PHILOSOPHICAL TRANSACTIONS A

royalsocietypublishing.org/journal/rsta

Research

Cite this article: Imai Aldeia GS, Romano JD, Olivetti de França F, Herman DS, La Cava WG. 2026 Towards symbolic regression for interpretable clinical decision scores. *Phil. Trans. R. Soc. A* **384**: 20240588.
https://doi.org/10.1098/rsta.2024.0588

Received: 1 June 2025
Accepted: 29 September 2025

One contribution of 15 to a discussion meeting issue 'Symbolic regression in the physical sciences'.

Subject Areas:
artificial intelligence, medical computing, biomedical engineering

Keywords:
symbolic regression, health informatics, clinical decision-making

Author for correspondence:
William G. La Cava
e-mail: william.lacava@childrens.harvard.edu

Towards symbolic regression for interpretable clinical decision scores

Guilherme Seidyo Imai Aldeia[1,2,3], Joseph D. Romano[4], Fabricio Olivetti de França[1], Daniel S. Herman[5] and William G. La Cava[2,3]

[1]Federal University of the ABC, Santo André, São Paulo, Brazil
[2]Computational Health Informatics Program, Boston Children's Hospital, Boston, MA, USA
[3]Harvard Medical School, Boston, MA, USA
[4]Institute for Biomedical Informatics, and [5]Department of Pathology and Laboratory Medicine, University of Pennsylvania, Philadelphia, PA, USA

GSIA, 0000-0002-0102-4958; FOdF, 0000-0002-2741-8736; WGLC, 0000-0002-1332-2960

Medical decision-making makes frequent use of algorithms that combine risk equations with rules, providing clear and standardized treatment pathways. Symbolic regression (SR) traditionally limits its search space to continuous function forms and their parameters, making it difficult to model this decision-making. However, owing to its ability to derive data-driven, interpretable models, SR holds promise for developing data-driven clinical risk scores. To that end, we introduce Brush, an SR algorithm that combines decision-tree-like splitting algorithms with nonlinear constant optimization, allowing for seamless integration of rule-based logic into SR and classification models. Brush achieves Pareto-optimal performance on SRBench and was applied to recapitulate two widely used clinical scoring systems, achieving high accuracy and interpretable models. Compared with decision trees (DTs), random forests (RFs) and other SR methods, Brush achieves comparable or superior predictive performance while producing simpler models.

This article is part of the discussion meeting issue 'Symbolic regression in the physical sciences'.

THE ROYAL SOCIETY PUBLISHING

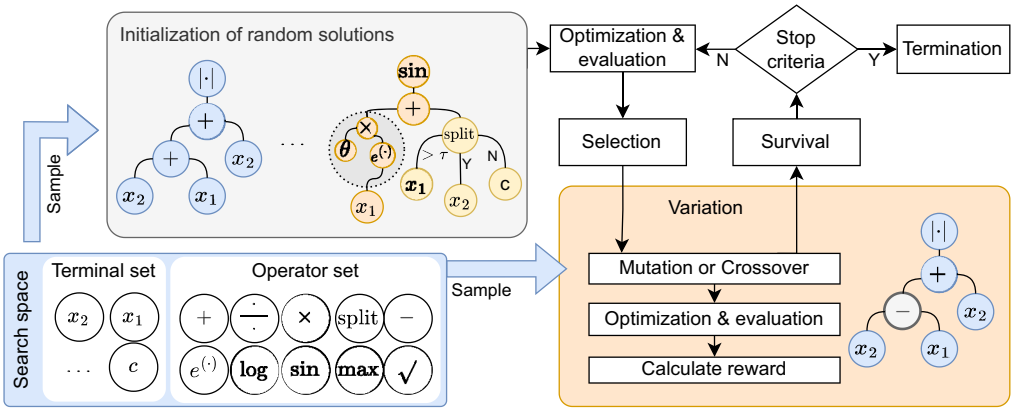

Figure 1. Brush overview. Nodes are sampled from the search space to build mathematical expression trees, including those with split operations. A randomly generated population of expressions goes into an evolutionary loop. The set of nodes is used to create variations on the population. After several generations, the final solution is selected from the evolved individuals.

1. Introduction

Since symbolic regression (SR) was first proposed [1], it has been applied in various fields, including physics [2–6], medical informatics [7–9], aerospace engineering [10,11] and material science [12]. The main appeal of SR is its potential to produce intrinsically interpretable solutions, often crucial for scientific discovery in basic sciences or when prediction models must justify their recommendations, e.g. in healthcare contexts, cf. the United States [13] guidelines for clinical decision support software.

While promising for healthcare, purely mathematical expressions can be a drawback in practice owing to the mathematical literacy they require, making them less attractive than decision trees (DTs) [14], which are often preferred when interpretability is needed [15]. In the healthcare context, widely used triage scoring systems in the US, such as cardiac arrest risk triage (CART) [16], are based on physiological parameters and can be promptly evaluated to predict urgent deterioration of patients [17]. Scoring systems require interpretability, consisting of simple decision rules, and have been associated with better hospital outcomes [18]. They are common clinical decision tools because they provide objective, quantifiable measures for decision-making and have historically been developed manually [19]. These systems could be explored with the aid of big data and artificial intelligence [20], especially through the usage of electronic health records (EHRs).

Contemporary SR algorithms that achieve good results in modelling regression equations for real-world and physics data [21] rely on local parameter optimization [22], but a limitation is not being able to incorporate split-wise operations, owing to the difficulty of optimizing such parameters with existing optimization methods. This task remains underexplored in SR.

We propose a genetic programming (GP) SR algorithm named *Brush*, aiming to bridge the gap between symbolic prediction models and split-wise operations, along with nonlinear parameter optimization. Brush addresses the multi-objective optimization problem of simultaneously maximizing performance and minimizing model complexity. It learns split operations at any point within the expression while maintaining compatibility with nonlinear optimization, even in the presence of discontinuities. Figure 1 depicts our proposed algorithm.

Our experiments are twofold. In the first set of experiments, we validate our method and place it in the context of other state-of-the-art approaches. We ran Brush on 122 real-world and 130 synthetic physics problems from SRBench [21], reporting R^2 and the solution accuracy of the governing physics equations. Brush achieved Pareto-optimal performance in SRBench and, compared

with other SR methods, produced significantly smaller expressions. It achieved $R^2 > 0.999$ in more than half of the runs, even under high amounts of noise.

In the second set of experiments, we applied Brush to regression and classification tasks using EHR data from the Beth Israel Deaconess Medical Center. We extracted data for 10 000 individuals from the Medical Information Mart for Intensive Care (MIMIC)-IV database [23] emergency department (ED) table, including vital signs and demographics collected at triage, and applied Brush to the regression task of learning triage scores—namely, cardiac arrest risk triage (CART) and modified early warning score (MEWS). Brush produced competitive results with much smaller expressions. We then modified the regression task to a classification scenario, aiming to identify patients at high risk of catastrophic deterioration based on vital signs. Brush successfully generated small models with simple split rules to flag high-risk patients.

Our results show that Brush is competitive with state-of-the-art SR algorithms while incorporating split-wise equations and can also support decision-making by generating models that reflect the features and logic of ground-truth systems. The algorithm is promising both as a first-principles modelling tool and as a method for learning interpretable classification models, expanding the practical applications of SR.

2. Related work

The idea of mixing DTs with linear regression models as leaves has been previously explored [24,25], but recent methods have integrated SR with DTs [26–28]. PS-Tree [26] builds DT-like structures with SR models as leaves and achieves competitive R^2 performance, but at the cost of overly large models. It lacks parameter optimization, relying on randomly generated constants. Another method, similar to PS-Tree, generates DTs with symbolic models as leaves [27], performing well on problems with or without required splits. A different approach, symbolic regression enhanced DTs (SREDT) [28], uses splits derived from an SR algorithm. SREDT outperforms traditional DTs but also lacks parameter optimization.

Optimizing free parameters remains challenging, as random constants hinder convergence and fail to reach global optima [29], with evolutionary methods alone converging slowly [30]. This burden can be reduced using optimization methods. Recent SR algorithms apply the Levenberg–Marquardt method [31,32] to tune model parameters [33–36], which has proven effective. Some pre-trained transformer-based methods also struggle with constants, highlighting the need for nonlinear optimization to improve model accuracy [37–39].

Previous work by La Cava *et al.* [7] modified the feature engineering automation tool (FEAT) [40] SR algorithm to perform logical comparisons, and it was applied to learn linear combination of meta-features in a logit function for developing computable phenotypes for hypertension. Other SR applications in healthcare include using SR for feature engineering inputs to classifiers for heart failure prediction [8] or learning meta-features from paediatric patients to predict scores from CT scans [9].

3. Brush: mixing split-wise functions and parameter optimization

Brush is an SR algorithm capable of learning expressions with split-wise operations, without being constrained to a DT-like structure. Its splits are compatible with nonlinear parameter optimization methods, which are lacking in current algorithms. In addition, Brush integrates several modern strategies within the GP framework to efficiently explore the search space. This section introduces the split node and its optimization approach, followed by a description of the mechanisms integrated into Brush.

Let a dataset be a set of d observed points $\{(\mathbf{x}_i, y_i)\}_{i=1}^{d}$, where $\mathbf{x}$ is a n-dimensional feature vector, and y is the target value. We denote the feature matrix as $\mathcal{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d]' \in \mathbb{R}^{n \times d}$, and the vector

of target values as $\mathbf{y} = [y_1, y_2, \ldots, y_d] \in \mathbb{R}^d$. Brush searches for $\hat{f}(\mathcal{X}, \hat{\theta}) : \mathbb{R}^n \to \mathbb{R}$ such that $\hat{f}(\mathcal{X}) \approx \mathbf{y}$, where $\hat{f}$ is a mathematical expression represented as a tree.

Brush introduces a specialized *split node operator*, inspired by classical DTs [14]—a split is a predicate determining whether to evaluate the left or right branch based on a boolean output. During learning, the algorithm exhaustively tests combinations of feature x_i and threshold τ, selecting the configuration $[x_{i^*} > \tau^*]$ that minimizes the total variance of the target values in the resulting partitions, $\mathbf{y}_{[x_{i^*} > \tau^*]}$ and $\mathbf{y}_{[x_{i^*} \leq \tau^*]}$.

Our split node has three branches: one real-valued subtree f_c as the condition input, and one subtree each for the true and false cases f_T and f_F, respectively. Each branch is another Brush tree. The conditional compares f_c with a learnable weight τ to determine the data flow between f_T and f_F. The optimal τ^* minimizes the sum of target variances on either side of the split, effectively performing one-dimensional clustering:

$$\tau^* = \min_{\tau} \left(\frac{\mathrm{Var}(\mathbf{l})}{|\mathbf{l}|} + \frac{\mathrm{Var}(\mathbf{r})}{|\mathbf{r}|} \right) \tag{3.1}$$

$$\text{such that } \min\left(f_c(\mathcal{X}, \theta)\right) < \tau < \max\left(f_c(\mathcal{X}, \theta)\right), \tag{3.2}$$

$$\text{where } \mathbf{l} = \left[y_i : \mathbf{y} \,|\, f_c(x_i, \theta) \leq \tau\right] \tag{3.3}$$

and

$$\mathbf{r} = \left[y_i : \mathbf{y} \,|\, f_c(x_i, \theta) > \tau\right]. \tag{3.4}$$

This operator masks the dataset, so downstream operations are applied only to a subset of the training data. Restricting subtrees to a subset of the data allows the resulting subtree to be more precise and enables faster parameter optimization.

In Brush, the initial population can optionally consist solely of split nodes, which are then progressively replaced by mathematical expressions through GP whenever predictive performance is improved. Each node is assigned an innate weight, which can be toggled during the search. In the initial population, only terminal weights are toggled on. This extends the Operon [36] mechanism that assigns weights exclusively to terminals. Let the node weights and any constants in an expression be adjustable parameters of the function (θ). Define the residual error as the result of the function $\hat{f}$ with parameters $\theta \in \mathbb{R}^p$ as $\mathrm{H}(\theta) = \hat{f}(\mathcal{X}, \theta) - \mathbf{y}$. Then, the optimization problem is performed by an iterative process of gradient descent to minimize the mean squared error using the Levenberg–Marquardt method [31,32]:

$$\theta^* = \arg\min_{\theta} \frac{1}{2} ||H(\theta)||^2. \tag{3.5}$$

Since any subtree is a valid Brush tree, we can isolate optimization to subtrees. We propose the following heuristic for optimization with split nodes. First, we optimize the conditional subtree f_c, then find the best threshold τ^* that clusters the data into two groups with minimal variance. Finally, we fit the remaining expression, ignoring the already optimized parameters and passing only the data subset matching the condition to each corresponding branch. We propose a *greedy split node*, where the conditional is a single feature and the threshold is learnt as in DTs, and a *flexible split node*, where the evolutionary framework generates a sub-expression and only its threshold is optimized.

Figure 2*A* illustrates tree evaluation, figure 2*B* shows how a split node partitions data across subtrees and figure 2*C* depicts the overall evaluation and optimization process. The split nodes minimize the sum of target variances between the two subtrees. The optimization is done in three steps: first, we optimize the decision criterion equation (3.5)—which may be a single feature with

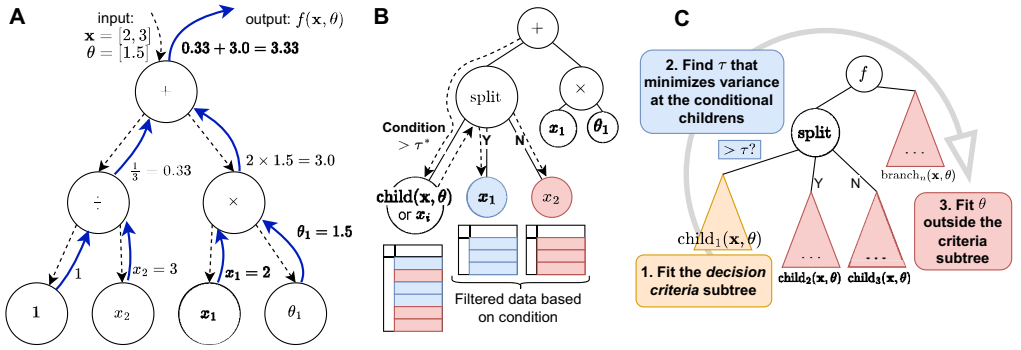

Figure 2. Brush evaluation, split and optimization. (A) Evaluation starts at the root, recursively propagating input **x** and parameters θ, applying each node's symbol to its children. (B) Splits can occur anywhere, with the conditional subtree directing data flow based on the condition. (C) Optimization with splits in three steps: optimize child$_1$; find the τ minimizing y variance across child$_2$ and child$_3$; then fit the rest of the tree, ignoring already fitted parameters.

a threshold or a subtree—then determine the optimal threshold for the split equation (3.1) and finally fit the remaining parameters equation (3.5) while keeping the split condition and threshold fixed.

Brush uses GP to optimize the overall fitness of a population of candidate models. Fitness is defined as a vector of objectives we want to optimize. Specifically, we use the mean squared error $\mathrm{MSE}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{d}\sum_{i=1}^{d}(\hat{y}_i - y_i)^2$ and the linear complexity to concurrently prioritize simplicity. For a node n with k arguments, the linear complexity is the sum of its children's linear complexities with the complexity of the node itself c_n, that is $C(n) = c_n + (\sum_{a=1}^{k} C(a))$, discouraging the use of operators with high complexity values.

The GP loop starts with a randomly initialized population, then iteratively performs selection, variation and survival steps. *Selection* chooses candidates based on fitness to undergo *variation*, generating offspring that inherit characteristics from parents. Variation doubles the population size, which returns to its original size during *survival*, where selection pressure favours better solutions, with probabilities proportional to their fitness. Brush uses ϵ-lexicase [41] for selection and non-dominated sorting [42,43] (NSGA-II) for survival, which are algorithms proven to yield better convergence performance than other selection and survival mechanisms.

Finally, we apply inexact simplification [44] as a post-processing step. This technique identifies and replaces large subtrees with simpler, approximately equivalent alternatives by comparing prediction vectors. This allows us to compress models while preserving their performance.

We implemented six different mutation operators and a subtree crossover. *Toggle weight on/off* randomly enables or disables a learnable weight on a node. *Subtree* replaces one node with a random subtree, generated with probabilistic tree creation (PTC2). *Point* replaces one random node with a new one of the same arity. *Delete* removes one node, keeping its children. *Insert* creates a new node. The *crossover* randomly switches two subtrees between the selected parents.

Finally, we extend Brush to classification tasks by introducing a dedicated classifier mode. In this setting, a logistic regression (LR) node is fixed at the root of the expression tree. We also enforce the presence of an offset parameter, which can be optimized using the same nonlinear optimization methods, without requiring any modifications to the algorithm, broadening the applicability of Brush to practical problems.

4. Performance on the symbolic regression benchmark

In our first set of experiments, we demonstrate that Brush can achieve competitive performance on SR benchmarks. The SR community has adopted SRBench [21] as the standard benchmark for SR.

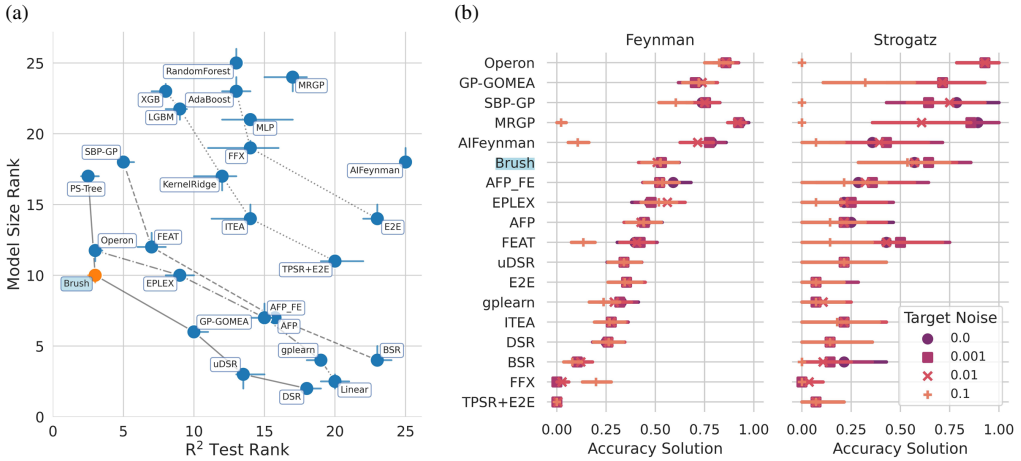

Figure 3. Brush's SRBench results. Bars denote the bootstrapped confidence intervals (CI). (a) Pareto plot comparing model size rank (y-axis) and R^2 score rank (x-axis) for black-box problems. Smaller ranks are better. Points denote the median, and bars denote the 95% CI. (b) Mean accuracy solution rate for ground-truth problems with varying noise levels, where the rate is defined as the percentage of results with $R^2 > 0.999$ (four significant figures). Colour/shapes indicate noise levels (normal distribution) added to y, and bars denote the 95% CI.

It is composed of two tracks: *black-box* and *ground-truth* problems. The former are 122 datasets derived from the Penn machine learning benchmarks [45], while the latter are 130 datasets generated with physics equations from the Feynman lectures [46,47] compiled by Udrescu & Tegmark [4], along with 14 nonlinear dynamical systems from Strogatz [48]. The objective in the ground-truth track is to recover the original equation from noisy observations.

SRBench performs 10 trials per dataset using a 75/25 train-test split. For the black-box track, the training is limited to 500 000 function evaluations or 48 h. For the ground-truth track, the limits are 1 000 000 evaluations or 8 h, with varying levels of noise $[0, 0.1, 0.01, 0.001]$ added to the target $\mathbf{y}$. In our experiments, Brush used fixed hyperparameters: population size of 1000, 100 generations, maximum tree depth of 10, maximum expression size of 128, 10 iterations of local optimization and function set: $+, -, \times, \div, \sin, \cos, \tanh, \exp, \log, \sqrt{\cdot}$, pow, Split.

Results for other algorithms were taken from the original SRBench benchmark [49]. In addition, we ran recent relevant SR methods, such as E2E [37], uDSR [38] and TPSR+E2E [39], using their default configurations. Although the inclusion of these methods provides only a rough estimate of their performance, we did it out of completeness, given their reported performances, as their raw results are not available, potentially limiting more fair comparisons.

Brush's performance on SRBench is summarized in figure 3. The Pareto plot for the black-box problems shows that Brush produces significantly smaller models than PS-Tree, and often smaller than Operon. It dominates FEAT in both model simplicity and accuracy, achieving the second-best rank in test R^2 score.

For the ground-truth track, Brush recovers accurate expressions across varying noise levels. Although split nodes and weighted terminals may prevent exact equation recovery, we observe that over 50% of solutions achieve $R^2 > 0.999$ with four significant figures for all levels of target noise, and Brush achieves a mean (s.d.) of 84.11(±25.77)% solution accuracy with no noise and 83.63(±25.55)% at maximum noise with three significant figures (i.e. $R^2 > 0.99$). Brush is robust to noise, unlike methods, such as AIFeynman and multiple regression genetic programming (MRGP), which show a significant drop in accuracy as noise increases. This suggests Brush successfully recovers approximately half of all Feynman and Strogatz equations, regardless of the noise level. PS-Tree does not appear in the ground-truth track, as its enforced use of split-wise

Table 1. CART scoring system [16].

score	0	4	6	8	9	12	13	15	22
respiratory rate	< 21	—	—	[21–24)	—	[24–26)	—	[26–29)	≥ 29
heart rate	< 110	[110–140)	—	—	—	—	≥ 140	—	—
diastolic BP (mmHg)	≥ 49	[40–50)	[35–40)	—	—	—	≤ 35	—	—
age	< 55	[55–70)	—	—	≥ 70	—	—	—	—

Table 2. Simplified MEWS scoring system [17] .

	3	2	1	0	1	2	3
systolic BP (mmHg)	< 71	[71–81)	[81–101)	[101–200)	—	≥ 200	—
heart rate	—	< 41	[41–51)	[51–101)	[101–111)	[111–130)	≥ 130
respiratory rate	—	< 9	—	[9–15)	[15–21)	[21–30)	≥ 30
temperature (°C)	—	< 35	—	[35–38.5)	—	≥ 38.5	—

operations prevents it from modelling exact equations, and their original authors opted out for this track.

These results show that Brush achieves state-of-the-art performance across a diverse set of over 250 datasets. Brush matches the accuracy of traditional nonlinear optimization methods, while allowing for flexible expression structures with decision splits at any node. Overall, Brush demonstrates good potential in regression and physics equation modelling tasks, with robust performance in noisy settings.

5. Methods

To demonstrate the split nodes utility, we designed a high-risk classification experiment based on clinical risk scores derived from the MIMIC-IV-ED v. 2.2 dataset. MIMIC is a publicly available dataset containing electronic medical records from 2008 to 2019, collected at the Beth Israel Deaconess Medical Centre [23]. These data are de-identified and made available to researchers under a data use agreement following completion of human subjects training. Using the MIMIC-IV-ED pipeline [20], we extracted one simple clinical calculation (mean arterial pressure (MAP)), the CART score and a simplified version of the MEWS. While these scores were originally designed for ordinal assessment, we transformed them into binary classification tasks by using thresholds [18] for high risk of catastrophic deterioration classification.

For MAP, we used the standard formula $\text{MAP} = \frac{1}{3}\text{SBP} + \frac{2}{3}\text{DBP}$, where SBP and DBP are the systolic and diastolic blood pressure, respectively.

The CART score (table 1) aggregates points from respiratory rate, heart rate, DBP and age, with a total score ranging from 0 to 57. We considered the threshold of greater or equal to 12 for high risk.

The simplified MEWS score (table 2) combines SBP, heart rate, respiratory rate, temperature and a responsiveness score (AVPU). We excluded the AVPU component because it was not accessible in a structured form. We used an interpretive threshold of greater or equal to 3. The maximum attainable score in our setting is 11.

For each score, we trained models to solve both the original regression task and the binary classification task based on the clinical thresholds. We compared Brush with interpretable machine learning (ML) baselines and SR methods, including split-wise operations: PS-Tree, FEAT, DTs and LR with L2 regularization (LR L2). We focused comparisons on PS-Tree owing to its structural similarity to Brush (both rely on split-wise expressions) and on FEAT owing to its successful application in clinical decision-making settings.

Hyperparameter settings were as follows. DT had no limit on maximum depth. LR used L2 regularization and the `liblinear` solver. Random forests (RFs) were fine-tuned for each run to optimize the number of estimators by searching the parameter grid: `n_estimators` $\in \{10, 100, 1000\}$, `min_weight_fraction_leaf` $\in \{0.0, 0.25, 0.5\}$ and `max_features` $\in$ 'sqrt' , 'log2' , none. FEAT was tuned with population sizes $\in \{100, 500, 1000\}$, generation steps $\in \{250, 500, 2500\}$ and learning rates $\in \{0.1, 0.3\}$. Brush used fixed hyperparameters set after preliminary experiments: `pop_size` = 500, `max_gens` = 100, `max_depth` = 12, `max_size` =100 and function set {`Add, Sub, Mul, Div, Ceil, Floor, Pow, Log, Min, Max, Split`}.

We conducted five runs of fivefold cross-validation, totalling 25 runs per method. Each dataset included 10 000 samples split in a stratified 75/25 train-test split. Class imbalance was present across tasks, with a positive case prevalence of 0.09 for CART and 0.11 for MEWS. Owing to this imbalance, we report the area under the precision–recall curve (AUPRC) using the average precision score calculation, a more suitable measure for imbalanced class problems. The feature set comprised 77 variables including: triage vital signs (temperature, heart rate, blood pressure), ED visits in the past 30 or 365 days, chief complaints and Charlson comorbidity index. We excluded post-admission ED vitals, which could act as proxies for triage admission. Even though the triage scores rely on at most five features, we kept the entire set to challenge the algorithm's feature selection when building interpretable models.

6. Results

We evaluated the performance of all methods on both the scoring and classification problems using the triage scores. In the scoring version, the objective was to predict the clinical score values (MAP, CART, simplified MEWS), formulated as a regression problem. Table 3 lists the average R^2 values and model size for each method, along with s.d. across the runs. Model size is defined as the number of nodes required to represent the model in a Brush-style expression tree.

Next, we report the classification versions of the CART and simplified MEWS tasks, where models predict whether a patient is at high risk of deterioration based on different triage scores. Table 4 lists the average AUPRC and model size. Owing to its lack of native support for classification, PS-Tree is excluded from this comparison.

We executed the experiments using Brush without split nodes to assess its effect on model performance. Disabling split nodes did not show a statistically significant effect on either size or R^2 scores for the regression problems but showed significant degradation to AUPRC for the deterioration classification tasks. On the test set, Brush without split nodes achieves a smaller AUPRC compared with its counterpart with split nodes in both CART deterioration (0.79 ± 0.09 versus 0.99 ± 0.05 ($p \leq 1.00 \times 10^{-4}$)) and MEWS deterioration tasks (0.88 ± 0.04 versus 0.91 ± 0.04 ($1.00 \times 10^{-3} < p \leq 1.00 \times 10^{-2}$)). Using the split nodes did not hurt the performance for other problems. This implies the gains are not only in interpretability but also in classification performance.

To further explore the interpretability of Brush models for clinical decision-making, we selected exemplar models from the classification tasks. We note that this model interpretation was conducted using the preliminary Brush settings that did not include split node seeding of the initial population. First, we filtered out models with an AUPRC below 0.90. In addition, we observed that some models with high AUPRC had a test balanced accuracy of 0.5 on the test partition, indicating random performance. Consequently, we excluded models with a balanced accuracy below 0.9. From the remaining models, we selected the smallest, which is visualized in figure 4.

Table 3. Average R^2 and model size for each scoring task (±s.d.).

	MAP		CART score		simplified MEWS Score	
	R^2	size	R^2	size	R^2	size
RF	0.84 ± 0.28	$(3.50 \pm 4.11) \times 10^6$	0.99 ± 0.00	$(1.58 \pm 0.55) \times 10^6$	0.98 ± 0.01	$(1.19 \pm 0.77) \times 10^5$
DT	0.81 ± 0.26	11662.24 ± 89.93	0.99 ± 0.00	194.68 ± 6.93	0.98 ± 0.01	206.44 ± 8.40
LR L2	1.00 ± 0.00	303.00 ± 0.00	0.56 ± 0.11	303.00 ± 0.00	0.43 ± 0.08	303.0 ± 0.00
PS-Tree	1.00 ± 0.00	217.64 ± 80.13	0.87 ± 0.18	813.12 ± 141.54	0.85 ± 0.12	815.36 ± 169.99
FEAT	1.00 ± 0.00	13.48 ± 4.99	0.71 ± 0.23	47.76 ± 21.42	0.65 ± 0.10	31.28 ± 20.13
Brush	1.00 ± 0.00	13.16 ± 4.34	0.82 ± 0.26	76.44 ± 15.17	0.74 ± 0.04	45.00 ± 20.42

Table 4. Average AUPRC and size (±s.d.) for each of the clinical decision problems.

	CART deterioration		simplified MEWS deterioration	
	AUPRC	size	AUPRC	size
RF	1.00 ± 0.01	$(8.39 \pm 15.01) \times 10^3$	0.99 ± 0.00	$(2.22 \pm 2.56) \times 10^4$
DT	0.99 ± 0.01	38.48 ± 3.11	0.99 ± 0.01	63.76 ± 2.86
LR L2	0.75 ± 0.02	303.00 ± 0.00	0.81 ± 0.03	303.00 ± 0.00
FEAT	0.69 ± 0.17	60.76 ± 77.74	0.89 ± 0.03	55.60 ± 27.82
Brush	0.99 ± 0.01	73.80 ± 15.18	0.95 ± 0.02	61.84 ± 11.63

To complement the tabular results in table 4, figure 5 presents bar plots of both performance and model size for the classification task. Figure 5a displays the R^2 scores and complexities for the regression-based scoring tasks, and figure 5b shows the AUPRC scores and complexities for the classification tasks. These plots also include statistical comparisons between Brush and the other methods using the Mann–Whitney–Wilcoxon two-sided test with Holm–Bonferroni correction. We set the y-axis lower bound to 0.5 in AUPRC plots and 0.25 in R^2 plots.

7. Discussion

We observe that the MAP score—a weighted sum of two features—proved to be a particular challenge for traditional ML methods. Tree-based models such as RF and DT performed poorly compared with other methods, often producing large models with greater variability. This is probably due to the continuous nature of the output, making it difficult for split-based methods to perform well owing to their inherent discretization mechanism with constants as leaves. LR L2 also found large and less interpretable models, failing on the feature selection aspect. PS-Tree consistently produced overly large models, possibly due to the high dimensionality of the feature space combined with its partitioning strategy, which creates numerous splits. Alternatively, the PS-Tree result may be due to its reliance on random guessing of the coefficients. In contrast, FEAT and Brush produced more compact models with similar or better performance.

CART deterioration

```
Logistic(
  if(resprate>=21):
  | 1.00
  else:
  | if(dbp>=50):
  | | if(heartrate>=140):
  | | | 0.39
  | | else:
  | | | if(heartrate>=110):
  | | | | if(age>=70):
  | | | | | 0.39
  | | | | else:
  | | | | | -5233.87
  | | | else:
  | | | | 1.00 - dbp
  | | else:
  | | | if(age>=69):
  | | | | if(age>=71):
  | | | | | 0.39
  | | | | else:
  | | | | | age - heartrate
  | | else:
  | | | | -189.94 / o2sat
)
```

Simplified MEWS deterioration

```
logistic(-42.07 +
  max(
  | if(heartrate>=111):
  | | 54.05
  | else:
  | | if(sbp>=200):
  | | | 54.00
  | | else:
  | | | if(temperature>=38.50):
  | | | | 55.00
  | | | else:
  | | | | resprate
  | ),
  | (1 + 1.16*heartrate - 1.05*sbp),
  | n_hospitalizations_365d
  ) + \
  log(resprate)
)
```

Figure 4. Smallest Brush models edited as Python code for catastrophic deterioration prediction using two different scoring systems CART and simplified MEWS, taken from 25 runs.

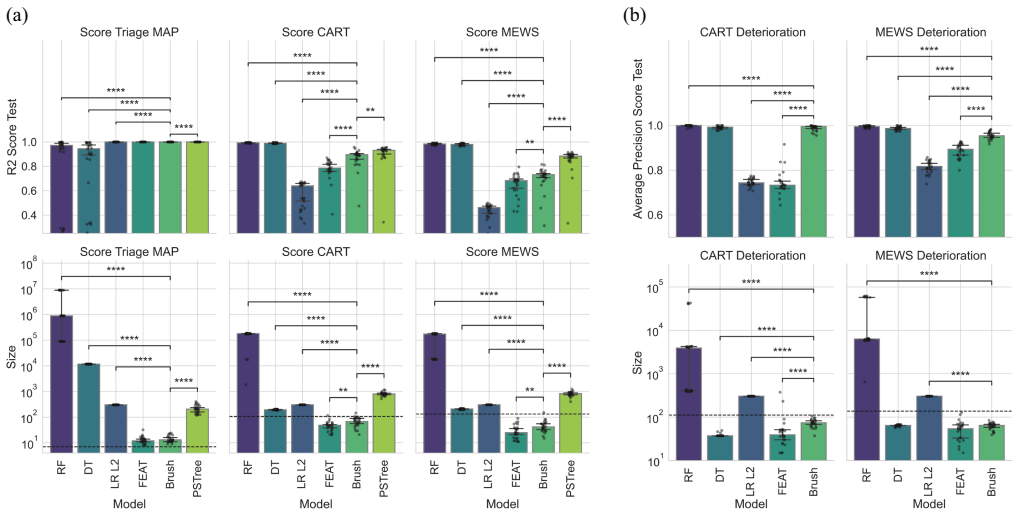

Figure 5. Metrics and sizes for the clinical decision experiment. The size of the original decision model is denoted as a dashed line. Statistical comparisons are conducted using the Mann–Whitney–Wilcoxon two-sided test with Holm–Bonferroni corrections. Asterisks indicate *: $1.00 \times 10^{-2} < p \leq 5.00 \times 10^{-2}$; **: $1.00 \times 10^{-3} < p \leq 1.00 \times 10^{-2}$; ***: $1.00 \times 10^{-4} < p \leq 1.00 \times 10^{-3}$ and ****: $p \leq 1.00 \times 10^{-4}$. Non-significant comparisons were omitted.

For the CART and simplified MEWS scoring tasks, we notice that LR L2 shows a drop in R^2, while FEAT and Brush achieved superior performance. This suggests that LR, when combined with the feature selection and feature engineering capabilities provided by GP (as used in both SR methods), can outperform classical LR. For these specific tasks, Brush consistently outperforms FEAT, at the cost of slightly larger models. PS-Tree achieves higher performance than Brush, but at the cost of models that are more than 20 times larger.

Figure 5a provides statistical comparisons, showing that the Brush models showed statistically significant differences except when compared with FEAT in the MAP score. It also reveals

that the trade-off between AUPRC and model size is overall unfavourable for PS-Tree owing to its extremely large model sizes. Brush models showed relatively low variance across runs when compared with other SR methods, indicating stable behaviour.

When model predictions are transformed into clinical decision tasks, we observe notable performance differences in table 4. In particular, both SR methods perform worse than DT on rule-based classification tasks. Compared with FEAT, Brush demonstrates superior performance on CART but yields comparable results on simplified MEWS. These differences are also present in regression tasks; however, Brush and FEAT generate more compact models than shallow methods such as RF, DT and LR L2. Brush outperforms LR L2 in all cases and produces smaller models than DT for simplified MEWS. Statistical comparisons in figure 5b show no significant size difference between Brush and DT for CART, but a significant reduction for MEWS.

Overall, we position Brush as a strong compromise between the three main modelling approaches. It approximates DT performance in classification tasks while yielding better regression models; it enhances logistic modelling compared with LR L2 through feature engineering and selection using GP and it outperforms FEAT and PS-Tree in different aspects by combining split-wise operations with parameter optimization. Allowing specific customization in the trade-off between performance and complexity could yield better task-specific performance. The Brush model uses a logistic root, meaning it outputs a probability estimate for deterioration risk. This probabilistic output can be evaluated using metrics like the AUPRC or receiver operator characteristic curve, and an optimal classification threshold can be chosen accordingly.

For the CART deterioration task, the model depicted in figure 4 has a size of 62 nodes, an AUPRC of 0.92 and a balanced accuracy of 0.99 on the test set. The model evaluation begins with a split on respiratory rate (greater than 21) and can be seen as the most immediate high-risk indicator, identifying high risk immediately—consistent with the original scoring system. If this condition fails, the model proceeds through a nested evaluation of DBP, heart rate and age, computing a final probability of deterioration. We can see combinations of DBP and heart rate in the first block of conditionals, which are the two factors other than respiration rate that have the highest scores in the CART scoring system of DBP less than or equal to 35 and heart rate greater than or equal to 140, resulting in a risk higher than the threshold, thus a positive label. Age has a smaller priority for probability modelling, and, in case the patient is not at advanced ages above 70, then a computation is made using 'age–heart rate'. We note that the final branch—when all conditionals fail, meaning a superior bound was set to all tested features—references O_2 saturation, which was not used during label generation and may be an artefact, and could be simplified. This structure mirrors the kind of heuristic reasoning a clinician might follow and resembles the CART scoring system, although in a compact, rule-based form.

The Brush model for simplified MEWS deterioration prediction had an AUPRC of 0.96 and a balanced accuracy of 0.91 on the test set, with a model size of 44 nodes. Impressively, it identified the relevant features aligned with the actual simplified MEWS scoring system from a pool of 77 possible features and learnt threshold values consistent with the actual clinical model. The model uses a `max` function to return a fixed output associated with high-risk prediction if any of the vital signs (heart rate, SBP or temperature) are critically high. If none of these is critical, the model considers a linear combination of heart rate and SBP, which identifies a combination of relative tachycardia and hypotension, and the number of previous admissions. Finally, this is added to the logarithm of respiratory rate. This algorithm integrates risk across vital signs in several ways, such that it is simpler than the MEWS score, which treats each element as independent.

8. Conclusions

In this article, we propose Brush, a multi-objective SR algorithm specially designed for solving problems where split-wise functions are desired. The novelty introduced by Brush is integrating split-wise functions with nonlinear optimization methods, also combining several state-of-the-art components into a GP framework, namely, ϵ-lexicase selection, NSGA-II survival, weighted

nodes, fast simplification methods and classification support. Brush contributes to building interpretable and high-performing models.

We have shown that Brush achieves Pareto-optimal performance on a collection of 122 real-world datasets and successfully retrieves more than half of the Feynman and Strogatz problems, even in noisy scenarios. These results highlight its potential as a tool for deriving interpretable and high-performing heuristics in healthcare and other domains where interpretability is essential.

Applied to real-world EHR data, Brush consistently identified relevant features and produced compact decision models with near-perfect predictive performance. Overall, the models are simple, interpretable and capture the essential structure of clinical scoring with minimal deviation, missing only some corner cases. Brush effectively balances interpretability, performance and stability—rarely achieved simultaneously in clinical modelling tools.

We notice some limitations. First, Brush was not fine-tuned, a step which could lead to better results, but which was out of the scope of this article. Second, Brush minimizes linear complexity based on arbitrarily defined complexities for each node, which could be adjusted or derived from prior distributions observed in different fields.

Data accessibility. SRBench [21] experiments and results are available at GitHub [49]. MIMIC-IV-ED dataset [50] is available at Physionet https://physionet.org/content/mimic-iv-ed/2.2/. MIMIC-IV-ED pipeline [51] is available at GitHub [52]. Brush source code [53] is hosted at GitHub https://github.com/cavalab/brush/. Our experiments are available at GitHub https://github.com/ cavalab/brush_paper_experiments.

Declaration of AI use. We have not used AI-assisted technologies in creating this article.

Authors' contributions. G.S.I.A.: data curation, investigation, methodology, software, validation, visualization, writing—original draft; J.D.R.: software, writing—review and editing; F.OdF.: supervision, writing—review and editing; D.S.H.: conceptualization, data curation, project administration, resources, supervision, writing—review and editing; W.L.C.: conceptualization, data curation, funding acquisition, investigation, methodology, project administration, resources, software, supervision, validation, visualization, writing—review and editing.

All authors gave final approval for publication and agreed to be held accountable for the work performed therein.

Conflict of interest declaration. We declare we have no competing interests.

Funding. W.G.L., D.S.H. and G.S.I.A. are supported by the Patient Centered Outcomes Research Institute (PCORI) ME-2020C1D-19393. The statements in this work are solely the responsibility of the authors and do not necessarily represent the views of PCORI, its Board of Governors or Methodology Committee. F.O.F. is supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) grant 301596/2022-0. G.S.I.A. is supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), finance code 001 and grant 88887.802848/2023-00. J.D.R. is supported by the US National Library of Medicine grant R00-LM013646.

References

1. Koza JR. 1994 Genetic programming as a means for programming computers by natural selection. *Stat. Comput.* **4**, 87–112. (doi:10.1007/BF00175355)
2. Udrescu SM, Tan A, Feng J, Neto O, Wu T, Tegmark M. 2020 AI Feynman 2.0: pareto-optimal symbolic regression exploiting graph modularity. In *Advances in neural information processing systems* (eds H Larochelle, M Ranzato, R Hadsell, MF Balcan, H Lin), pp. 4860–4871, vol. 33. Red Hook, NY: Curran Associates, Inc.
3. Lalande F, Matsubara Y, Chiba N, Taniai T, Igarashi R, Ushiku Y. 2023 A transformer model for symbolic regression towards scientific discovery. In *NeurIPS 2023 AI for science workshop*. Red Hook, NY: Curran Associates, Inc.
4. Udrescu SM, Tegmark M. 2020 AI feynman: a physics-inspired method for symbolic regression. *Sci. Adv.* **6**, eaay2631. (doi:10.1126/sciadv.aay2631)
5. Makke N, Chawla S. 2024 Interpretable scientific discovery with symbolic regression: a review. *Artif. Intell. Rev.* **57**, 1–32. (doi:10.1007/s10462-023-10622-0)
6. Angelis D, Sofos F, Karakasidis TE. 2023 Artificial intelligence in physical sciences: symbolic regression trends and perspectives. *Arch. Comput. Methods Eng.* **30**, 3845–3865. (doi:10.1007/s11831-023-09922-z)

7. La Cava WG, Lee PC, Ajmal I, Ding X, Solanki P, Cohen JB, Moore JH, Herman DS. 2023 A flexible symbolic regression method for constructing interpretable clinical prediction models. *NPJ Digit. Med.* **6**, 107. (doi:10.1038/s41746-023-00833-8)
8. Wilstrup C, Cave C. 2022 Combining symbolic regression with the Cox proportional hazards model improves prediction of heart failure deaths. *BMC Med. Inform. Decis. Mak.* **22**, 196. (doi:10.1186/s12911-022-01943-1)
9. Virgolin M, Alderliesten T, Bel A, Witteveen C, Bosman PAN. 2018 Symbolic regression and feature construction with GP-GOMEA applied to radiotherapy dose reconstruction of childhood cancer survivors. In *Proceedings of the genetic and evolutionary computationconference GECCO'18*, pp. 1395–1402. New York, NY, USA: Association for Computing Machinery. (doi:10.1145/3205455.3205604)
10. La Cava W, Danai K, Spector L, Fleming P, Wright A, Lackner M. 2016 Automatic identification of wind turbine models using evolutionary multiobjective optimization. *Renewable Energy* **87**, 892–902. (doi:10.1016/j.renene.2015.09.068)
11. La Cava W, Danai K, Lackner M, Spector L, Fleming P, Wright A. 2015 Automatic identification of closed-loop wind turbine dynamics via genetic programming. In *ASME 2015 Dynamic Systems and Control Conference*, Columbus, Ohio, USA. vol. 57250. New York, NY: American Society for Mechanical Engineers (ASME). (doi:10.1115/DSCC2015-9768)
12. Wang Y, Wagner N, Rondinelli JM. 2019 Symbolic regression in materials science. *MRS Commun.* **9**, 793–805. (doi:10.1557/mrc.2019.85)
13. Administration D, Food. 2022 Clinical decision support software - guidance for industry and food and drug administration staff https://www.fda.gov/regulatory-information/searchfda-guidance-documents/clinical-decision-support-software
14. Breiman L, Friedman J, Stone CJ, Olshen RA. 1984 *Classification and regression trees*. London, UK: Taylor & Francis.
15. Rudin C, Chen C, Chen Z, Huang H, Semenova L, Zhong C. 2022 Interpretable machine learning: fundamental principles and 10 grand challenges. *Stat. Surv.* **16**, 1–85. (doi:10.1214/21-SS133)
16. Churpek MM, Yuen TC, Park SY, Meltzer DO, Hall JB, Edelson DP. 2012 Derivation of a cardiac arrest prediction model using ward vital signs. *Crit. Care Med.* **40**, 2102–2108. (doi:10.1097/CCM.0b013e318250aa5a)
17. Subbe CP, Kruger M, Rutherford P, Gemmel L. 2001 Validation of a modified early warning score in medical admissions. *QJM* **94**, 521–526. (doi:10.1093/qjmed/94.10.521)
18. Tan ADA, Permejo CC, Torres MCD. 2022 Modified early warning score vs cardiac arrest risk triage score for prediction of cardiopulmonary arrest: a case-control study. *Indian J. Crit. Care Med.* **26**, 780–785. (doi:10.5005/jp-journals-10071-24242)
19. Guidetti V, Mandreoli F. 2024 Symbolic regression for transparent clinical decision support: a data-centric framework for scoring system development. In *CEUR workshop proceedings*, Villasimius, Sardinia, Italy, vol. 3741, pp. 604–614,
20. Xie F *et al*. 2022 Benchmarking emergency department prediction models with machine learning and public electronic health records. *Sci. Data* **9**, 658. (doi:10.1038/s41597-022-01782-9)
21. La Cava W, Orzechowski P, Burlacu B, Franca F, Virgolin M, Jin Y, Kommenda M, Moore J. 2021 Contemporary symbolic regression methods and their relative performance (eds J Vanschoren, S Yeung). In *Proceedings of the neural information processing systems track on datasets and benchmarks*. vol. 1. Red Hook, NY: Curran Associates, Inc.
22. Aldeia GSI, Zhang H, Bomarito G, Cranmer M, Fonseca A, Burlacu B, La Cava WG, França FO. 2025 Call for action: towards the next generation of symbolic regression benchmark. *arXiv Preprint* 2505.03977.
23. Johnson AE *et al*. 2023 MIMIC-IV, a freely accessible electronic health record dataset. *Sci. Data* **10**, 1. (doi:10.1038/s41597-022-01899-x)
24. Gama J. 2004 Functional trees. *Mach. Learn.* **55**, 219–250. (doi:10.1023/B:MACH.0000027782.67192.13)
25. Rusch T, Zeileis A. 2013 Gaining insight with recursive partitioning of generalized linear models. *J. Stat. Comput. Simul.* **83**, 1301–1315. (doi:10.1080/00949655.2012.658804)
26. Zhang H, Zhou A, Qian H, Zhang H. 2022 PS-tree: a piecewise symbolic regression tree. *Swarm Evol. Comput.* **71**, 101061. (doi:10.1016/j.swevo.2022.101061)

27. Doquet G. 2025 Unified piecewise symbolic regression (eds B Xue, L Manzoni, I Bakurov). In *Genetic programming*, pp. 190–206. Cham: Springer Nature Switzerland. (doi:10.1007/978-3-031-89991-1_12)
28. Fong KS, Motani M. 2024 Symbolic regression enhanced decision trees for classification tasks. AAAI **38**, 12033–12042. (doi:10.1609/aaai.v38i11.29091)
29. Wang HF, Wu KY. 2004 Hybrid genetic algorithm for optimization problems with permutation property. *Comput. Oper. Res.* **31**, 2453–2471. (doi:10.1016/S0305-0548(03)00198-9)
30. Chen C, Luo C, Jiang Z. 2017 Elite bases regression: a real-time algorithm for symbolic regression. In *2017 13th international conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD)*, Guilin, China, pp. 529–535. New York, NY: IEEE. (doi:10.1109/FSKD.2017.8393325)
31. Levenberg K. 1944 A method for the solution of certain non-linear problems in least squares. *Quart. Appl. Math.* **2**, 164–168. (doi:10.1090/qam/10666)
32. Marquardt DW. 1963 An algorithm for least-squares estimation of nonlinear parameters. J. Soc. Ind. Appl. Math. **11**, 431–441. (doi:10.1137/0111030)
33. Aldeia GSI, de França FO. 2022 Interaction-transformation evolutionary algorithm with coefficients optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion GECCO '22*, pp. 2274–2281. New York, NY, USA: Association for Computing Machinery. (doi:10.1145/3520304.3533987). https://dl.acm.org/doi/proceedings/10.1145/3520304.
34. Kommenda M, Burlacu B, Kronberger G, Affenzeller M. 2020 Parameter identification for symbolic regression using nonlinear least squares. *Genet. Program. Evolvable Mach.* **21**, 471–501. (doi:10.1007/s10710-019-09371-3)
35. Worm T, Chiu K. 2013 Prioritized grammar enumeration: symbolic regression by dynamic programming. In *Proceedings of the 15th annual conference on genetic and evolutionary computation GECCO '13*, pp. 1021–1028. New York, NY, USA: Association for Computing Machinery. (doi:10.1145/2463372.2463486)
36. Burlacu B, Kronberger G, Kommenda M. 2020 Operon C++: an efficient genetic programming framework for symbolic regression. In *Proceedings of the 2020 genetic and evolutionary computation conference companion GECCO '20*, pp. 1562–1570. New York, NY, USA: Association for Computing Machinery. (doi:10.1145/3377929.3398099)
37. Kamienny PA, d'Ascoli S, Lample G, Charton F. 2022 End-to-end symbolic regression with transformers (eds AH Oh, A Agarwal, D Belgrave, K Cho). In *Advances in neural information processing systems*, pp. 1–13.
38. Landajuela M, Lee CS, Yang J, Glatt R, Santiago CP, Aravena I, Mundhenk T, Mulcahy G, Petersen BK. 2022 A unified framework for deep symbolic regression (eds S Koyejo, S Mohamed, A Agarwal, D Belgrave, K Cho, A Oh). In *Advances in neural information processing systems*, vol. 35, pp. 33985–33998, New Orleans, LA: Curran Associates, Inc.
39. Shojaee P, Meidani K, Farimani AB, Reddy CK. 2023 Transformer-based planning for symbolic regression. In *Thirty-seventh conference on neural information processing systems*. Vancouver, Canada: Curran Associates, Inc.
40. La Cava W, Singh TR, Taggart J, Suri S, Moore JH. 2018 Learning concise representations for regression by evolving networks of trees. *arXiv Preprint* 1807.00981.
41. La Cava W, Spector L, Danai K. 2016 Epsilon-lexicase selection for regression. In *Proceedings of the genetic and evolutionary computation conference 2016*, pp. 741–748. Denver, CO: Association for Computing Machinery (ACM). (doi:10.1145/2908812.2908898)
42. Poli R, McPhee NF, Koza JR. 2008 *A field guide to genetic programming. [S.I]*. Morrisville, NC: Lulu Press.
43. Deb K, Pratap A, Agarwal S, Meyarivan T. 2002 A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Computat.* **6**, 182–197. (doi:10.1109/4235.996017)
44. Imai Aldeia GS, De França FO, La Cava WG. 2024 Inexact simplification of symbolic regression expressions with locality-sensitive hashing. In *Proceedings of the geneticand evolutionary computation conference GECCO '24*, pp. 896–904. New York, NY, USA: Association for Computing Machinery. (doi:10.1145/3638529.3654147)
45. Romano JD *et al*. 2022 PMLB v1.0: an open-source dataset collection for benchmarking machine learning methods. *Bioinformatics* (ed. J Kelso), **38**, 878–880. (doi:10.1093/bioinformatics/btab727)

46. Feynman RP, Leighton RB, Sands ML. 2006 The feynman lectures on physics. In *The feynman lectures on physics*, vol. 2. Boston, MA: Pearson/Addison-Wesley.
47. Feynman RP, Leighton RB, Sands M. 2015 The feynman lectures on physics, vol. I: the new millennium edition: mainly mechanics, radiation, and heat. In *The feynman lectures on physics*, vol. 1. Boston, MA: Basic Books.
48. Strogatz SH. 2018 *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. Boca Raton, FL: CRC press.
49. La Cava WG. 2025 *cavalab/srbench*. See https://github.com/cavalab/srbench.
50. Johnson A, Bulgarelli L, Pollard T, Celi LA, Mark R, Horng S. 2023 *MIMIC-IV-ED*. See https://physionet.org/content/mimic-iv-ed/2.2/.
51. Gupta M, Gallamoza B, Cutrona N, Dhakal P, Poulain R, Beheshti R. 2022 An extensive data processing pipeline for MIMIC-IV. In *Proceedings of the 2nd machine learning for health symposium*, vol. 193, pp. 311–325, New Orleans, LA: PMLR.
52. Gallamoza B, Cutrona N, mehak25, UDpranjal. 2024 *healthylaife/MIMIC-IV-Data-Pipeline*
53. Aldeia G, La Cava W, Romano J. 2025 *cavalab/brush*. See https://github.com/cavalab/brush.