Learning a Formula of Interpretability to Learn Interpretable Formulas

Marco Virgolin[1], Andrea De Lorenzo[2], Eric Medvet[2], and Francesca Randone[3]

[1] Centrum Wiskunde & Informatica, Amsterdam, the Netherlands
[2] Department of Engineering and Architecture, University of Trieste, Trieste, Italy
[3] IMT School for Advanced Studies Lucca, Lucca, Italy

Abstract. Many risk-sensitive applications require Machine Learning (ML) models to be interpretable. Attempts to obtain interpretable models typically rely on tuning, by trial-and-error, hyper-parameters of model complexity that are only loosely related to interpretability. We show that it is instead possible to take a meta-learning approach: an ML model of non-trivial Proxies of Human Interpretability (PHIs) can be learned from human feedback, then this model can be incorporated within an ML training process to directly optimize for interpretability. We show this for evolutionary symbolic regression. We first design and distribute a survey finalized at finding a link between features of mathematical formulas and two established PHIs, *simulatability* and *decomposability*. Next, we use the resulting dataset to learn an ML model of interpretability. Lastly, we query this model to estimate the interpretability of evolving solutions within bi-objective genetic programming. We perform experiments on five synthetic and eight real-world symbolic regression problems, comparing to the traditional use of solution size minimization. The results show that the use of our model leads to formulas that are, for a same level of accuracy-interpretability trade-off, either significantly more or equally accurate. Moreover, the formulas are also arguably more interpretable. Given the very positive results, we believe that our approach represents an important stepping stone for the design of next-generation interpretable (evolutionary) ML algorithms.

Keywords: explainable artificial intelligence · interpretable machine learning · symbolic regression · genetic programming · multi-objective

1 Introduction

Artificial Intelligence (AI), especially when intended as Machine Learning (ML), is increasingly pervading society. Although ML brings numerous advantages, it is far from being fault-prone, hence its use comes with risks [1,16,24,36]. In many cases, failures with serious consequences could have been foreseen and prevented, if the ML models had not been unintelligible, i.e., *black-boxes*. Nowadays, especially for high-stakes applications, practitioners, researchers, and policy makers increasingly ask for ML to be used responsibly, fairly, and ethically [8,15]. Therefore, decisions taken by ML models need to be explainable [1,2].

The field of eXplainable AI (XAI) studies techniques to provide explanations for the decisions taken by black-box models (or, more generally, AI systems), metrics that can be used as *Proxies of Human Interpretability* (PHIs), as well as ML algorithms meant for the synthesis of models that are immediately interpretable [1,36]. In this paper, we consider the latter case.

Several ML algorithms and techniques exist that are considered *capable* of synthesizing interpretable models. Among these, fitting linear models (e.g., by ordinary least squares or elastic net [53]), building decision trees [4], and evolutionary program synthesis [30] are often listed in surveys on XAI (see, e.g., [1,16]). Unfortunately, in general, it cannot be *guaranteed* that the model obtained by an ML algorithm will turn out to be interpretable. For example, when a decision tree is built, the more the tree grows deep, the less the chances of the tree being interpretable. Therefore, what is normally done is to repeat the ML training process (decision tree construction) with different hyper-parameter settings (tree depth) in a trial-and-error fashion, until a satisfactory model is obtained. Trial-and-error, of course, comes with time costs. Next to this, another important issue is the fact that hyper-parameters are mostly meant to control the bias-variance interplay [3], and are but loosely related to interpretability.

Multi-Objective Genetic Programming (MOGP) is a very interesting approach because, by its very nature, it mitigates the need for trial-and-error [30,52]. By evolving a population of solutions that encode ML models, MOGP can synthesize, in a single run, a plethora of models with trade-offs between accuracy and a chosen PHI. Obtaining multiple models at once enhances the chance that a model with a satisfying trade-off between accuracy and interpretability will be found quickly. Nonetheless, the problem of finding a suitable PHI remains. So far, the PHI that have been used were quite simplistic. For example, a common approach is to simply minimize the total number of model components (see Section 2 for more). In this paper, we propose a way to improve upon the use of simplistic PHIs, and we focus on the case of MOGP for symbolic regression, i.e., where models are sought that can be written as mathematical formulas.

Our proposal is composed of three main parts. We begin by showing how it is possible to learn a model of non-trivial PHIs. This can be seen as a concretization of an idea that was sketched in [11]: a data-driven approach can be taken to discover what features make ML models more or less interpretable. In detail, (1) we design a survey about mathematical formulas, to gather human feedback on formula interpretability according to two established PHIs: *simulatability* and *decomposability* [24] (see Section 3.1); (2) we process the survey answers and condense them into to a regression dataset that enables us to discover a non-trivial model of interpretability; (3) we incorporate the so-found model within an MOGP algorithm, to act as an estimator for the second objective (the first being the mean squared error): in particular, the model takes as input the features of a formula, and outputs an estimate of interpretability.

All of our code, including the data obtained from the survey, is available at: https://github.com/MaLeLabTs/GPFormulasInterpretability.

2 Related work

In this paper, we focus on using ML to obtain interpretable ML models, particularly in the form of formulas and by means of (MO)GP. We do not delve into XAI works where explanations are sought for the decisions made by a black-box model (see, e.g., [34,48]), nor where the black-box model needs to be approximated by an interpretable surrogate (e.g., a recent GP-based work on this is [14]). We refer to [1,16] as excellent surveys on various aspects of XAI. We describe the PHIs we adopt, and briefly mention works adopting them, in Section 3.1.

Regarding GP for the synthesis of ML models, a large amount of literature is focused on controlling complexity, but not primarily as a means to enable interpretability. Rather, the focus is on overfitting prevention, oftentimes (but not exclusively) by limiting bloat, i.e., the excessive growth of solution size [7,31,33,37,38,42,51]. Among these works, [13,46,39] share with us the use MOGP, but are different in that they use hand-crafted complexity metrics instead of taking a data-driven approach (respectively solution size, order of non-linearity, and a modification of solution size), and again these metrics are designed to control bloat and overfitting instead of enable interpretability ([46] does however discuss some effects on interpretability).

Among the works that use GP and focus on interpretability, [6] considers the evolution of rule-based classifiers, and evaluates them using a PHI that consists of dividing the number of conditions in the classifier by the number of classes. In [18], GP is used to evolve reinforcement learning policies as symbolic expressions, and complexity in interpretation is measured by accounting for variables, constants, and operations, with different ad-hoc weights. The authors of [43] study whether modern model-based GP can be useful when particularly compact symbolic regression solutions are sought, to allow interpretability. A very different take to enable or improve interpretability is taken in [22,41,45], where interpretability is sought by means of feature construction and dimensionality reduction. In [22] in particular, MOGP is used, with solution size as a simple PHI. Importantly, none of these works takes attempts to learn a PHI from data.

Perhaps the most similar work to ours is [27]. Like we do, the authors train an ML model (a deep residual network [17]) from pre-collected human-feedback to drive an evolutionary process, but for a very different aim, i.e., automatic art synthesis (the human-feedback is aesthetic rankings for images).

3 The survey

We prepared an online survey (http://mathquiz.inginf.units.it) to assess the simulatability and decomposability of mathematical formulas (we referred to [5] for survey-preparation guidelines). We begin by describing the two PHIs, and proceed with an overview of the content of the survey and the generation process. We provide full details on online supplementary material at: https://github.com/MaLeLabTs/GPFormulasInterpretability.

Given the formula $\frac{5x_1+1}{\cos(x_2-3.14)}$ and the input value(s) $[x_1 = 8.0, x_2 = 6.28]$, which option is closest to the output?
(a) -410.0
(b) -41.0
(c) 410.0
(d) -20.5

Consider the formula $5\sin^{0.5}(x - 3.14)$.
Which option best describes the behavior of the function as x varies in $[-1.0, 1.0]$?
(a) The function is bounded but not always defined
(b) The function is not bounded nor always defined
(c) The function is not bounded but always defined
(d) The function is bounded and always defined

Fig. 1. Examples of questions on simulatability (left) and decomposability (right).

3.1 Simulatability and decomposability

Simulatability and decomposability are two established PHIs, introduced in a seminal work on XAI [24]. Simulatability represents a measurable proxy for the capability of a person to contemplate an entire ML model, and is measured by assessing whether a human, given some input data, can reproduce the model's output within a reasonable error margin and time [24]. No specific protocol exists to perform the measurement. In [32], this PHI was measured as the absolute deviation between the human estimate for the output of a (linear) model and the actual output, given a set of inputs. With our survey, we measured the rate of correct answers to multiple choices questions on the output of a formula.

Decomposability represents the possibility that a model can be interpreted by parts: inputs, parameters, and (partial) calculations of a model need to admit an intuitive explanation [24]. For example, the coefficients of a linear model can be interpreted as the strengths of association between features and output. Decomposability is similar to the concept of *intelligibility* of [25]. As for simulatability, there exists no prescription on how to measure decomposability. We considered variables as important components to represent this PHI, and gathered answers (correct/wrong) on properties of the behavior of a formula when one of its variables varies within an interval.

3.2 Overview on the survey and results

We implemented the survey as a webpage, consisting of an introductory section to assess the respondents' level of familiarity with formulas, followed by eight questions, four about simulatability, and four about decomposability. The eight questions are randomly selected when the webpage is loaded, from a pre-generated database that contains 1000 simulatability and 1000 decomposability questions, each linked to one of 1000 automatically generated formulas. Figure 1 shows examples of these questions. Each and every question presents four possible answers, out of which only one is correct. Alongside each question, the user is asked to state how confident (s)he is about the answer, on a scale from 1 to 4.

The 1000 formulas were encoded with trees, and randomly generated with a half-and-half initialization of GP [30] (max depth 4). The set of leaf nodes for the trees included 4 different variables, and constants that were either integers (from 0 to 10) or multiples of π (3.14 or 6.28). The possible operations were $+$,

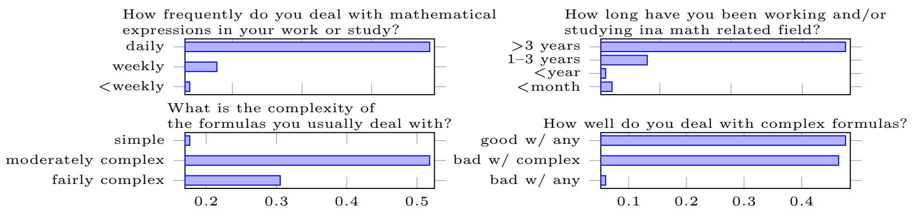


Fig. 2. Distribution of answers about user expertise.

$-$, $\times$, $\div$, $^\wedge$, $\sqrt{\cdot}$, sin, and cos. We performed rejection sampling and automatic simplifications to avoid presenting fundamentally uninteresting functions (e.g., constant ones), or functions with exploding output (e.g., due to $^\wedge$).

For simulatability questions, the user was either asked to pick the correct 2D graph representing the behavior of the (one variable) formula, or to choose the best estimate of the output of the (multi-variable) formula, given values for the variables. Decomposability questions asked whether the formula was (not) bounded, (not) always defined, (not) null in some points, (not) negative in some points, for one variable changing in a given interval and the others being fixed.

We distributed the survey by emailing research groups and departments within the institutes of the authors, targeting both students and faculty members. We further shared the survey on Reddit (subreddit CasualMath) and Twitter. We obtained 334 responses in ≈ 35 days, corresponding to 2672 answers. Figure 2 shows the distribution of answers to the introductory part of the survey.

4 Learning a formula of interpretability

We now describe how we condense the survey answers into a regression dataset, and use this dataset to learn an ML model (as a formula) of interpretability.

We begin with replacing each question with a set of feature values that represents the formula contained in the question (explained in detail below). We obtain multiple identical sets of feature values with different outcomes in terms of correctness and confidence. We merge equal sets of feature values into a single sample, taking the ratio of correct answers and the mean confidence. In doing so, we do not distinguish between answers belonging to simulatability or decomposability, assuming they are equally good PHIs. We also remark that we did not make expertise-based partitions because of the limited number of respondents and the skew in expertise distribution (Figure 2).

As label to regress, we take the product between correctness ratio and confidence (the latter normalized to have values $\frac{0}{3}$, $\frac{1}{3}$, $\frac{2}{3}$, $\frac{3}{3}$). We choose to weight by confidence because, arguably, the less a user is confident about the answer, the less (s)he feels (s)he interprets the formula correctly. Essentially, this is a new PHI synthesized out of simulatability and decomposability, that takes confidence into account. From now on, we refer to this PHI as ϕ.

The choice of what formula features are considered is of crucial importance as it determines the way the answers are merged. We ultimately consider the following features: the size ℓ of the formula (counting variables, constants, and operations), the number n_o of operations, the number n_nao of non-arithmetic operations, the number n_naoc of consecutive compositions of non-arithmetic operations. Note that the number of variables or constants is $\ell - n_\text{o}$, and the number of arithmetic operations is $n_\text{o} - n_\text{nao}$.

By merging answers sharing the same values for the aforementioned four features, and excluding merged samples composed by less than 10 answers for robustness, we obtain a small regression dataset with 73 samples.

4.1 Learning the model

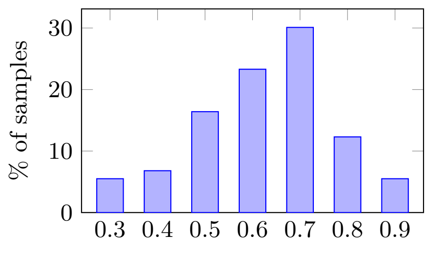
Figure 3a shows the distribution of ϕ. Since this distribution is not uniform, similarly to what is done for classification with imbalanced class frequency, we weight samples by the inverse frequency of the bin they belong to.

To obtain a readable ML model and due to the small number of samples, we choose to fit a elastic net linear model [53] of the four features with stochastic gradient descent, and validate it with leave-one-out cross-validation. We refer the reader interested in the details of this process (which includes, e.g., hyper-parameter tuning) to https://github.com/MaLeLabTs/GPFormulasInterpretability. The leave-one-out cross-validation scores a (weighted) training $R^2 = 0.506$, and (weighted) test $R^2 = 0.545$ (mean weighted absolute error of 26 %). The distribution of the model coefficients optimized across the folds is shown in Figure 3b.

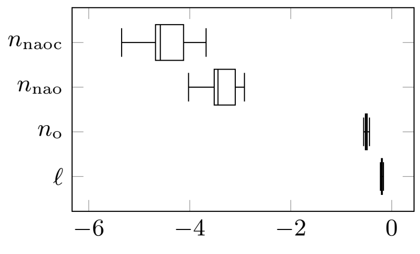
We take the average coefficients found during the cross-validation to obtain the final model of ϕ (from now on, considered as a percentage):

$$\mathcal{M}^\phi(\ell, n_\text{o}, n_\text{nao}, n_\text{naoc}) = 79.1 - 0.2\ell - 0.5 n_\text{o} - 3.4 n_\text{nao} - 4.5 n_\text{naoc}. \quad (1)$$

By observing $\mathcal{M}^\phi$, it can be seen that each feature plays a role in lowering interpretability, yet by different magnitudes; n_naoc is the most important factor.



(a) Histogram of ϕ values.



(b) Boxplots of learned coefficients.

Fig. 3. Salient information about the learning data and the linear model.

5 Exploiting the model of interpretability in MOGP

The experimental setup adopted for the use of the model $\mathcal{M}^{\phi}$ within an MOGP algorithm for symbolic regression is presented next. We describe the algorithm we use, its objectives, the datasets we consider, and the evaluation process.

MOGP by NSGA-II. We use a GP version of NSGA-II [9], the most popular multi-objective evolutionary algorithm, and refer to it as *NSGP* (the same name has been used in different works, e.g., [23,47,49]). We use traditional settings (all described in [30]): tree-based encoding; ramped half-and-half initialization (min and max depth of 1 and 6 respectively); and tournament selection (size 2, default in NSGA-II). The crossover operator is subtree crossover (probability of 0.9, default in NSGA-II). The mutation operator is one-point mutation (probability of $1/\ell$ for each tree node, with ℓ the number of nodes).

We set the population size to 1000 and perform 100 generations across all experiments. The possible tree leaves are the problem variables and an ephemeral random constant [30], with random values from $\mathcal{U}(-5,+5)$. The operations are $+$, $-$, $\times$, $\div_p$, sin, cos, exp, and $\log_p$. Protection of division by zero is implemented by $\div_p(i_1, i_2) := \text{sign}(i_2)\frac{i_1}{|i_2|+\epsilon}$. Similarly, the logarithm is protected by taking as argument the absolute value of the input plus ϵ. We use $\epsilon = 10^{-6}$. Trees are not allowed to grow past 100 nodes, as they would definitely be not interpretable.

Our (Python 3) implementation of NSGP (including an interface to scikit-learn [29]) is available at: https://github.com/marcovirgolin/pyNSGP.

Objectives. We consider two competing objectives: error vs. interpretability. For the first objective, we consider the Mean Squared Error (MSE) with *linear scaling* [20,21], i.e., $\text{MSE}^{\text{lin.scal.}}(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - a - b\hat{y}_i)^2$. The use of the optimal (on training data) affine transformation coefficients a, b corresponds to computing an absolute correlation. From now on, we simply refer to this as MSE.

For the second objective, we consider two possibilities. The first one is realized by using our model $\mathcal{M}^{\phi}$: we extract the features from the tree to be evaluated, feed them to $\mathcal{M}^{\phi}$, and take the resulting estimate of ϕ. To conform with objective minimization, we actually seek to minimize the opposite of this estimate (we also ignore the intercept term of Equation (1)). We call NSGP$^{\phi}$ the version of NSGP using this second objective.

The second possibility is to solely minimize the number of nodes ℓ. This approach, despite its simplicity, is very popular (see Section 2). Here we use it as a baseline for comparison. We refer to NSGP using ℓ minimization as NSGP$^{\ell}$.

The objectives based on ℓ and ϕ are in a comparable scale (considering ϕ as a percentage). To bring the first objetive to a similar scale (since scale impacts the crowding distance [9]), during the evolution we multiply the MSE by $\frac{100}{\sigma^2(y)}$ (i.e., predicting the mean $\mu(y)$ achieves an error of 100).

NSGP measures the quality of solutions according to the same criteria of [9] (domination ranking and crowding distance). We will report results relative to the front of non-dominated solutions $\mathcal{F}$ obtained at the end of the run. Recall that a solution is non-dominated if there exists no other solution that is better in at least one objective and not worse in all others. In other words, $\mathcal{F}$ contains the solutions with best trade-offs between the objectives.

Datasets and evaluation. We consider 13 regression datasets in total, see Table 1. The first 5 datasets are synthetic (S-) and are recommended in [50]. The other 8 regard real-world data (R-) and are (mostly) taken from the UCI machine learning repository [12] and used in recent literature (e.g., [35,44]).

We treat all datasets the same. We apply standardization, i.e., all features are set to have zero mean and unit standard deviation. Before each run, we partition the dataset in exam at random, splitting it into 80 % samples for training, 10 % for validation, and 10 % for testing. The training set is used by NSGP to evolve the solutions. The other sets are used to assess generalization, as is good practice in ML [3]. In particular, using the final population, we re-compute the MSE of the solutions w.r.t. the validation set, and compute the front of non-dominated solutions $\mathcal{F}$ based on this. The MSE of the solutions in this front is finally re-evaluated on the test set (test MSE).

Because dataset partitioning as well as NSGP are stochastic, we perform 50 runs per dataset. To evaluate whether differences in results between NSGP$^{\phi}$ and NSGP$^{\ell}$ are significant, we use the Wilcoxon signed-rank test [10] to the 95 % confidence level, including Holm-Bonferroni correction [19].

Table 1. Datasets used in this work. For the synthetic datasets, N is chosen by considering the largest between the training set and test set proposed in [50].

Dataset	Abbr.	N	D	$\mu(y)$	$\sigma(y)$
Keijzer 6	S-Ke6	121	1	4.38	0.98
Korns 12	S-K12	10 000	5	2.00	1.06
Nguyen 7	S-Ng7	20	1	0.79	0.48
Pagie 1	S-Pa1	625	2	1.56	0.49
Vladislav. 4	S-Vl4	5000	5	0.49	0.19

Dataset	Abbr.	N	D	$\mu(y)$	$\sigma(y)$
Airfoil	R-Air	1503	5	124.8	6.9
Boston housing	R-Bos	506	13	22.5	9.2
Dow chemical	R-Dow	1066	57	3.0	0.4
Diabetes	R-Dia	442	10	152.1	77.0
Energy cooling	R-EnC	768	8	24.6	9.5
Energy heating	R-EnH	768	8	22.3	10.1
Tower	R-Tow	4999	25	342.1	87.8
Yacht	R-Yac	308	6	10.5	15.1

6 Results

Fitting and generalization error. We begin by reporting quantitative results of the models in terms of training and test MSE. Although the test MSE is what ultimately matters in practical applications (i.e., a good formula is one that generalizes to unseen data), we also show the training MSE because it reflects the capability of an algorithm to optimize as much as possible. We present results for different trade-off levels τ. Specifically, τ is the percentile rank of the solutions in the non-dominated front $\mathcal{F}$ ordered by increasing MSE: $\tau = 1$ considers the solution with best MSE and worst PHI; $\tau = 100$ considers the solution with worst MSE and best PHI (see Figure 4). Table 2 shows the MSE obtained by NSGP$^{\phi}$ and NSGP$^{\ell}$ at training and test times, alongside the values of ϕ and ℓ, for the MSE-specialized part of the fronts ($\tau = 5, 25, 50$).

For a same τ, solutions found by NSGP$^{\phi}$ have typically larger ℓ than those found by NSGP$^{\ell}$. The vice versa also holds, as can be expected. Notable examples appear for $\tau = 25$ in S-Pa1 and R-EnC/H: NSGP$^{\phi}$ achieves approximately double ℓ compared to the NSGP$^{\ell}$, while the latter achieves approximately double ϕ compared to the former.

Regarding the training MSE, the use of ϕ leads to the best optimization. This is particularly evident for $\tau = 5$ where all results are significantly better when using NSGP$^{\phi}$, except for S-Vl4. Using ϕ instead of ℓ has a smaller detrimental impact on finding well-fitting formulas. A plausible explanation is that NSGP$^{\phi}$ explores the search space better than NSGP$^{\ell}$. This hypothesis is also supported by considering the sizes of the non-dominated fronts $|\mathcal{F}|$: although the fronts are generally small for both ϕ and ℓ (reasonable because both depend on discrete properties of the solutions [39]), they are consistently larger when ϕ is used.

Less differences between NSGP$^{\phi}$ and NSGP$^{\ell}$ are significant when considering the test MSE (also due to Holm-Bonferroni correction). This is a normal consequence of assessing generalization as gains in training errors are lost due to (some) overfitting. What is important tough is that NSGP$^{\phi}$ remains preferable. For $\tau = 5$ ($\tau = 25$), this is the case for 9 (7) out of 12 datasets.

Qualitative results. We delve deeper into the results to assess the behavior of NSGP using ϕ and ℓ, from a qualitative perspective. We consider three datasets: S-Vl4 where no version of NSGP is superior to the other; R-Bos where NSGP$^{\phi}$ is only better at training time; and R-EnH, where NSGP$^{\phi}$ is favorable also at test time. Figure 4 shows *all* validation fronts obtained from the 50 runs, re-evaluated in terms of test MSE for both versions of NSGP, and plotted w.r.t. ϕ (left plots) and ℓ (right plots). We also show, for $\tau \in \{1, 50, 100\}$, the solutions obtained by considering always the first run (seed 1 in the results on our online code repository at https://github.com/MaLeLabTs/GPFormulasInterpretability).

The scatter plots show that, in general, NSGP$^{\phi}$ obtains more points with small test MSE. This is most evident for R-EnH, where the results are found to be statistically significant. Note how, instead, this is not the case for $\tau = 100$ in S-Vl4, where in fact the use of ϕ is no better than the use of ℓ (see Table 2).

By visually inspecting the formulas, we find results that are in line with what found in Table 2. Formulas found by NSGP$^{\phi}$ with small MSE ($\tau = 1, 50$) can often be (slightly) longer than their counterpart found by NSGP$^{\ell}$ (except for S-Vl4), however, they typically contain less non-arithmetic operations, and less of their compositions. Even for very small formulas, those found NSGP$^{\ell}$ rely more on non-arithmetic operations, meaning these operations help achieving small MSE, at least up to the validation stage. All in all, the most complex formulas

Table 2. Median performance from 50 runs of the solutions found by NSGP$^\phi$ and NSGP$^\ell$ at different trade-off levels τ ($\tau = 1$ for best MSE, $\tau = 100$ for best PHI). Median front sizes ($|\mathcal{F}|$) are computed w.r.t. the validation set. MSE values in bold for one version of NSGP are significantly lower than the corresponding ones for the other version of NSGP at the 95 % confidence level after Holm-Bonferroni correction.

		NSGP$^\phi$					NSGP$^\ell$						
Dataset	τ	Train MSE	Test MSE	ϕ	ℓ	$\lvert\mathcal{F}\rvert$	Train MSE	Test MSE	ϕ	ℓ	$\lvert\mathcal{F}\rvert$	Train p-value	Test p-value
	5	**0.000**	**0.001**	11.4	11		0.007	0.006	14.5	8		0.000	0.000
S-Ke6	25	**0.001**	**0.002**	9.4	8	7	0.013	0.007	13.5	6	5	0.000	0.000
	50	**0.005**	**0.007**	3.8	7		0.023	0.023	7.4	4		0.000	0.000
	5	**0.997**	0.998	2.9	7		0.998	0.997	7.4	4		0.000	0.924
S-K12	25	**0.998**	0.998	2.9	7	3	0.998	0.997	7.4	4	2	0.000	0.941
	50	**0.998**	0.997	2.0	5		0.998	0.997	7.4	3		0.000	0.454
	5	**0.000**	**0.000**	4.7	9		0.004	0.003	12.6	4		0.000	0.000
S-Ng7	25	**0.001**	**0.001**	2.9	7	4	0.005	0.003	12.6	4	2	0.000	0.000
	50	**0.001**	**0.001**	2.0	5		0.005	0.003	12.6	3		0.000	0.000
	5	**0.174**	**0.190**	15.9	16		0.216	0.221	22.8	7		0.000	0.001
S-Pa1	25	0.221	0.231	14.1	12	10	0.257	0.269	19.7	6	6	0.038	0.004
	50	0.396	0.392	10.5	8		0.338	0.387	13.5	5		0.029	0.950
	5	0.509	0.536	13.9	9		0.580	0.563	18.1	8		0.194	0.241
S-Vl4	25	0.616	0.621	11.4	8	6	0.632	0.611	18.1	6	5	0.398	0.579
	50	0.770	0.719	10.5	6		**0.656**	0.684	12.0	5		0.000	0.004
	5	**0.501**	**0.519**	5.5	13		0.566	0.586	2.3	5		0.000	0.000
R-Air	25	**0.534**	**0.538**	4.7	10	6	0.566	0.586	2.3	5	3	0.000	0.000
	50	**0.565**	**0.586**	2.0	5		0.596	0.624	1.3	3		0.000	0.000
	5	**0.245**	0.287	4.7	9		0.281	0.338	7.4	4		0.000	0.057
R-Bos	25	**0.254**	0.290	3.8	9	5	0.282	0.338	7.4	4	3	0.000	0.021
	50	**0.283**	0.332	2.0	5		0.347	0.355	1.3	3		0.000	0.054
	5	**0.510**	0.546	2.9	7		0.531	0.578	1.3	3		0.000	0.051
R-Dia	25	**0.515**	0.546	2.9	7	4	0.533	0.577	1.3	3	2	0.000	0.046
	50	**0.525**	0.551	2.0	5		0.538	0.571	1.3	3		0.000	0.482
	5	**0.336**	**0.357**	3.8	9		0.449	0.445	2.3	3		0.000	0.000
R-Dow	25	**0.369**	**0.372**	3.8	9	4	0.449	0.451	2.3	3	2	0.000	0.000
	50	**0.395**	**0.418**	2.0	5		0.469	0.466	1.3	3		0.000	0.000
	5	**0.099**	**0.108**	7.3	15		0.149	0.145	14.5	7		0.000	0.000
R-EnC	25	**0.104**	**0.113**	5.5	12	6	0.157	0.155	13.8	7	4	0.000	0.000
	50	**0.117**	**0.127**	3.8	9		0.175	0.176	13.5	5		0.000	0.000
	5	**0.082**	**0.085**	6.0	13		0.130	0.132	14.5	8		0.000	0.000
R-EnH	25	**0.085**	**0.087**	4.7	11	5	0.142	0.141	13.5	7	5	0.000	0.000
	50	**0.089**	**0.098**	2.9	7		0.164	0.162	8.4	5		0.000	0.000
	5	**0.290**	**0.288**	3.8	9		0.373	0.381	8.4	6		0.000	0.000
R-Tow	25	**0.298**	**0.302**	2.9	7	4	0.379	0.389	3.3	5	4	0.000	0.000
	50	**0.371**	**0.370**	2.0	5		0.449	0.457	7.4	4		0.000	0.000
	5	**0.011**	**0.014**	11.4	13		0.013	0.017	7.4	4		0.000	0.000
R-Yac	25	**0.012**	0.016	5.5	11	9	0.013	0.017	7.4	4	2	0.000	0.037
	50	0.015	0.024	3.8	9		0.013	**0.018**	7.4	4		0.006	0.000

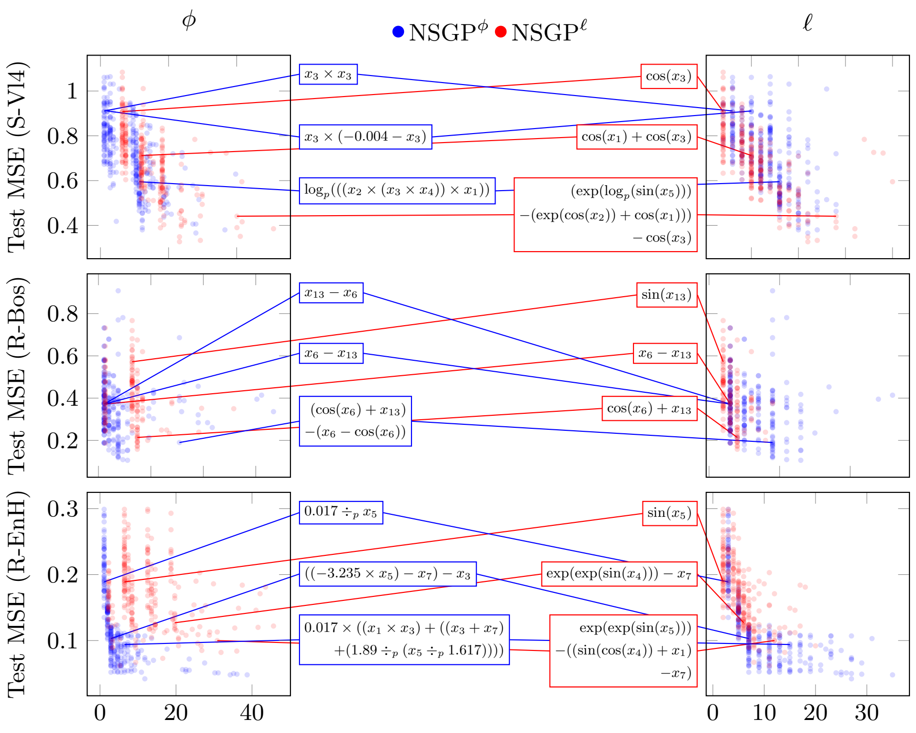


Fig. 4. Scatter plots of validation fronts as re-evaluated on the test set for all 50 runs, in terms of ϕ (left column) and ℓ (right column). Formulas in the middle are picked from the front of run 1, using $\tau = 1$ (bottom), 50 (middle), 100 (top). Note that $x_{13} - x_6$ and $x_6 - x_{13}$ (R-Bos) are equivalent due to linear scaling.

found by NSGP$^\phi$ are either more easily or similarly interpretable than those found by NSGP$^\ell$.

7 Discussion

To realize our data-driven approach, we relied on a survey aimed at measuring human-interpretability of mathematical formulas. While we did our best to design a survey that could gather useful human feedback, a clear limitation of our work is the relatively small number of survey respondents (334), which in turn led to obtaining a relatively small dataset (73 samples, 4 formula features). Fitting of a high-bias (linear) model resulted in a decent test R^2 of 0.5, while having the model be interpretable itself. Still, the model need not be interpretable. With more data available in the future, we will investigate the use of a larger number of more sophisticated features [26], and the use of more complex (possibly even black-box) models. Moreover, our approach can also be investigated for ML models other than formulas (e.g., decision trees).

In terms of results with NSGP, we found that ϕ allows the discovery of good solutions w.r.t. the competing objective, i.e., the MSE, better than ℓ. We also found that using ϕ leads to the discovery of larger fronts. There is no reason to expect this outcome beforehand, as ϕ was not designed to achieve this. We believe these findings boil down to one fundamental reason: diversity preservation. Because the estimation of ϕ relies on more features compared to the measurement of ℓ, more solutions can co-exist that do not dominate each other. Hence, the use of ϕ fares better against premature convergence [40].

Regarding the examples of formulas we obtained, one may think that ϕ leads to arguably more interpretable formulas than ℓ simply because it accounts for non-arithmetic operations (and their composition). In fact, we agree that ℓ is simplistic. However, we believe that minimizing ℓ represents one of the first baselines to compare against (and it was the only one we found being used to specifically promote interpretability [22]), and that designing a competitive baseline is non-trivial. We will investigate this further in future work.

What about formula simplification? We did not present results regarding formulas after a simplification step. We attempted to use the sympy library [28] to assess the effect of formula simplification during the evolution, but to no avail as runtimes exploded. Moreover, we looked at what happens if we simplify (with sympy) the formulas in the final front, and re-compute their ϕ and ℓ. Results were mixed. For example, regarding the three datasets of Figure 4, re-measuring ϕ and ℓ after simplification led to a mean improvement ratio of 1.08 (1.17) and 1.00 (1.00) respectively, when all (only the most complex) formulas from the fronts were considered. Hence, the use of ϕ seems more promising than ℓ w.r.t. simplification. Yet, as improvements were small (also in visual assessments), further investigation will be needed.

8 Conclusion

We presented a data-driven approach to learn, from responses to a survey on mathematical formulas we designed, a model of interpretability. This model is itself an interpretable (linear) formula, with reasonable properties. We plugged-in this model within multi-objective genetic programming to promote formula interpretability in symbolic regression, and obtained significantly better results when comparing with traditional formula size minimization. As such, our approach represents an important step towards better interpretable machine learning, especially by means of multi-objective evolution. Furthermore, the model we found can be used as a proxy of formula interpretability in future studies.

Acknowledgments and author contributions

We thank the Maurits en Anna de Kock Foundation for financing a high-performance computing system that was used in this work. Author contributions, in order of importance, follow. Conceptualization: M.V.; methodology: M.V., E.M.; software: M.V., A.D.L., F.R.; writing: M.V., E.M., A.D.L., F.R.

References

1. Adadi, A., Berrada, M.: Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). IEEE Access **6**, 52138–52160 (2018)
2. Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al.: Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion **58**, 82–115 (2020)
3. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
4. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth (1984)
5. Burgess, T.F.: Guide to the design of questionnaires. A general introduction to the design of questionnaires for survey research. University of Leeds, 2001 (2001)
6. Cano, A., Zafra, A., Ventura, S.: An interpretable classification rule mining algorithm. Information Sciences **240**, 1–20 (2013)
7. Chen, Q., Zhang, M., Xue, B.: Structural risk minimization-driven genetic programming for enhancing generalization in symbolic regression. IEEE Transactions on Evolutionary Computation **23**(4), 703–717 (2018)
8. Chouldechova, A.: Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. Big Data **5**(2), 153–163 (2017)
9. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
10. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7**(Jan), 1–30 (2006)
11. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608 (2017)
12. Dua, D., Graff, C.: UCI machine learning repository (2017), archive.ics.uci.edu/ml
13. Ekárt, A., Nemeth, S.Z.: Selection based on the Pareto nondomination criterion for controlling code growth in genetic programming. Genetic Programming and Evolvable Machines **2**(1), 61–73 (2001)
14. Evans, B.P., Xue, B., Zhang, M.: What's inside the black-box? A genetic programming method for interpreting complex machine learning models. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 1012–1020 (2019)
15. Goodman, B., Flaxman, S.: European union regulations on algorithmic decision-making and a "right to explanation". AI Magazine **38**(3), 50–57 (2017)
16. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM Computing Surveys **51**(5) (2018)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
18. Hein, D., Udluft, S., Runkler, T.A.: Interpretable policies for reinforcement learning by genetic programming. Engineering Applications of Artificial Intelligence **76**, 158–169 (2018)
19. Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics pp. 65–70 (1979)
20. Keijzer, M.: Improving symbolic regression with interval arithmetic and linear scaling. In: European Conference on Genetic Programming. pp. 70–82. Springer (2003)

21. Keijzer, M.: Scaled symbolic regression. Genetic Programming and Evolvable Machines **5**(3), 259–269 (2004)
22. Lensen, A., Xue, B., Zhang, M.: Genetic programming for evolving a front of interpretable models for data visualization. IEEE Transactions on Cybernetics pp. 1–15 (2020)
23. Liang, Y., Zhang, M., Browne, W.N.: Multi-objective genetic programming for figure-ground image segmentation. In: Australasian Conference on Artificial Life and Computational Intelligence. pp. 134–146. Springer (2016)
24. Lipton, Z.C.: The mythos of model interpretability. Queue **16**(3), 31–57 (2018)
25. Lou, Y., Caruana, R., Gehrke, J.: Intelligible models for classification and regression. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 150–158. ACM (2012)
26. Maruyama, M., Pallier, C., Jobert, A., Sigman, M., Dehaene, S.: The cortical representation of simple mathematical expressions. Neuroimage **61**(4), 1444–1460 (2012)
27. McCormack, J., Lomas, A.: Understanding aesthetic evaluation using deep learning. In: Artificial Intelligence in Music, Sound, Art and Design: 9th International Conference, EvoMUSART 2020. pp. 118–133. Springer (2020)
28. Meurer, A., Smith, C.P., Paprocki, M., Čertík, O., Kirpichev, S.B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J.K., Singh, S., Rathnayake, T., Vig, S., Granger, B.E., Muller, R.P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M.J., Terrel, A.R., Roučka, v., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., Scopatz, A.: SymPy: Symbolic computing in Python. PeerJ Computer Science **3**, e103 (2017)
29. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
30. Poli, R., Langdon, W.B., McPhee, N.F., Koza, J.R.: A field guide to genetic programming. Lulu.com (2008)
31. Poli, R., McPhee, N.F.: Parsimony pressure made easy: Solving the problem of bloat in GP. In: Theory and Principled Methods for the Design of Metaheuristics, pp. 181–204. Springer (2014)
32. Poursabzi-Sangdeh, F., Goldstein, D.G., Hofman, J.M., Vaughan, J.W., Wallach, H.: Manipulating and measuring model interpretability. arXiv preprint arXiv:1802.07810 (2018)
33. Raymond, C., Chen, Q., Xue, B., Zhang, M.: Genetic programming with Rademacher complexity for symbolic regression. In: IEEE Congress on Evolutionary Computation (CEC). pp. 2657–2664 (2019)
34. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should I trust you?" Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1135–1144 (2016)
35. Ruberto, S., Terragni, V., Moore, J.H.: SGP-DT: Semantic genetic programming based on dynamic targets. In: Genetic Programming: 23rd European Conference, EuroGP 2020. pp. 167–183. Springer (2020)
36. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence **1**(5), 206–215 (2019)
37. Sambo, A.S., Azad, R.M.A., Kovalchuk, Y., Indramohan, V.P., Shah, H.: Time control or size control? Reducing complexity and improving accuracy of genetic

programming models. In: Hu, T., Lourenço, N., Medvet, E., Divina, F. (eds.) Genetic Programming. pp. 195–210. Springer International Publishing (2020)
38. Silva, S., Dignum, S., Vanneschi, L.: Operator equalisation for bloat free genetic programming and a survey of bloat control methods. Genetic Programming and Evolvable Machines **13**(2), 197–238 (2012)
39. Smits, G.F., Kotanchek, M.: Pareto-front exploitation in symbolic regression. In: Genetic Programming Theory and Practice II, pp. 283–299. Springer (2005)
40. Squillero, G., Tonda, A.: Divergence of character and premature convergence: A survey of methodologies for promoting diversity in evolutionary optimization. Information Sciences **329**, 782–799 (2016)
41. Tran, B., Xue, B., Zhang, M.: Genetic programming for multiple-feature construction on high-dimensional classification. Pattern Recognition **93**, 404–417 (2019)
42. Vanneschi, L., Castelli, M., Silva, S.: Measuring bloat, overfitting and functional complexity in genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 877–884 (2010)
43. Virgolin, M., Alderliesten, T., Witteveen, C., Bosman, P.A.N.: Improving model-based genetic programming for symbolic regression of small expressions. Accepted in Evolutionary Computation. ArXiv preprint arXiv:1904.02050 (2019)
44. Virgolin, M., Alderliesten, T., Bosman, P.A.N.: Linear scaling with and within semantic backpropagation-based genetic programming for symbolic regression. In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 1084–1092. GECCO '19, Association for Computing Machinery (2019)
45. Virgolin, M., Alderliesten, T., Bosman, P.A.N.: On explaining machine learning models by evolving crucial and compact features. Swarm and Evolutionary Computation **53**, 100640 (2020)
46. Vladislavleva, E.J., Smits, G.F., Den Hertog, D.: Order of nonlinearity as a complexity measure for models generated by symbolic regression via Pareto genetic programming. IEEE Transactions on Evolutionary Computation **13**(2), 333–349 (2008)
47. Wang, P., Tang, K., Weise, T., Tsang, E., Yao, X.: Multiobjective genetic programming for maximizing ROC performance. Neurocomputing **125**, 102–118 (2014)
48. Wang, W., Shen, J.: Deep visual attention prediction. IEEE Transactions on Image Processing **27**(5), 2368–2378 (2017)
49. Watchareeruetai, U., Matsumoto, T., Takeuchi, Y., Kudo, H., Ohnishi, N.: Construction of image feature extractors based on multi-objective genetic programming with redundancy regulations. In: IEEE International Conference on Systems, Man and Cybernetics. pp. 1328–1333. IEEE (2009)
50. White, D.R., Mcdermott, J., Castelli, M., Manzoni, L., Goldman, B.W., Kronberger, G., Jaśkowski, W., O'Reilly, U.M., Luke, S.: Better GP benchmarks: Community survey results and proposals. Genetic Programming and Evolvable Machines **14**(1), 3–29 (2013)
51. Zhang, B.T., Mühlenbein, H.: Balancing accuracy and parsimony in genetic programming. Evolutionary Computation **3**(1), 17–38 (1995)
52. Zhao, H.: A multi-objective genetic programming approach to developing Pareto optimal decision trees. Decision Support Systems **43**(3), 809–826 (2007)
53. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **67**(2), 301–320 (2005)