Analytic Continued Fractions for Regression: Results on 352 datasets from the physical sciences

1st Pablo Moscato
School of Elect. Engg. and Computing
The University of Newcastle
Callaghan, NSW 2308, Australia
Pablo.Moscato@newcastle.edu.au

2nd Haoyuan Sun
California Institute of Technology
Pasadena, CA, USA
hsun2@caltech.edu

3rd Mohammad Nazmul Haque
School of Elect. Engg. and Computing
The University of Newcastle
Callaghan, NSW 2308, Australia
Mohammad.Haque@newcastle.edu.au

***Abstract*—We report on the results of a new memetic algorithm that employs analytic continued fractions as the basic representation of mathematical functions used for regression problems. We study the performance of our method in comparison with other ten machine learning approaches provided by the `scikit-learn` software collection. We used 352 datasets collected by Schaffer, which originated from real experiments in the physical sciences at the turn of the 20th century for which measurements were tabulated, and a governing functional relationship was postulated. Using leave-one-out cross-validation, in training our method ranks first in 350 out of the 352 datasets. Only six machine learning algorithms ranked first in at least one of the 352 datasets on testing; our approach ranked first 192 times, i.e. more all of the other algorithms combined. The results favourably speak about the robustness of our methodology. We conclude that the use of analytic continued fractions in regression deserves further study and we also advocate that Schaffer's data collection should also be included in the repertoire of datasets to test the performance of machine learning and regression algorithms.**

***Index Terms*—memetic computing, regression, analytic continued fraction.**

I. Introduction

In 2019, Sun and Moscato introduced a new approach for multivariate regression using analytic continued fractions as the basic representation of mathematical functions [1]. They presented results using a memetic algorithm [2] to identify the variables and the coefficients that best approximate an unknown target function. Six challenging real-world datasets were studied and compared with one of the state-of-the-art Genetic Programming approaches at the time. They have based their proposal in an old result of the theory of continued fractions. Indeed, the mathematical foundations on which this new representation for machine learning relies on can be traced to a theorem published by Leonard Euler in 1748 [3]. This great mathematician proved an equality between a finite sum of products and a finite continued fraction: let a sum of products, SP, be written as $SP = a_0 + a_0a_1 + a_0a_1a_2 + \cdots + a_0a_1a_2\cdots a_n$, then

$$SP = \cfrac{a_0}{1 - \cfrac{a_1}{1 + a_1 - \cfrac{a_2}{1 + a_2 - \cfrac{\ddots}{\ddots \cfrac{a_{n-1}}{1 + a_{n-1} - \cfrac{a_n}{1 + a_n}}}}}}. \quad (1)$$

The result can be proved by induction on n, which also implies that we can apply this result in the limit. More importantly, this also means that if the sum of products represents a convergent and infinite series, then the right-hand side represents a *convergent and infinite continued fraction*. This speaks of the power of this representation (e.g. examples of how it can represent a large variety of well-known mathematical special functions can be found in [4]).

Based on these results, the authors of [1] proposed that an unknown "target function" of a multivariate regression problem can well approximated by a multivariate function $f : \mathbb{R}^n \to \mathbb{R}$ of the form:

$$f(\mathbf{x}) = g_0(\mathbf{x}) + \cfrac{h_0(\mathbf{x})}{g_1(\mathbf{x}) + \cfrac{h_1(\mathbf{x})}{g_2(\mathbf{x}) + \cfrac{h_2(\mathbf{x})}{g_3(\mathbf{x}) + \ddots}}} \quad (2)$$

where $g_i(\mathbf{x}) \in \mathbb{R}$ for all integer $i \geq 0$, where each function $f_i : \mathbb{R}^n \to \mathbb{R}$ is associated with a vector $\mathbf{a_i} \in \mathbb{R}^n$ and a constant $\alpha_i \in \mathbb{R}$:

$$g_i(\mathbf{x}) = \mathbf{a_i}^{\mathrm{T}}\mathbf{x} + \alpha_i, \quad (3)$$

and, in addition, each function $h_i : \mathbb{R}^n \to \mathbb{R}$ is associated with a vector $\mathbf{b_i} \in \mathbb{R}^n$ and a constant $\beta_i \in \mathbb{R}$:

$$h_i(\mathbf{x}) = \mathbf{b_i}^{\mathrm{T}}\mathbf{x} + \beta_i. \quad (4)$$

In the multivariate regression analysis of a single target variable, the ultimate goal is to find the mathematical expression of an unknown target function that would fit a dataset

Work supported by UoN, Caltech SURF, Maitland Cancer Appeal and Australian Research Council Discovery Project, DP200102364.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

$S = \{(\mathbf{x^{(i)}}, y^{(i)})\}$, i.e. a set of pairs of an unknown multivariate target function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. As a consequence, in [1] it has been proposed that the problem of identifying the target function reduces to a non-linear optimization problem defined by the use of such analytic continued fractions representation. This requires identification of the subset of the variables of $\mathbf{x}$ which are more relevant together with the associated collection of sets of coefficients $\{\mathbf{a_i}\}$, $\{\mathbf{b_i}\}$, $\{\alpha_i\}$, and $\{\beta_i\}$ that jointly contribute to the best function given a particular metric, e.g. the Mean Squared Error (MSE). It is also expected that forcing the search process to reduce the objective function can be accompanied of other search bias that aims at reducing the number of variables, allowing the method to produce functions that are easier to interpret.

II. The depth of the representation

Having more terms in the right-hand side of (2) would lead to better performance on “fitting” the training data. In [5] an illustrative example is given showing how the Gamma function can be well-approximated with increasing values of what it is called as *Depth* and it is shown for the values of 2, 4 and 6 (see Fig. 3 of [5]). It is well-known that increasing the power of the representation on the training dataset would lead to overfitting and more inferior generalization. Therefore, we need to decide if it either the depth of the representation is fixed, or it is dynamically changed during the search/optimization process.

In [1] the authors opted for having the *Depth* fixed to the value of 3. This means that it is basically of the form given by (2) with $h_i(\mathbf{x}) = 0 \; \forall i \geq 2$ and $g_i(\mathbf{x}) = c_i$, where c_i is any real value independent of x and is nonzero for $i \geq 3$. This said, if when we search using a fixed representation with $Depth = 0$, then our technique corresponds to approximating $g_0(\mathbf{x})$ with a linear function (since (3) is linear according to (4)). In that case, we would be looking for a linear regression model and our technique would aim at just selecting a smaller subset of variables. Thus, our proposed representation naturally includes linear regression models, and it does so for any value of $Depth \geq 0$. It is then of interest to see if larger $Depth$ values can lead to a better representation of mathematical functions.

Readers familiar with the theory of analytic continued fractions will immediately notice that, truncating this way, we are looking at the *convergents* of the continued fraction. The first convergent then corresponds to $Depth = 0$; the second convergent (the one with $Depth = 1$) is defined as the ratio of two polynomials $p_2(\mathbf{x})$ and $q_2(\mathbf{x})$ and it is given by

$$\frac{p_2(\mathbf{x})}{q_2(\mathbf{x})} = \frac{g_0(\mathbf{x})g_1(\mathbf{x}) + h_0(\mathbf{x})}{g_1(\mathbf{x})} \tag{5}$$

and the convergent of $Depth = d$ is then the ratio of two polynomials p_{d+1} and q_{d+1} which can be computed by using Milne and Thomson’s matrix representation of a continued fraction [6].

III. Characteristics of the Memetic Algorithm

Memetic algorithm (MA) is gaining wide acceptance in business analytics and data science; a recent review in [2] showed the growing number of applications in Bioinformatics, Combinatorial Optimization, Machine Learning, Data Analytics. More recently, they have also been used for multivariate regression (see [1], [5]).

A. Normalization of the data

The objective of normalizing data is to bring a diverse range of feature values into a common scale, and it is an important prepossessing step. In this contribution, we applied MinMax normalization on the training portion of the data given to the algorithm (thus differing from our previous MA in [5]). The MinMax normalization is the simplest method which rescales the range of features to scale the range in $[0, 1]$ as:

$$x' = \frac{x - min(x)}{max(x) - min(x)}. \tag{6}$$

B. The Memetic Algorithm of this contribution

In [1] $Depth = 3$ was used for six datasets and $Depth = 4$ was selected for the 94 datasets of the *Penn Machine Learning Benchmark* in [5], so we have also chosen $Depth = 4$ for the experiments presented in this paper.

The memetic algorithm (MA) is as the one presented in [5]. For the analysis of the datasets of [1], we have used 10-fold cross-validation as the number of samples is significantly higher. While k-fold Cross Validation requires less times than Leave-One-Out Cross-Validation (LOOCV), in this case, we have a dataset from 352 studies, and most of them have less than 50 samples (the largest one only has 109 samples). It is then reasonable to adopt LOOCV for the whole dataset and be consistent. However, we have used the same data normalization as described in Sec. III-A.

A high-level overview of the proposed MA is shown in Algorithm 1 (due to the page limits, we refer to [5] for a detailed description and illustrative figures). Briefly, we apply MinMax normalization (as shown in (6)) on the training data (Line 1) inside the algorithm and save the scaling factors scl (in Line 2) which comprised with Max and min values of the features and target of the training dataset. The algorithm randomly initialized a population of *agents* which is structured as a ternary-tree (as in [1], [5]). The population consists of 13 such agents. Each agent of the population contains two models (i.e. two analytic continued fractions of $Depth = 4$), one is called ‘pocket’, and another is denoted as ‘current’. If at any stage the guiding function score of the ‘current’ became better than the guiding function score of ‘pocket’, we swap them.

The MA evolves continued fractions (models) for a fixed number of generations. In each generation, we mutate each current model in the population (Line 6). Then we generate a new population via a recombination operation (Line 7). Afterwards, we apply local search optimization to optimize the coefficients of a model as in [5]. This is based on a modified version of Nelder-Mead’s algorithm as proposed by Fajfar *et al.* [7]. We restart the population if the guiding function value stagnates for 5 consecutive generations by replacing the root solution with a model generated at random (Line 11–13). We replace the old population with this population of optimized models (Line 14). We keep track of the best solutions in the population (Line 15–17). When the evolution on the training set satisfied the terminating condition (Line 5), we return from the method (Line 19). As the output of the algorithm, it returns the best solution $best$ (based on the guiding function value) and the scaling factors scl.

We finally have the best model form the training data, and it is the time to apply it on the testing set. Before we apply the model, we scale the testing set applying the scaling factors scl calculated using the training set (when normalizing), and we compute the prediction error (MSE score) on the test set as the generalization score. The average value of the generalization score obtained form the LOOCV is reported in the paper. Since we are using normalization (n) and LOOCV (l) in the proposed Continued Fraction Regression (cfr), we will denote this algorithm as `cfr-nl` throughout the paper.

Algorithm 1: cfr-nl Algorithm

```
Input  : Num. of Vars $nVars$, Mutation Rate $\mu_r$,
         training data $D_{trn}$
Output: Best solution to fit the problem, $best$, Scaling
         Factors $scl$
   /* Normalize training data                  */
1  $D^{n}_{trn} \leftarrow$ Normalize ($D_{trn}$)
2  $scl < Max, min > \leftarrow$ Normalize.ScaleFactor($D_{trn}$)
   /* Generate Initial Population              */
3  $pop \leftarrow$ InitPopulation($nVars, D^{n}_{trn}$)
4  $best \leftarrow pocket(pop.root)$
5  for $gen \in 1, \cdots, numGen$ do
       /* apply variation operations           */
6      $pop_\mu \leftarrow$ Mutate($pop, \mu_r$)
7      $pop_r \leftarrow$ Recombine($pop_\mu$)
       /* Local Search Optimization            */
8      foreach $agent \in pop_r$ do
9          LocalSearch($agent$)
10     end
11     if guiding function stagnates for consecutive 5
         gens then
12         reset($pop_r.root$)
13     end
       /* Replace old with evolved pop         */
14     $pop \leftarrow pop_r$
       /* Keep track of best solution          */
15     if $guid\_func\_score(best) <$
         $guid\_func\_score(pocket(pop.root))$ then
16         $best \leftarrow pocket(pop.root)$
17     end
18 end
19 return $best$, $scl$
```

The parameter values of the `cfr-nl` used for the experiments are presented in Table I.

TABLE I
PARAMETER VALUE OF THE CFR-BASED MEMETIC ALGORITHM FOR REGRESSION.

Parameter	Value
Scale of Penalty (Δ)	0.10
Mutation Rate (μ)	0.10
Fraction's Depth	4
Population Size (tree-based)	13
Number of Generations	200
Reset root of population after stuck for generations	5
Feature Scaling/Normalization Method	MinMax
Number of Nelder-Mead Instances	4
Number of Iterations in Nelder-Mead	250
Nelder-Mead terminates if stagnates for consecutive iterations	10
Percentage of Samples used to evaluate model in local search	20%

IV. MACHINE LEARNING-BASED ALGORITHMS

In [5], [8], a set of 10 state-of-the-art Machine Learning (ML) based regression approaches have been used for the task of benchmarking the performance of regression methods. We have opted to use those same 10 algorithms (implemented in [9] as Scikit-learn API [10]) to compare the performance of our proposed approach, labelled `cfr-nl` method. We refer the reader to [8] for a detailed description of those ML algorithms; however, we are summarizing them with references in Table II to enhance the readability of this contribution.

TABLE II
LIST OF MACHINE LEARNING-BASED ALGORITHMS USED FOR COMPARISON, WITH DEFAULT PARAMETER VALUES FROM SCIKIT-LEARN LIBRARY.

Algorithm	Name	Ref.
Adaptive Boosting (Adaboost) Regression	`ada-b`	[11]
Gradient Boosting (Gradboost) Regression	`grad-b`	[12]
Kernel Ridge	`krnl-r`	[13]
Least-Angle Regression with Lasso (lasso-lars)	`lasso-l`	[14]
Linear Regression	`l-regr`	[10]
Linear Support Vector Regression (SVR)	`l-svr`	[15]
Multilayer Perceptrons (MLPs) Regressor	`mlp`	[16]
Random Forests Regression	`rf`	[17]
Stochastic Gradient Descent (SGD) Regression	`sgd-r`	[18]
Extreme Gradient Boosting (xgboost)	`xg-b`	[19]

V. DESCRIPTION OF SCHAEFFER'S 352 DATASETS

Cullen Schaffer collected 352 bivariate numeric datasets during his PhD Thesis work, mostly from investigations in the physical sciences, and he proposed a function-finding algorithm in [20]. The dataset[1] is now on the public domain and downloadable from the UCI-Machine Learning Repository [21]. From the accompanying description, we highlight that the datasets are organized into 217 “cases”, generally from a single source (publication or article). The first 62 cases are *“willfully chosen as useful, notable or interesting from a wide variety of sources including handbooks, theses, journal articles, textbooks, student laboratory reports and others”*. For instance, ‘Case 55’ is extracted from one of Robert Andrew Millikan's oil drop experiments reported in 1911 [22], part

[1] Function Finding Data Set at http://archive.ics.uci.edu/ml/datasets/Function+Finding

of the experimental work that leads him to obtain the Nobel Prize in 1923. The remaining ones are obtained from *Physical Review* from issues of the early years of the 20th century. They pose a challenge of finding functions fitting the data. Schaffer imposed these conditions for inclusion in his collection: First, the source reported a governing functional relationship, and it was bivariate, and second, the data were reported in tabular form and should be the result of a measurement (as opposed to just being theoretically postulated). Schaffer, designed his E* function-finding algorithm [20] on Cases 1 though 122 and then tested it on Cases 123 through 222. In that use we will differ as it is explained in Sec. VI.

In [1], six datasets were used. We have chosen to present also results on them and to compare with 10 ML methods on six datasets. In this case, the average MSE score obtained for 10-fold Cross-Validation on the normalized data in Table III. We advise, however, that these scores are computed on a normalized dataset, as a consequence the MSE scores reported in [1] are not directly comparable to the ones in this paper. However, this study, using the same $Depth$ value allows us to look at a different type of dataset and on which we have previous computational experience.

VI. Experimental Results

We employed The University of Newcastle's High-Performance Computing (HPC) grid[2]. The grid machines are configured with the GCC compiler version 4.8.5 and run using 64 bit Red Hat Enterprise Linux Server 7.5 (Maipo). C++11 is used to develop the algorithm and compiled with -O2 optimization. We split the execution of the algorithm on 352 datasets into 7 batches. For each batch, we assigned 4 CPU cores, 4GB RAM, and a total of 10 hours of CPU wall time to execute the Leave-One-Out Cross-Validation (LOOCV) on the datasets. CPU time has not been a concern of this particular study; we refer to [5] for comparative time performances of our algorithm against existing symbolic regression approaches based on genetic programming.

A. Results on the six datasets (from CEC 2019, previous contribution [1])

Table III resumes the results. In all cases the result obtained by `cfr-nl` is consistently very close to the performance of the best one of the 10 algorithms studied (in bold). We will return to these results in the Discussion section.

B. Results of cfr-nl on the Datasets of 352 Functions

We now present the average Mean Squared Error (MSE) obtained by the models of our purposed `cfr-nl` algorithm in Table IV for Leave-One-Out Cross-Validation (LOOCV) of 352 functions. We ordered the results from smallest to largest of the MSE score obtained by our method. We choose to include the tabulated scores as they can be useful in the future for other researchers willing to compare the performances of other methods on the same dataset.

TABLE III
Average MSE Score obtained by cfr-nl and 10 machine learning algorithms for 10 fold cross validation and Normalization on six datasets used in [1]

Methods	airfoil	concrete	cooling	heating	housing	yacht
`ada-b`	0.0106	0.0089	0.0043	0.0025	0.0049	0.0006
`grad-b`	0.0051	**0.0040**	0.0016	**0.0002**	**0.0039**	**0.0001**
`krnl-r`	0.0865	0.0170	0.0078	0.0066	0.0098	0.0216
`lasso-l`	0.0340	0.0436	0.0660	0.0742	0.0323	0.0598
`l-regr`	0.0166	0.0170	0.0075	0.0063	0.0078	0.0214
`l-svr`	0.0172	0.0185	0.0080	0.0066	0.0081	0.0314
`mlp`	0.0133	0.0112	0.0075	0.0058	0.0072	0.0209
`rf`	**0.0023**	0.0043	**0.0021**	**0.0002**	0.0045	0.0003
`sgd-r`	0.0329	0.0304	0.0107	0.0099	0.0186	0.0416
`xg-b`	0.0051	0.0041	0.0017	**0.0002**	**0.0039**	0.0002
`cfr-nl`	0.0119	0.0100	0.0064	0.0047	0.0080	0.0006

C. Performance Ranking on Schaeffer's 352 datasets

We show the box plots in Fig. 1 to illustrate the performances of the algorithms for training (in Fig. 1(a)) and testing (in Fig. 1(b)) sets. The MSE score in the y-axis are log_{10}-scaled (we displayed the points up to $1e$+02) and the boxes are ordered from smallest to largest of median score (50th percentile). The reason that we are displaying the results only on that interval is that all algorithms, including our own, had a poor generalization performance for a single dataset ('Case_28') corresponding to a dataset with only six samples for which a functional model of the form $\log y = k_1 \log x + k_2$ was proposed in [23].

From the Fig. 1 we can see that the proposed `cfr-nl` achieved the best result (considering the median or 50th percentile score) in the comparison against the 10 machine learning algorithms for both in training and testing sets.

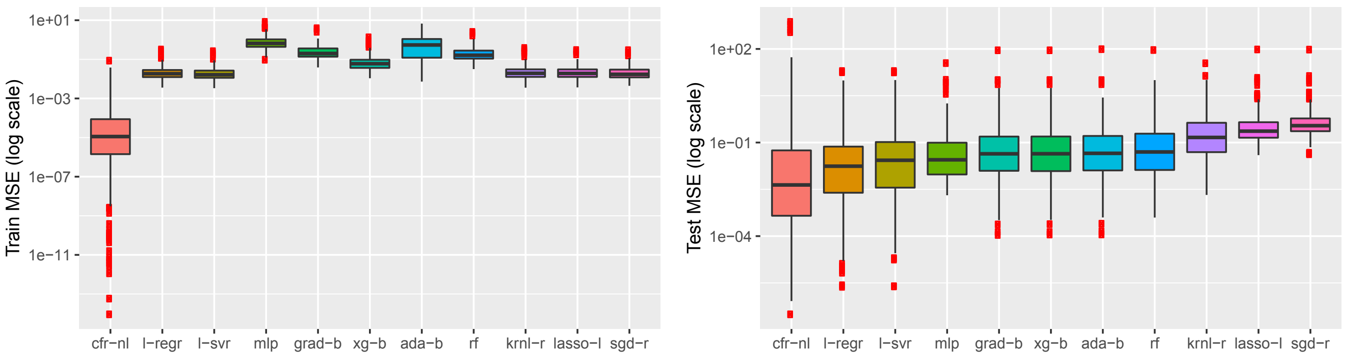
We computed the ranking of the algorithms for their performances for each of the 352 datasets. The violin plots in Fig. 2 shows the *box-and-whisker plots* with the quantitative distribution of the ranking results. We have shown the training performances of the algorithms in Fig. 2(a). The proposed `cfr-nl` achieves impressive rankings in training sets (most of the time being the best-ranked method). In training, the closest performing ML method to `cfr-nl` is the Linear Support Vector Regressor (`l-svr`). The testing performances of the methods are shown in Fig. 2(b). We can observe that `cfr-nl` and `l-regr` are the two best algorithms in terms of achieving better rankings in testing results. The performance of `l-regr` is quite surprising for us, due to the simplicity of the method, but it may be a part of the characteristics of these 352 datasets. In fact, for these regression problems, the full potential of `cfr-nl` in variable selection is not significantly showing, since the problems do not have many variables, but its representation power is present. The two worst-performing algorithms were `lasso-l` and `sgd-r`.

Table V complements the presentation of the violin plots of Fig. 2 with a listing of the number of times an algorithm ranked first (**1st**) and last (**L**) in the training and testing sets of the 352 datasets. From the table we observe that `cfr-nl`

[2] https://www.newcastle.edu.au/research-and-innovation/resources/research-computing-services/advanced-computing

TABLE IV
THE AVERAGE TEST MEAN SQUARED ERROR (MSE) SCORE OBTAINED BY THE PROPOSED ALGORITHM WITH LEAVE-ONE-OUT CROSS-VALIDATION (LOOCV) ON 352 FUNCTIONS (SORTED BY ASCENDING MSE SCORES).

Function	MSE	Function	MSE	Function	MSE	Function	MSE	Function	MSE	Function	MSE
Case_161b	5.24e-7	Case_61b	1.71e-4	Case_13	8.68e-4	Case_112b	4.10e-3	Case_130a	0.023	Case_181d	0.181
Case_192	9.44e-7	Case_106b	1.76e-4	Case_163	9.10e-4	Case_71a	4.20e-3	Case_124	0.024	Case_159b	0.183
Case_161a	1.13e-6	Case_43c	1.77e-4	Case_76c	9.21e-4	Case_108c	4.26e-3	Case_70a	0.025	Case_123	0.188
Case_82b	1.26e-6	Case_30a	1.82e-4	Case_101b	9.57e-4	Case_81a	4.26e-3	Case_188	0.025	Case_63	0.196
Case_170a	2.15e-6	Case_36	1.82e-4	Case_76a	1.02e-3	Case_31	4.30e-3	Case_185	0.026	Case_168	0.221
Case_82a	5.42e-6	Case_148a	1.91e-4	Case_164	1.05e-3	Case_173a	4.41e-3	Case_197d	0.028	Case_209a	0.246
Case_88	6.20e-6	Case_128a	1.92e-4	Case_78	1.06e-3	Case_104a	4.54e-3	Case_41b	0.029	Case_112a	0.254
Case_218c	8.78e-6	Case_100a	1.96e-4	Case_72	1.06e-3	Case_106a	4.61e-3	Case_212a	0.029	Case_147	0.257
Case_142	9.86e-6	Case_153d	2.06e-4	Case_116	1.07e-3	Case_181c	4.65e-3	Case_98c	0.030	Case_204d	0.261
Case_128c	9.89e-6	Case_77	2.16e-4	Case_184	1.13e-3	Case_52	4.78e-3	Case_80a	0.030	Case_53	0.271
Case_218a	1.05e-5	Case_136b	2.19e-4	Case_134a	1.18e-3	Case_186b	4.86e-3	Case_199a	0.031	Case_138a	0.346
Case_113c	1.07e-5	Case_197c	2.31e-4	Case_129a	1.23e-3	Case_43b	4.86e-3	Case_183c	0.032	Case_181a	0.363
Case_160	1.08e-5	Case_127c	2.54e-4	Case_221	1.24e-3	Case_193	5.29e-3	Case_122c	0.032	Case_158b	0.387
Case_110	1.22e-5	Case_54	2.75e-4	Case_187c	1.24e-3	Case_198	5.37e-3	Case_137	0.032	Case_135a	0.388
Case_145a	1.29e-5	Case_27	2.82e-4	Case_81b	1.26e-3	Case_16	5.71e-3	Case_14	0.033	Case_125b	0.394
Case_208	1.36e-5	Case_204a	2.87e-4	Case_93	1.28e-3	Case_183a	6.33e-3	Case_89	0.033	Case_131a	0.406
Case_218b	1.90e-5	Case_73	3.02e-4	Case_167	1.29e-3	Case_214a	6.37e-3	Case_182d	0.033	Case_132	0.425
Case_166b	1.94e-5	Case_219	3.14e-4	Case_38	1.30e-3	Case_37a	6.68e-3	Case_96a	0.033	Case_214d	0.441
Case_111b	2.13e-5	Case_10	3.20e-4	Case_200c	1.30e-3	Case_35	6.85e-3	Case_175	0.033	Case_211	0.496
Case_113a	2.85e-5	Case_84d	3.24e-4	Case_44	1.36e-3	Case_206a	6.96e-3	Case_62b	0.034	Case_56b	0.531
Case_140	2.96e-5	Case_190b	3.49e-4	Case_99a	1.44e-3	Case_6	7.19e-3	Case_71b	0.038	Case_41a	0.603
Case_148b	3.18e-5	Case_115	3.51e-4	Case_8	1.46e-3	Case_172	7.23e-3	Case_21	0.040	Case_216	0.676
Case_113b	3.44e-5	Case_205	3.68e-4	Case_30b	1.51e-3	Case_18	7.26e-3	Case_62a	0.041	Case_91b	0.772
Case_148d	3.48e-5	Case_59	3.77e-4	Case_68	1.72e-3	Case_214c	7.96e-3	Case_128d	0.041	Case_204b	0.853
Case_100c	3.80e-5	Case_171c	3.77e-4	Case_178c	1.73e-3	Case_134b	8.00e-3	Case_209c	0.042	Case_47	0.853
Case_11	4.17e-5	Case_126b	3.84e-4	Case_86	1.86e-3	Case_22	8.27e-3	Case_91a	0.049	Case_122a	0.857
Case_45	4.22e-5	Case_58	3.91e-4	Case_217	1.90e-3	Case_39	8.91e-3	Case_41c	0.050	Case_151	0.992
Case_100d	4.41e-5	Case_104b	3.94e-4	Case_180	1.92e-3	Case_196b	9.11e-3	Case_214b	0.054	Case_206c	0.999
Case_102	4.43e-5	Case_114	4.05e-4	Case_80b	1.94e-3	Case_96b	9.52e-3	Case_182b	0.059	Case_135d	1.156
Case_48	4.81e-5	Case_7	4.14e-4	Case_130b	1.96e-3	Case_187a	9.55e-3	Case_153c	0.062	Case_79	1.179
Case_87	4.89e-5	Case_165	4.24e-4	Case_187d	1.98e-3	Case_152c	9.64e-3	Case_152b	0.063	Case_181b	1.199
Case_46a	4.90e-5	Case_174	4.41e-4	Case_125a	1.99e-3	Case_144b	9.85e-3	Case_149	0.066	Case_135c	1.213
Case_1	5.05e-5	Case_9	4.46e-4	Case_210	2.03e-3	Case_127b	0.011	Case_176b	0.067	Case_159a	1.319
Case_173b	5.13e-5	Case_84b	5.01e-4	Case_117	2.05e-3	Case_201	0.012	Case_55	0.068	Case_121	1.483
Case_203	5.19e-5	Case_136d	5.17e-4	Case_191	2.08e-3	Case_83	0.012	Case_143	0.069	Case_66	1.825
Case_152a	5.56e-5	Case_186c	5.18e-4	Case_153b	2.11e-3	Case_200d	0.012	Case_194b	0.074	Case_109	1.982
Case_92	5.74e-5	Case_108b	5.23e-4	Case_171a	2.18e-3	Case_154	0.012	Case_196d	0.077	Case_215	2.873
Case_170b	6.64e-5	Case_50	5.28e-4	Case_108a	2.20e-3	Case_15	0.012	Case_196a	0.081	Case_103	3.955
Case_213a	6.87e-5	Case_162	5.31e-4	Case_194a	2.30e-3	Case_119	0.012	Case_122d	0.083	Case_156	4.356
Case_129c	7.28e-5	Case_20	5.72e-4	Case_64	2.55e-3	Case_200b	0.012	Case_204c	0.099	Case_25	4.807
Case_100b	7.38e-5	Case_190a	5.80e-4	Case_189	2.55e-3	Case_171b	0.013	Case_69	0.105	Case_178a	4.901
Case_70b	7.58e-5	Case_46b	6.01e-4	Case_97	2.60e-3	Case_84c	0.013	Case_131b	0.105	Case_37b	5.145
Case_138b	7.60e-5	Case_99b	6.05e-4	Case_212c	2.77e-3	Case_179b	0.014	Case_207	0.106	Case_182c	5.693
Case_145b	8.02e-5	Case_150	6.30e-4	Case_169	2.80e-3	Case_56a	0.015	Case_176a	0.110	Case_182a	6.438
Case_5	8.60e-5	Case_220	6.45e-4	Case_196c	2.83e-3	Case_190c	0.015	Case_178b	0.122	Case_202	6.830
Case_218d	9.44e-5	Case_128b	6.51e-4	Case_129b	2.84e-3	Case_98d	0.016	Case_84a	0.124	Case_194c	6.852
Case_111a	9.83e-5	Case_120	6.60e-4	Case_40	2.91e-3	Case_197b	0.016	Case_195b	0.125	Case_183b	7.026
Case_133	1.04e-4	Case_60	6.64e-4	Case_194d	2.96e-3	Case_130c	0.017	Case_122b	0.126	Case_90	7.247
Case_85	1.05e-4	Case_118	6.87e-4	Case_95	3.12e-3	Case_98b	0.017	Case_139	0.133	Case_146	8.899
Case_127a	1.09e-4	Case_200a	6.96e-4	Case_195a	3.12e-3	Case_144a	0.017	Case_197a	0.135	Case_2	14.417
Case_106c	1.20e-4	Case_43a	7.11e-4	Case_4	3.12e-3	Case_98a	0.018	Case_131c	0.145	Case_158a	14.571
Case_166a	1.24e-4	Case_187b	7.15e-4	Case_179a	3.13e-3	Case_67	0.018	Case_153a	0.147	Case_49	15.823
Case_136c	1.24e-4	Case_19	7.23e-4	Case_111c	3.21e-3	Case_51	0.020	Case_17	0.150	Case_158c	19.139
Case_136a	1.40e-4	Case_108d	7.29e-4	Case_56c	3.43e-3	Case_209b	0.020	Case_222	0.154	Case_157	19.673
Case_105	1.41e-4	Case_177	7.45e-4	Case_141	3.61e-3	Case_42	0.020	Case_41d	0.156	Case_135b	47.605
Case_126a	1.46e-4	Case_155	7.98e-4	Case_206b	3.62e-3	Case_107	0.021	Case_76b	0.170	Case_12	52.566
Case_74	1.53e-4	Case_186a	8.21e-4	Case_206d	3.66e-3	Case_186d	0.022	Case_75	0.175	Case_28	4.17e+8
Case_148c	1.65e-4	Case_61a	8.36e-4	Case_199b	3.72e-3	Case_24	0.023	Case_23	0.177		
Case_101a	1.66e-4	Case_213b	8.50e-4	Case_212b	3.84e-3	Case_94	0.023	Case_209d	0.178		



(a) Training Performance

(b) Testing Performance (cropped y-axis values $>$ 1e+02)

Fig. 1. Box plot showing the average MSE score (in log_{10} scale) by 10 machine learning algorithms and the proposed method (`cfr-nl`) for LOOCV of 352 Functions in a) Training Sets and b) Testing Sets. To better visualize the testing performance in (b) we have limited the y-axis to the maximum 1e+02 which removes a single case (a dataset with only six samples) in which all methods have terrible bad result due to the existence of a problematic outlier [23] way out of the normal range.

TABLE V
NUMBER OF TIMES THE ALGORITHM RANKED 1^{ST} AND LAST (L) FOR THE PERFORMANCES ON 352 DATASET FOR TRAINING (t.) AND TESTING (T.) WITH LOOCV

Alg.	t.**1st**	t.**L**	**T.1st**	**T.L**	**Alg.**	t.**1st**	t.**L**	**T.1st**	**T.L**
cfr-nl	**350**	0	**192**	25	rf	0	0	4	1
l-regr	0	0	97	4	krnl-r	0	0	3	109
l-svr	2	0	22	2	xg-b	0	0	3	1
mlp	0	**224**	15	5	lasso-l	0	0	0	9
ada-b	0	**128**	10	6	sgd-r	0	0	0	**196**
grad-b	0	0	6	1					

not only achieves impressive training (t.) performance (350 times ranked 1^{st}) but also achieved the 1^{st} rank 192 times (i.e. 54.5 percent of the time) for the testing (**T.**) performances. It has ranked last only for 25 testing cases (7.1 percent) and never ranked last for training. The closest performing algorithm of `cfr-nl` is the well-established linear regression (`l-regr`) method which has become 1^{st} 97 times in testing but never become 1^{st} in training. It has ranked last in only 4 cases in testing. Multilayer Perceptrons Regressor (`mlp`) and Adaptive Boosting Regression (`ada-b`) are two of the worst-performing algorithms based on the ranking of training performances. On the other hand, Least-Angle Regression with Lasso (`lasso-l`) and Stochastic Gradient Descent (`sgd-r`) regression are the only algorithms that have never be able to rank 1^{st} for any testing case. Moreover, `sgd-r` has ranked last in 196 times for testing sets, which is the worst performance among all algorithms.

D. Statistical Significance Test

We applied a modification of the Friedman test [24] by Iman and Davenport [25] to test whether there exist any significant differences in the median rankings of the algorithms based on their MSE scores. The statistical test found p-value $< 2.2e$-16 (with Corrected Friedman's chi-squared = 68.089, $df1 = 15$, $df2 = 1395$) which indicates that the null hypothesis *“all the algorithms perform the same”* can safely be *“rejected”*. Therefore, we proceeded with the post-hoc test.

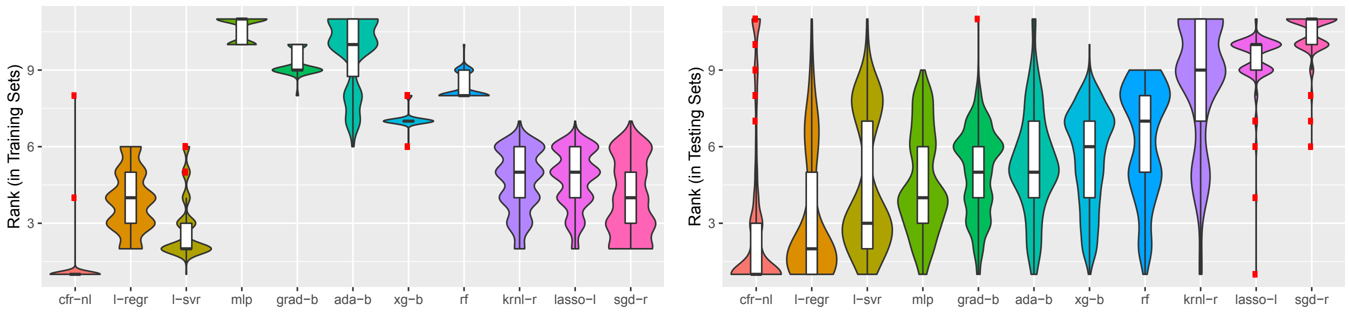
For post-hoc test, we have applied Freidman's post-hoc test on the median rankings of the algorithms based on test MSE scores. Obtained p-values are plotted in the heatmap shown in Fig. 3. In the heatmap, the dendogram showed the hierarchical grouping of the algorithms according to their similarity of p-values. From the heatmap, it is noticeable that there exist ‘*no significant differences*’ (see the Color Key for the corresponding p-value) in performances of the pairs: {`cfr-nl`, `l-regr`}, {`l-svr`, `mlp`}, {`xg-b`, `ada-b`} and {`grad-b`, `ada-b`}.

In addition to the post-hoc results illustrated with heatmap, the Critical Difference (CD) diagram proposed in [26] can also be used to visualize the differences among algorithms. The CD plot is based on the Nyemeni post-hoc test and might have slightly different results from the pairwise Friedman test we had done in the post-hoc test. It places the algorithm on the x-axis of their median ranking of performances and computes the *critical difference* of rankings between the algorithms. If the ranking difference of any pair of algorithms is greater than the critical difference, they are regarded as ‘significantly different’. Finally, the statistically ‘non-significant’ algorithms are connected in a group with horizontal lines.

We plot the CD graph (in Fig. 4) using the implementation of [27] for our experiments with a significance threshold of $p = 0.05$. The statistical *non significant different* algorithms are *connected* by the horizontal line in the plot. We can see that *no significant differences* exist in the performance rankings of {`cfr-nl`, `l-regr`}, {`l-svr`, `mlp`, `grad-b`, `ada-b`} and {`grad-b`, `ada-b`, `xg-b`} sets of algorithms on 352 datasets. Moreover, the median ranking of `cfr-nl` is less than three (3), which is the best result among all of the algorithms.

E. Discussion

After analyzing the median performance ranking of 10 state-of-the-art machine learning-based regression algorithms (on a



(a) Training Rank

(b) Testing Rank

Fig. 2. Violin plot showing the observed ranking (computed on MSE Scores in LOOCV) of all algorithms on 352 functions in the ascending order of their median ranking on testing data in a) Training Sets and b) Testing Sets.

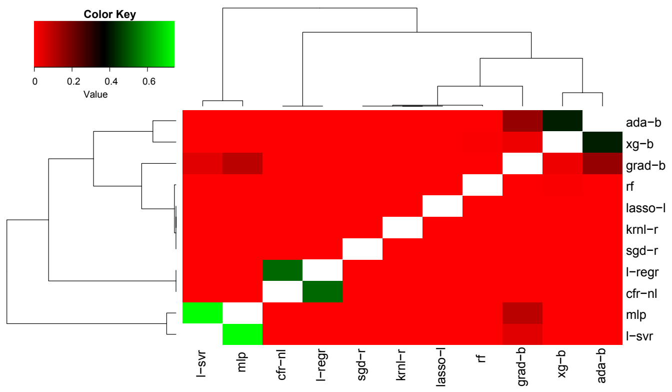


Fig. 3. Heatmap showing the levels of p-values obtained by the Post-hoc pairwise Test using Freidman's Aligned Rank Test for the performances of the algorithms.

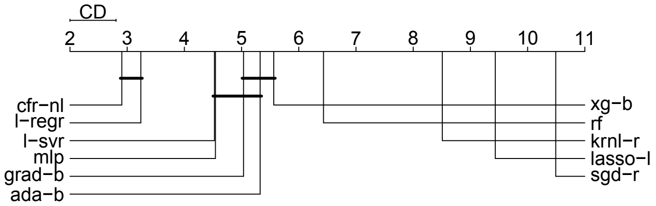


Fig. 4. The critical difference (CD) plot show the statistical significance of rankings for achieved MSE scores on 352 datasets. Groups of algorithms that are 'not significantly different' (at $p = 0.05$) are 'connected' with a horizontal line.

total of 358 datasets), our proposed approach exhibited better performances while compared with most of the approaches, particularly in the 352 datasets collected by Schaffer. We note that for these, an important fact distinguishes them, i.e. "a governing functional relationship was reported". The significance of the performance of `cfr−nl` is also supported by statistical testing. Nevertheless, surprisingly, in this collection of datasets, it has become closer and competitive to `l−regr`. This is in contrast with what we observed in [5], where the only statistically similar algorithm (in terms of generalization performance on the *Penn Machine Learning Benchmark Database*) to our approach based on analytic continued fractions was Extreme Gradient Boosting (`xg−b`). In this case, `xg−b` has a relatively lower performance since not only `cfr−nl` and but also `l−regr` have statistically better performances. Moreover, in this case `xg−b` is not statistically different to Support Vector Regression (`l−svr`), Multilayer Perceptrons Regressor (`mlp`), Gradient Boosting Regression (`grad−b`), and Adaptive Boosting Regression (`ada−b`).

In terms of the generalization performance (shown in Fig.1(b)), the proposed method outperformed all of the state-of-the-art machine learning algorithms for the median MSE scores (50th percentile). It maintains the same lead in terms of the rankings of the algorithms shown in Fig. 2. Furthermore, the method outperformed all algorithms for the number of times an algorithm was ranked 1st among the 11 methods. The results of these exploratory analyses, together with the lead is shown by our approach in a different dataset in [5], collectively advocate for the superiority of `cfr−nl` over all of the state-of-the-art machine learning-based regression methods in terms of its generalization performance.

We note that for this study, we conducted two types of statistical tests. The post-hoc test (illustrated in Fig. 3) revealed that there exists 'no significant difference' in performances between the leading methods, our `cfr−nl` and linear regression (`l−regr`), on this 352 datasets. In the critical difference plot (shown in Fig. 4), `cfr−nl` has shown the best ranking, just with 'no significant difference' against the ranking of `l−regr`. The statistical tests confirmed that `cfr−nl` is a better approach than most (9 out of 10) of the ML methods and at least comparable with the linear regression in predicting functions form real-world physics problems. We note that when using the *Penn Machine Learning Database*, `l−regr` did not perform as well, being the second last performing method. This also speaks of the robustness obtained by using analytic continued fractions and within a memetic algorithm approach that also performs variable selection.

VII. Conclusions and Future Work

In the future we would like to explore the possibility of introducing new variables which may be linked via simple arithmetic forms to the existing ones and let the memetic algorithm search for the best combinations to build models. We know, for instance, that in some cases it may lead to faster convergence of the analytic continued fraction. As an illustrative example, we know that $x\ tan(x)$ can be approximated as follows [28]:

$$x\ tan(x) = \cfrac{x^2}{1 - \cfrac{x^2}{3 - \cfrac{x^2}{5 - \cfrac{x^2}{7 - \ddots}}}} \tag{7}$$

so it seems reasonable to extend the methodology to include the possibility that given a dataset $S = \{(\mathbf{x}^{(\mathbf{i})}, y^{(i)})\}$, we can explore the combination of the variables. This is very interesting as the pioneering algorithm E* proposed by C. Schaffer [20], the curator of the dataset are we now using, is based in considering only the linear relationships $y = k_1x+k_2$, six types of "power proportionalities" (i.e. of the form $y = x^n$ for $n \in \{-2, -1, -.5, .5, 1, 2\}$), as well as the null answer *"No relationship identified"*. We aim then to explore if the introduction of combination of variables and the use of "power proportionalities" can help to extend the set of original variables and perhaps lead to faster convergent analytic continued fractions with better results in generalization.

The good performance of boosting based algorithms (e.g. grad-b, ada-b, and xb-g) in this study in Table III, together with the relatively good result of xb-b and grad-b in [5] motivates us to consider including boosting within our methodological framework. In conclusion, putting together the performance of our memetic algorithms that use analytic continued fractions, compared with many current algorithms for multivariate regression, indicate that this approach deserves further investigation.

Acknowledgment

We thank Markus Wagner for his thoughtful comments on an earlier version of the manuscript.

References

[1] H. Sun and P. Moscato, "A memetic algorithm for symbolic regression," in *IEEE Congress on Evolutionary Computation, CEC 2019, Wellington, New Zealand, June 10-13, 2019*. IEEE, 2019, pp. 2167–2174.

[2] P. Moscato and L. Mathieson, "Memetic algorithms for business analytics and data science: A brief survey," in *Business and Consumer Analytics: New Ideas*, P. Moscato and N. J. de Vries, Eds. Springer, 2019, pp. 545–608.

[3] L. Euler, *Introductio in analysin infinitorum*, 1748, vol. 1, ch. 18, Reprinted as Opera (1)8.

[4] F. Backeljauw and A. Cuyt, "Algorithm 895: A continued fractions package for special functions," *ACM Trans. Math. Softw.*, vol. 36, no. 3, Jul. 2009.

[5] P. Moscato, H. Sun, and M. N. Haque. (2019) Analytic continued fractions for regression: A memetic algorithm approach. arXiv. [Online]. Available: https://arxiv.org/abs/2001.00624

[6] O. Eğecioğlu, Ç. K. Koç, and J. R. Coma, "Fast computation of continued fractions," *Computers & Mathematics with Applications*, vol. 21, no. 2, pp. 167 – 169, 1991.

[7] I. Fajfar, J. Puhan, and A. Bũrmen, "Evolving a Nelder-Mead algorithm for optimization with genetic programming," *Evolutionary computation*, vol. 25, Jan 2016.

[8] P. Orzechowski, W. La Cava, and J. H. Moore, "Where are we now?: A large benchmark study of recent symbolic regression methods," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '18. New York, NY, USA: ACM, 2018, pp. 1183–1190.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python ," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[10] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[11] H. Drucker, "Improving regressors using boosting techniques," in *Proceedings of the Fourteenth International Conference on Machine Learning*, ser. ICML '97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 107–115.

[12] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, pp. 1189–1232, 2000.

[13] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[14] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32, pp. 407–499, 2004.

[15] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, Aug 2004.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.

[19] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794.

[20] C. Schaffer, "A proven domain-independent scientific function-finding algorithm," in *Proceedings of the 8th National Conference on Artificial Intelligence. Boston, Massachusetts, USA, July 29 - August 3, 1990, 2 Volumes*, H. E. Shrobe, T. G. Dietterich, and W. R. Swartout, Eds. AAAI Press / The MIT Press, 1990, pp. 828–833.

[21] D. Dua and C. Graff. (2017) UCI machine learning repository. Website. [Online]. Available: http://archive.ics.uci.edu/ml

[22] R. A. Millikan, "The isolation of an ion, a precision measurement of its charge, and the correction of Stokes's Law," *Phys. Rev. (Series I)*, vol. 32, pp. 349–397, Apr 1911.

[23] G. K. von Hevesy and F. A. Paneth, *A manual of radioactivity*. Oxford University Press, 1926.

[24] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the american statistical association*, vol. 32, no. 200, pp. 675–701, 1937.

[25] R. L. Iman and J. M. Davenport, "Approximations of the critical region of the fbietkan statistic," *Communications in Statistics-Theory and Methods*, vol. 9, no. 6, pp. 571–595, 1980.

[26] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[27] B. Calvo and G. Santafé Rodrigo, "scmamp: Statistical comparison of multiple algorithms in multiple problems," *The R Journal, Vol. 8/1, Aug. 2016*, 2016.

[28] R. E. Crandall, *Projects in scientific computation*. Springer, 1994.