Symbolic Regression Enhanced Decision Trees for Classification Tasks

Kei Sen Fong[1], **Mehul Motani**[1, 2]

[1]Department of Electrical and Computer Engineering, National University of Singapore
[2]N.1 Institute for Health, Institute for Digital Medicine (WisDM), Institute of Data Science, National University of Singapore
fongkeisen@u.nus.edu, motani@nus.edu.sg

Abstract

We introduce a conceptually simple yet effective method to create small, compact decision trees – by using splits found via Symbolic Regression (SR). Traditional decision tree (DT) algorithms partition a dataset on axis-parallel splits. When the true boundaries are not along the feature axes, DT is likely to have a complicated structure and a dense decision boundary. In this paper, we introduce SR-Enhanced DT (SREDT) – a method which utilizes SR to increase the richness of the class of possible DT splits. We evaluate SREDT on both synthetic and real-world datasets. Despite its simplicity, our method produces surprisingly small trees that outperform both DT and oblique DT (ODT) on supervised classification tasks in terms of *accuracy* and *F-score*. We show empirically that SREDTs decrease inference time (compared to DT and ODT) and argue that they allow us to obtain more explainable descriptions of the decision process. SREDT also performs competitively against state-of-the-art tabular classification methods, including tree ensembles and deep models. Finally, we introduce a local search mechanism to improve SREDT and evaluate it on 56 PMLB datasets. This mechanism shows improved performance on 77.2% of the datasets, outperforming DT and ODT. In terms of F-Score, *local* SREDT outperforms DT and ODT in 82.5% and 73.7% of the datasets respectively and in terms of inference time, *local* SREDT requires 25.8% and 26.6% less inference time than DT and ODT respectively.

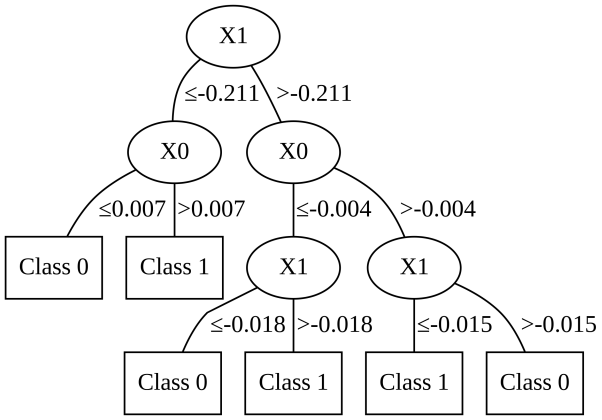
Introduction

Traditional decision tree (DT) and its ensembles have been shown to have state-of-the-art (SOTA) performance for tabular classification problems, despite advances in deep learning for tabular data (Borisov et al. 2021; Shwartz-Ziv and Armon 2022). Beyond quantitative performance, DT has been shown to have one of the best degrees of explainability among classification models as well (Xu et al. 2019; Kaya, Gurpinar, and Ali Salah 2017; Mahbooba et al. 2021; Blanco-Justicia and Domingo-Ferrer 2019; Došilović, Brčić, and Hlupić 2018). The structure of a single DT makes it easy to explain how a prediction is obtained. However, DTs with many leaves and large depth still suffer from poor explainability (Zhang et al. 2022), which is consistent with cognitive psychology theories which shows that explainability of a model can be evaluated by the number of cognitive visual chunks (e.g., number of DT nodes) (Lage et al. 2019; Abdul et al. 2020).

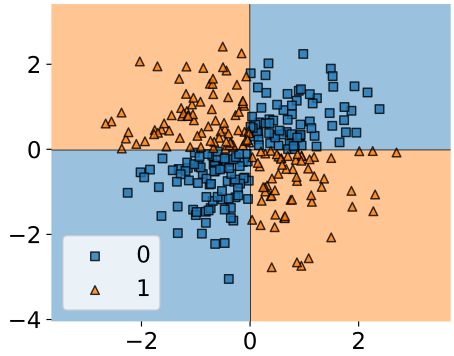
Traditional DT algorithms partition a dataset on axis-parallel splits. In a tree node with n data points and d features (i.e., $[x_1, x_2, ...x_d]$), each split is in the form of $x_i > k$, representing an axis-parallel hyperplane. The small class of possible splits may lead to bloated and less explainable trees, such as the tree shown in Fig. 1a when solving the XOR classification task from MLxtend (Raschka 2018) shown in Fig. 1b. This weakness becomes more apparent in real-world applications with a large number of features, d, where it is much harder to prune and simplify the tree. Furthermore, unlike the XOR classification task, decision boundaries in real-world problems are rarely axis-parallel.

To address the problems above, oblique DT (ODT) (Heath, Kasif, and Salzberg 1993; Murthy, Kasif, and Salzberg 1994; Ganaie, Tanveer, and Suganthan 2020; Montañana, Gámez, and Puerta 2021), which utilizes oblique linear hyperplanes to partition the dataset, has been proposed. ODT is also known as multivariate DT in other literature (Brodley and Utgoff 1995). The d-dimensional hyperplanes are stored in the form $H(x) = h_{d+1} + \sum_{i=1}^{d} h_i x_i$, where $H = \{h_1, h_2, \ldots, h_{d+1}\}$ is the hyperplane, $x = (x_1, x_2, \ldots, x_d)$ is a datapoint and h_{d+1} represents the constant term (Heath, Kasif, and Salzberg 1993). ODTs have been shown to be both smaller and more accurate than DT (Murthy, Kasif, and Salzberg 1994). However, this comes at the cost of having a splitting rule involving all d input features, making ODT more susceptible to adversarial data noise. This also potentially makes ODT less explainable (Abdul et al. 2020; Lage et al. 2019).

Our proposed Symbolic Regression (SR)-Enhanced DT (SREDT) addresses the weaknesses of DT and ODT. SREDT utilizes SR to discover a richer class of splitting rules, resulting in single splits that would have otherwise required multiple splits by DT and ODT. For instance, in the earlier example of XOR classification by DT (shown in Fig. 1), a single SR split shown in Fig. 2a yields the true decision boundary shown in Fig. 2b. SREDT is also highly explainable – achieved through compactness (Zhang et al. 2022; Lage et al. 2019; Abdul et al. 2020) and rich class of functions (Abdul et al. 2020). Furthermore, SREDT allows for short inference time.

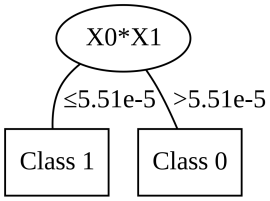


(a) DT for XOR problem.

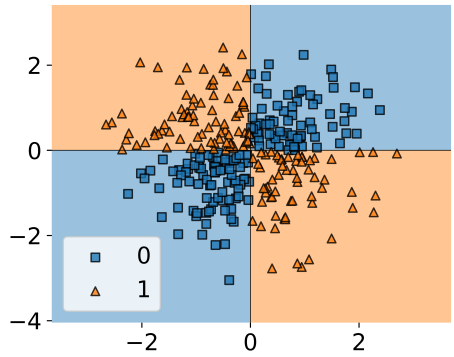


(b) Decision regions created by DT.

Figure 1: DT approach to XOR problem. X0 is the horizontal coordinate and X1 is the vertical coordinate.



(a) SREDT for XOR problem is compact.



(b) Decision regions created by SREDT.

Figure 2: SREDT approach to XOR problem. X0 is the horizontal coordinate and X1 is the vertical coordinate.

Algorithm 1: *SymbolicRegressor* Pseudo Code

Input: N, where N = set of classified instances, with D features, `CriteriaScoring` (e.g., Mean Squared Error) , $population\_size$, $generations$
Output: BestIndividual

1 $population \leftarrow$ `InitializePopulation`($population\_size$)
2 **for** $gen \leftarrow 1$ *to* $generations$ **do**
3 $population \leftarrow$ `EvaluateFitness`($population$, N, `CriteriaScoring`)
4 $population \leftarrow$ `SelectParents`($population$)
5 $population \leftarrow$ `Crossover`($population$)
6 $population \leftarrow$ `Mutate`($population$)
7 **end**
8 **return** *BestIndividual*($population$)

Contributions: In this paper, we propose the SREDT algorithm to enhance DT by utilizing SR and evaluate SREDT across a range of 65 datasets (3 Synthetic, 62 Real-World). Specific contributions include: (i) We show that SREDT is smaller (in depth, number of leaves and number of terms), faster in inference time, and performs better than DT and ODT. (ii) We show that SREDT has competitive performance with SOTA tabular classification models while still being explainable. (iii) We show that SREDT is innately robust to adversarial data noise, unlike DT and other models evaluated. (iv) We propose an update to SREDT to allow local search and demonstrate consistent performance improvement across a large range of 56 PMLB datasets.

Primer on SR

The field of applying machine learning to generate concise mathematical expressions to predict an output is known as SR. The expressions obtained from SR come in a compact and human-readable form that has fewer parameters than black-box models. These expressions allow for useful scientific insights by mere inspection. This property has led SR to be gradually recognized as a first-class algorithm in various scientific fields, including Physics (Udrescu and Tegmark 2020), Material Sciences (Wang, Wagner, and Rondinelli 2019; Sun et al. 2019) and Knowledge Engineering (Martinez-Gil and Chaves-Gonzalez 2020).

In the field of SR, genetic programming (GP) has become the most common paradigm to tackle the large search space of equations (Koza 1992; Schmidt and Lipson 2009; Augusto and Barbosa 2000; McKay, Willis, and Barton 1995). Even the best performing SR algorithms incorporate GP at the core (Petersen et al. 2019; Mundhenk et al. 2021). There are other non-GP approaches to SR, but most still utilize a variant of the evolutionary process as a central mechanism (TuringBot 2020; Kommenda et al. 2020).

The key steps for vanilla GP-SR (Koza 1992; Stephens 2016) are outlined in Algorithm 1, where steps 5 and 6 are the main steps involved in evolving top equations. SR is mostly used for regression tasks in literature, thus mean squared error is a common criteria for fitness. In this paper, we explain how SREDT modifies this criteria to classify.

Related Work

Classification with Standalone SR. There exist approaches utilizing SR as a standalone method for classification (Korns and May 2019; Ingalalli et al. 2014; Korns 2018; Muñoz, Silva, and Trujillo 2015; Korns 2017). These methods can obtain competitive performance, but are complex and not parsimonious. Further, the lack of an "if-else" structure in traditional SR algorithms makes it difficult for SR to partition datasets, which DTs do very well. We note that with a naive addition of an "if-else" function in the primitive set without adopting techniques used in the DT algorithm, the resultant model is likely to produce many incremental imbalanced splits. All these reasons provide motivation for combining SR and DT, which we do in this paper.

SOTA Oblique DT. ODT utilizes enhancements such as support vector machines (SVM) and neural networks (NN) to navigate the search space of oblique decision hyperplanes (Costa and Pedreira 2023). Stree is an SVM-enhanced ODT that supports multi-class classification and does so with a single tree (Montañana, Gámez, and Puerta 2021), making it a suitable benchmark against our work. In addition, experimental results have shown that Stree outperforms other variants of ODT (Montañana, Gámez, and Puerta 2021), making it our top choice SOTA ODT for benchmarking.

SOTA Tabular Classfication Models. SOTA deep learning models for tabular classification include both NODE (Popov, Morozov, and Babenko 2019) and TabNet (Arık and Pfister 2021). However, despite the progress in deep models, it has been shown that traditional models, especially tree ensemble models, have consistently performed better (Borisov et al. 2021; Shwartz-Ziv and Armon 2022). Thus, in our comparison analysis, we also include the performance of the following models: AdaBoost (Friedman, Hastie, and Tibshirani 2000), CatBoost (Prokhorenkova et al. 2018), ExtraTrees (Geurts, Ernst, and Wehenkel 2006), Extreme Gradient Boosting (XGBoost) (Chen and Guestrin 2016), Gradient Boosting (Friedman 2001), Light Gradient Boosting Machine (LightGBM) (Ke et al. 2017), Random Forest (Breiman 2001) and SVM (Cortes and Vapnik 1995).

SREDT Methodology

Both DT (with few leaves and low depth) and SR are regarded by machine learning practitioners to have high explainability (Quade, Isele, and Abel 2020; Xu et al. 2019). Hence, there is strong motivation for a hybrid of both approaches to achieve a high performing and still highly explainable solution. The weaving of SR into the DT algorithm is seamless – DT can utilize SR to discover a richer class of decision boundaries at each tree split. Using SR, SREDT can capture the complex relationship between variables which would have otherwise required many more splits or would have been impossible with existing DT variants.

SREDT Algorithm

Main Algorithm. We first adapt the classification and regression trees (CART) algorithm (Breiman et al. 2017; Lewis 2000) into our method, as outlined in Algorithm 2. Our proposed SREDT is shown in Algorithm 3, in which we propose that steps 2 to 8 of Algorithm 2 are replaced by SR. Instead of iterating through individual features, we propose that SR be utilized to find an equation for splitting. SR takes the `CriteriaScoring` function (e.g., Gini Impurity) for use as a fitness function, as shown in Algorithm 3.

Algorithm 2: Decision Tree Pseudo Code

Input: N, where N = set of classified instances, with D features, `CriteriaScoring` (e.g., Gini Impurity)
Output: Decision Tree

1 **Procedure** `BuildTree`**:**
2 $bestCriteriaScore \leftarrow null$
3 $bestSplit \leftarrow null$
4 **for** $feature \in D$ **do**
5 $CriteriaScore \leftarrow$ `CriteriaScoring`$(N, feature)$
6 **if** `BetterThan`$(CriteriaScore, bestScore)$ **then**
7 $bestScore \leftarrow CriteriaScore$
8 $bestSplit \leftarrow feature$
9 $N_{left}, N_{right} \leftarrow$ `Partition`$(N, bestSplit)$
10 `BuildTree`(N_{left})
11 `BuildTree`(N_{right})

Algorithm 3: SREDT Pseudo Code

Input: N, where N = set of classified instances, with D features, `CriteriaScoring` (e.g., Gini Impurity)
Output: SREDT

1 **Procedure** `BuildTree`**:**
2 $bestSplit \leftarrow$ `SymbolicRegressor`$(N,$ `CriteriaScoring`$)$
3 $N_{left}, N_{right} \leftarrow$ `Partition`$(N, bestSplit)$
4 `BuildTree`(N_{left})
5 `BuildTree`(N_{right})

In SREDT (Algorithm 3), `SymbolicRegressor` (Algorithm 1) generates candidate expressions (e.g., $X0 * X1$) for use as a splitting rule. In assessing each candidate expression, `SymbolicRegressor` calculates the expression's value for each of the classified instances it is given (based on the `CriteriaScoring` function) and finds the best threshold value (e.g., $X0 * X1 < 0.05$, where 0.05 is the best threshold for the candidate expression $X0 * X1$). In our implementation, SR evolves the candidate expressions based on the evolutionary functions, such as mutation, used in programming-based SR (GP-SR) (Koza 1992). The best scoring candidate expression (evaluated on its best possible threshold value), is then designated as the $bestSplit$.

SREDT Variants. We also experiment with 3 variants of SREDT to explore improving performance and to test the robustness of SREDT. The 3 variants are:

(i) Variant 1: Pretrained SREDT (P-SREDT), in which the initial population of equations (Step 2 of Algorithm 1) is generated from a recurrent neural network trained on a given equation corpus. In this paper, we select the well-known AI-Feynman equation corpus (Udrescu and Tegmark 2020).

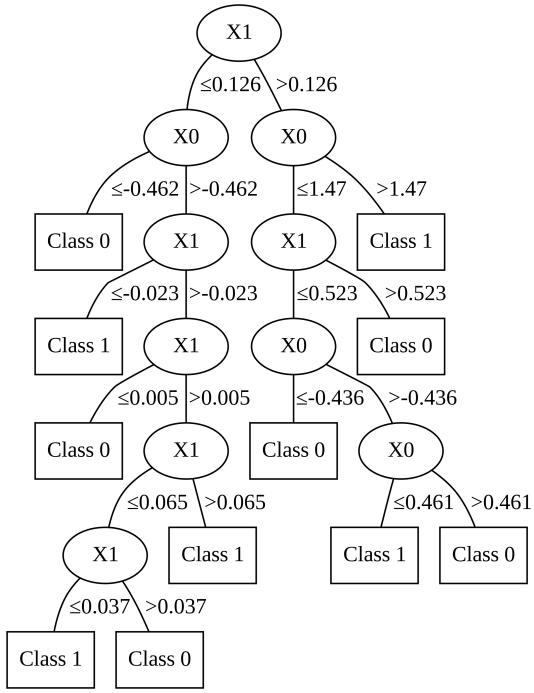
(ii) Variant 2: Lookahead SREDT (L-SREDT), in which each split is evaluated based on its performance an additional depth later (inspired by single lookahead DT (Murthy and Salzberg 1995)).
(iii) Variant 3: Local SREDT For readability, we will only discuss this variant in the further experiments sub-section.

Datasets and Experiment Set-up

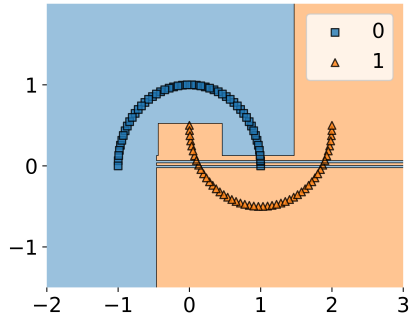
We evaluate SREDT performance on both synthetic and real-world data. The synthetic experiments serve to illustrate the improvement of SREDT over DT on easy-to-visualize examples. Real-world experiments are then used, comparing against a larger variety of methods, including ODT and SOTA tabular classification models.
Synthetic Dataset Experiments. We demonstrate through experiments the ability of SREDT to solve 3 public synthetic classification problems from MLxtend (Raschka 2018) – XOR, Half-Moons and Concentric Circles. We allow for unlimited tree depth for both DT and SREDT, and require 100% classification accuracy. Gini impurity is used as the `CriteriaScoring` function utilized in Algorithm 2 and Algorithm 3 and is given by:

$$Gini(t) = \frac{n_t^l}{n_t}\left[1 - \sum_{i=1}^{c}\left(\frac{n_{t_i}^l}{n_t^l}\right)^2\right] + \frac{n_t^r}{n_t}\left[1 - \sum_{i=1}^{c}\left(\frac{n_{t_i}^r}{n_t^r}\right)^2\right],$$

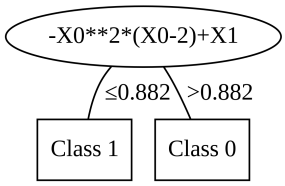
where t is the current node, c is the number of classes, n_t, n_t^l, n_t^r represents the number of data points at the current node, the left child of the current node and the right child of the current node respectively. $n_{t_i}^l, n_{t_i}^r$ are the number of data points belonging to class i that reached to the left child and right child of the current node, t respectively. We set our SR primitive function set to $\{add, mul, sub, div\}$. We also set other SR hyperparameters as follows: the parsimony coefficient to 0.001, number of generations to 40, population size to 400 and tournament size to 200.
Real-World Dataset Experiments. To evaluate the performance of SREDT on real-life datasets, we extensively study 6 commonly used tabular classification datasets: the *Cancer, Diabetes, Forest Type Mapping, Heart Disease, Iris and Raisin* datasets (Dua and Graff 2019; Smith et al. 1988; Ali 2020). We benchmark our results against a diverse range of models for tabular classification, including ensemble methods: AdaBoost, CatBoost, ExtraTrees, XGBoost, Gradient Boosting, LightGBM, Random Forest, SVM. We also benchmark against SOTA deep models for tabular classification: NODE (Popov, Morozov, and Babenko 2019), TabNet (Arık and Pfister 2021), though we note their classification performance has been shown to be consistently lower than that of traditional models (Borisov et al. 2021; Shwartz-Ziv and Armon 2022). Gini impurity is used as the `CriteriaScoring` function. SREDT is given a max depth of $log_2(n)+1$, where n is the number of unique labels. We also use the same settings as the synthetic dataset experiments. Performance is evaluated on a 60-10-30 random train-validation-test split and the hyper-parameters of the other models are tuned using the validation set. To demonstrate the broad applicability of SREDT, we also evaluate its performance on set of 56 other benchmark datasets.



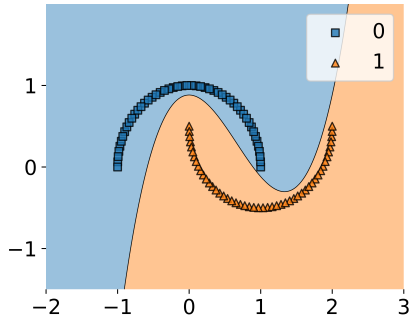
(a) DT for Half-Moon problem.



(b) Decision regions created by DT.



(c) SREDT for Half-Moon problem.



(d) Decision regions created by SREDT.

Figure 3: DT and SREDT approach to Half-Moon problem.

Model	CANCER	DIABETES	FOREST_TYPES	HEART_DISEASE	IRIS	RAISIN
SREDT	**0.9644** (0.011)	0.7403 (0.027)	0.8363 (0.034)	0.7915 (0.041)	0.9449 (0.034)	0.8598 (0.019)
P-SREDT	0.9546 (0.014)	0.7307 (0.028)	0.8153 (0.040)	0.7753 (0.038)	0.9422 (0.023)	0.8559 (0.017)
L-SREDT	0.9623 (0.013)	0.7472 (0.027)	0.8299 (0.035)	0.7926 (0.047)	0.9511 (0.029)	0.861 (0.019)
Local SREDT	0.9637 (0.012)	**0.7497** (0.021)	**0.8398** (0.028)	**0.8022** (0.040)	0.9472 (0.013)	**0.8626** (0.019)
ODT	0.9632 (0.012)	0.7166 (0.032)	0.7611 (0.04)	0.7299 (0.052)	**0.962** (0.028)	0.8309 (0.024)
DT	0.9345 (0.014)	0.699 (0.027)	0.7572 (0.041)	0.7368 (0.045)	0.9471 (0.03)	0.8086 (0.024)
AdaBoost	0.9538 (0.014)	0.7438 (0.025)	0.6704 (0.081)	0.7868 (0.041)	0.9409 (0.031)	0.8547 (0.019)
CatBoost	0.9666 (0.011)	0.7657 (0.023)	0.8528 (0.031)	0.8278 (0.039)	0.9498 (0.029)	0.8649 (0.017)
ExtraTrees	0.9667 (0.011)	0.7428 (0.024)	0.8543 (0.029)	0.8319 (0.038)	0.9502 (0.028)	0.8514 (0.017)
XGBoost	0.956 (0.012)	0.7402 (0.023)	0.8443 (0.03)	0.8091 (0.039)	0.948 (0.028)	0.8517 (0.018)
Gradient Boosting	0.9558 (0.012)	0.7545 (0.024)	0.824 (0.035)	0.7988 (0.04)	0.9496 (0.031)	0.859 (0.017)
LightGBM	0.9606 (0.012)	0.7423 (0.022)	0.8464 (0.03)	0.8073 (0.041)	0.9431 (0.033)	0.85 (0.019)
Random Forest	0.9677 (0.011)	0.7581 (0.021)	0.8359 (0.034)	0.8279 (0.039)	0.9507 (0.03)	0.8599 (0.018)
SVM	0.9534 (0.013)	0.5554 (0.13)	0.745 (0.091)	0.5936 (0.1)	0.794 (0.12)	0.5461 (0.091)
NODE	0.9683 (0.012)	0.7416 (0.052)	0.8001 (0.037)	0.7925 (0.051)	0.9447 (0.035)	0.867 (0.018)
TabNet	0.7575 (0.024)	0.7034 (0.029)	0.7947 (0.037)	0.7549 (0.042)	0.9382 (0.035)	0.8286 (0.022)

Table 1: *Accuracy*: Mean Accuracy (Standard Deviation [SD] in brackets) of SREDT variants, 8 Traditional Models and 2 Deep Models evaluated on 6 datasets. Results are for 70-30 random train-test splits averaged over 100 experiments per dataset per model. Best model is underlined and best single tree is bolded. See Supplementary Appendix for results on 56 datasets.

Results and Further Experiments

Synthetic Dataset Experiments

XOR Classification. As seen in Fig. 1a and Fig. 2a, DT has a depth of 3 and a total of 6 leaves, whereas SREDT has only a depth of 1 and a total of 2 leaves for the XOR classification task. For DT, the first split happens at the threshold value of -0.211 for X1, which is not visible in Fig. 1b. The reason is that based on the DT algorithm in Algorithm 2, it is not clear which horizontal or vertical split is best since every possible split would result in partitions that have an approximately even distribution, resulting in similar Gini impurity values regardless of the split chosen. By restricting the splits to be dependent on only a single feature, the DT makes a non-ideal first split which does not appear on the final decision region plot. SREDT utilizes SR to evaluate a variety of splits, experimenting with linear and non-linear combinations of features. This allows it to accurately determine a suitable non-linear boundary within the first and only split, resulting in a shorter, compact tree with less leaves. *This demonstrates the ability of SREDT to create compact trees by exploring single non-linear splits.*

Half-Moon Classification. In the Half-Moon problem, DT and SREDT show different decision boundaries as seen in Fig. 3b and Fig. 3d respectively. The DT has a depth of 6 and a total of 11 leaves, whereas the SREDT has only a depth of 1 and a total of 2 leaves as shown in Fig. 3a and Fig. 3c respectively. Unlike the XOR problem, the decision boundaries created by DT and SREDT are distinctively different. *This demonstrates the ability of SREDT to create compact trees by exploring single non-linear splits that would have otherwise not have been found even with an unlimited amount of axis-parallel splits.*

Concentric Circles Classification. Here, DT and SREDT show different decision boundaries in Supplementary Appendix Fig. 1b & 1d respectively. The decision boundary found by DT comprises of 4 linear splits visually, whereas the decision boundary found by SREDT comprise of only a single non-linear split. We note again that the DT has a depth of 5 and a total of 11 leaves, whereas the SREDT has only a depth of 1 and a total of 2 leaves as shown in Supplementary Appendix Fig. 1a & 1c respectively. In this case, we recognize that the single split made by SREDT resembles that of the equation of an ellipse. This demonstrates again the ability of SREDT to create compact trees by exploring single non-linear splits that would have otherwise have not been possible even with a large combination of axis-parallel splits. *The elliptical equation found by SREDT in Supplementary Appendix Fig. 1c also demonstrates how non-linear splits can provide insights into the underlying ground truth (i.e., equation of circle) that DT cannot.*

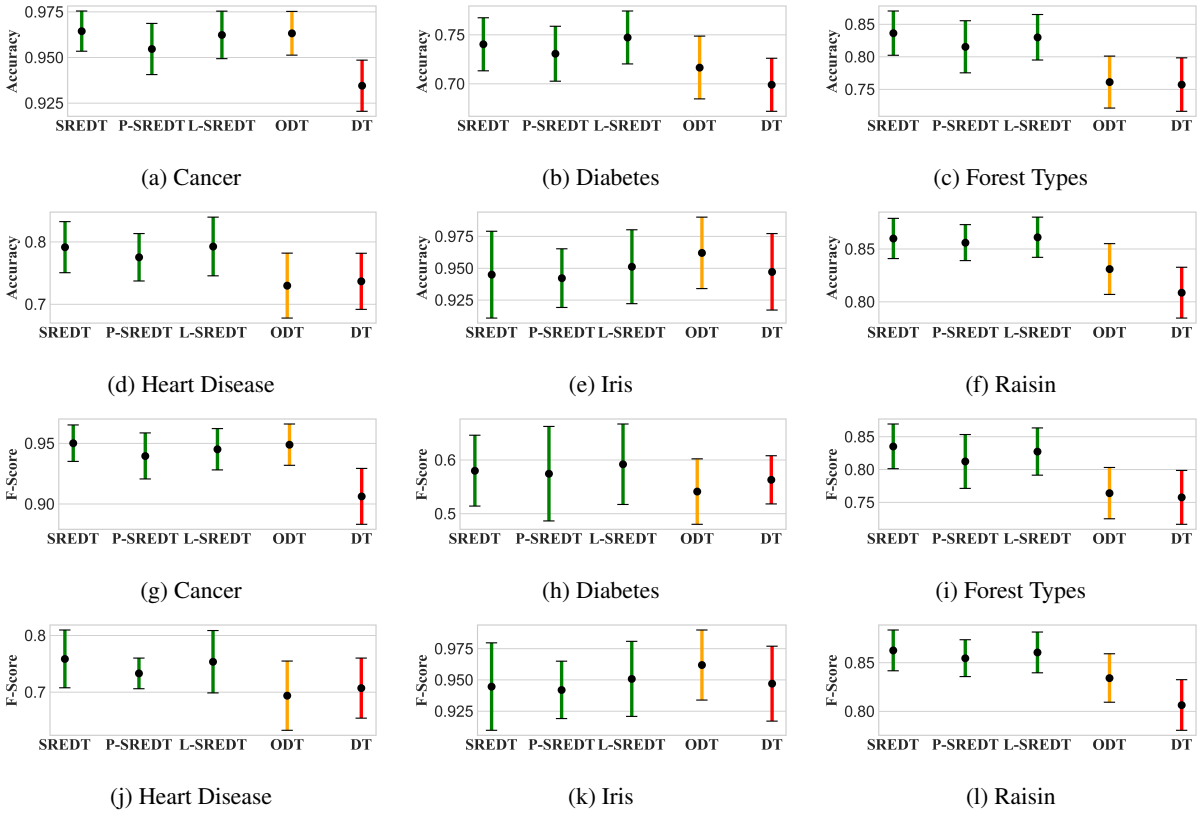
Common Observations. As demonstrated in the 3 synthetic datasets, the strength of SREDT is in creating boundaries that would have required many more linear splits, or would not have been possible even with an infinite combination of linear splits. Thus, SREDT is shorter and has less leaves, increasing the explainability of the model (Zhang et al. 2022; Lage et al. 2019; Abdul et al. 2020). However, one limitation of SREDT is that the distance between the boundary and the data points are not taken into account, unlike models such as SVM. For instance, in the concentric circles problem, SREDT found an elliptical boundary in the experiment instead of a circular boundary. To address this, we implement an improvement to SREDT using local search with squared hinge loss in Further Experiments.

Real-World Dataset Experiments

We present the performance of SREDT on real-world datasets, in terms of *accuracy* and *F-score* (i.e., harmonic mean of precision and recall). We compare SREDT with DT, ODT, and other SOTA methods (i.e., ensemble, deep models) on 6 real-world datasets previously described, and summarize the results in Tables 1 & 2. We also propose an

Model	CANCER	DIABETES	FOREST_TYPES	HEART_DISEASE	IRIS	RAISIN
SREDT	**0.9501** (0.015)	0.5801 (0.066)	0.8352 (0.034)	0.7585 (0.051)	0.9447 (0.035)	**0.8626** (0.021)
P-SREDT	0.9396 (0.019)	0.5744 (0.088)	0.8122 (0.041)	0.733 (0.027)	0.9420 (0.023)	0.8546 (0.019)
L-SREDT	0.9451 (0.017)	0.592 (0.075)	0.8273 (0.036)	0.7535 (0.055)	0.9508 (0.030)	0.8605 (0.021)
Local SREDT	0.9411 (0.015)	**0.5974** (0.075)	**0.8389** (0.031)	**0.7621** (0.061)	0.9451 (0.022)	0.8603 (0.015)
ODT	0.9489 (0.017)	0.5412 (0.061)	0.7640 (0.039)	0.6937 (0.061)	**0.9619** (0.028)	0.8342 (0.025)
DT	0.9063 (0.023)	0.563 (0.045)	0.7576 (0.041)	0.7071 (0.053)	0.947 (0.03)	0.8064 (0.026)
AdaBoost	0.9335 (0.02)	0.608 (0.039)	0.634 (0.088)	0.7537 (0.05)	0.9406 (0.031)	0.8584 (0.021)
CatBoost	0.9529 (0.017)	0.6366 (0.037)	0.8508 (0.031)	0.8015 (0.052)	0.9497 (0.029)	0.8686 (0.018)
ExtraTrees	0.9531 (0.016)	0.5801 (0.043)	0.8519 (0.03)	0.8037 (0.049)	0.9502 (0.028)	0.8542 (0.019)
XGBoost	0.9381 (0.018)	0.6142 (0.036)	0.8425 (0.031)	0.7813 (0.05)	0.9479 (0.029)	0.8532 (0.02)
Gradient Boosting	0.9369 (0.018)	0.6249 (0.035)	0.8223 (0.036)	0.7681 (0.053)	0.9494 (0.031)	0.862 (0.018)
LightGBM	0.9441 (0.018)	0.6133 (0.034)	0.844 (0.031)	0.7789 (0.053)	0.943 (0.033)	0.8522 (0.02)
Random Forest	0.9545 (0.016)	0.6126 (0.038)	0.8337 (0.035)	0.8009 (0.05)	0.9505 (0.03)	0.863 (0.019)
SVM	0.933 (0.021)	0.3582 (0.18)	0.7303 (0.11)	0.5017 (0.24)	0.7556 (0.16)	0.3849 (0.34)
NODE	0.9552 (0.017)	0.4951 (0.27)	0.799 (0.038)	0.7278 (0.098)	0.9446 (0.035)	0.8709 (0.019)
TabNet	0.7473 (0.027)	0.5674 (0.046)	0.7937 (0.037)	0.7228 (0.053)	0.9382 (0.036)	0.8292 (0.024)

Table 2: *F-Score*: Mean F-Score (SD in brackets) of SREDT variants, ODT, DT, 8 Traditional Models and 2 Deep Models evaluated on 6 datasets. Results are for 70-30 random train-test splits and averaged over 100 experiments per dataset per model. Best model is underlined and best single tree is bolded. See Supplementary Appendix for results on 56 datasets.



(a) Cancer (b) Diabetes (c) Forest Types

(d) Heart Disease (e) Iris (f) Raisin

(g) Cancer (h) Diabetes (i) Forest Types

(j) Heart Disease (k) Iris (l) Raisin

Figure 4: *Accuracy* and *F-Score* plots with error bars based on Table 1 & 2. SREDT shows outperformance.

improved SREDT, called local SREDT, and study its performance on a larger suite of 56 datasets.

SREDT Comparison with DT and ODT

(i) SREDT performs better than DT and ODT in *accuracy* and *F-score* performance. Fig. 4 is derived from Table 1 and Table 2. As can be seen, the performance of SREDT (and its variants) is superior to ODT in both *accuracy* (see Fig. 4a to 4f) and *F-score* (see Fig. 4g to 4l). This outperformance is greatest when comparing SREDT to DT, with little to no overlap in the error bars.

(ii) SREDT is smaller and more compact than DT and ODT. To compare the sizes of DT & ODT to SREDT, we take

	DT to SREDT (Depth)	DT to SREDT (No. of Leaves)	DT to SREDT (No. of Terms)	ODT to SREDT (Depth)	ODT to SREDT (No. of Leaves)	ODT to SREDT (No. of Terms)
CANCER	(454±48)%	(646±71)%	(242±38)%	(192±51)%	(167±51)%	(497±128)%
DIABETES	(732±95)%	(2400±142)%	(964±47)%	(1810±328)%	(1780±328)%	(4380±952)%
FOREST_TYPES	(501±59)%	(812±78)%	(317±45)%	(432±102)%	(407±101)%	(934±303)%
HEART_DISEASE	(416±49)%	(813±74)%	(287±22)%	(642±151)%	(617±150)%	(1710±345)%
IRIS	(237±42)%	(184±37)%	(118±21)%	(151±7)%	(126±7)%	(403±18)%
RAISIN	(738±115)%	(1980±113)%	(1020±63)%	(3530±652)%	(3510±652)%	(2600±117)%

Table 3: *Depth ratio, leaves ratio and terms ratio (in %)* of DT and ODT compared to SREDT across 6 datasets. Results are for 70-30 random train-test splits and averaged over 100 experiments per dataset per model. Higher is better for SREDT.

	DT to SREDT	ODT to SREDT
CANCER	(125±1.4)%	(119±2.1)%
DIABETES	(204±3.2)%	(251±1.2)%
FOREST_TYPES	(114±2.5)%	(106±0.5)%
HEART_DISEASE	(98±7.1)%	(104±1.4)%
IRIS	(138±1.5)%	(134±2.9)%
RAISIN	(186±2.2)%	(195±2.3)%

Table 4: *Inference time ratio (in %)* of DT & ODT to SREDT on 6 datasets. Results are for 70-30 random train-test splits and averaged over 100 experiments per dataset per model. Higher is better for SREDT.

	RPC			**RGN**		
	SREDT	DT	AOM	SREDT	DT	AOM
I	**41%**	100%	100%	**53%**	92%	100%
II	**12%**	91%	100%	**23%**	91%	100%
III	**11%**	98%	100%	**9%**	97%	100%
IV	**29%**	89%	100%	**70%**	100%	100%
V	**12%**	66%	100%	**12%**	66%	100%
VI	**4%**	98%	100%	**6%**	99%	100%

Table 5: *Percentage of trees 'tricked'* by noisy columns. Results are for 70-30 random train-test split and averaged over 100 experiments per dataset per model. Lower is better. The numerals, I to VI, represent the datasets in the order of cancer, diabetes, forest types, heart disease, iris and raisin.

the data from the experiments in Table 1 and Table 2, and compute the ratio of the tree depths (in %) of DT & ODT to SREDT. We do the same for number of leaves in the trees and the number of terms (i.e., splits, operands and operators) in the trees. As an example, the number of terms in Fig. 1a is 5, and the number of terms in Fig. 2a is 3. We tabulate the depth ratio, leaves ratio and terms ratio in Table 3. From these results, we see that SREDT is consistently smaller than both DT and ODT – while having better *accuracy* and *F-score* performance (see Tables 1 & 2).

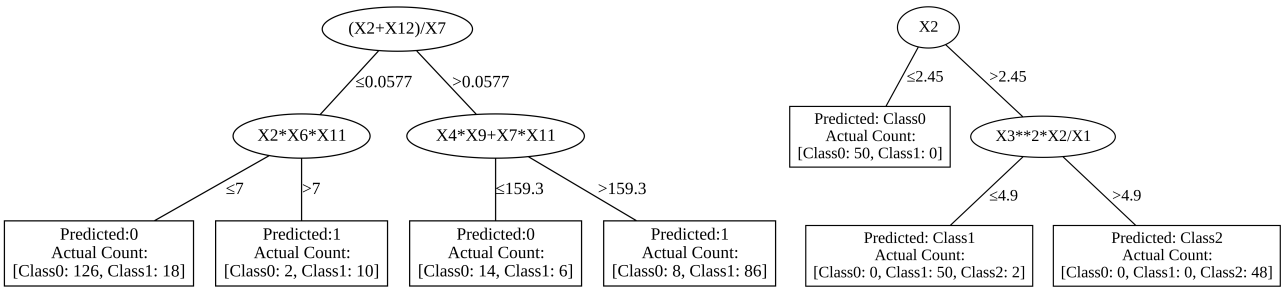
(iii) SREDT has a faster inference time than DT and ODT. We also measure the testing set inference time from the experiments in Table 1 and Table 2. In Table 4, we tabulate the inference time ratio (in %) of DT and ODT compared to SREDT. SREDT consistently provides faster inference time than both DT and ODT. The improvement in inference time for SREDT can be explained by the much lower depth of SREDT compared to both DT and ODT, as discussed above. However, since SREDT allows for more complex splitting functions, which requires more computation, the time taken to compute each individual split would likely be more than that of DT, which explains why the depth ratio is higher than the inference time ratio. Note that SREDT is still better in both ratios, just that it better to a larger extent in terms of depth ratio.

(iv) SREDT Limitations: In the Iris dataset, the performance of SREDT is not better than that of DT and ODT. This may be counterintuitive since SREDT is a more generalized version of DT and ODT. However, we should note that although it is possible for SREDT to discover the same tree as DT and ODT, it is not guaranteed to do so since the tree building process in Algorithm 3 is greedy. Regarding training, it is guaranteed that training time for SREDT would be higher since more candidate functions are to be evaluated to select the best split. Although SREDT is a GP algorithm and hence can technically be said to have time complexity of $O(1)$, GP based methods typically take a longer time to run in order to explore a sufficiently diverse range of solutions. In our experiments, it takes 1570 times longer to train an SREDT compared to DT. However, SREDT show key strengths in multiple other aspects, which mitigates the longer training time for several real-world problems. For instance, there is consistent reductions in inference time for SREDT, which is more important than training time for time-critical applications. Also, a criticism of GP-based algorithms is that it does not scale well with the size of the dataset. However, performance on large datasets should not be the be-all and end-all to evaluating a machine learning method. Many real-world problems in the industry come in the form of small tabular datasets and SREDT is well-poised to tackle such problems.

SREDT Comparison with SOTA Tabular Classifiers

(i) SREDT performs competitively compared to both traditional models and deep models in terms of *accuracy*, as recorded in Table 1, with overlapping standard deviation error bars. Similar trends are observed in the *F-score* as observed in Table 2.

(ii) SREDT also has robust innate feature selection. SREDTs are unique since they tend to select only a subset of features in the eventual tree. Compared to other machine learning models (including ODT, and to a smaller extent, DT), SREDT is highly selective of features. This is an advantage in critical real-world problems, where there may be large perturbations in unimportant features. To demonstrate this



(a) Heart Disease (b) Iris

Figure 5: Sample SREDT for selected real-world datasets. Full-size images and more samples are shown in Appendix B.

unique property, we run 2 additional tests tabulated in Table 5. In the first test, for every existing feature column, we generate 3 additional feature columns, each generated by randomly permuting the existing feature column (RPC). In the second test, for every existing feature column, we generate 3 additional feature columns, each generated by adding random Gaussian noise (RGN) to the original values (noise added has 0 mean and a standard deviation equal to that of the existing feature column). Other than SREDT and DT, all other models (AOM) are 'tricked' (model's decision is influenced by the noisy column) 100% of the time. Even when compared to DT, SREDT performs better as shown in Table 5. With respect to *accuracy* and *F-score*, SREDT performance (averaged over the 6 datasets and both noise models) decreased by only 1.7% in *accuracy* and 0.72% in *F-Score*.

(iii) SREDT is also explainable compared to the SOTA benchmark methods in Table 1. We show samples of SREDT in Fig. 5 (full-size images and more samples are shown in Supplementary Appendix B) for the real-world datasets. Unlike tree ensembles and deep models, we can easily visualize and understand the decision making process of SREDT. This is because SREDT has low depth, few leaves and few terms (while having competitive performance), reducing the cognitive load for the user which in turn increases explainability (Zhang et al. 2022; Lage et al. 2019; Abdul et al. 2020). Specifically: (a) Zhang et al. (2022) reasoned that explainability can be measured by the depth of a tree, (b) Lage et al. (2019) demonstrated that the number of visual cognitive chunks in a model is one of the most important factors to explainability and (c) Abdul et al. (2020) linked explainability with cognitive load which is in turn measured by the number of visual cognitive loads.

Further Experiments with Local SREDT. To improve SREDT, we propose to add a local search mechanism to the population (detailed outline in Supplementary Appendix Algorithm 1). Gini impurity is still used as a fitness function, but before the fitness is evaluated, every equation in the population has its constants optimized with respect to a squared hinge loss function. The squared hinge loss is given as follows: $L(\hat{z}) = (\max(0, -y * \hat{z}))^2$, where $\hat{z}$ is the difference between the value of a datapoint and the threshold at the candidate split function, and y indicates the class of the datapoint (y takes on $+1$ if majority of the data from the same label are above the threshold, and -1 otherwise). Thus, datapoints that are classified on the wrong side of the threshold are penalized by the squared hinge loss. The squared hinge loss is differentiable (Luo, Qiao, and Zhang 2021), allowing us to do efficient local search which was otherwise not possible with traditional DT objective functions (e.g., Gini impurity). In our experiments, we use the BFGS algorithm (Broyden 1970) to optimize numerical constants for the squared hinge loss. We refer to the proposed SREDT based on local search as *local SREDT*.

Tables 1 and 2 demonstrate the competitiveness of local SREDT compared to DT, ODT, SREDT and other SOTA methods on the 6 benchmark datasets. Further, we ran *local SREDT* along with our original SREDT, ODT and DTs of 2 different depths on 56 different PMLB datasets (Olson et al. (2017), dataset details in Supplementary Appendix D). The results for *accuracy* and *F-Score* are tabulated in Supplementary Appendix E. This mechanism improves accuracy performance on 77.2% of the datasets, outperforming DT and ODT. In terms of F-Score, *local* SREDT outperforms DT and ODT in 82.5% and 73.7% of the datasets, respectively. In terms of inference time, *local* SREDT requires 25.8% and 26.6% less inference time than DT and ODT, respectively. In short, by using the *differentiable* squared hinge loss, SREDT is empowered to perform local search for numerical constants optimization.

Conclusion

In this paper, we introduce SREDT, an algorithm that synergistically combines symbolic regression and decision trees to discover short, compact trees with much richer class decision boundaries. While DT provides the structure of a 'tree of splits', SR provides 'good splits' by efficiently exploring a variety of conditions for each split of the tree. In terms of *accuracy* and *F-score*, we show improvements over DT and ODT and competitive results with SOTA methods, while still being explainable. We also show how SREDT is superior in terms of compactness, explainability, robustness to adversarial data noise and inference time. Finally, we introduce further improvement in the form of local search for numerical constants of every candidate split. Future work could explore ensembles of SREDT and the performance-explainability trade-off.

Acknowledgments

This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG3-PhD-2023-08-052T), and A*STAR, CISCO Systems (USA) Pte. Ltd and National University of Singapore under its Cisco-NUS Accelerated Digital Economy Corporate Laboratory (Award I21001E0002).

References

Abdul, A.; von der Weth, C.; Kankanhalli, M.; and Lim, B. Y. 2020. COGAM: measuring and moderating cognitive load in machine learning model explanations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–14.

Ali, M. 2020. PyCaret: An open source, low-code machine learning library in Python. *PyCaret version*, 2.

Arık, S. O.; and Pfister, T. 2021. Tabnet: Attentive interpretable tabular learning. In *AAAI*, volume 35, 6679–6687.

Augusto, D. A.; and Barbosa, H. J. 2000. Symbolic regression via genetic programming. In *Proceedings. Vol. 1. Sixth Brazilian Symposium on Neural Networks*, 173–178. IEEE.

Blanco-Justicia, A.; and Domingo-Ferrer, J. 2019. Machine learning explainability through comprehensible decision trees. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, 15–26. Springer.

Borisov, V.; Leemann, T.; Seßler, K.; Haug, J.; Pawelczyk, M.; and Kasneci, G. 2021. Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889*.

Breiman, L. 2001. Random forests. *Machine learning*, 45(1): 5–32.

Breiman, L.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 2017. *Classification and regression trees*. Routledge.

Brodley, C. E.; and Utgoff, P. E. 1995. Multivariate decision trees. *Machine learning*, 19(1): 45–77.

Broyden, C. G. 1970. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1): 76–90.

Chen, T.; and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Cortes, C.; and Vapnik, V. 1995. Support-vector networks. *Machine learning*, 20(3): 273–297.

Costa, V. G.; and Pedreira, C. E. 2023. Recent advances in decision trees: An updated survey. *Artificial Intelligence Review*, 56(5): 4765–4800.

Došilović, F. K.; Brčić, M.; and Hlupić, N. 2018. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, 0210–0215. IEEE.

Dua, D.; and Graff, C. 2019. UCI Machine Learning Repository [http://archive. ics. uci. edu/ml]. Irvine, CA: University of California. *School of Information and Computer Science*, 25: 27.

Friedman, J.; Hastie, T.; and Tibshirani, R. 2000. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2): 337–407.

Friedman, J. H. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Ganaie, M.; Tanveer, M.; and Suganthan, P. N. 2020. Oblique decision tree ensemble via twin bounded SVM. *Expert Systems with Applications*, 143: 113072.

Geurts, P.; Ernst, D.; and Wehenkel, L. 2006. Extremely randomized trees. *Machine learning*, 63(1): 3–42.

Heath, D.; Kasif, S.; and Salzberg, S. 1993. Induction of oblique decision trees. In *IJCAI*, volume 1993, 1002–1007. Citeseer.

Ingalalli, V.; Silva, S.; Castelli, M.; and Vanneschi, L. 2014. A multi-dimensional genetic programming approach for multi-class classification problems. In *European Conference on Genetic Programming*, 48–60. Springer.

Kaya, H.; Gurpinar, F.; and Ali Salah, A. 2017. Multi-modal score fusion and decision trees for explainable automatic job candidate screening from video cvs. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 1–9.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T.-Y. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

Kommenda, M.; Burlacu, B.; Kronberger, G.; and Affenzeller, M. 2020. Parameter identification for symbolic regression using nonlinear least squares. *Genetic Programming and Evolvable Machines*, 21(3): 471–501.

Korns, M. F. 2017. Evolutionary linear discriminant analysis for multiclass classification problems. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 233–234.

Korns, M. F. 2018. An Evolutionary Algorithm for Big Data Multiclass Classification Problems. In *Genetic Programming Theory and Practice XIV*, 165–178. Springer.

Korns, M. F.; and May, T. 2019. Strong typing, swarm enhancement, and deep learning feature selection in the pursuit of symbolic regression-classification. In *Genetic Programming Theory and Practice XVI*, 59–84. Springer.

Koza, J. R. 1992. Genetic programming. On the programming of computers by means of natural selection. *Complex adaptive systems*.

Lage, I.; Chen, E.; He, J.; Narayanan, M.; Kim, B.; Gershman, S.; and Doshi-Velez, F. 2019. An evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1902.00006*.

Lewis, R. J. 2000. An introduction to classification and regression tree (CART) analysis. In *Annual meeting of the society for academic emergency medicine in San Francisco, California*, volume 14. Citeseer.

Luo, J.; Qiao, H.; and Zhang, B. 2021. Learning with smooth Hinge losses. *Neurocomputing*, 463: 379–387.

Mahbooba, B.; Timilsina, M.; Sahal, R.; and Serrano, M. 2021. Explainable artificial intelligence (xai) to enhance trust management in intrusion detection systems using decision tree model. *Complexity*, 2021.

Martinez-Gil, J.; and Chaves-Gonzalez, J. M. 2020. A novel method based on symbolic regression for interpretable semantic similarity measurement. *Expert Systems with Applications*, 160: 113663.

McKay, B.; Willis, M. J.; and Barton, G. W. 1995. Using a tree structured genetic algorithm to perform symbolic regression. In *First international conference on genetic algorithms in engineering systems: innovations and applications*, 487–492. IET.

Montañana, R.; Gámez, J. A.; and Puerta, J. M. 2021. STree: A Single Multi-class Oblique Decision Tree Based on Support Vector Machines. In *Conference of the Spanish Association for Artificial Intelligence*, 54–64. Springer.

Mundhenk, T. N.; Landajuela, M.; Glatt, R.; Santiago, C. P.; Faissol, D. M.; and Petersen, B. K. 2021. Symbolic Regression via Deep Reinforcement Learning Enhanced Genetic Programming

Seeding. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y. N.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 24912–24923.

Muñoz, L.; Silva, S.; and Trujillo, L. 2015. M3gp–multiclass classification with gp. In *European Conference on Genetic Programming*, 78–91. Springer.

Murthy, S.; and Salzberg, S. 1995. Lookahead and pathology in decision tree induction. In *IJCAI*, 1025–1033. Citeseer.

Murthy, S. K.; Kasif, S.; and Salzberg, S. 1994. A system for induction of oblique decision trees. *Journal of artificial intelligence research*, 2: 1–32.

Olson, R. S.; La Cava, W.; Orzechowski, P.; Urbanowicz, R. J.; and Moore, J. H. 2017. PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData mining*, 10(1): 1–13.

Petersen, B. K.; Larma, M. L.; Mundhenk, T. N.; Santiago, C. P.; Kim, S. K.; and Kim, J. T. 2019. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *arXiv preprint arXiv:1912.04871*.

Popov, S.; Morozov, S.; and Babenko, A. 2019. Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*.

Prokhorenkova, L.; Gusev, G.; Vorobev, A.; Dorogush, A. V.; and Gulin, A. 2018. CatBoost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31.

Quade, M.; Isele, T.; and Abel, M. 2020. Machine learning control—Explainable and analyzable methods. *Physica D: Nonlinear Phenomena*, 412: 132582.

Raschka, S. 2018. MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. *The Journal of Open Source Software*, 3(24).

Schmidt, M.; and Lipson, H. 2009. Distilling free-form natural laws from experimental data. *science*, 324(5923): 81–85.

Shwartz-Ziv, R.; and Armon, A. 2022. Tabular data: Deep learning is not all you need. *Information Fusion*, 81: 84–90.

Smith, J. W.; Everhart, J. E.; Dickson, W.; Knowler, W. C.; and Johannes, R. S. 1988. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the annual symposium on computer application in medical care*, 261. American Medical Informatics Association.

Stephens, T. 2016. *Genetic Programming in Python, with a scikit-learn inspired API: gplearn*.

Sun, S.; Ouyang, R.; Zhang, B.; and Zhang, T.-Y. 2019. Data-driven discovery of formulas by symbolic regression. *MRS Bulletin*, 44(7): 559–564.

TuringBot. 2020. *Symbolic Regression Software*.

Udrescu, S.-M.; and Tegmark, M. 2020. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16): eaay2631.

Wang, Y.; Wagner, N.; and Rondinelli, J. M. 2019. Symbolic regression in materials science. *MRS Communications*, 9(3): 793–805.

Xu, F.; Uszkoreit, H.; Du, Y.; Fan, W.; Zhao, D.; and Zhu, J. 2019. Explainable AI: A brief survey on history, research areas, approaches and challenges. In *CCF international conference on natural language processing and Chinese computing*, 563–574. Springer.

Zhang, M.; et al. 2022. Which neural network makes more explainable decisions? An approach towards measuring explainability. *Automated Software Engineering*, 29(2): 1–26.