Demystifying a Dark Art: Understanding Real-World Machine Learning Model Development

Angela Lee*
University of Illinois at Urbana-Champaign
alee107@illinois.edu

Doris Xin*, Doris Lee*, Aditya Parameswaran
University of California, Berkeley
{dorx,dorislee,adityagp}@berkeley.edu

ABSTRACT

It is well-known that the process of developing machine learning (ML) workflows is a dark-art; even experts struggle to find an optimal workflow leading to a high accuracy model. Users currently rely on empirical trial-and-error to obtain their own set of battle-tested guidelines to inform their modeling decisions. In this study, we aim to demystify this dark art by understanding how people iterate on ML workflows in practice. We analyze over 475k user-generated workflows on OpenML, an open-source platform for tracking and sharing ML workflows. We find that users often adopt a manual, automated, or mixed approach when iterating on their workflows. We observe that manual approaches result in fewer wasted iterations compared to automated approaches. Yet, automated approaches often involve more preprocessing and hyperparameter options explored, resulting in higher performance overall—suggesting potential benefits for a human-in-the-loop ML system that appropriately recommends a clever combination of the two strategies.

ACM Reference Format:
Angela Lee and Doris Xin*, Doris Lee*, Aditya Parameswaran. 2020. Demystifying a Dark Art: Understanding Real-World Machine Learning Model Development. In *Workshop on Human-In-the-Loop Data Analytics (HILDA'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3398730.3399153

1 INTRODUCTION

Machine learning (ML) has become a critical component of almost every domain, with users of all different levels of expertise relying on ML models to accomplish specific tasks. Developing an ML workflow can be tedious and time-consuming. Anecdotally, it may take dozens of iterations for a novice to converge on a satisfactory combination of ML model type, hyperparameters, and data preprocessing. Meanwhile, an expert might need only a small number of modifications to their original workflow to achieve comparable performance on the same task. In such cases, novices can benefit greatly from learning strategies employed by experts to accelerate the process of developing effective ML workflows. With a large-scale database of user-generated workflows and evaluation scores, it becomes possible to extract aggregate workflow iteration patterns to enable this transfer of knowledge.

In this work, we present useful insights about ML workflow development behavior of users on the OpenML[18] platform. OpenML is an open-source, hosted platform for users to upload datasets and run ML workflows on these datasets by calling an API. A relatively diverse mix of user skill levels is present on OpenML, from students just getting started with ML to experienced data scientists and ML researchers. OpenML publishes a database of the user-uploaded datasets as well as the workflow specifications submitted by users and their corresponding executions. By performing targeted analyses on the most common ML models and preprocessing operators as well as their associated performance, we shed light on common ML workflow design patterns and their general effectiveness. We study trends in iterative ML workflow changes across 295 users, 475,297 runs, and 793 tasks on the OpenML platform; and draw quantitative conclusions from this crowd-sourced dataset, leading to actionable insights that help inform the development of future human-in-the-loop ML systems targeting novices and experts alike.

Our main contributions can be summarized as follows:

- We characterize the frequency and performance of popular ML models and preprocessing operators used by OpenML users. We highlight the impact of the operator combinations and discuss their implications on general user awareness (or lack thereof) of particular ML concepts (Section 4).
- We analyze sequences of changes to workflows to extract different styles of ML workflow iteration and shed light on the most common types of changes for each iteration style, the amount of exploration typically performed, and performance gain users are generally able to achieve (Section 5).
- We conduct case studies on exemplary instances to understand effective iteration practices. (Section 6).

2 RELATED WORK

There is an incredible wealth of knowledge that can inform the design of automated and semi-automated ML (autoML) systems by understanding how people develop machine learning models. Studies through empirical code analysis and qualitative studies offer different lenses into studying human-centered practices in developing ML workflows.

Psallidas et al.[14] analyzed publicly-available computational notebooks and enterprise data science code and pipelines to illustrate growing trends and usage behavior of data science tools. Other studies have employed qualitative, semi-structured interviews to study how different groups of users engage with ML development, including how software engineers [2] and non-experts [21] develop ML-based applications, and how ML practitioners iterate on their data in ML development [8].

*Equal Contribution.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
HILDA'20, June 14–19, 2020, Portland, OR, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8022-5/20/06…$15.00
https://doi.org/10.1145/3398730.3399153

In this paper, we analyze practices and behaviors for users iterating on ML workflows, based on data collected from OpenML [18]—an online platform for organizing and sharing ML experiments to encourage open science and collaboration. Data from OpenML has been used extensively for meta-learning [17] to recommend optimal workflow configurations, such as preprocessing [4, 13], hyperparameters [16], and models [3, 15] for a given dataset and task. Benchmark datasets from OpenML, as well as other similar dataset repositories [1, 11], have also been used for evaluating AutoML algorithms [5–7]. However, these papers focus solely on using the dataset, model and result from each workflow and do not study the iterative behavior of users.

Our paper differs from existing work in that we analyze the trace of *user-generated* ML workflow iterations leading to insights such as which stages (preprocessing, model selection, hyperparameter tuning) users focus most of their iterations on, their impact on the effectiveness of the workflow, and which specific combinations of model and preprocessing operators are the most widely used. Our approach offers a more reliable view of iterative ML development than the previous work using experiments reported in applied ML papers to approximate iterations [20]. Moreover, our study offers a complementary perspective to existing interview studies by providing empirical (based on code), population-level statistics and insights on how people iterate on machine learning workflows. These insights are valuable for helping AutoML and mixed-initiative-ML system builders understand the trends in the ML development practices of target users.

3 DATA & METHODOLOGY

We define important terms used in our study, describe our dataset, and define the metrics for quantifying user effectiveness.

3.1 Terminology

In ML development, a user creates a *workflow* to accomplish a specific *task* and iterates on the workflow over a *sequence* of *runs*.

- **Task**: A task consists of a dataset along with a machine learning objective, such as clustering or supervised classification.
- **Workflow**: A workflow is a directed acyclic graph of data *preprocessing* and ML *model* operators with their associated hyperparameters.
- **Run**: A run is a workflow applied to a particular task. Each run is associated with performance measures, such as classification accuracy or Area Under the ROC Curve (AUC).
- **Sequence**: A sequence consists of the time-ordered set of runs from a single user for a particular task.

3.2 The Dataset

Our dataset is derived from a snapshot of the OpenML database from December 4, 2019. We focus on a specific subset of the runs on OpenML that a) were uploaded by users who are not OpenML developers[1] or bots[2], to focus on realistic human user behavior, b) use the Scikit-Learn package [12], to increase the fidelity of our data extraction by focusing on a single popular ML package used in 79% of the non-developer/non-bot runs, and c) have an associated AUC score associated with them. This subset contains 475,297 runs (4.8% of the total number of runs)

Limitations: While the OpenML dataset provides a valuable probe into how users iterate on ML models, the typical OpenML user may not be representative of general ML practitioners. The OpenML traces provide a limited view of the user's motivation and thought process behind their iteration choices. Our following analysis is intended to present a formative picture of how people iterate in ML development. Future studies with larger sample sizes and more representative samples are required to generalize these findings.

3.3 Workflow Effectiveness Metrics

Due to the variability in the range of evaluation metric values across tasks, raw AUC values of runs are not directly comparable across tasks. Instead, we account for the difficulty of each task by measuring *relative AUC*. For a run r, let a_r be its raw AUC on the corresponding task t. The relative AUC of a run r, is defined as $p_r = a_r - \mu_t$, where μ_t is the average AUC of all of the runs for task t. We measure the performance of a sequence S by the *relative maximum sequence AUC*, $p_S = \max_{r \in S} a_r - \frac{1}{|\mathcal{T}_t|} \sum_{Q \in \mathcal{T}_t} \max_{q \in Q} a_q$, where $\mathcal{T}_t$ is the set of all sequences for task t. In other words, p_S is the difference between the maximum AUC in the sequence S and the mean of the maximum AUCs of all sequences for task t. We use maximum AUC per sequence to capture the fact that the best-performing workflow out of all attempts is the final one that the user adopts.

4 RUN-LEVEL INSIGHTS

To better understand how users develop ML workflows, we first sought to understand: *What are the most prevalent ML operators in practice? In what combinations are these operators used together and when do they lead to better performance?*

Figure 1 shows the most common model and preprocessing combinations used by more than four users on OpenML. Ensemble models that combine one or more base estimators were excluded from the plot, to avoid duplication with other base estimators. The color indicates the performance of the combination, measured by p_r, averaged over all runs that include the combination, while the size of each circle is scaled by the number of users (Figure 1 top) and runs (Figure 1 bottom) that contain the specific combinations. Both the rows and columns are sorted based on their performance across all of the specified operators. In other words, the best-performing combinations are located at the top-right corner of the chart, whereas the worst-performing combinations are on the bottom-left.

Histograms showing the marginal distribution of frequency and performance for each operator are displayed on the sides. For instance, the uppermost user frequency histogram shows the average user count for each preprocessing operator, averaged across the models that were used in combination with it, normalized by the total number of users.

We highlight various insights from Figure 1, with the enumerated points corresponding to the enumerated boxed regions in the figure.

(1) Performance of Specific Combinations: Some combinations consistently yield high relative AUC when compared to other combinations used for the same ML task. For instance, there is a clear

[1] We filtered out runs uploaded by the core team and key contributors listed on https://www.openml.org/contact.

[2] We also filtered out runs uploaded by users whose names contained "bot." Removing developers and bots left us with 6.1% of the total number of runs on OpenML.

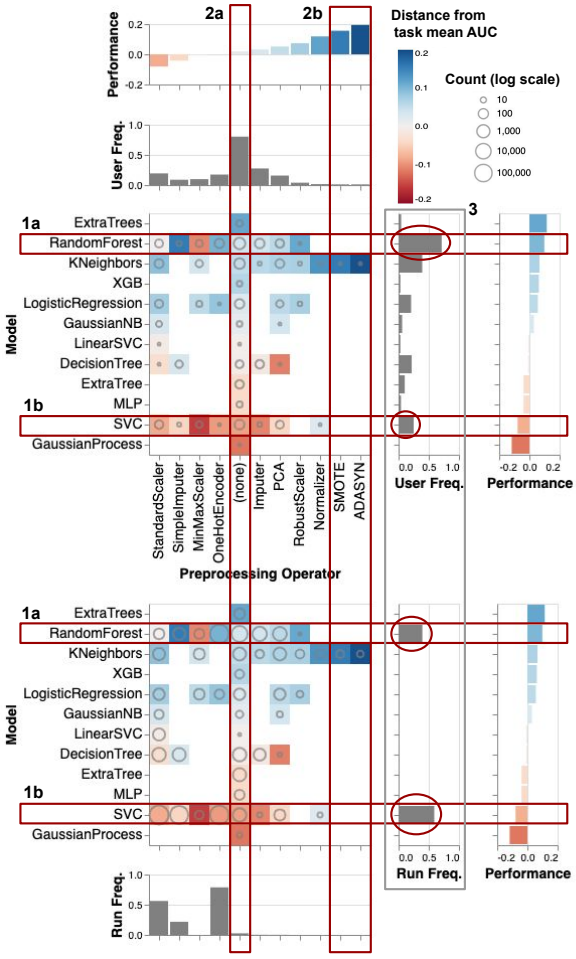

Figure 1: Frequency and performance of the most common (Scikit-Learn model, preprocessing) combinations on OpenML. The top displays frequency in terms of the user count, while the bottom shows run count.

difference in performance between Random Forest (1a), the model used by the most OpenML users, and SVC (C-Support Vector Classification) (1b), the model used in the most runs. On average, the relative AUC of runs that include Random Forest (RF) is $p_r = 9.89\%$. RF works especially well with SimpleImputer as an added preprocessing step, achieving $p_r = 15.57\%$ on average. On the other hand, users tend to perform worse on average ($p_r = -7.84\%$) when using SVC models for OpenML tasks.

(2) Effect of Preprocessing: Around 81% of users have run ML models on a dataset without performing any data preprocessing beforehand (2a). This could be attributed to the fact that many of the datasets on OpenML are already in a relatively clean, preprocessed state. However, it is evident from Figure 1 that users can often still achieve higher performance by including some form of dimensionality reduction, feature scaling and transformations, or data sampling, indicating how *preprocessing is often an overlooked but important aspect in ML development.* For instance, ADASYN and SMOTE [10] are two preprocessing operations that work remarkably well when combined with the K-Nearest Neighbors (KNN) model, but they are adopted by very few users (2b). This phenomenon suggests that there needs to be increased awareness of such preprocessing methods.

Model	Run Freq	User Freq	Avg p_r
DecisionTree	**65.64%**	6.06%	0.57%
KNeighbors	19.02%	**60.61%**	0.48%
RandomForest	9.82%	18.18%	3.1%

Table 1: Frequency and average distance from task mean AUC of the most commonly used models for large datasets.

(3) Frequency of Specific Combinations: The frequency distributions for both models and preprocessing operators vary depending on whether we look at the number of runs or the number of users (shown in box 3). For example, the most popular model measured by the number of unique users is RF (used by 68.2% of users in our dataset) followed by KNN (37.4% of users), with SVC in third place (24.2% of users). However, when determining frequency from the perspective of the number of runs that included the model, SVC makes up $\approx 50\%$ of the runs in our dataset, while RF accounts for $< 40\%$ of the runs. This suggests that there are variations in the iteration behavior across different users: some focus on tuning the same model for many iterations, while others experiment with several different models. We explore these trends and provide insights on their effectiveness in the Section 5.

Application of Run-Level Insights: Analyses such as those highlighted from (1)-(3) not only shed light on the usefulness of particular ML operators, but they can also be used to validate whether users' existing practices aligns with *conventional wisdom* in the form of guidelines from the ML community or whether there is a gap in adopting these guidelines. As an example of this application, we examine a particular case study of how users select which models to use for handling large datasets.

First of all, we observe that for large datasets, users do indeed focus on specific models in a different distribution than the overall trends shown in Figure 1. Table 1 shows the model frequencies for only the runs on large datasets (with greater than 110,313 instances—the mean across all the datasets that Scikit-Learn users constructed workflows for).

According to conventional guidelines [19], Decision Trees and ensemble methods like RF are well-suited for medium to large datasets, while KNN works well for small to medium datasets. Although Decision Tree makes up 65.64% of the runs, it is used by only 6% of the users. Instead, 60% of the users opted for KNN for the largest datasets on OpenML. However, KNN resulted in the lowest performance, (average $p_r = 0.48\%$), compared to average $p_r = 3.1\%$ for RF and 0.57% for Decision Tree. While the performance validates the efficacy of the guidelines, the usage statistics reveal that most users fail to follow these guidelines.

Insights drawn from empirical run-level data as demonstrated in this section can inform the ML community about the prevalence of different operators and whether or not users are able to effectively use them. By knowing which models and preprocessing operators are most commonly used in practice, ML system designers can learn which operations users are already aware of and able to utilize successfully, as well as which ones people are less likely to use yet have high potential for large performance improvements. A

human-in-the-loop ML system [9] could surface these lesser-known but high-potential operators to educate the users and bridge the gap between the system's capabilities and the user's knowledge on specific capabilities.

To better understand how users are constructing their workflows, next, we delve into how users are iterating on their workflows and the impact of different aspects of these iterative changes on the performance of the workflows.

5 SEQUENCE-LEVEL INSIGHTS

Users have a wide range of ML workflow development styles—some use a more manual approach where they run a model, look at the results, make a change or two to address the issue, and then repeat the process. Others may choose a more automated technique, e.g., looping through a set of pre-determined values for certain hyperparameters of a model. In the *manual* case, the human remains *in the loop*, while in the *automated* case, the user has already set a search space a-priori and the changes to the workflow at each iteration are independent of the previous iteration's result. Others use a *mixed* sometimes-manual, sometimes-automated strategy.

To classify a sequence as manual, automated, and mixed, we introduce the following metrics.

- Interval (Δt): difference between start times of consecutive runs
- Interval difference ($\Delta^2 t$): difference between consecutive Δts
- Sequence length ($|S|$): the number of runs in sequence S

Based on these metrics, we categorize each sequence S as follows:

- **Manual**: ($|S| \leq 2$, OR $\Delta t > 10$ minutes for $\geq 50\%$ of the runs in the sequence, OR $\Delta^2 t > 3$ minutes for $\geq 75\%$ of the runs in the sequence), AND $|S| < 300$
- **Automated** (Auto): $|S| > 2$, AND $\Delta t > 10$ minutes for $< 50\%$ of the runs in the sequence, AND $\Delta^2 t > 3$ minutes for $\leq 25\%$ of the runs in the sequence
- **Mixed**: the remaining sequences not in the two categories above

The thresholds were empirically determined through a process of random sampling and spot-checking two sequences from the manual category that had over 30 iterations and two from the automated category that had under 30 iterations (since these would be the more ambiguous cases than shorter manual sequences and longer auto sequences). After the final thresholds were set, five sequences from each of the categories were randomly sampled and spot-checked to validate the sequence labels.

The motivation behind looking at both Δt and $\Delta^2 t$ is that we would expect the actual time difference between run submissions to be at least a couple of minutes if the user was making adjustments manually, and we would expect the change in these Δt's to also be non-zero due to the variability in making changes (while constant time differences are often indicative of a loop). Sequence length is also highly indicative of whether or not most of the runs were manual, since it would be unlikely to find hundreds or thousands of manual runs, and indeed the highest number of runs in a manual sequence (after setting the 300 iteration threshold) is 69. This categorization results in 2181 manual, 208 automated, and 168 mixed sequences.

Overall, we observe that users who adopt a more hands-on approach achieve good performance much faster than users who automate the search. Across the three categories, users exert varying amounts of effort (number of iterations and number of model and preprocessing combinations attempted) and focus on different areas of their workflow (choosing the best model, optimizing a hyperparameter, or determining which data preprocessing operation to add). Our in-depth analyses of the three categories of sequences reveal three major insights on the effectiveness of users iterating with manual and automated iterations, described next.

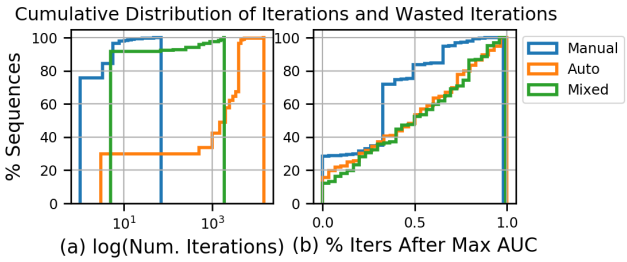

Figure 2: (a) Cumulative distribution of sequence length, (b) % iterations after reaching maximum AUC.

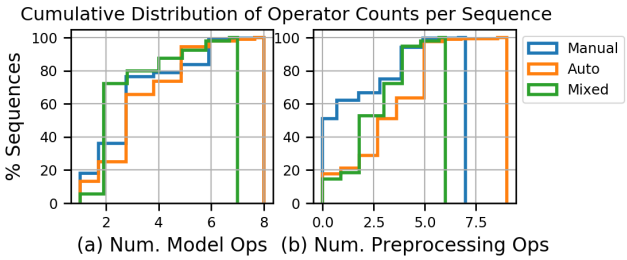

Figure 3: Cumulative distribution of (a) model and (b) preprocessing operators used across all iterations in a sequence.

5.1 On Efficiency

Users iterating with manual sequences are more efficient, achieving the same performance gain in a small fraction of the number of iterations as automated sequences, while wasting fewer iterations searching after reaching their highest-performing workflow. However, users iterating with automated sequences reach a higher p_S than manual for the same task.

Figure 2(a) shows that the distribution of the number of iterations per sequence is drastically different for each sequence category. Manual sequences are short ($\mu = 3.24, \sigma = 3.88$) and automated sequences are the longest ($\mu = 2183.99, \sigma = 1953.74$), while mixed lies somewhere in-between ($\mu = 83.14, \sigma = 283.98$). Even though manual sequences are on average $< 0.2\%$ the length of automated sequences, within the same span of time, the maximum increase in AUC is equal for manual and automated (manual: $\mu = 6.63\%, \sigma = 8.65\%$; auto: $\mu = 7.06\%, \sigma = 12.66\%$)[3]. Moreover, Figure 2(b) shows that the manual group has a lower percentage of wasted iterations, i.e., the iterations after p_S has been achieved, than the other two groups (manual: $\mu = 31\%$; mixed: $\mu = 49\%$; auto: $\mu = 47\%$). This means that on average, manual users waste 1 iteration, mixed users waste 41 iterations, and auto users waste 1026 iterations.

[3]Sequences with a length of 1 were excluded from this and other appropriate analyses in this section to avoid inflation by deltas of 0, counts of 1, or percentages of 100.

Change Type	Manual	Mixed	Auto
Model Operator	**45.04%**	3.95%	0.18%
Model Hyperparameter	28.31%	**75.23%**	**92.69%**
Preprocessing Operator	0.53%	0.55%	0.01%
Preprocessing Hyperparameter	0.18%	0.17%	0.04%
Model & Preprocessing	21.96%	6.60%	5.45%
No Change	3.98%	13.49%	1.63%

Table 2: Change types for manual, mixed, and automated.

However, automated and mixed sequences result in a 3% higher p_S compared to manual sequences for the same task. This is likely due to a greater coverage of search space from the mixed and automated sequences compared to manual. There is a delicate balance between iterating in an efficient manner but exploring enough to achieve better results. We now detail our approach to quantitatively estimating a sequence's breadth of exploration.

5.2 On Exploration

Exploring more model and preprocessing operators leads to higher p_S in manual sequences but does not improve p_S in automated sequences. Users iterating with manual sequences cover the same number of model types as automated sequences, but a lower variety of preprocessing techniques.

As shown in Figure 3, manual sequences explore a similar number of models as mixed and automated sequences (manual: $\mu = 3.08, \sigma = 1.63$; mixed: $\mu = 2.64, \sigma = 1.37$; auto: $\mu = 3.31, \sigma = 1.48$). However, there tends to be less manual exploration on data preprocessing when compared to the mixed and automated groups (manual: $\mu = 1.51, \sigma = 1.82$; mixed: $\mu = 2.49, \sigma = 1.44$; auto: $\mu = 3.22, \sigma = 1.91$). It is interesting to note that automated sequences explore the same number of preprocessing operators as models on average, while manual sequences have a higher tendency to completely leave out data preprocessing from their workflows.

When examining the number of model and preprocessing operators jointly, as shown in Figure 4, we can see that for manual iterations, trying more combinations leads to better performance, but this is not the case in mixed and auto sequences. For each of the combinations that auto sequences explore, they typically perform many more iterations of hyperparameter tuning than manual sequences do for a given combination, evident from the fact that most auto iteration sequences comprise of hyperparameter tuning, as shown in Table 2, and that auto sequences are much longer. These trends reveal two phenomena: 1) when a combination is explored using default or rule-of-thumb hyperparameters in just a few iterations, as is the case in typical manual sequences, the performance gap between different combinations is large; 2) when a combination is explored with many hyperparameter tuning iterations, as is the case in typical auto sequences, the performance gap shrinks between different combinations, leading to diminishing returns from exploring more combinations on p_S improvement. In other words, auto sequences often waste most iterations on hyperparameter tuning to improve the performance of suboptimal combinations without improving p_S, while the potential of a combination can be estimated quickly with just a few iterations. However, there is merit to extensive hyperparameter tuning, evident in the fact that

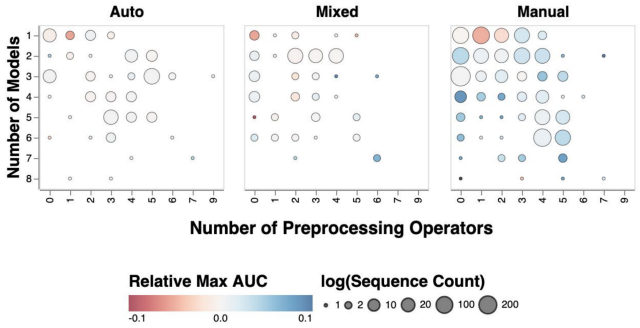

Figure 4: Joint distribution of model and preprocessing operators per sequence. Max AUC is relative to only the sequences within each category.

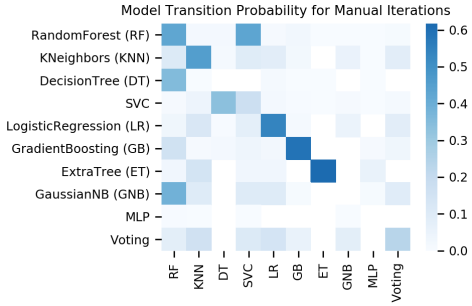

Figure 5: Model transition likelihood in consecutive manual iterations (row represents current iteration, and column represents next iteration).

Model	Manual	Mixed	Auto
RandomForest	40.83%	76.90%	53.32%
KNeighbors	37.99%	37.74%	50.54%
SVC	28.25%	23.73%	46.26%

Table 3: Mean % iterations per sequence for the top 3 models.

auto sequences achieve a 3% higher p_S than manual ones as discussed in Section 5.1. Together, these insights suggest that a hybrid approach, wherein coarse-grained search is first performed over the combinations of preprocessing and model operators, followed by fine-grained search doing hyperparameter tuning on the most promising combinations, would be both effective and also efficient at finding a high-performing workflow.

5.3 On Model Tuning

While automated sequences are dominated by hyperparameter changes, users iterating with manual sequences focus on model selection and eventually converge on high-performing models.

Since changing the ML model is the most common type of manual workflow change, making up 45.04% of all manual runs as shown in Table 2, we look into which models are abandoned and which are kept from one iteration to the next. Table 3 shows that in all groups (manual, mixed, and auto), users are much less likely to stick with SVC thank RF and KNN. Whenever SVC is used in a manual sequence, it accounts for only 28.25% of the runs, compared to 40.83% for RF and 37.99% for KNN. Combined with our findings

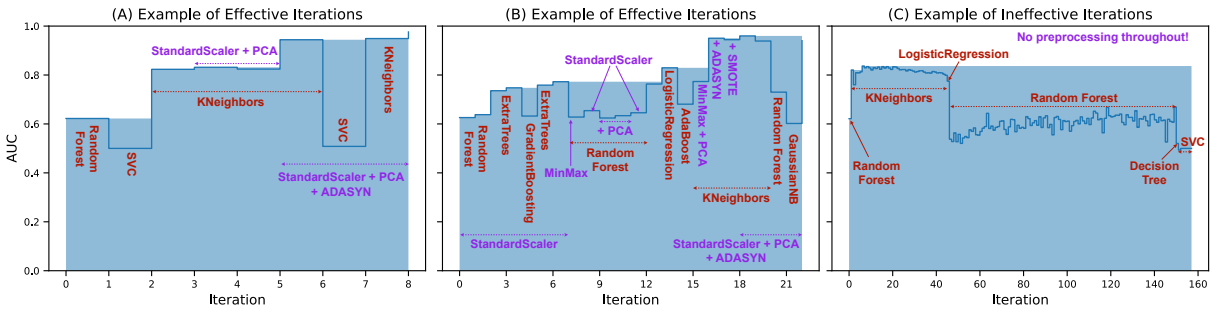

Figure 6: Examples of effective and ineffective sequences for a popular classification task. The shaded blue area represents the maximum AUC up until that point, the red text describes model changes, and the purple text describes preprocessing changes.

from Figure 1 that RF and KNN perform better than SVC on average, we conclude that users are able to waste fewer iterations on models that do not work as well.

Furthermore, the transition probabilities between different pairs of models in Figure 5 reveal that users tend to stay with the same model from one iteration to the next, as evidenced by the high probabilities along the diagonal of the matrix. However, when a switch occurs, the model that is switched to the most is RF, with 26.22% of all model changes transitioning to RF.

6 CASE STUDIES

So far, we have described insights aggregated across multiple users, reflecting the population-level trends in ML development. In this section, we dive deep into a few examples of both *effective* and *ineffective* practices to offer a complementary view on user behavior. We define a highly effective sequence as one that achieves a high AUC score over just a few iterations, wasting few to no iterations after p_S has been achieved.

In Figure 6, we show examples of highly-effective sequences (A and B), and an ineffective sequence (C) for the supervised classification task that was attempted by the highest number of users on OpenML. The dataset used in this binary classification task most notably has a very imbalanced class distribution, with 98.35% of the instances belonging to the majority class. Workflows that account for the class imbalance problem are able to achieve higher performance than those who do not.

Figure 6A) illustrates one such example of a user who was successful in creating a near-optimal ML workflow for this task. This user iterated on the workflow using a *manual* sequence (based on our definition in Section 5), starting off by selecting a model, then adding data preprocessing, beginning with StandardScaler and Principal Component Analysis (PCA). They then incorporated an oversampling strategy, ADASYN, to counter the the class imbalance problem, resulting in a significant performance boost.

Similarly, Figure 6B) also illustrates a user who iterated with a manual sequence, experimenting with different combinations of model and preprocessing operators, before discovering the high impact of data oversampling. As soon as they incorporated ADASYN, they were able to quickly run through a final model selection phase to achieve one of the highest AUC scores for the task.

Finally, as shown in Figure 6C), the user was off to a strong start by picking similar models as chosen in A) and B). However, the user then spent over one hundred *automated* iterations exhaustively tuning hyperparameters, resulting in over 95% of their iterations being wasted, i.e., they did not improve p_S. This user could benefit from learning the strategy adopted in manual approaches in A) and B) to first examine dataset characteristics to guide modeling decisions, instead of optimizing hyperparameters prematurely.

These case studies demonstrate the potential benefits of knowledge transfer from experts to novices, conceivably through a human-in-the-loop system for ML workflow development that guides users with crowdsourced best practices.

7 DISCUSSION AND CONCLUSION

In this study, we set out to demystify the dark art of machine learning workflow development. By analyzing over 475k user-generated runs on OpenML, we discovered that the performance gap between manual and automated approaches to workflow iteration is negligible, but the manual approach is able to find the best performing workflow with much fewer wasted iterations. The model performance parity explains the general enthusiasm around automated machine learning (auto-ML) systems in recent years, but the massive discrepancy in efficiency explains the sluggish growth in adoption of auto-ML despite the hype. Our case studies revealed how human users are often far better at minimizing wasted work than automation. While our study surfaced many interesting patterns in iterative ML workflow development, our dataset provides a limited view into the user's thought process behind performing the manual iterations, as shown in the case studies. A more in-depth user study is a promising direction for future work towards understanding the motivations and cognitive processes behind ML model development. This understanding can form the basis of a more effective and efficient auto-ML strategy by mimicking human experts. Likewise, human-in-the-loop ML systems can benefit from automated guidance suggesting areas of exploration that the ML developer may not have thought of. Our study aims to shed light on design insights for building more usable and more intelligent machine learning development systems, towards the ultimate goal of democratizing machine learning.

Acknowledgments. We thank the anonymous reviewers for their valuable feedback. We acknowledge support from grants IIS-1652750 and IIS-1733878 awarded by the National Science Foundation, grant W911NF-18-1-0335 awarded by the Army, and funds from the Alfred P. Sloan Foundation, Facebook, Adobe, Toyota Research Institute, Google, and the Siebel Energy Institute. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies and organizations.

REFERENCES

[1] [n.d.]. Data Driven Discovery of Models. https://docs.datadrivendiscovery.org/

[2] Saleema Amershi, Andrew Begel, Christian Bird, Rob DeLine, Harald Gall, Ece Kamar, Nachi Nagappan, Besmira Nushi, and Tom Zimmermann. 2019. Software Engineering for Machine Learning: A Case Study. In *International Conference on Software Engineering (ICSE 2019) - Software Engineering in Practice track*. IEEE Computer Society. https://www.microsoft.com/en-us/research/publication/software-engineering-for-machine-learning-a-case-study/

[3] Besim Bilalli, Alberto Abello, and Tomas Aluja-Banet. 2017. On the Predictive Power of Meta-Features in OpenML. *Int. J. Appl. Math. Comput. Sci* 27, 4 (2017), 697–712. https://doi.org/10.1515/amcs-2017-0048

[4] Besim Bilalli, Alberto Abelló, Tomàs Aluja-Banet, and Robert Wrembel. 2019. PRESISTANT: Learning based assistant for data pre-processing. *Data and Knowledge Engineering* (2019). https://doi.org/10.1016/j.datak.2019.101727 arXiv:1803.01024

[5] William La Cava, Heather Williams, Weixuan Fu, and Jason H Moore. 2019. *Evaluating Recommender System for AI-Driven Data Science: A Preprint*. Technical Report. arXiv:1905.09205v2 http://automl.chalearn.org/

[6] Nicolo Fusi, Rishit Sheth, and Melih Huseyn Elibol. 2018. Probabilistic Matrix Factorization for Automated Machine Learning. NeurIPS. arXiv:1705.05355v2

[7] Pieter Gijsbers, Erin LeDell, Janek Thomas, Sébastien Poirier, Bernd Bischl, and Joaquin Vanschoren. 2019. An Open Source AutoML Benchmark. (jul 2019). arXiv:1907.00909 http://arxiv.org/abs/1907.00909

[8] Fred Hohman, Kanit Wongsuphasawat, Mary Beth Kery, and Kayur Patel. 2020. Understanding and Visualizing Data Iteration in Machine Learning. *CHI '20: Proceedings of the 2020 annual conference on Human factors in computing systems* (2020), 1–13.

[9] Doris Jung-Lin Lee, Stephen Macke, Doris Xin, Angela Lee, Silu Huang, and Aditya Parameswaran. 2019. A Human-in-the-loop Perspective on AutoML: Milestones and the Road Ahead. *Data Engineering* 58 (2019).

[10] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. 2017. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research* 18, 17 (2017), 1–5. http://jmlr.org/papers/v18/16-365

[11] Randal S. Olson, William La Cava, Patryk Orzechowski, Ryan J. Urbanowicz, and Jason H. Moore. 2017. PMLB: A large benchmark suite for machine learning evaluation and comparison. *BioData Mining* 10, 1 (dec 2017), 36. https://doi.org/10.1186/s13040-017-0154-4 arXiv:1703.00512

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[13] Martijn J. Post, Peter Van Der Putten, and Jan N. Van Rijn. 2016. Does feature selection improve classification? A large scale experiment in OpenML. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 9897 LNCS. Springer Verlag, 158–170. https://doi.org/10.1007/978-3-319-46349-0_14

[14] Fotis Psallidas, Yiwen Zhu, Bojan Karlas, Matteo Interlandi, Avrilia Floratou, Konstantinos Karanasos, Wentao Wu, Ce Zhang, Subru Krishnan, Carlo Curino, and Markus Weimer. 2019. Data Science through the looking glass and what we found there. (2019). arXiv:1912.09536 http://arxiv.org/abs/1912.09536

[15] Benjamin Strang, Peter van der Putten, Jan N. van Rijn, and Frank Hutter. 2018. Don't rule out simple models prematurely: A large scale benchmark comparing linear and non-linear classifiers in OpenML. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 11191 LNCS. Springer Verlag, 303–315. https://doi.org/10.1007/978-3-030-01768-2_25

[16] Jan N Van Rijn and Frank Hutter. 2018. Hyperparameter Importance Across Datasets. (2018), 10. https://doi.org/10.1145/3219819.3220058 arXiv:1710.04725v2

[17] Joaquin Vanschoren. 2018. *Meta-Learning: A Survey*. Technical Report. arXiv:1810.03548v1

[18] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2013. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations* 15, 2 (2013), 49–60. https://doi.org/10.1145/2641190.2641198

[19] Brett Wujek, Patrick Hall, and Funda Güneş. 2016. *Best Practices for Machine Learning Applications*. Technical Report. 1–23 pages.

[20] Doris Xin, Litian Ma, Shuchen Song, and Aditya Parameswaran. 2018. How Developers Iterate on Machine Learning Workflows–A Survey of the Applied Machine Learning Literature. *KDD IDEA Workshop* (2018).

[21] Qian Yang, Nan-chen Chen, and Gonzalo Ramos. 2018. Grounding Interactive Machine Learning Tool Design in How Non-Experts Actually Build Models. (2018).