Whither AutoML? Understanding the Role of Automation in Machine Learning Workflows

Doris Xin*
University of California, Berkeley
dorx@berkeley.edu

Eva Yiwei Wu*†
University of California, Berkeley
University of Zurich
eva.wu@berkeley.edu

Doris Jung-Lin Lee
University of California, Berkeley
dorislee@berkeley.edu

Niloufar Salehi
University of California, Berkeley
nsalehi@berkeley.edu

Aditya Parameswaran
University of California, Berkeley
adityagp@berkeley.edu

ABSTRACT

Efforts to make machine learning more widely accessible have led to a rapid increase in Auto-ML tools that aim to automate the process of training and deploying machine learning. To understand how Auto-ML tools are used in practice today, we performed a qualitative study with participants ranging from novice hobbyists to industry researchers who use Auto-ML tools. We present insights into the benefits and deficiencies of existing tools, as well as the respective roles of the human and automation in ML workflows. Finally, we discuss design implications for the future of Auto-ML tool development. We argue that instead of full automation being the ultimate goal of Auto-ML, designers of these tools should focus on supporting a partnership between the user and the Auto-ML tool. This means that a range of Auto-ML tools will need to be developed to support varying user goals such as simplicity, reproducibility, and reliability.

CCS CONCEPTS

• **Human-centered computing** → **Empirical studies in HCI**; • **Computing methodologies** → *Machine learning*.

ACM Reference Format:
Doris Xin, Eva Yiwei Wu, Doris Jung-Lin Lee, Niloufar Salehi, and Aditya Parameswaran. 2021. Whither AutoML? Understanding the Role of Automation in Machine Learning Workflows. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3411764.3445306

1 INTRODUCTION

Machine Learning (ML) holds great promise in providing solutions to some of our most pressing societal problems, ranging from public health, to climate change, to precision agriculture and food waste management. At the same time, ML is perceived to be a dark art, accessible only to a few, involving a cumbersome iterative process of trial-and-error to craft a model with desired accuracy. Users often have to experiment with a large set of modeling decisions, spanning data pre-processing and feature engineering, model types, and hyperparameters, and empirically evaluate the performance of resulting models.

In recent years, automated machine learning, or Auto-ML[1], a new field targeted at increasing automation during ML, has witnessed a rapid rise in popularity, driving an explosion in self-proclaimed Auto-ML tools. Auto-ML holds the promise to make ML more easily accessible to users to employ for new domains and to reduce cumbersome manual effort when applying ML to existing ones. Auto-ML was initially introduced to automate model hyperparameter search. Over time, Auto-ML has evolved beyond that to include, as part of its goal, automation for other tasks in the ML workflow such as feature engineering, data cleaning, model interpretability, and model deployment. The ultimate promise of Auto-ML tools is to make ML more accessible by providing off-the-shelf solutions for users with less technical backgrounds.

In order for Auto-ML tools to effectively meet user needs, we must first understand what those needs are and what roles automation currently plays in the ML workflow. Towards this goal, we conducted an in-depth, qualitative study of sixteen Auto-ML users. We conducted semi-structured interviews with current users of Auto-ML, ranging from hobbyists to industry researchers across a diverse set of domains, to take a careful look at their use cases and work practices, as well as their needs and wants. We asked about their uses of ML, their experiences working with and without Auto-ML tools, and their perceptions of Auto-ML tools. Our approach enabled us to not only gain a better understanding of user needs with respect to Auto-ML, and the strengths and weaknesses of existing Auto-ML tools, but also to gain insights into the respective roles of the human developer and automation in ML development.

Our findings indicate that human developers are indispensable in ML development. Not only do humans excel at bringing in valuable domain knowledge, human intuition can also be incredibly effective in filling in the blind-spots in Auto-ML. Furthermore, human involvement is crucial for the effective and socially-responsible use

*Equal Contribution
†The majority of the work was performed when the author was at University of California, Berkeley

CHI '21, May 8–13, 2021, Yokohama, Japan

ACM ISBN 978-1-4503-8096-6/21/05.
https://doi.org/10.1145/3411764.3445306

[1]Stylized as Auto-ML to avoid confusion with Google AutoML.

of ML in real-world applications. Therefore, rather than attempting to automate human developers "out of the loop" as has been the objective of many Auto-ML tool builders, we advocate for a symbiotic relationship between the human developer and Auto-ML to integrate the human into the loop in the most productive manner. We, in fact, advocate that the moniker "Auto-ML" be discarded, because our evidence suggests that complete automation is infeasible; instead, these tools can be better thought of as offering mixed-initiative ML solutions.

Several prior studies have advocated for a human-guided approach to Auto-ML [19, 28, 42] and proposed design requirements. Rather than designing top-down, we build our case for human-AI integration based on bottom-up findings of real work practices, allowing us to arrive at unique, specific insights that are difficult to develop without taking the perspectives of the practitioner into account. We make concrete recommendations on what functionalities tool developers should enhance while preserving existing benefits, and, more importantly, what roles of humans they should preserve rather than attempt to replace. Our recommendations are based on both the perception and usage of a broad range of users of state-of-the-art Auto-ML tools. We hope that these insights and design recommendations can guide future development of Auto-ML tools to expand the current focus on system and model performance to emphasize human agency and control.

The rest of the paper is structured as follows: we discuss the relationships between our work and relevant HCI contributions and present state-of-the-art on Auto-ML tooling in Section 2; we describe our study methodology in Section 3; we present findings on the benefits and deficiencies of current Auto-ML tools as well as the respective roles of the human developer and automation in the ML workflow in Section 4; we discuss the design implications of our findings on Auto-ML tools in Section 5 and conclude in Section 6.

2 RELATED WORK

Auto-ML systems are often aimed at developing an end-to-end ML workflow or model, unlike other human-in-the-loop ML tools that are more focused on specific parts of the ML workflow, such as collection of training examples, model debugging, and model interpretation. Our work builds on multiple areas of prior work: existing Auto-ML systems, human-centered ML work practices, and human-in-the-loop ML tools.

2.1 Auto-ML Systems

The current landscape of Auto-ML tools is fast-growing and diverse. Existing Auto-ML offerings can be categorized into three groups: open-source software, hosted commercial solutions offered by cloud providers, and enterprise solutions offered by companies dedicated to developing Auto-ML platforms. Cloud-hosted Auto-ML solutions exist as part of a larger ecosystem of tools in the cloud, whereas enterprise solutions are standalone and therefore must either provide end-to-end support or integrate with external tools, resulting in appreciable differences in the user experience.

A typical ML workflow can be partitioned into three stages: *data preprocessing*, *modeling*, and *post processing*, and Auto-ML solutions provide varying levels of support for each of these stages. Below we offer a brief overview of tools in these three categories and comment on their support for the three stages in the ML workflow.

Open-Source Software (OSS). Auto-ML tools in this category include libraries such as Auto-sklearn (based on Scikit-learn) [17], TPOT (based on Scikit-learn) [26], and AutoKeras (based on Keras) [24], all developed in academic labs, as well as TransmogrifAI (based on Apache Spark) [39], AdaNet (based on Tensorflow) [11], Ludwig [32], and H2O [21] from industry. Of these tools, AutoKeras, AdaNet, and Ludwig are designed specifically for deep learning, tackling issues such as efficient neural architecture search, while the others are designed for traditional ML. Of the three categories of Auto-ML tools, OSS tools are the best at keeping up with cutting-edge ML research since many of the OSS tool developers are also involved in ML research. Overall, users of these libraries are afforded great flexibility since they can easily integrate custom code with the Auto-ML API, but they must provision their own computation resources. For the specific stages in the ML workflow, the Scikit-learn-based libraries are better suited for structured data and have better support for automated data preprocessing than the other libraries, while the Tensorflow and Keras-based libraries can support more complex models involving text and images. OSS tools are generally lacking on post-processing support, which involves evaluation, deployment, and monitoring of models.

Cloud Provider Solutions. The major players in this category are Google Cloud AutoML, Microsoft Azure Automated ML, and Amazon SageMaker Autopilot [3, 20, 31]. Solutions in this category differ from the previous category in three significant ways: 1) Since they are hosted, compute resources are provided and managed by the cloud provider, and users pay proportional to the amount of compute consumed during the process of Auto-ML; 2) They tend to be much more end-to-end and include model evaluation and deployment to help users derive business value from the models trained; 3) Some system internals are opaque to the user, who can only interact with the system at specific decision points. Overall, while cloud-hosted solutions tend to require less programming expertise to use, they are also *less configurable and transparent*. For example, Google Cloud AutoML, which boasts "more than 10 years of proprietary Google Research technology to help your ML models achieve faster performance and more accurate predictions", neither allows users to specify the type of models nor provides visibility into the model internals. Amazon SageMaker Autopilot, on the other hand, places a strong emphasis on visibility and control by allowing users to easily export the Auto-ML code and intermediate results into computational notebooks. All three providers offer no-code UIs for non-programming users, in conjunction with Python APIs. While Microsoft and Amazon enable additional customizability through their Python APIs, Google's APIs are solely designed for programmatic compatibility and offer no additional control or transparency. When it comes to model evaluation, Microsoft and Amazon provide summaries of the models explored while Google offers only high-level information about the final model.

Auto-ML Platforms. As self-proclaimed "platforms", tools in this category tend to position themselves as turnkey, end-to-end Auto-ML solutions. They manage compute resource provisioning by integrating with either cloud providers or on-premise hardware infrastructure. Major players in this category include DataRobot [12]

and H20 Driverless AI [21]. Compared to their cloud provider counterparts, solutions in this category tend to be more feature-complete, providing more technical support and customizability in each stage of the workflow. Since these solutions target business users, special attention is paid to the operationalization of the resulting model, including an expansive set of model interpretability and deployment options. Additionally, to address increasing concerns around data privacy and security, these solutions allow on-premise deployment instead of forcing users to migrate their data to the cloud.

Thus far, there has been little evaluation on how these three types of Auto-ML tools are used in practice. In this paper, we sought to understand the adoption of Auto-ML tools, their usage, and the current bottlenecks that users face with these tools. Our eventual goal is to identify design requirements and considerations for more effective collaboration between the human developer and ML/AI.

2.2 Human-Centered ML Work Practices

HCI research in Auto-ML has proposed a human-guided approach to ML [19, 28, 42], with the aim of balancing the trade-off between user control and automation. Human-guided ML builds on the premise that ML development should be a collaborative activity between human users (i.e., data scientists or model developers) and the machine, wherein users specify their domain knowledge or desirable model behavior at a high-level and the system performs some form of automated search to generate an appropriate ML pipeline or model [19, 28]. This work is related to a larger body of studies on ML work practices [4, 22, 23, 25, 48] of specific groups of practitioners, including software engineers using ML [4], non-expert ML users [48], as well as specific aspects of work practices, such as model interpretation [23] and iterative behavior in ML development [22, 27].

Existing work has explored the use of visualization to help users gain insights into the black-box process of Auto-ML and obtain higher control during Auto-ML [43, 44]. Another crucial step towards human-guided ML requires understanding the perspective of data science practitioners' and their attitudes towards Auto-ML systems [42]. In particular, Drozdal et al. study the perceptions of transparency and trust in Auto-ML and identify components within Auto-ML tools most likely to improve trust [14]. We expand upon this work by studying the role of humans and Auto-ML in the end-to-end ML workflow, as well as additional important user requirements, such as customizability, completeness, ease-of-use, efficiency, effectiveness, and generalizability.

These prior studies have largely focused on participants without Auto-ML experience. As a result, most of these studies elicited user feedback regarding Auto-ML by demoing new Auto-ML prototypes to study subjects. These prototypes have limited features, largely supporting only the modeling stage of the ML process. For example, the study by Drozdal et al. involves university students and uses hypothetical tasks to determine the relative value of components of Auto-ML in communicating trust, rather than centering on the real-world experiences of Auto-ML tool users. Our paper builds on this line of work by studying users who have worked with present-day Auto-ML tools in real-world applications to investigate how Auto-ML fits into their end-to-end ML workflow, as well as its limitations. Our methodology and choice of participants afford novel findings beyond existing studies on the matter of trust and transparency, such as the broader social context surrounding the use of Auto-ML in the real world, e.g., getting others in the organization to adopt the results from an Auto-ML model, or Auto-ML playing a role in reproducibility and institutional knowledge should the original user leave the organization. Moreover, we identify various ways hands-on human agency and control promotes trust—something left unaddressed by studies involving hypothetical Auto-ML interfaces.

2.3 Human-in-the-loop ML Tools

The idea of incorporating human knowledge into ML workflow development has been well studied in research on interactive tools for ML model interpretation. Interactive machine learning (IML) is a paradigm wherein human users train an ML model by manually evaluating and correcting the model result through a tight interactive feedback loop [5, 16, 18, 36]. IML often focuses on collecting user input for labels for training data in order to train and correct ML classification models. IML lowers the barrier to interacting with ML-powered systems by empowering end-users without ML expertise to interactively provide exemplar feedback to the system [5, 48] (e.g., recommendation systems can learn from the relevance feedback of approving/rejecting an item), often eliminating the need for the highly-technical feature engineering step [16]. In addition to IML, human-in-the-loop systems have also been developed for model debugging and verification [6, 33, 35], iterative model development [47], and model interpretation [2, 38, 46] often through an interactive environment and/or visualizations. We study the impact of such capabilities as reflected in present-day Auto-ML tools.

Prior work questions the validity of humans-in-the-loop [40]. However, the study experiments only examined the effects of humans as the final arbitrators, anchoring on ML recommendations. Our study shows that humans are in the loop throughout the ML workflow even as Auto-ML attempts to automate away ML development. Along the path, humans can iteratively influence the directions of ML outputs via several touch points. Further research is necessary to understand the effects of mixing ML and humans in real-world settings.

3 STUDY DESIGN

Our paper seeks to understand how users incorporate Auto-ML tools in their existing workflows and their perceptions of the tools.

3.1 Recruitment

Since, to the best of our knowledge, no prior work has focused on participants with real-world Auto-ML experience, we focused on users who have applied Auto-ML to real-world use cases across a broad set of application domains. We recruited participants by posting recruitment messages to relevant mailing lists and Slack channels (N=5), through personal connections (N=9), and by reaching out to users on social media who mentioned or were mentioned in posts about Auto-ML (N=2).

We invited participants to fill out a screening survey that asked whether they had experience using Auto-ML tools. Select participants either had experience using at least one of the tools that were listed in the survey (the list includes tools discussed in Section 2.1) or had used other tools that our team verified to be Auto-ML tools.

The tool-based eligibility criteria was motivated by our pilot recruitment strategy where a large number of survey respondents expressed that they had general ML experience but were unfamiliar with Auto-ML, mistaking manual ML for "automated ML".

3.2 Participants

We interviewed a total of 16 participants who had prior experience applying Auto-ML in professional capacities. Information about each participant is shown in Table 1. Of the 16 participants, 14 were male (87.5%) and 2 were female (12.5%). Recruiting participants past March 2020 was difficult due to COVID-19, but we gathered a sample of users that spanned a diverse set of organizations and use cases.

Participants had, on average, 10 years of experience in programming and an average of 5 years of experience with ML. Participants spanned across three continents and a diverse set of job roles and industries, from a product manager at a large retail corporation, to a director of commercial data science at a travel technology company, to academic researchers at universities. The tools that the participants used also varied from proprietary tools, to open-source software, to commercial solutions. To preserve participant anonymity, we omit the identities of the specific tools used, as some tool-application combinations can reveal the identity of the participant.

3.3 Interview Procedure

We conducted semi-structured interviews with participants about their experience in using Auto-ML for real-world applications. Interviews were conducted from October 2019 to March 2020, either in person (N =2) or remotely (N=14). Each interview lasted for approximately one hour. Every participant received a $15 gift card for compensation. The interviews were largely semi-structured and involved three main components:

- Participants were asked to describe their job role, organization, and ML use cases.
- We asked the participants about their experience in developing ML workflows (without Auto-ML) and the challenges they faced.
- We asked the participants about their experience using Auto-ML tools, including the features of the specific tool used and how they integrated Auto-ML in their ML workflows, and their perceptions of Auto-ML tools in general, including customizability, effectiveness, interpretability, and transparency.

The interview guide can be found in the supplementary materials.

3.4 Study Analysis

We audio-recorded and transcribed all but one interview, where the participant did not consent to being recorded. After completing all the interviews, we engaged in an iterative and collaborative process of inductive coding to extract common themes that repeatedly arose in our data. Each of the first three co-authors independently coded the data using Dedoose [13], an online tool for open coding, to map data onto these categories. The first three authors met weekly and discussed themes and concepts to clarify ambiguity in the codes and established consensus in a code book. Afterwards, we conducted a categorization exercise, wherein some of our initial categories included advantages and disadvantages of Auto-ML, perceptions of Auto-ML, workflow strategies with and without Auto-ML, Auto-ML desiderata, general ML challenges, and Auto-ML adoption decisions. We used codes to facilitate the process of theory development and refrained from calculating inter-rater reliability to avoid potential marginalization of perspectives [29].

3.5 Limitations

Our interviews were structured around the participants' experience with a single ML task. While this practice afforded concreteness to the discussion, we may have missed out on diversity of use cases. The interviews focused on work practices around Auto-ML, which allow us to gain a deep understanding of the strategies participants employed to integrate Auto-ML into their data science workflow, but limited our time in studying any particular conceptual perception (i.e. transparency, interpretability) of Auto-ML in-depth.

We acknowledge that while our participant population comes from diverse background, it may not be a representative sample of the overall Auto-ML user base. For example, the proportion of women in our sample (14 male, 2 female) is slightly lower than the gender ratio (15% women) in the data science profession referenced in the 2020 BCG report [15].

4 RESULTS

In this section, we present our findings from the interviews. This section is organized as follows: we describe how we group the users and use cases into high-level categories in Section 4.1; we enumerate common tasks in ML workflows reported by the participants in Section 4.2; we present the benefits of Auto-ML perceived by the participants in Section 4.3, the deficiencies of existing Auto-ML tools that we believe can be addressed by system and UI improvements in Section 4.4, and the roles of the human developer that the Auto-ML tools must respect and preserve in Section 4.5. Note that while the functionalities presented in Section 4.4 and Section 4.5 are both absent or lacking in existing Auto-ML tools, the deficiencies as described in Section 4.4 have near-term solutions using existing techniques (we provide many such suggestions in Section 5), whereas the roles in Section 4.5 may one day become within reach of Auto-ML tools through fundamental breakthroughs in knowledge representation and programming paradigms. We believe that assuming the roles in Section 4.5 should not be the objective of Auto-ML tool developers until then.

4.1 User and Use Cases Segmentation

A given participant's ML skill-set and use cases can potentially influence how they use and perceive Auto-ML tools. We, therefore, wanted to study the relationship between users' expertise and their behavior around and perception of Auto-ML. To facilitate the discussion on how contextual information serves to explain user behavior and sentiment, we categorize the participants and their use cases as follows.

4.1.1 User Skill Levels.

We group participants based on their past experience with ML into the following three categories.

ML Innovators. Participants in this group have formally conducted ML research, either on fundamental algorithms (P4, P11, P14,

Table 1: Interviewee demographics, from the left: (1) Participant ID, 2) Participant's years of experience (Yrs of Exp) in programming and in ML, (3) Domain of their companies and organization size in parentheses with "(S)" corresponding to small, "(M)" to medium, and "(L)" to large, (4) Participant job title, (5) Auto-ML application, (6) Type of Auto-ML tool(s) used by the participant. "(CP)" indicates that the commercial solution is provided by a major cloud provider and (PF) indicates that the commercial solution belongs to the Auto-ML platform category introduced in Section 2.1.

PID	Yrs of Exp (programming/ML)	Organization Domain (Organization Size)	Participant Role	Application	Auto-ML Tool Category
P1	30/5	Finance (M)	Data Science	Financial credit scoring	Commercial (PF)
P2	8/2	Healthcare (S)	Data Science	Object identification in videos	Commercial (CP)
P3	4/7	Retail (L)	Product Management	Fraud detection in e-commerce transactions	Commercial (PF)
P4	11/6	Social Network (L)	ML Engineering	Content curation and recommendation	Proprietary
P5	10/3	University (L)	Biomedical Research	Drug response prediction	Open Source
P6	10/4	Healthcare (S)	Data Science	Medical claims analysis for healthcare program referral	Commercial (PF)
P7	12/2.5	Finance (L)	Data Science	Fraud detection in financial transactions	Commercial (CP)
P8	5/4	University (L)	Neuroscience Research	Diagnostics and phenotype analysis with brain scan images	Open Source
P9	8/8	ML Software (S)	CTO	Enterprise document processing automation	Commercial (CP)
P10	20/8	Travel Technology (M)	Data Science	Personalized travel recommendations	Commercial (PF)
P11	6/5	Information System (S)	ML Research	Computer vision for brand logo detection	Open Source
P12	16/12.5	Educational Software (M)	Data Science	Content identification in educational materials	Commercial (CP)
P13	6/3	Electronic Product Manufacturing (L)	Research	Gesture recognition in sensor data from edge devices	Commercial (CP)
P14	12/5	Consulting (S)	Software Engineer	Nonprofit AI consulting for businesses	Open Source
P15	4/3	Pharmaceutical (L)	ML Researcher	Drug response prediction	Open Source
P16	2/1.5	Supply Chain (L)	Program Management	Supply chain procurement and risk-management	Proprietary

P15) or the application of ML for scientific discoveries (P5, P8). They command deep understanding of the mathematical underpinnings of the ML models they use.

ML Engineers. Participants in this group are skilled and experienced ML practitioners with formal training in applied ML (P1, P2, P3, P6, P7, P10, P12, P13). They are intimately familiar with popular ML models as well as ML libraries and tools.

Novices. Participants in this group (P9, P16) have no formal training in ML. They are domain experts in non-ML fields.

4.1.2 Use Case Categories.

Production Applications. Use cases in this category pertain to developing ML models that drive applications or decision making with significant impact. Examples include training models to assist in financial service decisions, building recommendation systems for content curation, and fraud detection. P1, P3, P4, P7, P12, and P16 have use cases in this category.

Prototype. Use cases in this category involve prototyping ML models for industry applications, which are lower stake than the production application scenario since the model performance has no direct impact on business metrics or end-user experience. P2, P6, P9, P10, P11, P13, P14, and P15 have use cases in this category.

Research. Use cases in this category involve building ML modelings for academic research, where model results are shared and examined in detail. P5 and P8 build ML models for research.

Figure 1 shows the cross tabulation of the Auto-ML tools category with use case categories and user expertise, the level of support for each of the three ML workflow stages based on features offered by each category of tools, and the average Likert scores that our participants gave to their Auto-ML tools. The support ratings for "proprietary" tools are omitted as we do not have full visibility into the these tools. While Likert scores are commonly treated as ordinal data, they are treated as interval data in our setting due to the natural correspondence between the scores and quintiles. Thus, the average Likert scores are sound and meaningful.

	Prod. Apps	Prototype	Research	Novice	ML Engineers	ML Innovators	Data Preproc.	Modeling	Post Proc.	Transparency	Interpretability	Customizability	Generalizability	Completeness	Ease of Use	Effectiveness	Efficiency
Cloud Provider	2	3		1	4		★★	★★	★★	2.2	3	3	3.2	3	4.4	3	3.6
Auto-ML Platform	2	2			4		★★★	★★★	★★★	3.5	4.5	3.5	3.75	4	5	4.25	4.5
OSS		3	2			5	★★	★★★	★	3.6	4.2	4.2	4.4	3.6	3.8	4.2	4
Proprietary	2			1		1	N/A	N/A	N/A	2	3.5	3	4.5	2.5	4.5	4.5	4.5
									Avg.	2.825	3.8	3.425	3.9625	3.275	4.425	3.9875	4.15

Figure 1: Tools: Characterization of Auto-ML tools used by participants by category. The tool categories are the same as the ones presented in Section 2.1, with the addition of "proprietary" for anonymous in-house Auto-ML solutions. The first three columns contain the cross tabulation of tool categories and use case categories. The next three columns in blue contain the cross tabulation of the tool categories and user expertise levels. The following three orange columns indicate the level of support for each of the three ML workflow stages based on features offered (not based on interview results). The next eight columns contain the average Likert scores provided by our participants. Cells in green are above average.

4.2 Data Pre-processing, Modeling, and Post Processing Tasks

In this section, we focus on how practitioners integrate Auto-ML tools into the end-to-end ML workflow. As mentioned in Section 2.1, an ML workflow is commonly partitioned into three stages: data pre-processing, modeling, and post-processing. In this section, we report the common patterns of how Auto-ML fits into the each stage. Figure 2 shows a complete set of tasks the participants reported for each stage of the ML workflow. We characterize these three stages as follows:

(1) In the data pre-processing stage, users prepare the data for ML, performing tasks including data acquisition, cleaning, transformation, labeling, and feature engineering.
(2) In the modeling stage, models are trained on the data prepared in the pre-processing stage.
(3) After models are trained, the post-processing stage span a broad range of activities, including model evaluation, interpretation, deployment, and user studies.

4.2.1 Data Pre-processing. Overall, data pre-processing is a time consuming and primarily manual process for most participants. On average, participants reported spending roughly 50% of their time on data pre-processing, and roughly 80% of all data pre-processing tasks were performed manually. While some participants expressed the desire for Auto-ML to provide more automated support for data pre-processing, others believe that data pre-processing relies on human intuition and knowledge that is impossible or at least extremely challenging to codify and therefore cannot be automated. In either case, participants agree that data pre-processing is a crucial component in the workflow due to the adage "garbage in, garbage out." We defer discussions on desired but missing automated data pre-processing features and the role(s) of the human to Section 4.4 and Section 4.5, respectively. Here, we report on the common tasks in the data pre-processing stage and whether each task is currently carried out by Auto-ML or human developers.

Data collection. The first task in data pre-processing is *data collection* [37] for most participants except P3, P5, P6, and P15, who were provided and limited to specific data sources. Others had the freedom to incorporate organizational data assets obtained from relevant stakeholders (P4, P7, P8, P9, P10, P11, P12, P13, P14, P16) as well as open datasets (P1, P2) from the web, which are much less commonly used. A specific data collection task worth noting is *data labeling*, where significant human attention is required to annotate examples with the desired target label. P9 and P13 reported performing labeling themselves because it provided them with intuition about the data and allowed for fast turnaround. P11 and P12, both of whom worked with unstructured data, relied on crowdsourcing for data labeling. Data labeling is inherently manual, although there are recent developments in systems to lower the manual effort for data labeling [7, 9].

Data Wrangling & EDA. Following data collection is data wrangling, including data cleaning and missing data imputation, formatting, and transformation. Exploratory data analysis (EDA) is integral to all data wrangling tasks. All participants except P11, who worked with image data, reported that they spent time on data wrangling. In addition to Auto-ML tools used by P5, P8, P16 to help with data wrangling, the Python pandas [30] and numpy [41] libraries are the most popular choices for data wrangling, followed by SQL. Tools in the "Other" category in Figure 2 include domain specific tools, Spark, and proprietary tools.

Data cleaning, especially missing data identification and imputation, is a common task. While most Auto-ML tools handle missing data imputation, participants stated manual intervention is necessary to decide on the appropriate imputation technique based on context (P1, P4, P5, P7). Participants reported that they spent significant manual effort to format the data for ingestion by the Auto-ML tool (P2, P3, P8, P10, and P12.) Specialized data wrangling tasks included data anonymization (P3), time series test set generation (P10), and resampling for class imabalance (P16).

Feature engineering. Feature engineering and feature selection are among the most automated data pre-processing tasks. Most Auto-ML tools are capable of both simple feature engineering such as one-hot encoding [45] and building complex features involving multiple input signals. However, over half the participants who performed feature engineering manually because they felt existing Auto-ML solutions are incomplete or inefficient (P6, P10, P14) or they believed that human intuition and domain knowledge could not be replaced by automation (P3, P4, P12). Interestingly, P5 and

		P6	P10	P1	P3	P7	P12	P2	P13	P9	P16	P4	P5	P8	P11	P14	P15
	Tool Type	PF	PF	PF	PF	CP	CP	CP	CP	CP	P	P	OSS	OSS	OSS	OSS	OSS
	Expertise	E	E	E	E	E	E	E	E	N	N	I	I	I	I	I	I
	Use Case	T	T	PA	PA	PA	PA	T	T	T	PA	PA	R	R	T	T	T
Data Preprocessing Tasks	Data Collection		M	M		M	M	M	M	M	M	M		M	M	M	
	DC: Data Labeling						M		M	M					M		
	Data Wrangling & EDA	M	M	M	M	M	M	M	M	M	M,A	M	M,A	M,A		M	M
	Feature engineering	M	M,A	A	M,A		M					M	A	A		M	
Data Preprocessing Tools	Python (pandas, numpy)																
	Python (ML libraries)																
	SQL																
	Other PL (R, C++/#)																
	Visualization tools																
	Other																
Modeling Tasks	Hyperparameter Tuning			A	A	A	A	A	A	A		A	A			A	A
	HT: Neural Architecture Search						A			A					A	A	
	Model Selection	A		A	A	A		A			A	A	A				
	Feature Selection		A		A				M			M	A	A			
	Inform Manual Development		A					A	A	A	A			A			A
	Business Req. to ML Objectives	M	M	M							M						
	Feature Engineering												A			A	
Modeling Tools	Custom Python/R																
	Scikit-Learn																
	Pytorch/Keras/Tensorflow																
Post Processing Tasks	Generate Reports	M	A	M,A	A	M	A	M	M	M	A	M	M	M	M	M	M
	Present to Stakeholders			M	M	M							M		M	M	M
	Model Export/Deployment	M	A		A		A	A	A	M	A	M		M		M	
	Fine tune/Validate predictions		M	A		A	M	M	M		M		M		M		
	Model interpretation	M	M	A	M								M				
	Monitoring	M	A		A							M					
	Model Management				A			A				M					
	Other		M						M								M
Post processing tools	Visualization Tools																
	Python (pandas, NumPy)																
	Deployment (Flask, Algorithmia)																
	Other (proprietary, W&B)																

Figure 2: Use cases: tasks performed and tools used by participants in each stage of the ML workflow. The three rows at the top contain metadata about each participants. "Tool type" corresponds to the tool types in Figure 1: PF = Auto-ML platform, CP = Cloud Provider, and P = Proprietary. "Use case" corresponds to the use case categories introduced in Section 4.1: T = Prototype, PA = Production Application, R = Research. "Expertise" corresponds to the user skill levels introduced in Section 4.1: E = ML Engineers, N = novices, I = ML Innovators. In each task cell, "M" indicates that the participant performed the task manually while "A" indicate automation. A green tool cell indicates that the participant has used the tool for the given ML workflow stage. Columns as ordered to cluster participants who use the same type of tools and have the same expertise levels.

P14 both reported repurposing, or *"misusing"* (P14) modeling capabilities for feature engineering.

4.2.2 Modeling. As one would expect, the most common tasks that participants used Auto-ML for during modeling are *hyperparameter tuning* and *model selection*. *Feature selection* is generally carried out by the Auto-ML tool as a byproduct of model training. Participants who perform feature selection all use Auto-ML for feature selection except P4, who manually selects features before model training because they work with petabyte-scale data, and P13, who deploy models to edge devices with limited memory.

An interesting use case for Auto-ML modeling is to *inform manual development* (P2, P8, P9, P10, P11, P13, P15.) For example, P13 reported that they use the Auto-ML tool as a quick check for data quality and to validate manual data pre-processing. They would perform data wrangling manually if the Auto-ML tool identified anomalies in the data. Auto-ML also empowers *exploration of unfamiliar models and hyperparameters* (P2, P8, P10, P13, P15, and P16.) Auto-ML results are often used as a *benchmark for validating manual model performance* (P4, P5, P9.) Many participants reported that

they performed the same modeling tasks manually alongside Auto-ML to understand and verify the Auto-ML results and to correct any errors made by Auto-ML.

The modeling tools in Figure 2 refer to tools for manual modeling.

4.2.3 Post-processing. Post-processing spans a large variety of tasks depending on the participants' use cases. Visualization is an integral part of many of the tasks below.

Generating reports and sharing results. The most common post-processing task is to generate a report of the model results and relevant model search history. Most platform tools automatically generate such reports, in the form of summary statistics, leaderboards, and other visualizations. P10 enthusiastically shared that their platform tool was able to auto-generate documentation for legal compliance thereby greatly reducing the manual overhead for governance. While all of the cloud-hosted Auto-ML tools also auto-generate reports and visualizations, it is interesting that many participants adopted manual approaches to modify the default reports. In addition to generating reports for their own consumption, a subset of the participants (P1, P3, P5, P7, P11, P14, P15) who had to share and explain their results to other stakeholders needed to present their findings in text documents or slides with human readable explanations.

Deploying Models. Model export and deployment is the second most common post-processing task. Participants who did not perform model deployment were either using the models to inform human decision making (P1), handed off the model to a separate Dev Ops team for deployment (P7, P11) or used model results solely for research findings (P5, P15). Automated deployment was only afforded to users of hosted Auto-ML tools. The reason for manual deployment for participants who used hosted tools include 1) the model for a financial application needed to be vetting for security (P7), 2) the Auto-ML tool did not support automated deployment (P6), 3) the application relied on complex logic to incorporate the model output (P9), and 4) the model directly impacted end-user experienced and required staggered roll-out with human supervision (P4). Tools for deployment included Flask, Algorithmia, custom Python, and proprietary infrastructure.

Validating and Interpreting Models and Predictions. The two main techniques for building trust in and understanding Auto-ML outputs are point queries on the predictions, feature importance, and holistic visualization of high-level model characteristics. Most hosted solutions focus on supporting point queries and feature importance via visualizations. Manual efforts in this category were mainly for 1) cross validating with manual modeling results (P3, P10, P11), 2) application domain specific checks (P6, P12), 3) field testing specific predictions (P13, P16), 4) custom feature importance computation (P5).

Miscellaneous. Less common, nevertheless noteworthy tasks included model versioning (P2, P3, P4), on-device user studies (P13, P15), debugging manual implementation (P15), and most interestingly, converting an ML model into a set of if-else statements for more predictable and interpretable inference (P10).

4.3 Benefits of Auto-ML

In this section, we present findings on the major benefits of Auto-ML from our interview study. As shown in Figure 1, ease of use, effectiveness, and efficiency are the highest-rated qualities of Auto-ML tools by the interview participants. We present specific benefits reported by the participants that corroborate these ratings below.

Enables and empowers novices. To ML novices, the greatest benefit of Auto-ML is that it enables business users to nimbly use ML to inform business decisions without "*a massive engagement with multiple consultants in multiple different teams*"(P16). P16 believes that Auto-ML leads to "*the democratization of advanced analytics throughout business units for people that don't have experience doing that kind of work*", and this sentiment is echoed by P10:

> "It allows for ... citizen data science to become a reality with the proper governance controls and proper management in place. At the bank I worked at previously ... Auto-ML was becoming adopted ... for robotic process automation. ... Anyone who's analytically competent ... starts rolling with it immediately."

However, Auto-ML can be a double-edged sword for novice users—users who treat it as largely a blackbox find Auto-ML to be a great enabler, while curious users who attempt to look inside the blackbox can become distracted and suffer from choice paralysis. P9 reported that while they were able to achieve a few percentage points in model performance improvement, Auto-ML *increased* their development time, due to the fact that the Auto-ML tool exposed them to a large number of new model types that led to many lengthy manual explorations out of curiosity.

In addition to avoiding distractions, treating Auto-ML as a blackbox also leads to standardization.

Standardizes the ML workflow for better reproducibility, code maintainability, knowledge sharing. Another benefit of the blackbox nature of Auto-ML tools is that by having a predetermined search space that doesn't change, there is more standardization of the ML development process, leading to better comparisons across models, code maintainability, and effortless knowledge transfer. The need to search through a large number of model types, which are often implemented in different libraries, has prompted Auto-ML tools to create a standardized abstraction of the ML workflow decoupled from specific APIs and model types. As a result,

- different models are easily comparable using standardized, normalized metrics (P3),
- the amount of code needed to implement different models is greatly reduced (P3),
- models are more reproducible (P15, P3),
- latest ML research can be easily incorporated into existing workflows (P3, P13),
- model training requires less human intervention as there are fewer errors (P2)

For hosted Auto-ML solutions that generate extensive reports on the end-to-end process, Auto-ML serves as a self-documenting, reproducible knowledge sharing platform (P10, P3). This is especially beneficial in industry, as P3 points out:

> "Data scientists are expensive and very in demand, and people leave the job a lot and, and the algorithms change all the time. If I quit my job today ... I could be like, here's all the history."

Prevents suboptimal results due to idiosyncrasies of ML Innovators. A unique benefit of Auto-ML applicable only to ML

Innovators is its ability to prevent suboptimal model performance resulting from idiosyncratic practices by ML Innovators with extensive ML experience. P5 relayed instances where the Auto-ML tool found models that they never would have tried manually but outperformed the conventional choices they made. They therefore dubbed Auto-ML the "no assumptions approach" to ML development. P8 preferred the fact that the only factor driving the decisions made by the Auto-ML tool is "the statistical properties of the data" and not their own familiarity of specific model types, thus removing "bias" from the process.

Builds models more effectively and efficiently. The ability to *build better models faster* is a major benefit of Auto-ML tools reported by many participants, especially ML Engineers. Most participants reported that Auto-ML led to significant improvements in *efficiency*, the time for developing models, and a moderate increase in *effectiveness*, the performance of the final model obtained. On a five-point Likert scale, participants on average rated improvement in efficiency ($\mu = 4.1$) higher than improvement in effectiveness ($\mu = 3.88$). Some participants reported an order of magnitude reduction in development time (P10, P16). Improvements in effectiveness experienced by the participants were often incremental; thus, participants did not deem more accurate models in isolation as a compelling reason for Auto-ML adoption.

In addition to reduction in model development time, another dimension of efficiency is the ability to experiment with significantly more models in the same time it took for manual development. Even if the overall model performance does not change, exploring more models in itself provides downstream benefits. Many participants felt that they were much more productive because they were able to explore more models in the same amount of time. For P10, extensive experimentation also led to better insights into the features and models:

> "The team that built out the manual process were only able to review two or three different models. I was able to look at 50.
> I was able to report on more insights, more understanding of the variable inputs than they were in the same amount of time, more understanding around why the model performed the way that it did." (P10)

Enables rapid prototyping. Traditionally, incorporating ML into an application from scratch is a lengthy process involving many stakeholders. The substantial overhead of adoption, on top of the uncertainty about whether ML will improve the application behavior, deters many from ML adoption. Auto-ML has significantly lowered the barrier to entry by enabling users to build quick prototypes to gauge the feasibility and potential impact of ML, without the cumbersome process of setting up infrastructure and codebase (P9, P11) and full integration (P12, P13), especially in industry settings. The caveat is that for many industry users, Auto-ML is used for rapid prototyping only but not full development, due to a lack of confidence in its performance (P4, P13)

> "[I use Auto-ML] just to prototype and see how it works.
> I don't use it within the system within my industry job, but I use it as a prototype system there just to see how easy this task is." (P4)

> "I would use Auto-ML first to try things out, to understand. It would be a good starting point for a new application. I am convinced that it would generalize, but if I want a particular performance number, then I am less confident about how it would perform." (P13)

Fosters Learning.

Users who were able to inspect the search history of the Auto-ML tools reported that they learned about new modeling techniques (P8), implementation of specific ML algorithms (P9), model architecture (P11, P14), model performance on specific types of tasks (P10), and model resource consumption (P15). These learning opportunities emerged serendipitously, as the users were validating the predictions and interpreting the models. However, for P2 who specifically sought to learn from their Auto-ML tool T (tool name anonymized to preserve participant privacy), the lack of transparency greatly hindered learning:

> "Getting the model out of T proved to be extremely challenging. It was like 94% accuracy 94% precision. And when we tried that we didn't see anywhere close to that. We tried to actually open up the model and see how it actually structured it, which was extremely challenging. It took a couple of hours, and then we learned that their structure was something that [they have written a paper on], so it was extremely hard to use. ...It basically looks more like a black box."

Prior to their experience with T, P2 regarded T highly on account of the cutting edge ML algorithms that T claims to incorporate. However, due to their frustrations with the blackbox nature of the tool and the lack of offline reproducibility, P2 eventually abandoned Auto-ML and reverted back to manual ML development.

4.4 Deficiencies of Auto-ML

In this section, we present findings on the deficiencies of existing Auto-ML tools that can potentially be addressed via systems innovations. We discuss the design implications of our findings in Section 5. There are other limitations of Auto-ML tools that stem from the complex social and psychological implications of human-machine collaboration in ML workflows, and we discuss these limitations in Section 4.5.

Lacks comprehensive end-to-end support. Figure 1 shows that *completeness*, the extent to which Auto-ML covers the end-to-end ML workflow requirements, is the second lowest scoring rating. As evident in Figure 2, Auto-ML is currently used primarily for automating model training, requiring users to do the heavy lifting for both data pre-processing and post-processing using other tools. This reality directly contradicts some claims made by Auto-ML tool developers. In the Auto-ML platform category:

> "[DataRobot] supports all of the steps needed to prepare, build, deploy, monitor, and maintain powerful AI applications at enterprise scale. [It] even automates model deployment, monitoring, and management."
> "[H2O Driverless AI delivers] automatic feature engineering, model validation, model tuning, model selection and deployment, machine learning interpretability, bring your own recipe, time-series and automatic pipeline generation for model scoring."

Cloud and OSS solution developers are less aggressive in claiming end-to-end support, since OSS could rely on programmatic interoperability with other libraries, and cloud providers in theory could integrate with their other offerings for other stages of the ML workflow. While intended for flexibility, the interoperable design led to a *fragmented data ecosystem*, causing users to *"spend most of the time gluing everything together."* (P14) In practice, all participants who used cloud-hosted Auto-ML reported using it in isolation, necessitating significant manual effort for data ingress and egress. P12, user of a cloud-based Auto-ML tool *C* lamented:

> "The biggest challenge is . . . manipulating the data in a way that can be used with [*C*]. That is not something that we would have done if not for *C*. . . . For each project you'll have to spend considerable amount of time structuring the data in a way that can be fed into *C*." (P12)

This drawback places cloud solutions below OSS in participant-rated completeness, despite the fact that cloud solutions provide more built-in features for model deployment and interpretability.

Limited data pre-processing. In terms of functionalities, a common complaint across all tools categories is inadequate support for data wrangling. As P1 pointed out, data pre-processing support in Auto-ML tools is primarily for feature engineering, which is deemed satisfactory by many Auto-ML users, and does not cover data cleaning and wrangling needs. In fact, P1, P3, and P15 stated that their Auto-ML tools did not support data pre-processing even though they acknowledged the feature engineering functionalities of their tools, since most of their time is spent on data wrangling and not feature engineering. P6's choice of Auto-ML tool was determined entirely by the tool's ability to automate domain specific data wrangling.

Among the host of data wrangling tasks enumerated in Section 4.2.1, participants expressed the need for improved system support for the following tasks:

- automated data discovery (P1)
- data cleaning for domain specific data (P6) and with more user control (P7) to avoid "garbage in, garbage out"
- data transformation for domain specific data (P9, P10, P13) and for mitigating common data problems such as class imbalance (P16)
- large-scale data processing using distributed architecture (P14)
- dataset augmentation using state-of-the-art research (P11).

Limited Support for complex models and data types. Classification and regression are currently the only types of tasks supported by Auto-ML tools. While some tools are beginning to support unstructured data such as text and images by harnessing recent development in deep learning, tabular data remains the focus for most Auto-ML tools. The existing user-base for Auto-ML is self selected to fit into the capabilities of current offerings. Even so, many participants expressed the desire for more broad-ranging support, such as unsupervised learning (P3, P6, P10, P15, P16) and domain-specific data models (P6: healthcare, P10: time series).

Causes system failures due to compute intensive workloads. A common complaint about Auto-ML by participants who used OSS solutions is system performance, a major contributing factor to OSS being rated lower than the other categories of tools on *ease of use* in Figure 1. P14 reported that *"running out of main memory was the biggest technical challenge."* P8, whose models had *18 million columns*, also reported that running out of memory, which crashed model training, was a frequent frustration. P5 had to run experiments on limited computation resources provided to her research lab, and she had to modify the model search space to reduce the total run time:

> "[It was] time consuming for [Auto-ML] for large dataset, and some pipelines are just too heavy and crash the process. For large feature set and sample set, some operators to further expand the data set were [slow and had to be] removed from the search space." (P5)

P11, P14, and P15 also reported that they needed to define the model search space carefully to cope with the compute-intensive nature of Auto-ML workloads, and P15 would even revert back to manual development for large models. The need to switch between development on laptops and on servers posed a major *"annoyance"* for P11. P5, P8, P11, P14, and P15 all reported spending only a fraction of their time on model development, thus their heavy Auto-ML compute needs are intermittent.

Lacks customizability. Customizability is the third lowest rated quality of Auto-ML tools by the participants, as shown in Figure 1. Interestingly, both too little customizability and too much customizability contributed to the low rating for customizability.

Wanting more customizability is a sentiment shared by many ML Engineers and ML Innovators, especially users of cloud-hosted solutions. Participants wanted more custom control for computation resource allocation (P2, P4), data cleaning procedures (P6, P7), model search space (P4, P13), and model interpretability techniques (P5).

On the other hand, too many customization options could lead to cognitive overload and hinder progress. P15 reported feeling overwhelmed by the number of hyperparameters that could be customized and needed to consult the documentation. P9, a novice, shared that they believed "there's a lot of tools higher than a five [for customizability] but in a bad way." P3 described the phenomenon of gratuitous customizability:

> "You don't necessarily know what some of the hyperparameters mean some of the time in extensive detail, but you do have the ability to control them all."

Most notably, P16 realized during the course of the interview that the additional customizability they wanted was in fact unnecessary for their use case:

> "I've kind of been harping on how it's not as customizable . . . But the tools that I've looked at lets you select the types of models to evaluate and change your features. They give you capability of managing the the information and shaping the underlying model . . . It's handled quite well."

We will delve deeper into the issue of customizability and control in Section 4.5.

Lacks Transparency and Interpretability.

The lowest rated quality of Auto-ML tools is transparency for the cloud solutions due to their black-box nature and relative lack of

opportunities for user agency in comparison to other tool categories. For increased transparency and usability, a couple of participants expressed the desire for a simple progress bar that gives them insights into how long Auto-ML would take (P2, P13.) However, different user populations desire different levels of transparency to ensure trust:

> "Auto-ML is also an ML model. What that ML model is, how the ML model was trained, how the ML model learns from newer data—that piece is a black box. And so that makes it less trustworthy for people like me who are also ML engineers who knows the success probability of machine learning models. For someone who's not in ML, it's like magic. You just click a button and it works. But for someone who knows ML …I know one of the things that can go wrong [is] a ML model that is not trained well. One of the problems …with Auto-ML is that they don't give you a lot of information about what is actually going on behind the scenes, and that makes it really hard for me to trust." (P12)

Although widely reported by prior work on human-centered Auto-ML [14, 43] that transparency mechanisms increase user trust in Auto-ML systems, our results indicate that transparency mechanisms alone, such as visualization, does not suffice for the level of trust required in high-stake industrial settings, wherein participants need to reason and justify for how and why the model design and selection decisions are made. In order to gain the level of interpretability and trust required for mission-critical projects, participants reported switching to complete manual development for increased user agency and control (P1, P4, P9, P11, P12, P15), as further discussed in Section 4.5. Conversely, lack of agency and tinkering can result in a lack of interpretability and the type of non-transparency caused by illiteracy [10]. Humans develop understanding by doing, as illustrated by P3:

> "You don't have a fundamental understanding of what's happening under the hood. And the other challenges with that …are interpretability …The onus is on me to actually build a competency in them …It makes it basically impossible to go to a business person …[to explain] how do you decide on this transaction. …I actually have to go back …and look at the 300 algorithms documentation, not the best one that I just deployed without really reading up all the details. There is a lot that you let go."

4.5 Roles of the Human in Auto-ML

Contrary to the moniker "automated ML", practitioners do not use Auto-ML tools as push-button, one-shot solutions. Instead, they get into the trenches with Auto-ML during the model development process—instructing, advising, and safeguarding Auto-ML. Humans are valuable contributors, mentors, and supervisors to Auto-ML, improving its efficiency, effectiveness, and safety. Their place "in the loop" cannot be replaced with complete automation. The consequences of removing humans out of the process are ineffectiveness, unpredictability, and potential danger. Below we discuss the crucial roles humans play alongside Auto-ML.

4.5.1 Humans boost Auto-ML's performance and efficiency. Auto-ML, despite all its benefits, cannot be efficient and effective without humans-in-the-loop, because humans' contextual awareness, domain expertise, and ML experience cannot be automated away. Humans are especially indispensable for non-standard uses cases and domains. Without human involvement, Auto-ML cannot meet the stringent requirements of real-world applications.

Human guidance constricts the search space of Auto-ML. The flip side of Auto-ML's comprehensiveness (as a benefit in Section 4.3) is the enormous search space that requires an unwieldy amount of compute and time resources, when attempting to maximize model performance (accuracy, or other performance metrics). Due to Auto-ML's exhaustive search process, participants often only use Auto-ML for light and basic models that can be trained very quickly with a data set that is not too big (P9, P11, P14, P15). Auto-ML blindly searches through the entire space possible, resulting in intractable compute requirements, where "you will be there for years and years searching for models. It's not possible." (P11). Paradoxically, present day Auto-ML tools have no memory of and do not learn from the previous searches to narrow down the space in the next iteration, as P11 described:

> "But [Auto-ML] doesn't learn how to learn, and it doesn't learn the environment in which it learns …I could put in more samples if I want to iterate again …I have to do a whole new Auto-ML search, which is much more time consuming."

Humans, in contrast, learn from their previous experiences, and can guide Auto-ML using heuristics, thus narrowing the search space for Auto-ML, so that it can return outputs that meet the quality standards within time constraints. The most prevalent strategy that participants applied is to define the inclusion and/or exclusion criteria for models and/or hyperparameters (P7, P8, P12.) For example, P8 reported having to curate a list of models to feed into Auto-ML to constrain the search space. P14 also discussed the importance of manually limiting the hyperparameter search space:

> "I think one problem with Auto-ML is that it's very compute intensive. So you actually need to *define your space in a good way*. For example, if you want to find the range of hyperparameters, you need to have some kind of notion of what you want to use as initial parameters. So basically you have a trade-off between time and model accuracy." (P14)

P14's use case requires them to hold the time constraint as a constant, wherein model performance and search space size are inversely related, because for large search spaces untamed by human guidance, Auto-ML would 'cut corners' by skipping some search areas, weakening its performance. P9 described their mental model of Auto-ML's search process when they configured a time limit on Auto-ML runs:

> "It (Auto-ML) wouldn't necessarily do hyperparameter tuning. I'm only realistically going to be able to do that, if I'm using simpler models. Otherwise, I have to sit down with someone who knows the models really well and get a good default set of hyperparameters."

Humans compensate for Auto-ML shortcomings, boosting its performance. Auto-ML's shortcomings become more evident

in non-standard use cases and domains (P4, P6, P9, P11, P12). In such cases, humans use Auto-ML 1) to establish an baseline for performance scores 2) to learn from Auto-ML's strategies 4.3 and manually improve ML model performance. For example, P4 reported:

> "When the task gets too complicated, Auto-ML breaks down . . . you won't get good state-of-the-art results. As soon as the task becomes something out of the normal, I switch to manual. Auto-ML is just a good way to get the intuition behind what kind of performance you should expect in these cases. And then you try to beat that."

Auto-ML also often "over-thinks" and "over-complicates", resulting in diminished performance. Some participants respond by comparing manual and Auto-ML models and objectively select the best performing one based on certain performance metrics.

> "A couple of times, I ended up not using Auto-ML, because it was giving me a very complicated pipeline for regression problem. So I limited it to elastic net only, and that works better than what Auto-ML gave me. . . .I select the best ones among manual and auto models."(P5)

Many participants describe their working relationship with Auto-ML as if they are collaborators [42], working in alternating cycles, iterating based on the feedback from each other (P10, P11, P13.) Humans leverage Auto-ML's strengths and compensate for its shortcomings, engaging with Auto-ML when it needs help. Together, they achieve higher performance (in speed and accuracy) than if they were each to work on their own.

For example, P10 jump-starts their ML workflow with a quick Auto-ML run to narrow down the variables/features, followed by manual feature engineering and refinement via several iterations with Auto-ML before finalizing the training sample for Auto-ML to build models with.

> "*What types of pre-processing is Auto-ML automating for you?* Oh, it's just sample selection. We can quickly run through modeling exercises and see which features or variables that we've tossed in are most important, and we're going to start chopping things out . . . really quickly. And then go back to potentially doing some additional feature engineering ourselves manually or pulling more data in, but on a much limited scope . . . And I'm going to continue to refine with multiple iterations of modeling, what ultimately I'm going to use as a training sample." (P10)

P10 compensates for Auto-ML's shortcoming in feature engineering, while also supporting Auto-ML with data collection—one of the tasks that are extremely to automate.

Humans do what Auto-ML cannot do at all, using contextual awareness and domain expertise. As reported in Section 2, many participants do not believe that tasks such as data pre-processing can ever be automated. The most common steps in the ML workflow where humans are indispensable are data collection (including data labeling), data cleaning, and feature engineering.(P1, P2, P3, P5, P6, P9, P10, P12, P14). Some ML tasks are extremely difficult to automate, such as unsupervised learning and semi-supervised learning (P7, P16.) Certain common data types also require substantial domain expertise, such as text (P9.) Automation's rigidity and lack of contextual awareness are well-studied in HCI [1]. The same is true for Auto-ML. As P6 puts it "Auto-ML is not smarter. It doesn't do what humans cannot do. It is just faster."

4.5.2 Humans increase ML safety and prevent misuse of Auto-ML. Auto-ML's ease-of-use also becomes its Achilles' heel. Participants reported Auto-ML's excessive ease-to-use to be a downfall (P9, P15). P10 suggested that Auto-ML is effective "under proper governance and management." P9 responded to the question "If an Auto-ML and a manual model have the same performance, in which model would you have more confidence?" with "It depends on the person who used Auto-ML." To ensure the safety of Auto-ML decisions and prevent misuse, participants actively engage in the entire workflow as supervisors of Auto-ML, implementing governance and management. The common strategies include the following:

Humans compare manual and Auto-ML strategies as a safety check and to increase trust. Because of the shortcomings of Auto-ML, participants expressed the need to personally validate Auto-ML to establish confidence in Auto-ML models, which often entails manually developing models to compare with Auto-ML outputs (P5, P10, P11, P12, P16.) In addition, practitioners often engage with Auto-ML with a prior expectation of performance. As P16 reported:

> "There are certain times where running your own model, even if it's just as another perspective on the approach, and a **validation step** is still a good idea . . . making sure that the direction of the predictions are in line with expectations [and] there's nothing abnormal happening."(P16)

Prior work found that ML interpretability seeks to establish trust not only between humans and models, but also between humans [23]. Participants reported comparing Auto-ML to manual ML for building trust between people.

> "My goal is to bring to my collaborators interpretable models . . . the biggest challenge [with Auto-ML] is how to convince my collaborators to trust it. [The way I convince them is] we produce the standard approach models and show them that this one (Auto-ML) is actually better in performance terms." (P5)

Humans correct for Auto-ML idiosyncrasies. Ironically, as much as Auto-ML prevents suboptimal performance due to practitioners' idiosyncrasies (as mentioned in Section 4.3), humans correct for Auto-ML's idiosyncracies to safeguard Auto-ML outputs and improve overall performance. P10 describes one such issue with data leakage:

> "The only way that [Auto-ML]detects target leakages is if you have like a 99.99% correlation to the target label. So there's still things you need to know as a data scientists to use Auto-ML effectively."

P10 manually structured the data to ensure Auto-ML has the appropriate data as input to safeguard the model's generalizability and safety.

P16 worked with heavily imbalanced data and had to manually correct the biases in the data to prevent suboptimal performance before feeding the dataset to Auto-ML.

> "I know Auto-ML features allow you to do variable weighting. But . . . I haven't found that to always work necessarily. "(P16)

Humans manually develop ML for understandability and reliability for mission-critical projects. Visibility into Auto-ML work process does not suffice for the level of understanding, trust, and explainability participants need for high-stake projects, where humans are ultimately accountable for the reliable performance of ML models. Participants reported switching back to manually developing ML pipelines for mission-critical settings, because they need to reason and justify why the architecture and hyperparameters are chosen and how classification or prediction decisions are arrived at (P1, P4, P9, P11, P12, P15.)

Prior work concludes that transparency mechanisms, such as visualization, increase user understanding and trust in Auto-ML [14]. We found that transparency alone does not suffice for trust and understanding between humans and the tool-built model. Humans need agency to establish the level of understanding to trust Auto-ML. For example, P11, despite having visibility into Auto-ML, was afforded little understanding. P11 rated transparency of Auto-ML as being extremely transparent.

> "*To what extent do you have visibility into the inner workings of what the Auto-ML is doing?* If you want, it is a five (extremely visible). You can log whatever you want. And you can see what it's doing and it saves all the run and you can look at them in tensor board, you can even explore the architectures, so extremely [visible]."

However, P11 distrusted in Auto-ML and resorted to manual involvement.

> "[With] Auto-ML models, I can look at the architecture and I got no idea what it's doing. It is making connections all over the place. It is using non-conventional convolutional layers and you're like this is not predictable. Granted it's got good performance on one run . . . You can't afford to have that kind of uncertainty . . . You haven't had the **hands-on experience** to at least put your signature on the models and say put into production, so that's the issue."

P11 reported that lack of certainties and predictability in how Auto-ML selects model architecture undermined their trust in Auto-ML and that human agency increased model reliability and assures confidence in the model. This sentiment is also supported by other participants. For example P7, P13 prefer manual development for use cases where Auto-ML's strength in efficiency is irrelevant, because through manual ML development they can gain a deeper understanding and higher trust in the models.

5 DISCUSSION

Our findings show that Auto-ML tools have been effective at making ML more accessible by creating high-level abstractions that streamline the process for training and deploying models. By effectively hiding away the complexities commonly associated with ML and acting as a source of knowledge, Auto-ML tools make ML accessible to novices and increase productivity for experts. However, we argue that current efforts in Auto-ML that strive for an eventual fully automated system through more tool capabilities and more accurate models fail to consider the ways the technology is actually used in practice and what users really need. In this work, we found that *complete automation is neither a requirement nor a desired outcome for users of Auto-ML*. To make Auto-ML tools more effective, the goal should *not* be to completely remove the user from the process but instead to build human-compatible tools by working to achieve trust, understanding, and a sense of agency in the user. In this section, we discuss directions for the further development of Auto-ML tools based on our findings.

Adapt to the proficiency level of the intended user. Especially because Auto-ML tools mainly target citizen data scientists who often are intimated by ML due to perceived self-inefficacy, developers need to consider users' psychological readiness, designing from the position of a partner who has a sensitivity to the other's level of comfort (P12, P16.) P16 expressed this as follows:

> "A lot of the Auto-ML tools make an assumption about the technical competency of the user. That's to their detriment. I think if the goal is really to try and make it easier to use, there needs to be substantial effort put into the UX, and understanding of potential users that don't have. . . the depth of knowledge. . . [It is used] more as exploratory analytics or proof of concept."

Therefore Auto-ML tools need to be designed to adapt to users with varying comfort level instead of taking a one-size-fits-all approach.

Lingua Franca for ML. Developers of Auto-ML tools need to grapple with all the different roles that humans play in real-world data science work practices and how tools often need to take on a translator role among collaborators. P1 reported that even though many data science projects are initiated by business teams, the data scientists also often proactively propose innovative solutions to business teams based on their awareness of the challenges faced by business teams. ML engineers in real-world working environments do not simply passively react to requests from stakeholders. They also make contributions to the framing of the problem based on their ML expertise. Prior work highlights a translator role who sits between the data scientists and other stakeholders in collaborative data science projects [42]. We found that many ML engineers perform the role of the translator, directly interfacing with stakeholders, interpreting business problems and framing them into objectives that can be evaluated by ML. P9 described this challenge of mismatched mental models among collaborators:

> "The way business thinks about accuracy is often really really different from how you would calculate any sort of traditional metrics. . . You have to sync with a lot of people on exactly how they think accuracy works and how they want to report it."

Auto-ML tools need to be aware of this mismatch and adapt their language accordingly to proactively act as a translator.

Holistic platform. An important lesson to be learned from the large difference in favorability between the different categories of Auto-ML tools is that an *end-to-end* solution that handles all stages of the ML workflow in a single environment is the optimal design choice for Auto-ML solutions. As mentioned in Section 2.1, Auto-ML platforms tend to have self-sufficient end-to-end ML workflow support due to limited options to integrate with external solutions,

unlike OSS and Cloud Provider solutions. This has led to Auto-ML platforms being not only the most complete solution but also the easiest to use, the most efficient (no data transfer), and the most interpretable (comprehensive data lineage). Since Auto-ML is a highly complex system catered towards users with diverse backgrounds performing cognitively taxing tasks, interoperability with external solutions is an anti-pattern. This is not to say that there is no use for non-platform Auto-ML solutions, but rather there needs to be a common substrate for the disparate Auto-ML tools and libraries, akin to Weld for data analytics [34].

Serverless Computing. As presented in Section 4.4, Auto-ML workloads are compute intensive but bursty, and the optimal hardware is highly variable depending on the dataset and model characteristics. These conditions motivate the need for elastic, ephemerally provisioned computation resources with variable specs. For example, for P8 whose dataset contained 18 million columns, they could be temporarily allocated machines with ample RAM, instead of the fixed-architecture university cluster they were using, to avoid crashing due to out-of-memory errors and accelerate model search. Recent advancements in serverless computing [8] can be harnessed to solve this problem. However, the Auto-ML setting poses new challenges in terms of the economics of trading compute cost for model accuracy.

Adaptive UI. Supporting users with diverse skills and expertise is an inherent challenge Auto-ML solutions must embrace. Evidence from Section 4.3 suggests that a blackbox interface for Auto-ML can be beneficial to novices and lower the maintenance overhead for all, but low customizability and transparency associated with blackbox interfaces hinder trust and agency based on evidence from Section 4.4. Auto-ML tool developers are forced to grapple with competing design objectives, and a natural solution to this conundrum is to provide multi-modal interfaces covering a spectrum of interaction levels.

The Auto-ML tooling landscape has progressed towards the low code/no code direction, with some solutions allowing experienced users to assume more control via a secondary programmatic interface. However, these tools do not offer true multi-modal interfaces but merely the option to export the raw model training code for further manual exploration, creating a clunky user interface "geared towards ML engineers who can learn to deal with broken UIs" (P12). Furthermore, the success of a multi-modal interface is contingent upon a user's ability to self select the most appropriate modality, which is not easily achievable as illustrated by P16's experience with customizability. A possible solution to this UX challenge is instead of having multiple distinct modalities, the Auto-ML tool can adaptively reveal or hide customization capabilities piecemeal, based on some approximation of user skill level and intent.

Interactive Exploration. Our findings in Section 4.5 show that the human "in-the-loop" can be an indispensable resource to complement Auto-ML and make it more efficient and effective. In tools that support search space customization, human supervision is provided in a one-shot fashion. If the user wanted to iteratively refine the search space, they would have to manually kick off multiple rounds of the one-shot process, keeping track of intermediate results manually between iterations. Iterating with Auto-ML can be better supported by an interactive exploration interface designed specifically with iteration in mind. Such an interface needs to display summaries of all previous iterations and make it easy for the user to specify new search sub-spaces. It also needs to reconcile the high latencies of Auto-ML workloads and interactivity.

Balance between Human Control and Automation. In Section 4, we presented several deficiencies of the tools that can be improved with some of the suggestion above, as well as roles that are important for the human to assume when using Auto-ML. The decisions for what functionalities belong in either group are predicated upon, first and foremost, how to establish trust and agency in the human users, and the current state of ML, systems, and HCI research. Depriving users of trust and agency prevents Auto-ML from making an impact in real-world use cases, while attempting to automate certain features prematurely, without fundamental shifts in the underlying technology, requires strenuous efforts with little payoff in the user experience.

Certain tasks cannot be addressed with human control or automation alone but rather require a delicate balance between the two. Take data pre-processing for example. Evidence in Section 4.5 suggests that many users deem complex feature engineering simply out-of-reach for existing Auto-ML systems, due to their inability to capture domain knowledge. However, many also feel that existing tools lack basic support for mechanical tasks such as distributed processing and canonical data transformations (Section 4.4). Thus, the future of data preprocessing is a combination of adding support for mechanical tasks in the near term, and providing intuitive ways to specify high-level domain knowledge to integrate human intuition with Auto-ML long term.

Another example is the collaboration of humans and the machine for efficient hyperparameter tuning. While expert users believe that they can improve the efficiency of Auto-ML through intuition and experience, this belief is juxtaposed with their recognition that automated search can also correct for their biases and idiosyncrasies (Section 4.3). To strike a balance, we envision an interactive dialog between the human and the machine to iteratively discover and fill in each other's blindspots. Human guidance will be treated as strong priors on certain model subspaces but does not preclude the exploration of other subspaces, allowing the machine to nudge the human towards promising but underexplored subspaces.

6 CONCLUSION

In this paper, we presented results from a semi-structured interview study of 16 practitioners who have used Auto-ML in real-world applications. Our participants all reported various types of hybrid manual-auto strategies in leveraging Auto-ML in their development workflow, providing hints, safeguarding outputs, and massaging inputs into a form digestible by Auto-ML tools, among others. Current work practices around Auto-ML and perceptions of these tools demonstrate that complete automation of ML is neither realistic nor desirable. Our study sheds light on various forms of partnership or collaboration between humans and ML/AI, depending on user motivations, needs, skill-set, and use-cases, and provides next-generation Auto-ML tool developers with design guidelines for how to best incorporate pragmatic guidance and empower effective engagement with Auto-ML tools.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback. We acknowledge support from grants IIS-1940759 and IIS-1940757 awarded by the National Science Foundation, and funds from the Alfred P. Sloan Foundation, Facebook, Adobe, Toyota Research Institute, Google, and the Siebel Energy Institute. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies and organizations.

REFERENCES

[1] Mark S. Ackerman. 2000. The Intellectual Challenge of CSCW: The Gap between Social Requirements and Technical Feasibility. *Hum.-Comput. Interact.* 15, 2 (Sept. 2000), 179–203. https://doi.org/10.1207/S15327051HCI1523_5

[2] Google AI. 2017. Facets: An Open Source Visualization Tool for Machine Learning Training Data. https://ai.googleblog.com/2017/07/facets-open-source-visualization-tool.html

[3] AmazonSagemakerAutopilot. 2020. Automate model development with Amazon SageMaker Autopilot. Website. Retrieved July 26, 2020 from https://docs.aws.amazon.com/sagemaker/latest/dg/autopilot-automate-model-development.html

[4] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software Engineering for Machine Learning: A Case Study. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '19)*. IEEE Press, Montreal, Quebec, Canada, 291–300. https://doi.org/10.1109/ICSE-SEIP.2019.00042

[5] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. Power to the People: The Role of Humans in Interactive Machine Learning. *AI Magazine* 35, 4 (2014), 105–120. https://doi.org/10.1609/aimag.v35i4.2513

[6] Saleema Amershi, Max Chickering, Steven M. Drucker, Bongshin Lee, Patrice Y. Simard, and Jina Suh. 2015. ModelTracker: Redesigning Performance Analysis Tools for Machine Learning. In *Proceedings of the 33rd ACM SIGCHI Conference on Human Factors in Computing Systems*. ACM, Seoul, Korea, 337–346. https://doi.org/10.1145/2702123.2702509

[7] Alexander Ratner Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid Training Data Creation with Weak Supervision. *Proceedings of the VLDB Endowment* 11, 3 (2017), 269–282.

[8] Ioana Baldini, Paul Castro, Kerry Chang, Perry Cheng, Stephen Fink, Vatche Ishakian, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Aleksander Slominski, et al. 2017. Serverless computing: Current trends and open problems. In *Research Advances in Cloud Computing*. Springer, none, 1–20.

[9] Eran Bringer, Abraham Israeli, Yoav Shoham, Alex Ratner, and Christopher Ré. 2019. Osprey: Weak Supervision of Imbalanced Extraction Problems without Code. In *Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning (DEEM'19)*. Association for Computing Machinery, New York, NY, USA, Article 4, 11 pages. https://doi.org/10.1145/3329486.3329492

[10] Jenna Burrell. 2016. How the machine 'thinks': Understanding opacity in machine learning algorithms. *Big Data & Society* 3, 1 (2016), 2053951715622512. https://doi.org/10.1177/2053951715622512

[11] Corinna Cortes, Xavi Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. 2017. AdaNet: Adaptive Structural Learning of Artificial Neural Networks. arXiv:cs.LG/1607.01097

[12] Datarobot. 2020. DataRobot Automated Machine Learning. Website. Retrieved July 18, 2020 from https://www.datarobot.com/platform/automated-machine-learning/

[13] dedoose. 2020. Dedoose. Website. Retrieved September 15, 2020 from https://www.dedoose.com

[14] Jaimie Drozdal, Justin Weisz, Dakuo Wang, Gaurav Dass, Bingsheng Yao, Changruo Zhao, Michael Muller, Lin Ju, and Hui Su. 2020. Trust in AutoML: Exploring Information Needs for Establishing Trust in Automated Machine Learning Systems. In *Proceedings of the 25th International Conference on Intelligent User Interfaces (IUI '20)*. Association for Computing Machinery, New York, NY, USA, 297–307. https://doi.org/10.1145/3377325.3377501

[15] Sylvain Duranton, Jörg Erlebach, Camille Brégé, Jane Danziger, Andrea Gallego, and Marc Pauly. 2020. What's Keeping Women Out of Data Science? https://www.bcg.com/en-us/publications/2020/what-keeps-women-out-data-science.

[16] Jerry Alan Fails and Dan R. Olsen. 2003. Interactive Machine Learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces (IUI '03)*. Association for Computing Machinery, New York, NY, USA, 39–45. https://doi.org/10.1145/604045.604056

[17] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and Robust Automated Machine Learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'15)*. MIT Press, Cambridge, MA, USA, 2755–2763.

[18] Rebecca Fiebrink, Perry R Cook, and Daniel Trueman. 2011. Human Model Evaluation in Interactive Supervised Learning. *CHI 2011* 4, 2 (2011), 147–156. https://doi.org/10.14529/jsfi170202

[19] Yolanda Gil, James Honaker, Shikhar Gupta, Yibo Ma, Vito D'Orazio, Daniel Garijo, Shruti Gadewar, Qifan Yang, and Neda Jahanshad. 2019. Towards Human-Guided Machine Learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces (IUI '19)*. Association for Computing Machinery, New York, NY, USA, 614–624. https://doi.org/10.1145/3301275.3302324

[20] GoogleCloudAutoML. 2020. Google Cloud AutoML. Website. Retrieved July 18, 2020 from https://cloud.google.com/automl

[21] H2O.ai. 2020. H2o.ai Automated Machine Learning. Website. Retrieved July 18, 2020 from https://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html#automl-interface

[22] Fred Hohman, Kanit Wongsuphasawat, Mary Beth Kery, and Kayur Patel. 2020. Understanding and Visualizing Data Iteration in Machine Learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376177

[23] Sungsoo Ray Hong, Jessica Hullman, and Enrico Bertini. 2020. Human Factors in Model Interpretability: Industry Practices, Challenges, and Needs. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW1 (2020), 1–26.

[24] Haifeng Jin, Qingquan Song, and Xia Hu. 2019. Auto-Keras: An Efficient Neural Architecture Search System. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. Association for Computing Machinery, New York, NY, USA, 1946–1956. https://doi.org/10.1145/3292500.3330648

[25] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. 2012. Enterprise Data Analysis and Visualization: An Interview Study. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (Dec 2012), 2917–2926. https://doi.org/10.1109/TVCG.2012.219

[26] Trang T Le, Weixuan Fu, and Jason H Moore. 2020. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. *Bioinformatics* 36, 1 (2020), 250–256.

[27] Angela Lee, Doris Xin, Doris Lee, and Aditya Parameswaran. 2020. Demystifying a Dark Art: Understanding Real-World Machine Learning Model Development. arXiv:cs.LG/2005.01520

[28] Doris Jung Lin Lee, Stephen Macke, Doris Xin, Angela Lee, Silu Huang, and Aditya G. Parameswaran. 2019. A Human-in-the-loop Perspective on AutoML: Milestones and the Road Ahead. *IEEE Data Eng. Bull.* 42 (2019), 59–70.

[29] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. 2019. Reliability and Inter-rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice. *Proceedings of the ACM on Human-Computer Interaction* 3 (11 2019), 1–23. https://doi.org/10.1145/3359174

[30] Wes McKinney et al. 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, Vol. 445. SciPy 2010, Austin, TX, 51–56.

[31] MicrosoftAzureAutomatedML. 2020. Microsoft Azure Automated Machine Learning. Website. Retrieved September 15, 2020 from https://azure.microsoft.com/en-us/services/machine-learning/automatedml/

[32] Piero Molino, Yaroslav Dudin, and Sai Sumanth Miryala. 2019. Ludwig: a type-based declarative deep learning toolbox. arXiv:cs.LG/1909.07930

[33] Jorge Piazentin Ono, Sonia Castelo, Roque Lopez, Enrico Bertini, Juliana Freire, and Claudio Silva. 2020. PipelineProfiler: A Visual Analytics Tool for the Exploration of AutoML Pipelines. arXiv:cs.HC/2005.00160

[34] Shoumik Palkar, James J Thomas, Anil Shanbhag, Deepak Narayanan, Holger Pirk, Malte Schwarzkopf, Saman Amarasinghe, Matei Zaharia, and Stanford InfoLab. 2017. Weld: A common runtime for high performance data analytics. In *Conference on Innovative Data Systems Research (CIDR)*. CIDR, Chaminade, California, 45.

[35] Kayur Patel, Naomi Bancroft, Steven M. Drucker, James Fogarty, Andrew J. Ko, and James Landay. 2010. Gestalt: Integrated Support for Implementation and Analysis in Machine Learning. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. Association for Computing Machinery, New York, NY, USA, 37–46. https://doi.org/10.1145/1866029.1866038

[36] Gonzalo Ramos, Christopher Meek, Patrice Simard, Jina Suh, and Soroush Ghorashi. 2020. Interactive machine teaching: a human-centered approach to building machine-learned models. *Human-Computer Interaction* 35 (04 2020), 1–39. https://doi.org/10.1080/07370024.2020.1734931

[37] Yuji Roh, Geon Heo, and Steven Euijong Whang. 2019. A Survey on Data Collection for Machine Learning: a Big Data – AI Integration Perspective. arXiv:cs.LG/1811.03402

[38] Justin Talbot, Bongshin Lee, Ashish Kapoor, and Desney S. Tan. 2009. EnsembleMatrix: Interactive Visualization to Support Machine Learning with Multiple Classifiers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. Association for Computing Machinery, New York, NY, USA, 1283–1292. https://doi.org/10.1145/1518701.1518895

[39] TransmogriFAI. 2020. TransmogrifAI. Website. Retrieved September 5, 2020 from https://github.com/salesforce/TransmogrifAI

[40] Michelle Vaccaro and Jim Waldo. 2019. The Effects of Mixing Machine Learning and Human Judgment. *Commun. ACM* 62, 11 (Oct. 2019), 104–110. https://doi.org/10.1145/3359338

[41] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. 2011. The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering* 13, 2 (2011), 22.

[42] Dakuo Wang, Justin D. Weisz, Michael Muller, Parikshit Ram, Werner Geyer, Casey Dugan, Yla Tausczik, Horst Samulowitz, and Alexander Gray. 2019. Human-AI Collaboration in Data Science: Exploring Data Scientists' Perceptions of Automated AI. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 211 (Nov. 2019), 24 pages. https://doi.org/10.1145/3359313

[43] Qianwen Wang, Yao Ming, Zhihua Jin, Qiaomu Shen, Dongyu Liu, Micah J. Smith, Kalyan Veeramachaneni, and Huamin Qu. 2019. ATMSeer: Increasing Transparency and Controllability in Automated Machine Learning. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300911

[44] Daniel Karl I. Weidele, Justin D. Weisz, Eno Oduor, Michael Muller, Josh Andres, Alexander Gray, and Dakuo Wang. 2019. AutoAIViz: Opening the Black-box of Automated Artificial Intelligence with Conditional Parallel Coordinates. arXiv:cs.LG/1912.06723

[45] Wikipedia contributors. 2020. One-hot — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=One-hot&oldid=975049657 [Online; accessed 17-September-2020].

[46] Kanit Wongsuphasawat, Daniel Smilkov, James Wexler, Jimbo Wilson, Dandelion Mané, Doug Fritz, Dilip Krishnan, Fernanda B. Viégas, and Martin Wattenberg. 2018. Visualizing Dataflow Graphs of Deep Learning Models in TensorFlow. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 1–12. https://doi.org/10.1109/TVCG.2017.2744878

[47] Doris Xin, Stephen Macke, Litian Ma, Jialin Liu, Shuchen Song, and Aditya Parameswaran. 2018. Helix: holistic optimization for accelerating iterative machine learning. *Proceedings of the VLDB Endowment* 12, 4 (2018), 446–460.

[48] Qian Yang, Jina Suh, Nan-Chen Chen, and Gonzalo Ramos. 2018. Grounding Interactive Machine Learning Tool Design in How Non-Experts Actually Build Models. In *Proceedings of the 2018 Designing Interactive Systems Conference (DIS '18)*. Association for Computing Machinery, New York, NY, USA, 573–584. https://doi.org/10.1145/3196709.3196729