boldsymbol{Z}).$$

Utilizing Eq. (10), this formula for the interaction VIM can also be written in terms of prediction errors ϵ, i.e., as

$$\mathrm{VIM}(X_{i_1} \wedge \cdots \wedge X_{i_k}) = \sum_{\{j_1,\dots,j_l\} \subseteq \{i_1,\dots,i_k\}} (-1)^{l+1} \cdot \epsilon(\boldsymbol{X} \setminus \{X_{j_1}, \dots, X_{j_l}\}) - \epsilon(\boldsymbol{X}).$$

This formula can be used for efficiently computing the interaction VIM by directly considering prediction errors.

The interaction VIM (9) is similar to the interaction effect statistic proposed by Friedman and Popescu (2008), which utilizes the same effect decomposition and is based on the explained variance of partial dependence functions instead of VIMs. Friedman and Popescu (2008) theoretically justified this effect decomposition by showing that their statistic is zero, if the null hypothesis of no present interaction effect holds true. For example, for analyzing a two-way interaction $X_1 \wedge X_2$, Friedman and Popescu (2008) evaluate $F_{X_1,X_2} - F_{X_1} - F_{X_2}$, in which $F_.$ denotes partial dependence functions of the considered input variables. This term is analogous to the interaction VIM in Eq. (11) for $X_1 \wedge X_2$ with the difference that VIMs, i.e., performance metrics, are used instead of partial dependence functions. Moreover, the input feature effect decomposition utilized by the proposed interaction VIM is also used by the Shapley interaction index (Lundberg et al., 2020; Fujimoto et al., 2006). However, in machine learning applications, Shapley values are based on direct predictions of the fitted model instead of performance metrics such as VIMs.

For all three procedures for constructing VIMs mentioned in Sect. 4.1, the reduced joint model can be intuitively constructed.

In the permutation VIM, the input variables of interest, i.e., the input variables participating in the interaction for which the interaction VIM should be computed, are simply permuted together by, e.g., permuting the values of each input variable separately.

For the removal VIM, the set of input variables of interest is removed as a whole from the total set of input variables.

The logic VIM proposed in Sect. 4.1.2 performs uninformative predictions of an input variable by considering both possible decision tree paths for an observation and averaging the prediction. To generalize the logic VIM to multiple input variables at once for computing the interaction VIM, all possible predictor settings $\boldsymbol{x} \in \{0,1\}^p$ for the p input variables that shall be informatively excluded are used to generate predictions. These 2^p predictions are averaged to create the prediction of the reduced model.

In logicDT, the logic VIM is used in conjunction with the proposed adjustment for interaction effects. Quantifying the importance of specific conjunctions, that are, e.g., identified by logicDT, will be discussed in the following section. In Sect. 5, the permutation VIM, the removal VIM, and the logic VIM are evaluated in empirical studies.

4.3 Adjustment for conjunctions

The VIM adjustment approach introduced in Sect. 4.2 only captures the importance of a general meaning of interactions, i.e., it just considers the question whether some variables do interact in some way. Since logicDT is aimed at identifying specific conjunctions (and also determines the values of a VIM for them, if the conjunctions have been identified by logicDT), a further adjustment approach is proposed that tries to identify the specific conjunction leading to an interaction effect. For example, if the importance of the interaction between X_1 and X_2 was quantified using the interaction adjustment proposed in Sect. 4.2, the approach presented in the following assigns a Boolean conjunction to this importance, e.g., the Boolean conjunction $X_1 \land X_2^c$. The proposed procedure is, again, applicable to any kind of supervised learning model. However, due to considering Boolean conjunctions, the input variables for which the importance should be quantified need to be binary.

This approach considers each possible conjunction of the identified interaction and chooses the conjunction that leads to the most severe deviation in the outcome, i.e., the conjunction with the strongest effect on the outcome. The VIM of this conjunction is the corresponding interaction VIM derived in Sect. 4.2.

The idea of this method is to consider the values of the outcome for each possible scenario of the interacting variables, e.g., for $X_1 \land (X_2^c \land X_3)$, where we assume that the terms X_1 and $X_2^c \land X_3$ were identified by logicDT. In this example, thus, two interacting terms are regarded, i.e., the $2^2 = 4$ possible scenarios $X_1 = 0$ or $X_1 = 1$ in combination with $X_2^c \land X_3 = 0$ or $X_2^c \land X_3 = 1$ are considered. For each setting, the corresponding outcome values are compared to the outcome values of the complementary set, i.e., the set in which the considered conjunction is equal to zero. This means that in the considered example the four statistical tests

$$\begin{aligned} & H_0 : \mathbb{E}\left[Y \mid C_i = 1\right] = \mathbb{E}\left[Y \mid C_i = 0\right] \\ \text{vs.} \quad & H_1 : \mathbb{E}\left[Y \mid C_i = 1\right] \neq \mathbb{E}\left[Y \mid C_i = 0\right], \end{aligned}$$

with

$$\begin{aligned} C_1 &= X_1 \land (X_2^c \land X_3), \quad & C_2 &= X_1^c \land (X_2^c \land X_3), \\ C_3 &= X_1 \land (X_2^c \land X_3)^c, \quad & C_4 &= X_1^c \land (X_2^c \land X_3)^c \end{aligned}$$

potentially negating the subterms, are performed for $i \in \{1, 2, 3, 4\}$. For continuous outcomes, Welch's t-test is performed for comparing the means between these two groups, i.e., the group in which the considered conjunction is equal to one and the group in which the considered conjunction is equal to zero. For binary outcomes, Fisher's exact test is performed for testing different underlying case probabilities. The combination with the lowest p-value is chosen as the explanatory term for the interaction effect. E.g., in the above example, if the smallest p-value results from considering $X_1 = 0$ and $(X_2^c \land X_3) = 1$, the term $X_1^c \land (X_2^c \land X_3)$ is chosen as the conjunction responsible for the interaction effect.

5 Experiments

In the following, we evaluate the performance of logicDT on simulated and real data considering classification and regression problems and compare logicDT with other similar methods. More precisely, we compare logicDT and bagged logicDT with conventional decision trees (Breiman et al., 1984), DL8.5 (Aglin et al., 2020a), random forests (Breiman, 2001), gradient boosting (Friedman, 2001), logic regression (Ruczinski et al., 2003), logic regression with bagging (Schwender & Ickstadt, 2007), MOB (model-based recursive partitioning, Zeileis et al., 2008), interaction forests (Hornung & Boulesteix, 2022), and RuleFit (Friedman & Popescu, 2008). Since DL8.5 (as similar openly available optimal decision tree algorithms such as MurTree proposed by Demirović et al., 2022) are currently only implemented for classification tasks, DL8.5 is only applied to the considered classification tasks. All analyses are carried out using `R` (R Core Team, 2022), except for the application of DL8.5, which is performed using the `Python` implementation of Aglin et al. (2020b).

5.1 Simulation study

We, first, consider the situation of genetic association studies in which single genes/genetic pathways are analyzed and typically not more than a few tens of SNPs (single nucleotide polymorphisms) are considered. Afterwards, we consider a more complex setting with more SNPs to evaluate if logicDT is also applicable to high-dimensional problems.

5.1.1 First simulation setup

We analyze the performance of logicDT and the other supervised learning procedures first in four different simulation scenarios in which we consider binary predictors and

- a binary outcome (such as a disease status) without an additional continuous covariable,
- a binary outcome with a continuous covariable,
- a continuous outcome (such as the blood pressure) without a continuous covariable, and
- a continuous outcome with a continuous covariable.

Our simulations are based on the problem of analyzing risk factors in genetic epidemiology. Thus, the generated input variables can be interpreted as SNPs that count the number of minor alleles at a specific locus, i.e., the number of occurrences of a less frequent base-pair substitution at a specific location in the DNA. Due to humans being diploid organisms, i.e., carrying two complete sets of chromosomes, SNPs can take the values 0, 1, or 2. Similar to, e.g., logic regression, for the application of logicDT to SNP data, each SNP is divided into the binary input variables $\text{SNP}_D = \mathbb{1}(\text{SNP} \neq 0)$ and $\text{SNP}_R = \mathbb{1}(\text{SNP} = 2)$, coding for a dominant and a recessive effect, respectively, such that no information is lost. Conventional decision trees also implicitly divide SNPs into dominant and recessive effects by considering SNPs as numerical variables such that a split can occur on $(\{0\}, \{1, 2\})$ or on $(\{0, 1\}, \{2\})$. Combined with the greedy search of decision trees over all possible splits, this is equivalent to directly considering the binary variables SNP_D and SNP_R (Lau et al., 2022).

The genotypes of the SNPs are generated independently, resembling sets of SNPs from which, as often done in practice, highly correlated SNPs have been removed using linkage-disequilibrium-based pruning (see, e.g., Purcell et al., 2007). The distributions of the SNPs are defined via the MAF (minor allele frequency), i.e., the proportion of minor allele occurrences, yielding the binomial distribution $\text{Bin}(2, \text{MAF})$ for each SNP. For all simulated SNPs, we consider a MAF of 0.25. For each data set, 50 SNPs are generated so that $X = (\mathrm{SNP}_1, \dots, \mathrm{SNP}_{50})$. However, in the considered scenarios described below, only a small fraction influences the outcome such that most input variables act as noise regarding the outcome.

For the analysis of the influence of a continuous covariable, an environmental variable (e.g., an air pollution indicator) is generated following a truncated normal distribution (truncated at zero, since values below zero often do not occur in practice). In particular, the environmental term E is generated by considering a $\mathcal{N}(20, 100)$-distributed random variable E' and setting values below zero to zero so that $E = \max(0, E')$. The truncated values might, e.g., be interpreted as measurements below a detection limit.

Since DL8.5 can only incorporate binary input variables, E is dichotomized into a binary variable by considering $E_{\mathrm{bin}} = \mathbb{1}(E > 20)$ for fitting and evaluating DL8.5 models, where the cutoff 20 is chosen due to $\mathbb{E}[E] = \mathbb{P}(E' > 0)\mathbb{E}[E' \mid E' > 0] \approx 20$.

For the first simulation scenario considering a binary outcome without any continuous covariables, the outcome is generated following the model

$$\begin{aligned}\mathrm{logit}(\mathbb{P}(Y = 1 \mid X)) \; = \; -0.4 + \Big(&\sqrt{\log(1.5)} \cdot \mathbb{1}(\mathrm{SNP}_1 > 0) \\ &+ \sqrt{\log(2)} \cdot \mathbb{1}(\mathrm{SNP}_2 > 0 \land \mathrm{SNP}_3 = 0)\Big)^2.\end{aligned}$$

Therefore, SNP_1 exhibits a moderate marginal effect and SNP_2 and SNP_3 interact with each other. The linear predictor on the right-hand side is squared which means that, on the scale of the total linear predictor, the term $\mathbb{1}(\mathrm{SNP}_1 > 0)$ interacts with the term $\mathbb{1}(\mathrm{SNP}_2 > 0 \land \mathrm{SNP}_3 = 0)$. Thus, this resembles a situation in which it might be useful to be able to model interactions between interactions, since the underlying scale of the linear predictor is unknown prior to the analyses, which is usually the case in practice. The intercept of −0.4 ensures that the resulting data sets are approximately balanced, i.e., that the fraction of cases is approximately equal to 50%.

In the second scenario, a gene-environment interaction is introduced such that the outcome in this case is modeled by

$$\begin{aligned}\mathrm{logit}(\mathbb{P}(Y = 1 \mid X, E)) \; = \; &-0.45 + \log(2) \cdot \mathbb{1}(\mathrm{SNP}_1 > 0) \\ &+ \log(3) \cdot \frac{E}{20} \cdot \mathbb{1}(\mathrm{SNP}_2 > 0 \land \mathrm{SNP}_3 = 0).\end{aligned}$$

Thus, the environmental variable only influences the outcome, if the term $\mathbb{1}(\mathrm{SNP}_2 > 0 \land \mathrm{SNP}_3 = 0)$ holds true. This kind of gene-environment interaction might be reasonable for substances that are usually harmless, but might cause, e.g., allergic reactions in individuals with a certain genetic makeup.

Analogously to the first scenario, the third scenario consists of data sets in which the outcome is modeled by

$$\begin{aligned}\mathbb{E}[Y \mid X] \; = \; -0.4 + \Big(&\sqrt{\log(1.5)} \cdot \mathbb{1}(\mathrm{SNP}_1 > 0) \\ &+ \sqrt{\log(2)} \cdot \mathbb{1}(\mathrm{SNP}_2 > 0 \land \mathrm{SNP}_3 = 0)\Big)^2.\end{aligned}$$

Here and in the following scenario, random noise generated from $\mathcal{N}(0, 1)$ is added to the linear predictor.

As in the second scenario, the fourth scenario follows the underlying model

$$\begin{aligned}\mathbb{E}[Y \mid X, E] = &-0.75 + \log(2) \cdot \mathbb{1}(\mathrm{SNP}_1 > 0) \\ &+ \log(4) \cdot \frac{E}{20} \cdot \mathbb{1}(\mathrm{SNP}_2 > 0 \land \mathrm{SNP}_3 = 0).\end{aligned}$$

For each simulation scenario, 100 independent data sets are generated. For each data set, it is assumed that this is the only data set available. Thus, for each replication, the data set is randomly divided into a training, a validation, and a test data set. Thus, for the evaluation of logicDT and comparable methods, we perform 100 independent evaluations. In many practical applications such as in the construction of genetic risk scores, there is only data for a relatively small number of observations available. Hence, in our simulations, the randomly generated data sets consist of 1000 observations each. From each of these data sets, $0.7 \cdot 1000 = 700$ randomly chosen observations are used as the intermediary data set for training and validating the model and the remaining 300 observations yield the test data set for the final evaluation. The intermediary data set is further randomly divided into $0.25 \cdot 700 = 175$ observations for choosing the best set of hyperparameters and $0.75 \cdot 700 = 525$ observations for training in the hyperparameter optimization. After the optimal hyperparameter setting has been identified, the final models are trained on the intermediary data set consisting of 700 observations.

The predictive performance of logicDT and the comparable methods is assessed using the AUC for binary outcomes and using the complement of the NRMSE (normalized root mean squared error) for continuous outcomes on test data predictions.

5.1.2 Hyperparameter optimization

As described in Sect. 3.6, the model complexity parameters `max_vars` (maximum number of total variables) and `max_conj` (maximum number of conjunctions) of logicDT should be tuned. In this application, we prohibit removing complete conjunctions to ensure that the models consist of exactly `max_conj` conjunctions. Furthermore, the minimum number `nodesize` of observations belonging to a leaf and the minimum number `conjsize` of observations belonging to a conjunction and its negation are tuned using the same value, respectively. This ensures that the trees are grown to the ideal depth and prevents that models using uninformative conjunctions are evaluated.

For bagged logicDT models, `max_vars` and `max_conj` are tuned using the same parameter setting and allowing the removal of complete conjunctions in contrast to fitting single logicDT models.

In Table 1, the considered hyperparameter settings for logicDT, bagged logicDT, and the comparable tree-based statistical learning methods are summarized. For logicDT, the hyperparameter settings proposed in Sect. 3.6 are considered. For the regarded comparable methods, common hyperparameter choices are considered and the best performing one is chosen. For all methods except for gradient boosting and RuleFit, a grid search among all proposed settings is performed, due to relatively few plausible settings. For gradient boosting and RuleFit, a sequential Bayesian hyperparameter search is carried out (Bergstra et al., 2011; Wilson, 2021), since a finetuning of the learning rate parameter (for a fixed number of boosting iterations) is required. Additionally, the subsample fraction and the minimum node size, which can also be considered as continuous hyperparameters, have to be configured jointly in gradient boosting and RuleFit. For this sequential search, 100 different settings are evaluated.

Table 1 Regarded hyperparameter settings with according descriptions

Algorithm	Software package	Hyperparameter	Description	Considered realizations
logicDT	`logicDT` (Lau, 2023)	(`max_vars`, `max_conj`)	Maximum number of variables and maximum number of conjunctions	$\{(i,j) \in \{1,\dots,10\}^2 \mid i \geq j \wedge i \leq 2j\}$
		`nodesize`/`conjsize`	Minimum number of observations per leaf and minimum number of observations per conjunction	$\lfloor \{0.01, 0.05, 0.1\} \cdot N \rfloor$
logicDT–Bagging	`logicDT` (Lau, 2023)	(`max_vars`, `max_conj`)	Maximum number of variables and maximum number of conjunctions	$\{(1,1),\dots,(10,10)\}$
		`nodesize`/`conjsize`	Minimum number of observations per leaf and minimum number of observations per conjunction	$\lfloor \{0.01, 0.05, 0.1\} \cdot N \rfloor$
		`bagging.iter`	Number of bagging iterations	500
Decision Tree	`rpart` (Therneau & Atkinson, 2019)	`cp`	Complexity parameter for optimal pruning	$\{0, 10^{-3}, 10^{-2.5}, 10^{-2}, 10^{-1.5}, 10^{-1}, 10^{-0.5}, 10^0\}$
		`minbucket`	Minimum number of observations per leaf	$\lfloor \{0.01, 0.05, 0.1\} \cdot N \rfloor$
DL8.5	`PyDL8.5` (Aglin et al., 2020b)	`min_sup`	Minimum number of observations per leaf	$\lfloor \{0.01, 0.05, 0.1, 0.2\} \cdot N \rfloor$
Random Forests	`ranger` (Wright & Ziegler, 2017)	`mtry`	Number of randomly drawn input variables at each split	$\left\lfloor \{0.5, 1, 2\} \cdot \lfloor \sqrt{p} \rfloor \right\rfloor \cup \left\lfloor \{0.5, 1, 2\} \cdot \lfloor \frac{p}{3} \rfloor \right\rfloor$
		`min.node.size`	Minimum number of observations per leaf	$\lfloor \{0.01, 0.05, 0.1\} \cdot N \rfloor$
		`num.trees`	Number of trees grown	2000
Gradient Boosting	`xgboost` (Chen & Guestrin, 2016)	`subsample`	Subsample fraction for drawing samples without replacement for each tree	$[0.5, 1.0]$
		`min_child_weight`	Minimum number of observations per leaf	$\{m \in \mathbb{N} \mid m \in [0.01 \cdot N, 0.1 \cdot N]\}$
		`eta`	Learning rate	$[10^{-6}, 10^{-1}]$
		`nrounds`	Number of boosting iterations	500

Table 1 (continued)

Algorithm	Software package	Hyperparameter	Description	Considered realizations
Logic Regression	`LogicReg` (Kooperberg & Ruczinski, 2022)	(`nleaves`, `ntrees`)	Maximum number of (total) leaves and maximum number of trees	$\{(i,j) \in \{1,\dots,20\} \times \{1,\dots,5\} \mid i \geq j\}$
		`anneal.control`	Simulated annealing cooling schedule	Experimental
Logic Bagging	`logicFS` (Schwender & Ickstadt, 2007)	(`nleaves`, `ntrees`)	Maximum number of (total) leaves and maximum number of trees	$\{(i,j) \in \{1,\dots,10\} \times \{1,\dots,5\} \mid i \geq j\}$
		`B`	Bagging iterations	500
MOB	`party` (Zeileis et al., 2008)	`minsplit`	Minimum number of observations per leaf	$\lfloor \{0.01, 0.02, \dots, 0.09, 0.1\} \cdot N \rfloor$
Interaction Forests	`diversityForest` (Hornung, 2022)	`npairs`	Number of randomly drawn input variable pairs at each split	$\left\lfloor \{0.5, 1, 2\} \cdot \lfloor \sqrt{p} \rfloor \right\rfloor \cup \left\lfloor \{0.5, 1, 2\} \cdot \lfloor \frac{p}{3} \rfloor \right\rfloor$
		`min.node.size`	Minimum number of observations per leaf	$\lfloor \{0.01, 0.05, 0.1\} \cdot N \rfloor$
		`num.trees`	Number of trees grown	2000
RuleFit	`pre` (Fokkema, 2020)	`sampfrac`	Subsample fraction for drawing samples without replacement for each tree	[0.5, 1.0]
		`minbucket`	Minimum number of observations per leaf	$\{m \in \mathbb{N} \mid m \in [0.01 \cdot N, 0.1 \cdot N]\}$
		`learnrate`	Learning rate	$[10^{-6}, 10^{-1}]$
		`ntrees`	Number of boosting iterations	500

The mentioned hyperparameter names are the names of the corresponding arguments in the respective `R`/`Python` packages

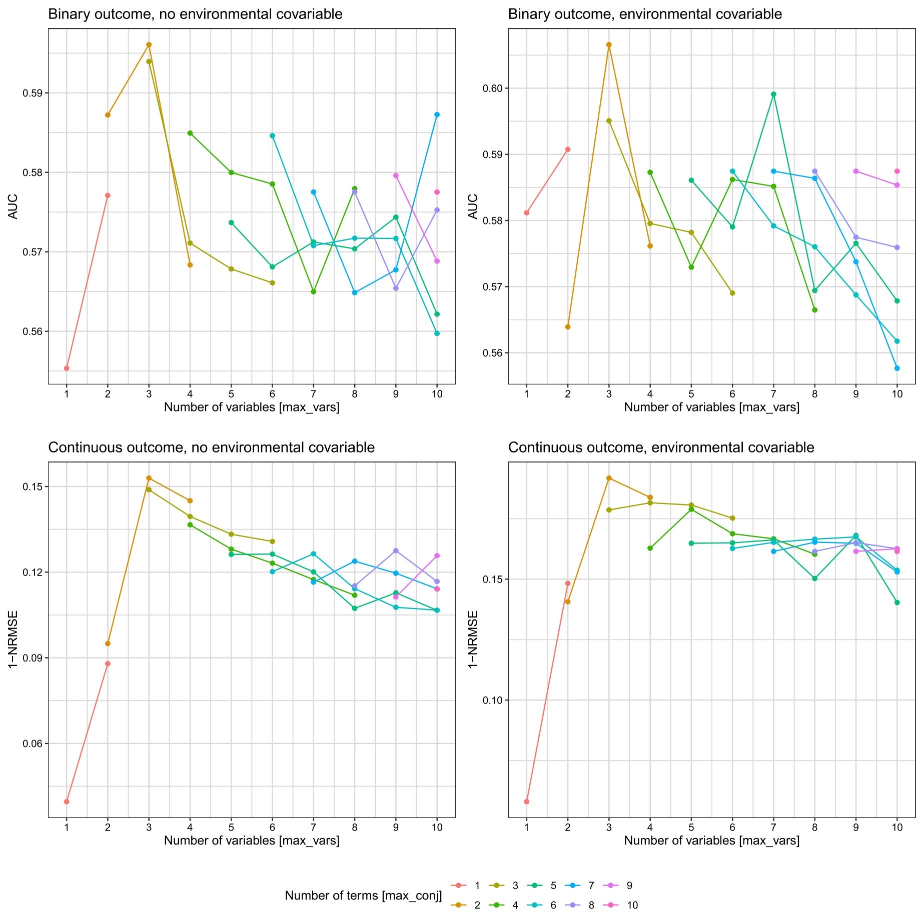


Fig. 5 Predictive performances of different hyperparameter settings for the parameters `max_vars` (maximum number of variables) and `max_conj` (maximum or exact number of terms) in logicDT in the simulation study considering four different scenarios. The performance for binary outcomes is measured by the AUC and the performance for continuous outcomes is measured by the complement of the NRMSE (normalized root mean squared error). Results on validation data sets for the best respective setting of the parameter `nodesize`/`conjsize` in the set $\{1\%, 5\%, 10\%\}$. The evaluated hyperparameter settings are listed in Table 1. Justifications for evaluating these settings are given in Sect. 3.6

For logicDT, Fig. 5 shows the validation data performances for the considered settings of `max_vars` and `max_conj` combined with the respective best setting for `nodesize` /`conjsize`. For each scenario, the highest performance is yielded by $\texttt{max\_vars} = 3$ and $\texttt{max\_conj} = 2$ corresponding to the true underlying simulation models. Generally, the following pattern can be observed. For many `max_conj` settings, the maximizing setting is given by $\texttt{max\_vars} = \texttt{max\_conj} + 1$, which is due to the fact that in this case, additionally to single variables as terms, a conjunction of two variables is contained in the model.

For most considered hyperparameter settings, the validation performance does not seem to vary too severely between similar settings, which indicates that a slight hyperparameter misspecification might not substantially impair the predictive performance of the resulting logicDT model.

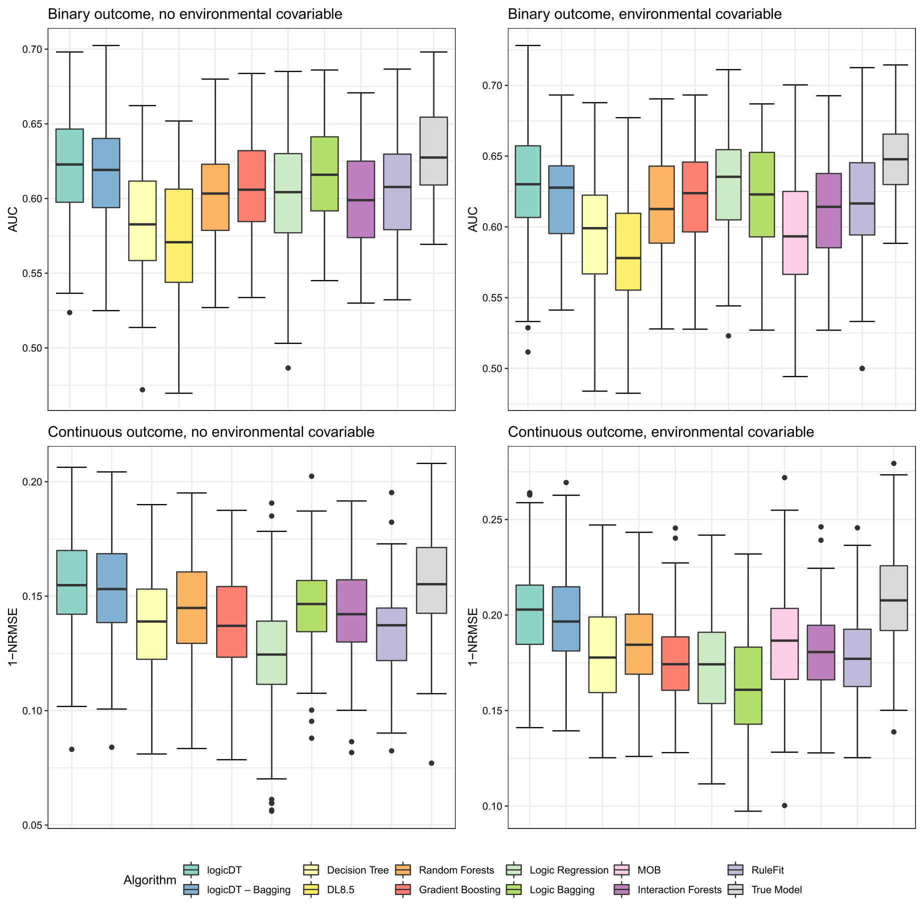


Fig. 6 Predictive performances of logicDT and the comparable methods in the simulation study considering four different scenarios. The performance for binary outcomes is measured by the AUC and the performance for continuous outcomes is measured by the complement of the NRMSE (normalized root mean squared error)

5.1.3 Predictive performance

Figure 6 depicts the performances of logicDT, the comparable methods, and the true underlying model in the simulation study, where the performance of the true model was assessed by performing predictions using the true regression functions presented in Sect. 5.1.1.

In the first simulation scenario considering a binary outcome without an environmental covariable, most notably, standard logicDT and bagged logicDT lead to the best performances, i.e., the largest AUC values, which almost coincide with the performance of the true model. Among the comparable methods, logic bagging seems to be the best method.

For the second scenario in which also a gene-environment interaction is considered, logicDT, bagged logicDT, gradient boosting, logic regression, and logic bagging induce similar results superior to the remaining methods. Here, logicDT and logic regression seem to produce slightly better results than the other procedures.

For the third and fourth simulation scenarios considering a continuous outcome without or with an environmental covariable, logicDT and bagged logicDT yield the highest predictive performances close to the true underlying models. When considering no environmental covariable, logic bagging seems to be the best method among the comparable methods. MOB yields the highest performance among the comparable methods when including an environmental covariable.

5.1.4 Variable importance

Using the VIMs and adjustment approaches for interactions and conjunctions proposed in Sect. 4, we computed variable importances in the four different simulation scenarios. We fitted bagged logicDT models on the 100 complete sub data sets for each scenario. The VIMs themselves were computed using out-of-bag data. For properly summarizing the 100 repetitions, means of the 100 repetitions were computed. A term not occurring in one repetition received a VIM of zero. Additionally, asymptotic 95% confidence intervals for these means μ were calculated by $\hat{\mu} \pm 1.96 \cdot \widehat{\text{se}}$, where $\widehat{\text{se}}$ is the estimated standard error. For binary outcomes, the AUC was used for determining VIMs, while for continuous outcomes, the MSE was employed.

Figure 7 depicts the determined VIMs. For all four scenarios and all three considered measures, the true influential input variables $\text{SNP1D}, \text{SNP2D}, \text{SNP3D}$ receive the highest importance values. Theoretically non-influential terms comprised of variables not influencing the outcome were assigned importance values close to zero in all cases. In the first simulation scenario, the logic VIM and the removal VIM both assign the triplet $\text{SNP1D} \wedge \text{SNP2D} \wedge \text{SNP3D}^c$ the highest importance among all interactions. The permutation VIM favors the sub-conjunction $\text{SNP2D} \wedge \text{SNP3D}^c$ of this triplet. Both interpretations are correct regarding the true model in their own sense, since the term $\text{SNP2D} \wedge \text{SNP3D}^c$ interacts with SNP1D due to squaring the linear predictor.

In the remaining three scenarios, all VIMs assign the term $\text{SNP2D} \wedge \text{SNP3D}^c$ the highest importance among all interactions. However, in the third scenario considering, as in the first scenario, the square of the linear predictor, the conjunction $\text{SNP1D} \wedge \text{SNP2D} \wedge \text{SNP3D}^c$ and additionally sub-conjunctions receive importance values greater than zero. In the last scenario considering a continuous outcome and an influential environmental covariable, the interaction $\text{SNP2D} \wedge \text{SNP3D}^c$ received the highest importance overall for all three importance measures.

In the first three scenarios, the three single input variables yield the highest importances. This is due to the fact that the VIM of single input variables coincides with the standard definition of VIMs, i.e., the difference in error when informatively removing a single input variable. Thus, the VIM of a single input variables captures all of its effects, including effects of interaction in which this input variable participates. In the fourth scenario, the two-way interaction $\text{SNP2D} \wedge \text{SNP3D}^c$ seems to be identified in almost every logicDT application so that the single input variables SNP2D and SNP3D receive lower importances due to being identified less often. Hence, the importances should be compared in groups corresponding to the interaction order, i.e., marginal importances should be compared to each other, two-way interactions should be compared to each other, and so forth.

In summary, all measures yield very similar and plausible results. The determination of the logic VIM is considerably faster than the determination of the removal VIM and the permutation VIM, since the model does not have to be refitted and predictions do

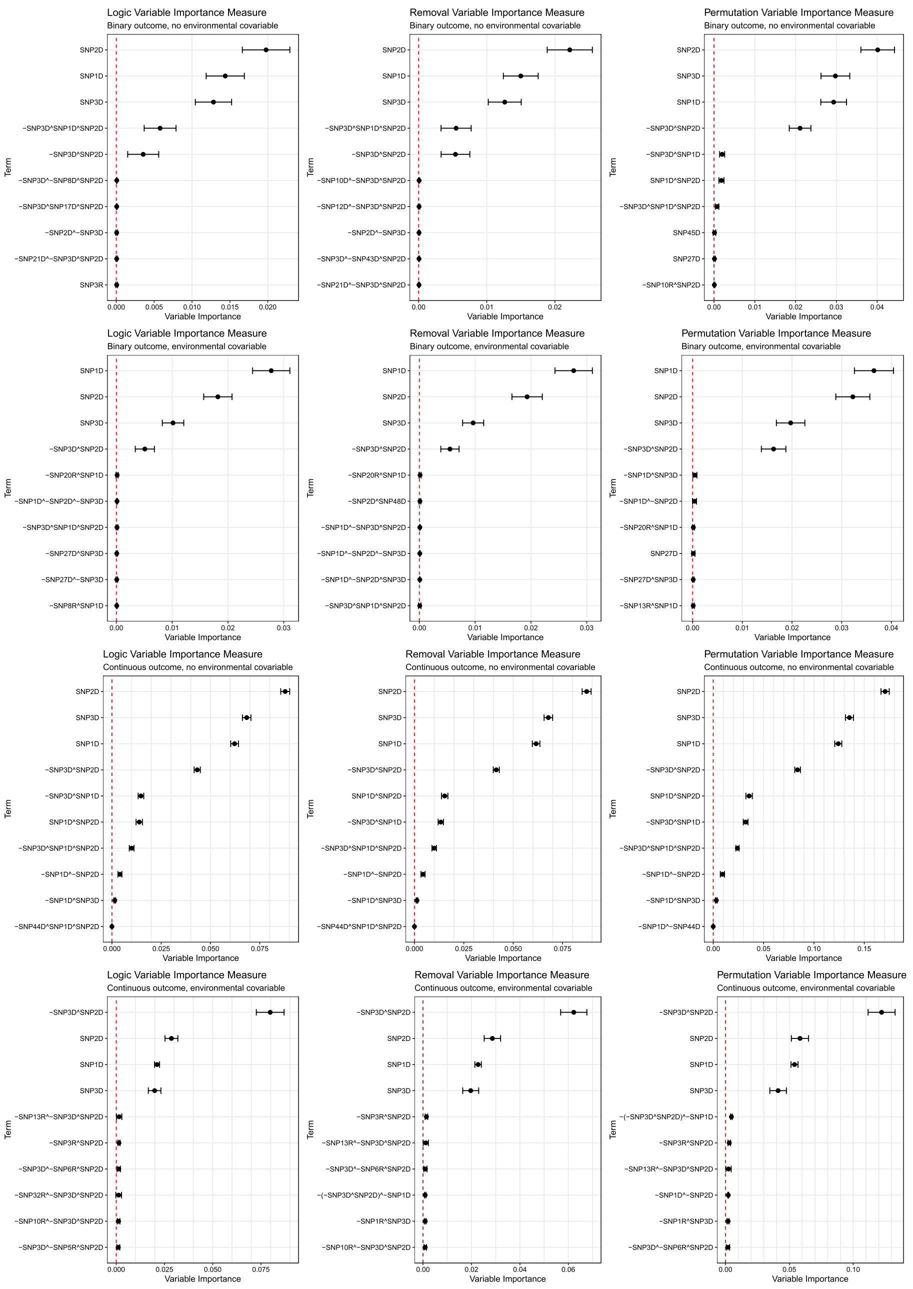


Fig. 7 Logic, removal, and permutation VIMs yielded by bagged logicDT models for the four scenarios in the simulation study. Adjustment for interactions and conjunctions was performed. Means and asymptotic 95% confidence intervals for the 100 repetitions are presented. Negations of input variables are denoted using a minus sign in the front

not have to be performed for a high number of permutations for computing the logic VIM. Instead, for a term consisting of k variables, only 2^k predictions have to performed and compared to the original prediction.

5.1.5 Second simulation setup

To investigate if logicDT is also suitable in scenarios in which a larger amount of input variables is considered and more input variables influence the outcome, we evaluate logicDT and the comparable methods in additional simulations. Two scenarios are investigated, one considering a binary outcome and one considering a continuous outcome, that are both simulated according to the linear model

$$\begin{aligned} g(\mathbb{E}[Y \mid \boldsymbol{X}, E]) = \; & -0.25 + \log(2) \cdot \mathbb{1}(\mathrm{SNP}_1 > 0) + \log(2.5) \cdot \frac{E}{20} \cdot \mathbb{1}(\mathrm{SNP}_2 > 0) \\ & - \log(1.5) \cdot \mathbb{1}(\mathrm{SNP}_3 = 2) - \log(1.5) \cdot \mathbb{1}(\mathrm{SNP}_4 = 0) \\ & + \log(3) \cdot \frac{E}{20} \cdot \mathbb{1}(\mathrm{SNP}_5 > 0) \cdot \mathbb{1}(\mathrm{SNP}_6 = 2) \\ & - \log(3) \cdot \mathbb{1}(\mathrm{SNP}_7 > 0) \cdot \mathbb{1}(\mathrm{SNP}_8 = 0) \cdot \mathbb{1}(\mathrm{SNP}_9 < 2), \end{aligned} \tag{12}$$

where g is the logit function for the binary outcome and the identity function for the continuous outcome. This model was chosen, since it exhibits a more complex structure, as nine SNPs influence the outcome as main effects, two-way interactions, three-way interactions, or gene-environment interactions. In total, 1000 SNPs (i.e., 2000 binary input variables coding for dominant and recessive modes of inheritance for these SNPs) and one continuous covariable were simulated for data sets with sample size $n = 1000$. The input variables are simulated analogously to the ones in Sect. 5.1.1. Both scenarios are, again, evaluated based on 100 independent replications, i.e., 100 random data sets, which are analogously to Sect. 5.1.1 divided into training, validation, and test data sets.

5.1.6 Predictive performance

In Fig. 8, the predictive performance of logicDT and the comparable methods in the application to the two additional simulation scenarios are depicted. Both scenarios seem to be relatively complex, since the discrepancy between the predictive performance of the true model and the fitted models is larger than, e.g., in the previously conducted simulations.

For the binary outcome, the best performance is induced by gradient boosting, closely followed by logicDT, bagged logicDT, random forests, logic regression, and logic bagging. Out of these methods, logicDT and logic regression are the only methods that yield interpretable models. Conventional decision trees, DL8.5, MOB, and RuleFit lead to lower AUCs.

For the continuous outcome, the best results are induced by logicDT, bagged logicDT, gradient boosting, logic regression, logic bagging, and RuleFit. The other interpretability-focused methods, namely conventional decision trees and MOB, yield lower predictive performances.

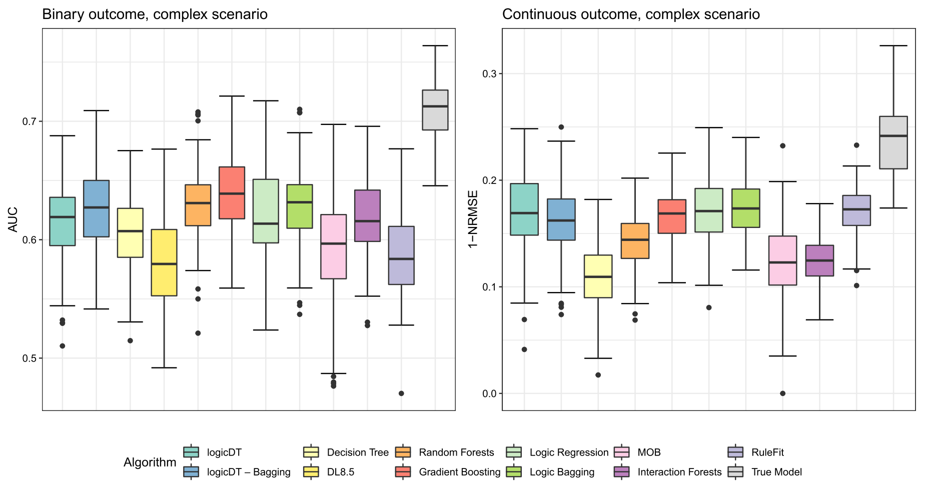


Fig. 8 Predictive performances of logicDT and the comparable methods in the simulation study considering two more complex scenarios. The performance for the binary outcome is measured by the AUC and the performance for the continuous outcome is measured by the complement of the NRMSE (normalized root mean squared error)

Hence, logicDT seems to be also applicable and yielding comparatively high predictive performances, when considering scenarios with larger numbers of input variables (here, 2000 binary input variables) and influential input variables.

5.1.7 Variable importance

In Fig. 9, the estimated variable importances by bagged logicDT in the application to the two additional simulation scenarios are displayed. Since a relatively complex scenario is considered, not every influential term is identified. Nonetheless, for the binary outcome and each considered VIM type, each term with a strongly positive variable importance is truly influential in the underlying data-generating model (12). Moreover, for both the binary and the continuous outcome and all VIM types, the two-way interaction $\text{SNP8D}^c \wedge \text{SNP7D}$ is correctly identified.

For the continuous outcome and the permutation VIM, the five top-ranking importances correspond to truly influential terms. However, the terms showing the next highest importances corresponding to theoretically non-influential terms such as $(\text{SNP8D}^c \wedge \text{SNP7D})^c \wedge \text{SNP2D}$ indicate that these terms are influential as well due to their importance confidence intervals fully being above zero. This issue of falsely identified terms seems to be alleviated when employing the logic VIM or the removal VIM due to less non-influential terms that yield VIM confidence intervals fully above zero when using these VIMs. This, thus, indicates that the logic VIM and the removal VIM in conjunction with the adjustment for interactions can also be employed in more complex scenarios with a larger number of input variables.

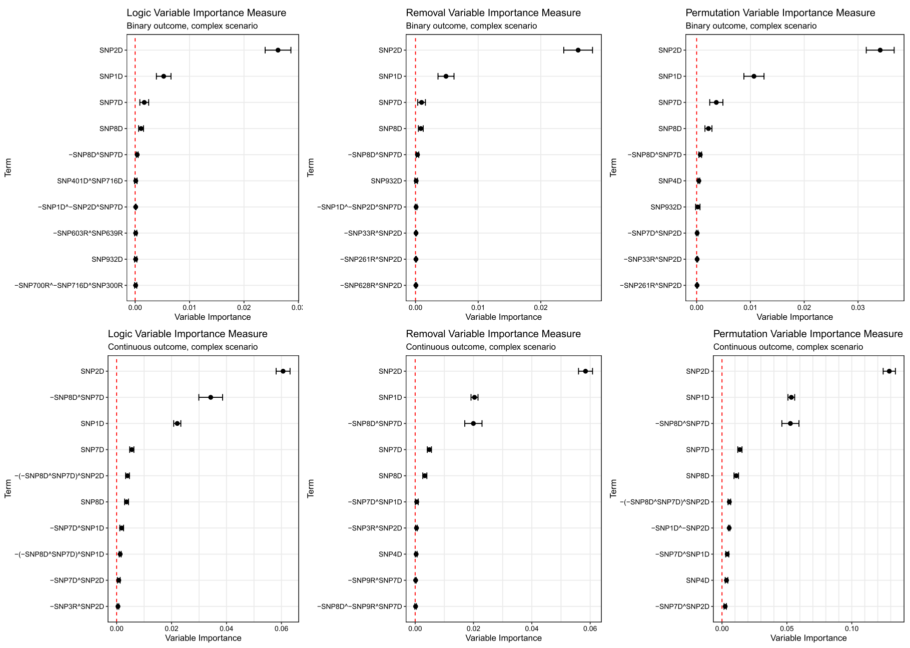


Fig. 9 Logic, removal, and permutation VIMs yielded by bagged logicDT models for the two more complex scenarios in the simulation study. Adjustment for interactions and conjunctions was performed. Means and asymptotic 95% confidence intervals for the 100 repetitions are presented. Negations of input variables are denoted using a minus sign in the front

5.2 Real data application

We have also applied logicDT and the comparable statistical learning methods to several real data sets, from which the data set of the SALIA study is of particular interest. Therefore, we consider, first, in the following subsections this study and the performance of logicDT and the comparable methods in their application to the data from the SALIA study. Afterwards, we summarize the results of the analyses of the other data sets in Sect. 5.2.4. A more detailed discussion of these evaluations can be found in Appendix 4.

5.2.1 SALIA study

logicDT was applied to a real data set from a German cohort study called the SALIA study (**S**tudy on the Influence of **A**ir Pollution on **L**ung, **I**nflammation and **A**ging, Schikowski et al., 2005). The results of logicDT were compared to the results of the methods also considered in the comparisons in Sect. 5.1. The data set consists of data from 517 women, from which 123 had a rheumatic disease so that 394 women did not show a rheumatic disease. For these women, data from 77 SNPs from the HLA-DRB1 gene, which presumably plays a major role in the heritability of rheumatoid arthritis (Clarke & Vyse, 2009), are available. For more details about the SALIA study itself and

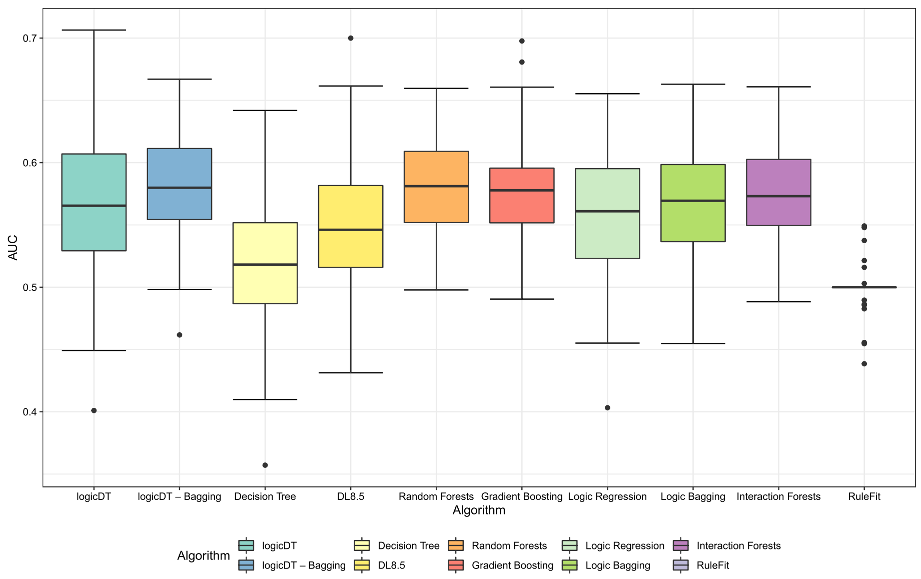


Fig. 10 Predictive performances of logicDT and the comparable methods in the evaluation of the SALIA data

an analysis of rheumatic diseases in the SALIA study, see Krämer et al. (2010) and Lau et al. (2022), respectively.

The analysis was performed using a similar scheme as in the simulation study. For 100 independent repetitions, training, validation and test data sets were randomly drawn from the total data set. Hyperparameter optimization was performed using, again, the parameter values summarized in Table 1.

5.2.2 Predictive performance

In Fig. 10, the performances of logicDT and the comparable methods in their application to the SNP data from the SALIA study are shown. This figure reveals that all evaluated statistical learning procedures induce similarly high AUCs, except for conventional decision trees, DL8.5, and RuleFit, which show inferior predictive performances. RuleFit seems to have issues to detect a signal in the data set at all, despite optimizing its hyperparameters.

We would like to point out that logicDT is the only other procedure than conventional decision trees, DL8.5, logic regression, and RuleFit that yields easily interpretable prediction models. In contrast to these models, logicDT still leads to comparatively high predictive performances. Single logic regression models yield similar AUCs as logicDT. However, due to logic regression models including complex terms consisting of mixtures of Boolean conjunctions and disjunctions, logic regression models tend to be harder to interpret than logicDT models.

Figure 11 shows the fitted logicDT model on the complete SALIA data. This tree is still relatively easy to interpret, i.e., it is easy to understand how predictions are made and which interactions are involved in the prediction. In comparison, the fitted logic regression model on the complete SALIA is given by

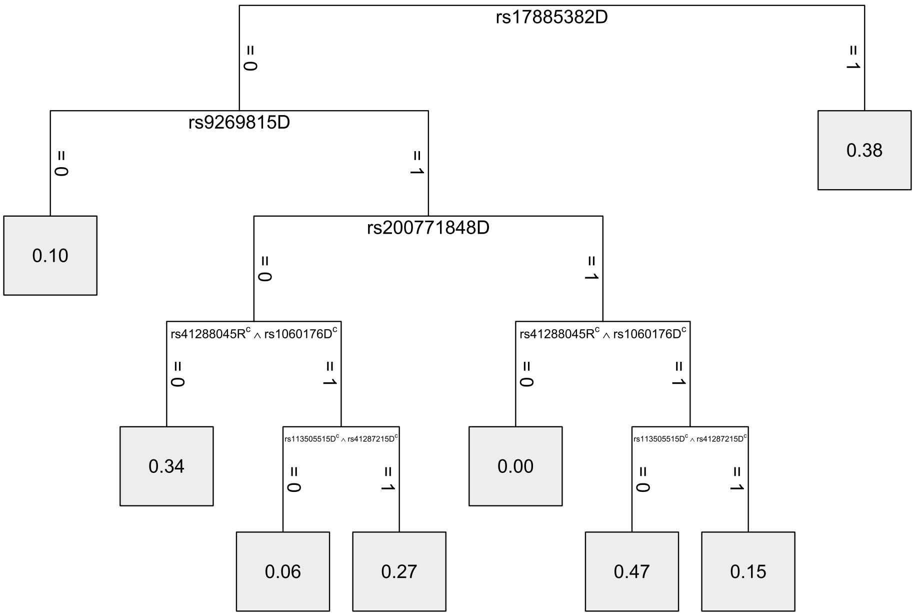

rs17885382D
= 0
= 1
0.38
rs9269815D
= 0
= 1
0.10
rs200771848D
= 0
= 1
rs41288045R^c ∧ rs1060176D^c
= 0
= 1
0.34
rs41288045R^c ∧ rs1060176D^c
= 0
= 1
0.00
rs113505515D^c ∧ rs41287215D^c
= 0
= 1
0.06
0.27
rs113505515D^c ∧ rs41287215D^c
= 0
= 1
0.47
0.15

Fig. 11 Fitted logicDT model on the complete SALIA data

$$
\begin{aligned}
\text{logit}(\mathbb{P}(Y=1 \mid \boldsymbol{X})) = &- 1.14 \\
&- 19.63 \cdot \mathbb{1}((\text{rs113608847D} \wedge (\text{rs113505515D}^c \vee \text{rs9270143R})) \\
&\qquad \wedge (\text{rs1060176D} \vee (\text{rs28724138R}^c \wedge \text{rs17884945R}^c))) \\
&- 2.91 \cdot \mathbb{1}((\text{rs34578704D}^c \wedge \text{rs34084957D}) \\
&\qquad \vee ((\text{rs41288045R} \vee \text{rs9269814D}^c) \vee \text{rs72844253R})) \\
&+ 1.41 \cdot \mathbb{1}((\text{rs113322920D} \vee \text{rs36101847R}) \wedge \text{rs17879702D}^c).
\end{aligned}
$$

For this model, it is not trivial which interactions are involved in the prediction and how predictions for $\mathbb{P}(Y=1 \mid \boldsymbol{X})$ are constructed.

5.2.3 Variable importance

Figure 12 illustrates the measured variable importances in the application to the SALIA data for the three proposed VIM approaches using bagged logicDT models. In the top row, the importances for the top 5 single input variables are depicted. In the second and third row, the importances for the top 5 two-way and three-way interactions are shown, respectively.

For verifying whether the terms identified by logicDT really have an influence on the outcome of interest, i.e., the rheumatic disease status, we considered for each identified term X in Fig. 12 a logistic regression model

$$\text{logit}(\mathbb{P}(Y=1 \mid X=x)) = \beta_0 + \beta_1 x \tag{13}$$

and performed statistical hypothesis tests testing whether the respective term has an influence on the outcome, i.e., testing $H_0 : \beta_1 = 0$ vs. $H_1 : \beta_1 \neq 0$ using a Wald test. For each

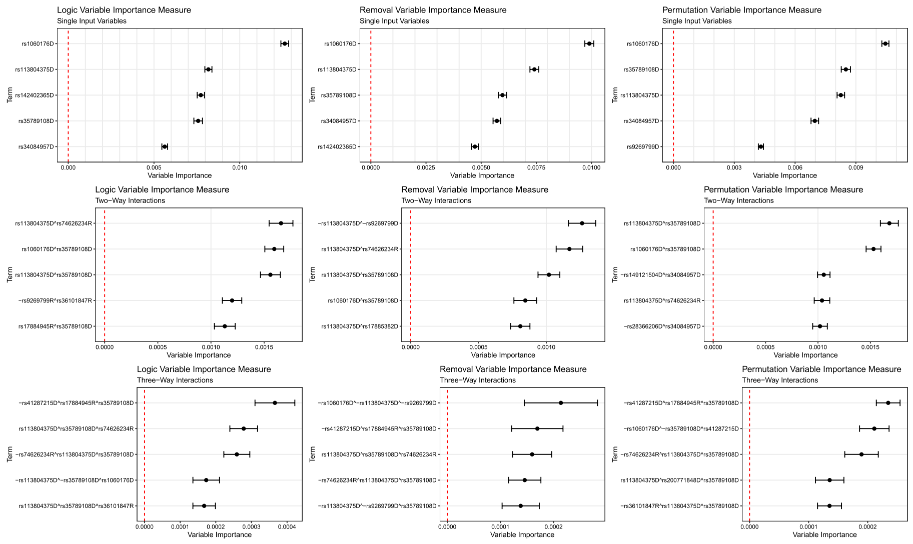


Fig. 12 Logic, removal, and permutation VIMs yielded by bagged logicDT models in the evaluation of the SALIA data—divided into VIMs of single input variables, two-way interactions and three-way interactions. Adjustment for interactions and conjunctions was performed. Means and asymptotic 95% confidence intervals for the 100 repetitions are presented. Negations of input variables are denoted using a minus sign in the front

Table 2 Numbers of identified terms from Fig. 12 that were significant with respect to $\alpha = 5\%$ using a false discovery rate adjustment

Significant terms/5	Logic VIM	Removal VIM	Permutation VIM
Single Input Variables	0	0	0
Two-Way Interactions	4	2	2
Three-Way Interactions	5	3	4

set of five identified terms, we evaluated how many terms lead to significant coefficients in the model from Eq. (13) using a significance level of $\alpha = 5\%$ and adjusting for multiple testing using the method by Benjamini and Hochberg (1995).

Table 2 shows the results for this post-hoc analysis. None of the identified single input variables proves to be significant. However, for the logic VIM, four of the five identified two-way interactions and all five three-way interactions seem to have a significant influence on the outcome. For the more computationally intensive removal and permutation VIMs, the results seem to be inferior, since only two of the five two-way interactions are significant, and three or four of the five three-way interactions, respectively, are significant.

Note that the VIMs of the single input variables depicted in Fig. 12 are considerably higher than the VIMs of the interaction terms, yet the single input variables were not significant. As discussed in the simulation study in Sect. 5.1.4, this is due to the fact that the VIMs for single input variables also capture the importance of interactions that contain the input variable of interest. Thus, if a single input variable is part of many interactions, this inflates its importance value without leading to a significant main effect of the variable. For example, the most influential input variable across all three VIM calculation approaches, rs1060176D, is in every considered situation part of one identified interaction term.

5.2.4 Additional real data evaluations

logicDT and the comparable methods are also evaluated in additional experiments using 24 real data sets from various application fields. The main result is that logicDT induces high predictive performances among single-model procedures in the application to these additional real data sets. Among the ensemble methods, bagged logicDT also induces for most data sets relatively high predictive performances. More details on the analyses of the additional real data sets can be found in Appendix 4.

6 Discussion

In this article, we have presented a statistical learning procedure called logicDT that is specifically tailored to finding interactions between binary input variables and that can also take continuous covariables into account by fitting regression models in the decision tree branches. In contrast to, e.g., logic regression, all possible interactions of the binary input data with this continuous covariable can be included in the prediction model as well as interactions between interactions of the binary input data. logicDT is aimed at maximizing both predictive power and interpretability motivated by applications in genetic epidemiology.

As a simulation study as well as real data applications show, logicDT is able to fulfill these objectives and yields comparable or better predictive performances as similar methods, while maintaining interpretability, which is lost when applying most other approaches. Moreover, through simulated annealing and theory on decision trees, theoretical success of logicDT, i.e., that the true underlying regression function is asymptotically attained, could be proven.

For maximizing the predictive performance regardless of being able to interpret how exactly predictions are made, bagging can be applied to logicDT, yielding performances as state-of-the-art algorithms such as random forests or gradient boosting.

Through different VIMs and VIM adjustment approaches for measuring the importances of interactions and specific conjunctions, highly predictive bagged logicDT models are still very useful for deriving which variables influence the outcome in interaction with which other variables. In comparison to standard VIM approaches, the proposed interaction VIM is able to capture influences of interactions and is not restricted to single input variables. Note that the proposed VIM adjustment approaches can also be applied to other statistical learning procedures, e.g., black-box methods such as deep neural networks or random forests, since no restricting assumptions on the model fitting procedure itself are made in these approaches.

Fitting logicDT models is computationally intensive due to the global search via simulated annealing, and takes, in particular, more time than fitting conventional decision trees that employ a greedy algorithm. However, as could be seen in the simulation study and the real data applications, logicDT consistently outperformed conventional decision trees considering the predictive performance. Moreover, logicDT still does not seem to be slower than other interpretability-focused methods such as logic regression or RuleFit. A model fitting time evaluation of logicDT and other procedures in the simulation study and real data application can be found in Appendix 5.

logicDT was designed for interpretable modeling in low- to mid-dimensional problems, e.g., considering single genes, pathways, or selections of SNPs that were significantly influencing the outcome in prior analyses. However, in theory, logicDT can be applied to problems with an arbitrarily large number p of input variables. Nonetheless, as shown in Sect. 3.8, the computational complexity of logicDT is polynomial in p under certain assumptions. Moreover, in practice, only finitely many computational resources are available. In simulations considering 1000 SNPs (i.e., $p = 2000$ input variables due to splitting each SNP into two binary variables) and a more complex underlying model, logicDT still induced relatively high predictive performances (see Sect. 5.1.5). Hence, we recommend applying logicDT in situations with $p \leq 2000$. For comparison, in the software implementation of logic regression, where also a stochastic search algorithm is employed, the authors allow a maximum of $p = 1000$ input variables (Kooperberg & Ruczinski, 2022).

The main issue of conventional decision trees is its instability issue, i.e., that small modifications of the training data set imply unproportionally severe alterations of the fitted model. This behavior is mainly induced by the greedy fitting algorithm (Li & Belford, 2002; Murthy & Salzberg, 1995). logicDT aims at identifying the globally optimal set of predictors and interactions responsible for the variation in the outcome. Thus, only important predictors are used for fitting the decision tree and interactions are already covered by single splits. Therefore, the instability issue should be diminished by logicDT.

The search procedure in logicDT utilizes the training data both for fitting decision trees and scoring them for guiding the search, which might suggest that this might lead to overfitting. However, both training trees based on states and evaluating states are part of the logicDT fitting procedure and the balance of overfitting and underfitting is controlled by the hyperparameters tuned using independent validation data (see Sect. 3.6). Moreover, established statistical modeling approaches such as stepwise linear regression or logic regression also employ the full training data set for both fitting the models and guiding the search. Nonetheless, one idea might be to further split the available training data into training data for fitting the decision tree based on the considered state and inner validation data for scoring the state's performance. However, due to the need for further splitting the available data, less observations are available for both the tree fitting step and the scoring step, leading to a decreased performance (on independent test data) compared to the original algorithm in empirical experiments (see Appendix 6). Moreover, the resulting model should not heavily rely on the data split used for this inner validation. Hence, ideally, multiple data splits—fitting and scoring multiple trees for one state and averaging the results as in (inner) cross-validation—should be used, leading to an increased computational burden.

Bagged logicDT was designed for situations in which a larger number of input variables influences the outcome or variable/interaction term importances shall be measured. In the simulation study conducted in Sect. 5.1.1, bagged logicDT performed similarly well compared to logicDT due to single logic decision trees being able to fully capture the considered underlying models. In additional simulations considering scenarios with larger numbers of influential input variables (see Sect. 5.1.5) and real data evaluations (see Appendix 4), bagged logicDT was able to achieve higher predictive performances in comparison to logicDT. Nevertheless, in these additional analyses, logicDT induced strong performances compared to other single-model methods.

For bagged logicDT, one idea to further increase its performance might be to further randomize the search similar to random forests. This could be realized by selecting a random sample of the neighbor states to be evaluated in each iteration of the greedy search, which is similar to randomly sampling potential splitting variables in random forests. However, this would create another hyperparameter—the number of randomly drawn candidate neighbor states—that potentially should be tuned and could depend on the total number of neighbor states that can change for each considered state.

logicDT is motivated by applications in genetic epidemiology in which mainly binary input data is analyzed. Although not considered in this article, it is possible to generalize logicDT to numerical input data by considering numerical interactions $\prod_j X_j$ instead of Boolean conjunctions $\bigwedge_j X_j$, where in the case of binary input data, these two definitions coincide.

The development of logicDT was, more precisely, motivated by the problem of constructing genetic risk scores that are usually built based on linkage-disequilibrium-based pruned SNPs, i.e., SNPs that can be interpreted as independent variables (So & Sham, 2017; Dudbridge & Newcombe, 2015). Therefore, throughout this manuscript, the assumption was made that there are no strong correlations between the considered input variables. In future research, logicDT and the interaction VIM could be analyzed and potentially adjusted for settings in which strong correlations between input variables exist so that, ideally, input variables (highly) correlated with truly predictive input variables do not diminish the importance of these truly predictive input variables.

If, additionally, a quantitative variable such as a quantitative environmental variable is considered, logicDT uses this covariable to fit regression models in the leaves of the decision tree. Since logicDT splits, in the context of genetic epidemiology, on genetic variants, a gene-environment is present if and only if the leaf regression models differ more than by fixed offsets describing marginal effects of the genetic variants. Thus, in future research, logicDT could be expanded for statistically testing the presence of a gene-environment interaction in the considered subregion of the DNA.

Moreover, the proposed interaction importance measuring methodology could also be expanded for statistically testing if certain single input variables or interaction terms significantly influence the outcome. This can, e.g., be used in the context of genetic epidemiology, testing the presence of gene-gene interactions. For implementing this testing procedure, the variable importance testing framework proposed by Watson and Wright (2021) might be applied to the importance measures proposed in this manuscript.

7 Conclusion

logicDT yields highly interpretable decision trees with superior predictive performances compared to other single-model procedures such as standard decision trees by being able to detect interaction effects between binary predictors on split level. Fitting ensembles of logicDT models through bagging can further increase the predictive performance if many predictors have effects on the outcome. The novel VIM adjustment procedure can be applied to these logicDT ensembles to derive which input variables influence the outcome in which interplay and magnitude—also measuring the importance of interaction effects between input variables.

Appendix 1: Simulated-annealing-based search procedure

The main methodology of logicDT, for which consistency is proven, employs simulated annealing as its search algorithm. In applications of logicDT, we suggest using an adaptive cooling schedule that requires no temperature tuning at all, which is in contrast to a geometric cooling schedule that is, e.g., used in logic regression. An adaptive cooling schedule automatically tunes the cooling behavior of simulated annealing, i.e., the start temperature, the temperature lowering steps or the Markov chain lengths, and the end temperature. Using an adaptive cooling schedule simplifies the application of simulated annealing, since these parameters do not have to be fine-tuned manually.

The start temperature is generally chosen such that at the beginning of the algorithm essentially a random walk is performed. For finding an appropriate initial temperature, a brief random walk over the state space (e.g., visiting 10,000 states) is carried out in logicDT and the state scores are recorded. Since the acceptance function

$$\gamma(\epsilon(s), \epsilon(s'), t) = \min\left\{1, \exp\left(\frac{\epsilon(s) - \epsilon(s')}{t}\right)\right\}$$

in simulated annealing is chosen so that better or equal states are automatically accepted, the temperature only influences the acceptance behavior of proposed worse states. Thus, only moves leading to worse states are used to estimate a temperature at which, e.g., 90% of the worse states are accepted.

We employ the homogeneous version of simulated annealing that runs through many consecutive homogeneous Markov chains. In practice, we limit the number of iterations per chain to, e.g., 1000 and adaptively choose the next temperature in a way that equilibrium of the next chain can be easily reattained. More precisely, we employ the temperature lowering scheme proposed by Huang et al. (1986) that is given by

$$t' = t \cdot \exp\left(-\lambda \frac{t}{\sigma(t)}\right),$$

where t is the current temperature, t' is the new temperature of the next Markov chain, and $\sigma(t)$ is the standard deviation of the scores observed in the finished Markov chain (see also, e.g., Van Laarhoven & Aarts, 1987). Here, $\lambda \in (0, 1]$ is a parameter controlling the speed of the total algorithm, which means that a higher value of λ leads to larger decreases in the temperature t, and hence, to less total iterations. Consequently, a value closer to 0 leads to a finer search, requiring more iterations. Generally, more iterations are preferable for approximating the theoretical asymptotic search. However, in practice, we recommend using a value of $\lambda \in [0.01, 0.1]$ for performing at least a few hundred thousand iterations.

For stopping the stochastic search, we evaluate the fraction of accepted states yielding a different score than the previous one, i.e., ignoring two neighbor states that yield the exact same score. If, e.g., for five consecutive chains the fraction of this adjusted state acceptance ratio is below 1%, the search is terminated. Alternatives include using the total number of chains instead of restricting to consecutive ones or using, similar to Triki et al. (2005), the standard deviation of the scores in a chain. For very small temperatures, simulated annealing should only move to better or equal states in terms of the score function. Thus, if an ideal state is reached, the score should no longer change, leading to a standard deviation of the score of 0.

Similar to the cooling schedule proposed by Triki et al. (2005), in the beginning of the search, the lowering of the temperature will also be triggered, if a threshold of accepted states in a single Markov chain is reached. This threshold might, e.g., be set to 50% of the total Markov chain length and prevents the search from focusing too long on the initial near random walk type of search, but instead focusing on the middle part of simulated annealing.

The theory of simulated annealing is based on two convergences, namely

- the convergence of the individual Markov chains, i.e., reaching equilibrium or the respective stationary distribution,
- the convergence of the temperature to 0, i.e., approaching an infinitesimal low temperature.

In practice, since limited computing resources are available, there is no guarantee that simulated annealing finishes in a global minimum. Thus, it might be possible that a globally optimal state is visited in the initial exploration of the state space, but due to relatively few iterations another local optimum is reached afterwards and is not abandoned anymore. We, therefore, let the algorithm also remember the best visited state so far in the search.

Due to noninformative terms or noninformative predictors in a conjunction, it might be possible that two neighbor states yield the exact same score. In this case, generally the simpler model is preferred. Thus, when the search is finished, each term is inspected for variables and conjunctions that do not improve the score and these variables or conjunctions are removed from the model. Furthermore, if a new neighbor is proposed that leads to exactly the same score as the current state, this new neighbor is accepted in simulated annealing, regardless of the current temperature.

Visiting a single state multiple times can also occur due to the random nature of simulated annealing itself. To account for this behavior in the searching procedure, a hash table containing sorted linked lists of the specific states and their respective scores is used for remembering already visited states. Thus, if a state is reached multiple times, the predictor transformation and the decision tree fitting do not have to be repeated.

Appendix 2: Consistency proof

In this appendix, we prove Theorem 1 that was stated in Sect. 3.7. For proving this theorem, some preliminary results are necessary that are proven in the following lemmata. We start by proving that simulated annealing leads to an optimal solution in logicDT.

Lemma 1 *The Markov chains constructed in logicDT fulfill the prerequisites of simulated annealing such that the stationary distributions* $\pi_t = \lim_{q\to\infty} \mathbb{P}(Q_t(q) = \cdot)$ *exist and it holds that*

$$\lim_{t \searrow 0} \pi_t(s) = \begin{cases} \dfrac{|\mathcal{R}_s|}{\sum_{s' \in \mathcal{R}_{\text{opt}}} |\mathcal{R}_{s'}|}, & s \in \mathcal{R}_{\text{opt}} \\ 0, & s \notin \mathcal{R}_{\text{opt}} \end{cases}$$

for the set $\mathcal{R}_s$ *of neighbor states of state* s *and the set* $\mathcal{R}_{\text{opt}}$ *of optimal states.*

Proof For establishing convergence of the individual (finite and homogeneous) Markov chains to their stationary distributions, it is sufficient to prove their irreducibility and aperiodicity (e.g., Theorem 1 in Section 3.1.2, Van Laarhoven & Aarts, 1987).

The Markov chains Q_t in simulated annealing are generally based on the transition probabilities

$$\tau\big(s, s', t\big) \;:=\; \mathbb{P}\big(Q_t(q+1) = s' \mid Q_t(q) = s\big) \;=\; \gamma\big(\epsilon(s), \epsilon(s'), t\big) \cdot \beta\big(s, s'\big)$$

for $s \neq s'$ and all $q \in \mathbb{N}$, where $\gamma(\epsilon(s), \epsilon(s'), t)$ describes the acceptance probability depending on the scores $\epsilon(\cdot)$ of the states and $\beta(s, s')$ yields the generation probability of s' given state s. In logicDT, the standard acceptance function

$$\gamma(\epsilon(s), \epsilon(s'), t) \;=\; \min\left\{1, \exp\left(\frac{\epsilon(s) - \epsilon(s')}{t}\right)\right\} \tag{14}$$

is used together with the uniform distribution for the generation probability

$$\beta(s, s') \;=\; \begin{cases} \frac{1}{|\mathcal{R}_s|}, & s' \in \mathcal{R}_s \\ 0, & s' \notin \mathcal{R}_s. \end{cases} \tag{15}$$

Since $\gamma(\epsilon(s), \epsilon(s'), t) > 0$ for every pair of states s, s' and $t > 0$ and the choice of the moves, i.e., modifications of states, proposed in Sect. 3.2 ensure that each state can be reached from any other state in a finite number of steps, the Markov chains in logicDT are irreducible.

Aperiodicity is fulfilled, if for all states s the greatest common divisor (gcd) of

$$\Big\{n \in \mathbb{N} \mid \tau_n(s, s, t) := \mathbb{P}(Q_t(1+n) = s \mid Q_t(1) = s) > 0\Big\}$$

is equal to 1. This property would be directly fulfilled, if the chains would be reflexive, i.e., fulfilling $\tau(s, s, t) > 0$ for each state s. However, since it might be the case that a state has only neighbors exhibiting better scores, leading to $\gamma(\epsilon(s), \epsilon(s'), t) = 1$ for each $s' \in \mathcal{R}_s$, the probability of staying in state s can be equal to 0, as, for the probability of proposing the current state, it holds that $\beta(s, s) = 0$ by choice of β. Therefore, three different cases for states s have to be considered.

Case 1: s has a neighbor state s' with $\epsilon(s') < \epsilon(s)$. In this case, the probability $\tau(s, s', t)$ of changing to state s' is positive. The probability $\tau(s', s, t)$ of returning to s is positive as well, which is due to $\gamma > 0$. Furthermore, the probability $\tau(s', s', t)$ of remaining in s' is also positive, since, if s is generated by $\beta(s', \cdot)$, s will be accepted with probability $\gamma(s', s, t) < 1$ because of $\epsilon(s) - \epsilon(s') > 0$ and the choice of γ in Eq. (14). Thus, $\tau_2(s, s, t) > 0$ and $\tau_3(s, s, t) > 0$ hold true yielding the greatest common divisor of $\gcd(2, 3) = 1$.

Case 2: s has at least one neighbor state s' with $\epsilon(s') > \epsilon(s)$, but no neighbor s'' with $\epsilon(s'') < \epsilon(s)$. In this case, it holds that $\gamma(\epsilon(s), \epsilon(s'), t) \in (0, 1)$, and thus,

$$\tau_1(s, s, t) \;=\; \tau(s, s, t) \;>\; 0.$$

Case 3: For all neighbor states s' of s, it holds that $\epsilon(s') = \epsilon(s)$. Here, we have

$$\gamma(\epsilon(s), \epsilon(s'), t) \;=\; \gamma(\epsilon(s'), \epsilon(s), t) \;=\; 1,$$

and therefore, $\tau_2(s, s, t) > 0$. Let s'' be another state with $\epsilon(s'') \neq \epsilon(s)$. Such a state has to exist, since otherwise each state would have the exact same score. The state s'' can be chosen such that, due to the irreducibility, there exists a sequence of states $(s, s_1, s_2, \dots, s_n, s'')$, in which each succeeding state is a neighbor of its predecessor, with

$$\epsilon(s) = \epsilon(s_1) = \epsilon(s_2) = \cdots = \epsilon(s_n)$$

for any $n \in \mathbb{N}$. Thus, it follows $\epsilon(s_n) \neq \epsilon(s'')$.

Case 3.1: $\epsilon(s_n) > \epsilon(s'')$. Due to $\epsilon(s'') - \epsilon(s_n) < 0$ and Eq. (14), it follows $\gamma(s'', s_n, t) < 1$, and hence, $\tau(s'', s'', t) > 0$. Using the state sequence $(s, s_1, \dots, s_n, s'', s'', s_n, \dots, s_1, s)$, it becomes obvious that $\tau_{2n+3}(s, s, t)$ is positive. Furthermore, it follows that $\gcd(2, 2n + 3) = 1$, as $2n + 3$ is odd.

Case 3.2: $\epsilon(s_n) < \epsilon(s'')$. Analogously to Case 3.1, it follows that $\tau(s_n, s_n, t) > 0$. Using the state sequence $(s, s_1, \dots, s_n, s_n, \dots, s_1, s)$, the probability $\tau_{2n+1}(s, s, t)$ has to be positive so that $\gcd(2, 2n + 1) = 1$, since $2n + 1$ is odd.

Thus, aperiodicity is given so that the individual limiting distributions exist.

Applying Theorem 2 from Section 3.1.3 of Van Laarhoven and Aarts (1987) to the constructed Markov chains using the choices for γ in Eq. (14) and β in Eq. (15) directly shows that the stationary distributions converge to a distribution that exactly has the set of optimal states as its support. □

Now we have to show that the empirical risk minimization (ERM), which is performed by simulated annealing, is asymptotically equivalent to a true risk minimization in logicDT.

Lemma 2 (ERM consistency of logicDT) *Let the outcome Y be bounded. Then, logicDT is strongly consistent with respect to empirical risk minimization, i.e.,*

$$\sup_T \left| R_{\text{emp}}(T) - R_{\text{true}}(T) \right| \xrightarrow[n\to\infty]{\text{a.s.}} 0,$$

where $R_{\text{emp}}(T) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, T(\boldsymbol{x}_i))$ is the empirical risk, $R_{\text{true}}(T) = \mathbb{E}_{(X,Y)}[L(Y, T(\boldsymbol{X}))]$ is the true risk, and $L(y, \hat{y}) = (y - \hat{y})^2$ is the squared error loss.

Proof By assumption, Y is bounded. Thus, as the predictions of decision trees are generated by means of observed values, the predictions are bounded by the same bound. Furthermore, the L_2 loss is bounded likewise. Let this bound be given by $B > 0$, i.e., $L(y, \hat{y}) \in [0, B]$.

In order to prove distribution-independent ERM consistency, it is necessary and sufficient that the VC (Vapnik and Chervonenkis) dimension is finite (Vapnik, 2000), where the VC dimension is defined as the maximum number m of data points $z_1, \dots, z_m := (\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m)$ that can be shattered by a binary loss function $L(y, T(\boldsymbol{x})) \in \{0, 1\}$. For $m \in \mathbb{N}$, there thus exists a sample $z_1, \dots, z_m$ such that for all possible 2^m binary outcomes $\in \{0, 1\}^m$ there exists a prediction function T in the considered space that divides the sample according to the label setting using the loss function $L(y, T(\boldsymbol{x}))$. In the general regression setting, the VC dimension is defined as the VC dimension of the indicators $\mathbb{1}(L(y, T(\boldsymbol{x})) \geq \beta)$, where $\beta \in [0, B]$ is interpreted as part of the function space for the determination of the VC dimension so that for each outcome setting a function T and a value for β have to be found.

For deriving the VC dimension of logicDT, note that the prediction values for each predictor setting can be chosen independently, i.e., it is only necessary to consider for how many data points the data can be shattered along one single predictor setting. In the case of not fully grown trees with shared leaves for different possible predictor settings (for example, a tree stump only splitting on $X_1 \in \{0, 1\}$ such that $T((X_1, 0)) = T((X_1, 1))$), the

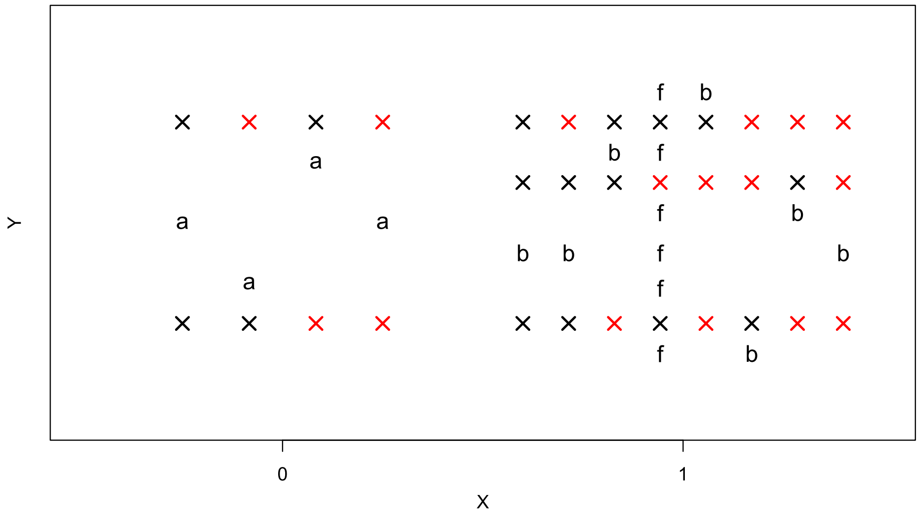


Fig. 13 VC dimension illustration for logicDT models. Here, one binary predictor $X \in \{0, 1\}$ is considered. For $X = 0$, all $2^2 = 4$ classifications for two data points are depicted. For $X = 1$, all $2^3 = 8$ classifications for three data points are shown. Black crosses indicate $\mathbb{1}(L(y, T(\boldsymbol{x})) \geq \beta) = 0$. Red crosses indicate $\mathbb{1}(L(y, T(\boldsymbol{x})) \geq \beta) = 1$. a and b are the (fixed) predictions $T(0) = a$, $T(1) = b$ such that the corresponding classification pattern can be achieved, i.e., there exists an appropriate β. f depicts the situation in which an appropriate prediction value, and thus, an appropriate tree cannot be constructed

prediction values are not necessarily independent of each other. However, in this case, the number of shatterable data points decreases compared to the independent prediction case so that this case does not have to be considered with regard to the VC dimension. Thus, it is sufficient to consider one single predictor $X \in \{0, 1\}$, since the shattering behavior only has to be analyzed independently for each setting.

Figure 13 depicts the shatterability for two and three data points, respectively. Two observations on one axis, as depicted here for $X = 0$, can be shattered by properly positioning the corresponding prediction value $a = T(0)$ and choosing an adequate β so that

$$\mathbb{1}((y-a)^2 \geq \beta) = \begin{cases} \text{Red} \times, & (y-a)^2 \geq \beta \\ \text{Black} \times, & (y-a)^2 < \beta, \end{cases}$$

i.e., choosing a and β such that red crosses are "far away" from a and black crosses are "close" to a.

For three data points, there is only one problematic labeling: If three different observations are considered that lie on one axis, one data point has to be the middle point. This middle point cannot be classified as 1/red while classifying the outer points as 0/black. This is due to the fact that the middle point needs to be far away from the prediction $b = T(1)$ to achieve this labeling, while the surrounding points need to be close to b, which is not possible.

Thus, for each prediction axis/tree branch, two is the maximum number of points that can be shattered. Since for p predictors there are 2^p possible predictor settings and two data points can be shattered for each setting, the VC dimension $\mathcal{VC}$ of logicDT is equal to

$$\mathcal{VC} = 2 \cdot 2^p = 2^{p+1}.$$

For bounded loss composition functions $L \circ \tilde{T} : \mathcal{X} \times \mathcal{Y} \to [0, B]$ with $\tilde{T}(\boldsymbol{x}, y) := (y, T(\boldsymbol{x}))$, where L is in here given by $L(y, \hat{y}) = (y - \hat{y})^2$ so that $(L \circ \tilde{T})(\boldsymbol{x}, y) = (y - T(\boldsymbol{x}))^2$, a uniform bound

$$\mathbb{P}\left(\sup_T \left|R_{\mathrm{emp}}(T) - R_{\mathrm{true}}(T)\right| > \varepsilon\right) \le 4 \exp\left\{\left(\frac{G(2n)}{n} - \frac{\varepsilon^2}{B^2}\right) n\right\} \tag{16}$$

involving the growth function G holds for all $\varepsilon > 0$ (see Equation (3.10), Vapnik, 2000). This growth function G is bounded by a function of the VC dimension. In particular, for $n > \mathcal{VC}$, it holds that

$$G(n) \le \mathcal{VC}\left(\log\left(\frac{n}{\mathcal{VC}}\right) + 1\right) \tag{17}$$

(see Equation (3.23), Vapnik, 2000).

For proving almost sure convergence of (16), i.e., for proving

$$\sup_T \left|R_{\mathrm{emp}}(T) - R_{\mathrm{true}}(T)\right| \xrightarrow[n \to \infty]{\text{a.s.}} 0, \tag{18}$$

it suffices to show that the corresponding series converges (see, e.g., Corollary 1, Section 1.11.1, Vapnik, 1998), i.e., that

$$\sum_{n=1}^{\infty} \mathbb{P}\left(\sup_T \left|R_{\mathrm{emp}}(T) - R_{\mathrm{true}}(T)\right| > \varepsilon\right) < \infty.$$

Using (17), the right-hand side of (16) is bounded by

$$4 \exp\left\{\left(\frac{G(2n)}{n} - \frac{\varepsilon^2}{B^2}\right) n\right\} \le 4 \exp\left\{\mathcal{VC}\left(\log\left(\frac{2n}{\mathcal{VC}}\right) + 1\right) - \frac{\varepsilon^2}{B^2} n\right\}.$$

Using the ratio test for checking the convergence of series, the ratio of two consecutive summands is given by

$$\begin{aligned}
\mathcal{R}_n^{n+1} &:= \frac{4 \exp\left\{\mathcal{VC}\left(\log\left(\frac{2(n+1)}{\mathcal{VC}}\right) + 1\right) - \frac{\varepsilon^2}{B^2}(n+1)\right\}}{4 \exp\left\{\mathcal{VC}\left(\log\left(\frac{2n}{\mathcal{VC}}\right) + 1\right) - \frac{\varepsilon^2}{B^2} n\right\}} \\
&= \frac{\exp\left\{\mathcal{VC}\left(\log\left(\frac{2(n+1)}{\mathcal{VC}}\right)\right)\right\}}{\exp\left\{\mathcal{VC}\left(\log\left(\frac{2n}{\mathcal{VC}}\right)\right)\right\}} \exp\left\{-\frac{\varepsilon^2}{B^2}\right\} \\
&= \frac{\left(\frac{2(n+1)}{\mathcal{VC}}\right)^{\mathcal{VC}}}{\left(\frac{2n}{\mathcal{VC}}\right)^{\mathcal{VC}}} \exp\left\{-\frac{\varepsilon^2}{B^2}\right\} = \underbrace{\left(\frac{n+1}{n}\right)^{\mathcal{VC}}}_{\searrow 1 \text{ as } n \to \infty} \underbrace{\exp\left\{-\frac{\varepsilon^2}{B^2}\right\}}_{<1 \text{ fixed}}.
\end{aligned}$$

Thus, for this ratio $\mathcal{R}_n^{n+1}$, it follows that there exists a number $\tilde{n} \in \mathbb{N}$ so that for all $n > \tilde{n}$ it holds $\mathcal{R}_n^{n+1} < 1$. Therefore, the series converges and almost sure convergence in (18) is established. □

Now it has to be shown that the true regression function can be fitted by logicDT so that the risk minimizing logicDT function asymptotically becomes the true regression function.

Lemma 3 (Each model is possible in logicDT) *Let $\mu : \{0,1\}^p \to \mathcal{Y}$ be a p-dimensional regression function with $\mathcal{Y} \subseteq \mathbb{R}$. Then, μ can be fitted by logicDT, i.e., $\mu \in \mathcal{L}$ with $\mathcal{L}$ being the class of all logicDT models.*

Proof Since μ takes only binary predictors as its input, μ can be expressed as

$$\mu(\boldsymbol{X}) \;=\; g_0 + \sum_{j=1}^{m} g_j \cdot \mathbb{1}\left(X_{k_{j,1}}^{(c)} \wedge \cdots \wedge X_{k_{j,p_j}}^{(c)}\right)$$

for values $g_0, g_j \in \mathcal{Y}$ and distinct conjunctions $C_j(\boldsymbol{X}) := X_{k_{j,1}}^{(c)} \wedge \cdots \wedge X_{k_{j,p_j}}^{(c)}$, where these conjunctions are distinct in the sense that for a given $\boldsymbol{x} \in \{0,1\}^p$ it holds that $\mathbb{1}(C_j(\boldsymbol{x})) = 1$ is true for at most one $j \in \{1, \dots, m\}$. Let $\mathcal{D}_n$ be a noise-free training data set that fully resembles μ, i.e.,

$$\mathcal{D}_n \;\subseteq\; \left\{ (\boldsymbol{x}, y) : \quad \left[y = g_0 + g_j \wedge j \neq 0 \wedge \mathbb{1}(C_i(\boldsymbol{x})) = \mathbb{1}(i = j)\; \forall i \right] \right.$$
$$\left. \vee\; \left[y = g_0 \wedge \mathbb{1}(C_i(\boldsymbol{x})) = 0\; \forall i \right] \right\}$$

with the additional restriction that each conjunction scenario C_j and the null scenario with $C_j = 0$ for all j have to occur at least once in $\mathcal{D}_n$. Using a proper logicDT state, i.e., a set of conjunctions that distinguish between the conjunctions that compose μ, the corresponding fitted logic decision tree assigns the ideal values g_0 or $g_0 + g_j$ to its leaves. Thus, the resulting model is equal to μ. □

Now the lemmata can be assembled for proving Theorem 1.

Proof of Theorem 1 Simulated annealing operates on a finite state space, which is also the case for logicDT. In logicDT, simulated annealing leads with probability 1 to an ideal model on the training data (see Lemma 1), i.e.,

$$\lim_{t \searrow 0} \lim_{q \to \infty} \mathbb{P}(Q_t(q) \in \mathcal{R}_{\text{opt}}) \;=\; 1$$

for a temperature $t \geq 0$, the homogeneous Markov chains Q_t, and the set of optimal states $\mathcal{R}_{\text{opt}}$. More specifically, the stationary distribution $\pi_t = \lim_{q \to \infty} \mathbb{P}(Q_t(q) = \cdot)$ converges for $t \searrow 0$ to a specific distribution on $\mathcal{R}_{\text{opt}}$, namely

$$\lim_{t \searrow 0} \pi_t(s) \;=\; \begin{cases} \frac{|\mathcal{R}_s|}{\sum_{s' \in \mathcal{R}_{\text{opt}}} |\mathcal{R}_{s'}|}, & s \in \mathcal{R}_{\text{opt}} \\ 0, & s \notin \mathcal{R}_{\text{opt}}, \end{cases} \tag{19}$$

where $\mathcal{R}_s$ is the set of neighbor states of state s. Thus, if this final stationary distribution is reached, an optimal model has to be attained due to the finiteness of the state space.

For proving consistency of random forests with respect to the number of observations, Scornet et al. (2015) studied a theoretical random forest with an infinite number of trees due to the pointwise almost sure convergence resulting from the law of large numbers. Similarly, we assume that the convergences in simulated annealing have occurred, and therefore, an empirical-risk-minimizing logicDT model is given due to the stationary distribution in Eq. (19). The CART methodology ensures that, for a given predictor/conjunction setting, the empirical-risk-minimizing decision tree is grown, if the tree is allowed to fully develop, i.e., not using any stopping criteria, since the prediction values are obtained by empirical risk minimization in the respective leaves (Breiman et al., 1984).

Note that this model will be in the set $\mathcal{R}_{\text{opt}}$ of empirical risk minimizing models, if the original predictor model consisting of the input variables $X_1, \dots, X_p$ is included in the considered state space. However, if the true underlying model μ is not a linear function of the individual predictors, the original model $\{\{X_1\}, \dots, \{X_p\}\}$ and equivalent extensions may be excluded from the state space while maintaining consistency. Thus, this theorem shows that logicDT models different from the original CART are consistent as long as the true function exhibits an adequate structure.

Let T_n be the empirical risk minimizer and μ be the true regression function. Applying Lemma 10.1 from Györfi et al. (2002) yields

$$\begin{aligned}\mathbb{E}_{(X,Y)}\left[(\mu(\boldsymbol{x}) - T_n(\boldsymbol{x}))^2\right] \;\leq\; & 2\sup_T \left| \frac{1}{n}\sum_{i=1}^{n}(y_i - T(\boldsymbol{x}_i))^2 - \mathbb{E}_{(X,Y)}\left[(Y - T(\boldsymbol{X}))^2\right] \right| \\ & + \inf_T \mathbb{E}_{(X,Y)}\left[(\mu(\boldsymbol{X}) - T(\boldsymbol{X}))^2\right],\end{aligned} \tag{20}$$

where the supremum and the infimum are determined over all logicDT models T.

Using Lemma 2, ERM consistency is established, i.e.,

$$\begin{aligned}\sup_T \left| R_{\text{emp}}(T) - R_{\text{true}}(T) \right| \;=\; & \sup_T \left| \frac{1}{n}\sum_{i=1}^{n}(y_i - T(\boldsymbol{x}_i))^2 - \mathbb{E}_{(X,Y)}\left[(Y - T(\boldsymbol{X}))^2\right] \right| \\ & \xrightarrow[n\to\infty]{\text{a.s.}} \; 0,\end{aligned}$$

where the almost sure convergence occurs with respect to the training data distribution $\mathbb{P}_{\mathcal{D}_n} = \mathbb{P}_{(X,Y)}^{\otimes n}$. Therefore, the first term on the right-hand side of (20) converges almost surely to zero.

Lemma 3 states that logicDT can lead to every possible regression function μ. Thus, it follows

$$\inf_T \mathbb{E}_{(X,Y)}\left[(\mu(\boldsymbol{X}) - T(\boldsymbol{X}))^2\right] \;=\; \mathbb{E}_{(X,Y)}\left[(\mu(\boldsymbol{X}) - \mu(\boldsymbol{X}))^2\right] \;=\; 0$$

so that the second term on the right-hand side of (20) vanishes.

Hence, in total, we obtain

$$\mathbb{E}_{(X,Y)}\left[(\mu(\boldsymbol{X}) - T_n(\boldsymbol{X}))^2\right] \;\xrightarrow[n\to\infty]{\text{a.s.}}\; 0,$$

which was to be shown. □

Appendix 3: Computational complexity proof

In this appendix, we prove Theorem 2 and Corollary 1 that were stated in Sect. 3.8.

Proof of Theorem 2 Following Algorithm 2, logicDT modifies the current state, creates a tree training data set, and fits and evaluates a decision tree based on this tree training data set to decide if the newly proposed state is accepted in every search iteration. Hence, the computational complexities of these individual steps have to be determined.

State modifications are performed randomly by modifying one variable in the current state. Therefore, the complexity of state modifications is given by $\mathcal{O}(1)$.

Tree training data sets are obtained by computing Boolean conjunctions using the variables in the considered term for each term in the considered state and each training observation (see Sect. 3.2). Since a state contains at most `max_vars` variables, transforming a training data set into a tree training data set amounts to a complexity of $\mathcal{O}(n \cdot \texttt{max\_vars})$.

Decision trees are fitted by recursively screening all p input variables for the best split (see Algorithm 1). This screening amounts to a complexity of $\mathcal{O}(np)$ for n training observations and p input variables and it is performed for at most $\left\lfloor \frac{n}{\texttt{nodesize}} \right\rfloor - 1$ inner nodes (corresponding to the worst-case scenario of an unbalanced tree in which the observations are perfectly divided into leaves of sample size `nodesize`). Thus, the (worst-case) complexity of fitting decision trees is given by $\mathcal{O}(n^2 p/\texttt{nodesize})$. This complexity remains valid for the case in which one additional continuous covariable is included due to univariate linear regression/LDA models being fitted and evaluated using closed-form solutions (i.e., each fitting/evaluation of these univariate regression models amounts to a complexity of $\mathcal{O}(n)$).

Since a maximum of `max_conj` input variables are used for fitting a logic decision tree, the tree fitting (and scoring) complexity in logicDT is given by $\mathcal{O}(n^2 \texttt{max\_conj}/\texttt{nodesize})$. Therefore, using the aforementioned complexities, the computational complexity of logicDT is given by

$$\mathcal{O}\left(M\left[n \cdot \texttt{max\_vars} + n^2 \frac{\texttt{max\_conj}}{\texttt{nodesize}}\right]\right) = \mathcal{O}\left(Mn\left[\texttt{max\_vars} + \texttt{max\_conj} \frac{n}{\texttt{nodesize}}\right]\right),$$

which was to be shown. □

Proof of Corollary 1 The number M of search steps that are conducted in similar simulated-annealing-based search procedures is in the magnitude of $\mathcal{O}(L \log(|S|))$ (Van Laarhoven & Aarts, 1987), where L is the number of iterations performed per Markov chain and S is the search space. Since the search space considered in logicDT consists of sets of possible Boolean conjunctions that include at most `max_conj` conjunctions and at most `max_vars` input variables, the magnitude of this search space is given by

$$|S| \in \mathcal{O}((2p)^{\texttt{max\_vars}} \cdot \texttt{max\_conj}^{\texttt{max\_vars}}).$$

The first factor amounts for all selections of input variables or their negations of size `max_vars`, while the second factor amounts for the number of possibilities to assign the variables to terms. The rationale behind the second factor is assigning each of the `max_vars` variables a number in $\{1, \ldots, \texttt{max\_conj}\}$ that specifies to which term the variable belongs. Hence, it follows that

$$M \in \mathcal{O}(L \cdot \texttt{max\_vars}(\log(p) + \log(\texttt{max\_conj}))).$$

Since, by assumption, the parameters `max_vars` and `max_conj` both scale linearly with p and the parameter `nodesize` is constant, it follows with Theorem 2 that the computational complexity of logicDT is given by

$$\mathcal{O}\big(L \cdot n^2 p^2 \log(p)\big).$$

If it is assumed that the Markov chain length L is fixed, the computational complexity of logicDT becomes

$$\mathcal{O}\big(n^2 p^2 \log(p)\big).$$

The number of neighbor states per state in logicDT is in the magnitude of $\mathcal{O}(\texttt{max\_vars} \cdot p)$, since each variable in the state might be exchanged by another variable. Therefore, if instead the Markov chain length L is chosen in the magnitude of the number of neighbor states per state, the computational complexity of logicDT is given by

$$\mathcal{O}\big(n^2 p^4 \log(p)\big).$$

□

Appendix 4: Additional real data evaluations

In the following, logicDT, bagged logicDT, and the comparable methods are evaluated on 24 real data sets that were also analyzed in Aglin et al. (2020a) and Demirović et al. (2022). These data sets exclusively contain binary input variables and binary outcomes and were obtained from CP4IM[1] that provides (modified) data sets from the UCI Machine Learning Repository[2] that were modified by dichotimizing continuous variables into binary variables.

In Table 3, the dimensions of the considered data sets are summarized. Similar to Sect. 5, each method was applied to each data set 100 times using random splits into training, validation, and test data sets.

Figure 14 shows the predictive performance (as, again, measured by the AUC) of logicDT and the comparable methods in their applications to the 24 additional real data sets. This figure shows that logicDT achieves for most data sets a superior performance compared to conventional decision trees and DL8.5. logicDT seems to be on par with logic regression, since, in most cases, both methods yield similar results and, in the remaining cases, sometimes logicDT and sometimes logic regression induce better performances (see, e.g., the results from the applications to the vehicle and zoo-1 data set).

Ensemble methods that produce less interpretable models such as random forests, gradient boosting, and logic bagging yield better performances compared to logicDT for most data sets. However, when also considered logicDT in an ensemble framework, i.e., when considering bagged logicDT, then the performances are on a similar level as the other ensemble methods.

[1] CP4IM: https://dtai.cs.kuleuven.be/CP4IM/.

[2] UCI Machine Learning Repository: https://archive.ics.uci.edu.

Table 3 Dimensions of the 24 real data sets used for evaluating logicDT and the comparable methods

Data set	n	p	n_1	n_0
anneal	812	93	625	187
audiology	216	148	57	159
australian-credit	653	125	357	296
breast-wisconsin	683	120	444	239
diabetes	768	112	500	268
german-credit	1000	112	700	300
heart-cleveland	296	95	160	136
hepatitis	137	68	111	26
hypothyroid	3247	88	2970	277
ionosphere	351	445	225	126
kr-vs-kp	3196	73	1669	1527
letter	20,000	224	813	19,187
lymph	148	68	81	67
mushroom	8124	119	4208	3916
pendigits	7494	216	780	6714
primary-tumor	336	31	82	254
segment	2310	235	330	1980
soybean	630	50	92	538
splice-1	3190	287	1655	1535
tic-tac-toe	958	27	626	332
vehicle	846	252	218	628
vote	435	48	267	168
yeast	1484	89	463	1021
zoo-1	101	36	41	60

n denotes the sample size and p the number of input variables in the respective data set. n_1 and n_0 denote the numbers of observations with $Y = 1$ and $Y = 0$, respectively, since binary outcomes are considered

Appendix 5: Computation times

For the simulation study conducted in Sect. 5.1.1 and the application to the SALIA data conducted in Sect. 5.2, model fitting and prediction times were recorded. The calculations were performed using an Intel Xeon Gold 6346 CPU running on 3.6GHz. For the time measurement, no parallel computing was performed to reflect realistic single model evaluation times.

In Table 4, the mean model fitting and prediction times over ten replications is summarized. logicDT seems to be faster than logic regression, which also employs a stochastic search algorithm. In the application to the SALIA data, a more complex setting consisting of five terms was identified for logicDT compared to three terms for logic regression, which might explain the higher fitting time of logicDT compared to logic regression in the real data application.

Due to the computationally intensive global search, logicDT takes more time than comparable methods that employ greedy fitting algorithms such as conventional decision trees, random forests, gradient boosting, and MOB. Nonetheless, logicDT seems to

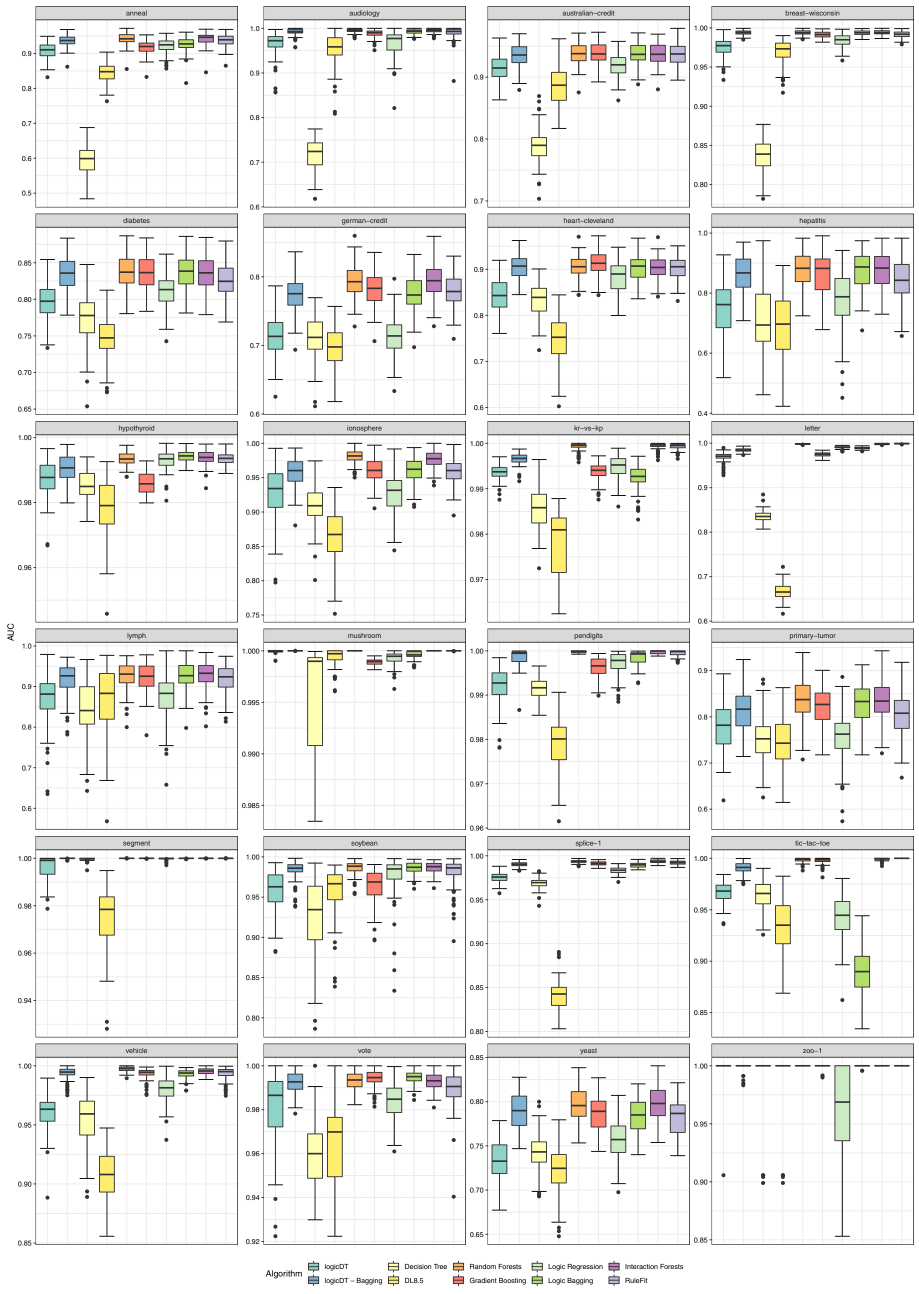


Fig. 14 Predictive performance of logicDT and the comparable methods in the evaluation of 24 real data sets

Table 4 Mean model evaluation times in seconds for the simulation study conducted in Sect. 5.1.1 and the real data application conducted in Sect. 5.2

Algorithm	Simulation scenario/study				
	Binary No E	Binary E	Continuous No E	Continuous E	SALIA
logicDT	29.334	87.615	12.826	33.414	38.727
logicDT–Bagging	260.063	307.279	82.151	770.853	1960.524
Decision Tree	0.184	0.183	0.184	0.179	0.186
DL8.5	2.907	3.571	–	–	700.399
Random Forests	5.704	6.440	6.875	7.133	1.980
Gradient Boosting	3.012	2.901	3.440	3.559	2.434
Logic Regression	27.004	206.816	33.671	22.710	15.803
Logic Bagging	82.584	40.730	57.809	63.202	575.047
MOB	–	0.513	–	0.479	–
Interaction Forests	82.682	344.738	322.515	501.945	40.416
RuleFit	77.568	96.647	71.394	78.370	92.420

The first line of the simulation scenario name corresponds to the considered outcome type (binary or continuous) and the second line corresponds to whether a continuous environmental covariable was incorporated (no E or E)

be faster than RuleFit. DL8.5 was faster than logicDT in the simulation study. For the real data application, DL8.5 was substantially slower than logicDT.

Bagged logicDT models take more time to be evaluated than bagged logic regression models. This is, in particular, due to the fact that the hyperparameter optimization for logic bagging identified `ntrees` = 3 with `nleaves` = 3 as the best setting for the simulation scenario with a binary outcome and an environmental covariable, i.e., a linear model involving three predictors, while bagged logicDT fits trees of depth of up to three in every greedy search step. This explanation also holds true for the other three scenarios and the real data application, since the hyperparameter optimization also yielded simpler settings for logic bagging compared to bagged logicDT.

Interaction forests are similarly fast as bagged logicDT and logic bagging in the simulation study. In the application to the SALIA data, interaction forests are comparably fast, since the hyperparameter optimization yielded for the number of randomly drawn input variable pairs per split `npairs` = 4, which is smaller than in the considered simulation study scenarios.

Unsurprisingly, for most methods, the computation time increases when also considering a continuous (environmental) covariable in comparison to not including a continuous (environmental) covariable. For some methods, this trend does not seem to be true, for example for logic regression, since the mean computation decreases when additionally considering a continuous covariable for a continuous outcome. However, this phenomenon is presumably caused by the identified hyperparameter setting, which is `ntrees` = 4 with `nleaves` = 8 for the continuous outcome scenario without a continuous covariable, corresponding to a rather complex model, and `ntrees` = 2 with `nleaves` = 3 for the continuous outcome scenario including a continuous covariable, corresponding to a rather simple model.

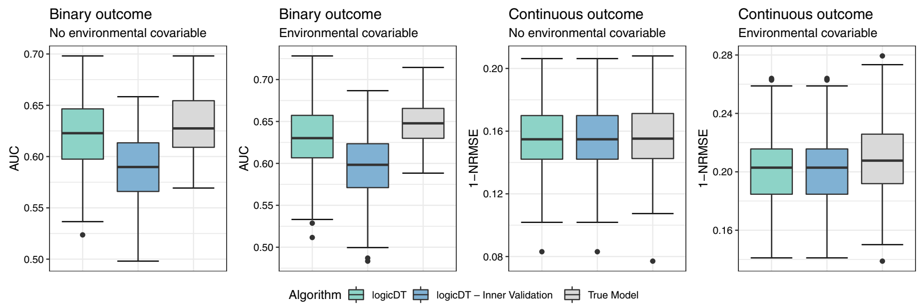


Fig. 15 Predictive performances of logicDT, logicDT using inner validation, and the true underlying model in the simulation study considering four different scenarios. The performance for binary outcomes is measured by the AUC and the performance for continuous outcomes is measured by the complement of the NRMSE (normalized root mean squared error)

Appendix 6: Inner validation

An idea to further robustify logicDT against overfitting might be to separate the decision tree fitting and evaluation steps in the search procedure by splitting the available training data into independent data sets for these two steps. We refer to this approach as *inner validation*, due to validating the states on independent validation data and the fitting procedure being nested in an outer validation that evaluates the performance of resulting logicDT models for tuning hyperparameters (see Sect. 3.6). This approach is similar to a nested cross-validation, which is, however, typically used for estimating unbiased prediction errors (see, e.g., Varma & Simon, 2006).

The trained logicDT model should not be heavily depending on the data split used such that a k-fold cross-validation approach is employed that randomly splits the training data into k approximately equally sized data sets $\mathcal{D}_1, \dots, \mathcal{D}_k$. For every $j \in \{1, \dots, k\}$, $k-1$ of these data sets $\mathcal{D}_{j'}$ ($j' \in \{1, \dots, k\} \setminus j$) are combined to one data set and used for training the decision trees (Line 9 in Algorithm 2) and the remaining data set $\mathcal{D}_j$ is used for computing the score (Line 10 in Algorithm 2). The total score of the state used to guide the search is then obtained by averaging the k scores.

In Fig. 15, the predictive performances of logicDT are summarized that were obtained using the aforementioned inner validation approach with 5-fold cross-validation in the simulation study presented in Sect. 5.1.1. For the binary outcome scenarios, the performance is worse compared to standard logicDT. For the continuous outcome scenarios, the performance is identical.

The performance loss can presumably be explained by the need to further split the available training data so that both the tree training step and the score computation step have to use less observations as opposed to standard logicDT. Moreover, the inner validation also leads to an increased computational burden due to fitting k trees in comparison to fitting a single tree in each search iteration. Therefore, the outer validation for hyperparameter optimization seems to be sufficient to balance the amount of underfitting and overfitting.

Acknowledgements The authors would like to thank Gaël Aglin for insightful discussions on the proper application of DL8.5. Computational infrastructure and support were provided by the Centre for Information and Media Technology at the Heinrich Heine University Düsseldorf.

Author Contributions ML and HS developed logicDT and the interaction VIM and designed the simulation study. ML and TS conceived the analyses of the real data application. The simulation study and the real data evaluations were conducted by ML. ML was the major contributor in writing the manuscript. All authors read and approved the final manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. This work has been supported by the Research Training Group "Biostatistical Methods for High-Dimensional Data in Toxicology" (RTG 2624, Project R3) funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation—Project Number 427806116). The SALIA follow-up study 2007–2010 was funded by the German Statutory Accident Insurance (DGUV) Grant No: 617.0-FP266 and the European Community's Seventh Framework Programme (FP 7/2007-2011) under the Grant Agreement Number 211250.

Data availability The simulated data sets are available upon request. The modified real data sets from the UCI Machine Learning Repository (https://archive.ics.uci.edu) that are analyzed in Appendix 4 have been downloaded in May 2023 from https://github.com/aia-uclouvain/pydl8.5.

Code availability The proposed methods are implemented and publicly available in the `R` package logicDT on CRAN (Lau, 2023).

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Consent for publication Not applicable.

Ethics approval and consent to participate The SALIA cohort study was conducted in accordance to the declaration of Helsinki and has been approved by the Ethics Committees of the Ruhr-University Bochum and the Heinrich Heine University Düsseldorf. We received written informed consent from all participants.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Aarts, E., & Van Laarhoven, P. (1985). Statistical cooling: A general approach to combinatorial optimization problems. *Philips Journal of Research, 40*(4), 193–226.

Aglin, G., Nijssen, S., & Schaus, P. (2020). Learning optimal decision trees using caching branch-and-bound search. In *Proceedings of the AAAI conference on artificial intelligence,* (Vol. 34, pp. 3146–3153).

Aglin, G., Nijssen, S., & Schaus, P. (2020b). PyDL8.5: A library for learning optimal decision trees. In *Proceedings of the twenty-ninth international joint conference on artificial intelligence, IJCAI-20* (pp. 5222–5224). International Joint Conferences on Artificial Intelligence Organization.

Bellinger, C., Mohomed Jabbar, M. S., Zaïane, O., & Osornio-Vargas, A. (2017). A systematic review of data mining and machine learning for air pollution epidemiology. *BMC Public Health, 17*, 907. https://doi.org/10.1186/s12889-017-4914-3

Bénard, C., Biau, G., da Veiga, S., & Scornet, E. (2021). Interpretable random forests via rule extraction. In *Proceedings of the 24th international conference on artificial intelligence and statistics*, Volume 130 of *Proceedings of machine learning research* (pp. 937–945). PMLR.

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological), 57*(1), 289–300. https://doi.org/10.1111/j.2517-6161.1995.tb02031.x

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11 (pp. 2546–2554). Curran Associates Inc.

Bertsimas, D., & Dunn, J. (2017). Optimal classification trees. *Machine Learning, 106*, 1039–1082. https://doi.org/10.1007/s10994-017-5633-9

Blockeel, H., & De Raedt, L. (1998). Top-down induction of first-order logical decision trees. *Artificial Intelligence, 101*(1), 285–297. https://doi.org/10.1016/S0004-3702(98)00034-4

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*(2), 123–140. https://doi.org/10.1007/BF00058655

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Breiman, L., Friedman, J. H., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC Press.

Bureau, A., Dupuis, J., Falls, K., Lunetta, K. L., Hayward, B., Keith, T. P., & Van Eerdewegh, P. (2005). Identifying SNPs predictive of phenotype using random forests. *Genetic Epidemiology, 28*(2), 171–182. https://doi.org/10.1002/gepi.20041

Carrizosa, E., Molero-Río, C., & Romero Morales, D. (2021). Mathematical optimization in classification and regression trees. *TOP, 29*, 5–33. https://doi.org/10.1007/s11750-021-00594-1

Che, R., & Motsinger-Reif, A. (2013). Evaluation of genetic risk score models in the presence of interaction and linkage disequilibrium. *Frontiers in Genetics, 4*, 138. https://doi.org/10.3389/fgene.2013.00138

Chen, C. C., Schwender, H., Keith, J., Nunkesser, R., Mengersen, K., & Macrossan, P. (2011). Methods for identifying SNP interactions: A review on variations of logic regression, random forest and Bayesian logistic regression. *IEEE/ACM Transactions on Computational Biology and Bioinformatics, 8*(6), 1580–1591. https://doi.org/10.1109/TCBB.2011.46

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, KDD '16, New York, NY, USA (pp. 785–794). Association for Computing Machinery.

Clarke, A., & Vyse, T. J. (2009). Genetics of rheumatic disease. *Arthritis Research & Therapy, 11*(5), 248. https://doi.org/10.1186/ar2781

Demirović, E., Lukina, A., Hebrard, E., Chan, J., Bailey, J., Leckie, C., Ramamohanarao, K., & Stuckey, P. J. (2022). MurTree: optimal decision trees via dynamic programming and search. *Journal of Machine Learning Research, 23*(26), 1–47.

Dudbridge, F., & Newcombe, P. J. (2015). Accuracy of gene scores when pruning markers by linkage disequilibrium. *Human Heredity, 80*(4), 178–186. https://doi.org/10.1159/000446581

Fokkema, M. (2020). Fitting prediction rule ensembles with R package pre. *Journal of Statistical Software, 92*(12), 1–30. https://doi.org/10.18637/jss.v092.i12

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics, 29*(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

Friedman, J. H., & Popescu, B. E. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics, 2*(3), 916–954. https://doi.org/10.1214/07-AOAS148

Fujimoto, K., Kojadinovic, I., & Marichal, J. L. (2006). Axiomatic characterizations of probabilistic and cardinal-probabilistic interaction indices. *Games and Economic Behavior, 55*(1), 72–99. https://doi.org/10.1016/j.geb.2005.03.002

Györfi, L., Kohler, M., Krzyżak, A., & Walk, H. (2002). *A distribution-free theory of nonparametric regression*. Springer.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction*. Springer.

Ho, D. S. W., Schierding, W., Wake, M., Saffery, R., & O'Sullivan, J. (2019). Machine learning SNP based prediction for precision medicine. *Frontiers in Genetics*. https://doi.org/10.3389/fgene.2019.00267

Hornung, R. (2022). Diversity forests: Using split sampling to enable innovative complex split procedures in random forests. *SN Computer Science, 3*(1), 1–16. https://doi.org/10.1007/s42979-021-00920-1

Hornung, R., & Boulesteix, A. L. (2022). Interaction forests: Identifying and exploiting interpretable quantitative and qualitative interaction effects. *Computational Statistics & Data Analysis, 171*, 107460. https://doi.org/10.1016/j.csda.2022.107460

Huang, M., Romeo, F., & Sangiovanni-Vincentelli, A. (1986). An efficient general cooling schedule for simulated annealing. In *Proceedings of the IEEE international conference on computer-aided design*, Santa Clara, California, USA (pp. 381–384). IEEE Computer Society.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Kooperberg, C., & Ruczinski, I. (2022). LogicReg: Logic regression. R Package Version 1.6.5.

Krämer, U., Herder, C., Sugiri, D., Strassburger, K., Schikowski, T., Ranft, U., & Rathmann, W. (2010). Traffic-related air pollution and incident type 2 diabetes: Results from the salia cohort study. *Environmental Health Perspectives, 118*(9), 1273–1279. https://doi.org/10.1289/ehp.0901689

Van Laarhoven, P., & Aarts, E. (1987). *Simulated annealing: Theory and applications*. Springer.

Lau, M. (2023). logicDT: Identifying interactions between binary predictors. R Package Version 1.0.3.

Lau, M., Wigmann, C., Kress, S., Schikowski, T., & Schwender, H. (2022). Evaluation of tree-based statistical learning methods for constructing genetic risk scores. *BMC Bioinformatics, 23*, 97. https://doi.org/10.1186/s12859-022-04634-w

Li, R. H., & Belford, G. G. (2002). Instability of decision tree classification algorithms. In *Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining*, New York, NY, USA (pp. 570–575). Association for Computing Machinery.

Louppe, G. (2014). Understanding random forests: From theory to practice. Dissertation, University of Liège, Department of Electrical Engineering & Computer Science. arXiv:1407.7502.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., & Lee, S. I. (2020). From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence, 2*, 56–67. https://doi.org/10.1038/s42256-019-0138-9

Malley, J. D., Kruppa, J., Dasgupta, A., Malley, K. G., & Ziegler, A. (2012). Probability machines: Consistent probability estimation using nonparametric learning machines. *Methods of Information in Medicine, 51*(1), 74–81. https://doi.org/10.3414/ME00-01-0052

Meinshausen, N. (2010). Node harvest. *The Annals of Applied Statistics, 4*(4), 2049–2072. https://doi.org/10.1214/10-AOAS367

Mentch, L., & Hooker, G. (2016). Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *Journal of Machine Learning Research, 17*(26), 1–41.

Menze, B. H., Kelm, B. M., Splitthoff, D. N., Koethe, U., & Hamprecht, F. A. (2011). On oblique random forests. In *Proceedings of the joint European conference on machine learning and knowledge discovery in databases*, Berlin, Heidelberg (pp. 453–469). Springer.

Murthy, S. K., Kasif, S., & Salzberg, S. (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research, 2*, 1–32. https://doi.org/10.1613/jair.63

Murthy, S. K., & Salzberg, S. (1995). Decision tree induction: How effective is the greedy heuristic? In *Proceedings of the first international conference on knowledge discovery and data mining*, KDD'95 (pp. 222–227). AAAI Press.

Nijssen, S., & Fromont, E. (2010). Optimal constraint-based decision tree induction from itemset lattices. *Data Mining and Knowledge Discovery, 21*, 9–51. https://doi.org/10.1007/s10618-010-0174-x

Ottman, R. (1996). Gene-environment interaction: Definitions and study design. *Preventive Medicine, 25*(6), 764–770. https://doi.org/10.1006/pmed.1996.0117

Provost, F., & Domingos, P. (2003). Tree Induction for probability-based ranking. *Machine Learning, 52*(3), 199–215. https://doi.org/10.1023/A:1024099825458

Purcell, S., Neale, B., Todd-Brown, K., Thomas, L., Ferreira, M. A., Bender, D., Maller, J., Sklar, P., De Bakker, P. I., Daly, M. J., & Pak, C. S. (2007). PLINK: A tool set for whole-genome association and population-based linkage analyses. *The American Journal of Human Genetics, 81*(3), 559–575. https://doi.org/10.1086/519795

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers Inc.

R Core Team. (2022). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing.

Ruczinski, I., Kooperberg, C., & LeBlanc, M. (2003). Logic regression. *Journal of Computational and Graphical Statistics, 12*(3), 475–511. https://doi.org/10.1198/1061860032238

Ruczinski, I., Kooperberg, C., & LeBlanc, M. (2004). Exploring interactions in high-dimensional genomic data: An overview of logic regression, with applications. *Journal of Multivariate Analysis, 90*(1), 178–195. https://doi.org/10.1016/j.jmva.2004.02.010

Rusch, T., & Zeileis, A. (2013). Gaining insight with recursive partitioning of generalized linear models. *Journal of Statistical Computation and Simulation, 83*(7), 1301–1315. https://doi.org/10.1080/00949655.2012.658804

Schikowski, T., Sugiri, D., Ranft, U., Gehring, U., Heinrich, J., Wichmann, H. E., & Krämer, U. (2005). Long-term air pollution exposure and living close to busy roads are associated with COPD in women. *Respiratory Research, 6*, 152. https://doi.org/10.1186/1465-9921-6-152

Schwender, H., & Ickstadt, K. (2007). Identification of SNP interactions using logic regression. *Biostatistics, 9*(1), 187–198. https://doi.org/10.1093/biostatistics/kxm024

Scornet, E., Biau, G., & Vert, J. P. (2015). Consistency of random forests. *The Annals of Statistics, 43*(4), 1716–1741. https://doi.org/10.1214/15-aos1321

So, H. C., & Sham, P. C. (2017). Improving polygenic risk prediction from summary statistics by an empirical Bayes approach. *Scientific Reports, 7*, 41262. https://doi.org/10.1038/srep41262

Sorokina, D., Caruana, R., Riedewald, M., & Fink, D. (2008). Detecting statistical interactions with additive groves of trees. In *Proceedings of the 25th international conference on machine learning*, ICML '08, New York, NY, USA (pp. 1000–1007). Association for Computing Machinery.

Tang, C., Garreau, D., & von Luxburg, U. (2018). When do random forests fail? In *Proceedings of the 32nd international conference on neural information processing systems*, NIPS'18, Montréal, Canada (pp. 2987–2997).

Therneau, T., & Atkinson, B. (2019). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-15.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological), 58*(1), 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Tomita, T. M., Browne, J., Shen, C., Chung, J., Patsolic, J. L., Falk, B., Priebe, C. E., Yim, J., Burns, R., Maggioni, M., & Vogelstein, J. T. (2020). Sparse projection oblique randomer forests. *Journal of Machine Learning Research, 21*(104), 1–39.

Triki, E., Collette, Y., & Siarry, P. (2005). A theoretical study on the behavior of simulated annealing leading to a new cooling schedule. *European Journal of Operational Research, 166*(1), 77–92. https://doi.org/10.1016/j.ejor.2004.03.035

Vapnik, V. N. (1998). *Statistical learning theory*. Wiley-Interscience.

Vapnik, V. N. (2000). *The nature of statistical learning theory*. Springer.

Varma, S., & Simon, R. (2006). Bias in error estimation when using cross-validation for model selection. *BMC Bioinformatics, 7*, 91. https://doi.org/10.1186/1471-2105-7-91

Watson, D. S., & Wright, M. N. (2021). Testing conditional independence in supervised learning algorithms. *Machine Learning, 110*, 2107–2129. https://doi.org/10.1007/s10994-021-06030-6

Wilks, S. S. (1938). The large-sample distribution of the likelihood ratio for testing composite hypotheses. *The Annals of Mathematical Statistics, 9*(1), 60–62. https://doi.org/10.1214/aoms/1177732360

Wilson, S. (2021). ParBayesianOptimization: Parallel Bayesian optimization of hyperparameters. R Package Version 1.2.4.

Winham, S. J., Colby, C. L., Freimuth, R. R., Wang, X., de Andrade, M., Huebner, M., & Biernacka, J. M. (2012). SNP interaction detection with random forests in high-dimensional genetic data. *BMC Bioinformatics, 13*, 164. https://doi.org/10.1186/1471-2105-13-164

Wright, M. N., & Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software, 77*(1), 1–17. https://doi.org/10.18637/jss.v077.i01

Wright, M. N., Ziegler, A., & König, I. R. (2016). Do little interactions get lost in dark random forests? *BMC Bioinformatics, 17*, 145. https://doi.org/10.1186/s12859-016-0995-8

Yang, B. B., Shen, S. Q., & Gao, W. (2019). Weighted oblique decision trees. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 33, pp. 5621–5627).

Zeileis, A., Hothorn, T., & Hornik, K. (2008). Model-based recursive partitioning. *Journal of Computational and Graphical Statistics, 17*(2), 492–514. https://doi.org/10.1198/106186008X319331

Zhi, S., Li, Q., Yasui, Y., Edge, T., Topp, E., & Neumann, N. F. (2015). Assessing host-specificity of *Escherichia coli* using a supervised learning logic-regression-based analysis of single nucleotide polymorphisms in intergenic regions. *Molecular Phylogenetics and Evolution, 92*, 72–81. https://doi.org/10.1016/j.ympev.2015.06.007

Zhu, H., Murali, P., Phan, D., Nguyen, L., & Kalagnanam, J. (2020). A scalable MIP-based method for learning optimal multivariate decision trees. In *Advances in neural information processing systems* (Vol. 33, pp. 1771–1781). Curran Associates, Inc.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.