machine learning & knowledge extraction

MDPI

Article

A Meta Algorithm for Interpretable Ensemble Learning: The League of Experts

Richard Vogel [1,*], Tobias Schlosser [2,*], Robert Manthey [1], Marc Ritter [1], Matthias Vodel [1], Maximilian Eibl [3] and Kristan Alexander Schneider [4]

[1] Media Informatics, University of Applied Sciences Mittweida, 09648 Mittweida, Germany; robert.manthey@hs-mittweida.de (R.M.); marc.ritter@hs-mittweida.de (M.R.); vodel@hs-mittweida.de (M.V.)
[2] Media Computing, Chemnitz University of Technology, 09107 Chemnitz, Germany
[3] Media Informatics, Chemnitz University of Technology, 09107 Chemnitz, Germany; maximilian.eibl@cs.tu-chemnitz.de
[4] Modeling and Simulation, University of Applied Sciences Mittweida, 09648 Mittweida, Germany; kristan.schneider@hs-mittweida.de
* Correspondence: richard.vogel@hs-mittweida.de (R.V.); tobias.schlosser@cs.tu-chemnitz.de (T.S.)

Abstract: Background. The importance of explainable artificial intelligence and machine learning (*XAI/XML*) is increasingly being recognized, aiming to understand how information contributes to decisions, the method's bias, or sensitivity to data pathologies. Efforts are often directed to post hoc explanations of black box models. These approaches add additional sources for errors without resolving their shortcomings. Less effort is directed into the design of intrinsically interpretable approaches. **Methods.** We introduce an intrinsically interpretable methodology motivated by ensemble learning: the *League of Experts (LoE)* model. We establish the theoretical framework first and then deduce a modular meta algorithm. In our description, we focus primarily on classification problems. However, LoE applies equally to regression problems. Specific to classification problems, we employ classical decision trees as classifier ensembles as a particular instance. This choice facilitates the derivation of human-understandable decision rules for the underlying classification problem, which results in a derived rule learning system denoted as *RuleLoE*. **Results.** In addition to 12 KEEL classification datasets, we employ two standard datasets from particularly relevant domains—medicine and finance—to illustrate the LoE algorithm. The performance of LoE with respect to its accuracy and rule coverage is comparable to common state-of-the-art classification methods. Moreover, LoE delivers a clearly understandable set of decision rules with adjustable complexity, describing the classification problem. **Conclusions.** LoE is a reliable method for classification and regression problems with an accuracy that seems to be appropriate for situations in which underlying causalities are in the center of interest rather than just accurate predictions or classifications.

Keywords: ensemble learning; multiagent systems; explainability; glass box models

Citation: Vogel, R.; Schlosser, T.; Manthey, R.; Ritter, M.; Vodel, M.; Eibl, M.; Schneider, K.A. A Meta Algorithm for Interpretable Ensemble Learning: The League of Experts. *Mach. Learn. Knowl. Extr.* **2024**, *6*, 800–826. https://doi.org/10.3390/make6020038

Academic Editor: Luca Longo

Received: 29 February 2024
Revised: 2 April 2024
Accepted: 3 April 2024
Published: 9 April 2024

1. Introduction and Motivation

Machine learning (ML) is an integral part of many products and activities in our everyday life, including developments in autonomous driving [1], health care [2], and law enforcement [3]. Regulations regarding automated algorithmic decision making—"the right for explanation[s]" [4]—and other judicial reasons [5] emphasize the need for general accountability [6] of ML-based systems, particularly of black box models. Black box models are optimized for performance by adding complexity to an extent that renders them infeasible to interpret. Current attempts try to overcome the drawbacks of lacking explainability by adding external components distilling human-understandable information within additional layers of complexity (post hoc methods). These additional layers, however, can only be an approximation of the actual decision-making process at the risk of not being faithful [7]. Explainable artificial intelligence as well as machine learning (*XAI/XML*)

is increasingly being recognized [8–11]. While different application areas highlight the emerging trends in XAI and XML [12–14], performance explainability trade-offs are also within the focus of current research [15,16].

1.1. Contribution of This Work

In this contribution, the *League of Experts (LoE)*, a novel and transparent machine learning model and meta algorithm based on the idea of ensemble learning [17] (p. 605), is introduced. LoE is a variant of dynamic classifier selection (DCS) methods [18] (see Section 2.7). However, in comparison, LoE trains its ensemble as part of the involved selection and assignment method, unlike other DCS methods that often overproduce a rather large set of classifiers using static methods such as bagging or boosting [19]. So far, this is, to our knowledge, unexplored in the context of DCS [18]. LoE is a construction framework, allowing for modifications with the goal of leveraging explainability. It enables the design of user interfaces to directly interfere with the model and its components. However, this work will not dive into concrete user interactions, but will instead mention possible interactions where appropriate, for which LoE allows users to manually adjust the selection process and the ensemble members. LoE allows users to interactively explore trade-offs between model performance and complexity. In this contribution, we exemplify and evaluate LoE using two datasets. We particularly focus on controlling the amount of complexity involved in explaining the learned model instances (experiments in Section 4.1). Furthermore, we also introduce a specific solution to instantiate LoE, which enables the transformation of LoE into an almost equivalent rule set learner (section 3.5). In order to reduce the complexity while jointly improving the approximation to LoE, we further explore ways to adapt the training procedure.

1.2. Section Overview

The following sections introduce our methodology in Section 2 and LoE's implementation in Section 3. The main idea of LoE is motivated in Section 3, followed by a concise definition and description of its basic training and inference procedures. Section 3.2 shows how to explain a trained LoE model using decision rules. The proposed methods also allow the reduction in the length of these decision rules, which enables the reduction in their complexity, which in turn facilitates interpretations made by human users (Section 3.3.1). Subsequently, we present a method to extract an analogous rule learner from the derived explanations in Section 3.5, called *RuleLoE*. Finally, Section 4 provides an evaluation regarding feature space reduction and rule set generation given the example of two prominent datasets from medicine and finance.

2. Materials and Methods

In the following, we introduce the *League of Experts (LoE)* algorithm with its motivation in terms of explainability and accuracy. For this purpose, we provide some background information and taxonomy of currently and previously developed and investigated approaches within XAI and XML. For this purpose, Sections 2.1–2.7 provide insights into the use of black and glass box models (Sections 2.1 and 2.2), the principles on rule learning and explanations (Sections 2.3 and 2.4), decision trees and their use with surrogate models (Sections 2.5 and 2.6), and the fundamentals of ensemble learning and dynamic selection (Section 2.7).

2.1. Black Box Models

Computational advances facilitate more powerful but also more complex models such as black box models [20–23]. Feed forward artificial neural networks (ANNs) [24], e.g., for image classification, possess about 5 to 155×10^6 trainable parameters while performing up to 8×10^{10} computational operations for a single prediction [25]. Since an ANN consists of a graph of operations, it is in theory possible to follow along the performed calculations in a stepwise fashion. However, in practice, such enormous complexity is not cognitively

comprehensible by a human. Even other popular methods involving much fewer parameters, e.g., support vector machines (SVMs) [26] or ensemble methods such as boosting (i.e., AdaBoost [27]) and random forests [28], cannot be easily interpreted. These approaches are inappropriate for applications requiring a good understanding of the decision-making process. For ensemble learning, this holds true even if the ensemble's members themselves are easy to interpret, e.g., with decision trees [29]. However, the decision boundary of the whole ensemble remains highly complex as the final prediction consists of a fusion of all members.

2.2. Glass Box Models

In contrast, glass box models [22,30,31], also known as white box models, are characterized by transparent states, allowing for deriving explanations regarding their decision boundaries. Such models include decision trees [17] (p. 305), (logistic) regression [17] (p. 119), and neighborhood models describing decisions by the similarity of related data points, e.g., k-nearest neighbors (k-NN) [32]. However, adding too much complexity jeopardizes practical interpretability, e.g., too many levels in decision trees or too many dependencies in Bayesian networks [33]. Importantly, this is due to the cognitive limits of the users and not the decision-making process itself.

2.3. Rule Learning

A particular example of glass box models is decision rules that conveniently present knowledge to users as logical patterns. They are realized either unordered (i.e., independent) as rule sets or ordered as rule lists. The latter are more complicated to interpret [34] as each rule's applicability depends on the negations of its predecessors (i.e., the previous rules have to *not* apply). Decision trees have representations as rule sets, where each path in the tree becomes a single rule. For example, the rules shown in Figure 1 are as follows: if $(\texttt{sex is female} \wedge \texttt{age} > 40 \wedge \texttt{weight} > 80) \vee (\texttt{sex is female} \wedge \texttt{age} \leq 40)$, predict `low-risk group`; otherwise, predict `high-risk group`. Unlike decision trees, most rule learners do not produce rule sets with full coverage and no sample space overlap—several or no rules are applicable to data points (or queries). The former is resolved by determining a resolution order, the latter by adding an additional fallback rule. Both cases must be addressed when transforming LoE into a rule learner (see Section 3.5).

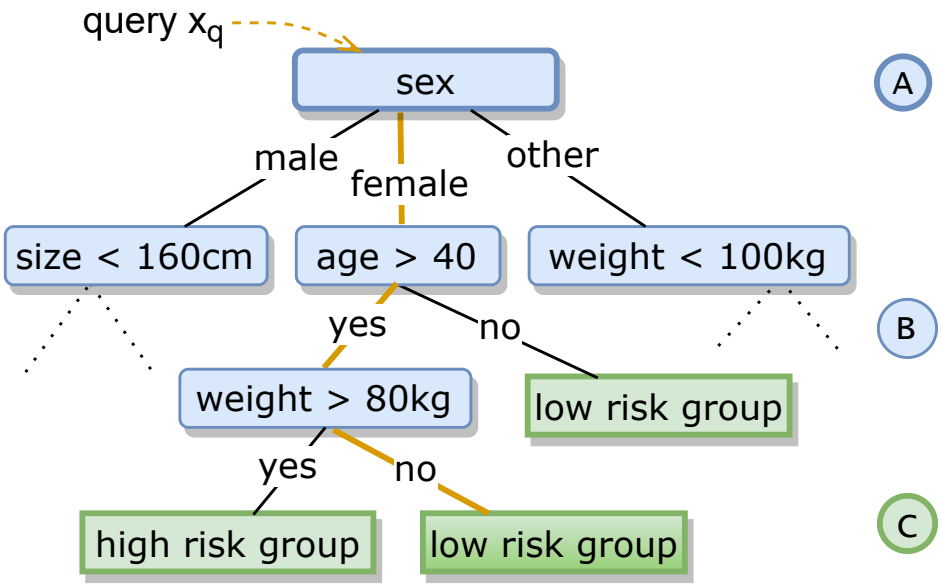

Figure 1. (Partial) decision tree example with (**A**–**C**) being the root, inner nodes, and leaves, respectively. An example query with corresponding trace is highlighted in orange.

Rule learning systems, including CN2 [35], RIPPER [36], Bayesian rule sets [37], Boolean decision rules via column generation (BRCG) [38], and interpretable decision sets [34], derive rules from directly optimizing a target function, e.g., by using integer programming to optimize specific criteria such as classification performance, coverage, or simplicity. The optimal solution is often intractable due to the exponential amount of possible clauses, resulting in the NP-complete set cover problem.

2.4. Explanations

In the context of XAI, "explanation" refers to human-understandable information in a *local* or *global* sense, justifying (i) the process generating a query-dependent output of the method (e.g., the classification result) or (ii) the state of the model as a whole. Local explanations can be misleading about a model's global state, whereas query-dependent properties might not apply in the global context, i.e., for arbitrary queries.

Post hoc explainers are popular methods. They add methods and visual components aiming to explain the decision process of an already-trained model. Post hoc methods can be (i) model agnostic (not tied to a specific model class), e.g., LIME [39], Anchors [40], Shapley values [41], or visualizations such as partial dependence plots (PDPs) or individual conditional expectation plots (ICE) [42] (Chapters 5.1 and 5.2), or (ii) model specific, e.g., decision-tree-based SHAP variants (TreeSHAP) [43] or explainers specific to neural networks [44].

However, the drawbacks of post hoc methods are that (i) external components add complexity to ML pipelines, and (ii) users must have confidence in the underlying ML model and the explanation method. Both can be wrong, inaccurate, or only applicable in certain unobserved circumstances. Importantly, users cannot notice the latter. Glass box models, which are algorithmically transparent, do not require post hoc approaches, as do more generally global simulatable models that are characterized as being simple enough to be comprehensible in reasonable time [45].

Simulatable models include small decision trees, small rule sets [46], (generalized) linear models with a small amount of model parameters (after applying shrinkage methods, e.g., Lasso), or instance-based learners with explainable features (e.g., k-NN [32]). Additionally, the features describing entities, such as data points, must be reasonably comprehensible. Models satisfying those requirements are strongly limited, typically weak in performance, and limited in generalization properties [47]. Notably, the definition of simulatable is somewhat fuzzy as it depends on individual users' abilities.

2.5. Decision Trees

Decision trees [48] directly display their reasoning processes due to their graphical structure (Figure 1). They typically focus only on a subset of relevant attributes or features [49], rendering them cognitively less demanding. The graphical structure of a decision tree allows the user to investigate details of a given reasoning process, which also enables the investigation of alternative paths, i.e., by assuming that an attribute has a different value to observe how this would influence the decision-making process. That allows for judging for fairness and relevance of the decisions made.

2.6. Surrogate Models

Surrogate (or proxy) models [50] attempt to explain the decisions-making process of a complex black box model $f_c(\cdot)$ by imitating the black box model's output profile on a dataset $\mathcal{L}$ by utilizing a simpler glass box model $f_g(\cdot)$ [42] (Chapter 5.6) (e.g., by using a decision tree to approximate an ANN). The approximation $f_c \sim f_g$ can hold globally (global surrogate), i.e., on $\mathcal{L}$, or locally (local surrogate), i.e., in a subset of $\mathcal{L}$. For instance, the LIME method [40] is a local surrogate.

Generally, it is unclear whether the surrogate f_g captures the same causalities as the explained model of f_c. Several alternative surrogates might perform equally well, which can be explained by the Rashomon effect [51], describing circumstances under which

multiple alternative explanations yield the same result. The latter is less problematic when replacing the complex model by a sufficiently accurate global surrogate. However, for local surrogates, the problems amplify, particularly if users are inclined to induce global decision making from local explanations. Although the mentioned problems do not vanish, they can be controlled by directing the training procedure (see Section 3.3).

2.7. Ensemble Learning and Dynamic Selection

Ensemble learning methods [52,53] train multiple models as an ensemble for classification or regression tasks used in specific combinations, also called aggregation or fusion. With our approach, LoE also follows this idiom. More specifically, it is a dynamic selection (DS) method [18], which, in contrast with static methods (e.g., AdaBoost [27] and random forests [28]), does not use the whole ensemble to form a decision. It uses a separate selection method to select ensemble subsets for individual queries. The rationale of DS is to dynamically find a subset of the models that are likely to be knowledgeable about a given query presented to the system, for which a filter method that decides which members of the ensemble may be knowledgeable in predicting a query is employed. This filter is often denoted as a selection or assignment function. DS methods fall into the class of either dynamic classifier selection (DCS) or dynamic ensemble selection (DES) methods. While DCS methods will choose exactly one model from the pool when presented a query, DES methods may choose multiple models. LoE is categorized as a DCS method, as it selects exactly one model given a query to retain explainability.

3. The League of Experts Classifier

For the realization of the *League of Experts* as a *classifier*, we adapt a particular case of ensemble learning, in which each ensemble member is associated with a specific part of the feature space. More specifically, we introduce an algorithm that adaptively learns a partition of the feature space in order to assign one specialized ensemble member—the *expert*—to each subset of the partition. Since the partition divides the feature space into disjoint sets, each expert is trained on a different subset of the training data. This diversifies the learning process, which implies that each expert performs particularly well only on its assigned subset but not necessarily on the whole feature space. Since each expert is not required to model the whole feature space, the complexity of experts can be limited without sacrificing the overall performance.

Motivated by the way in which inferences are made, the resulting ensemble of experts is named accordingly. Once an LoE classifier is trained, inference follows a two-step approach: (i) a query x_q is assigned to the correct expert g, and (ii) the query output $g(x_q)$ is processed by the assigned expert (see also Figure 2). This process mimics human teamwork, having a number of experts that are specialized in specific domains, in which a coordinator assigns tasks to the most appropriate expert. Allegorically, imagine a query being a patient visiting a primary physician (coordinator) that performs an initial examination. In this case, the coordinator is referring the patient to the specialist.

The following sections introduce our definitions and model formulations (Section 3.1), our approach to deriving explanations (Section 3.2), the reduction in the complexity of explanations (Section 3.3), the analysis of LoE's runtime complexity (Section 3.4), and LoE's rule set learner, also called RuleLoE (Section 3.5).

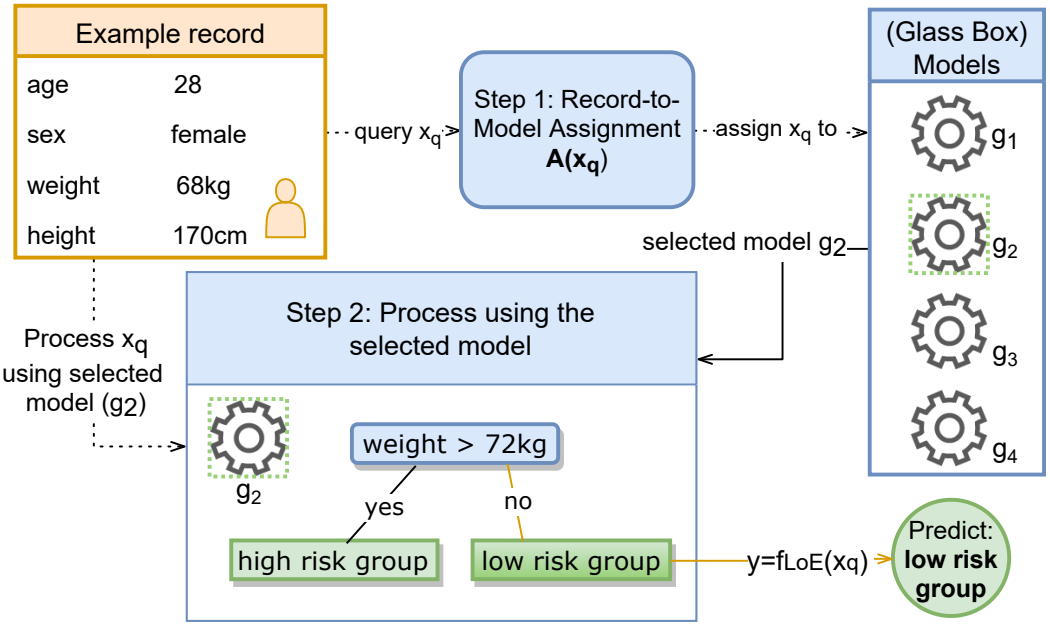

Figure 2. LoE's two-step decision process: (i) expert assignment to query (here, g_2, a one-level decision tree) and (ii) query prediction by the assigned expert (here, prediction `low-risk group`). Here, experts are illustrated as gear wheels.

3.1. Basic Definitions and Model Formulation

Consider a typical supervised learning problem based on a labeled dataset $\mathcal{L} \subseteq \mathcal{X} \times \mathcal{C}$. Here, each labeled data point $(x, y) \in \mathcal{L}$ is generated by a probability distribution while consisting of m real-valued inputs of a domain $\mathcal{X} \subseteq \mathbb{R}^m$ and a categorical (or, in the case of regression problems, metrical) label within a set $\mathcal{C}$ (for metric outputs $\mathcal{C} \subseteq \mathbb{R}^l$). Notably, categorical inputs are also permitted, for which they are assumed to be one-hot encoded. For a point $(x, y) \in \mathcal{L}$, we seek to maximize the probability of the output Y conditioned on the input X based on the model $P(Y = y|X = x) = g_s(x, y)$, where $g_s : \mathcal{X} \times \mathcal{C} \to [0, 1]$ is a (soft) classifier, belonging to a family $\mathcal{G}_s$.

The corresponding derived hard classifier is defined as follows:

$$g(x) := \arg\max_{y \in \mathcal{C}} P(Y = y|X = x),$$

where $g : \mathcal{X} \to \mathcal{C}$. The resulting family of hard classifiers derived from $\mathcal{G}_s$ is denoted by $\mathcal{G}$. For regression rather than classification problems, the conditional expectation of the output Y given the input X is modeled rather than the conditional probability. More precisely, the underlying regression model is $\mathbb{E}(Y|X = x) = g(x)$.

The following function:

$$A : \mathcal{X} \to \{1, \ldots, n\}$$

is called an *assignment function* of degree n ($|A| = n$). Each assignment function A defines a partition $\mathcal{X} = \bigcup_{k=1}^{n} \mathcal{X}_k^{(A)}$ and $\mathcal{X}_k^{(A)} \cap \mathcal{X}_l^{(A)} = \emptyset$ for $k \neq l$ of the feature space $\mathcal{X}$ of rank n by the following:

$$\mathcal{X}_k^{(A)} := A^{-1}(k).$$

If one classifier (g_k) is assigned to each part ($\mathcal{X}_k^{(A)}$) of the partition, we obtain an n-tuple $\mathcal{G}^*$ in $\mathcal{G}^n$ (derived from $\mathcal{G}_s^* \in \mathcal{G}_s^n$). A pair $(\mathcal{G}^*, A)$ is called an LoE of degree n inherited from A. The set $\mathcal{X}_k^{(A)}$ is called the *territory* of the expert g_k.

An LoE defines a single new classifier by the map:

$$x \mapsto g_{A(x)}(x),$$

which generally is not an element of $\mathcal{G}$. The set of all admissible assignment functions $\mathcal{A}$ (or equivalently the set of all corresponding partitions $\mathcal{A}_{\text{part}}$) and classifiers $\mathcal{G}$ defines the set of all possible LoE classifiers, denoted by $\mathbb{G}$. At this point, it is also noted that an LoE is conceptually similar to DCS algorithms (Section 2.7) with the assignment taking the role of the selection function. In the following, because an assignment function is equivalent to the partition it defines, we identify an assignment function by its equivalent partition.

The quality of an LoE classifier is measured by a performance function $p : \mathcal{C} \times \mathcal{C} \to \mathbb{R}$. A typical choice is the Kronecker delta:

$$p(\hat{y}, y) := \begin{cases} 1 & \text{if} \quad \hat{y} = y, \\ 0 & \text{if} \quad \hat{y} \neq y. \end{cases} \tag{1}$$

For a metrical output (regression task), the performance function typically follows from a distance measure between $\hat{y}$ and y, e.g., the negative mean-squared distance.

An optimal LoE classifier maximizes the average performance over the whole input and output domain $\mathcal{X} \times \mathcal{C}$; i.e., it is defined by the following:

$$f_{\text{LoE}}(x) := \underset{f \in \mathbb{G}}{\arg\max} \int\limits_{\mathcal{X} \times \mathcal{C}} p\big(f(x), y\big) dF(x, y), \tag{2}$$

where $F(x, y)$ is the joint probability distribution of inputs X and outputs Y.

The maximization in (2) is performed in two steps: (i) for each assignment function $A \in \mathcal{A}$ (or its equivalent partition), an optimal ensemble of experts $\mathcal{G}_A^* = \left(g_k^{(A)}\right)_{k=1}^{|A|}$ is found, and (ii) these optimal LoEs $(\mathcal{G}_A^*, A)$ are maximized over all assignment functions $A \in \mathcal{A}$ (or equivalently all partitions). This yields an optimal LoE classifier $(\mathcal{G}_{A'}^*, A')$, i.e.,

$$f_{\text{LoE}}(x) := g_{A'(x)}^{(A')}(x), \tag{3}$$

where

$$A' = \underset{A \in \mathcal{A}}{\arg\max} \int\limits_{\mathcal{X} \times \mathcal{C}} p\big(g_{A(x)}^{(A)}(x), y\big) dF(x, y), \tag{4}$$

and

$$\mathcal{G}_A^* = \left(g_k^{(A)}\right)_{k=1}^{|A|} = \underset{\mathcal{G}^* \in \mathcal{G}^{|A|}}{\arg\max} \int\limits_{\mathcal{X} \times \mathcal{C}} p\big(g_{A(x)}(x), y\big) dF(x, y). \tag{5}$$

Notably, an optimal LoE classifier is not necessarily unique. Furthermore, it is infeasible to find one, as the distribution $F(x, y)$ is unknown and the number of possible partitions is exhaustive. In practice, a pseudo-optimal LoE is empirically determined by replacing $F(x, y)$ by the empirical distribution function $\hat{F}(x, y)$ obtained from the labeled dataset $\mathcal{L}$. This yields the following:

$$f_{\text{LoE}}(x) := g_{A'(x)}^{(A')}(x), \tag{6a}$$

where

$$A'(x) = \arg\max_{A \in \mathcal{A}} \frac{1}{|\mathcal{L}|} \sum_{(x,y)\in\mathcal{L}} p\big(g^{(A)}_{A(x)}(x), y\big), \tag{6b}$$

and for each $A \in \mathcal{A}$

$$\mathcal{G}^*_A = \arg\max_{\mathcal{G}^* \in \mathcal{G}^{|A|}} \frac{1}{|\mathcal{L}|} \sum_{(x,y)\in\mathcal{L}} p\big(g_{A(x)}(x), y\big). \tag{6c}$$

However, this optimization is also infeasible. In general, an uncountable amount of tuples $\mathcal{G}^* \in \mathcal{G}^{|A|}$ (6c) and assignment functions $A \in \mathcal{A}$ (6b) exist. However, this can be heuristically approximated for a specific class of assignment functions. We provide an algorithm yielding an LoE $(\hat{\mathcal{G}}^{(A^*)}, A^*)$, performing almost as good as an optimal LoE.

3.1.1. Class of Assignment Functions

The set of assignment functions $\mathcal{A}$ of degree n is restricted to the class of functions defined by n pairwise different anchor points $\theta_1, \ldots, \theta_n \in \mathbb{R}^m$ by the following:

$$A : \mathcal{X} \to \{1, \ldots, n\}; \quad A(x) := \min\Big(\arg\min_{k\in\{1,\ldots,n\}} d(x, \theta_k)\Big),$$

where $d(\cdot,\cdot)$ is a given metric on $\mathcal{X}$. Without considering the minimum, A would not be well defined in the cases $d(x,\theta_k) = d(x,\theta_l)$ for $k \neq l$. Therefore, we assume the following:

$$\begin{aligned}\mathcal{A}_n := \big\{A \,\big|\, A(x) := \min\big(\arg\min_{k\in\{1,\ldots,n\}} d(x,\theta_k)\big) \\ \text{for pairwise different } \theta_k \in \mathbb{R}^m,\ k = 1,\ldots,n\big\}.\end{aligned} \tag{7}$$

Here, $\mathcal{A}_n$ depends on the choice of the metric d. It is advisable to use a metric that accounts for the variation of the input features. We employ the weighted L_1 metric $d(x,y) = \sum_{i=1}^m |x_i - y_i|/\sigma_i$, with σ_i being the empirical standard deviation of the i-th input. In practice, ς_i is replaced by its estimate from the labeled dataset $\mathcal{L}$. An alternative would be to employ a Mahalanobis distance.

As there is a one-to-one correspondence between the assignment function A and an n-tuple of anchor points $(\theta_1, \ldots, \theta_n)$ for each $A \in \mathcal{A}_n$, they are in turn identified. For an LoE classifier, anchor points correspond to the center points of the experts' territories, for which they are referred to as such.

3.1.2. Algorithm Training

An assignment function A partitions not only the feature space $\mathcal{X}$ but also the labeled dataset $\mathcal{L}$. Namely,

$$\mathcal{L}^{(A)}_k := \big\{(x,y) \in \mathcal{L} \,\big|\, x \in \mathcal{X}^{(A)}_k\big\}. \tag{8}$$

We approximate the optimization (6) for the class of assignment functions $\mathcal{A}_n$ with the following heuristic algorithm illustrated in Figure 3. The algorithm starts with n initial anchor points $\theta_1, \ldots, \theta_n$, and a corresponding assignment function A. Next, (6b) is approximated for the assignment function A by training one classifier $g_k \in \mathcal{G}$ for part $\mathcal{L}^{(A)}_k$ of the partition of the labeled dataset $\mathcal{L}$. This yields an ensemble $\hat{\mathcal{G}}^*_A$ and, hence, an LoE

$(\hat{\mathcal{G}}_A^*, A)$ being "quasi-optimal for A". Next, the anchor points θ_k are updated. We shift the center points of the experts' territories, whereby θ_k is replaced by the following:

$$\theta_k' = \theta_k + \frac{\eta}{|\mathcal{L}|} \sum_{(x,y)\in\mathcal{L}} (x - \theta_k) p(g_k(x), y), \tag{9}$$

where η is the update's step size. Here, each expert's performance is evaluated over the whole training data $\mathcal{L}$. The centers θ_k are shifted into the direction of points, for which the expert g_k performs well as measured by the performance function $p(g_k(\cdot), \cdot)$. The new center points define an assignment function A' and a new partition of the feature space, for which a new LoE is trained. This step is repeated with potentially decreasing step sizes. Finally, the LoE classifier with the overall best performance is chosen, which is referred to as a "quasi-optimal LoE". This procedure is described as a pseudocode in Algorithm 1. The updating step depends on the choice of the performance function. For simplicity, the Kronecker delta (1) is used for the examples presented here.

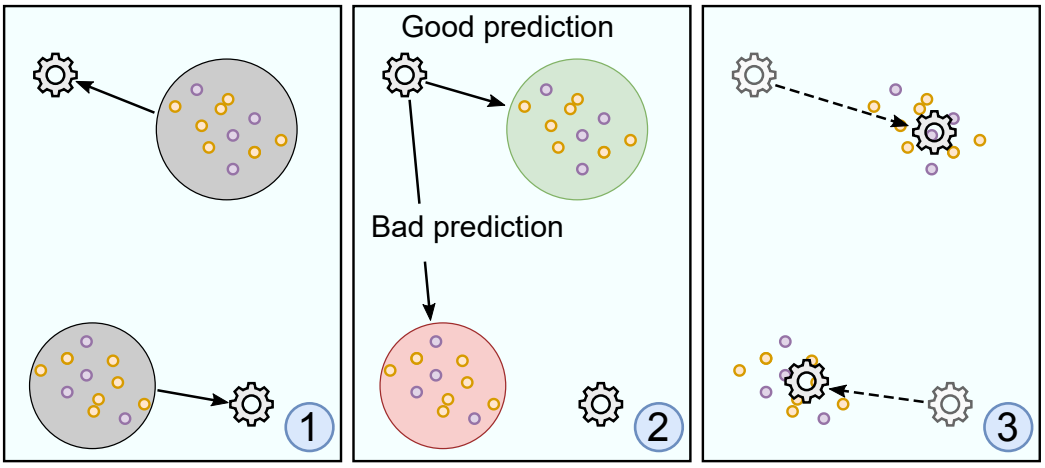

Figure 3. Training process of LoE with two experts separated into three steps: (**1**) data-to-model assignment (A), (**2**) evaluation phase, and (**3**) optimization phase (movement). Experts are illustrated as gear wheels. The training data are shown using purple and yellow circles, whereas each color depicts a distinct class of the learning problem. In **(2)**, a model's expected prediction performance on different partitions of the feature space are colored in green and red, relating to good and a bad performance, respectively.

Algorithm 1: Basic League of Experts Training Procedure

Input:
$\mathcal{G}$: set/class of (explainable) models, e.g., decision trees;
$\mathcal{L}$: Labeled dataset with $(x, y) \in \mathcal{L}$, $x \in \mathcal{X}$, $y \in \mathcal{C}$;
n: degree of LoE classifier;
η: step size;
T: number of iterations;
d: metric/distance, e.g., $d(x, x') = \|x - x'\|_2$;
p: performance function, e.g., $p(y, y') = \delta(y, y')$.

Result:
$(\hat{\mathcal{G}}^{(A^*)}, A^*)$: trained quasi-optimal LoE classifier, i.e., expert ensemble $\mathcal{G}^* = \{g_1^*, \ldots, g_n^*\}$, territories' center points $\theta_1^*, \ldots, \theta_n^*$, defining A^*

Initialize:
$S \leftarrow \emptyset$ # stores pairs of experts and center points
$S' \leftarrow S$, $i \leftarrow 0$, $p' \leftarrow -\infty$
initialize experts at random position in $\mathcal{X}$
$S = \bigcup_{i=1}^{n} \{(g_i, \theta_i)\}$

Algorithm 1: *Cont.*

while $t < T$ **do**
  # train experts on their territories:
  foreach $(g_i, \theta_i) \in S$ **do**
    $\mathcal{L}_g \leftarrow \{(x, y) | d(\theta_i, x) \leq d(\theta_j, x), j = 1, \ldots, n\}$
    Fit model g_i on dataset $\mathcal{L}_g$
  end
  foreach $(g_i, \theta_i) \in S$ **do**
    # move territories' center points
    $\theta_i' \leftarrow 0|_1^m$ # stores the movement vector
    foreach $(x, y) \in \mathcal{L}$ **do**
      # move towards x weighted by performance
      $\theta_i' \leftarrow \theta_i' + p\big(g_i(x), y\big) * (x - \theta_i)$
    end
    $\theta_i \leftarrow \theta_i + \eta / |\mathcal{L}| * \theta_i'$
  end
  $P' \leftarrow 0$ # calculate current performance
  foreach $(x, y) \in \mathcal{L}$ **do**
    # inference using the current experts and positions
    $y' \leftarrow f_{\text{LoE}}(x)$
    $P' \leftarrow P' + p(y, y')$
  end
  # keep track of best performing LoE
  if $P' > P$ **then**
    $P \leftarrow P'$
    $S \leftarrow \bigcup_{i=1}^{n} \{(g_i, \theta_i')\}$
  end
  $t \leftarrow t + 1$
end
Return: S

3.1.3. LoE Inference

Let $(\hat{\mathcal{G}}^{(A^*)}, A^*)$ be a quasi-optimal LoE with A^* being characterized by the anchor points $\theta_1^*, \ldots, \theta_n^*$. The tuple $(\hat{\mathcal{G}}^{(A^*)}, A^*)$ is a substitute for f_{LoE} in (3). For a query x_q, the distances $d(x_q, \theta_k^*)$ are calculated for $k = 1, \ldots, n$. Then, x_q is assigned to the associated closest expert, say g_i^* in (7), to predict the outcome $\hat{y} = g_i^*(x_q)$ (Figure 2).

3.2. Deriving Explanations

In the previous sections, we described a functional and modular machine learning algorithm. Instantiated using glass box models as experts, the model itself can already be employed to train a set of diverse (i.e., due to being trained on disjoint data) and interplaying classification problems (i.e., solving a common task). While the resulting experts can unveil information about the learning problem on their own by directly representing a part of the underlying decision process, the assignment function is still not yet directly visible. The following sections will therefore extend on this missing part by further utilizing the properties of the model in order to derive explanations.

In our context, explainability involves two aspects: (i) understanding the assignment process, i.e., why a query is assigned to a specific expert, and (ii) understanding the experts themselves, i.e., why an expert predicts a specific outcome inside its territory. The latter implies either that the experts themselves are glass box models—as assumed here—or that post hoc methods are applied otherwise (see also Section 2.1). To explain the assignment process, the decision boundaries of the assignment function A must be understood, which might be nontrivial. Their explanation should be accurate, comprehensible, and comprehensive, where requirements are naturally leading to a trade-off as "One Explanation Does

Not Fit All [Alike]" [54]. While there is a multitude of options to explain an assignment function A (or the implied partitions), we will focus on revealing a functional relationship and their interactions between relevant attributes. To achieve this, we transform the assignment process into an equivalent classification problem that is solved by employing a glass box model that acts as a surrogate for A. Although employing surrogates to model a process imposes risks, as mentioned in Section 2.6, LoE exhibits properties that allow the mitigation of the mentioned risks. Particularly, a part of the model's decision process remains intact as the experts themselves will not be approximated by the surrogate. By steering the concrete complexity of the surrogate models and by employing different modeling strategies (Section 3.5.1) and feature space reduction techniques induced by a slightly modified training routine (Section 3.3.1), the faithfulness of the surrogate can be further optimized. Although any model can be employed for this purpose, this work will focus on using decision trees as they allow for a direct transformation of the learned LoE model into the rule set learner, called *RuleLoE*.

3.2.1. Making the Assignment Function Explainable

To make the assignment function explainable, a glass box model is trained as a surrogate model for the assignment function A. Let $\mathcal{S}$ denote the family of such surrogates, which require a training dataset. The inputs of the assignment function are elements of the feature space $\mathcal{X}$, while the labels are elements of the set $\{1, \ldots, |A(\mathcal{X})|\}$. For a subset $\mathcal{X}_s \subseteq \mathcal{X}$ of the feature space, the graph of the assignment function A restricted to $\mathcal{X}_s$ is as follows:

$$\tilde{\mathcal{L}}_{A,\mathcal{X}_s} = \{(x, A(x)) | x \in \mathcal{X}_s\}. \tag{10}$$

Surrogates are assignment functions, but in general, their corresponding partitions are not members of $\mathcal{A}$. Such a set $\tilde{\mathcal{L}}_{A,\mathcal{X}_s}$ can be used as labeled data to train a surrogate $h \in \mathcal{S}$. The set $\tilde{\mathcal{L}}_{A,\mathcal{X}_s}$ is called assignment dataset. In the following, A will be the assignment function of a given LoE $(\mathcal{G}^*, A)$. We model A as a decision tree trained on $\tilde{\mathcal{L}}_{A,\mathcal{X}_s}$ (i.e., A is learning how to map a data point to a specific expert). The whole decision process for classifying a data point x_q is expressed as a concatenation of the decision paths of the two models A and g_k with $k = A(x_q)$, whereas the final prediction is $\hat{y} = g_{A(x_q)}(x_q)$. This process is illustrated in Figure 4.

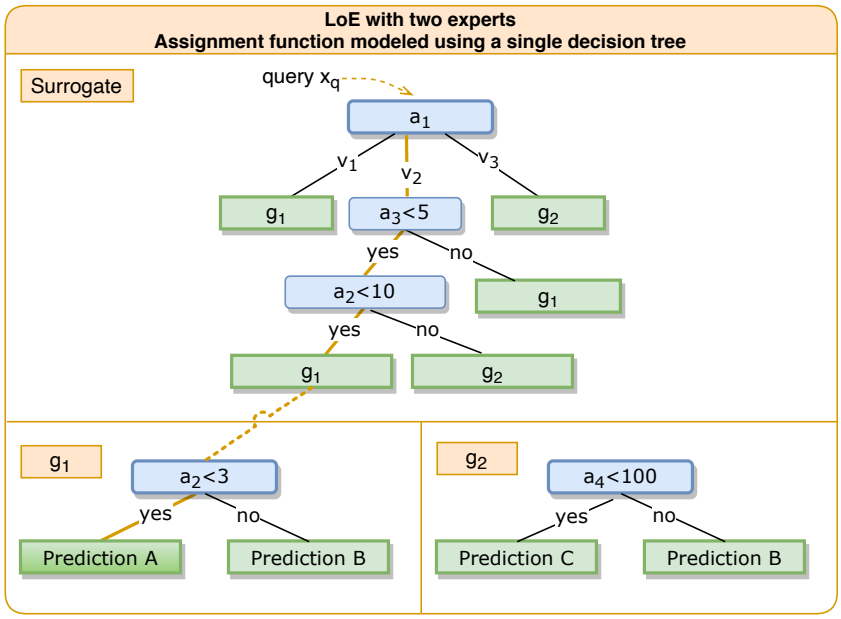

Figure 4. Exemplified LoE instance containing two experts (g_1 and g_2) with A being replaced by a single decision tree (surrogate). The orange line denotes a decision path of a query x_q (where $x_q = (a_1, a_2, a_3)$ satisfies $a_1 = v_2 \wedge a_3 < 5 \wedge a_2 < 3$) through both the surrogate and one expert. At this point, it is also noted that only the expert g_1 is relevant for the query x_q.

3.2.2. The Surrogate's Faithfulness

The surrogate's performance in mimicking the original assignment function A (i.e., the assignment accuracy) must be evaluated over a sample $\mathcal{X}_{\tilde{s}} \subseteq \mathcal{X}$. The level of agreement between the surrogate h and A is measured by $p_{\text{ass}}(h, A; \mathcal{X}_{\tilde{s}})$ ($p_{\text{ass}} : \mathcal{S} \times \mathcal{A} \times (\mathscr{P}(\mathcal{X}) \setminus \emptyset) \to \mathbb{R}^{\geq 0}$, where $\mathscr{P}$ denotes the power set), with high values indicating good agreement. If surrogates have unsatisfactory agreement, the class $\mathcal{S}$ might not capture the same mechanisms as the elements in $\mathcal{A}$, or it does cover the correct mechanisms but is too constrained. In these cases, the class $\mathcal{S}$ can be extended (i.e., relaxing regularizations, e.g., allowing larger decision trees). Alternatively, the amount of input features considered by the surrogate can be reduced, e.g., by reducing their feature space. Appropriate projections can be identified during LoE's training procedure (Section 3.3.1).

As the ultimate goal is to obtain comprehensible explanations of the whole decision process, the surrogate's comprehensibility has to be ensured. Hence, surrogates must be understandable. As the employed decision trees can be of arbitrary complexity, we have to introduce a measure thereof. To do so, we first introduce a basic notation.

3.2.3. Decision-Tree-Related Notation

For a decision tree t, its root node, the set of leaves, and the set of ancestors from node v to the root are denoted by $\text{head}(t)$, $\text{leafs}(t)$, and $\text{trace}(t, v)$, respectively. The decision tree assigns each data point a leaf. The path of this leaf is the nodes from the root to its starting point. For a subset $\mathcal{X}_s \subseteq \mathcal{X}$ of the data, evaluated over the tree t, let $|v|_{(\mathcal{X}_s, t)}$ be the number of data points in $\mathcal{X}_s$ that follow paths passing through node v. As the root is contained in every path, and exactly one path is leading to each leaf, $|\text{head}(t)|_{(\mathcal{X}_s, t)} = |\mathcal{X}_s|$ (i.e., all data points start at the root node).

3.2.4. Complexity of a Decision Tree

Although decision trees can always be visualized as a graphical representation or linearized as rules, this can be impractical due to their size, i.e., depth, width, or structure, which are measuring the tree's complexity. This argument ignores the distribution of queries. Namely, if the majority of queries utilize only a simple subtree, a complex tree remains practical for most instances. The distribution of queries is taken into account when measuring the decision tree's complexity by the average trace length to the leafs of a representative sample $\mathcal{X}_1 \subseteq \mathcal{X}$, i.e.,

$$\mathrm{E}_{\mathcal{X}_1}\left(\text{trace}(t, .)\right) = \frac{1}{|\mathcal{X}_1|} \sum_{v \in \text{leafs}(t)} |v|_{(\mathcal{X}_1, t)} \left|\text{trace}(t, v)\right|. \tag{11}$$

If the choice of $\mathcal{X}_1$ does not follow naturally, it can be constructed for a uniform sample distribution over the leaves, i.e.,

$$\mathrm{E}\left(\text{trace}(t, .)\right) = \frac{1}{|\text{leafs}(t)|} \sum_{v \in \text{leafs}(t)} \left|\text{trace}(t, v)\right|. \tag{12}$$

3.3. Reducing the Complexity of Explanations

Comprehensiveness depends on the cognitive load confronting users. We introduce a concept to reduce cognitive load by effectively restricting the LoE model to a lower-dimensional feature space. The process impacts the training process of LoE while preserving its predictive power. This also affects the assignment process and inherits to surrogates. Although surrogates approximate complex nonlinear decision boundaries, most practically relevant datasets are aligned along a lower-dimensional manifold [55] being well approximated by projections. A high-dimensional feature space, i.e., a large number of attributes, typically compromises the generalization properties of ML models due to overfitting. For LoE's training procedure, the number of classifiers $|\mathcal{S}|$ increases

exponentially such that several surrogates not faithfully explaining A will still have good performance, which justify the assignment but do not reflect true causality (see Section 2.6). This is known as Rashomon set [54].

3.3.1. Feature Space Reduction

We mitigate high-dimensional drawbacks by projecting the feature space onto a lower-dimensional subspace. Let $\pi : \mathbb{R}^m \to \mathbb{R}^r$, defined by $\pi(x) = (x_{i_1}, \ldots, x_{i_r})$, be the projection of $x = (x_1, \ldots, x_m)$ onto the r components $i_1, \ldots, i_r$ ($1 \leq i_1 < i_2 \ldots < i_r \leq m$). The metric d is replaced by the (pseudo) metric $d_\pi(x, y) := d(\pi(x), \pi(y))$, which is the restriction of d to the projection space defined by π. Thus, (7) is replaced by the following:

$$\mathcal{A}_n := \Big\{ A \,\Big|\, A(x) := \min \Big\{ \operatorname*{arg\,min}_{k \in \{1,\ldots,n\}} d_\pi\big(x, \pi(\theta_k)\big) \Big\} \text{ for pairwise different } \theta_k \in \mathbb{R}^m,\ k = 1, \ldots n \Big\}. \tag{13}$$

The updating step of the center points of the experts' territories is restricted to the lower-dimensional projection space, while leaving the remaining components unchanged; i.e., (9) becomes the following:

$$\pi(\theta_k') = \pi(\theta_k) + \frac{\eta}{|\mathcal{L}|} \sum_{(x,y) \in \mathcal{L}} \big(\pi(x) - \pi(\theta_k)\big)\, p\big(g_k(x), y\big), \tag{14}$$

where the performance is still evaluated over the original unprojected sample $\mathcal{L}$. However, surrogates are trained on the projected feature space. This reduces complexity and the Rashomon effect (Section 2.6), thereby increasing the faithfulness of the surrogate's explanations. Moreover, users' understandability could be increased as they are confronted with interpreting fewer attributes.

If decision trees are built with fewer attributes, they are more likely to appear multiple times along a decision path. They can be merged into a single rule, facilitating explainability. For instance, if $\texttt{age} < 40$ and $\texttt{age} < 30$ occur along a path, the two rules can be reduced to $(\texttt{age} < 30)$. Similarly, $(\texttt{age} < 40$ and $\texttt{age} > 30)$ reduce to $(\texttt{age}$ in range 30 to $40)$.

3.3.2. Projections Obtained from Leveraging LoE's Properties

A projection π is based on a selection of attributes. Ideally, "important" attributes are retained. For instance, attributes that seem to be strongly correlated, have low predictive power, or seem noisy can be removed. Advanced approaches make use of permutation feature importances [42] (Chapter 5.5), SHAP feature importances [42] (Chapter 5.10) based on Shapley values [41], or individual experts' assessments of the importances of the attributes.

Concerning LoE (and other DCS methods), attributes contributing most to the assignment process also determine their importance. Each expert in LoE is able to individually calculate the importances of the attributes, which can be combined into an importance score. Approximately, the importance scores can be determined based on a surrogate h rather than on the original assignment function A (see also Section 3.2.1). A requirement is that the choice of the surrogate is capable of reporting feature importance. One possibility is to choose a second surrogate family $\tilde{\mathcal{S}}$ for this purpose, which is not necessarily explainable but able to report feature importances. A possible choice is extra trees (extremely randomized trees) [56], which are ensembles of decision trees using a randomized collection of attributes and cut points. They do not optimize a split criterion and, hence, can be generated fast while yielding good approximations of the feature importance given a sufficiently large ensemble. Moreover, they are not biased by an optimization procedure (e.g., by optimizing for the Gini index) during the construction step.

The feature importance within LoE is partly determined during its iterative procedure. The importance scores are obtained in every step. These can be combined to rolling means or other time-dependent quantities indicative of attributes' importances.

3.3.3. Concrete Calculation of Feature Importances

We present a concrete implementation for calculating feature importances by utilizing decision trees as a primary example. Although decision trees are a natural choice for this purpose, it is worth noting that other models capable of reporting feature importances can also be used as alternatives. For the construction of decision trees, the importance of attributes is naturally incorporated. An attribute's importance is identified with its performance as readily calculated by the relative change of a measure such as the Gini index during the decision tree's construction [57]. In the described example of extra trees, the measure is not already used when generating the ensemble.

For an LoE $(\mathcal{G}^*, A)$ with expert ensemble $\{g_1, \ldots, g_n\}$ (here, decision trees) and data $\mathcal{L}$, let $\mathcal{L}_k$ be the subsample assigned to expert g_k (corresponding to $\mathcal{X}_k$). An importance function ϕ assigns every expert the relative importance of each attribute; i.e., it is a map from the LoE ensemble to the $m-1$-dimensional simplex ($\phi : \mathcal{G}^* \to \mathcal{S}_{m-1}$) defined by $\phi : g_k \mapsto (\phi_1(g_k), \ldots, \phi_m(g_k))$. The ensemble's feature importance of the attributes is the weighted relative importances across all experts, i.e.,

$$\phi_{\text{experts}}(\mathcal{G}^*) = \sum_{k=1}^{n} \frac{|\mathcal{X}_k|}{\mathcal{X}} \phi(g_k) \in S_{m-1}. \tag{15}$$

For the assignment process, the importance of the features are calculated separately. From the second surrogate family $\tilde{\mathcal{S}}$, here, an extra tree ensemble of 1000 decision trees, a surrogate is trained to approximate the assignment function A. An importance function $\tilde{\phi}$ analogous to ϕ is evaluated at the trained surrogate h to yield each attribute's importance. The total importance averages the above quantities with weight α, i.e.,

$$\phi_{\text{total}} = \alpha \phi_{\text{experts}}(\mathcal{G}^*) + (1-\alpha)\tilde{\phi}(h). \tag{16}$$

The total feature importance is updated in every step of the LoE algorithm. Rather than using these importances directly, a time average is calculated. The time-averaged total feature importance in step t with geometric decay β is derived as follows:

$$\Phi_{\text{total}}^{(t)} = \beta \phi_{\text{total}} + (1-\beta)\Phi_{\text{total}}^{(t-1)}. \tag{17}$$

The projection π to reduce the dimension of the feature space is constructed by projecting on the r features with the highest importance.

3.4. Runtime Analysis

To determine the runtime complexity of LoE in terms of its training process, the following considerations are taken into account. First, LoE's runtime complexity's determinative factors are reiterated. Second, LoE's training complexity is derived.

- T = number of iterations
- $\mathcal{G}$ = ensemble of experts
- $\mathcal{L}$ = labeled dataset
- $C(g, (x,y) \subseteq \mathcal{L})$ = training complexity for model g for data (x,y)
- $E(g, (x,y) \subseteq \mathcal{L})$ = evaluation complexity for model g for data (x,y)

$$\mathcal{O}(T \cdot (|\mathcal{G}| \cdot |\mathcal{L}| + \max_{g \in \mathcal{G}} C(g, \mathcal{L}) + \max_{g \in \mathcal{G}} E(g, \mathcal{L}))) \text{, with}$$

$$\begin{aligned} &\text{1. Assignment } \texttt{data} \twoheadrightarrow \texttt{model}: && = && \mathcal{O}\big(T \cdot |\mathcal{G}| \cdot |\mathcal{L}|\big) \\ &\text{2. Expert Training:} && + && \mathcal{O}\big(T \cdot \max_{g \in \mathcal{G}} C(g, \mathcal{L})\big) \\ &\text{3. Model Movement:} && + && \mathcal{O}\big(T \cdot \max_{g \in \mathcal{G}} E(g, \mathcal{L})\big) \\ &\text{4. Performance Evaluation:} && + && \mathcal{O}\big(T \cdot \max_{g \in \mathcal{G}} E(g, \mathcal{L})\big) \end{aligned} \tag{18}$$

The upper bound of the process in (18) is determined by four factors that repeat for T times: First, the assignment is evaluated (1). This step calculates the distance between each expert $g \in \mathcal{G}$ and each data point. Second, the expert training process and model movement is carried out (2). The experts are trained on their respectively allocated data points (i.e., trained within their territory), which is bound by the training complexity of the concrete model types of $\mathcal{G}$. Third, the trained models are moved in the direction of their best-performing feature space. Lastly, the trained experts are used for inference in order to measure their performance (4). This step is bound by the chosen models' inference complexity.

The inference complexity of the method given a sample $\mathcal{X}$ is calculated as follows:

$$|\mathcal{G}| \cdot |\mathcal{X}| + \max_{g \in \mathcal{G}} E(g, \mathcal{X}), \tag{19}$$

which constitutes finding the relevant expert (i.e., the assignment process) and using this expert for inference. Contrary to ensemble methods such as random forests or AdaBoost, the training runtime scales with the additional factor T. This implies that the algorithm trains T times slower. However, on inference, only one model is evaluated instead of the whole ensemble, for which the results of each ensemble member do not need to be aggregated. This effectively results in $|\mathcal{G}|$ distance calculations plus $E(g, (x, y))$ for the assigned or "closest" model for each $x \in \mathcal{X}$. This means that the inference time of LoE scales predictably with the number of experts, while other ensemble methods scale with the aggregated complexity of their experts.

3.5. LoE as a Rule Set Learner—RuleLoE

LoE is not a native rule set learner, but it can be transformed into an almost equivalent one, denoted as RuleLoE, by employing its surrogate (Section 3.2.1). For this purpose, we focus on decision trees as experts and surrogates. However, they can also be replaced by other rule learners. Concrete decision rules are derived as logical conjunctions of (i) decision paths leading to specific experts (i.e., the assignment path `surrogate` $\rightarrow$ `expert`) and (ii) the decision paths of the assigned expert to the leaves. Derived decision rules are separated into conjunctions of two components. The complexity can be influenced by user adjustments to the two components: (i) the quality of the surrogate's approximation and its complexity (e.g., restricting the maximal depth of a decision tree or using feature reductions, Section 3.3) and (ii) the amount and complexity of the experts (a larger, less complex ensemble vs. a smaller, more complex one). Additionally, if a few experts have good global performance (i.e., on the whole sample), pruning assignment rules can efficiently increase performance since pruned assignment rules are more likely to direct to well-performing experts.

3.5.1. One-versus-Rest Surrogate

The derived rules from a decision tree as surrogate have no rule overlap; i.e., no pair of rules is true at the same time. While this is a desired property due to having no ambiguities, it also has potential drawbacks: No matter what query is presented to the process, there will always be some partial overlap as every query starts at the head of the tree, hence sharing at least one common test. This reduces the flexibility and versatility of the resulting set of rules. However, rule overlaps and partial rule overlaps are not desirable as emerging rules will become too similar. In practice, it is worth sacrificing the first property to obtain more diverse rules by minimizing partial overlap. This is achieved by training one *surrogate* for *each expert* as follows. The k-th surrogate will learn to model the assignment process relative to the k-th expert in a binary fashion (one-versus-rest strategy, OvR). Effectively, each of the k surrogates is presented with a binary-labeled dataset where the value 1 represents $A(x) = k$ (and 0 otherwise). It learns to decide if the k-th model should be used in predicting a given query. If appropriate, the k-th expert is assigned (i.e., its decision path is followed). Otherwise, no assignment is made (i.e., the expert is not taken into account). A visual example of a scenario using two surrogates is shown in Figure 5. Note that using OvR in a scenario with two experts is equivalent to using exactly one surrogate as the labels that are presented to both surrogates are the inverse of each other.

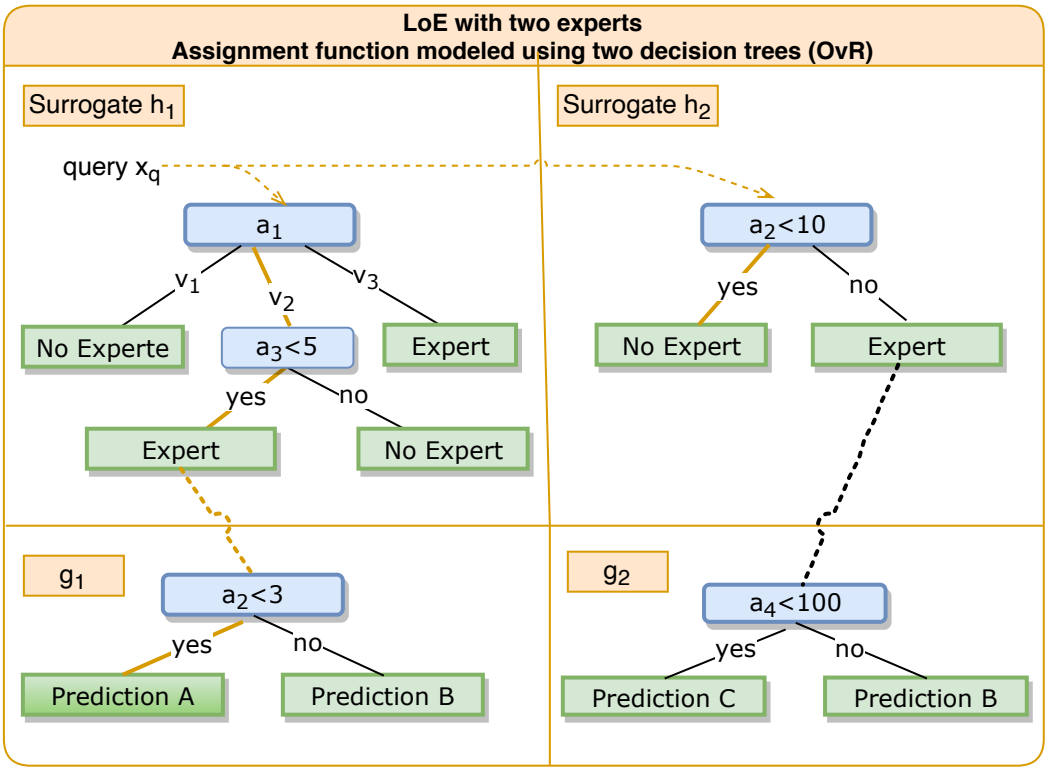

Figure 5. Exemplified LoE instance containing two experts (g_1 and g_2) with A being replaced by two decision trees (surrogates). Note that this example is only illustrative as the surrogates would be inverse instances to each other in a 2-expert scenario.

Since the labels are different for each expert when using more than two experts, each surrogate is trained on a different dataset. Hence, each of them learns different structures, in particular because decision trees are sensitive to the input, ultimately resulting in more diverse and potentially simpler rules. When using surrogates instead of the original assignment function, the process may not always be captured perfectly since the assignment function itself may be arbitrarily complex, whereas the surrogate's complexity may be limited or not be able to fully model the original function. Therefore, it can happen that none or several of the n surrogates are assigned an expert to the query.

3.5.2. Query Strategy

The problem of overlapping rules and uncovered queries potentially occurring in decision sets created by OvR is still to be resolved. This is done by mimicking human thinking. For competing rules, the rule giving the best overall result is chosen. If no rule applies, the best-fitting one is chosen. Formally, let $\mathcal{L}_{\text{train}}$ be the training dataset and r_i a decision rule (e.g., $r_i \leftarrow \texttt{age} > 20 \wedge \texttt{size} < 180cm \Rightarrow \texttt{positive class}$). If N_i is the number of data points to which r_i is applicable, the rule's coverage is defined as $c_i = {}^{N_i}/_{|\mathcal{L}|}$. The number of covered points correctly classified by r_i is N_i^+, so that the rule's precision is $\theta_i = {}^{N_i^+}/_{N_i}$. Rules are assigned to a query x_q as follows:

1. Rank rules by descending precision (and coverage on tie).
2. Assign x_q to the covering rule with the lowest rank.
3. If no rule covers x_q, return the rule with the highest fraction of clauses being evaluated as "true" (partial rule coverage) with preference given to the rule with higher precision on ties.

This approach, particularly step 3, is possible only because RuleLoE is a multiclass rule learner. It is not applicable to approaches learning only one positive class, including algorithms such as RIPPER, CN2, or BRCG. Consequently, the latter might produce fewer rules but less data insights.

In conclusion, the proposed *League of Experts* algorithm and its framework reaches out to an audience seeking to apply, develop, and evaluate explainable artificial intelligence by providing the necessary tools. These are available at the project page of LoE and its repository. The project page of the *League of Experts (LoE)* framework can be found via its repositories under https://github.com/Mereep/loe, https://github.com/Mereep/rule_loe, and https://github.com/Mereep/HDTree, accessed on 2 April 2024.

4. Test Results, Evaluation, and Discussion

In the following sections, we exemplify the utilization of the League of Experts algorithm and its methodology given two different datasets from medicine and finance, the UCI breast cancer dataset and the HELOC dataset. Medicine and finance are two critical domains for the application of explainable models as the decision process may have strong implications. For example, in medicine, a patient's treatment can be evaluated, adjusted, or justified. In the domain of finance, aspects of accountability and discrimination are to be considered when, for example, credits are to be approved. Therefore, two comprehensive experiments are conducted and analyzed as follows: First, LoE is applied and the effect of feature space reduction is studied (Section 3.3.1). Second, LoE is transformed into RuleLoE, whose explainability and performance are compared with alternative rule learners and black box models (Section 4.2) and commonly known ensemble learners. Subsequently, an in-depth analysis of decision rules is performed on the HELOC dataset. Additionally, we conducted a series of experiments on a collection of datasets from the KEEL dataset repository using a static model setting to allow for a performance-level overview. Following, we provide a dataset overview.

Example dataset 1: UCI breast cancer

The UCI breast cancer dataset (UCI breast cancer dataset, https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic), accessed on 2 April 2024) [58] contains 569 data points with 32 real-valued attributes, describing the characteristics of cell nuclei found in digitized images of a fine-needle aspirate of breast mass, including information on, e.g., `mean radius`, `worst perimeter`, or `smoothness` of the contour. Data points are classified as `malignant` or `benign`.

Example dataset 2: HELOC

The HELOC dataset, used in the FICO Explainable Machine Learning Challenge (HELOC dataset, https://community.fico.com/community/xml, accessed on 2 April 2024) and different applications of explainable AI [38,59,60], describes the credit history of 10,549 individuals (9872 after removing identical entries), which are characterized by 23 metric features. The risk of default is categorized as `good` or `bad` if the individual had overdue payments for at least 90 days during a 24-month period.

Example dataset 3: KEEL

The KEEL dataset repository (Knowledge Extraction based on Evolutionary Learning) is a dataset that "aims at providing to the machine learning researchers a set of benchmarks to analyze the behavior of (...) learning methods" [61]. KEEL contains various datasets of different properties. For this work, we chose a subset of the repository using medium-sized datasets with a size between 5000 and 10,000 samples (Table 1).

Table 1. Overview of our selected subset of the KEEL dataset with sample sizes between 5000 and 10,000 samples.

Dataset	Data Points	Features	Classes	Class Entropy
banana	5300	2	2	0.99
coil2000	9822	85	2	0.33
marketing	6876	13	9	3.10
mushroom	5644	98	2	0.96
optdigits	5620	64	10	3.32
page-blocks	5472	10	5	0.63
phoneme	5404	5	2	0.87
ring	7400	20	2	1.00
satimage	6435	36	6	2.48
texture	5500	40	11	3.46
thyroid	7200	21	3	0.45
twonorm	7400	20	2	1.00

Used Software

All implementations were developed in Python (Python project page, https://www.python.org, accessed on 2 April 2024, version 3.8). The package scikit-learn (scikit-learn project page, https://scikit-learn.org, accessed on 2 April 2024, version 0.22.2.post1) was used for our experiments with random forests, gradient boosting, extra trees, AdaBoost, support vector classifiers (SVC), and multilayer perceptrons (MLP). The repositories Wittgenstein (Wittgenstein project page, https://github.com/imoscovitz/wittgenstein, accessed on 2 April 2024, version 0.1.6) and AIX360 (AI Explainability 360) [54] (AIX360 project page, https://github.com/IBM/AIX360, accessed on 2 April 2024, version 0.2.0) were used for RIPPER and BRCG, respectively.

4.1. Feature Space Reduction

The amount of features affects the assignment of the surrogate model h and the updates of the expert territories' center points. In this case, the surrogate model is a decision tree. Intuitively, the surrogate's complexity reduces with fewer features while becoming more accurate in the sense of approximating LoE's assignment function A.

For our experiments, the datasets are split into $2/3$ training data and $1/3$ test data. The family of surrogates consists of decision trees of maximum depth 6. LoE's structure is fixed to an ensemble of $n = 3$ experts from the class $\mathcal{G}$ of decision trees with maximum

depth 3. However, the number of features retained by the projections π varies between 10 and 100 % in 5 % increments ($r = n \times p$ with $p = 0.1, 0.15, \ldots, 1$). For each r, LoE is trained and evaluated over 20 runs. For these runs, different training and test data partitions are employed, which are repeatedly used for every value of r. The test and training accuracies, the average length of the surrogates' decision trees h, and the assignment performance θ_{ass} (i.e., the level of the surrogate's agreement with LoE's assignment function) are obtained.

The feature importance is calculated via (17) with weights $\alpha = 0.5$ (Equation (16)) and geometric decay $\beta = 0.3$ (Equation (17)). The parameters themselves were not optimized. The surrogates' decision rules were not simplified by merging.

Test Results

For the UCI breast cancer dataset, LoE with three experts slightly overfits the training data as seen from comparing the training with the test accuracy (Figure 6, comparison of C and D). The former results in almost 100 % accuracy, while the latter is close to 94 % accuracy. We surmise that it would be advisable to remove one of the experts or to constrain them.

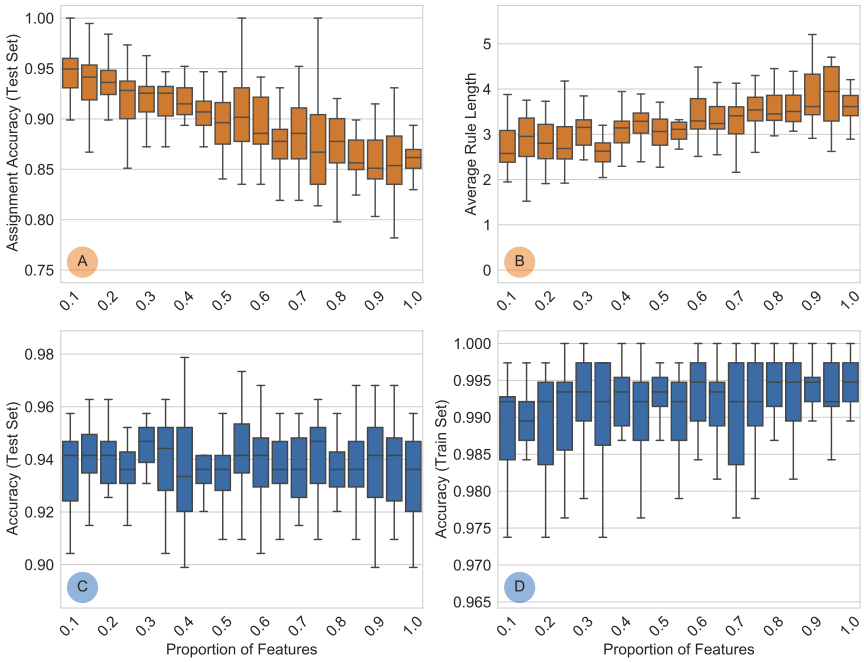

Figure 6. UCI breast cancer dataset results. Shown are paired box plots with their proportions of retained features over 20 runs for (**A**) assignment accuracy (p_{ass}), measured on test data, (**B**) average rule/path length, (**C**) accuracy for the test data, and (**D**) accuracy for the train data.

Feature reduction strongly influences the complexity and quality of the surrogate model. As shown in Figure 6B,A, the average rule length (average weighted path length) has a significant positive correlation with the amount of features (Pearson's χ^2 test: $\chi^2 = 0.48$, $p < 0.01$), while showing a significant negative correlation with the assignment accuracy (surrogate's level of agreement with $\mathcal{A}$; $\chi^2 = -0.62$, $p < 0.01$).

Similar results are obtained for the FICO data (Figure 7). The accuracy is lower with both the training and the test dataset. However, the effect of overfitting is similar to the UCI dataset (Figure 7C,D). However, feature reduction has an even stronger effect, whereas the rule length has a stronger positive correlation with the amount of features ($\chi^2 = 0.57$, $p < 0.01$). A stronger negative correlation with the assignment accuracy is observable ($\chi^2 = -0.94$, $p < 0.01$). The strong correlations underline the importance of exploring the optimal dimensionality for feature space reduction. When retaining more than 35% of the attributes, the average rule length quickly approaches the maximum admissible length

of 6 imposed by the surrogate family. This trend suggests that the derived decision rules would become more complex if they were allowed to; i.e., the surrogate would tend to overfit $\mathcal{A}$.

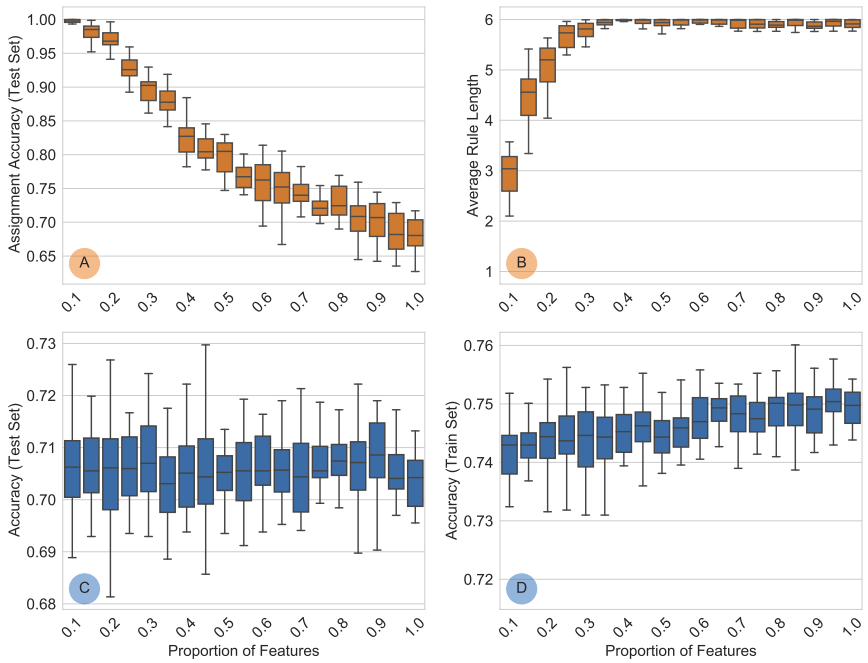

Figure 7. FICO dataset results. Shown are paired box plots with their proportions of retained features over 20 runs for (**A**) assignment accuracy (p_{ass}), measured on test data, (**B**) average rule/path length, (**C**) accuracy for the test data, and (**D**) accuracy for the train data.

4.2. Rule Set Generation with RuleLoE

The properties of RuleLoE (Section 3.5) concerning (i) explainability (average rule length, ∅ rules, and # rules), (ii) faithfulness (rule coverage), and (iii) accuracy are compared with alternative rule set learners, in particular with (1) RIPPER [36] (despite its age still state of the art; see [46] (p. 52) and [38]), (2) Boolean decision rules via column generation (BRCG) [38], and (3) classical CART decision trees. In terms of accuracy, RuleLoE is compared with two black box models, namely, SVC and MLP, using, if not noted otherwise, standard configurations.

As LoE is a form of ensemble learning, we furthermore compared LoE with a set of different well-known ensemble learners, namely, random forest, gradient boosting, extra trees, and AdaBoost, to allow for a more comparative analysis. As tree-based ensemble methods possess a notion of ensemble size and and maximum tree depth analogous to LoE, we trained two versions each: *default* and *adjusted*. *Default* refers to the algorithm having default hyperparameter settings. The *adjusted* versions diverge from the default by setting the ensemble size and the maximum tree depth to match LoE's parameterization. Our default ensemble configurations as pairs of ensemble size and max tree depth are as follows:

- Random forest: 100/∞
- Gradient boosting: 100/3
- Extra trees: 100/∞
- AdaBoost: 50/1

At this point, it is also noted that, technically, default AdaBoost only learns decision stumps, i.e., trees with a maximum depth of one. Although the employed ensemble learners are using decision trees, the ensemble methods cannot easily be used for explanations as

their results must be blended in the final stage. Especially using default hyperparameters, the models produce very big ensembles, which essentially renders them to black box models.

4.2.1. UCI Breast Cancer Dataset

A LoE ensemble of $n = 9$ decision trees with a maximum depth of 1 level and surrogate trees of a maximum depth of 2 with a feature reduction to $r = 11$ attributes was initially used. After training, one expert having a small territory was manually removed.

The complexity of rules differs substantially between the algorithms with no clear pattern emerging. RIPPER and BRCG focus on a single class to determine rules (positive *or* negative), whereas LoE and CART incorporate *both* classes, influencing complexity and performance. Consequently, coverage is lower for RIPPER and BRCG as they assign fewer data points to a rule. Coverage of decision trees (CART) shows 100 % accuracy as every query has a valid rule down to the leafs. Here, RuleLoE's performance is competitive with the other models—including the black box models. Glass box models were furthermore outperformed in terms of test accuracies (Table 2). None of the evaluated algorithms outperform in each other in all categories. In this case, there is no advantage to prefer black over glass box models.

Table 2. UCI breast cancer results. Shown are the number of rules (# rules), their average length (∅ rules), rule coverage, and (test/training) accuracies. Italic values for RIPPER and BRCG specify the learned positive class. Our found hyperparameters in Python notation using grid search are `ccp_alpha` $= 0.01$ (CART, post pruning); `hidden_layer_sizes` $= (200,)$ and `max_iter` $= 500$ (MLP).

Algorithm		**# Rules**	**∅ Rules**	**Rule Coverage (%)**	**Accuracy (%)**
RuleLoE		10	1.70	96.06/95.21	94.22/96.28
RIPPER	*malignant*	9	1.56	39.90/36.17	95.54 / 95.21
RIPPER	*benign*	13	1.53	61.41/61.70	93.20 / 89.89
BRCG	*malignant*	5	2.20	37.00/35.11	98.95 / 95.21
BRCG	*benign*	5	2.80	61.42/62.77	99.48 / 92.02
CART		7	3.00	100.00/100.00	97.11/95.74
Random forest	*default*	-	-	-	100.00/96.28
Random forest	*adjusted*	-	-	-	94.75/92.02
Gradient boosting	*default*	-	-	-	100.00/95.21
Gradient boosting	*adjusted*	-	-	-	94.23/94.15
Extra trees	*default*	-	-	-	100.00/97.87
Extra trees	*adjusted*	-	-	-	76.64/78.19
AdaBoost	*default*	-	-	-	100.00/95.21
AdaBoost	*adjusted*	-	-	-	98.95/96.81
SVC		-	-	-	90.55/95.21
MLP		-	-	-	94.22/95.21

4.2.2. FICO Dataset

A LoE ensemble with $n = 4$ experts with a maximum depth of 1 and surrogate decision trees with a maximum depth of 2 retaining $r = 8$ features was initially used. One expert was manually augmented to two levels as one of its leaves was assigned too many data points with very low purity (i.e., skewed class distribution).

RIPPER and CART learn a large amount of rules without achieving a higher accuracy (Table 3), whereas RIPPER shows slightly worse results. CART also learned complex rules. Interestingly, BRCG (positive class `bad`) learns exactly one short rule (**Predict** `y=bad` **if:**

`ExternalRiskEstimate` < 73), which performs well (Table 3). Naturally, `ExternalRiskEstimate` is a relevant predictor for the credit rating as it is a composition of risk markers scoring the risk by an (undisclosed) functional relationship. Hence, BRCG's simple for rule determining the credit score is based on a hard threshold that is intransparent to outsiders. This is different for RuleLoE, leading to the following rules (Figure 8).

Predict `y=good` **if one of the following applies:**

(1): ExternalRiskEstimate ≥ 76
(2): AverageMInFile $\geq 54 \ \wedge$ MSinceMostRecentInqexcl7days $\geq 1 \wedge$ ExternalRiskEstimate $\in [70, 76[$

Predict `y=bad` **if one of the following applies:**

(3): ExternalRiskEstimate $< 69 \ \wedge$ ExternalRiskEstimate $\neq 68$
(4): AverageMInFile $\geq 54 \ \wedge$ MSinceMostRecentInqexcl7days $< 1 \wedge$ ExternalRiskEstimate $\in [70, 76[$
(5): ExternalRiskEstimate INSIDE $[68, \ldots, 70[$
(6): AverageMInFile $< 54 \ \vee$ ExternalRiskEstimate $\in [70, 76[$

Figure 8. FICO dataset results. Shown are the obtained rules via RuleLoE.

Table 3. FICO dataset results. Shown are the number of rules (# rules), their average length (∅ rules), rule coverage, and (test / training) accuracies. Italic values for RIPPER and BRCG specify the learned positive class. Our found hyperparameters using grid search are `ccp_alpha` $= 0.001$ (CART, post pruning); `hidden_layer_sizes` $= (8, 8)$ (MLP).

Algorithm		**# Rules**	**∅ Rules**	**Rule Coverage (%)**	**Accuracy (%)**
RuleLoE		6	2.00	100.00/100.00	72.04/71.73
RIPPER	*bad*	18	2.50	50.44/49.66	73.95/70.03
RIPPER	*good*	25	2.40	47.05/47.39	73.95/71.49
BRCG	*bad*	1	1.00	56.01/56.29	71.12/70.38
BRCG	*good*	2	3.00	48.50/48.22	72, 35/70.53
CART		21	4.86	100.00/100.00	74.05/71.24
Random forest	*default*	-	-	-	100.00/72.41
Random forest	*adjusted*	-	-	-	100.00/72.41
Gradient boosting	*default*	-	-	-	62.25/62.83
Gradient boosting	*adjusted*	-	-	-	76.70/73.20
Extra trees	*default*	-	-	-	71.12/70.38
Extra trees	*adjusted*	-	-	-	62.49/62.86
AdaBoost	*default*	-	-	-	74.89/72.71
AdaBoost	*adjusted*	-	-	-	71.05/69.28
SVC		-	-	-	73.20/72.41
MLP		-	-	-	71.58/70.17

Some of these rules could be further merged, e.g., rules (3) and (5). However, the decisions naturally also involve the same nontransparent attribute `ExternalRiskEstimate`. Unlike BRCG, however, we gather some further information. If the attribute `ExternalRiskEstimate` has no clear indication of a `good` or a `bad` credit score ($\geq$76 or $<$70, respectively), the attribute is disputable and the decision is based on two transparent attributes. Individuals are considered to have a good credit score if they appear on average more than

54 months in files and their last credit inquiry dates back by more than 1 month. Concretely, a credit score is `good` if AverageMInFile $\geq 54 \wedge$ MSinceMostRecentInqexcl7days < 1. Otherwise, it is considered `bad`.

These more insightful rules of RuleLoE are less complex than the alternatives with the single exception of BRCG with the positive class `bad`, which can lead to a slightly improved performance (Table 2). Consequently, black box models are, in general, not superior to glass box models in terms of their accuracy given our evaluated test cases, whereas the SVC shows only a slightly higher accuracy than the other models. Considering all aspects, RuleLoE outperformed the other methods as it contributes to knowledge discovery by providing simple and interpretable decision rules, thereby leveraging explainability.

4.2.3. The KEEL Dataset

For all KEEL-based experiments, we used a LoE ensemble of three experts with a maximum depth of 2. This restricted configuration was selected to prioritize the generation of succinct and interpretable rule systems, aligning with the research's emphasis on explanation-focused outcomes. The data were divided using a $^2/_3$ split for training and testing purposes. For a comparative analysis, a random forest classifier was again evaluated, adhering to identical ensemble size and tree depth constraints (see Table 4). Despite no dataset-specific fine-tuning being conduced, LoE demonstrates a reasonably good performance. Some of our experiments show a low assignment accuracy (e.g., for the data subsets marketing and texture) and, consequently, yield a RuleLoE with significantly less performance. This effect might be attributed to an insufficient depth of the related assignment trees or an insufficient feature reduction. The data subsets marketing, optdigits, and texture, which show an overall suboptimal performance, feature many classes (Table 1). Those cannot be fully expressed by trees having two levels of depth, naturally leading to misclassifications. On average, LoE and RuleLoE surpass the random forest classifier in accuracy, underscoring the possible benefits of LoE in producing interpretable and efficient rule-based systems within the constraints of the experimental setup. Therefore, future investigations should also address these observed distinctions in more detail, for which further models should be investigated.

Table 4. Test results for the KEEL dataset, comparing the accuracy of LoE, a derived RuleLoE, and a random forest classifier as pairs of train/test accuracies. Additionally, the number of rules and their average length is reported. Best average results are highlighted in bold.

Dataset	Random Forest (%)	LoE (%)	RuleLoE (%)	Assignment Accuracy (%)	# Rules (∅ Length)
banana	64.74/63.87	83.75/83.02	83.55/82.68	99.35/99.31	10 (2.20)
coil2000	94.01/94.08	94.18/93.99	94.01/94.08	99.83/99.91	4 (2.75)
marketing	27.25/27.97	32.48/32.91	18.35/18.06	28.29/26.87	1 (2.00)
mushroom	90.08/89.43	99.84/99.89	98.92/99.41	99.07/99.52	8 (3.62)
optdigits	55.35/56.93	65.34/63.83	52.88/51.27	57.24/58.22	8 (3.88)
page-blocks	93.97/93.47	95.69/94.57	94.27/93.58	98.01/98.28	5 (2.80)
phoneme	75.58/76.18	79.53/79.93	76.69/76.35	90.08/89.69	9 (3.11)
ring	76.38/74.53	76.32/74.12	73.26/71.25	84.79/84.77	9 (2.44)
satimage	71.12/71.05	82.16/83.57	77.38/78.30	83.86/84.42	10 (3.40)
texture	36.28/35.32	70.91/70.19	60.22/59.89	62.61/63.86	9 (3.33)
thyroid	94.63/93.98	97.57/98.15	98.11/98.36	98.13/98.40	5 (1.60)
twonorm	80.33/79.81	94.61/94.27	74.16/75.39	75.47/75.23	3 (3.00)
Average	71.64/71.38	**81.03/80.70**	75.15/74.88	81.39/81.54	6.75 (2.84)

5. Conclusions and Outlook

In this contribution, we introduced the *League of Experts (LoE)* framework within the context of explainable artificial intelligence and machine learning (*XAI*/*XML*). LoE is a particular instance of an ensemble learner that combines surrogate models in order to leverage explainability and possibilities for human interaction. Moreover, LoE is accompanied by *RuleLoE*, a derived rule set learner. By choosing the ensemble members—the *experts*—from a class of glass box models, LoE itself becomes a glass box model, which, as demonstrated, is competitive in performance to existing glass and black box models. Importantly, feature space reductions can be incorporated into the training process of LoE, reducing the complexity of derived explanations. This is contrary to existing methods that navigate through a high-dimensional feature space. Additionally, we would like to address hyperparameter optimization, in terms of both sound default values and the automatic tuning of ensemble members based on the provided dataset and training performance.

Consequently, many possibilities to improve LoE and RuleLoE have yet to be further explored in regard to usability, performance, and presentation. Within our experiments, LoE was based on decision trees. However, LoE is not limited to them. Different models may be able to capture different attribute distributions, whereas a combination of different models might capture more complex distributions while furthermore retaining explainability. To guarantee the applicability of these methods, possibilities for user interaction (e.g., cutting nodes from decision trees, white- or blacklisting specific attributes to specific experts, or removing individual rules from RuleLoE) have to be integrated into accessible user interfaces, for which user studies will have to be conducted. Such an interface ideally facilitates the integration of intuitive human understanding of a concrete application into the corresponding machine learning pipeline. Proper user interactivity potentially reduces hidden biases and improves causal decision mining. This introduction into the LoE framework is hence a starting point for future developments.

Author Contributions: R.V., K.A.S., and T.S. conducted this contribution's writing process and the related research project's implementation and evaluation with the help of R.M. in typesetting and finalizing this manuscript. R.V., K.A.S., and T.S. designed this contribution, whereas M.R., M.V., M.E., and K.A.S. designed and supervised the related research project. All authors have read and agreed to the published version of the manuscript.

Funding: This study was carried out in the group of junior scientists "Agile Publika", which is funded by the European Social Fund (ESF) and the Free State of Saxony, Germany. The work of Kristan Alexander Schneider was partially performed within the DFG project "SCH 1480/2-1".

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The project page of the *League of Experts (LoE)* framework can be found via its repositories under https://github.com/Mereep/loe, https://github.com/Mereep/rule_loe, and https://github.com/Mereep/HDTree, accessed on 2 April 2024 . Our used datasets are available with their related publications (see also Section 4).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ANN	artificial neural network
BRCG	Boolean decision rules via column generation
CART	classification and regression trees algorithm
DCS	dynamic classifier selection
DES	dynamic ensemble selection
DS	dynamic selection
DSEL	split of training data used in some MCS algorithms

ICE	individual conditional expectation plots
k-NN	*k*-nearest neighbors
LoE	League of Experts
MCS	multiple classifier system
ML	machine Learning
MLP	multilayer perceptron
OvR	one-versus-rest
PDP	partial dependence plots
RuleLoE	League of Experts rule set learner
SVC	support vector classifier
SVM	support vector machine
TreeSHAP	decision-tree-based SHAP
XAI	explainable artificial intelligence
XML	explainable machine learning

References

1. Maurer, M.; Gerdes, J.C.; Lenz, B.; Winner, H. *Autonomous Driving*; Springer: Berlin/Heidelberg, Germany, 2016.
2. Haynes, R.B.; Wilczynski, N.L. Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: Methods of a decision-maker-researcher partnership systematic review. *Implement. Sci. IS* **2010**, *5*, 12. [CrossRef] [PubMed]
3. Bennett Moses, L.; Chan, J. Algorithmic prediction in policing: Assumptions, evaluation, and accountability. *Polic. Soc.* **2018**, *28*, 806–822. [CrossRef]
4. Goodman, B.; Flaxman, S. European Union regulations on algorithmic decision-making and a "right to explanation". *AI Mag.* **2017**, *38*, 50–57. [CrossRef]
5. Deeks, A. The Judicial Demand for Explainable Artificial Intelligence. *Columbia Law Rev.* **2019**, *119*, 1829–1850.
6. Doshi-Velez, F.; Kortz, M.; Budish, R.; Bavitz, C.; Gershman, S.; O'Brien, D.; Schieber, S.; Waldo, J.; Weinberger, D.; Wood, A. Accountability of AI Under the Law: The Role of Explanation. *arXiv* **2017**, arXiv:1711.01134.
7. Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* **2019**, *1*, 206–215. [CrossRef] [PubMed]
8. Swartout, W.; Paris, C.; Moore, J. Explanations in knowledge systems: Design for explainable expert systems. *IEEE Expert* **1991**, *6*, 58–64. [CrossRef]
9. Paris, C.L. Generation and explanation: Building an explanation facility for the explainable expert systems framework. In *Natural Language Generation in Artificial Intelligence and Computational Linguistics*; Springer: Berlin/Heidelberg, Germany, 1991; pp. 49–82.
10. Confalonieri, R.; Coba, L.; Wagner, B.; Besold, T.R. A historical perspective of explainable Artificial Intelligence. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2021**, *11*, e1391. [CrossRef]
11. Longo, L.; Brcic, M.; Cabitza, F.; Choi, J.; Confalonieri, R.; Del Ser, J.; Guidotti, R.; Hayashi, Y.; Herrera, F.; Holzinger, A.; et al. Explainable artificial intelligence (XAI) 2.0: A manifesto of open challenges and interdisciplinary research directions. *arXiv* **2023**, arXiv:2310.19775.
12. Markus, A.F.; Kors, J.A.; Rijnbeek, P.R. The role of explainability in creating trustworthy artificial intelligence for health care: A comprehensive survey of the terminology, design choices, and evaluation strategies. *J. Biomed. Inform.* **2021**, *113*, 103655. [CrossRef]
13. Zini, J.E.; Awad, M. On the explainability of natural language processing deep models. *ACM Comput. Surv.* **2022**, *55*, 1–31. [CrossRef]
14. Band, S.S.; Yarahmadi, A.; Hsu, C.C.; Biyari, M.; Sookhak, M.; Ameri, R.; Dehzangi, I.; Chronopoulos, A.T.; Liang, H.W. Application of explainable artificial intelligence in medical health: A systematic review of interpretability methods. *Inform. Med. Unlocked* **2023**, *40*, 101286. [CrossRef]
15. Ali, S.; Abuhmed, T.; El-Sappagh, S.; Muhammad, K.; Alonso-Moral, J.M.; Confalonieri, R.; Guidotti, R.; Del Ser, J.; Díaz-Rodríguez, N.; Herrera, F. Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence. *Inf. Fusion* **2023**, *99*, 101805. [CrossRef]
16. Crook, B.; Schlüter, M.; Speith, T. Revisiting the performance-explainability trade-off in explainable artificial intelligence (XAI). In Proceedings of the 2023 IEEE 31st International Requirements Engineering Conference Workshops (REW), Hannover, Germany, 4–5 September 2023 ; IEEE: Piscataway, NJ, USA, 2023; pp. 316–324.
17. Hastie, T.; Tibshirani, R.; Friedman, J.; Franklin, J. The elements of statistical learning: Data mining, inference and prediction. *Math. Intell.* **2005**, *27*, 83–85.

18. Cruz, R.M.O.; Sabourin, R.; Cavalcanti, G.D.C. Dynamic classifier selection: Recent advances and perspectives. *Inf. Fusion* **2018**, *41*, 195–216. [CrossRef]
19. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]
20. Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; Pedreschi, D. A survey of methods for explaining black box models. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 1–42. [CrossRef]
21. Adadi, A.; Berrada, M. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access* **2018**, *6*, 52138–52160. [CrossRef]
22. Loyola-Gonzalez, O. Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view. *IEEE Access* **2019**, *7*, 154096–154113. [CrossRef]
23. Bodria, F.; Giannotti, F.; Guidotti, R.; Naretto, F.; Pedreschi, D.; Rinzivillo, S. Benchmarking and survey of explanation methods for black box models. *Data Min. Knowl. Discov.* **2023**, *37*, 1719–1778. [CrossRef]
24. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [CrossRef]
25. Canziani, A.; Paszke, A.; Culurciello, E. An Analysis of Deep Neural Network Models for Practical Applications. *arXiv* **2016**, arXiv:1605.07678.
26. Drucker, H.; Burges, C.J.C.; Kaufman, L.; Smola, A.J.; Vapnik, V. Support vector regression machines. In Proceedings of the Advances in Neural Information Processing Systems, 1997 ; pp. 155–161.
27. Freund, Y.; Schapire, R.E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.* **1997**, *55*, 119–139. [CrossRef]
28. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
29. Breiman, L.; Friedman, J.H.; Olshen, R.A.; Stone, C.J. *Classification And Regression Trees*; Routledge: London, UK, 2017.
30. Das, A.; Rad, P. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv* **2020**, arXiv:2006.11371.
31. Hassija, V.; Chamola, V.; Mahapatra, A.; Singal, A.; Goel, D.; Huang, K.; Scardapane, S.; Spinelli, I.; Mahmud, M.; Hussain, A. Interpreting black-box models: A review on explainable artificial intelligence. *Cogn. Comput.* **2024**, *16*, 45–74. [CrossRef]
32. N. S. Altman. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *Am. Stat.* **1992**, *46*, 175–185. [CrossRef]
33. Friedman, N.; Geiger, D.; Goldszmidt, M. Bayesian Network Classifiers. *Mach. Learn.* **1997**, *29*, 131–163. [CrossRef]
34. Lakkaraju, H.; Bach, S.H.; Leskovec, J. Interpretable Decision Sets: A Joint Framework for Description and Prediction. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 13–17 August 2016; KDD '16; pp. 1675–1684.
35. Clark, P.; Niblett, T. The CN2 Induction Algorithm. *Mach. Learn.* **1989**, *3*, 261–283. [CrossRef]
36. Cohen, W.W. Fast Effective Rule Induction. In Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, CA, USA, 9–12 July 1995; Morgan Kaufmann: Burlington, MA, USA, 1995; pp. 115–123.
37. Yang, H.; Rudin, C.; Seltzer, M. Scalable Bayesian Rule Lists. *arXiv* **2016**, arXiv:1602.08610.
38. Dash, S.; Gunluk, O.; Wei, D. Boolean Decision Rules via Column Generation. In *Advances in Neural Information Processing Systems 31*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc. : 2018; pp. 4655–4665.
39. Ribeiro, M.T.; Singh, S.; Guestrin, C. Why Should I Trust You? Explaining the Predictions of Any Classifier. *arXiv* **2016**, arXiv:1602.04938.
40. Ribeiro, M.T.; Singh, S.; Guestrin, C. Anchors: High-Precision Model-Agnostic Explanations. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
41. Lundberg, S.M.; Lee, S. A Unified Approach to Interpreting Model Predictions. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 4–9 December 2017; NIPS'17; pp. 4768–4777.
42. Molnar, C. *Interpretable Machine Learning: A Guide for Making Black Box Models Interpretable*; Lulu: Morisville, NC, USA, 2019.
43. Lundberg, S.M.; Erion, G.G.; Lee, S. Consistent Individualized Feature Attribution for Tree Ensembles. *arXiv* **2018**, arXiv:1802.03888.
44. Buhrmester, V.; Münch, D.; Arens, M. Analysis of Explainers of Black Box Deep Neural Networks for Computer Vision: A Survey. *arXiv* **2019**, arXiv:1911.12116.
45. Arrieta, A.B.; Díaz-Rodríguez, N.; Ser, J.D.; Bennetot, A.; Tabik, S.; Barbado, A.; García, S.; Gil-López, S.; Molina, D.; Benjamins, R.; et al. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. *Inf. Fusion* **2019**, *58*, 82–115. [CrossRef]
46. Fürnkranz, J.; Gamberger, D.; Lavrač, N. *Foundations of Rule Learning*; Springer Publishing Company, Incorporated: Berlin/Heidelberg, Germany, 2014.
47. Gunning, D. DARPA's Explainable Artificial Intelligence (XAI) Program. In Proceedings of the 24th International Conference on Intelligent User Interfaces, New York, NY, USA, 16–20 March 2019; IUI '19; p. ii.
48. Kotsiantis, S.B. Decision trees: A recent overview. *Artif. Intell. Rev.* **2013**, *39*, 261–283. [CrossRef]
49. Freitas, A.A. Comprehensible classification models. *ACM SIGKDD Explor. Newsl.* **2014**, *15*, 1–10. [CrossRef]
50. Alizadeh, R.; Allen, J.K.; Mistree, F. Managing computational complexity using surrogate models: A critical review. *Res. Eng. Des.* **2020**, *31*, 275–298. [CrossRef]
51. Heider, K.G. The Rashomon effect: When ethnographers disagree. *Am. Anthropol.* **1988**, *90*, 73–81. [CrossRef]

52. Dong, X.; Yu, Z.; Cao, W.; Shi, Y.; Ma, Q. A survey on ensemble learning. *Front. Comput. Sci.* **2020**, *14*, 241–258. [CrossRef]
53. Mienye, I.D.; Sun, Y. A survey of ensemble learning: Concepts, algorithms, applications, and prospects. *IEEE Access* **2022**, *10*, 99129–99149. [CrossRef]
54. Arya, V.; Bellamy, R.K.E.; Chen, P.-Y.; Dhurandhar, A.; Hind, M.; Hoffman, S.C.; Houde, S.; Liao, Q.V.; Luss, R.; Mojsilović, A.; et al. One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques. *arXiv* 2019, arXiv:1909.03012.
55. Alvarez-Melis, D.; Jaakkola, T.S. Towards Robust Interpretability with Self-Explaining Neural Networks. *arXiv* **2018**, arXiv:1806.07538.
56. Geurts, P.; Ernst, D.; Wehenkel, L. Extremely randomized trees. *Mach. Learn.* **2006**, *63*, 3–42. [CrossRef]
57. Safavian, S.R.; Landgrebe, D. A survey of decision tree classifier methodology. *IEEE Trans. Syst. Man, Cybern.* **1991**, *21*, 660–674. [CrossRef]
58. Street, W.N.; Wolberg, W.H.; Mangasarian, O.L. Nuclear feature extraction for breast tumor diagnosis. In Proceedings of the Biomedical Image Processing and Biomedical Visualization, San Jose, CA, USA, 1–4 February 1993; Volume 1905, pp. 861–870.
59. Wei, D.; Dash, S.; Gao, T.; Gunluk, O. Generalized Linear Rule Models. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; Chaudhuri, K.; Salakhutdinov, R., Eds.; Proceedings of Machine Learning Research; Volume 97, pp. 6687–6696.
60. Chen, C.; Lin, K.; Rudin, C.; Shaposhnik, Y.; Wang, S.; Wang, T. An Interpretable Model with Globally Consistent Explanations for Credit Risk. *arXiv* **2018**, arXiv:1811.12615.
61. Derrac, J.; Garcia, S.; Sanchez, L.; Herrera, F. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *J. Mult.Valued Log. Soft Comput.* **2015**, *17*, 255–287.