Grounding Interactive Machine Learning Tool Design in How Non-Experts Actually Build Models

Qian Yang[1] **Jina Suh**[2] **Nan-Chen Chen**[3] **Gonzalo Ramos** [2]
Human-Computer Interaction Institute, Carnegie Mellon University [1]
Microsoft Research [2]
Human-Centered Design & Engineering, University of Washington [3]
yangqian@cmu.edu jinsuh@xbox.com nanchen@uw.edu goramos@microsoft.com

ABSTRACT
Machine learning (ML) promises data-driven insights and solutions for people from all walks of life, but the skill of crafting these solutions is possessed by only a few. Emerging research addresses this issue by creating ML tools that are easy and accessible to people who are not formally trained in ML ("non-experts"). This work investigated how non-experts build ML solutions for themselves in real life. Our interviews and surveys revealed unique potentials of non-expert ML, as well several pitfalls that non-experts are susceptible to. For example, many perceived percentage accuracy as a sole measure of performance, thus problematic models proceeded to deployment. These observations suggested that, while challenging, making ML easy and robust should both be important goals of designing novice-facing ML tools. To advance on this insight, we discuss design implications and created a sensitizing concept to demonstrate how designers might guide non-experts to easily build robust solutions.

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

Author Keywords
Interactive Machine Learning; End-user Machine Learning; Machine Teaching; Empirical Study; User-Centered Design; Sensitizing Concept.

INTRODUCTION
Machine learning (ML) promises data-driven insights and solutions for a wide variety of domains and people from all walks of life, but crafting these solutions generally requires knowledge that is possessed by only a few. Emergent ML and HCI research aims to make solutions easy to build and accessible, enabling more people to build ML solutions for their respective domains of interest [21, 24].

DIS '18, June 9–13, 2018, , Hong Kong

ISBN 978-1-4503-5198-0/18/06…$15.00
DOI: https://doi.org/10.1145/3196709.3196729

Making ML accessible to people beyond formally trained data scientists is a reality that some researchers are focused on achieving. Technical ML community has worked to improve amateurs' efficiency and reliability in labeling, aiding feature engineering and error-proving, for instance [17]. HCI researchers have created interactive machine learning (iML) tools for developers and end users [9, 16, 29, 7]. Many in the industry even promote the idea of "ML for everyone", creating tools that amateurs could walk up and use [3, 14, 26, 27].

Given the potential benefits and rapid growth in the work that makes ML accessible, it seems that understanding how amateurs build ML solutions for themselves should be a part of iML research and tool design. Interestingly, such investigation is rare. Extensive prior work instead focused on lab studies, where crowd workers accomplish pre-identified ML tasks following preset procedures (i.e. [5, 6, 17]). Some HCI researchers noted a lack of user understanding in the design of these systems and called for "power to the people" in ML [1, 12, 20].

To bridge the gap between accessible iML technology and its intended users, we conducted an empirical study. We focused on those whom we call "non-experts" – *people who are not formally trained in ML, and are actively building ML solutions to serve their needs in the real world* (e.g., a corporate recruiter who built a job candidate profile classifier to aid his decision; or a hobbyist developer who built a signal classifier for her context-aware mobile app). We wanted to understand how they built ML solutions in real-life contexts, including their goals, their approaches to ML and the challenges they encountered. We wanted to identify opportunities where accessible ML tools might help.

We interviewed 24 non-experts who were building ML models and the ML consultants some of them hired to help. We surveyed another 98 non-experts to collect more diverse ML experiences. The study revealed that non-experts are more satisfied and trusting toward the learning results than their professional counterparts. However, they were also susceptible to several technical pitfalls. For example, they tend to perceive percentage accuracy as a sole measure of model performance, thus problematic or even invalid models sometimes proceeded to deployment.

These observations helped us shift the focus away from thinking of non-expert ML as being simply a problem of facilitating the needs of data scientists, or one of reducing tool complexities. While challenging, making the process of building ML models easy, flexible, and robust should be an important goal of designing accessible ML tools. But how? The design seems inevitably need to trade off achieving one goal against interfering with another. [20]. To help both ourselves and fellow designers address this challenge, we designed a new interaction flow for accessible iML tools, namely *Test-Driven Machine Teaching*, exemplifying one possible solution. It functions as a sensitizing concept [31], informing and inspiring HCI researchers of the rich opportunities in this open design space. We discuss its design implications and other open research questions the empirical findings inspire.

This work makes three contributions. 1) Our empirical study provides a rare description of how non-experts actually build ML solutions in the real world. 2) This work provides an alternative perspective to the common assumption that novice-facing tools are GUI tools that reduce or hide ML complexities. An ML tool might better support non-experts if it considered their unique mental model of ML and guided them to build more robust models. 3) Our sensitizing concept exemplifies one possibility in this new design space, offering a starting place for future design innovation.

RELATED WORK

Understanding People in Interactive Machine Learning

In HCI and ML literature, people who build ML models fall into three categories: experts, intermediate users of ML [16], and amateurs. This work focuses on the latter two populations; together we refer to them as "non-experts".

Interestingly, it remains unclear how much ML knowledge qualifies one as an "intermediate user", and what level of ML knowledge stratifies across intermediate and expert users. Researchers defined intermediate users as those "who have *some* experience with ML, but without a *deep* knowledge the experts have" [7, 16]. Without a clear-cut definition, some researchers identified intermediate users based on the tools they use: Those who use graphical user interface (GUI) tools or libraries like Weka are intermediate users; Those who implement their own algorithms are experts [16].

Little to no research has investigated how intermediate users work with ML empirically. A related strand of work has looked at how software engineers build applications that use ML [12, 20]. HCI researchers have also shared their own experiences [9, 10, 29]. Together they revealed many challenges of applying ML in software applications. For example, they had difficulties in understanding the limits of what can be learned; in exploring different formulations of an ML problem, and in evaluating model performance in the context of its application.

Extensive research has investigated ML amateurs–those who interact with ML systems but have no knowledge of ML, such as some domain experts and end users. They typically are engaged in ML when automatic ML approaches fail or deliver unsatisfactory results [1]. Through various crowd-sourcing systems, amateurs help label data, repair data flaws, reason about data, test and troubleshoot learning algorithms. Research has created many techniques that improve crowd workers' efficiency and reliability in these distinct steps of ML pipelines [5, 6, 8, 18, 25, 28].

HCI researchers pointed out that some design of these systems failed to account for users; that the role of amateurs in ML should not be finite state machines that repeatably tell algorithms what is right versus wrong [1]. They called for increasing collaborations across the fields of HCI and ML, and for bringing "*Power to The People in iML*" [1, 20]. They found that integrating user understanding into crowd-sourced ML systems can improve both crowd-workers' experience and efficiency [1, 25].

While making important contributions, these crowd-sourced systems remain focused on predefined ML problems and pipelines. We are not aware of studies that investigated how amateurs build learning algorithms in real-world contexts.

Designing Accessible iML Tools

By accessible iML tools, we refer to tools that enable non-experts to build ML models. Interestingly, most academic iML systems do not specify the amount of ML knowledge required for proper use, making it practically impossible to differentiate accessible ML tools from expert-facing ones. Few other tools, such as Gestalt [19], supported the integration of ML and larger software development workflows. While making ML more accessible, these tools assumed sufficient ML knowledge among their users, therefore fall out of the purview of this work.

When designing explicitly for non-experts, researchers removed complex steps and interactions in the ML process. These tools uniformly adopted GUI designs, eliminating the need for users to write code (i.e. [7, 9, 16]). In addition, some tools became focused on particular application domains and limited the types of input a user can provide [7, 21]. Users have little or no recourse if their problem does not conform to the assumptions made by the tool. Researchers concluded this design choice as the trade-off between generalizability and accessibility. To make accessible ML tools, designers had to sacrifice flexibility for ease of use [20].

There is a noticeable lack of connection between these designs and the work that investigated non-experts. Accessible iML tools in HCI research rarely came out of an elaborate understanding of their users, or the challenges they face in real-world situations. Commercial iML tools that promote the idea of ML for everyone did not document how their design choices support their diverse users [11, 14]. Bridging these two threads of research, this work brings a user-centered lens to the currently technology-driven iML tool design and research.

EMPIRICAL STUDY DESIGN

We aimed to investigate how non-experts actually build ML models for their own purposes in real-world contexts. We aimed to identify opportunities where accessible ML tools might help. Towards these goals, we chose to conduct a qualitative study consisting of interviews and an open-question

Non-Experts			Supporting Experts*
Profession	**Example ML Problem**	**count**	**count**
Professional Software Engineer	Bug report classifier	4	2
Project Manager	User feedback classifier	2	2
Manager	HR Policy Q&A bot	2	1
Business Analytics	Predictive machine maintenance	1	2
Artist	Emotion classifier for wearables	1	
Botanist	Predictive plant nutrient management	1	1
Academic Researcher	Sensor signal classifier	1	
Clinical Researcher	Prognostic classifier	1	
Mechanical Engineer	Insurance risk estimate	1	1

Table 1. Interview study participants. We focused on non-experts, those who are not formally trained in ML, and are actively teaching learners to solve particular problems. [*] We also interviewed the ML consultants whom the non-experts hired to help.

survey. We chose interviews because we wanted to capture in rich detail the thought processes of the non-experts when building ML models. We used an online survey because we wanted to collect more narratives of such processes from a broader population.

Participants

Previous research offered us no clear division between experts and non-experts. In this study, we chose to use their educational degree as approximation. We had two criteria when screening non-expert participants: 1) Participants do not have a degree in ML, statistics, mathematics or artificial intelligence; 2) Participants are pro-actively building one or more ML models for his/her own purposes. Our interviews and survey were then devised to probe participants' specific knowledge and skillset, including subject domain knowledge, statistics and ML knowledge, programming skills, etc.

Field Interviews

We conducted semi-structured interviews with users of four consumer-facing ML services. The four services are an iML tool for building ML and deep learning applications, a consulting service for ML analytics, a consulting service for data-driven business decision making, and a chat-bot-building framework. We recruited an initial set of participants through our professional networks in these ML teams, and later expanded via snowball sampling.

We interviewed 14 non-expert users of these services. We asked participants to walk us through in detail the problems had and how they solved or failed to solve the problems with ML. Several participants also offered to share with us their data schema, iML tool workbench status, documents, or notes. All interviews were audio-recorded and transcribed for analysis. Table 1 provides a summary of the participants' profession and the type of projects they described in our study.

During these interviews, we found that some of the non-experts hired ML consultants to help. In order to better understand where and how they sought help, we also interviewed 10 of these consultants. We were also invited to observe two of the clients' meetings with their ML consultants for ongoing projects. The consultants are professional data scientists trained in ML, mathematics or statistics. In the data analysis and the rest of the paper, we strictly differentiate the narratives of non-experts and ML consultants.

Online Survey

We then sought to collect more diverse ML experiences through an online survey. The survey centered on one open question: "*Describe in detail a recent project where you built or attempted to build an ML model*". Follow-up questions inquired about this model's inputs and outputs, data, tool use, code functionality (if applicable), participants' expectations, frustrations and results. We publicized links to the survey on two *Reddit* boards for ML hobbyists. We also promoted it via ML-related mailing lists of a large-scale technology company and of two universities. We collected 98 valid survey responses from self-identified non-experts.

We analyzed both survey responses and interview transcriptions using affinity diagrams [2] and sequence-flow models. We created multiple affinity diagrams based on ML workflow, the expertise participants have, the problems they encountered, the application domains, etc. We discussed patterns in non-expert ML as they emerged.

EMPIRICAL FINDINGS

We organized our empirical findings around four themes: non-experts' motivations to build ML models, their overall experience, their knowledge of ML and their blind spots.

Motivation Toward Machine Learning

A majority of non-experts began to experiment with ML because they had a dataset at hand, rather than a clear ML goal in mind. ML consultants shared that a vast majority of their clients come with available data, and may "*want ML but do not yet know exactly what for*" (P9,10,23).

Many non-experts built ML models solely for gaining data insights (Figure 1b). A model's accuracy or related metrics

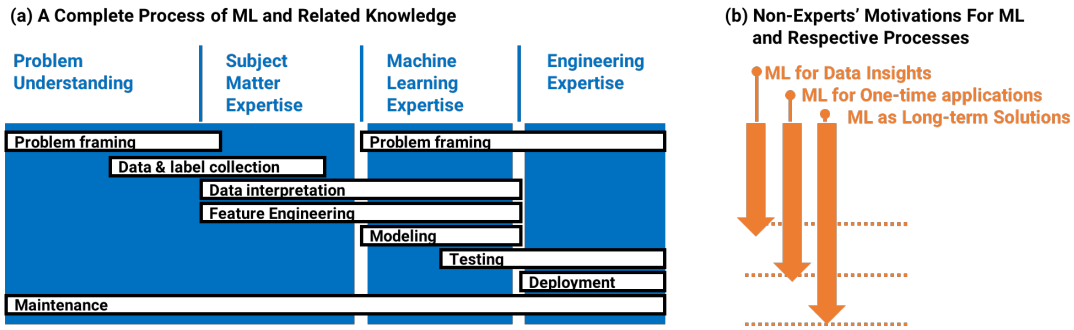

Figure 1. (a) A consolidated ML process and the four fields of expertise involved; (b) non-experts' varied goals. Their goals anchored the fractions of an ML pipeline they need to implement, and frequently veered them away from some of the thorny challenges of ML.

was not their focus. Rather, they value the byproducts of the ML model: the choices of features that generated the best-performing model. For example, a mortgage website manager (P9) applied ML to the web log data, in searching for insights on why the users drop their mortgage applications halfway.

> *Tell me what I don't know about the data.* (P24)

> *The fun part was seeing what a most predicted phrase would be.* (Survey respondent #37, building a chatbot)

Others who build models for intelligent applications evaluated ML success as a matter of improvement; their goal was to build a good enough model that improves the current manual or heuristic-based solution. Very few of them had a predefined learning problem in mind that must be modeled as accurately as possible.

> P4 (artist who creates wearable robots): *it doesn't need to be a perfect algorithm for that (my art) to work. It's nice if it is. The algorithm is not the important part of that, it was the demonstration of the wearable and my ideas.*

> *Machine learning is just a chainsaw in a workshop. I'd love to work in a workshop that has a chainsaw, but I don't necessarily need it.* (P2, developer, building a smart sensor for his home):

Experience of Machine Learning

Non-experts in our study were able to build models that they perceive as satisfactory. They first formulated a task for the ML system to perform based on the data they have. When the available data is insufficient for the problem, they adjusted their expectations and designed a new model to build. As such, non-experts did not need to acquire additional data or labels. Other participants who did not have a ready dataset at hand took a similar approach. They started with a vaguely defined problem space and searched for publicly available ML tutorials or scripts that solve comparable problems. The search results provided non-experts with possible ML problem formulations and solutions. They then implement the fledgling problem-solution and iterate, until they arrived on a matching problem-solution pair.

> *I don't understand the frustrations yet.* (#28)

> *This is my fist project with ML, due to its success I'm planning to create my next project in one month.* (#50)

Non-experts implemented learning models by showing desired outputs and inputs. They understood an ML algorithm as an input-output (I/O) mapping mechanism; a mechanism in which data inputs go in, and the prediction and a performance score come out. They tweaked the algorithms that their tools offered until the accuracy was good enough; otherwise, they adjusted their ML goal and iterated on this experimental process.

> *I need to be see what the model looks like so I can be sure of what I'm building, and can modify and adjust details.* (Survey respondent #93)

In our interviews, all non-experts expressed no doubt on validity of the models they built, regardless of their level of ML experience or the interpretability of the model. They trusted their ML results more than their consultant collaborators.

Non-experts' level of trust, in the ML consultants' opinion, depends on the extent to which they are engaged in the building process. In order to gain their clients' trust on their ML model, ML consultants frequently reported data insights to the clients, presented the models built in every debugging iteration, and kept the algorithms explainable even at the cost of model performance. This interactivity in this process made the non-experts trust the result models.

Knowledge of Machine Learning and Tool Use

How did non-experts build satisfying and trustworthy models by merely mapping in/outputs? To take a closer look, we created a sequence-flow map, detailing the knowledge and skills that our participants utilized at different stages of the process. This surfaced four fields of expertise instrumental for the success of ML: problem understanding, subject matter expertise, engineering expertise and ML expertise (Figure 1a). Drawing on them, we note non-experts' skillset and discuss the challenges they knowingly or unknowingly experienced.

Problem Understanding and Subject Matter Expertise

Problem understanding is the in-depth knowledge about what predictions could be valuable to a real-world situation, and how the prediction would be used in that context. All

of the non-experts in our study have profound problem understandings as they built models for their work or hobbies.

Subject matter expertise includes the abilities to explain jargon and nominate possible feature interrelations, and a detailed understanding of how the data is collected in the field and its potential flaws. Most of the non-experts in our study are subject matter experts too. Even when collaborating with ML consultants, they played leading roles in interpreting data, guarding against insensible data maneuvers, suggesting important features to include, and directing how to visualize and use ML outputs.

Engineering Expertise

Engineering expertise includes the skills to write code, and the knowledge of software engineering best practices.

Non-experts did not perceive programming as a challenge. Formally trained software engineers make up a majority of the non-expert participants. Only one of the 98 survey respondents had no programming skills at all and s/he managed to build the model using GUI tools (Excel and Weka). Others relied on publicly available scripts and learning from online working examples. They expressed frustration that they felt limited by the possibilities these tools offer, but none cited inability to write code as barrier to their ML success.

It is worth noting that, even among those who do have a degree in computer science, many did not write code at all. More than half of them wrote less than a hundred lines of code. When building models for personal use or for hobby projects, they only wrote code to prepossess data or craft features.

Machine Learning Expertise

Beyond the ability to build a working model, ML expertise also includes a variety of aspects of practical data science knowledge: 1) the statistical knowledge, such as processing data based on descriptive statistics and interpreting algorithm performances based on visualizations and performance matrices; 2) topical knowledge such as "X model works well with Y kind of data"; and finally, 3) procedural knowledge of ML, such as how to assemble multiple components of an ML solution via large masses of "glue codes".

Non-experts learned how to build an ML model by reading the documentation of iML tools/APIs, at the time when they started building a model. One ML consultant shared that, across all her clients, "*how well they build a model depends on how much effort they put into reading documentations*".

Interestingly, non-experts rarely read API documentation at all, unless pushed by their consultants. Non-experts instead simply relied on pivoting publicly-available working ML examples, re-purposing these scripts toward their needs. The knowledge and reasons underlying these working examples remained unknown to the them.

> *The main challenge along the way was the problem domain and the nature of the data were quite unique – no publicly available analogous datasets or literature.* (Survey respondent #42)

Non-experts rarely attempted to introspect the internal mechanism of learning algorithms. They did not attempt to use data visualization or descriptive statistics to understand their data or introspect their models. None of them has mentioned any consideration about unintelligible features, unseen data or model generalizability.

The characteristics of different learning algorithms were not in non-experts' sphere of knowledge either. Among the survey respondents, nearly half non-experts have only explored one algorithm throughout their ML processes; Another 10% of the survey respondents either "*tried all algorithms that I could think of*" or "*simply followed system recommendation*" when building models. ML consultants described this topical knowledge as undocumented "*or on the fringes of ML textbooks*", and could only be acquired via years of ML experience.

Non-experts did not mention the procedural knowledge of ML either. Contrarily, ML consultants often preserved and reused the structure of their previously built ML solutions, and considered the pipeline design an important part of their expertise. This difference led to their distinct preferences for ML tools.

Preference for Machine Learning Tools

ML consultants uniformly and strongly preferred GUI-based, drag-and-drop tools that allow them to reuse their ML pipeline setup and application deployment codes, for example, knowledge-flow tools and scripting tools that support movable code sections. They thus could focus on implementing their learning algorithms with minimal code changes in other parts of the application.

In contrast, most non-experts expressed a strong preference for iML tools that take the form of a code editor. Except the one participant who does not program at all, non-experts prefer text-editor tools so that they can more easily re-use online scripts and ML solutions. Non-experts with formal training in software engineering particularly preferred these tools. Code-versioning and collaboration features of these tools (i.e. Git integration) enabled them to easily integrate various APIs, control versions, track bugs, and collaborate in real-time. They described these features as essential parts of their work. These observations are very different from the assumptions in previous literature, that non-experts are those who use GUI tools, while experts write customized code and use "*expert tools*" [16].

> *Writing code is just so much faster, more efficient. [...] After my consultant did all the exploratory work in [a knowledge-flow GUI tool], I always re-wrote them all in my [code] editor.*

Pitfalls of Machine Learning

Many non-experts successfully built ML solutions; they mapped algorithm in/outputs and groped their way towards a better model performance. This was effective in many modeling tasks, particularly the ones that do not require statistical reasoning or model generalizability. For example, P2 built an application that recognizes environmental noises and alerts ongoing events at home (i.e., microwave finished

and made a "ding" noise). He recorded microphone stream data, labeled segments of the stream with event names, and trained the model. The model functioned well only at his home, and this was good enough to be useful for him.

In other application domains, however, non-experts' I/O mapping approach seeded frustrations and performance problems. With little to no understanding of a learning algorithm's internal mechanism, non-experts were more often baffled when model performance stalls, and more often abandoned their tasks.

Designing an Achievable Learning Task

A vast majority of non-experts are unaware of the complexities in translating a problematic real-world situation into a feasible ML problem (namely, "problem design"). Often, they directly map the high-level goal they have in mind into the ML problem they have to design: "When I put in X, the application should return Y" (Figure 2). The data inputs in this scenario are the training data, and the output is the labels.

This strategy for problem design did not pertain to a particular learning algorithm's capability or the available data, and thus sometimes led to difficult or unrealistic learning goals. For example, P6 (a botanist) planned to build a signal classifier that detects plants' health status. She presumed to collect video streams of her plants as training data, and her strategy was to "*just collect the data that we (biologists) use now*".

In contrast, most of the ML consultants stated that formulating the right ML problem to solve is the greatest challenge in the whole process. They carried out "*long and difficult processes*" to gradually identify and articulate a meaningful and achievable goal.

They first exhausted simpler analytic methods: they ran various analyses to gain insights into the data, and then arrived on "*a decision point whether to proceed to ML*". If they decided to proceed, they then extensively investigated what real-world problems were worth solving and solvable by ML. When building a user-feedback classifier, for example, one ML consultant interviewed multiple levels of enterprise leadership to understand how the ML result would be used, how long it would stay active and the benefit it could generate. She factored these observations into the design and the evaluation criteria of her model.

Finally, ML consultants put explicit efforts in weighing "*how deep the learner can and should go*" (P5). In the previous analytic stage, they have accumulated some understandings of the data; at this stage they more carefully investigated data

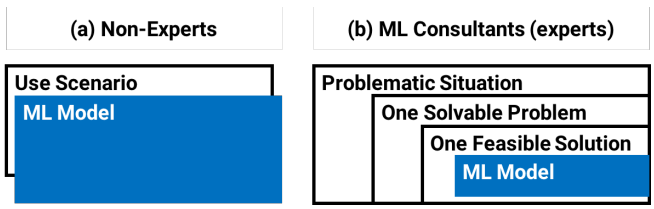

Figure 2. Non-experts' and their consultants' (experts) approaches to formulating an ML problem.

distribution and the extent of missing values. In dealing with familiar problems, consultants applied tried-and-true measures to quickly estimate the feasibility of their learning goal. For example, "*if a human cannot tell apart the intention of this sentence (class value), you have set the classes too fine-grained*" (P12). In dealing with more complex ML systems, participants also weighed the resources required before they start to build a model:

> *If I can predict 20% of these failures, that can save you how much. Is that gonna be worth our time and effort? If I can only predict 10%, is that enough to prove to you that you can invest a little bit more so that we can improve?* (P24, building predictive maintenance solution)

> *I estimated the amount of labeled sample data that we will need to accurately predict [topics]. There is a formula to calculate that [labeled data needed].* (P16, building text topic classifier)

In sharp contrast to non-experts' approach, ML consultants factored in all of the above considerations–the real-world situation, available data, resources and intended investments, and the capability of ML techniques–in their problem design. Their investigative processes took up to two months.

Debugging Strategically

Debugging refers to the iterative testing and revision of an ML model toward a certain level of performance [17]. Most non-experts' only strategy to improve a model's performance is to "*add more data*". Across all of the non-experts in our interviews and survey, none has ever removed or simplified any features while debugging. Very often non-experts ask their consultants "The model is not doing well. Should I add more instances or add more features?" As one ML consultant concluded: "*They just expect a white list of examples improves model performance.*"(P12)

Assuming adding more data directly improves performance, non-experts started training a model with all data and features available. When the performance stalled, they added more training inputs or turned to more complex models. After they exhausted both options, "*the next step is experimenting (with) deep learning*" (P3,18).

Some non-experts turned to API documentation to seek additional actions that they could take. However, many struggled with understanding the abstract concepts in the documents. One most-mentioned such concept is *feature*. Non-experts conceptualized features as concrete entities, such as columns of data in spreadsheet, or certain words and phrases to match in a text. We observed that this method frequently failed when the model involved composite or hierarchical features, or features that are difficult to interpret.

> *Whoever started using [an iML tool] that doesn't have ML background doesn't understand what a feature means, [...] to grasp the idea behind features.* (P12, ML consultant)

Although unclear about what a feature exactly is, non-experts examined the value of features by its impact on model performance. If the performance gets better, they added more

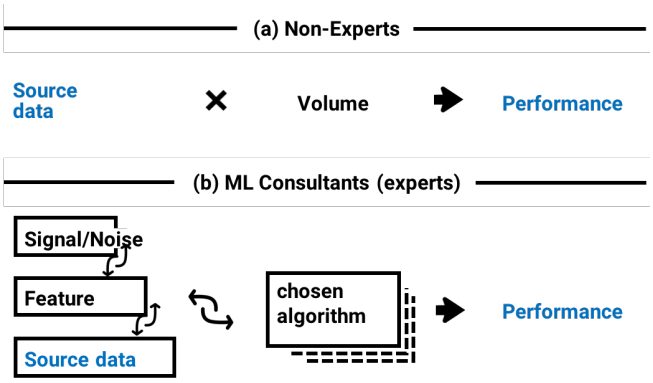

Figure 3. Non-experts' and their consultants' (experts) approaches to improve model performance.

features. Otherwise, they gave up on feature engineering altogether.

> *Feature is magic. I don't know what exactly is happening, but I can feel its impact.* (Survey response)

> *I wasted a week on it. It doesn't change anything. Feature is useless.* (P26, software engineer)

Contrarily, ML consultants starting model building with a "*solid, basic model*". As they evaluated the model solution, they weighed and decided whether to "*fancy it up or to add another component*". This incremental approach allowed them to debug only one component of the model in every iteration, "*so when it breaks you know why*". The most frequently mentioned debug "components" include: the choice of learning algorithm and its parameters, the choices of including a signal or not ("*is this information a signal or a noise?*"), the representation of the signal (using what and how many features?), and the design of a feature (Figure 3).

> *We used a very basic set of features there and there was a question, do we continue to make a fancier and fancier feature, or do we go after new signal. We weigh that constantly: Am I getting the right signal or wrong signal from the data source I have? If I am, should I add on another one, or should I continue trying to extract signals from the same set of information?* (P17, ML consultant)

While debugging, ML consultants constantly judged whether the power of a feature, signal or algorithm of certain complexity "*has saturated*", and took actions accordingly. They made these judgments based on an understanding of the affordances and limitations of the chosen algorithm, as well as monitoring metrics regarding data distribution and model performance. These knowledge and tools helped them probe into the interplay between different components of the model and improve its performance in a systematic manner.

Measuring and Interpreting Model Performance

Most non-experts consider their task completed once their model reached a "good enough" percentage accuracy. They expressed no concern over how well their solution generalizes to new data. None mentioned any over-fitting issues: The need of considering new, previously unseen data instances is not an activity non-experts actively engage. Some shared that they struggled with grasping unseen data in general, "the data that don't fit into my memory".

Almost all non-experts viewed percentage accuracy as the only measure of ML success. Our interviews captured a number of cases where non-experts deployed their ML application once the percentage accuracy "*looked good*", unaware of the likely problems and risks in their models such as bias and overfitting.

ML consultants, in contrast, expressed a constant alertness to the possible breakdowns of their ML solutions. "*Things can go wrong with no obvious reasons in ML*," one ML consultant said. With such a mindset, almost all ML consultants inserted "*checkpoints*" and "*gates*" into their processes, and closely monitored potential problems in the data or model. They regularly briefed each step of their ML process to subject matter experts or problem owners to assure their every step was statistically and practically valid. After deployment, they closely monitored incoming data and model performance for weeks "*just to see if it (the model) actually works*".

> *You shouldn't proceed to train the model if the quality of your data doesn't meet certain requirement, say, the percentage of missing data and so on.* (ML consultant)

DESIGN IMPLICATIONS AND TRADE-OFFS

We have depicted how non-experts build ML models for their respective purposes in real life. We intend to draw implications from these observations to inform future iML tool design and research. Yet we were alerted to the fact that participants in our study used publicly available toolkits rather than state-of-art iML systems in academic research. In this light, we searched HCI, ICML and AAAI publications for systems and techniques that intend to facilitate non-experts. We reflected on whether and how these tools and techniques could have improved non-expert ML. After much discussion, we found three issues where prior research and the empirical findings seem to be in conflict.

The first issue concerns non-experts' role in building ML models. Accessible iML tools to date are mostly intended to increase performance of a given model. This somewhat conflicts with the observation that non-experts proactively build models for their own purposes. We found their overall approach to be similar to that of a reflective practitioner [23]. They use their repertoire of ML knowledge to experiment with ML solutions for solving issues of practice. They continually assess the viability of the solution, and pivot when it seems to fail.

The second issue regards non-experts' knowledge of ML and tool use. Previous HCI research identified those who use GUI tools as non-experts [16]. Accessible ML tools also uniformly took the form of GUIs. Our empirical study provided an alternative perspective on these commonly held assumptions: Many non-ML experts used scripting tools to more easily make use of publicly available ML solutions. These solutions embedded implicit knowledge that non-experts find valuable, such as data pre-processing procedures and model evaluation methods. GUI tools do not offer such embedded knowledge,

thus were unattractive to many non-ML experts, especially to those who are proficient programmers. On reflection, we should not equate users' ML expertise with programming skills. Instead, we need to more carefully examine their needs in both ML and programming tasks respectively.

Lastly, ML tools should guide and safeguard users to build robust models. Our study revealed several technical pitfalls non-experts are susceptible to. This is a missing perspective in prior research on accessible ML, which has instead been focusing on making ML easily accessible to everyone. Guided by this framing, accessible ML tools frequently adopted simplified interfaces and functionality, hiding ML's complexities. The critical role of model quality control is left to the users. Such tools are unlikely to prevent the technical pitfalls of ML. When blindly used, they could even increase the risk of misleading or erroneous results.

These observations helped us shift the focus away from thinking of non-expert ML as being simply a problem of facilitating the needs of data scientists, or one of reducing tool complexities. While challenging, we need to make the process of building ML models *flexible*, *easy* and *robust*. Taking a lesson from the HCI work on end-user programming, we need to make tools for non-experts based on how they react to ML challenges, instead of by simplifying or appropriating expert tools for them.

This new framing led us to look for ideas where non-experts gained awareness of ML's technical pitfalls, and for opportunities to guide and safeguard non-experts to easily build robust models. But how? The trade-off between functional elasticity and ease of use poses a fundamental challenge for designing general-purpose, accessible iML tools [22]. To help both ourselves and fellow designers and researchers, we created an iML tool design as a sensitizing concept [31], demonstrating one possibility of creating novice-friendly yet robust iML processes.

A SENSITIZING CONCEPT

In the second stage of our work, we set out to design a sensitizing concept for accessible iML tools. We deliberately broke the norm and chose to create a sensitizing concept rather than elaborating on a set of discrete design implications. This is because our goal is to inform the HCI community of the *ready* opportunities in this research space, and to offer a starting place for future innovation. We had three goals for this design:

- Grounding its interaction design in non-experts' intuitive approach to ML;
- Scaffolding and safeguarding a robust model building process;
- Supporting users of various needs and skillsets, both in terms of ML knowledge and programming skills.

Toward these goals, we sketched various possible iML tool forms and functions. We also synthesized technical advances in HCI and ML research. We mapped them onto the workflow and pitfalls our empirical study has revealed and discussed how they might benefit non-experts. This process resulted in a new design, namely "*Test-Driven Machine Teaching*".

The Concept of Test-Driven Machine Teaching

Test-driven machine teaching is the interaction flow of an iML tool. It takes inspirations from two technical concepts: "Machine Teaching" [24] and "Test-Driven ML" [4].

With *Machine Teaching*, we refer to a human-centered perspective to the process of ML [24]. While ML focuses on improving the accuracy of "learners" (learning algorithms), machine teaching focuses on improving the efficacy of "teachers" –people who interact with data and "teach" how algorithms should behave. We adapted this focus in the design of our iML tool. Instead of aiming to better explain algorithm behaviors to users, our design aims to help users better express their expectations towards ML, and guide users towards features and algorithms that match their expectations.

Test-Driven ML is the practice of defining test cases *before* model building. Users first define examples of model inputs and outputs as test cases, then build a simple model to get one test to pass, and then improve the model until passing all tests [4]. This approach has several benefits: 1) It documents how the models are intended to work; 2) It reduces user errors; 3) It helps novices learn to start with simple solutions and improve incrementally [15]. Given these potential benefits for non-experts, we designed the iML tool to scaffold a test-driven ML process.

Interaction Flow (Figure 4)

Before training learning models, the iML tool displays data and *requires* users to elicit a set of observations as test cases. Users curate a test set that encapsulates their ML goals and priorities. This hand-picking process forces users to examine their data, grounding their ML goals in concrete data instances. It also facilitates those who "*want ML but do not know what exactly for*" to crystallize their expectations.

Accordingly, the iML tool alerts users of the potential skews and biases in their selections, guiding them to understand their data more comprehensively and to curate a statistically representative test set. For example, our empirical study showed that non-experts tend to select a "white list" of test cases (cases whose class values are uniformly positive). The iML tool could remind the user to also provide negative examples.

Next, the iML tool recommends features and learning algorithms that are likely to work well with the characteristics of the data, the prediction task, and user priorities captured from the test set. It also alerts users of significant data quality flaws. For small datasets, the iML tool may automatically run multiple basic algorithms, and suggests a subset based on standard performance measures such confusion matrix and F score. We imagine ML best practices and existing mixed-initiative ML techniques could be leveraged in support of this (i.e. [6, 28, 32]).

After a model is built, the iML tool shows model predictions on all test cases. In contrast to statistics and visualizations whose interpretation requires additional know-how, these hand-picked test cases provide concrete, direct probes into the data and the model performance. They also represent the subset of data instances that the user understands and cares

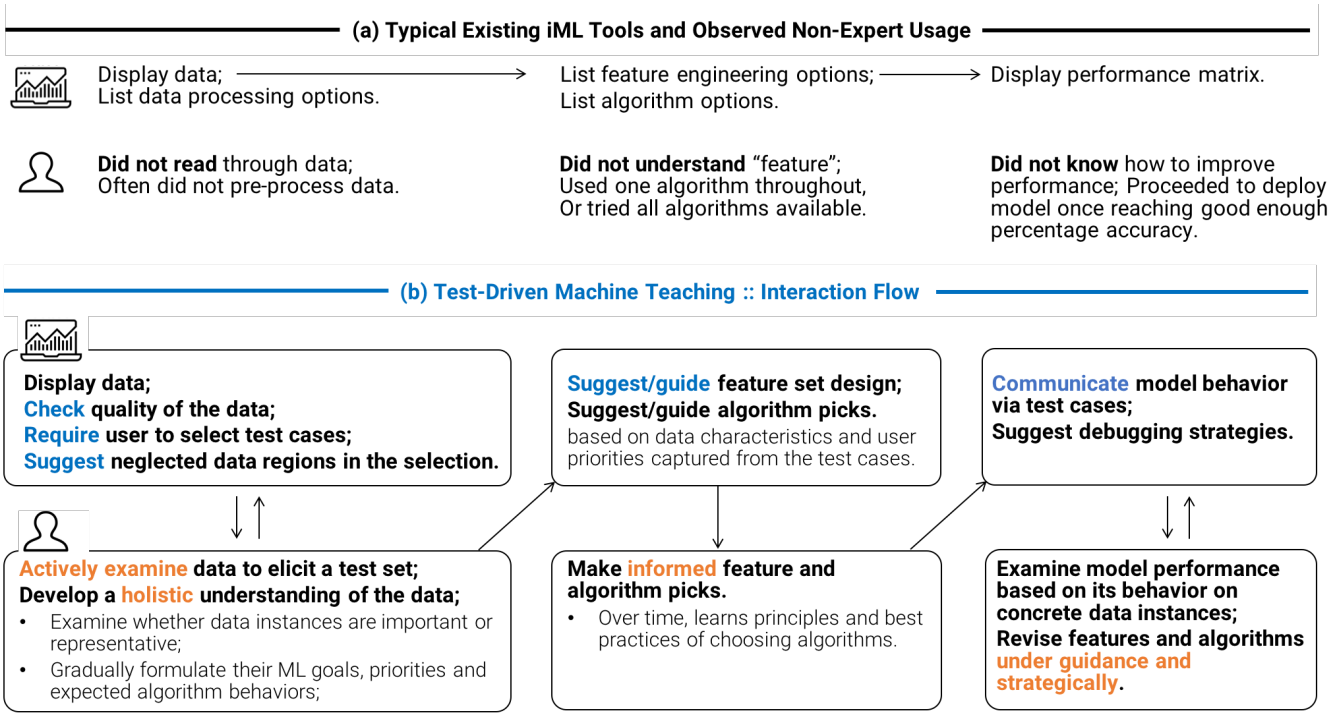

Figure 4. Test-driven machine teaching workflow. It uses user-selected test cases as the major interface between participants and learning algorithms. Central to this design is the idea that novice-facing iML tools should not only make it easy to teach ML models, but also actively and intelligently support robust teaching processes and activities.

about. We imagine users are more likely to engage with and be able to understand algorithm behaviour through these test cases. Central to this design is to prevent users from taking percentage accuracy as the sole performance measure.

While communicating model behaviours, the iML tool also suggests debugging strategies and actions for users to take. One simple way to generate these suggestions is to automatically run principal component analysis, informing users of strongly relevant features. Previous research on explanatory debugging, mixed-initiative ML and automatic ML techniques can inform more sophisticated debugging suggestions [6, 13, 17, 32].

This interaction flow can take various UI forms. Designers may choose one or more interface manifestations of this interaction flow, based on their intended users or typical ML tasks.

Actualizing the Design Implications

Test-driven machine teaching (TDMT) functions as a sensitizing concept for designing accessible ML tools. It exemplifies one concrete solution for the design trade-offs among functional flexibility, ease of use and robust ML results. It embodies three implications for mediating these trade-offs in the design of accessible ML tools.

1. *Test cases as the interface between non-experts and learning models*: TDMT captures how users expect the algorithms to behave via test cases, helping users better articulate these expectations. Such articulation enables the iML tool to offer personalized, in-situ safeguard mechanisms and guidance in the modeling process.

   This design is rooted in the observation that non-experts understand model behaviour by mapping inputs and outputs. Test cases as a form of model input-output pairs fit how non-experts intuitively understand ML.

   Moreover, the process of hand-picking test cases invites users to carefully examine their data and their expectations toward ML. The empirical study showed that non-expert ML is highly experimental and exploratory. TDMT thus guides users to examine their data and formulate ML goals that should be part of the ML process. At a higher level, accessible ML tools should support non-expert ML as a co-evolution of problem and solution, rather than as a gradual procession to an optimum in the loss function.

2. *Scaffolding and safeguarding a robust ML process by blending user and machine intelligence*:

   Clearly, there are opportunities in embedding data-driven techniques in iML tool design to make non-expert ML more robust. Research has produced many techniques that improve amateurs' ML reliability and efficiency [30, 6, 8]. Previous crowd-sourcing work [1, 20] as well as our empirical study have offered observations of how non-experts work with ML. IML tool design can be a perfect fusion point of these parallel research efforts in making ML accessible to non-experts.

   TDMT offers one example of this. It leverages test cases to capture user expectations toward ML. The iML tool thus can

deliver personalized, ad-hoc suggestions targeted to a user's goal and the dataset in-use. Future research is necessary to evaluate this concept, and more importantly, to explore solutions to bridge user and machine intelligence via the design of iML tool.

3. *Adaptive interfaces to support users with various needs and background knowledge*: Previous iML research claimed that general-purpose ML tools require complex functionality and interactions; thus they are difficult to design for non-experts [22]. We see opportunities in adaptive user interfaces (UI) and interactions to help with this challenge. As an example, TDMT offers intelligently pliable functions, surfacing only the data procedures and algorithm choices that are relevant to the user task at the moment.

   TDMT is also pliable for various UI forms. As the empirical study showed, we cannot assume ML novices are those who use GUI tools [16]. Designers should deliberately choose one or more interface forms based on their target audience's tooling, mental model, data types and ML tasks they are likely face. At a higher level, there needs be a more reflective and principled discussion on the relationship between ML tools for those who are not experts in ML and tools for those who do not code.

DISCUSSION

ML promises powerful data-driven insights and solutions for people from all walks of life. Emergent ML and HCI research aim to make ML solutions easy to build and accessible to non-experts. Adding to this growing body of research, this work investigated how non-experts build ML solutions for themselves. We further suggested that iML tools might play a more vital role in making ML accessible if it helps non-experts prevent technical pitfalls. We put forward a sensitizing concept to exemplify one near-future solution in this new design space. We encourage the HCI community to join us in evaluating and evolving this design, and more importantly to innovate on more creative forms of user-centered, accessible iML tool designs. Here, we also want to more broadly reflect on our work and the idea of *ML for everyone*.

Making Machine Learning Accessible

The idea of human-centered ML is not new to the research that aims to make ML available to non-experts. Yet this field has largely focused on technical advances. The major vehicle that drives the work to understand end users is the need for more efficient and reliable "human oracles" that compliment automatic ML methods [1, 18, 24].

Through the empirical study and the sensitizing concept, we hope to formulate some changes to the research on accessible ML. We hope to start a discourse centered on how to help people better build learning algorithms, rather than how to enable the algorithms to learn better. We hope more human-centered ML research not only focuses on non-experts' affordability and objective efficiency, but is rooted in an elaborate understanding of their goals, needs, and the contexts in which they use ML.

An elaborate understanding of non-experts could not only make ML more accessible, but inspire creation of new ML applications. For example, our empirical findings suggested that the interactive process of building ML models might have lead to non-experts' greater trust in the learning results. We see critical value in evaluating this observation, as it could potentially open up a new path to improving end-users' trust in ML.

Risks and Limits of Non-Expert Machine Learning

In parallel to the work that makes ML accessible, there should be research to better understand the risks and limitations of non-expert ML. This work has illuminated some signs of ML misuse and its preliminary breakdown by type. Accessible ML tools, if used blindly, could cause perplexities.

One straightforward solution is to embed additional corrective measures in these tools, as we have shown via the sensitizing concept. Future research could extend, evaluate and improve these measures to safeguard the validity and quality of ML models. Additionally, when creating an accessible ML tool, we HCI researchers could more explicitly document its intended users, intended ML tasks, as well as the situations in which its use is not recommended.

As the industry is increasingly calling for "*ML for everyone*", HCI research could provide a reflective and principled discussion on whether this vision is achievable or desirable. Can we identify some general criteria such that, if satisfied, we would agree this is an ML task no longer achievable for a non-expert? Can we identify some basic principles of designing novice-facing ML tools, to more easily and objectively identify the level of ML knowledge needed for their proper use? These are important issues for our community to grapple with and debate moving forward.

Non-expert ML is an exciting area in both industry and HCI research. Having a reflective discussion on the potentials, risks, and limits of non-expert ML would be a vital next step in our collective endeavor to make ML accessible in moving forward. This work serves as one step into this direction.

REFERENCES

1. 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. DOI: http://dx.doi.org/10.1609/aimag.v35i4.2513

2. Hugh Beyer and Karen Holtzblatt. 1999. Contextual design. *interactions* 6, 1 (1999), 32–42.

3. Bonsai. 2017. *BONSAI WHITEPAPER: A fundamentally different approach for building intelligent industrial systems*. Technical Report. https://bons.ai/resources.

4. Justin Bozonier. 2015. *Test-Driven Machine Learning*. Packt Publishing Ltd.

5. Joseph Chee Chang, Saleema Amershi, and Ece Kamar. 2017. Revolt: Collaborative Crowdsourcing for Labeling Machine Learning Datasets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 2334–2346.

6. Justin Cheng and Michael S. Bernstein. 2015. Flock: Hybrid Crowd-Machine Learning Classifiers. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW '15*. ACM Press, New York, New York, USA, 600–611. DOI: http://dx.doi.org/10.1145/2675133.2675214
7. Jerry Alan Fails and Dan R Olsen Jr. 2003. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*. ACM, 39–45.
8. Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*. 2962–2970.
9. Rebecca Fiebrink, Perry R. Cook, and Dan Trueman. 2011. Human model evaluation in interactive supervised learning. In *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11 (CHI '11)*. ACM Press, New York, New York, USA, 147. DOI: http://dx.doi.org/10.1145/1978942.1978965
10. Marco Gillies, Bongshin Lee, Nicolas D'Alessandro, Joëlle Tilmanne, Todd Kulesza, Baptiste Caramiaux, Rebecca Fiebrink, Atau Tanaka, Jérémie Garcia, Frédéric Bevilacqua, Alexis Heloir, Fabrizio Nunnari, Wendy Mackay, and Saleema Amershi. 2016. Human-Centred Machine Learning. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '16*. ACM Press, New York, New York, USA, 3558–3565. DOI: http://dx.doi.org/10.1145/2851581.2856492
11. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter* 11, 1 (2009), 10–18.
12. David J Hand. 1998. Data mining: Statistics and more? *The American Statistician* 52, 2 (1998), 112–118.
13. Andreas Holzinger, Markus Plass, Katharina Holzinger, Gloria Cerasela Crişan, Camelia-M Pintea, and Vasile Palade. 2016. Towards interactive Machine Learning (iML): applying ant colony algorithms to solve the traveling salesman problem with the human-in-the-loop approach. In *International Conference on Availability, Reliability, and Security*. Springer, 81–95.
14. IBM Watson Machine Learning: Machine Learning for everyone 2016. (2016). http://datascience.ibm.com/blog/machine-learning-for-everyone/
15. David Janzen and Hossein Saiedian. 2005. Test-driven development concepts, taxonomy, and future direction. *Computer* 38, 9 (2005), 43–50.
16. Seungjun Kim, Dan Tasse, Anind K Dey, S Kim, D Tasse, and A K Dey. 2017. Making Machine-Learning Applications for Time-Series Sensor Data Graphical and Interactive. *ACM Trans. Interact. Intell. Syst* 7, 8 (2017). DOI:http://dx.doi.org/10.1145/2983924
17. Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. 2015. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*. ACM, 126–137.
18. Besmira Nushi, Ece Kamar, Eric Horvitz, and Donald Kossmann. 2017. On Human Intellect and Machine Failures: Troubleshooting Integrative Machine Learning Systems.. In *AAAI*. 1017–1025.
19. 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*. ACM Press, New York, New York, USA, 37. DOI: http://dx.doi.org/10.1145/1866029.1866038
20. Kayur Patel, James Fogarty, James A. Landay, and Beverly Harrison. 2008. Examining difficulties software developers encounter in the adoption of statistical machine learning. In *23rd AAAI Conference on Artificial Intelligence and the 20th Innovative Applications of Artificial Intelligence Conference*. Chicago, IL, United States, 1563–1566.
21. Kayur Dushyant Patel. 2012. *Lowering the Barrier to Applying Machine Learning*. Ph.D. Dissertation. University of Washington.
22. Kayur Dushyant Patel. 2013. *Lowering the Barrier to Applying Machine Learning*. Ph.D. Dissertation.
23. Donald A Schon. 1984. *The reflective practitioner: How professionals think in action*. Vol. 5126. Basic books.
24. Patrice Simard, Saleema Amershi, Max Chickering, Alicia Edelman Pelton, Soroush Ghorashi, Chris Meek, Gonzalo Ramos, Jina Suh, Johan Verwey, Mo Wang, and John Wernsing. 2017. *Machine Teaching: A New Paradigm for Building Machine Learning Systems*. Technical Report. Microsoft Research. https://arxiv.org/pdf/1707.06742
25. Simone Stumpf, Vidya Rajaram, Lida Li, Weng Keen Wong, Margaret Burnett, Thomas Dietterich, Erin Sullivan, and Jonathan Herlocker. 2009. Interacting meaningfully with machine learning systems: Three experiments. *International Journal of Human Computer Studies* 67, 8 (2009), 639–662. DOI: http://dx.doi.org/10.1016/j.ijhcs.2009.03.004
26. TensorFlow: Smarter machine learning, for everyone 2016. (2016). https://www.google.com/intl/en/about/main/tensorflow/.
27. Udemy Online Course: Applied machine learning for Everyone 2017. (2017). https://www.udemy.com/applied-machine-learning-for-everyone/.
28. Steven A Wolfman, Tessa Lau, Pedro Domingos, and Daniel S Weld. 2001. Mixed initiative interfaces for learning tasks: SMARTedit talks back. In *Proceedings of*

the 6th international conference on Intelligent user interfaces. ACM, 167–174.

29. Qian Yang, John Zimmerman, Aaron Steinfeld, and Anthony Tomasic. 2016. Planning Adaptive Mobile Experiences When Wireframing. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems - DIS '16*. ACM Press, Brisbane, QLD, Australia, 565–576. DOI:http://dx.doi.org/10.1145/2901790.2901858
30. Xiaojin Zhu. 2015. Machine Teaching: An Inverse Problem to Machine Learning and an Approach Toward Optimal Education. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*. AAAI Press, 4083–4087. http://dl.acm.org/citation.cfm?id=2888116.2888288
31. John Zimmerman, Erik Stolterman, and Jodi Forlizzi. 2010. An analysis and critique of Research through Design: towards a formalization of a research approach. In *Proceedings of the 8th ACM Conference on Designing Interactive Systems*. ACM, 310–319.
32. Martin Zinkevich. 2017. Rules of Machine Learning: Best Practices for ML Engineering. (2017).