Using Expert Patterns in Assisted Interactive Machine Learning: A Study in Machine Teaching

Emily Wall[1], Soroush Ghorashi[2], and Gonzalo Ramos[2]

[1] Georgia Tech, Atlanta, GA
emilywall@gatech.edu
[2] Microsoft Research, Redmond, WA
{sorgh,goramos}@microsoft.com

Abstract. Machine Teaching (MT) is an emerging practice where people, without Machine Learning (ML) expertise, provide rich information beyond labels in order to create ML models. MT promises to lower the barrier of entry to creating ML models by requiring a softer set of skills from users than having ML expertise. In this paper, we explore and show how end-users without MT experience successfully build ML models using the MT process, and achieve results not far behind those of MT experts. We do this by conducting two studies. We first investigated how MT experts build models, from which we extracted expert teaching patterns. In our second study, we observed end-users without MT experience create ML models with and without guidance from expert patterns. We found that all users built models comparable to those built by MT experts. Further, we observed that users who received guidance perceived the task to require less effort and felt less mental demand than those who did not receive guidance.

Keywords: Interactive Machine Learning · Machine Teaching · User Studies.

1 Introduction

Over the past decades, the Machine Learning (ML) field has devoted its attention to the study of algorithms that extract knowledge from data. It is common to hear today about solutions where machines can make predictions with almost, or better-than, human precision. This success comes at a cost: building these ML models requires an expert model builder with knowledge of the underlying learning algorithm. Further, creating such solutions often requires large amounts of pre-labeled data, which might be easy or practical to obtain for certain problems (e.g., Google Photo's image labeling), but not others (e.g., face detection in a personal photo collection).

As the ML field considers addressing an emerging set of problems, it faces the challenge of meeting a growing demand of being an accessible tool for creating models. For example, a paralegal may want to sift through and classify tens of thousands of legal documents according to a concept unique to a case; or a developer may want to create an app that classifies and filters news feeds according to the user's preferences. Instead, access to the specialists who can build ML models like the one described above is limited by their scarcity and cost.

In addition to the above challenges, there is a growing demand [16] for explainable [11] or intelligible models [35] to not only enable accountability, but also to facilitate model inspection and debugging [27]. To address some of these issues, interactive machine learning (iML) has emerged as a field at the intersection of HCI and ML that focuses on ML model-building interactive processes with a human-in-the-loop [1].

Machine Teaching (MT) [34] is an emerging perspective on iML that is complementary to the discipline of ML. It aims to address the aforementioned challenges by extending iML's notions of interactivity, and by focusing on information exchange between a human (teacher) as a non-ML expert who has rich, useful knowledge beyond labels. In particular, a *machine teacher* can be an active participant choosing elements to populate the training set, labeling them, and choosing when and how to fix prediction errors by creating semantically meaningful features influenced by what they have seen. As teachers are active protagonists in this teaching process, they can take different paths to teach a model. This is akin to the many ways there are for a programmer to implement a function. For the scope of this paper we will use the term *machine teacher* or *teacher* to denote the subject-domain expert who trains an ML model using MT's process.

While MT has not yet been widely adopted, we believe it is a promising approach to help subject-matter experts build ML models that are accessible, scalable, and intelligible. Part of its appeal is that building a model using the MT process requires from the teacher expertise about a subject-domain, and no knowledge about the details, such as type and parameters, of the underlying learning algorithm. This set-up leads to a process requiring a softer skillset than traditional ones that require ML knowledge. In this paper, we focus on exploring and showing how the MT process enables end-users (MT and ML novices) to construct ML models comparable to those built by MT experts through two studies in which participants construct binary classification models of text documents.

This work makes the following contributions. First, we conduct a qualitative study in which we characterize MT phases and expert patterns by studying how current MT experts teach. Second, we encode these patterns and practices into an MT system to guide end-users. Third, we conduct an experiment that shows (a) that given minimal prior training in ML and MT, novices are able to construct models of similar quality to those built by experts, and (b) that guiding novices with the aforementioned expert practices helps them to approach expert teacher behavior and build models with less effort than novices without such guidance. Fourth, to our knowledge, this is the first paper that presents a study of end-users engaging in a MT process or loop, through a system that instantiates MT as stated by [34]. Finally, we discuss additional MT teaching patterns and design considerations for MT systems, as next-steps for researchers and designers to consider.

2 Background and Related Work

Interactive Machine Learning and Teaching. Our work takes place in the context of a supervised iML workflow where people build ML models by providing knowledge in an interactive loop [1, 12]. In this flow, it is worth noting three main activities where people can actively express knowledge: choosing what to label (sampling), labeling, and featuring. Sampling can help the learning system see an example of something it has not seen before, and thus can learn from; labeling taps into a person's subject-domain knowledge to provide the learning system with sources of truth; and featuring enables people to identify or select the properties that can help the learning system improve its internal representation of the concept one is trying to teach. Most prior work has focused on the latter two activities in isolation, where people either (1) label while keeping features constant [13, 14, 17], or (2) feature while keeping labels constant [5, 8, 24]. Others have evaluated alternative strategies within the iML loop [4]. Only a few works have looked into systems where people provide both labels and features [3, 10]; even fewer have looked into people choosing a sampling strategy to interactively build and refine the training set [7]. Our work looks at the iML loop holistically where the above three activities occur in concert, and with a deeper emphasis on human interaction. We do this within the framing of machine teaching, as defined by [34], a point of view we will expand in the next sections.

In addition to the above, the term machine teaching has been used in different contexts. Zhu et al. [38] refer to machine teaching as the inverse problem to machine learning. In this definition, the problem is about finding the optimal training set, given a particular learning algorithm and a target model. This non human-in-the loop definition addresses a different problem from the one we are trying to solve. Our focus is about facilitating the interactive extraction of knowledge from a human teacher, towards the building of a model, while not knowing details about the learning algorithm under the hood.

Democratizing ML. The promises and enthusiasm around ML fuel the desire to make it a tool that everybody can either create or use. Many efforts actively pursue and address these goals. We group these efforts to make ML accessible into two schools of thought: (1) making it easier to become an ML expert, or (2) giving end-users access to tools that hide the complexities of interacting with an ML algorithm. In the former category, places like Udacity [3] provide courses to train individuals into ML engineers, but these solutions address a different form of democratization. We are not interested in a subject-domain expert having to go to school in order to solve an immediate problem, or one that happens rarely. Instead, we focus on changing the curricula of what an end-user needs to know to create an ML model, by abstracting the complexities of ML algorithms.

There are too many tools and experiences to enumerate that claim to lower the barrier for the general public of problem-owners to use ML solutions. Places like Amazon's Web Services ML, Microsoft's Azure ML, or Google's Cloud AI provide visual front-end solutions for the "easy" creation and deployment of ML models. Other tools hide the complexity of ML trainers in the form of end-user libraries such as Scikit-Learn [33], ml5.js [4], or turicreate [5]; or visual tools such as prodi.gy [6] or lobe.ai [7]. Some tools exist to support model-building including both data and coding capabilities [32]. Many of these tools succeed in different ways at reducing the amount of detailed ML knowledge

[3] https://www.udacity.com

[4] https://ml5js.org/

[5] https://github.com/apple/turicreate

[6] https://prodi.gy/

[7] https://lobe.ai/

needed in order to use them, but do not fully remove the need to be familiar with the underlying ML algorithms or theory.

Automated Machine Learning (AutoML) is a time-saving alternative designed to automate parts of the ML pipeline, including hyperparameter selection [21], feature engineering [25], and so on. Some work has begun to combine parts of this process [30, 31], and while AutoML can lower the barrier to building ML models, it is still subject to the same struggles of training ML models in the traditional way. For example, automated feature engineering can come at the cost of model interpretability, since features may not be semantically meaningful. Furthermore, AutoML still requires significant labeled data, not readily available for many problems, and is hence not a catch-all solution to democratizing ML. In contrast, our efforts seek to combine the above efforts: to educate and guide subject-domain experts to be good machine teachers, and do so within a tool that hides the need to know the learner algorithm's details.

Helping Novices Use (Complex) Software. Fraser et al. [15] provide a good summary of the different strategies that can be used to help novices effectively use software. In particular, this work discusses the pros and cons of (online) tutorials, enabling expert functionality over time, and command, or task-level action suggestions. Our work is inspired by this last approach of a system providing task-level action suggestions, but applied to a different context than the aforementioned work.

Notifications are a common way to push timely information (like suggestions) to users across a wide range of applications. When designing a notification system, many factors need to be taken into consideration. Horvitz [18] presented twelve principles for mixed-initiative user interfaces that guide the ways systems balance human and machine efforts. For us, principles such as considering a user's goals and attention, minimizing cost of guesses, and maintaining working memory are particularly relevant to our thinking for notifications to help novice teachers. A significant number of works also look into identifying the proper time to show notifications by building statistical and ML models of users engaged in particular task flows [19, 22, 23]. We choose to use notifications to guide novice teachers utilizing heuristics based on moments around specific actions teachers take, as we wanted a first approximation to a reasonable solution.

3 Machine Teaching

Our introduction underlines how ML's current renaissance is also accompanied by significant challenges. MT as described in [34] is an emerging discipline that has the promise to address many such challenges. As a form if iML, MT is inherently a supervised learning process that can be applied to classification and entity extraction problems for different types of data such as text, structured documents, or images. We have not consider the MT process's suitability to address regression problems, a topic that falls outside the scope of this paper. MT sits at the intersection of HCI and ML, focusing on the exchange of knowledge between a (human) teacher and a (computer or algorithm) learner. According to Simard et al.'s [34] vision, MT systems are at their core iML systems, and follow a process and philosophy of incremental iteration, stating a point-of-view about how teaching (transferring human knowledge) to a learning algorithm should be done. MT's aim is not only making the process of creating an ML model accessible

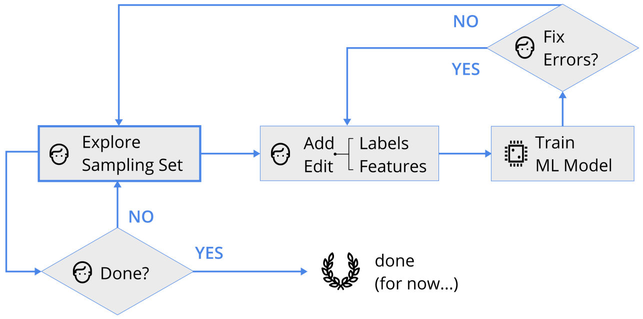


Fig. 1: Machine Teaching Loop. One often starts this loop by exploring the sampling set.

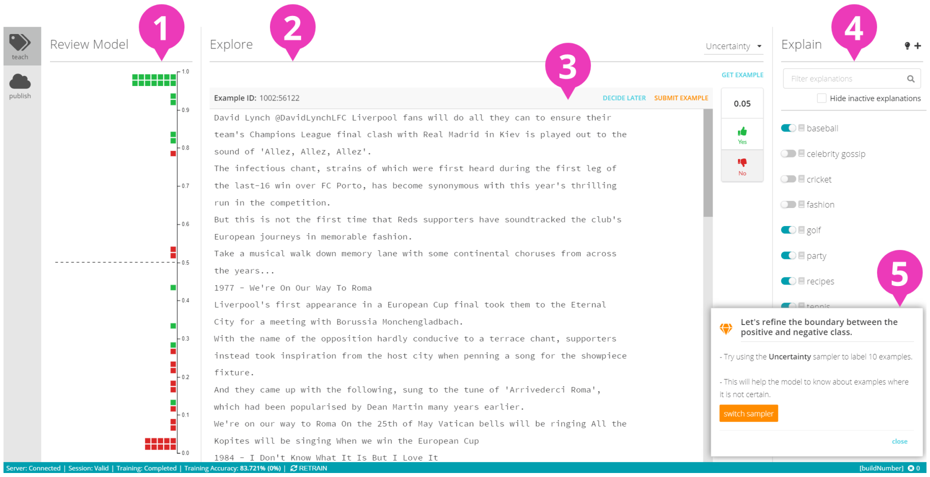
and efficient for the teacher, but also producing models that are intelligible by design. MT targets domain-subject experts as its users by abstracting the complexities of the interface of the ML algorithm and its parameters.

Teachers express their domain knowledge through decisions they make while teaching. They do so in ways that hide and abstract the parameters of a learning algorithm. At a fundamental level, all teachers need to know how to do, is to *explain why they have labeled a document as a member of a certain class*. For MT to be effective, some conditions need to be met. First, having a consistent, underlying learning algorithm guarantees that teachers will only encounter three types of errors (mislabeling errors, learner errors, and representation errors), each with a known remedy [29]. Second, a person should be able to articulate the relevant concepts that explain why an example is (or is not) a member of a class. Third, the teacher has access to a searchable and large set of unlabeled data (sampling set). As teachers use this process to build an ML model, they are participating in a teaching loop.

During this MT loop, there is no predetermined test or ground-truth set a teacher can use to assess the quality of their model. Evaluation in this case can happen by judging the number of correct predictions in a dynamically-generated set of positively predicted documents. This type of context is where MT shows the most promise, and it is not uncommon. ML services such as LUIS [8] are an example of MT in the wild where examples, labels, and features come incrementally from a subject-domain expert. The above context stands in contrast to cases where one has access to a large set of pre-labeled documents, such as images labeled through a CAPTCHA service. These later cases can be best-served with unsupervised ML methods and are not something MT aims to solve.

Figure 1 illustrates the teaching loop, and the types of teacher knowledge exchange that take place. This teacher knowledge comes in the form of choosing what example to label next; labeling an example; creating features; and detecting training errors, assessing them, and fixing them. For example, following this loop, a teacher creating a Sports news classifier from scratch (no labels or features) can (1) search for articles containing the name “Lionel Messi”, (2) label them positive or negative according to their content

[8] http://luis.ai



Example ID: 1002:56122

```
David Lynch @DavidLynchLFC Liverpool fans will do all they can to ensure their
team's Champions League final clash with Real Madrid in Kiev is played out to the
sound of 'Allez, Allez, Allez'.
The infectious chant, strains of which were first heard during the first leg of
the last-16 win over FC Porto, has become synonymous with this year's thrilling
run in the competition.
But this is not the first time that Reds supporters have soundtracked the club's
European journeys in memorable fashion.
Take a musical walk down memory lane with some continental choruses from across
the years...
1977 - We're On Our Way To Roma
Liverpool's first appearance in a European Cup final took them to the Eternal
City for a meeting with Borussia Monchengladbach.
With the name of the opposition hardly conducive to a terrace chant, supporters
instead took inspiration from the host city when penning a song for the showpiece
fixture.
And they came up with the following, sung to the tune of 'Arrivederci Roma',
which had been popularised by Dean Martin many years earlier.
We're on our way to Roma On the 25th of May Vatican bells will be ringing All the
Kopites will be singing When we win the European Cup
1984 - I Don't Know What It Is But I Love It
```

Let's refine the boundary between the positive and negative class.

- Try using the **Uncertainty** sampler to label 10 examples.

- This will help the model to know about examples where it is not certain.

Server: Connected | Session: Valid | Training: Completed | Training Accuracy: 83.721% (0%) |

Fig. 2: MATE's main screen. (1) Model tracker. (2) Sampling interface. (3) Labeling interface. (4) Featuring interface. (5) Notifications (added after Study 1).

(i.e., add to the training set), (3) (after the system retrains a new model) see if articles' labels coincide (or not) with the current model's predictions (i.e., detecting prediction errors), (4) create a feature or explanation to the system the concept of "soccer" as articles containing the word "soccer", and (5) (after the system retrains a new model) see that the training set is predicted correctly. This loop showcases how teachers can be more [1] than just a source of labels.

MATE System. We created a system, MATE (MAchine TEaching), that implements the above MT flow. This implementation of MATE supports binary classification, multi-class classification, and entity extraction for text documents. However, for the purposes of our study, we scope our description of the system and subsequent studies to binary classification of text documents, as such models often form the building blocks of more complex predictive models. The details of this implementation fall outside of the scope of this paper, hence we will only focus on its main functionality and interface. The MATE system's interface is divided into three areas. Its center area has a sampler selector (Figure 2-2) that lets teachers decide how to get the next document to label. MATE provides six sampling or exploration strategies for teachers, described in Table 1. These strategies are a representative set of the different ways of identifying useful examples to show to the learning algorithm.

The center area also displays documents and lets teachers *label* them as a positive (yes) or negative (no) example of the main class (Figure 2-3). Teachers can postpone their decision by "deciding later" (i.e., to support concept evolution when the teacher is uncertain about whether a document should be positive or negative [26]). On the left side, teachers can examine and review training errors using a *model tracker*-like control [2] which displays all positive (green), negative (red), and undecided (blue) labels (Figure 2-1) sorted vertically by prediction score. This panel lets teachers select and review previously labeled (or undecided) documents. The right side of the interface allows teachers to create or edit explanations (*features*) by defining lists of *keywords* (Figure 2-4). When teachers see a document, the system highlights the words within that document that match keywords contained in each feature. Hovering over a highlight lets teachers see what features fire on a particular keyword or phrase.

Sampler	**Description**
KS: Keyword Search	Get a document containing one or more keywords.
RS: Random	Get a random document.
US: Uncertainty	Get a document near the decision boundary.
EBS: Explanation-Based	Given a feature (explanation), get a document where it activates.
PES: Possible Errors	Get a document the model is likely predicting incorrectly, when contrasted with a bag-of-words model.
PPS: Predicted Positives	Get a document the model predicts as positive.

Table 1: Sampling Strategies supported by the MATE system.

Teaching Skills as Problem-Solving Skills. There are interesting parallels between the practice of MT and programming. For example, a predictor is a function that a person ideates (main concept), codes (sampling, labeling, and featuring), and debugs (detect and fix training errors). These similarities underline that good teachers need to have a certain mindset to problem solving, e.g., how they decompose and explain a concept using a determined language. Problem-solving skills are not something one is born with; one develops such skills over time through training and experience. Unlike programming or software engineering, MT expert patterns are yet to be codified and evaluated; i.e., there is no unique way to move through the flow defined in Figure 1. In the next section we describe how we seek to encode these expert patterns.

4 Study 1: Modeling Expert Patterns

We aim to improve and accelerate the practice of MT by helping ML and MT novices become good machine teachers. In order to do that, we first seek to understand teaching practices among MT experts. In this qualitative study, we observe MT experts building ML models using the MATE system and analyze the rationale for their teaching decisions.

Participants. One of the challenges of this study is to encode expert behavior and decision making in an emerging area whose known expert user population is small. For our studies, we had access to a group at a research institution that developed the MT principles and has experience using them for the past four years. The ML models built by this group are used internally in a large software company. For our study, we recruited six individuals (two female) from the above pool of experts, who have more than 1 year of experience building ML models following the MT process. The average age for the participants is 37. We compensated participants with a $10 cafeteria credit and gave the incentive that the person who built the best model (according to an F-score) would receive an additional $25 Amazon gift card.

Materials. Participants used the MATE tool, described in Section 3, loaded with a dataset containing 90,000 documents scraped from recent news articles and blog posts from a data-aggregation service. [9] The documents belonged to sources classified as finance, health, sports, movies & TV, music, politics, travel, and undefined.

Procedure. We conducted the study in two separate one-hour long sessions. In the first session, participants used the MATE tool, which they were already familiar with from significant prior use, to build an ML model using the MT process. Participants spent 35 minutes building a binary classifier for articles about movies & TV. We observed and collected screen and audio recordings, in order to avoid interrupting or altering the experts' teaching flow. After the first session, we codified the videos on a data sheet marking important moments in each participant's teaching process. This included when they changed their method of sampling and labeling the data, when they chose to review training errors, when they created or modified features, or when their process seemed to have otherwise shifted, etc.

We interviewed each participant for a second session that took place 2-3 days after the first one, to permit time for us to codify the session video and so that the expert participants could reflect on their process while their teaching experience was fresh on their minds. During this session, we used our notes from the codified videos of the first session to ask participants about their rationale for a particular teaching decision or apparent change in behavior. To assist participants with this activity, we replayed the corresponding video from the first session to jog their memory as needed.

5 Study 1: Results

Participants (expert teachers) generated an average of 69 labels (SD=27) during their 35-minute sessions. They created an average of 76 keywords (SD=98) across 7 features (SD=4). The high standard deviation in keywords was due to participant S6 who searched the web for a list of 248 recent movie titles. Figure 3 summarizes experts' interactions over time, including the samplers used (colored rectangles), when they reviewed labeled documents, including errors (in red), from the model tracker (+ and - symbols), and when they created or modified features (circle symbols). In the following, we highlight the key teaching phases and expert practices we found in the study.

Machine Teaching Phases. From this study, we observed four high-level phases that experts went through during their teaching process: Cold Start phase, Boundary phase, Challenge phase, and Testing phase . These phases are non-overlapping stages of the MT process. Experts begin in the Cold Start phase, after which they iteratively cycle through the other three phases. The phases can be characterized as follows.

In the Cold Start phase (CSP), experts begin by using a Keyword Search to find and label positive examples of the classifier (S1-S5). Then, they use a Random sampling strategy to find and label diverse negative examples of the classifier (S1, S3, S6). During

[9] `webhose.io`

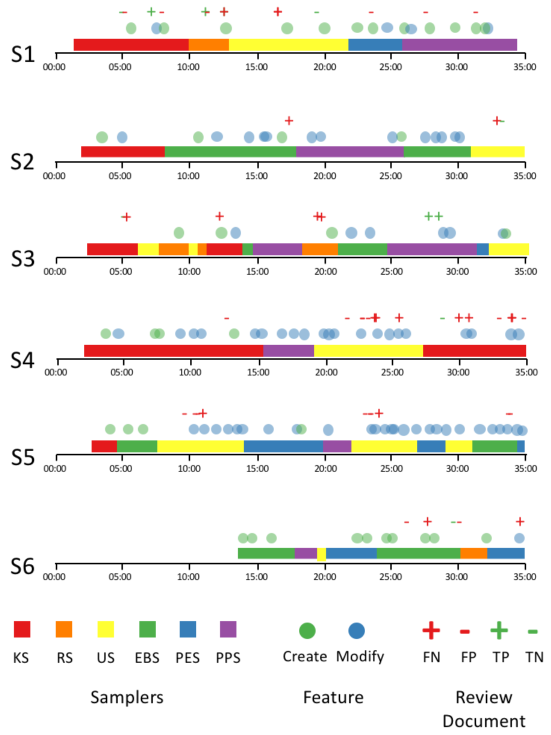


Fig. 3: The teaching actions experts went through over time, including samplers used (Table 1), creation or modification of features, and documents reviewed (false positives (FP), false negatives (FN), true positives (TP), and true negatives (TN)).

this phase they create features to roughly capture the positive class (S-all); they are not particularly concerned with training errors at this point. After they initialize a model with a minimum amount of positive and negative labels plus some basic features, experts transition into Boundary or Challenge phase.

The Boundary phase (BP) is characterized by a lack of clear boundary separating positive and negative labels in the training set. Experts use Uncertainty sampling to find and label documents that the model does not predict confidently (S-all), i.e., documents with a prediction score around 0.5. The main goal during this phase is refining the boundary or improving the separation between the positive and negative labels. Based on the labels and features they create, experts transition from this phase to Challenge.

The Challenge phase (CP) is characterized by a training set where there is a clear boundary separating positive and negative labels. Experts use a variety of sampling strategies to (1) label diverse data and (2) attempt to challenge or 'break' the model. For example, some experts indicated they would try to frequently change sampling strategies to increase the diversity of the labeled data (S1-S3, S5), but the main task during this phase was to find documents that the model does not predict correctly (S-all). This involved particular focus on the Possible Errors (S1, S3, S5-S6) and Predicted Positives

(S-all) sampling strategies (Table 1). Based on the labels and features they create, experts can transition to from this phase to Testing or back to Boundary.

The Testing phase (TP) is characterized by the explicit activity of evaluating the quality of the model on unseen data. This phase can occur periodically throughout the teaching process, most commonly transitioning from CP. Experts test their model by looking at the documents fetched using the Predicted Positives sampling strategy, and counting how many predictions out of a batch are correct; no new labels or features are created in this phase. For example, if nine out of ten documents are correctly predicted as positive, the teacher feels more confident of the model's behavior on unseen data. In our study, only one subject (S2) performed this task. However, all experts indicated it is an important part of the MT process but neglected this phase due to the session's 35-minute time constraint.

Machine Teaching Expert Patterns. In addition to the four MT phases, we were also able to observe a number of expert patterns throughout the different teaching phases. Experts went through a state of flow where they alternated between labeling and featuring; sometimes adding features based on relevant keywords they saw in documents and sometimes adding other features whenever they came to mind (S-all). In this context, teachers added and refined features to try to generalize them and make them representative of a relevant concept. Occasionally, some experts would also look to feature suggestions (a list of potential keywords provided by MATE that could fix training errors) for inspiration on relevant keywords to include (S5).

Experts tended to review errors whenever the model tracker visualization on the left side of the MATE interface appeared to change significantly (S-all), e.g., after a transition from CSP to BP. No one immediately addressed every training error that appeared. Each expert had a different threshold of the acceptable amount of errors to have, generally $< 20\%$ of labels.

To fix False Positive errors, experts would create features that explain a negative label (e.g., if the document was a White House press release and thus not about movies and television, they might create a feature to describe the concept of politics) (S-all). Conversely, to fix False Negative errors, they would create features that explain a positive label (e.g., a document talking about movie studios) (S-all).

After creating a new feature, some experts would 'test' the feature using the Explanation-Based sampling strategy (S2-S3, S5-S6). MATE showed in context the feature's keywords in the document's view and let experts assess whether the feature was working as desired. This practice happened during all phases (except Testing).

Sometimes experts faced a document that was too rare, or difficult to explain with the provided teaching language of keyword lists. Sometimes experts were just not sure how to label a document. In either of these cases they used the "decide later" option so that the document would not be used to train the model, or until they were certain how to label it.

6 Study 2: Guiding Novice Teachers

While the MT loop is something that can be straightforward for someone to learn, its effective use requires insight in terms of when and how to sample, label, and fix errors.

At a different level, problem solving skills such as breaking a larger problem or concept into smaller ones are also valuable for MT. As these are not innate skills, it follows that becoming a proficient machine teacher can require practice and experience.

In Study 1, we manually extracted expert teacher patterns within the MT loop based on our observations of expert behavior. In our second study, we synthesized guiding notifications from these expert practices. In addition to observing how successfully end-users apply the MT process, we want to explore whether guiding novice machine teachers with these expert practices could help them to become better machine teachers. As a tool, MATE sits in a class of its own in terms of user prerequisites and functionality, hence we do not aim to compare it with other ML-building tools. Rather, our goal is to observe how successfully end-users apply the MT process under different guiding conditions, and see how similarly they perform compared to expert teachers. We design this guidance to help teachers by *unblocking* them in their teaching progress. We did not design or consider this type of guidance to serve as a tutoring system and hence we do not focus on retention of expert practices.

Translating Expert Practices into Notifications. Our goal of adding a notification system to MATE is to provide guidance throughout the MT process to novice teachers, including sampling, featuring, and labeling guidance. When synthesizing our observations from experts' teaching strategies into actionable guidance, we sought to characterize prescriptive actions as well as conditions for when such actions are useful. Table 2 summarizes the guiding notifications we produced from our observations.

The timing of the guidance notifications is an important factor in their effectiveness. Of the four types of interruption strategies [28], we follow a naïve mediated approach that chooses moments between tasks (a strategy supported by work like [9]), or situations of apparent inactivity. There are several such moments during a teaching session:

1. after submitting a label,
2. after closing the review of a labeled document,
3. (while reading a new document) after 30, or 60 seconds of inactivity[10],
4. after getting a document with the same sampler without labeling 10 times in a row,
5. after creating/editing a feature, and
6. after creating/editing features for 5 consecutive minutes.

The basic mechanics of our desired notifications work as follows: whenever an interruptible moment happens, the system chooses the proper notification to show based on the system's current state. If there is ambiguity as to which notification to show, the system picks one at random. The chance of a notification significantly decreases if the notification has been seen (and acknowledged) within the past 10 minutes.

In this study, we are inspired by [20]'s wizard-of-oz methodology and adopt a similar strategy to both observe the significant interactions (such as hovering, clicking on a particular UI element/area) and to emulate the system's decision about when to present suggestions. In our implementation of the guidance functionality, we dispatch notifications through a 'Wizard of Oz' who follows the above heuristics in a consistent way. The wizard is able to work by being able to see a live-feed of the MATE system's

[10] a proxy for reading: hovering over, or scrolling a document.

Name	Guidance	Heuristic
CSP: Positive	Use KS to find and label 10 positive examples.	At the start of the MT process.
CSP: Negative	Use RS to find and label 10 negative examples.	After 10 positive examples have been labeled.
BP	Use US to find and label 10 examples.	Predictions are not well-separated.
CP	Change samplers often and find diverse documents to label.	Predictions are well-separated.
TP	Use PPS to see how accurately the model predicts 10 unseen documents.	After every 50 labels; predictions are well-separated.
Tip: Create Feature	Create a feature that explains a positive label.	Reviewing a false negative.
Tip: Test Feature	Use EBS to see if a feature captures documents in the context the teacher expects.	After creating or modifying a feature, during BP or CP. Do not show during CSP.
Tip: Possible Errors	Use PES to find documents the model is likely predicting incorrectly.	After CSP; predictions are well-separated.
Tip: Review Errors	Review training errors.	Predictions are not well-separated; the number of errors is high.
Tip: Feature Suggestion	Check if the system suggests any sensible feature.	The tool suggests features; number of errors is high; predictions are not well-separated.
Tip: Decide Later	Use "decide later" if the teacher is not sure how to label; or they cannot ideate a feature to explain a label.	Reviewing a document for $\geq$ 30 seconds, or revisiting the same document $\geq$ 3 times.
Tip: Negative Feature	Create a feature to describe why a label is negative.	Reviewing a false positive.

Table 2: The guidance provided to participants along with heuristics for when notifications can be dispatched. We say predictions are *well-separated* when $\leq$ 10% of prediction scores are between 0.3-0.7. We say the number of errors is *high* when $\geq$ 20% of documents are predicted incorrectly in the model. The full text of each notification can be found in supplemental materials.

screen and the teacher's actions within, including interactions such as mouse movements and clicks. We chose this implementation for a number of reasons: first, it let us add functionality into a complex code base quickly and with ease; second, it allowed us to fine tune policies quickly through pilot studies; and last, it let us monitor our heuristics while opportunistically discovering important state signals to consider in the future.

While it would be feasible to implement an automated version of this guidance system, it is not our focus and remains a topic of future work.

Participants. In this study, we recruited 24 [6] MT novices (9 female), which we screened as individuals who self reported having little or no familiarity with either ML and MT. The average age was 31. We also recruited an additional 3 MT experts (2 female). The MT experts served as a performance benchmark in our data analysis.

Materials. We randomly assigned novice participants to either the Non-Guided (NG) or the Guided (G) condition. Experts (E) performed the task under the same conditions as Non-Guided participants. In both conditions participants built models using the same 90K article dataset from Study 1. In the Non-Guided condition, participants used MATE as described in Section 3. In the Guided condition, participants used a version of MATE that was modified with guidance notifications, as described above, that appeared at the bottom right of the interface (Figure 2-5).

For both conditions, we prepared three short training videos (totaling approximately 15 minutes) to introduce fundamentals of ML, MT, and the MATE system, respectively. We also gave participants reading materials in the form of an MT diagram (Figure 1) and a summary description of sampling strategies in the MATE system (Table 1). The purpose of these training materials was to provide a minimal common ground language and knowledge for all novices participating in our experiment.

Earlier work points out that despite having basic knowledge, novice ML practitioners can struggle with notions such as features, or negative concepts [1, 37]. Because of this, while these off-line tutorials can help the progress from novice to expert, we still hypothesize that the addition of in-situ guidance can accelerate even further the adoption of MT expert practices.

Procedure. We divided our study into two stages. After passing an initial screening for selecting ML and MT novices, participants took part in the first stage of our study, in which we sent them our training materials. To verify that participants studied the materials, we asked them to complete an on-line quiz that tested the fundamental concepts they needed to know to use the MATE system and teach a binary classifier with it. Only participants with a score $\geq 80\%$ were allowed to proceed to stage two of our study. Participants could re-take the quiz as many times as necessary. We compensated participants that completed this stage with a $15 Amazon gift card.

We scheduled participants that passed the quiz to complete stage two of our study: building a binary classifier for articles about travel and tourism. Sessions took place within one week of passing the quiz. During this session we gave participants 5 minutes to ask questions regarding the learning materials they studied beforehand. Afterward, we gave a 2-minute interactive tour of the MATE system and allowed them to use it for another 5 minutes in a sports classifier demonstration. When ready, we gave participants 45 minutes to build the classifier.

We did not want to test participants' memory on the details of the system or their quality as labelers. Instead, we sought to observe their teaching process. Hence, we encouraged participants to ask questions about MATE as needed (e.g., where to click to sample data, or how to label a document as "decide later"). Further, they could ask, if in doubt, if a document belonged to the travel and tourism class. We demonstrated the notifications to participants in the Guided condition, and told them that they were designed to help them build a better model. We instructed them to follow their guidance whenever possible, and that not dismissing them will give the system the impression that they did not see the notification, leading to the system 'insisting' on showing them again.

Upon completion of this second stage, we compensated participants with a $35 Amazon gift card. The novice participant in each condition who achieved the highest F1-score (described next) was awarded an additional $50 Amazon gift card.

Measuring a Model's Performance. We measured the quality of the model each participant created by computing the F1-Score over the same sampling set used during the teaching session. As participants only labeled less than 0.2% of the set, the amount of overlap is negligible, thus we consider these measures good estimates of each model's performance. While having access to these measurements will allow us to have a sense of how novices' output compares to the experts', we do not expect the overall quality of the models to be high after 45 minutes of teaching.

7 Study 2: Results

Non-Guided novices generated an average of 64 labels ($SD = 23$) and 70 keywords ($SD = 34$) across 12 features ($SD = 6$). Guided novices generated an average of 72 labels ($SD = 24$) and 61 keywords ($SD = 25$) across 11 features ($SD = 3$). Experts who participated as a performance baseline generated an average of 91 labels ($SD = 37$) and 131 keywords ($SD = 114$) across 11 features ($SD = 5$). One expert (E2) differed from others and focused a large portion of time creating keywords by searching the web for a list of 198 countries. The distributions of these results can be seen in Figure 4.

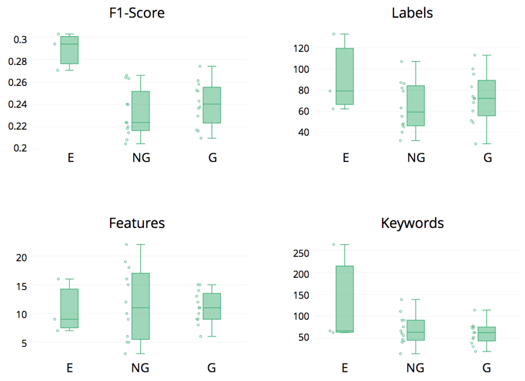


Fig. 4: Summary of models produced by participant groups Experts (E), Non-Guided (NG), and Guided (G) in Study 2, including F1-Scores in the top-left, number of labels produced in the top-right, number of features or explanations produced in the bottom-left, and number of keywords across features in the bottom-right.

Model Quality. We computed the F1-Score from the model that each participant built, as described in Section 6. After confirming that the sample variances were the same using Levene's test ($W = 0.0425$, $p = 0.8385$), we conducted an independent samples t-test to determine if Guided novices performed better than Non-Guided. We found $t = -0.9812$, $p = 0.3373$ and were unable to reject the null hypothesis.

While we do not see a significant difference between the groups' model quality, we see a trend by examining their distributions (Figure 4). We observed that experts built better models, with a median F1-Score of 0.2938, followed by Guided novices with a median F1-Score of 0.2403, and lastly by Non-Guided novices with a median F1-Score of 0.2238.

Qualitative Observations.

Besides looking at the effects of guidance on the model's quality, we also wanted to see how guidance affected participants' overall user experience. After they completed the task, participants rated their experience across 5 dimensions (perceived success level, frustration, mental demand, pace, and effort) on a 7-point Likert scale. For success level, 1 = worst and 7 = best, while for the other four dimensions 1 = best and 7 = worst.

To see if there was a difference between Non-Guided and Guided novices, we conducted a Mann-Whitney U test for each of the 5 dimensions. We found the following results: success ($U = 71$, $p = 0.9762$), frustration ($U = 74.5$, $p = 0.9059$), demand ($U = 101.5$, $p = 0.0866$), pace ($U = 69$, $p = 0.8818$), and effort ($U = 81.5$, $p = 0.5824$), none of which indicate statistically significant differences.

Despite these results, we do see qualitative patterns among the distributions of ratings (Figure 5). All conditions perceived a median success level of 4.

For three dimensions, Guided novices perceived a lower median {mental demand, rushed pace, effort level}. Guided and Non-Guided novices rated a median rush of pace of 4, compared to Expert median rating of 5. Guided novices rated a median mental demand of 3.5 compared to Non-Guided novice and Expert median ratings of 5. Similarly, Guided novices rated a median effort level of 4.5 compared to Non-Guided novice and Expert median ratings of 5. Guided novices rated a median frustration level of 3, Non-Guided novices rated a median frustration level of 2.5, and Experts rated a median frustration level of 4. While Non-Guided novices rated a lower frustration level (median 2.5) than Guided novices (median 3), both groups rated this dimension fairly low, possibly due to the general perception that MT is an enjoyable process. For example, one participant indicated that it "feels like a game" (P20), while another indicated he "would play this all day if it were a game" (P30). When creating features to fix training errors, another participant indicated it's like a "word association game" (P24).

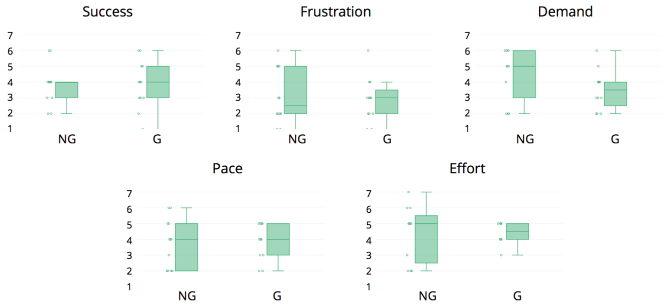


Fig. 5: Likert ratings (1-7) for Non-Guided (NG) and Guided (G) conditions in Study 2.

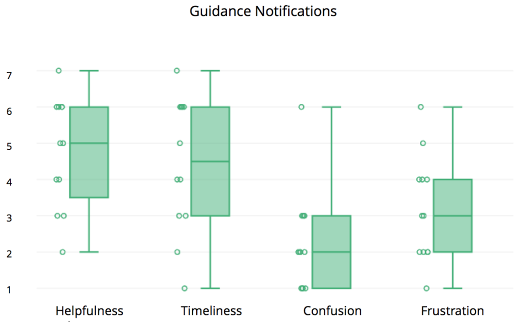


Fig. 6: Summary of Likert ratings (1-7) about the guidance notifications in Study 2, for helpfulness, timeliness, confusion, and frustration. Higher values for helpfulness and timeliness are good, while lower values for confusion and frustration are desired.

Experts tended to feel higher levels of frustration, mental demand, rushed pace, and effort level, possibly due to a common perception that with more time, they could have had a much higher quality model.

In addition to these 5 dimensions, participants in the Guided condition also rated the helpfulness, timeliness, confusion, and frustration specifically with respect to the guidance notifications (Figure 6). Most participants expressed that the notifications were helpful (median rating of 5) and they were largely receptive to the help. For example, when presented a notification to move on to the next phase of training, one participant indicated "sure, I'll switch samplers; thank you" (P15). Some participants indicated dissatisfaction with the timing of the notifications. One participant indicated that notifications "don't give me enough time to focus on one aspect before telling me to do something else" (P01). Despite feedback about the timing, participants overall found the notifications to be timely (median rating of 5). Similarly, participants experienced low levels of confusion (median rating of 2) and frustration (median rating of 3).

8 Discussion and Future Work

Study Limitations. The low number of expert participants in both studies was a function and challenge of the relatively small population of machine teaching expert participants. From a total of around 10 experts, 6 and 3 for Studies 1 and 2, respectively, were what was possible without having overlapping subject pools. Nonetheless, the presence of expert participants in Study 2 served only as an anecdotal baseline to put novice results in context. Furthermore, because of the relatively short duration in Study 1, experts did not fully evaluate their models. However, MATE's model tracker provided always up-to-date information regarding model performance on the training set.

Optimal Teaching Process. We started our research looking for a gold standard teaching playbook, that could lead novices on the path to teaching good models every time. More than a single playbook, we found expert patterns which we can take as current best practices. What we learned by observing experts and novices alike, is that even though the MT loop provides a sensible path to good teaching, there are still several ways to combine teaching patterns to build a good model.

For example, the Cold Start phase pattern we encouraged consisted of adding positive labels, then negative labels, and later a feature. However, these activities could have occurred in any order. In fact, we observed that many participants (e.g., P15) tended to review false positive errors, and hence created features describing "negative" concepts. Because of this, these teachers often neglected to create features describing positive examples. To address this, an alternative starting point during CSP could be to prompt users to first create a feature describing the positive concept of the classifier.

The Importance of Features. In addition, one of the fundamental observations we took from our studies, is that a model is only as good as the knowledge a teacher puts in it, regardless of the process or order of actions. In particular, the quality or generalizability of the features that teachers ideate affected the quality of their model, and thus the teaching phase one could be in. While some participants showed some understanding of the concept of overfitting (e.g., while training P08 indicated "it's probably super overfit") and the importance of precise features (e.g., P12 indicated "this is a bad word to use, I think [it's] too generic"), many participants still struggled to come up with good features. A guidance notification reminding users of the important balance between precise yet generalizable features could have been helpful.

Similarly, many novices frequently asked us about how to create negative features, or to see how positively or negative correlated an existing feature was (e.g., P04). A feature in MATE becomes negatively or positively correlated to a label depending on that feature 'firing up' on positive or negative examples. This suggests a common mental model that novices think about keywords as either positively or negatively associated with a concept. We also observed that the number of features teachers produced is negatively correlated with F1-Score ($r = -0.4700, p = 0.0205$). This is likely caused by models overfitting the training data. These observations underscore the importance of tooling during the MT loop supporting the creation and evaluation of good features.

Model Quality. Our studies showed that the performance of models built by Guided and Non-Guided novices was not significantly different. While considerable in terms of effort for a user study, 45 minutes was not sufficient to teach a model of high quality, and perhaps see differences between the two guiding conditions. Even though our quantitative results do not show statistically significant differences, our study clearly highlights that with minimal on-boarding, novice machine teachers are not too far behind from experienced teachers. This reinforces the benefits of MT requiring a softer set of skills than processes that require ML or Data Science expertise. This underlines further the potential of MT as a way to democratize ML. In future research, we want to observe teaching sessions longer than 45 minutes where the nuanced effects of different teaching patterns might emerge.

Activities' Cadence. Our results suggest that the novices' qualitative experience benefited from guidance. We also saw how the absence of guidance affected some. For example, it was common to see Non-Guided novices spend time thinking about what to do after the model perfectly fit the training data. We also observed moments of significant focus where a Non-Guided teacher would do only one thing, such as refining features, or use a single sampler for a long time. In the absence of guidance other participants found some tasks just fun to do, and saw feature creation/refinement as a “word association game” (P24).

This is a behavior hinted at during our pilot, and our notification triggers accounted for some of these “sticky” behaviors by suggesting teachers change what they are doing. Nonetheless, we only had an opportunity to observe these behaviors in full during the study, which reinforced our view that characterizing the proper cadence of labeling, sampling and featuring activities is important future work.

Automating Guidance. Participants never realized that the system’s notifications were driven by a Wizard-of-Oz. Having a consistent set of rules to follow allowed investigators to simulate this behavior, in a believable and affordable way. Part of our future efforts is to refine and automate the notification system so that it can run without a wizard’s oversight. While we focused on proactive guidance in this study, future work might also compare the effectiveness of proactive guidance to reactive [36] or on-demand guidance in the context of the MT process.

9 Conclusions

In this paper we build onto the promise for Machine Teaching as a way to widen access to the use of ML by end-users and domain experts to solve problems. We look at how users without experience in the MT process apply it to create a binary classifier. Further, we look into lowering the barrier of entry to MT, by identifying and synthesizing MT expert patterns, and presenting them to novices at appropriate times during different teaching sessions. While our results on the effects of this guidance are statistically inconclusive, we see a trend that Guided novices’ produce better models as well as perceive a better overall teaching experience (less effort and less mental demand) than Non-Guided novices. Chief among the results from our studies is the observation that people without special ML knowledge can successfully engage in a teaching loop to create ML classification models, and that those models are not far behind the ones created by MT experts. MT is a process that arguably requires a person’s full attention, including making richer decisions than simply labeling. We believe that this apparent cost is actually low, as we observed that the proper experience and interaction loop can make it an engaging activity end-users are not only willing, but also happy, to participate in. In particular we are encouraged by feedback from users who expressed a willingness to participate in a “fun” problem-solving process. We believe there are many areas to explore in the context of MT, and we see a green field of opportunity at the intersection of HCI and ML to advance the discipline and adoption of this form of end-user ML.

References

1. Amershi, S., Cakmak, M., Knox, W.B., Kulesza, T.: Power to the people: The role of humans in interactive machine learning. AI Magazine **35**(4), 105–120 (2014)

2. Amershi, S., Chickering, M., Drucker, S.M., Lee, B., Simard, P., Suh, J.: Modeltracker: Redesigning performance analysis tools for machine learning. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. pp. 337–346. CHI '15, ACM, New York, NY, USA (2015)
3. Amershi, S., Fogarty, J., Weld, D.: Regroup: Interactive machine learning for on-demand group creation in social networks. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 21–30. ACM (2012)
4. Bernard, J., Zeppelzauer, M., Lehmann, M., Müller, M., Sedlmair, M.: Towards user-centered active learning algorithms. In: Computer Graphics Forum. vol. 37, pp. 121–132. Wiley Online Library (2018)
5. Brooks, M., Amershi, S., Lee, B., Drucker, S.M., Kapoor, A., Simard, P.: Featureinsight: Visual support for error-driven feature ideation in text classification. In: Visual Analytics Science and Technology (VAST), 2015 IEEE Conference on. pp. 105–112. IEEE (2015)
6. Caine, K.: Local standards for sample size at chi. In: Proceedings of the 2016 CHI conference on human factors in computing systems. pp. 981–992. ACM (2016)
7. Chen, N.C., Suh, J., Verwey, J., Ramos, G., Drucker, S., Simard, P.: Anchorviz: Facilitating classifier error discovery through interactive semantic data exploration. In: 23rd International Conference on Intelligent User Interfaces. pp. 269–280. ACM (2018)
8. Cheng, J., Bernstein, M.S.: Flock: Hybrid crowd-machine learning classifiers. In: Proceedings of the 18th ACM conference on computer supported cooperative work & social computing. pp. 600–611. ACM (2015)
9. Cheng, J., Teevan, J., Iqbal, S.T., Bernstein, M.S.: Break it down: A comparison of macro- and microtasks. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. pp. 4061–4064. CHI '15, ACM, New York, NY, USA (2015)
10. Das, S., Cashman, D., Chang, R., Endert, A.: Beames: Interactive multi-model steering, selection, and inspection for regression tasks. Symposium on Visualization in Data Science (2018), to appear
11. Doshi-Velez, F., Kortz, M., Budish, R., Bavitz, C., Gershman, S., O'Brien, D., Schieber, S., Waldo, J., Weinberger, D., Wood, A.: Accountability of AI under the law: The role of explanation. CoRR **abs/1711.01134** (2017), `http://arxiv.org/abs/1711.01134`
12. Dudley, J.J., Kristensson, P.O.: A review of user interface design for interactive machine learning. ACM Transactions on Interactive Intelligent Systems (TiiS) **8**(2), 8 (2018)
13. Fails, J.A., Olsen, Jr., D.R.: Interactive machine learning. In: Proceedings of the 8th International Conference on Intelligent User Interfaces. pp. 39–45. IUI '03, ACM, New York, NY, USA (2003)
14. Fogarty, J., Tan, D., Kapoor, A., Winder, S.: Cueflik: interactive concept learning in image search. In: Proceedings of the sigchi conference on human factors in computing systems. pp. 29–38. ACM (2008)
15. Fraser, C.A., Dontcheva, M., Winnemöller, H., Ehrlich, S., Klemmer, S.: Discoveryspace: Suggesting actions in complex software. In: Proceedings of the 2016 ACM Conference on Designing Interactive Systems. pp. 1221–1232. DIS '16, ACM, New York, NY, USA (2016)
16. Goodman, B., Flaxman, S.: European Union regulations on algorithmic decision-making and a "right to explanation". AI Magazine **38**(3), 50–57 (2017)
17. Hartmann, B., Abdulla, L., Mittal, M., Klemmer, S.R.: Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In: Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 145–154. ACM (2007)
18. Horvitz, E.: Principles of mixed-initiative user interfaces. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 159–166. CHI '99, ACM, New York, NY, USA (1999)

19. Horvitz, E., Apacible, J.: Learning and reasoning about interruption. In: Proceedings of the 5th International Conference on Multimodal Interfaces. pp. 20–27. ICMI '03, ACM, New York, NY, USA (2003)
20. Hudson, S., Fogarty, J., Atkeson, C., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J., Yang, J.: Predicting human interruptibility with sensors: A wizard of oz feasibility study. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 257–264. CHI '03, ACM, New York, NY, USA (2003)
21. Hutter, F., Lücke, J., Schmidt-Thieme, L.: Beyond manual tuning of hyperparameters. KI-Künstliche Intelligenz **29**(4), 329–337 (2015)
22. Iqbal, S.T., Bailey, B.P.: Investigating the effectiveness of mental workload as a predictor of opportune moments for interruption. In: CHI '05 Extended Abstracts on Human Factors in Computing Systems. pp. 1489–1492. CHI EA '05, ACM, New York, NY, USA (2005)
23. Iqbal, S.T., Bailey, B.P.: Effects of intelligent notification management on users and their tasks. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 93–102. CHI '08, ACM, New York, NY, USA (2008)
24. Jandot, C., Simard, P., Chickering, M., Grangier, D., Suh, J.: Interactive semantic featuring for text classification. arXiv preprint arXiv:1606.07545 (2016)
25. Kanter, J.M., Veeramachaneni, K.: Deep feature synthesis: Towards automating data science endeavors. In: Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on. pp. 1–10. IEEE (2015)
26. Kulesza, T., Amershi, S., Caruana, R., Fisher, D., Charles, D.: Structured labeling for facilitating concept evolution in machine learning. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 3075–3084. ACM (2014)
27. Kulesza, T., Burnett, M., Wong, W.K., Stumpf, S.: Principles of explanatory debugging to personalize interactive machine learning. In: Proceedings of the 20th international conference on intelligent user interfaces. pp. 126–137. ACM (2015)
28. McFarlane, D.: Comparison of four primary methods for coordinating the interruption of people in human-computer interaction. Hum.-Comput. Interact. **17**(1), 63–139 (Mar 2002)
29. Meek, C.: A characterization of prediction errors. CoRR **abs/1611.05955** (2016), `http://arxiv.org/abs/1611.05955`
30. Olson, R.S., Bartley, N., Urbanowicz, R.J., Moore, J.H.: Evaluation of a tree-based pipeline optimization tool for automating data science. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016. pp. 485–492. ACM (2016)
31. Olson, R.S., Urbanowicz, R.J., Andrews, P.C., Lavender, N.A., Moore, J.H., et al.: Automating biomedical data science through tree-based pipeline optimization. In: European Conference on the Applications of Evolutionary Computation. pp. 123–137. Springer (2016)
32. Patel, K., Bancroft, N., Drucker, S.M., Fogarty, J., Ko, A.J., Landay, J.: Gestalt: integrated support for implementation and analysis in machine learning. In: Proceedings of the 23nd annual ACM symposium on User interface software and technology. pp. 37–46. ACM (2010)
33. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. Journal of machine learning research **12**(Oct), 2825–2830 (2011)
34. Simard, P.Y., Amershi, S., Chickering, D.M., Pelton, A.E., Ghorashi, S., Meek, C., Ramos, G., Suh, J., Verwey, J., Wang, M., et al.: Machine teaching: A new paradigm for building machine learning systems. arXiv preprint arXiv:1707.06742 (2017)
35. Weld, D.S., Bansal, G.: Intelligible artificial intelligence. CoRR **abs/1803.04263** (2018), `http://arxiv.org/abs/1803.04263`
36. Xiao, J., Catrambone, R., Stasko, J.: Be quiet? evaluating proactive and reactive user interface assistants. In: Proceedings of INTERACT. vol. 3, pp. 383–390 (2003)
37. Yang, Q., Suh, J., Chen, N.C., Ramos, G.: Grounding interactive machine learning tool design in how non-experts actually build models. ACM (June 2018)

38. Zhu, X.: Machine teaching: an inverse problem to machine learning and an approach toward optimal education. In: The Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)