COMPUTER SCIENCE

Generative and reproducible benchmarks or comprehensive evaluation machine learning classifiers

Patryk Orzechowski[1,2]* and Jason H. Moore[3]*

Understanding the strengths and weaknesses of machine learning (ML) algorithms is crucial to determine their scope of application. Here, we introduce the Diverse and Generative ML Benchmark (DIGEN), a collection of synthetic datasets for comprehensive, reproducible, and interpretable benchmarking of ML algorithms for classification of binary outcomes. The DIGEN resource consists of 40 mathematical functions that map continuous features to binary targets for creating synthetic datasets. These 40 functions were found using a heuristic algorithm designed to maximize the diversity of performance among multiple popular ML algorithms, thus providing a useful test suite for evaluating and comparing new methods. Access to the generative functions facilitates understanding of why a method performs poorly compared to other algorithms, thus providing ideas for improvement.

INTRODUCTION

The development of new machine learning (ML) algorithms has accelerated to meet the demands of a variety of big data applications. An important type of ML algorithm is the classifier that is designed to accept discrete and/or continuous input features and produce a binary prediction or outcome that matches as close as possible a binary target such as the presence or absence of disease or success or failure of a device. This class of algorithms, sometimes referred to as supervised ML, is useful in many domains and is often used to complement parametric statistical methods such as logistic regression. Examples include tree-based approaches such as decision trees (*1*–*3*), random forests (*4*, *5*), kernel-based methods such as support vector machines (*6*), and gradient boosted trees (*7*), as well as and their many variants (*8*, *9*).

Central to the development of ML algorithms is their evaluation. A good evaluation should document the strengths and weaknesses of the method and allow a fair and robust comparison to other state-of-the-art methods. The evaluation criteria often include measures of the accuracy of the predictions made, the computational efficiency of the algorithms, the degree of fairness and bias, and the user's ability to interpret the results from a fitted model. The evaluation results help practitioners understand when it is appropriate to use the method, help readers assess the trustworthiness of the results, and help developers generate ideas for how to make improvements. One approach to evaluation is to use real and/or simulated datasets as "benchmarks." Two commonly used collections of real and synthetic data are the University of California Irvine (UCI) ML repository (*10*) and the Library for Support Vector Machines (LIBSVM) (*11*), which provide hundreds of real datasets with open access. Public efforts focused on reorganizing, standardizing, and expanding those repositories lead to the emergence of benchmarking repositories, such as the Penn Machine Learning Benchmark (*12*), Open Machine Learning 100 (OpenML100), and its curated successor OpenML-CC18 (*13*). A lot of effort has been made to organize benchmarks for regression problems (*12*, *14*–*16*). Benchmarks have also become more popular by competitions such as Kaggle (*17*) that provide data for the comparison of algorithms provided by contestants. The results of such competitions have helped the ML community evaluate and improve numerous algorithms.

Increased data accessibility allowed extensive testing of different ML algorithms. A common benchmarking practice is to select a subset of these datasets to illustrate one method performing better than others. There are several problems with this approach. It is rare to know what the true patterns are in real data. Hence, it is difficult know whether an algorithm is performing well because it is modeling the truth or exploiting the noise in the data (i.e., overfitting). This issue can be addressed by the inclusion of replicate datasets drawn from the same experiment or observational study. Ideally, at least three datasets of sufficient sample size and consistent structure would be available for training a model (dataset one), tuning the parameters of the model (dataset two), and validation (dataset three). Additional validation datasets provide additional confidence that the model is generalizable and thus likely to be modeling the signal in the data. However, real data are time-consuming and expensive, and thus, multiple real datasets are rarely available. Even when multiple datasets are available, it is often difficult to know whether the examples are representative of the population that they were drawn from or whether the features were measured in the same way. Furthermore, the release of real data to the public can be problematic because of privacy, intellectual property, or confidentiality issues. These limitations have led some to turn to simulated data.

A major advantage of simulated data is that the ground truth is known because the signal and noise are specifically engineered. This may provide some insights into why a method performs better or worse on certain datasets. Furthermore, a generative function makes it possible for the user to generate as many replicate datasets and with any shape to evaluate the methods. An important limitation of simulated data is that it may not be possible to know whether the patterns being generated are consistent with those from real data. Despite the strengths, there is a noticeable lack of simulated data for benchmarking classification algorithms. Additional limitation is that existing datasets are not focused on differentiating the accuracy of the classifiers. Thus, multiple methods commonly end up having similar performance for multiple available datasets. Those datasets in terms of benchmarking are noninformative and abundant, as

[1]Institute for Biomedical Informatics, University of Pennsylvania, 3700 Hamilton Walk, Philadelphia, PA 19104, USA. [2]Department of Automatics and Robotics, AGH University of Science and Technology, al. Mickiewicza 30, 30-059 Kraków, Poland. [3]Department of Computational Biomedicine, Cedars-Sinai Medical Center, 700 N. San Vicente Blvd., Suite G540, West Hollywood, CA 90069, USA.
*Corresponding author. Email: patryk.orzechowski@gmail.com (P.O.); jason.moore@csmc.edu (J.H.M.)

they do not sufficiently highlight strengths and weaknesses of classification methods.

There are several aspects that a good synthetic benchmark should deliver. A desirable quality of a good synthetic benchmark is the ability to differentiate the performance of multiple methods not only within a dataset but also between datasets, such that no one method would dominate all the others (this is commonly referred as no free lunch theorem in ML). A useful benchmark should also deliver insights regarding types of generative functions that are not suitable for the method being evaluated so that strategies for improving the method can be developed. It should be compact to offer rapid evaluation yet comprehensive to cover varying types of problems. Ideally, it would not only cover small problems but also provide the possibility to scale them to larger problems of any size, with the same built-in ground truth. It would be also great if the benchmark provided precomputed information on expected performance of the methods on replicated datasets initialized with different random seeds. Last, the benchmark should be reproducible so that the results across multiple machines and operating systems remained the same.

With the aforementioned qualities in mind, we introduce the Diverse and Generative ML Benchmark (DIGEN), a synthetic data resource for comprehensive, reproducible, and interpretable benchmarking of ML algorithms for classification of binary outcomes. A central goal is to generate a diverse set of mathematical functions that map continuously distributed features to binary outcomes or class variables for the purpose of revealing the strengths and weaknesses of the ML algorithms being evaluated.

METHODS

The resulting collection of datasets has been created as a result of multistep optimization built on top of a heuristic algorithm that discovers the generative mathematical functions. The datasets are simulated in a specific way that yields maximum diversity of the performance of multiple commonly used ML algorithms. The heuristic algorithm has two optimization objectives: accuracy [measured as the area under receiver operating characteristic curve (AUROC) between two selected methods] and SD between the remaining methods. The candidate functions have emerged as the result of duels between pairs of ML algorithms from the following list: decision trees, gradient boosting, *k*-nearest neighbors (*18*), light gradient boosting (LightGBM) (*9*), logistic regression (*19*), random forests, support vector machines, and extreme gradient boosting (XGBoost) (*8*). Within each duel, the target variables of the datasets were modified so that one method excelled and the other underperformed. Additional requirement was to maximize diversity of the performance of all the algorithms. Using the optimization objectives, we selected 40 benchmarks with unique generative mathematical functions maximizing the diversity and ranking of ML algorithm performance. We provide not only synthetic datasets but also the generative functions used as the ground truth, multiple different analyses for each of the datasets, source code for running analysis, and a Docker script to replicate our study, everything as an open-source contribution for the ML community.

Details of the heuristic algorithm and its implementation can be found in the Supplementary Materials, which is available on GitHub Pages. Briefly, the DIGEN resource was built using a heavily modified version of the Heuristic Identification of Biological Architectures for Simulating Complex Hierarchical Genetic Interactions (HIBACHI) method and software (*20*) reinforced by selection of pareto-optimal solutions using the Non-dominated sorting genetic algorithm III (NSGA-III) strategy (*21*) and the state-of-the-art hyperparameter optimization framework Optuna (*22*). The popular scikit-learn ML library (*23*) was used for analysis with addition of XGBoost and LightGBM Python packages. Parameter ranges were set on the basis of the recommendations of the leading hyperoptimization frameworks, such as Optuna, auto-sklearn (*24*) and auto-sklearn2 (*25*), and hyperopt (*26*, *27*), and expert knowledge. The training and testing datasets were split 80/20 using a 10-fold cross-validation. The exact parameter settings of the methods could be found on GitHub.

The DIGEN benchmark resource includes 40 datasets simulated from each of the generative mathematical functions. Each includes 10 features or an independent variable generated from a normal distribution with a mean of zero and a variance of one, $N(0,1)$. The generative mathematical function accepts these features as input returning a continuously distributed outcome variable. We then sort and convert this outcome to binary values (0,1) to create a balanced binary class variable. Each of the 40 datasets has a sample size of $n = 1000$. We provide each of these 40 datasets as the benchmark. Datasets initiated with different random seeds or with different sizes could be generated using a Docker container available at GitHub. The functions themselves can be used with features drawn from the same normal distribution to generate as many datasets as desired with any sample and feature sizes. It would not be difficult to extend the heuristic algorithm to generate functions mapping other distributions of input and output data for problems such as those with binary inputs or multiclass or continuous outcomes.

RESULTS

There are several properties that make DIGEN a unique benchmark in the ML community. First, DIGEN differentiates the performance of multiple ML methods. Figure 1A shows a heatmap of the performance of reference ML algorithms (presented in columns) initialized with the default random seed and optimized with 200 hyperparameter evaluations across all 40 benchmark datasets (rows). The shade of the color is proportional to the AUROC. Also shown are the results of a hierarchical cluster analysis of the rows and columns visualized using dendrograms with shorter branches indicating higher similarity. Note that no method dominates all datasets. Second, DIGEN includes a diverse set of generative mathematical functions. Figure 1B shows a heatmap of the Ruzicka similarity of the 40 generative functions. The Ruzicka similarity between each pair of datasets represented as vectors x and y is calculated as the number of mathematical pairs of operators used one after another in the generative function and is defined as $\frac{\sum \min(x_i, y_i)}{\sum \max(x_i, y_i)}$ (*28*). Note the low similarity between all benchmark functions. Third, DIGEN is comprehensive. The heuristic algorithm optimization considered hundreds of thousands of combinations of mathematical operators using millions of hours of computing time. Furthermore, every ML algorithm considered by the heuristic algorithm was tuned by with 200 hyperparameter combinations evaluated on the basis of AUROC score computed with a 10-fold cross-validation.

In addition to the computational properties outlined above, DIGEN has several practical qualities for benchmarking. First, DIGEN datasets and results are reproducible. We have provided a Docker container, which allows the reproduction of our analyses. New datasets can be generated from the same feature distributions and mathematical functions producing the same patterns. This will allow the

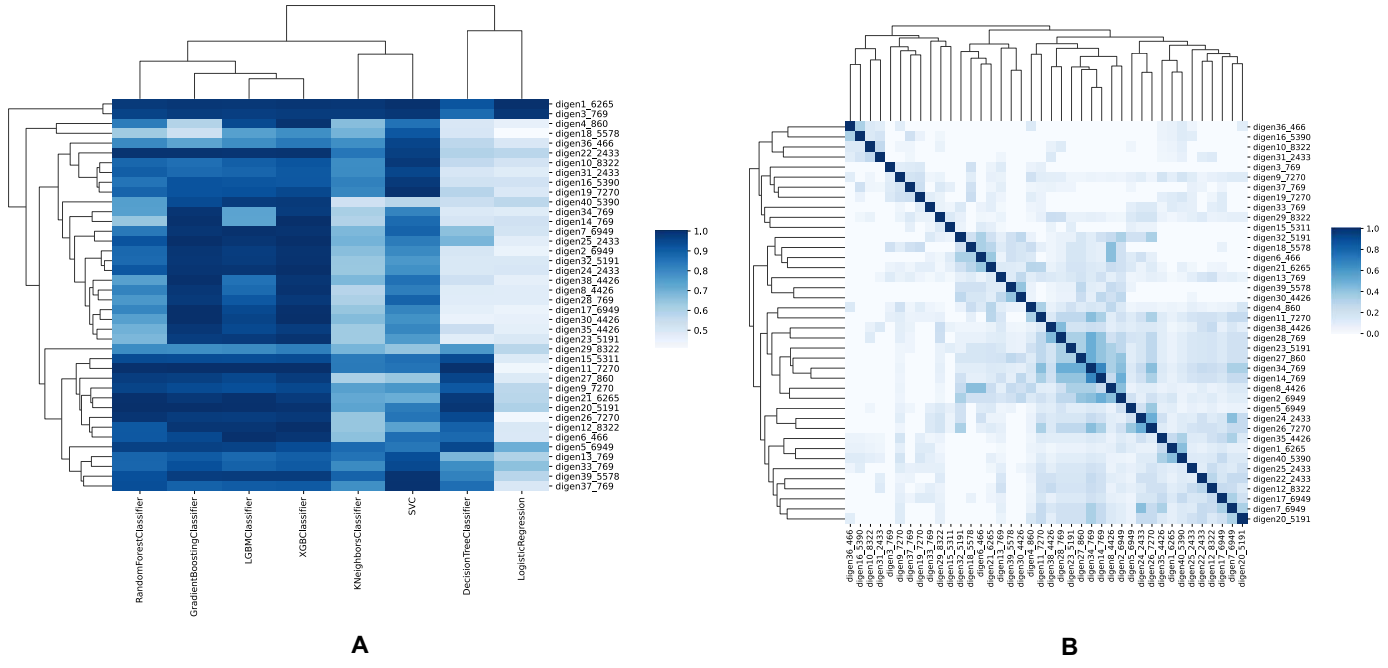

Fig. 1. Performance and similarity of ML methods across the 40 DIGEN benchmark datasets. (**A**) Heatmap of AUROC scores for each ML method (columns) with respect to each of the datasets (rows). (**B**) Heatmap of Ruzicka similarity (*28*) between each pair of datasets with regard to the number of common mathematical operators executed in proper order in the generative functions. All ML methods were tuned with respect to their hyperparameters.

benchmark to be recreated as necessary, yielding similar ML results depending on the random seed used. Second, DIGEN is scalable. The generative functions allow users to generate as many datasets as they want and with any desired sample size. DIGEN is also compact. The 40 benchmarks were selected from all of those generated by the heuristic to be diverse and to avoid unnecessary redundancy. The goal was to maximize the utility without burdening the user with too many benchmarks to evaluate. Third, DIGEN is simple, as it allows the user to very quickly discover where and why their method is not performing as good as the reference methods. To facilitate this, we have provided the code in Jupyter Notebooks for running the analyses and comparing the ML methods covered in the benchmark. We have included multiple precomputed statistics for each of the 40 generative functions, such as a feature correlation chart and box plots (Fig. 2A), which reflect the optimized performance with 100 evaluations of the ML algorithms across 100 replicate datasets initiated with different random seeds (Fig. 2B), receiver operating characteristics (ROC) plots (Fig. 2C), and precision-recall curve (PRC) plots (Fig. 2D). All the statistics were computed using tuned ML methods with a specific random seed given next to the name of the DIGEN dataset. Last, DIGEN is open source. All the source code along with the benchmark datasets are available on GitHub, with extensive documentations and tutorials on benchmarking new methods against DIGEN, and a docker container used to provide reproducibility for our experiments.

DISCUSSION

Despite the widespread use of benchmarks, there is general recognition that they have important limitations both in their design and their application. We review a few of those limitations here and then present the results of a study designed to generate a comprehensive suite of simulated benchmarks designed to reveal the strengths and weaknesses of ML classification methods.

Introducing a novel ML algorithm usually requires performing a comparison with well-established methods that would show that state-of-the-art methods are outperformed on collection of datasets. Thus, it comes with no surprise that more modern methods (such as gradient boosting, XGBoost, or LGBM classifiers) tend to perform similar or better than earlier ones on multiple tests. It also creates a challenge for benchmark designers to develop such datasets that would demonstrate the opposite. DIGEN approaches this by battling one method against the other, which is supposed to promote weaker classifiers against stronger ones, and ultimately lead to finding more complex and challenging ML tasks for modern methods.

The reasons why a given method shines on some datasets and trails on the other are yet to be understood. Several aspects come into play when analyzing this phenomenon. First, it is clear that the choice of a random seed affects both the dataset and initial settings of the ML methods. Our analysis of 100 runs, each starting from a different random seed, has shown that such differences may be notable. Second, the choice of sampler for hyperparameter optimization might play an important role in determining which combinations of parameters are chosen and ultimately lead to choosing a different model. Choosing a different sampler may ultimately lead to exploring different settings of ML method and improving its final performance. Third, the settings of the ML method hyperparameters and the number of evaluations may play a role. A too narrow search space affects the performance of the method, and a too broad one may simply result in missing important settings and ultimately in suboptimal performance. DIGEN addresses this by allowing each parameter to be found randomly with predefined distribution according to established guidelines for each method. The combination of the

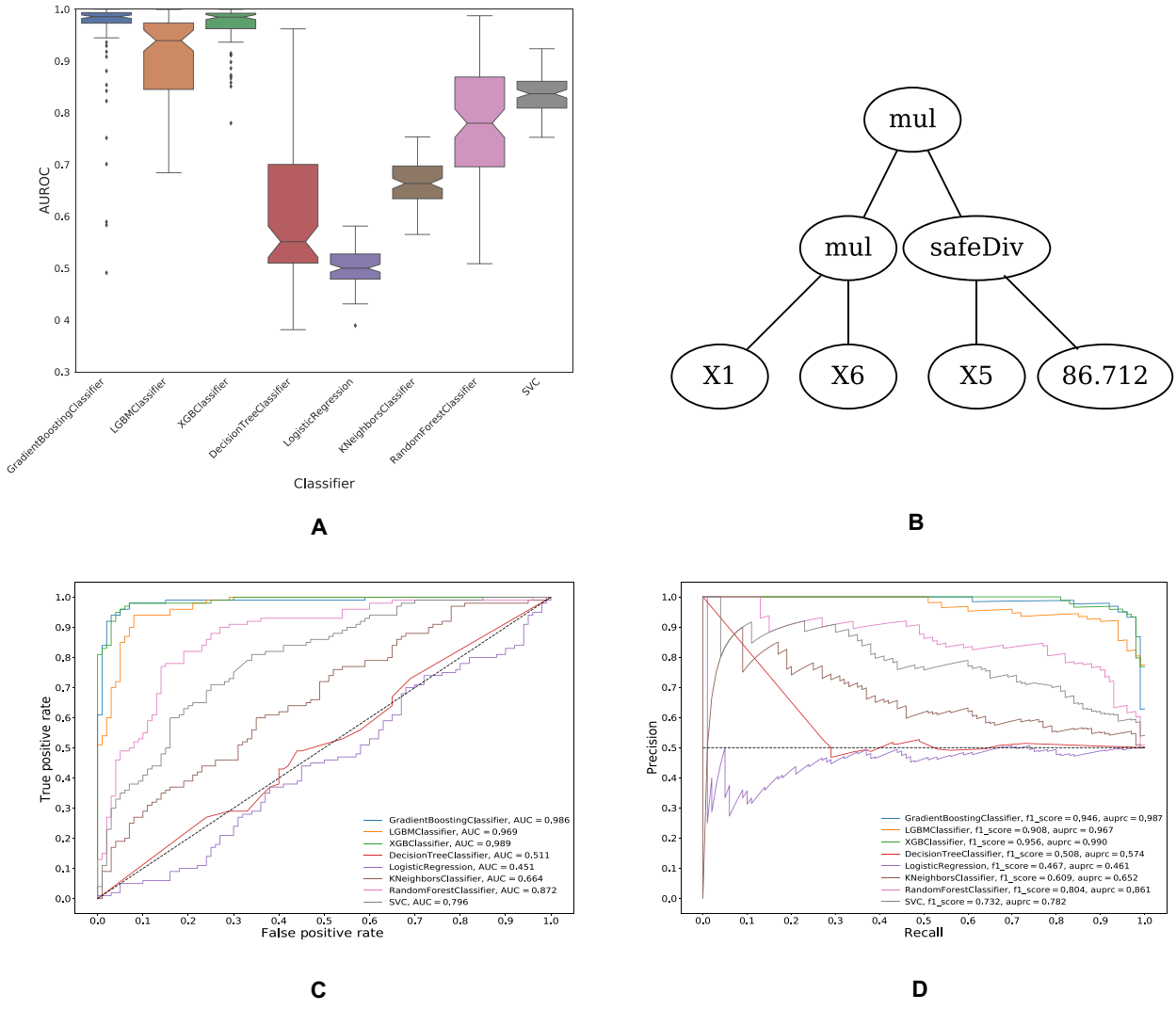

Fig. 2. Dataset-specific statistics. (**A**) Box plot showing performance of each tuned ML method after 100 iterations of hyperparameter tuning on 100 replicate datasets initialized with different random seeds. (**B**) Mathematical model used to generate the discrete outcome for this dataset (safeDiv performs a safe division, which prevents dividing by zero). (**C**) Plot showing combined ROC curves for ML methods run on the dataset initiated with the default seed. (**D**) Plot showing combined PRC plots for ML methods. AUROC, area under receiver operating characteristic curve; AUPRC, area under precision-recall curve.

three aforementioned elements apparently leads to situations in which a given ML method is unable to deliver the expected performance for a specific run initiated with a given random seed under a limited number of optimization constraints, although, in general, with multiple random seeds, it clearly outperforms the other method on the vast majority of runs, each with a different random seed.

In summary, we generated a comprehensive set of synthetic benchmark data to facilitate the evaluation and comparison of ML algorithms for classification of discrete outcomes. Each benchmark dataset comes with the generative mathematical function that can be used to create additional datasets or be used to provide some clues as to why an ML algorithm might not be performing well on that particular generative function. The resource is diverse, easy to use, and open source. Furthermore, it includes tutorials, the source code and mathematical formulas used to simulate the data. This allows us to easily extend the resource or benchmark the methods under similar environment. Sculley *et al.* (*29*) emphasized that counting the number of wins of a particular ML method is less important than discovering its strengths and weaknesses. In line with this proposal, DIGEN was designed as a tool for explaining where the method underperformed and hopefully also understanding why.

SUPPLEMENTARY MATERIALS

Supplementary material for this article is available at https://science.org/doi/10.1126/sciadv.abl4747

REFERENCES AND NOTES

1. L. Breiman, J. Friedman, C. J. Stone, R. A. Olshen, *Classification and Regression Trees* (CRC Press, 1984).
2. J. R. Quinlan, Learning efficient classification procedures and their application to chess end games, in *Machine Learning* (Springer, 1983), pp. 463–482.
3. J. R. Quinlan, Induction of decision trees. *Mach. Learn.* **1**, 81–106 (1986).
4. T. K. Ho, Random decision forests, in *Proceedings of 3rd International Conference on Document Analysis and Recognition* (IEEE, 1995), vol. 1, pp. 278–282.
5. L. Breiman, Random Forests. *Mach. learn.* **45**, 5–32 (2001).

6. C. Cortes, V. Vapnik, Support-vector networks. *Mach. learn.* **20**, 273–297 (1995).
7. J. H. Friedman, Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **29**, 1189–1232 (2001).
8. T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2016), pp. 785–794.
9. G. Ke, Q. Meng, T. Finly, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, in *Advances in Neural Information Processing Systems* (NeurIPS, 2017), pp. 3146–3154.
10. D. Dua, C. Graff, UCI Machine Learning Repository (NeurIPS, 2017).
11. C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* **2**, 1–27 (2011).
12. R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, J. H. Moore, PMLB: A large benchmark suite for machine learning evaluation and comparison. *BioData Mining* **10**, 36 (2017).
13. B. Bischl, G. Casalicchio, M. Feurer, P. Gijsbers, F. Hutter, M. Lang, R. G. Mantovani, J. N. van Rijn, J. Vanschoren, OpenML benchmarking suites (2021). Accepted at NeurIPS 2021; https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/c7e1249ffc03eb9ded908c236bd1996d-Paper-round2.pdf.
14. P. Orzechowski, W. La Cava, 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* (Association for Computing Machinery, 2018), pp. 1183–1190.
15. S.-M. Udrescu, M. Tegmark, AI Feynman: A physics-inspired method for symbolic regression. *Sci. Adv.* **6**, eaay2631 (2020).
16. W. La Cava, P. Orzechowski, B. Burlacu, Fabrício Olivetti de França, M. Virgolin, Y. Jin, M. Kommenda, J. H. Moore, Contemporary symbolic regression methods and their relative performance (NeurIPS, 2021); https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/c0c7c76d30bd3dcaefc96f40275bdc0a-Paper-round1.pdf.
17. Kaggle Network, Kaggle network: Your home for data science (2011); https://www.kaggle.com/.
18. T. Cover, P. Hart, Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **13**, 21–27 (1967).
19. P. McCullagh, Generalized linear models. *Eur. J. Oper. Res.* **16**, 285–292 (1984).
20. J. H. Moore, R. S. Olson, P. Schmitt, Y. Chen, E. Manduchi, How computational thought experiments can improve our understanding of the genetic architecture of common human diseases, in *Artificial Life Conference Proceedings* (MIT Press, 2018), pp. 23–30.
21. K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Trans. Evol. Comput.* **18**, 577–601 (2014).
22. T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A Next-generation hyperparameter optimization framework, in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2019), pp. 2623–2631.
23. 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, É. Duchesnay, Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).
24. M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, F. Hutter, Efficient and Robust Automated Machine Learning, in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett, eds. (Curran Associates Inc., 2015), pp. 2962–2970.
25. M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, F. Hutter, Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning. arXiv:2007.04074 [cs.LG]. (2020).
26. J. Bergstra, D. Yamins, D. Cox, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, in *Proceedings of the 30th International Conference on International Conference on Machine Learning* (JMLR, 2013), vol. 28, pp. 115–123.
27. J. Bergstra, D. Yamins, D. D. Cox, Hyperopt: A Python library for optimizing the hyperparameters of machine learning algorithms, in *Proceedings of the 12th Python in Science Conference* (Citeseer, 2013), vol. 13, p. 20.
28. M. Ruzicka, Anwendung mathematisch-statistischer Methoden in der Geobotanik (synthetische Bearbeitung von Aufnahmen). *Biologia* **13**, 647–661 (1958).
29. D. Sculley, J. Snoek, A. B. Wiltschko, A. Rahimi, Winner's curse? On pace, progress, and empirical rigor (ICLR, 2018).
30. J. R. Koza, J. R. Koza, Genetic Programming: on the Programming of Computers by Means of Natural Selection (MIT Press, 1992), vol. 1.

Acknowledgments: We would like to thank W. La Cava for comments and discussion and the Penn Medicine Academic Computing Services (PMACS) for supporting the computational experiments. **Funding:** This research was supported, in part, by PL-Grid infrastructure and NIH grants AI116794, LM010098, AG066833, and LM012601. **Author contributions:** J.H.M. and P.O. conceived the study. P.O. implemented the software and carried out the experiments and wrote the initial draft of the paper. Both authors discussed the results and contributed to the final manuscript. J.H.M. supervised the project. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data needed to evaluate the conclusions in the paper are present in the paper and/or Supplementary Materials. The most recent version of the software, along with tutorials and extensive documentation, was stored in Zenodo https://doi.org/10.5281/zenodo.6484789 and can also be found on GitHub at https://github.com/EpistasisLab/digen/.

Submitted 3 August 2021
Accepted 7 October 2022
Published 23 November 2022
10.1126/sciadv.abl4747