Chapter 8
A System for Accessible Artificial Intelligence

Randal S. Olson, Moshe Sipper, William La Cava, Sharon Tartarone, Steven Vitale, Weixuan Fu, Patryk Orzechowski, Ryan J. Urbanowicz, John H. Holmes, and Jason H. Moore

Abstract While artificial intelligence (AI) has become widespread, many commercial AI systems are not yet accessible to individual researchers nor the general public due to the deep knowledge of the systems required to use them. We believe that AI has matured to the point where it should be an accessible technology for everyone. We present an ongoing project whose ultimate goal is to deliver an open source, user-friendly AI system that is specialized for machine learning analysis of complex data in the biomedical and health care domains. We discuss how genetic programming can aid in this endeavor, and highlight specific examples where genetic programming has automated machine learning analyses in previous projects.

R. S. Olson · W. La Cava · S. Tartarone · S. Vitale · W. Fu · R. J. Urbanowicz · J. H. Holmes
J. H. Moore (✉)
Institute for Biomedical Informatics, University of Pennsylvania, Philadelphia, PA, USA
e-mail: rso@randalolson.com; lacava@upenn.edu; ryanurb@pennmedicine.upenn.edu; jhmoore@upenn.edu

M. Sipper
Institute for Biomedical Informatics, University of Pennsylvania, Philadelphia, PA, USA

Department of Computer Science, Ben-Gurion University, Beer-Sheva, Israel
e-mail: sipper@cs.bgu.ac.il

P. Orzechowski
Institute for Biomedical Informatics, University of Pennsylvania, Philadelphia, PA, USA

Department of Automatics and Biomedical Engineering, AGH University of Science and Technology, Krakow, Poland

W. Banzhaf et al. (eds.), *Genetic Programming Theory and Practice XV*,
Genetic and Evolutionary Computation, https://doi.org/10.1007/978-3-319-90512-9_8

8.1 Introduction

A central goal of artificial intelligence (AI) is to use computational hardware and software to solve complex problems in a human-competitive manner [9]. The practicality of this goal is that AI can be tasked with solving problems or performing functions that humans cannot perform or simply do not have time for. Most AI methodologies can be grouped into top-down approaches, wherein cognition is viewed as a high-level phenomenon that is independent of the low-level details, or bottom-up approaches, which define basic computational building blocks such as artificial neurons that collectively give rise to "emergent" [29] intelligent behavior. The top-down approach has been difficult to realize given the inherent complexity of human cognition. However, the bottom-up has had some success owing to the availability of sophisticated algorithms such as genetic programming (GP) [10] and deep neural networks [6]. This is particularly true today with abundant and inexpensive high-performance computing, leading to many human-competitive success stories [9].

Medical applications of AI have had a long history with both successes and failures. One of the early successes was the Mycin system, which was designed to predict the antibiotic that a patient with an infection should receive in the intensive care unit [2]. Mycin combined a knowledge base along with a set of rules implemented as part of an expert system. The system was demonstrated to be human-competitive, but was never put into clinical practice because of legal concerns and the time it took clinicians to enter the patient data required for Mycin to make the predictions. The field of AI has matured since Mycin was developed and, importantly, computing power has grown tremendously in parallel. Examples of modern AI successes include IBM's Watson, which beat the world champion of the game show Jeopardy [5]. The Watson AI system that won Jeopardy combined knowledge representation, information retrieval, natural language processing, and machine learning along with high-performance computing to access and exploit a knowledge base that included the Wikipedia text corpus. This was a milestone in AI because it showed that a computational system could compete with humans on difficult language processing tasks. Watson is now being marketed in the health care domain although the jury is still out on its effectiveness.

Commercial AI systems such as Watson show potential but are not yet accessible to individual researchers nor the general public due the cost and the complexity of working with a team from IBM. It is our working hypothesis that,

***AI has matured to the point where it should be an accessible technology for everyone*.**

Democratization of AI will be important if we seek to integrate this exciting new technology into multiple different domains, as demonstrated by recent efforts such as Orange [4]. We describe here the early development stages of an open source and user-friendly AI system—PennAI (http://pennai.org)—for machine learning analysis of complex data in the biomedical and health care domains. We focus our initial efforts on the classification of biomedical endpoints such as disease susceptibility. We describe in turn below each of the components of our AI system and then end with an example and a discussion of how we envision this system

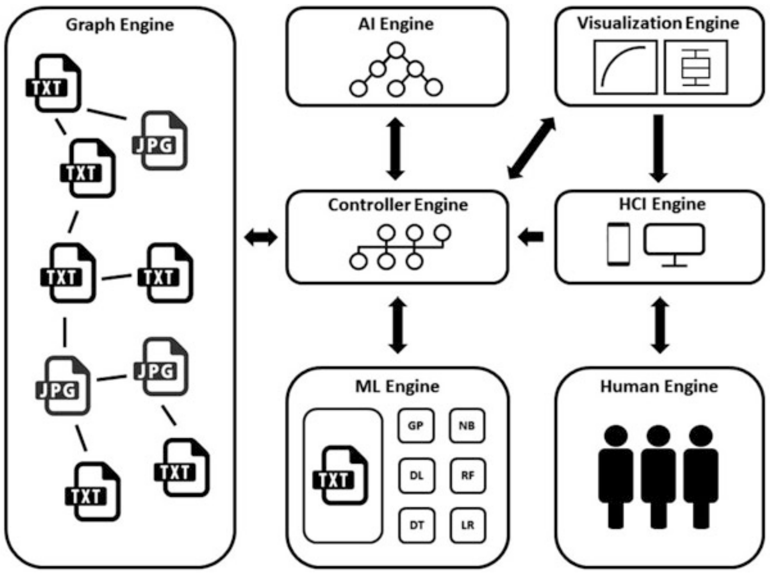


Fig. 8.1 The components of PennAI, a user-friendly AI system developed at the University of Pennsylvania

being used to solve complex biomedical problems. Further, we discuss how GP can aid in enhancing PennAI, and highlight specific examples where GP has automated machine learning analyses in previous work.

The components of PennAI include a human engine (i.e., the user); a user-friendly interface for interacting with the AI; a machine learning engine for data mining; a controller engine for launching jobs and keeping track of analytical results; a graph database for storing data and results (i.e., the memory); an AI engine for monitoring results and automatically launching or recommending new analyses; and a visualization engine for displaying results and analytical knowledge (Fig. 8.1). This AI system provides a comprehensive set of integrated components for automated machine learning (AutoML), thus providing a data science assistant for generating useful results from large and complex data problems. PennAI is housed in the "Idea Factory," a facility designed to facilitate collaboration and promote new methods of communicating and presenting scientific innovation. The Idea Factory makes sophisticated data visualization and artificial intelligence analytics easy for users across the entire Penn community (Fig. 8.2).

8.2 The Human Engine

The most important component of the proposed AI system is the user. Contrary to some claims that AI will replace human users, we see the human as an integral part of the discovery process and a partner with the AI. One way to view this

Fig. 8.2 The "Idea Factory," home of PennAI

partnership is with the human as the driver of the discovery process and the AI as the data science assistant. Thus, the AI provides an additional set of hands in a modern data science discovery environment that might include human teammates with expertise in computer science, statistics, and applied mathematics. We have previously suggested this idea of human-computer interaction that places the human user at the epicenter [22]. This idea has also previously been explored from the point of view of the user or domain expert [16].

Langley [16] provides five important tips that are relevant to thinking about the relationship humans have with AI for data mining using machine learning. First, traditional machine learning notations are not easily communicated to scientists. This consideration is important because a machine learning model may not be interpretable by a user. Second, scientists often have initial models that should influence the discovery process. Domain-specific knowledge can be critical to the discovery process. Third, scientific datasets are often rare and difficult to obtain. It often takes years to collect and process the data before it can be analyzed. As such, it is important that the analysis is carefully planned and executed, and that any general feedback about the performance of the learning process is not lost between studies. Fourth, scientists want models that move beyond description and provide explanations of the data. Explanation and interpretation are paramount to the user. Finally, scientists want computational assistance rather than a complete replacement of themselves. Langley [16] further suggests that users want interactive discovery environments that help them understand their data while at the same time giving them control over the modeling process. Collectively, these five lessons suggest that synergy between the user and the AI is critical. With this in mind, our proposed AI system includes a graphical user interface (GUI) that allows the user to easily launch analyses, view the results, and give the AI feedback about what results are useful or interesting.

8.3 The Human-Computer Interaction Engine

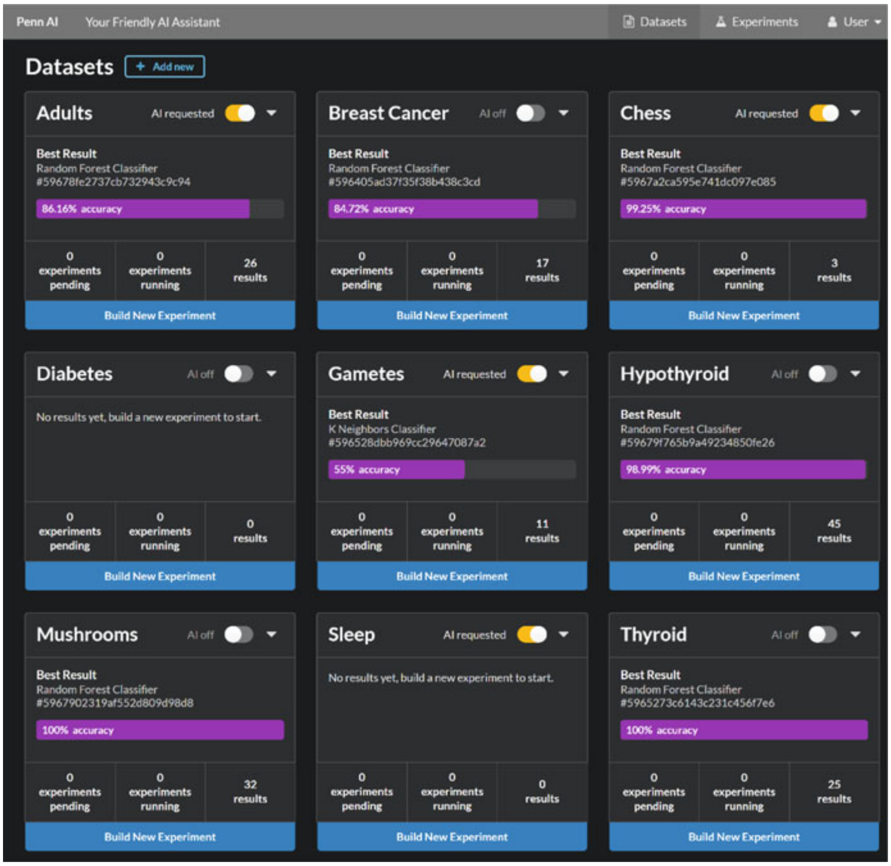
As described above, a key component of PennAI is human-computer interaction. The first important feature is to make it easy for the user to directly launch machine learning analyses by choosing a method and its parameter settings from an intuitive push-button menu implemented through the web using JavaScript. The user can launch single analyses or, in an advanced mode, launch a grid search across multiple methods and parameter settings. The methods and the controller that keep track of these analyses is described below. Figures 8.3 and 8.4 show prototypes of our GUI for uploading and viewing datasets for analysis and launching machine learning analyses on those datasets, respectively. Our JavaScript implementation is compatible with mobile devices, which allows the user to interact with the AI system from any Internet-connected device.

The second key feature of PennAI is the ability to toggle the AI on and off for automated analysis, shown in Fig. 8.3. An AI toggle allows the user to turn the AI on and set parameters controlling the maximum number of runs the AI can launch, as well as the frequency of updates the user would like to receive by email or text message. The GUI also provides a simple thumbs up/down selection for each result received by PennAI, which provides feedback to PennAI that is incorporated into its expert knowledge system.

8.4 The Machine Learning Engine

Our first application of PennAI is for data mining using machine learning in the biomedical domain. Here, we make use of an extensive open source machine learning library in Python called scikit-learn [28]. Scikit-learn provides peer-reviewed implementations of several common supervised and unsupervised machine learning algorithms, data preprocessing methods, feature engineering and selection methods, hyperparameter optimization procedures, and more. To most users, scikit-learn is considered to be the standard machine learning library in Python.

Of course, there are dozens of machine learning algorithms, preprocessors, etc. to choose from in scikit-learn, which can be overwhelming to a novice user. To simplify the algorithm selection process for PennAI users, we currently limit PennAI to six machine learning algorithms that we believe will handle most supervised classification use cases, shown in Table 8.1. We also limit the parameter choices for each algorithm to a handful of the most important parameters and parameter



Adults — AI requested
Best Result
Random Forest Classifier
#59678fe2737cb732943c9c94
86.16% accuracy
0 experiments pending | 0 experiments running | 26 results

Breast Cancer — AI off
Best Result
Random Forest Classifier
#596405ad37f35f38b438c3cd
84.72% accuracy
0 experiments pending | 0 experiments running | 17 results

Chess — AI requested
Best Result
Random Forest Classifier
#5967a2ca595e741dc097e085
99.25% accuracy
0 experiments pending | 0 experiments running | 3 results

Diabetes — AI off
No results yet, build a new experiment to start.
0 experiments pending | 0 experiments running | 0 results

Gametes — AI requested
Best Result
K Neighbors Classifier
#596528dbb969cc29647087a2
55% accuracy
0 experiments pending | 0 experiments running | 11 results

Hypothyroid — AI off
Best Result
Random Forest Classifier
#59679f765b9a49234850fe26
98.99% accuracy
0 experiments pending | 0 experiments running | 45 results

Mushrooms — AI off
Best Result
Random Forest Classifier
#5967902319af552d809d98d8
100% accuracy
0 experiments pending | 0 experiments running | 32 results

Sleep — AI requested
No results yet, build a new experiment to start.
0 experiments pending | 0 experiments running | 0 results

Thyroid — AI off
Best Result
Random Forest Classifier
#5965273c6143c231c456f7e6
100% accuracy
0 experiments pending | 0 experiments running | 25 results

Fig. 8.3 Prototype of the graphical user interface for managing and viewing datasets

Table 8.1 Machine learning algorithms available in PennAI

Classification	Regression
Logistic regression	ElasticNet
Decision tree	Decision tree
k-Nearest neighbors	k-Nearest neighbors
Support vector machine	Support vector machine
Random forest	Random forest
Gradient boosting	Gradient boosting

options, which makes it easier for users to choose a parameter configuration at the expense of algorithm customizability. An example of the interface to the Machine Learning Engine can be found in Fig. 8.4, where only a handful of the most important parameters and parameter options are available for the k-Nearest Neighbors classification algorithm.

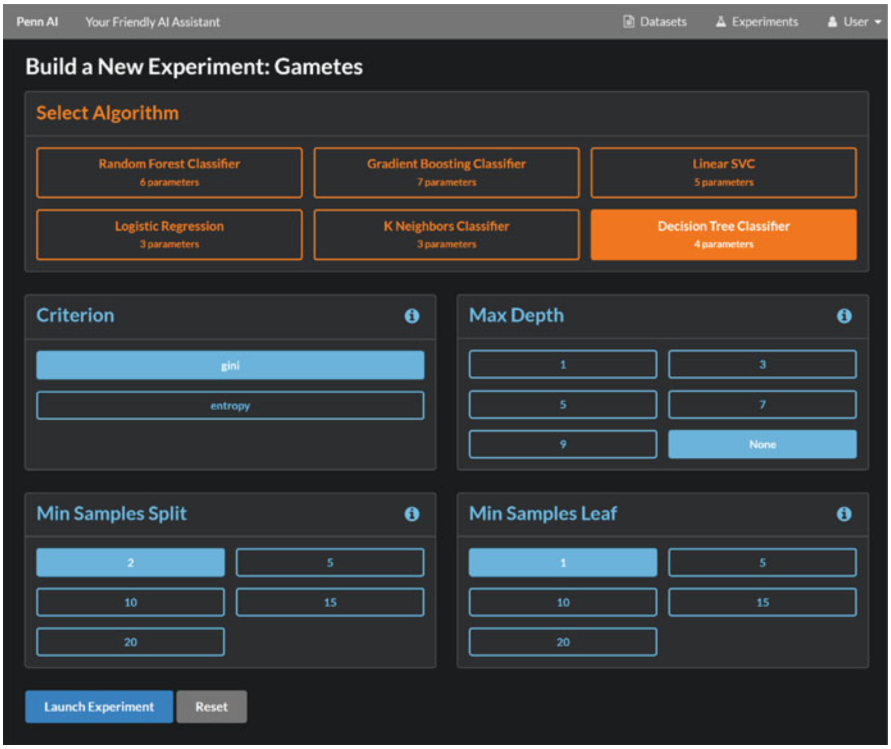


Fig. 8.4 Prototype of the Machine Learning Engine graphical user interface

In an upcoming PennAI implementation, we will provide simplified descriptions of the machine learning algorithms and parameters so users can make use of the algorithms without fully understanding their implementation. For example, when using a random forest it is not necessary for the user to understand what tuning the `n_estimators` parameter does to the model. Instead, it is more important for the user to understand that adding more decision trees to the random forest (i.e., increasing `n_estimators`) improves model performance but increases training time, whereas removing decision trees from the random forest decreases model performance but decreases training time [7].

Once the Machine Learning Engine finishes training and evaluating a machine learning model, it stores the machine learning model, the model predictions, and an analysis of the model in the Graph Database Engine, which are used in the Visualization Engine (both described below).

8.5 The Controller Engine

The Controller Engine acts as the interface between the high-performance computing system and the user or AI. This component is hidden from the user but facilitates the automatic launching of jobs on a multi-CPU machine, computing cluster, or cloud computing system. The controller must not only coordinate the launching of jobs but also keep track of when they finish and deposit the results in the Graph Database Engine (described below) that serves as the memory of the system.

For the Controller Engine, we selected an open source package called the Future Gadget Lab (FGLab), which is available on GitHub.[1] FGLab functions as a server with individual runs launched as clients, called FGMachines. FGLab uses node.js to coordinate distributed jobs and uses MongoDB [3] as the backend database in the Graph Database Engine.

8.6 The Graph Database Engine

Another key component of PennAI is a memory system that keeps track of every analysis that is run on each data set. We keep track of the details of the machine learning method, the parameter settings, the data set analyzed, and results such as the model, model error, and area under the receiver operating characteristic curve (AUC). These are all stored in a JSON file that is deposited in a MongoDB NoSQL database. The advantage of using a NoSQL database is that new data elements can be added without creating tables and without strict format specifications. This flexibility is important for the rapidly changing landscape of machine learning. MongoDB can also function as a graph database that allows the documents to be linked in a network according to shared index terms related to the analysis and data. This feature facilitates more complex semantic queries of the database, such as "Return the machine learning algorithm configurations that achieved the highest accuracy on any study involving prostate cancer."

8.6.1 Knowledge Base

The Graph Database Engine serves as the memory of PennAI and provides the raw materials for the AI to learn which methods and parameter settings are working better than others for particular kinds of problems. The initial knowledge base consists of results from a previously published benchmark of scikit-learn algorithms [24], in which 14 machine learning algorithms were run with full

[1]FGLab: https://github.com/Kaixhin/FGLab.

hyperparameter optimization on a suite of 165 supervised classification problems. The results are combined with meta-information about the datasets (e.g., number of features, number of instances, correlations between features, etc.) in order to allow the creation of a mapping from ‘problem instance space’, i.e. dataset meta-features and model performance, to ‘learning space’, i.e. machine learning algorithms and their parameters. This data can then be modelled to extract rules that represent the knowledge used by the Artificial Intelligence Engine to make informed analyses. The knowledge base will be updated with all future analyses.

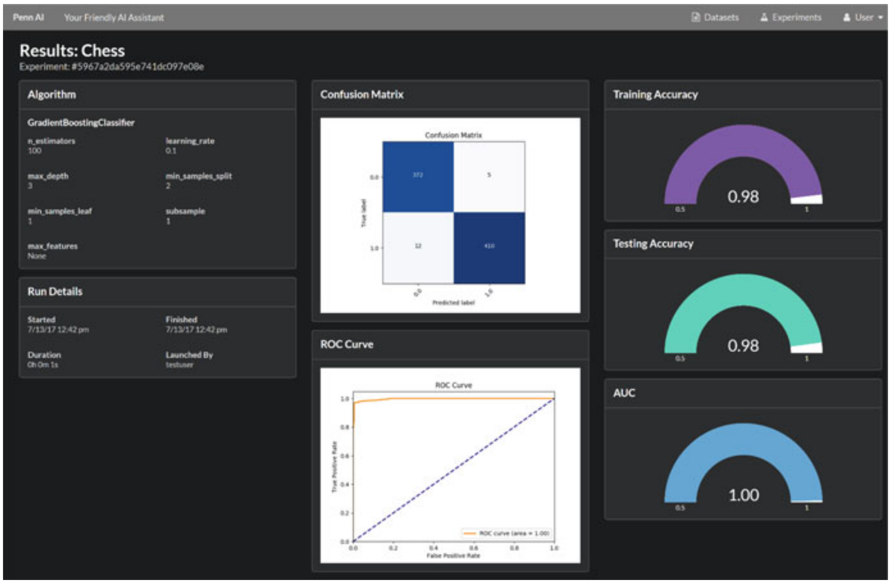
8.7 The Artificial Intelligence Engine

Each component described above provides the raw materials for the Artificial Intelligence Engine which then (1) searches the graph database for results related to one or more data sets, (2) performs statistical analysis comparing algorithms and their parameters, (3) combines facts and rules in an expert system to make new analysis recommendations, (4) communicates findings to the user, and (5) automatically launches new analyses using suggestions from the expert system. The first function uses the search capabilities of the MongoDB graph database to identify relevant machine learning results in the form of JSON files. All returned JSON files can be parsed to extract the machine learning algorithm, parameters, and information about the model performance. These results are collated in a tab-delimited file and a statistical analysis performed to determine the best algorithm configurations for certain problem types, similar to meta-learning techniques [8].

New statistical results are used to populate the knowledge base of an expert system that has a set of decision rules provided by developers and advanced machine learning practitioners. This expert system is then used to make suggestions for additional analyses, for example by recommending better parameter settings or even entirely different machine learning algorithms that might be better-suited for the user’s dataset. The user can access these suggestions manually or PennAI can use the suggestions to automatically launch new jobs, thus continually growing the PennAI knowledge base. Essentially, the Artificial Intelligence Engine becomes a research assistant who tinkers with new ways of modeling the dataset and reports back to the user with their best findings.

8.8 The Visualization Engine

Visualization will be critical for fostering the human-AI collaboration described above. The user will need to be able to see individual machine learning models and results as well as higher-level results from statistical analyses across machine learning runs. We extract visual results such as the receiver operating characteristic (ROC) curves and models to store in the graph database, as shown in Fig. 8.5.



Results: Chess
Experiment: #5967a2da595e741dc097e08e

Algorithm
GradientBoostingClassifier
n_estimators 100
learning_rate 0.1
max_depth 3
min_samples_split 2
min_samples_leaf 1
subsample 1
max_features None

Run Details
Started 7/13/17 12:42 pm
Finished 7/13/17 12:42 pm
Duration 0h 0m 1s
Launched By testuser

Fig. 8.5 Prototype of the Visualization Engine graphical user interface

PennAI will also generate heatmaps and other visualizations that summarize results across different machine learning methods and datasets. These higher-level visualizations will aid the user with making decisions about new manual analyses to launch and will help them assess how well the PennAI assistant is doing. These images will be linked to the datasets and results in the Graph Database Engine, and will thus be easily searchable.

8.9 Discussion and Future Work

Thus far, we have described PennAI as a system that provides a simple interface for users to upload their datasets, launch machine learning analyses, view the results of the analyses in an intuitive manner, and use those results to refine their machine learning analyses. We also described how PennAI will use a combination of expert knowledge from advanced machine learning practitioners and prior statistical knowledge of machine learning algorithm performance on datasets to recommend new analyses to the user, as well as launch its own analyses to later report to the user. In essence, the primary goal of PennAI is to provide an AI research assistant for its users. However, considering the name of this workshop and book—Genetic Programming Theory and Practice—one may be left wondering how GP can be incorporated into PennAI. In the following paragraphs, we will describe our plans for integrating GP into PennAI.

The first point of entry is to include GP as a machine learning option since a number of successful biomedical applications have been reported (e.g., [17–21, 34]). A GP system for classification based on multidimensional clustering [31] was recently demonstrated on biomedical classification problems [15] as a competitive alternative to traditional machine learning approaches. Recently GP has been proposed as a general feature engineering wrapper (FEW)[2] in order to harness its feature learning capability to improve scikit-learn estimators, both for regression [13] and classification [14]. FEW allows GP to provide readable feature transformations to users while still utilizing existing modeling techniques for making predictions. As mentioned in Sect. 8.2, interpretation and explanatory power are important aspects of using AI for data mining, and therefore GP methods that produce concise models, e.g. by local search [11] or Pareto optimization [12], are important options to include. Further down the road, it could be possible for PennAI to allow advanced users to incorporate custom machine learning algorithms into PennAI by providing a scikit-learn formatted interface to their project (e.g. ellyn[3]). PennAI could then provide a "bring your own learner" type of service [1] to allow researchers to tackle complex data mining tasks with customized learning approaches, and incorporate the results into its knowledge base for improving future data science projects.

Beyond using GP to perform the machine learning itself, recent work has shown that GP can also be harnessed to optimize a sequence of existing data analysis and machine learning operations on a dataset to maximize the predictive performance of the final machine learning model [30, 35]. For example, TPOT[4] is an early prototype that uses GP to optimize a sequence of scikit-learn operations for both classification and regression problems [25–27], and has been shown to work quite well across a broad range of application domains ranging from epidemiological studies to image classification to time series prediction [23]. Given the general design of TPOT, the operations it optimizes over can be specialized for particular problem domains. As another example, the TPOT-MDR project [33] showed that TPOT can be specialized for genome-wide association studies (GWAS), and it outperforms several state-of-the-art modeling methods on both simulated and real-world GWAS problems because it considers a broad range of operations in with one another. As such, we view GP as a strong candidate for a future version of the PennAI Artificial Intelligence Engine, where the GP is seeded with the best known algorithm configurations and uses the core principles of GP (inheritance, mutation, and crossover)—distributed over a high-performance computing cluster—to improve the algorithm configurations from there. This brand of GP-based AI system would be useful for automatically launching new analyses, but less useful for recommending particular algorithm configurations to the user because GP does not provide a notion of the "next best" solution to attempt.

[2]http://lacava.github.io/few.

[3]http://epistasislab.github.io/ellyn.

[4]https://github.com/rhiever/tpot.

Another extension of PennAI is the use of a meta genetic algorithm to find parameters (population size, generation count, etc.) for a GP instance that work well, i.e., solve a given problem [32]. This meshes well with the idea that the AI of PennAI will aid non-machine learning experts run complex algorithms, such as GP, without having to find or even understand every single parameter.

Ultimately, PennAI will likely be comprised of several disparate AI algorithms that use meta-data and meta-learning to improve the user experience and user productivity by suggesting machine learning algorithms and parameters, as well as providing other insights. As a result, we will be able to harness ensemble techniques to collate the advice given by the numerous AI algorithms.

The time is now to bring AI technology to anyone that wants to use it for big data analytics. The software and hardware technology exists and data has never been bigger, more complex, and more plentiful. PennAI will provide both machine learning and AI capability to both naive and expert users alike with a user-friendly web and smartphone-enabled interface. We see AI technology such as PennAI not as a replacement for the data scientist but rather as a data science assistant that can suggest analyses to the user or provide automatically generated results that are informed by previous analyses across different data sets. The user can take these results as-is or use them as inspiration in manual analyses. The democratization of AI is here.

Acknowledgements This work was generously funded by the Perelman School of Medicine and the University of Pennsylvania Health System. Additional funding was provided by National Institutes of Health grants AI116794, DK112217, ES013508, and TR001878.

References

1. Arnaldo, I., Veeramachaneni, K., Song, A., O'Reilly, U.M.: Bring your own learner: A cloud-based, data-parallel commons for machine learning. IEEE Computational Intelligence Magazine **10**(1), 20–32 (2015)
2. Bruce, G., Buchanan, B., Shortliffe, E.: Rule-based expert systems: The MYCIN experiments of the Stanford heuristic programming project (1984)
3. Chodorow, K., Dirolf, M.: MongoDB: The Definitive Guide, 1st edn. O'Reilly Media, Inc. (2010)
4. Demšar, J., Curk, T., Erjavec, A., Črt Gorup, Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M., Zupan, B.: Orange: Data mining toolbox in Python. Journal of Machine Learning Research **14**, 2349–2353 (2013)
5. Ferrucci, D.A.: Introduction to "This is Watson". IBM Journal of Research and Development **56**(3.4), 1–1 (2012)
6. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT Press (2016)
7. Hastie, T.J., Tibshirani, R.J., Friedman, J.H.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York, NY, USA (2009)
8. Kalousis, A.: Algorithm selection via meta-learning. Ph.D. thesis, Universite de Geneve (2002)
9. Kannappan, K., Spector, L., Sipper, M., Helmuth, T., La Cava, W., Wisdom, J., Bernstein, O.: Analyzing a decade of human-competitive ("HUMIE") winners: What can we learn? In: Genetic Programming Theory and Practice XII, pp. 149–166. Springer International Publishing (2015)

10. Koza, J.R.: Genetic programming: on the programming of computers by means of natural selection, vol. 1. MIT Press (1992)
11. La Cava, W., Danai, K., Spector, L.: Inference of compact nonlinear dynamic models by epigenetic local search. Engineering Applications of Artificial Intelligence **55**, 292–306 (2016)
12. La Cava, W., Danai, K., Spector, L., Fleming, P., Wright, A., Lackner, M.: Automatic identification of wind turbine models using evolutionary multiobjective optimization. Renewable Energy **87**, 892–902 (2016)
13. La Cava, W., Moore, J.: A general feature engineering wrapper for machine learning using ϵ-lexicase survival. In: European Conference on Genetic Programming, pp. 80–95. Springer (2017)
14. La Cava, W., Moore, J.H.: Ensemble representation learning: an analysis of fitness and survival for wrapper-based genetic programming methods. In: GECCO '17: Proceedings of the Conference on Genetic and Evolutionary Computation. ACM (2017)
15. La Cava, W., Silva, S., Vanneschi, L., Spector, L., Moore, J.: Genetic programming representations for multi-dimensional feature learning in biomedical classification. In: European Conference on the Applications of Evolutionary Computation, pp. 158–173. Springer (2017)
16. Langley, P.: Lessons for the Computational Discovery of Scientific Knowledge (2002)
17. Moore, J.H., Andrews, P.C., Barney, N., White, B.C.: Development and evaluation of an open-ended computational evolution system for the genetic analysis of susceptibility to common human diseases. In: European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics, pp. 129–140. Springer (2008)
18. Moore, J.H., Greene, C.S., Hill, D.P.: Identification of novel genetic models of glaucoma using the "emergent" genetic programming-based artificial intelligence system. In: R. Riolo, W.P. Worzel, M. Kotanchek (eds.) Genetic Programming Theory and Practice XII, pp. 17–35. Springer International Publishing, Cham (2015)
19. Moore, J.H., Greene, C.S., Hill, D.P.: Identification of novel genetic models of glaucoma using the "emergent" genetic programming-based artificial intelligence system. In: Genetic Programming Theory and Practice XII, pp. 17–35. Springer (2015)
20. Moore, J.H., Hill, D.P., Fisher, J.M., Lavender, N., Kidd, L.C.: Human-computer interaction in a computational evolution system for the genetic analysis of cancer. In: R. Riolo, E. Vladislavleva, J.H. Moore (eds.) Genetic Programming Theory and Practice IX, pp. 153–171. Springer New York, New York, NY (2011)
21. Moore, J.H., Hill, D.P., Saykin, A., Shen, L.: Exploring interestingness in a computational evolution system for the genome-wide genetic analysis of alzheimer's disease. In: R. Riolo, J.H. Moore, M. Kotanchek (eds.) Genetic Programming Theory and Practice XI, pp. 31–45. Springer New York, New York, NY (2014)
22. Moore, J.H., White, B.C.: Genome-wide genetic analysis using genetic programming: The critical need for expert knowledge. In: Genetic Programming Theory and Practice IV, pp. 11–28. Springer (2007)
23. Olson, R.S., Bartley, N., Urbanowicz, R.J., Moore, J.H.: Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. In: GECCO 2016, GECCO '16, pp. 485–492. ACM, New York, NY, USA (2016)
24. Olson, R.S., La Cava, W., Orzeshowski, P., Urbanowicz Ryan J Moore, J.H.: PMLB: A large benchmark suite for machine learning evaluation and comparison. arXiv e-print. https://arxiv.org/abs/1703.00512 (2017)
25. Olson, R.S., Moore, J.H.: Identifying and Harnessing the Building Blocks of Machine Learning Pipelines for Sensible Initialization of a Data Science Automation Tool. arXiv e-print. http://arxiv.org/abs/1607.08878 (2016)
26. Olson, R.S., Moore, J.H.: TPOT: A Tree-based Pipeline Optimization Tool for Automating Machine Learning. JMLR **64**, 66–74 (2016)
27. Olson, R.S., Urbanowicz, R.J., Andrews, P.C., Lavender, N.A., Kidd, L.C., Moore, J.H.: Automating Biomedical Data Science Through Tree-Based Pipeline Optimization. In: G. Squillero, P. Burelli (eds.) Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30–April 1, 2016, Proceedings, Part I, pp. 123–137. Springer International Publishing (2016)

28. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
29. Ronald, E.M., Sipper, M., Capcarrère, M.S.: Design, observation, surprise! A test of emergence. Artificial Life **5**(3), 225–239 (1999)
30. de Sá, A.G., Pinto, W.J.G., Oliveira, L.O.V., Pappa, G.L.: RECIPE: A Grammar-Based Framework for Automatically Evolving Classification Pipelines. In: European Conference on Genetic Programming, pp. 246–261. Springer (2017)
31. Silva, S., Muñoz, L., Trujillo, L., Ingalalli, V., Castelli, M., Vanneschi, L.: Multiclass classification through multidimensional clustering. In: Genetic Programming Theory and Practice XIII, pp. 219–239. Springer (2016)
32. Sipper, M., Fu, W., Ahuja, K., Moore, J.H.: Investigating the parameter space of evolutionary algorithms (2017). arXiv:1706.04119
33. Sohn, A., Olson, R.S., Moore, J.H.: Toward the automated analysis of complex diseases in genome-wide association studies using genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '17, pp. 489–496. ACM, New York, NY, USA (2017)
34. Vanneschi, L., Archetti, F., Castelli, M., Giordani, I.: Classification of oncologic data with genetic programming. Journal of Artificial Evolution and Applications p. 6 (2009)
35. Zutty, J., Long, D., Adams, H., Bennett, G., Baxter, C.: Multiple objective vector-based genetic programming using human-derived primitives. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 1127–1134. ACM (2015)