A Systematic Literature Review on Explainability for Machine/Deep Learning-based Software Engineering Research

SICONG CAO, Yangzhou University, China
XIAOBING SUN*, Yangzhou University, China
RATNADIRA WIDYASARI, Singapore Management University, Singapore
DAVID LO, Singapore Management University, Singapore
XIAOXUE WU, Yangzhou University, China
LILI BO, Yangzhou University, China
JIALE ZHANG, Yangzhou University, China
BIN LI, Yangzhou University, China
WEI LIU, Yangzhou University, China
DI WU, University of Southern Queensland, Australia
YIXIN CHEN, Washington University in St. Louis, USA

The remarkable achievements of Artificial Intelligence (AI) algorithms, particularly in Machine Learning (ML) and Deep Learning (DL), have fueled their extensive deployment across multiple sectors, including Software Engineering (SE). However, due to their black-box nature, these promising AI-driven SE models are still far from being deployed in practice. This lack of explainability poses unwanted risks for their applications in critical tasks, such as vulnerability detection, where decision-making transparency is of paramount importance. This paper endeavors to elucidate this interdisciplinary domain by presenting a systematic literature review of approaches that aim to improve the explainability of AI models within the context of SE. The review canvasses work appearing in the most prominent SE & AI conferences and journals, and spans 63 papers across 21 unique SE tasks. Based on three key Research Questions (RQs), we aim to (1) summarize the SE tasks where XAI techniques have shown success to date; (2) classify and analyze different XAI techniques; and (3) investigate existing evaluation approaches. Based on our findings, we identified a set of challenges remaining to be addressed in existing studies, together with a roadmap highlighting potential opportunities we deemed appropriate and important for future work.

*Corresponding author

Authors' addresses: Sicong Cao, School of Information Engineering, Yangzhou University, Yangzhou, China, DX120210088@yzu.edu.cn; Xiaobing Sun, School of Information Engineering, Yangzhou University, Yangzhou, China, xbsun@yzu.edu.cn; Ratnadira Widyasari, School of Computing and Information Systems, Singapore Management University, Singapore, ratnadiraw.2020@phdcs.smu.edu.sg; David Lo, School of Computing and Information Systems, Singapore Management University, Singapore, davidlo@smu.edu.sg; Xiaoxue Wu, School of Information Engineering, Yangzhou University, Yangzhou, China, xiaoxuewu@yzu.edu.cn; Lili Bo, School of Information Engineering, Yangzhou University, Yangzhou, China, lilibo@yzu.edu.cn; Jiale Zhang, School of Information Engineering, Yangzhou University, Yangzhou, China, jialezhang@yzu.edu.cn; Bin Li, School of Information Engineering, Yangzhou University, Yangzhou, China, lb@yzu.edu.cn; Wei Liu, School of Information Engineering, Yangzhou University, Yangzhou, China, weiliu@yzu.edu.cn; Di Wu, School of Mathematics, Physics, and Computing, University of Southern Queensland, Toowoomba, Australia, di.wu@unisq.edu.au; Yixin Chen, Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, USA, chen@cse.wustl.edu.

Manuscript submitted to ACM

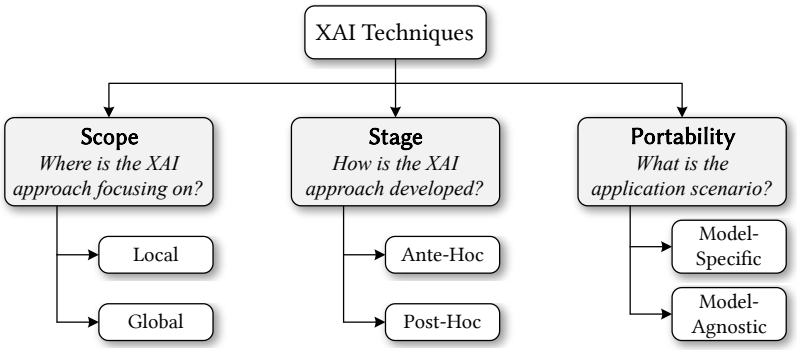


Fig. 1. General taxonomy of the survey in terms of scope, stage, and portability.

challenges that still need to be solved in future work and outlines a clear research roadmap of research opportunities. Section 9 concludes this paper.

2 EXPLAINABLE ARTIFICIAL INTELLIGENCE: PRELIMINARIES

This section first details the definitions of several critical terminologies commonly used in the XAI field. Then, we offer a general overview of the taxonomy of XAI approaches, aiming to furnish the reader with a solid comprehension of this topic.

2.1 Definition

The greatest challenge in establishing the concept of XAI in SE is the ambiguous definition of *interpretability* and *explainability*. Those terms, together with *interpretation* and *explanation*, are often used interchangeably in the literature [5, 65]. For example, quoting Doshi-Velez and Kim et al. [19], interpretability is the ability "to explain or to present in understandable terms to a human." By contrast, according to Lent et al. [95], an explainable AI means it can "present the user with an easily understood chain of reasoning from the user's order, through the AI's knowledge and inference, to the resulting behavior." Some argue that the terms are closely related but distinguish between them, although there is no consensus on what the distinction exactly is [3, 66]. To ensure that we do not exclude work because of different terminologies, we equate them (and use them interchangeably) to keep a general, inclusive discussion regardless of this debate. In this survey, we frame explanations in the context of SE research using ML/DL and adopt the phrasing of Dam et al. [11] as follows:

> ***Definition 1:*** *Explainability or Interpretability of an AI-powered SE model measures the degree to which a human observer can understand the reasons behind its decision (e.g. a prediction).*

Under this context, there are two distinct ways of achieving explainability: (❶) making the entire decision-making process transparent and comprehensible (i.e., white-box/interpretable models); and (❷) explicitly providing an explanation for each decision (i.e., surrogate models). In addition, since SE is task-oriented, explanations in SE tasks should be viewed from a perspective that values practical use [111]. As we observed in Section 5.2, there are multiple legitimate types of explanations for SE practitioners who have different intents and expertise.

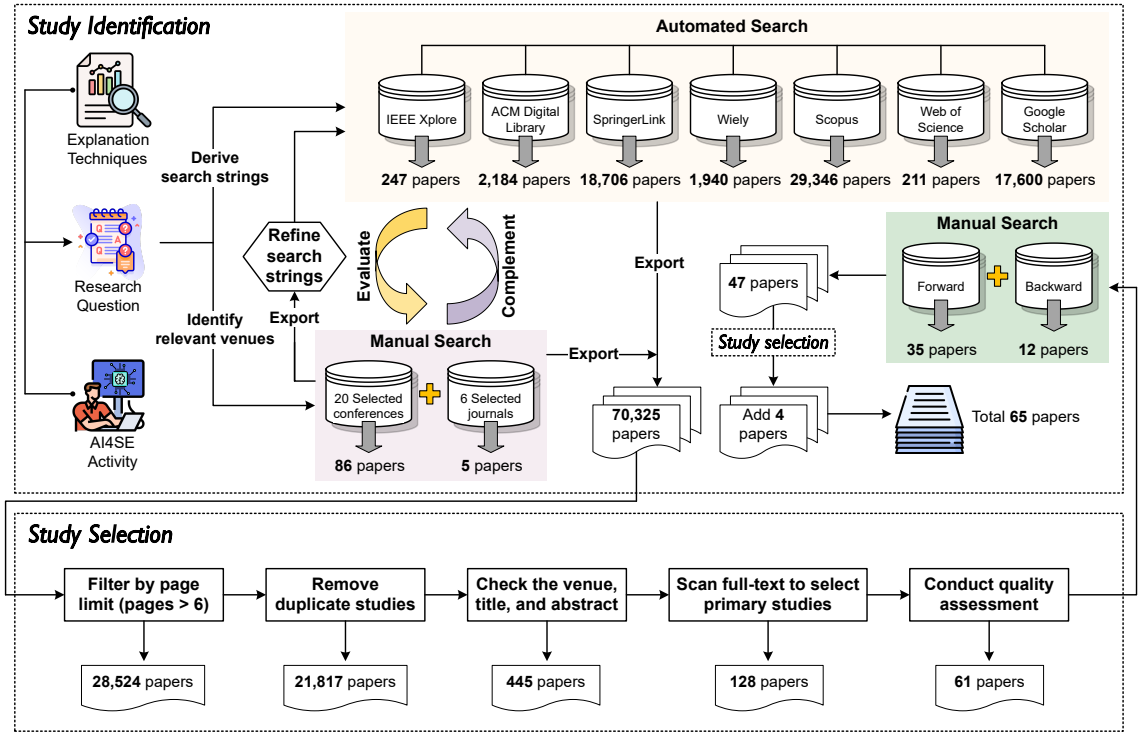


Fig. 2. Study identification and selection process.

3.2 Search Strategy

As shown in Fig.2, following the standardized practice within the field of SE [128], our first step involves identifying primary studies to enhance our ability to address the formulated RQs effectively. Given that the DL revolution – triggered by AlexNet [42] in 2012 – has transformed AI research and became the catalyst for the ML/DL boom in all fields including SE, we chose a 12-year period of January 1st, 2012, to December 31st, 2023, to collect the literature related to XAI4SE. Next, we identified the top peer-review and influential conference and journal venues in the domains of SE and Programming Languages (PL), as outlined in Table 2. In total, we included 15 conferences (ICSE, ASE, ESEC/FSE, ICSME, ICPC, RE, ESEM, ISSTA, MSR, SANER, ISSRE, COMPSAC, QRS, OOPSLA, PLDI) and six journals (TSE, TOSEM, EMSE, JSS, IST, ASEJ). We chose to include PL venues in our study given the frequent overlap of SE and PL research. Furthermore, we also include five top conferences (AAAI, ICML, ICLR, NeurIPS, IJCAI) that centered on machine learning (ML) and deep learning (DL) as these conferences might feature papers applying ML and DL techniques to SE tasks.

Apart from manually searching primary studies from top-tier venues, we also retrieved relevant papers from five popular digital libraries, including IEEE Xplore[1], ACM Digital Library[2], SpringerLink[3], Wiely[4], and Scopus[5], and two of

[1] https://ieeexplore.ieee.org
[2] https://dl.acm.org
[3] https://link.springer.com
[4] https://onlinelibrary.wiley.com
[5] https://www.scopus.com

Table 7. SE Task Taxonomy

SE Activity	SE Task	# Papers	References
Software Development	Code Understanding	6	[9, 75, 79, 99, 103, 117]
	Program Synthesis	3	[12, 21, 67]
	Code Summarization	3	[30, 39, 71]
	Code Search	1	[100]
Software Testing	Test Case-Related	4	[1, 41, 91, 124]
	Debugging	3	[8, 31, 43]
	Vulnerability Detection	11	[26, 36, 47, 52, 68, 90, 93, 129, 130, 135, 137]
	Bug/Fault Localization	2	[48, 110]
Software Maintenance	Program Repair	1	[58]
	Malware/Anomaly Detection	4	[46, 51, 113, 132]
	Bug/Defect Prediction	12	[17, 28, 29, 40, 44, 57, 63, 74, 76, 107, 116, 133]
	OSS Sustainability Prediction	1	[114]
	Incident Prediction	1	[131]
	Root Cause Analysis	1	[15]
	Code Review	1	[118]
	Code Smell Detection	3	[77, 102, 122]
	Bug Report-Related	2	[14, 84]
Software Management	Information Search on Q&A Sites	1	[50]
	Configuration Extrapolation	1	[16]
	Effort/Cost Estimation	1	[27]
	Developer Recommendation	1	[115]

39.7% of the total publications. Fig. 3b illustrates the cumulative publication trend over years. It is observable that the slope of the curve fitting the distribution experiences a significant increase between 2019 and 2023. This pronounced upward trend indicates a burgeoning research interest in the field of XAI4SE.

We also analyzed the publication trend of primary studies in selected conferences and journal venues, respectively. As shown in Fig. 4a, ESEC/FSE stands out as the predominant conference venues favored by XAI4SE studies, with a contribution of 28.3% of the total. Other venues making noteworthy contributions include ICSE (13%), ASE (10.9%), and SANER (6.5%). Fig. 4b shows the distribution of primary papers published in different journal venues. It can be seen that 76.5% of relevant papers were published in TSE and TOSEM, which indicates a booming trend of XAI4SE research in top-tier SE journals in the past few years.

4 RQ_1: WHAT TYPES OF AI4SE STUDIES HAVE BEEN EXPLORED FOR EXPLAINABILITY?

This RQ aims to investigate the application scenarios of XAI techniques in helping improve the explainability of various AI4SE models. Out of the 63 primary studies we analyzed for this SLR, we identified 21 separate SE tasks where an XAI technique had been applied. These tasks span across the four main phases of the **Software Development Life Cycle (SDLC)** [82], including software development, software testing, software maintenance, and software management. The full taxonomy is displayed in Table 7, which associates the relevant primary study paired with the SE task & activity it belongs to.

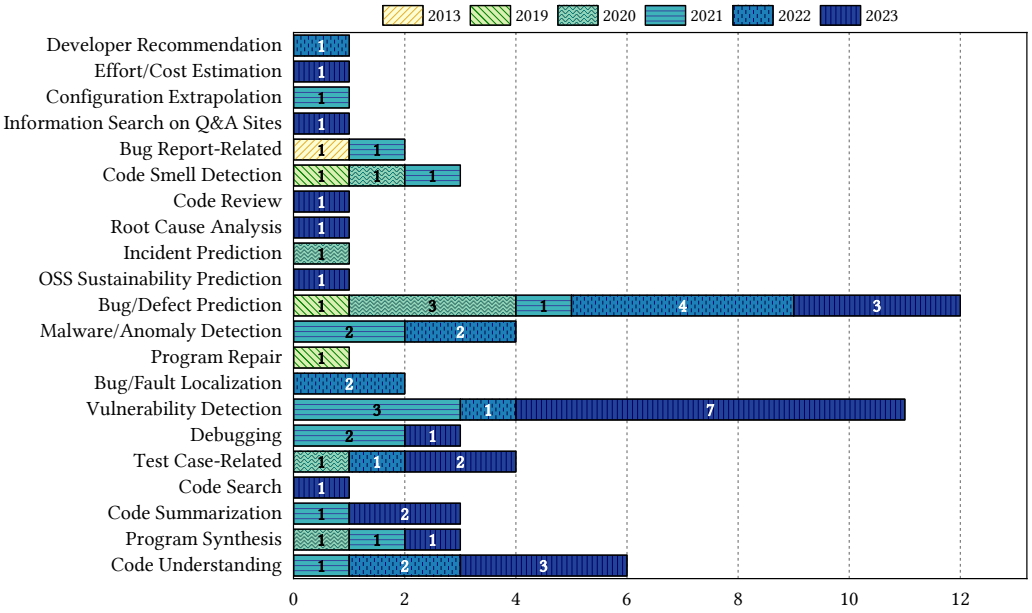


Fig. 6. Papers published per year according to SE task.

4.2 How XAI Were Used in Specific SE Tasks?

In this subsection, we delved into the progress of various SE tasks that applied XAI techniques. By investigating this RQ, we aimed to obtain a clear understanding of what has been done and what else can be done in advancing practices for explainable AI4SE solutions.

4.2.1 SE Tasks in Software Development. There are wide-ranging applications of XAI techniques in software development, encompassing code understanding, program synthesis, code summarization, and code search.

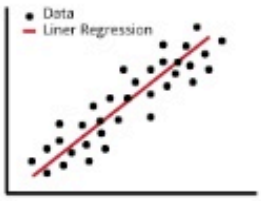
Code Understanding. Code understanding refers to the process of comprehending and analyzing source code deeply. Within the context of data-driven SE research, code understanding aims to seek an effective way to map source code into high-dimensional semantic space, thereby supporting a variety of code-centric downstream tasks, such as such as *Variable Misuse Detection* [75], *Method Name Prediction* [103], and *Vulnerability Detection* [117]. Inspired by the capability of complex AI models, deep neural networks in particular, in learning rich representations of raw data, a series of code models are trained on labeled (e.g., CodeSearchNet [37]) or unlabeled code corpus (e.g., CodeXGlue [53]). This training process produces code embeddings with rich contexts and semantics. Yang et al. [117] proposed Graph Tensor Convolution Neural Network (GTCN), a novel code representation learning model which is capable of comprehensively capturing the distance information of code sequences and structural code semantics, to generate accurate code embeddings. GTCN was self-explainable because the tensor-based model reduced model complexity, which was beneficial for capturing the data features from the simpler model space. Wan et al. [99] proposed three types of structural analysis, including attention analysis, probing on the word embedding, and syntax tree induction, to explore why the pre-trained language models work and what they indeed capture in SE tasks.

Fig. 7. XAI techniques taxonomy & distribution.

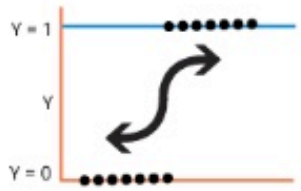
tree structure is ideal for capturing interactions between features in the data, and also has a natural visualization of a decision making process. Taking Fig. 8 as an example, each node in a decision tree may refer to an explanation, e.g. when the `time_days` variable (i.e., the number of days to fix the bug) is greater than 13.9 and the last status is `Resolved Fixed`, then the bug will be re-opened [84]. In addition, interpretable models can serve as post-hoc surrogates to explain individual predictions of black-box models. The goal behind this insight is to leverage a relatively simpler and transparent model to approximate the predictions of the complicated model as best as possible, and at the same time, provide explainability. Surrogate models have shown effectiveness in explaining AI4SE approaches built upon more complex ML/DL models, e.g., deep neural networks, SVMs, or ensemble models. Examples include vulnerability detection [137], defect prediction [74], and program repair [58].

Feature Attribution (FA). Feature attribution-based explanations aim to measure the relevance (i.e., attribution score) of each input feature to a model's prediction by ranking the explanatory score. The score could range from a positive value that shows its contribution to the model's prediction, to a zero that would mean the feature has no contribution, to a negative value which means that removing that feature would increase the probability of the predicted class. They can be broadly divided into (❶) perturbation-based approaches that make feature perturbations while analyzing prediction change, e.g., LIME [78], GNNExplainer [123], and SHAP [55], (❷) gradient-based approaches that propagate importance signals backward through all neurons of the network, e.g., Integrated Gradients (IG) [92] and Grad-CAM [83], and (❸)

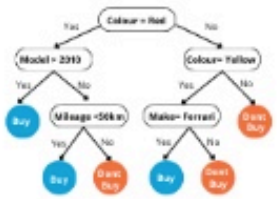
Linear Regression



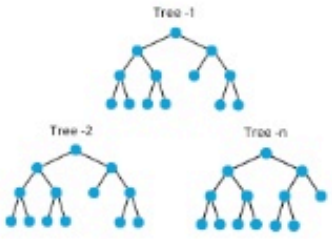
Logistic Regression



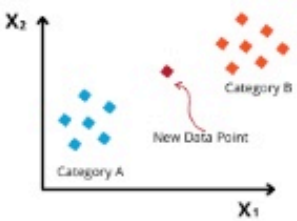
Decision Trees



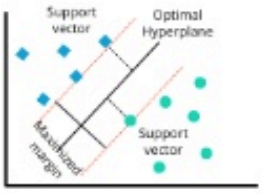
Random Forest



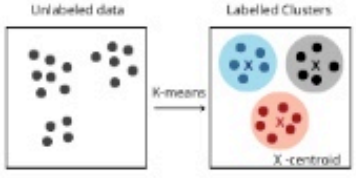
K-Nearest Neighbor



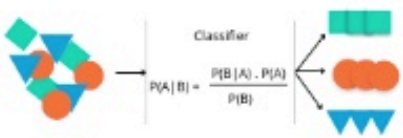
Support Vector Machine



K-Means Clustering



Naïve Bayes



Explainable Artificial Intelligence: a Systematic Review

Giulia Vilone, Luca Longo

School of Computer Science, College of Science and Health,
Technological University Dublin, Dublin, Republic of Ireland

Abstract

Explainable Artificial Intelligence (XAI) has experienced a significant growth over the last few years. This is due to the widespread application of machine learning, particularly deep learning, that has led to the development of highly accurate models but lack explainability and interpretability. A plethora of methods to tackle this problem have been proposed, developed and tested. This systematic review contributes to the body of knowledge by clustering these methods with a hierarchical classification system with four main clusters: review articles, theories and notions, methods and their evaluation. It also summarises the state-of-the-art in XAI and recommends future research directions.

Keywords: Explainable artificial intelligence, method classification, survey, systematic literature review

1. Introduction

The number of scientific articles, conferences and symposia around the world in eXplainable Artificial Intelligence (XAI) has significantly increased over the last decade [1, 2]. This has led to the development of a plethora of domain-dependent and context-specific methods for dealing with the interpretation of machine learning (ML) models and the formation of explanations for humans. Unfortunately, this trend is far from being over, with an abundance of knowledge in the field which is scattered and needs organisation. The goal of this article is to systematically review research works in the field of XAI and to try to define some boundaries in the field. From several hundreds of research articles focused on the concept of explainability, about 350 have been considered for review by using the following search methodology. In a first phase, Google Scholar was queried to find papers related to "explainable artificial intelligence", "explainable machine learning" and "interpretable machine learning". Subsequently, the bibliographic section of these articles was thoroughly examined to retrieve further relevant scientific studies. The first noticeable thing, as shown in figure 2 (a), is the distribution of the publication dates of selected research articles: sporadic in the 70s and 80s, receiving preliminary attention in the 90s, showing raising interest in 2000 and becoming a recognised body of knowledge after 2010. The first research concerned the development of an explanation-based system and its integration in a computer program designed to help doctors make diagnoses [3]. Some of the more recent papers focus on work devoted to the clustering of methods for explainability, motivating the need for organising the XAI literature [4, 5, 6]. The upturn in the XAI research outputs of the last decade is prominently due to the fast increase in the popularity of ML and in particular of deep learning (DL), with many applications in several business areas, spanning from e-commerce [7] to games

Table 1: Definition of the notions related to the concept of explainability

Notion	Description & Reference
Algorithmic transparency	The degree of confidence of a learning algorithm to behave 'sensibly' in general [26, 2]
Actionability	The capacity of a learning algorithm to transfer new knowledge to end-users [60, 61]
Causality	The capacity of a method for explainability to clarify the relationship between input and output [12, 58, 57, 59, 56, 23]
Completeness	The extent to which an underlying inferential system is described by explanations [53, 60, 61]
Comprehensibility	The quality of the language used by a method for explainability [62, 63, 64, 65, 66, 13, 67, 68, 69]
Cognitive relief	The degree to which an explanation decreases the "surprise value" which measures the amount of cognitive dissonance between the explanandum and the user's beliefs. The explanandum is something unexpected by the user that creates dissonance with his/her beliefs [58]
Correctability	The capacity of a method for explainability to allow end-users make technical adjustments to an underlying model [60, 61]
Effectiveness	The capacity of a method for explainability to support good user decision-making [70, 71, 72]
Efficiency	The capacity of a method for explainability to support faster user decision-making [70, 55, 71]
Explicability	The degree of association between the expected behaviour of a robot to achieve assigned tasks or goals and its actual observed actions [73]
Explicitness	The capacity of a method to provide immediate and understandable explanations [74]
Faithfulness	The capacity of a method for explainability to select truly relevant features [74]
Intelligibility	The capacity to be apprehended by the intellect alone [75, 76, 5, 77, 78]
Interactivity	The capacity of an explanation system to reason about previous utterances both to interpret and answer users' follow-up questions [79, 80]
Interestingness	The capacity of a method for explainability to facilitate the discovery of novel knowledge and to engage user's attention [64, 81, 67, 65, 82]
Interpretability	The capacity to provide or bring out the meaning of an abstract concept [64, 50, 83, 66, 13, 22, 29, 84, 85, 4, 6, 86]
Informativeness	The capacity of a method for explainability to provide useful information to end-users [56]
Justifiability	The capacity of an expert to assess if a model is in line with the domain knowledge [1, 64, 50, 33]
Mental Fit	The ability for a human to grasp and evaluate a model [64, 87]
Monotonicity	The relationship between a numerical predictor and the predicted class that occurs when increasing the value of the predictor leads to either always increase or decrease the probability of an instance's membership to the class [88]
Persuasiveness	The capacity of a method for explainability to convince users perform certain actions [70, 55, 71]
Predictability	The capacity to anticipate the sequence of consecutive actions in a plan [73]
Refinement	The capacity of a method to guide experts in improving the performance/robustness of a model [89]
Reversibility	The capacity to allow end-users to bring a ML-based system to an original state after it has been exposed to an harmful action that makes its predictions worse [60, 61]
Robustness	The persistence of a method for explainability to withstand small perturbations of the input that do not change the prediction of the model [90, 89]
Satisfaction	The capacity of a method to increase the ease of use and usefulness of a ML-based system [70, 55, 71]
Scrutability / diagnosis	The capacity of a method for explainability to inspect a training process that fails to converge or does not achieve an acceptable performance [89, 70, 55]
Security	The reliability of a model to perform to a safe standard across all reasonable contexts [91]
Selection / simplicity	The ability of a method for explainability to select only the causes that are necessary and sufficient to explain the prediction of an underlying model [23]
Sensitivity	The capacity of a method for explainability to reflect the sensitivity of the underlying model with respect to variations in the input feature space [92, 93]
Simplification	The capacity to reduce the number of variables under consideration to a set of principal ones [94]
Soundness	The extent to which each component of an explanation's content is truthful in describing an underlying system [60, 61]
Stability	The consistency of a method to provide similar explanations for similar/neighboring inputs [74]
Transparency	The capacity of a method to explain how the system works even when it behaves unexpectedly [76, 26, 13, 95, 84, 14, 15, 70, 55, 16, 96, 86]
Transferability	The capacity of a method to transfer prior knowledge to unfamiliar situations [56]
Understandability	The capacity of a method of explainability to make a model understandable [75, 63, 64, 89, 97]

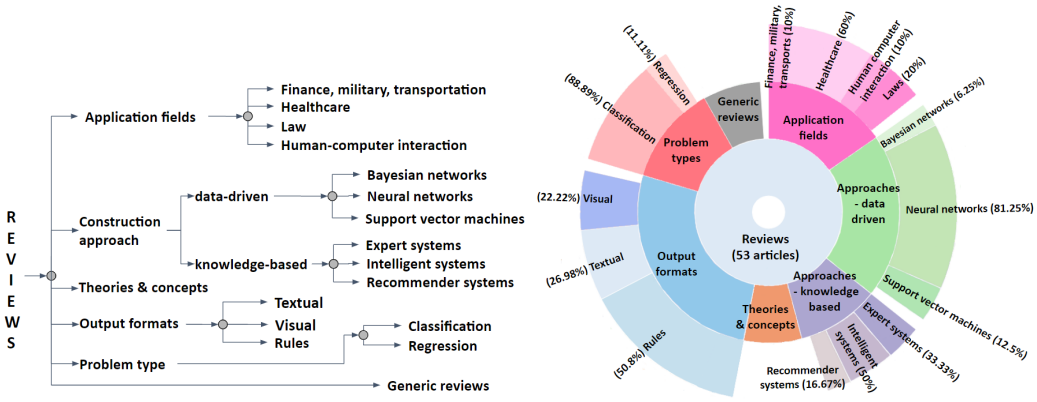


Figure 3: Hierarchical classification of the review articles on explainable artificial intelligence and machine learning interpretability (left) and distribution of the review articles across categories (right).

notions, which are discussed in depth in section 5.1. One of these studies presented an overview of different theories of explanation borrowed from the cognitive science and philosophy disciplines, contextualised within case-based reasoning [35]. In details, it is believed that, in order to be effective, an AI system should: (I) explain how it reached the answer and (II) why it is a good answer, (III) why a question is relevant or not, (IV) clarify the meaning of the terms used in the system that might not be understood by the users and, lastly, (V) teach the user about the domain. In short, the goals that an explanation must achieve depend on the domain under consideration, the underlying model and end-users. Similarly, [23] suggested that explanations should take into account the preferences and preconceptions of end-users. This can be achieved by incorporating more findings from the behavioural and social sciences into the newly emerging field of XAI. For example, people explain their behaviour based on their beliefs, desires and intentions hence these elements must be considered in an explanation. Eventually, explanations based on counterfactual examples should help end-users to understand the logic of an underlying model by leveraging on people's capability to infer general rules from a few examples. Counterfactuals add also something new to what is already known from the existing data and provide additional information on how a model behaves in novel, unseen situations [36]. The fourth cluster contains reviews of methods for explainability generating a specific output format for an explanation (further discussed in section 6). Methods generating textual explanations are surveyed in [32] and compared according to some requirements about the structure and content of the explanations to adapt them to the users' needs and knowledge. [37] focused on written explanations generated from fuzzy rules integrated with natural language generation tools. The underlying reasonable assumption is that the understandability of these rules cannot be given for granted. Researchers studied the capabilities of 'data-to-text' approaches that automatically create linguistic descriptions from a complex dataset by means of aggregation functions, implemented as fuzzy rules, that aggregate 'computational perceptions'. A computational perception is "a unit of meaning for the phenomenon under analysis and is identified by a set of linguistic expressions and their corresponding validity values given a situation." Some methods combine Logical AI and Statistical AI to generate textual explanations [38]. The former is concerned with 'formal languages' to represent and reason with qualitative specifications, while the latter is focused on learning quantitative specifications from data. However, the authors claimed that the search for

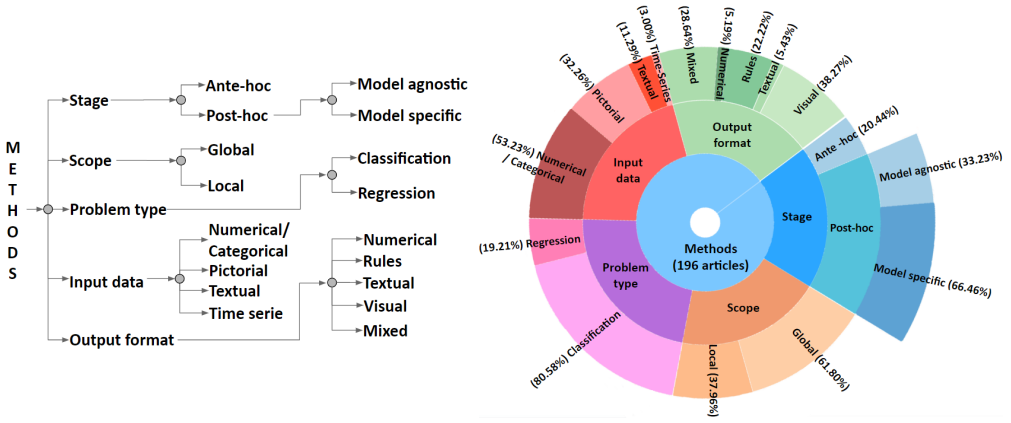


Figure 4: Classification of methods for explainability (left) and distribution of articles across categories (right).

The following sections try to succinctly describe the main classes of methods for explainability found during this systematic review, accompanied by tables for reporting their stage, scope, problem type, input data and output format and sorting them in alphabetic order. Given the large number of methods found, it was decided to group them into five thematic classes.

6.1. Output formats

Visual explanations are probably the most natural way of communicating things and a very appealing way to explain them. Visual explanations can also be used to illustrate the inner functioning of a model via graphical tools. For instance, heat-maps can highlight specific areas of an image or specific words of a text that mostly influence the inferential process of a model by using different colours [134, 135]. Similarly, a graphical representation can be employed to represent the inner structure of a model, such as the graphs proposed in [136] where each node is a layer of the network and the edges the connections between layers. Another intuitive form of explanation for humans are *textual explanations*, natural language statements that can be either written or orally uttered. An example is the phrase "This is a Brewer Blackbird because this is a blackbird with a white eye and long pointy black beak" shown by an explainer of an image classification model [137]. A schematic, logical format, more structured than visual and textual explanations but still intuitive for humans, are *rules* that can be used to explain the inferences produced by models induced from data. Rules can be in the form of 'IF ... THEN' statements with *AND/OR* operators and they are very useful for expressing combinations of input features and their activation values [138, 139]. Technically, rules of these type employ symbolic logic, a formalized system of primitive symbols and their combinations (example: '$(Country = USA) \wedge (28 < Age <= 37) \rightarrow (Salary > 50K)$' [140]). The parts before and after the $\rightarrow$ logical operator are respectively referred to as antecedent and consequent. Given this logic, rules can be implemented as fuzzy rules, linking one or more premises to a consequent that can be true to a degree, instead of being entirely true or false. This can be obtained by representing each antecedent and consequent as fuzzy sets [43]. Combining fuzzy rules with learning algorithms can become a powerful tool to perform reasoning and, for instance, explain the inner logic of neural networks [141]. Similarly, the combination of antecedents and consequent can be seen as an argument in the discipline of argumentation, and a set of arguments can

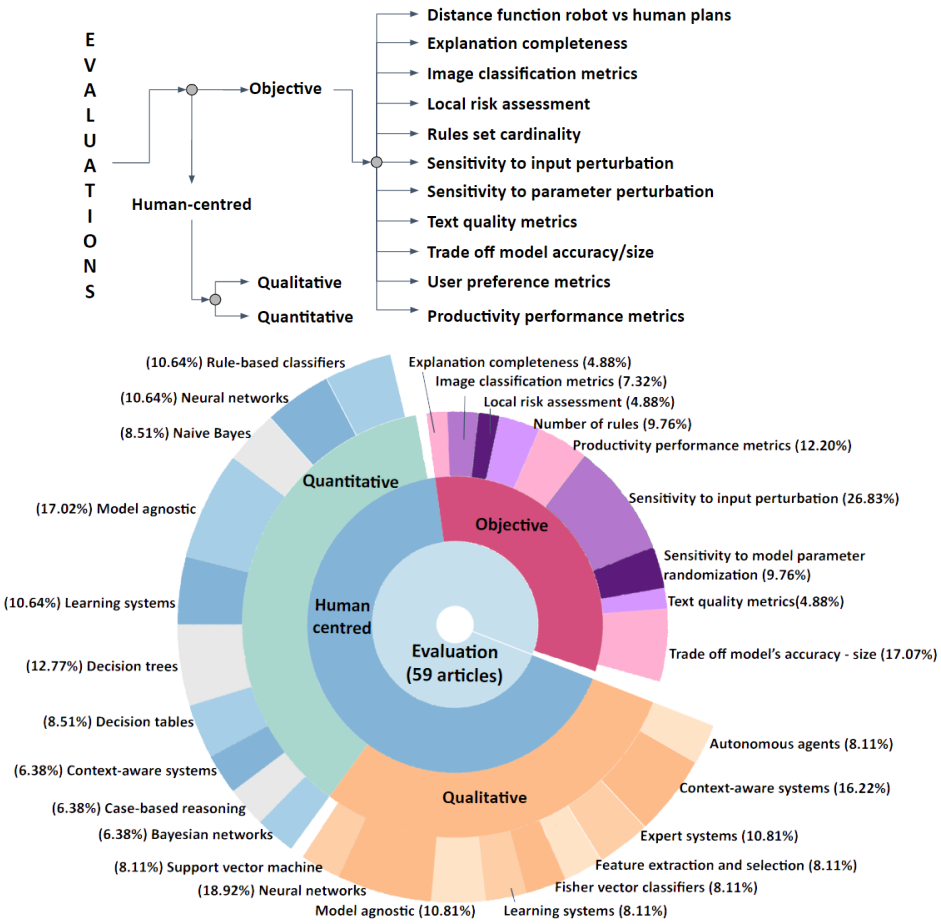


Figure 17: Classification of the approaches to evaluate methods for explainability (up) and distribution of the relative scientific studies across categories (down).

users of these models when following these explanations instead of those produced by human engineers, as done in [26, 2]. Because they involve humans, they are merged into the human-grounded metrics.

7.1. Objective evaluations

Scholars proposed several metrics to evaluate, formally and objectively, the methods for explainability, listed in table 1. In the scientific literature, there is consensus that simpler learning techniques, such as linear regression and DTs, can lead to more transparent inferences than more complex techniques, such as neural network, as they are intrinsically self-interpretable [26]. However, these simpler techniques usually do not lead to the construction of models with the same level of accuracy than those induced by more complex learning techniques. The interpretability of these models depends on many factors such as the learning algorithm, the learning architecture and its configuration (hyper-parameters).

Applications of Psychological Science for Actionable Analytics

Di Chen, Wei Fu, Rahul Krishna, Tim Menzies
North Carolina State University, USA
Raleigh, NC
{dchen20,wfu,rkrish11}@ncsu.edu,tim.menzies@gmail.com

ABSTRACT

Actionable analytics are those that humans can understand, and operationalize. What kind of data mining models generate such actionable analytics? According to psychological scientists, humans understand models that most match their own internal models, which they characterize as lists of "heuristic" (i.e., lists of very succinct rules). One such heuristic rule generator is the Fast-and-Frugal Trees (FFT) preferred by psychological scientists. Despite their successful use in many applied domains, FFTs have not been applied in software analytics. Accordingly, this paper assesses FFTs for software analytics.

We find that FFTs are remarkably effective. Their models are very succinct (5 lines or less describing a binary decision tree). These succinct models outperform state-of-the-art defect prediction algorithms defined by Ghortra et al. at ICSE'15. Also, when we restrict training data to operational attributes (i.e., those attributes that are frequently changed by developers), FFTs perform much better than standard learners.

Our conclusions are two-fold. Firstly, there is much that software analytics community could learn from psychological science. Secondly, proponents of complex methods should always baseline those methods against simpler alternatives. For example, FFTs could be used as a standard baseline learner against which other software analytics tools are compared.

KEYWORDS

Decision trees, heuristics, software analytics, psychological science, empirical studies, defect prediction

ACM Reference Format:
Di Chen, Wei Fu, Rahul Krishna, Tim Menzies. 2018. Applications of Psychological Science for Actionable Analytics. In *Proceedings of Florida (FSE'18).* ACM, New York, NY, USA, 12 pages. https://doi.org/XX.YY/ZZ

1 INTRODUCTION

Data mining tools have been applied to many applications in Software Engineering (SE). For example, it has been used to estimate how long it would take to integrate new code into an existing project [15], where defects are most likely to occur [46, 55], or how long will it take to develop a project [33, 66], etc. Large organizations like Microsoft routinely practice data-driven policy development where organizational policies are learned from an extensive analysis of large datasets [6, 65].

Despite these successes, there exists some drawbacks with current software analytic tools. At a recent workshop on "Actionable Analytics" at ASE'15, business users were very vocal in their complaints about analytics [27], saying that there are rarely producible models that business users can understand or operationalize.

Accordingly, this paper explores methods for generating actionable analytics for:

- Software defect prediction;
- Predicting close time for Github issues.

There are many ways to define "actionable" but at the very least, we say that something is actionable if people can *read* and *use* the models it generates. Hence, for this paper, we assume:

Actionable = Comprehensible + Operational.

We show here that many algorithms used in software analytics generate models that are not actionable. Further, a data mining algorithm taken from psychological science [9, 14, 22–24, 42, 43, 54], called Fast-and-Frugal trees (FFTs[1]), are very actionable.

Note that demanding that analytics be actionable also imposes certain restrictions on (a) the kinds of models that can be generated and (b) the data used to build the models.

(a) Drawing on psychological science, we say an automatically generated model is *comprehensible* if:
   - The model matches the models used internally by humans; i.e., it comprises small rules.
   - Further, for expert-level comprehension, the rules should quickly lead to decisions (thus freeing up memory for other tasks).

   For more on this point, see Section 2.2.

(b) As to *operational*, we show in the historical log of software projects that only a few of the measurable project attributes are often changed by developers. For a data mining algorithm to be operational, it must generate effective models even if restricted to using just those changed attributes.

Using three research questions, this paper tests if these restrictions damage our ability to build useful models.

RQ1: Do FFTs models perform worse than the current state-of-the-art? We will find that:

FSE'18, Nov 2018,

ACM ISBN ZZ-YY-24-ZZ/QQ/A…$15.00
https://doi.org/XX.YY/ZZ

[1] The reader might be aware that FFT is also an acronym for "Fast Fourier Transform". Apparently, the psychological science community was unaware of that acronym when they named this algorithm.

For defect prediction, FFTs out-perform the state-of-art.
When compared to state-of-the-art defect prediction algorithms surveyed by Ghotra et al. [21], FFTs are more effective (where "effective" is measured in terms of a recall/false alarm metric or the P_{opt} metric defined in § 3.4).

RQ2: Are FFTs more operational than the current state-of-the-art? This research question tests what happens when we learn from less data; i.e., if we demand our models *avoid* using attributes that are rarely changed by developers. We show that:

When learning from less data, FFTs performance is stabler than some other learners.
When data is restricted to attributes that developers often change, then FFTs performance is only slightly changed while the performance of some other learners, can vary by alarmingly large amounts.

The observed superior performance of FFT raises the question:

RQ3: Why do FFTs work so well? Our answer to this question will be somewhat technical but, in summary we will say:

FFTs match the structure of SE data
SE data divides into a few regions with very different properties and FFTs are good way to explore such data spaces.

In summary, the contributions of this paper are:

- A novel inter-disciplinary contribution of the application of psychological science to software analytics.
- A cautionary tale that, for software analytics, *more* complex learners can perform *worse*.
- A warning that many current results in software analytics make the, possibly unwarranted, assumption that merely because an attribute is observable, that we should use those attributes in a model.
- Three tests for "actionable analytics": (a) Does a data mining produce succinct models? (b) Do those succinct models perform as well, or better, than more complex methods? (c) If the data mining algorithm is restricted to just the few attributes that developers actually change, does the resulting model perform satisfactorily?
- A demonstration that the restraints demanding by actionable analytics (very simple models, access to less data) need not result in models with poor performance.
- A new, very simple baseline data mining method (FFTs) against which more complex methods can be compared.
- A reproduction package containing all the data and algorithms of this paper, see http://url_blinded_for_review.

The rest of this paper is structured as follows. In Section 2, we introduce the concepts of "operational" and "comprehensible" as the preliminaries. Our data, experimentation settings and evaluation measures will be described in Section 3. In Section 4, we show our results and answer to research questions. Threats and validity of our work is given in Section 5. In Section 6, we conclude this paper with the following:

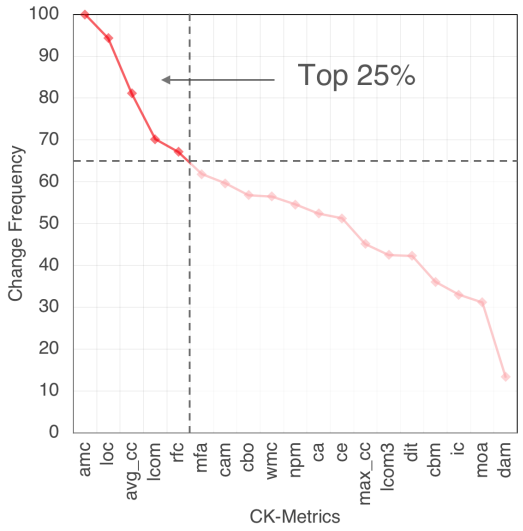


Figure 1: Only some metrics change between versions i and $i+1$ of a software system. For definitions of the metrics on the x-axis, see Table 1. To create this plot, we studied the 26 versions of the ten datasets in Table 3. First we initialize $total = 0$, then for all pairs of versions $i, i+1$ from the same data set, we (a) incremented $total$ by one; (b) collected the distributions of metric m seen in version i and $i+1$ of the software; (c) checked if those two distributions were different; and if so, (d) added one to $changed_m$. Afterwards, the y-axis of this plot was computed using $100 * changed_m/total$.

- There is much the software analytics community could learn from psychological science.
- Proponents of complex methods should always baseline those methods against simpler alternatives.

Finally, we discuss future work.

2 PRELIMINARIES

2.1 Operational

This paper assumes that for a data mining algorithm to be operational, it must generate effective models even if restricted to using just those attributes which, in practice, developers actually change. We have two reasons for making that assumption.

Firstly, this definition of operational can make a model much more acceptable to developers. If a model says that, say, $x > 0.6$ leads to defective code then developers will ask for guidance on how to reduce "x" (in order to reduce the chances of defects). If we define "operational" as per this article, then it is very simple matter to offer that developer numerous examples, from their own project's historical log, of how "x" was changed.

Secondly, as shown in Figure 1 there exist attributes that are usually not changed from one version to the next. Figure 1 is important since, as shown in our **RQ2** results, when we restrict model construction to just the 25% most frequently changed attributes, this can dramatically change the behavior of some data mining algorithms (but not FFTs).

Technical aside: in Figure 1, we defined "changed" using the A12 test [67] which declares two distributions different if they differ by more than a small effect. A recent ICSE'11 article [5] endorsed the use of A12 due to its non-parametric nature, it avoids any possibly incorrect Gaussian assumptions about the data.

2.2 Comprehensible

Why Demand Comprehensibility? This paper assumes that better data mining algorithms are better at explaining their models to humans. But is that always the case?

The obvious counter-argument is that if no human ever needs to understand our audited model, then it does not need to be comprehensible. For example, a neural net could control the carburetor of an internal combustion engine since that carburetor will never dispute the model or ask for clarification of any of its reasoning.

On the other hand, if a model is to be used to persuade software engineers to change what they are doing, it needs to be *comprehensible* so humans can debate the merits of its conclusions. Several researchers demand that software analytics models needs to be expressed in a simple way that is easy for software practitioners to interpret [16, 39, 45]. According to Kim et al. [32], software analytics aim to obtain actionable insights from software artifacts that help practitioners accomplish tasks related to software development, systems, and users. Other researchers [64] argue that for software vendors, managers, developers and users, such comprehensible insights are the core deliverable of software analytics. Sawyer et al. comments that actionable insight is the key driver for businesses to invest in data analytics initiatives [62]. Accordingly, much research focuses on the generation of simple models, or make blackbox models more explainable, so that human engineers can understand and appropriately trust the decisions made by software analytics models [1, 19].

If a model is not comprehensible, there are some explanation algorithms that might mitigate that problem. For example:

- In *secondary learning*, the examples given to a neural network are used to train a rule-based learner and those learners could be said to "explain" the neural net [13].
- In *contrast set learning* for instance-based reasoning, data is clustered and users are shown the difference between a few exemplars selected from each cluster [35].

Such explanation facilities are post-processors to the original learning method. An alternative simpler approach would be to use learners that generate comprehensible models in the first place.

The next section of this paper discusses one such alternate approach for creating simple comprehensible models.

Theories of Expert Comprehension. Psychological science argues that models comprising small rules are more comprehensible. This section outlines that argument.

Larkin et al. [36] characterize human expertise in terms of very small short term memory, or STM (used as a temporary scratch pad for current observation) and a very large long term memory, or LTM. The LTM holds separate tiny rule fragments that explore the contents of STM to say "when you see THIS, do THAT". When an LTM rule triggers, its consequence can rewrite STM contents which, in turn, can trigger other rules.

Short term memory is very small, perhaps even as small as four to seven items [12, 52] [2]. Experts are experts, says Larkin et al. [36] because the patterns in their LTM patterns dictate what to do, without needing to pause for reflection. Novices perform worse than experts, says Larkin et al., when they fill up their STM with too many to-do's where they plan to pause and reflect on what to do next. Since, experts post far fewer to-do's in their STMs, they complete their tasks faster because (a) they are less encumbered by excessive reflection and (b) there is more space in their STM to reason about new information. While first proposed in 1981, this STM/LTM theory still remains relevant [40]. This theory can be used to explain both expert competency and incompetency in software engineering tasks such as understanding code [69].

Phillips et al. [57] discuss how models containing tiny rule fragments can be quickly comprehended by doctors in emergency rooms making rapid decisions; or by soldiers on guard making snap decisions about whether to fire or not on a potential enemy; or by stockbrokers making instant decisions about buying or selling stock. That is, according to this psychological science theory [9, 14, 22–24, 42, 43, 54, 57], humans best understand a model:

- When they can "fit" it into their LTM; i.e., when that model comprises many small rule fragments;
- Further, to have an expert-level comprehension of some domain meaning having rules that can very quickly lead to decisions, without clogging up memory.

Psychological scientists have developed FFTs as one way to generate comprehensible models consisting of separate tiny rules [22, 42, 57]. A FFT is a decision tree with exactly two branches extending from each node, where either one or both branches is an exit branch leading to a leaf [42]. That is to say, in an FFT, every question posed by a node will trigger an immediate decision (so humans can read every leaf node as a separate rule).

For example, Table 2 (at left) is an FFT generated from the log4j JAVA system of Table 3. The goal of this tree is to classify a software module as "defective=true" or "defective=false". The four nodes in this FFT reference four static code attributes *cbo, rfc, dam, amc* (these metrics are defined in Table 1).

FFTs are a binary classification algorithm. To apply such classifiers to mulit-classes problems: (a) build one FFTs for each class for classX or not classX; (b) run all FFTs on the test example, then (c) then select conclusion with most support (number of rows).

An FFT of depth d has a choice of two "exit policies" at each level: the existing branch can select for the negation of the target (denoted "0") or the target (denoted "1"). The left-hand-side log4j tree in Table 2 is hence an 01110 tree since:

- The first level exits to the negation of the target: hence, "0".
- While the next tree levels exit first to target; hence, "111".
- And the final line of the model exits to the opposite of the penultimate line; hence, the final "0".

To build one FFT tree, select a maximum depth d, then follow the steps described in Table 4

For trees of depth $d = 4$, there are $2^4 = 16$ possible trees which we denoted 00001, 00010, 00101,... , 11110. Here, the first four digits

[2] Recently, Ma et al. [40] used evidence from neuroscience and functional MRIs to argue that STM capacity might be better measured using other factors than "number of items". But even they conceded that "the concept of a limited (STM) has considerable explanatory power for behavioral data".

Table 1: The C-K OO metrics studied in Figure 1. Note that the last line. *'defect'*, denotes the dependent variable.

Metric	Name	Description
amc	average method complexity	Number of JAVA byte codes
avg_cc	average McCabe	Average McCabe's cyclomatic complexity seen in class
ca	afferent couplings	How many other classes use the specific class.
cam	cohesion amongst classes	Summation of number of different types of method parameters in every method divided by a multiplication of number of different method parameter types in whole class and number of methods.
cbm	coupling between methods	Total number of new/redefined methods to which all the inherited methods are coupled
cbo	coupling between objects	Increased when the methods of one class access services of another.
ce	efferent couplings	How many other classes is used by the specific class.
dam	data access	Ratio of private (protected) attributes to total attributes
dit	depth of inheritance tree	It's defined as the maximum length from the node to the root of the tree
ic	inheritance coupling	Number of parent classes to which a given class is coupled (includes counts of methods and variables inherited)
lcom	lack of cohesion in methods	Number of pairs of methods that do not share a reference to an instance variable.
locm3	another lack of cohesion measure	If m, a are the number of $methods$, $attributes$ in a class number and $\mu(a)$ is the number of methods accessing an attribute, then $lcom3 = ((\frac{1}{a}\sum_j^a \mu(a_j)) - m)/(1-m)$.
loc	lines of code	Total lines of code in this file or package.
max_cc	Maximum McCabe	maximum McCabe's cyclomatic complexity seen in class
mfa	functional abstraction	Number of methods inherited by a class plus number of methods accessible by member methods of the class
moa	aggregation	Count of the number of data declarations (class fields) whose types are user defined classes
noc	number of children	Number of direct descendants (subclasses) for each class
npm	number of public methods	npm metric simply counts all the methods in a class that are declared as public.
rfc	response for a class	Number of methods invoked in response to a message to the object.
wmc	weighted methods per class	A class with more member functions than its peers is considered to be more complex and therefore more error prone
defect	defect	Boolean: where defects found in post-release bug-tracking systems.

Table 2: Three example FFTs.

```
if      cob <= 4    then false    # 0
else if rfc > 32    then true     # 1
else if dam >  0    then true     # 1
else if amc < 32.25 then true     # 1
else false                        # 0
```

```
if      cbo    <   4   then true # 1
else if max_cc <   3   then true # 1
else if wmc    <  10   then true # 1
else if rfc    <= 41.5 then true # 1
else false                       # 0
```

```
if      dam > 0 then false # 0
else if noc > 0 then false # 0
else if wmc > 5 then false # 0
else if moa > 0 then false # 0
else true                  # 1
```

denote the 16 exit policies and the last digit denotes the last line of the model (which makes the opposite conclusion to the line above). For example:

- A "00001" tree does it all it can to avoid the target class. Only after clearing away all the non-defective examples it can at levels one, two, three, four does it make a final "true" conclusion. Table 2 (right) shows the log4j 00001 tree. Note that all the exits, except the last, are to "false".
- As to "11110" trees, these fixate on finding the target. Table 2 (center) shows the log4j 11110 tree. Note that all the exits, except the last, are to "true".

During FFT training, we generate all 2^d trees then, using the predicate *score*, select the best one (using the training data). This single best tree is then applied to the test data.

Table 3: Some open-source JAVA systems. Used for training and testing showing different details for each. All data available on-line at http://tiny.cc/seacraft.

	Training		Testing		
Data Set	Versions	Cases	Versions	Cases	% Defective
jedit	3.2, 4.0, 4.1, 4.2	1257	4.3	492	2
ivy	1.1, 1.4	352	2.0	352	11
camel	1.0, 1.2, 1.4	1819	1.6	965	19
synapse	1.0, 1.1	379	1.2	256	34
velocity	1.4, 1.5	410	1.6	229	34
lucene	2.0, 2.2	442	2.4	340	59
poi	1.5, 2, 2.5	936	3.0	442	64
xerces	1.0, 1.2, 1.3	1055	1.4	588	74
log4j	1.0, 1.1	244	1.2	205	92
xalan	2.4, 2.5, 2.6	2411	2.7	909	99

Following the advice of [57], for all the experiments of this paper, we use a depth $d = 4$. Note that FFTs of such small depths are very succinct (see above examples). Many other data mining algorithms used in software analytics are far less succinct and far less comprehensible (see Table 5).

The value of models such as FFTs comprising many small rules has been extensively studied:

- These models use very few attributes from the data. Hence they tend to be robust against overfitting, especially on small and noisy data, and have been found to predict data at levels comparable with regression. See for example [14, 42, 71].

Table 4: Steps for building FFTs

(1) First discretize all attributes; e.g., split numerics on median value.
(2) For each discretized range, find what rows it selects in the training data. Using those rows, score each range using some user-supplied *score* function e.g., recall, false alarm, or the P_{opt} defined in §3.4.
(3) Divide the data on the best range.
(4) If the exit policy at this level is (0,1), then exit to (false,true) using the range that scores highest assuming that the target class is (false,true), respectively.
(5) If the current level is at d, add one last exit node predicting the opposite to step 4. Then terminate.
(6) Else, take the data selected by the non-exit range and go to step1 to build the next level of the tree.

Table 5: Comprehension issues with models generated by data mining algorithms used in software analytics.

For very high dimensional data, there is some evidence that complex deep learning algorithms have advantages for software engineering applications [25, 68, 72]. However, since they do not readily support explainability, they have been criticizing as "data mining alchemy" [63].
Support vector machines and principle component methods achieve their results after synthesizing new dimensions which are totally unfamiliar to human users [50].
Other methods that are heavily based on mathematics can be hard to explain to most users. For example, in our experience, it is hard for (e.g.,) users to determine minimal changes to a project that mostly affect defect-proneness, just by browsing the internal frequency tables of a Naive Bayes classifier or the coefficients found via linear regression/logistic regression [50].
When decision tree learners are many pages long, they are hard to browse and understand [18].
Random forests are even harder to understand than decision trees since the problems of reading one tree are multiplied N times, one for each member of the forest [38].
Instance-based methods do not compress their training data; instead they produce conclusions by finding older exemplars closest to the new example. Hence, for such instance-based methods, it is hard to generalize and make a conclusion about what kind of future projects might be (e.g.,) most defective-prone [4].

- Other work has shown that these rule-based models can perform comparably well to more complex models in a range of domains e.g., public health, medical risk management, performance science, etc. [28, 37, 59].
- Neth and Gigerenzer argue that such rule-bases are tools that work well under conditions of uncertainty [54].
- Brighton showed that rule-based models can perform better than complex nonlinear algorithms such as neural networks, exemplar models, and classification/regression trees [9].

3 METHODS

The use of models comprising many small rules has not been explored in the software analytics literature. This section describes the methods used by this paper to assess FFTs.

3.1 Data

3.1.1 Defect Data: To assess the FFTs, we perform our experiments using the publicly available SEACRAFT data [30], gathered by Jureczko et al. for object-oriented JAVA systems [30]. The "Jureczko" data records the number of known defects for each class using a post-release defect tracking system. The classes are described in terms of nearly two dozen metrics such as number of children (noc), lines of code (loc), etc (see Table 1). For details on the Jureczko data, see Table 3. The nature of collected data and its relevance to defect prediction is discussed in greater detail by Madeyski & Jureczko [41].

Table 6: Metrics used in issue lifetimes data

Commit	Comment	Issue
nCommitsByActorsT	meanCommentSizeT	issueCleanedBodyLen
nCommitsByCreator	nComments	nIssuesByCreator
nCommitsByUniqueActorsT		nIssuesByCreatorClosed
nCommitsInProject		nIssuesCreatedInProject
nCommitsProjectT		nIssuesCreatedInProjectClosed
		nIssuesCreatedProjectClosedT
		nIssuesCreatedProjectT
Misc.	nActors, nLabels, nSubscribedByT	

We selected these data sets since they have at least three consecutive releases (where release $i + 1$ was built after release i). This is important for our experimental rig (see section 3.2).

3.1.2 Issue Lifetime Data: This paper will conclude that FFTs are remarkable effective. To check the external validity of that conclusion, we will apply FFT to another SE domain [60, 61]. Our Github issue lifetime data[3] consists of 8 projects used to study issue lifetimes. In raw form, the data consisted of sets of JSON files for each repository, each file contained one type of data regarding the software repository (issues, commits, code contributors, changes to specific files as shown in Table 6). In order to extract data specific to issue lifetime, we did similar preprocessing and feature extraction on the raw datasets as suggested by [61].

3.2 Experimental Rig

For the defect prediction data, we use versions i, j, k of the software systems in Table 3.

Using versions i, j, we track what attributes change by from version i to j (using the calculation shown in Figure 1). Then we build a model using *all* the attributes from version j or just the top 25% most changed attributes. Note that this implements our definition of "operational", as discussed in our introduction.

After building a model, we use the latest version k for testing while the older versions for training. In this way, we can assert that all our predictions are using past date to predict the future.

For the issue lifetime data, we do not have access to multiple versions of the data. Hence, for this data we cannot perform the operational test. Hence, for that data we conduct a 5*10 cross-validation experiment that ensures that the train and test sets are different. For that cross-val, we divide the data into ten bins, then for each bin b_i we train on $data - b_i$ then test on bin b_i. To control for order effects (where the conclusions are altered by the order of the input examples) [2], this process is repeated five times, using different random orderings of the data.

3.3 Data Mining Algorithms

The results shown below compare FFTs to state of the art algorithms from software analytics. For a list of state-of-algorithms, we used the ICSE'15 paper from Ghotra et al. [21] which compared 32 classifiers for defect prediction. Their statistical analysis showed that the performance of these classifiers clustered into four groups shown in Table 7. For our work, we selected one classifier at random from each of their clusters: i.e., Simple Logistic (SL), Naive Bayes (NB),

[3] https://doi.org/10.5281/zenodo.197111

Table 7: For the purposes of predicting software defects, Ghotra et al. [21] found that many learners have similar performance. Here are their four clusters of 32 data mining algorithms. For our work, we selected learners at random, one from each cluster (see the underlined entries).

Overall Rank	Classification Technique	Median Rank	Average Rank	Standard Deviation
1	Rsub+J48, **SL**, Rsub+SL, Bag+SL, LMT, RF+SL, RF+J48, Bag+LMT, Rsub+LMT, and RF+LMT	1.7	1.63	0.33
2	RBFs, Bag+J48, Ad+SL, KNN, RF+NB, Ad+LMT, **NB**, Rsub+NB, and Bag+NB	2.8	2.84	0.41
3	Ripper, **EM**, J48, Ad+NB, Bag+SMO, Ad+J48, Ad+SMO, and K-means	5.1	5.13	0.46
4	RF+SMO, Ridor, **SMO**, and Rsub+SMO	6.5	6.45	0.25

Expectation Maximization (EM), Sequential Minimal Optimization (SMO).

Simple Logistic and Naive Bayes falls into the 1st and 2nd rankings layers. They are both statistical techniques that are based on a probability based model [34]. These techniques are used to find patterns in datasets and build diverse predictive models [7]. Simple Logistic is a generalized linear regression model that uses a logit function. Naive Bayes is a probability-based technique that assumes that all of the predictors are independent of each other.

Clustering techniques like EM divide the training data into small groups such that the similarity within groups is more than across the groups [26]. EM is a clustering technique based on cluster performance Expectation Maximization [17] (EM) technique, which automatically splits a dataset into an (approximately) optimal number of clusters [8].

Support Vector Machines (SVMs) use a hyperplane to separate two classes (i.e., defective or not). In this paper, following the results of Ghotra et al., we use the Sequential Minimal Optimization (SMO) SVM technique. SMO analytically solves the large Quadratic Programming (QP) optimization problem which occurs in SVM training by dividing the problem into a series of possible QP problems [75].

3.4 Evaluation Measures

Our rig assess learned models using an evaluation function called *score*. For FFTs, this function is called three times:

- Once to rank discretized ranges;
- Then once again to select the best FFT out of the 2^d trees generated during training.
- Then finally, *score* is used to score what happens when that best FFT is applied to the test data.

For all the other learners, *score* is applied on the test data. For this work, we use the two *score* measures: *dis2heaven* and P_{opt}.

Ideally, a perfect learner will have perfect recall (100%) with no false alarms.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \quad (1)$$

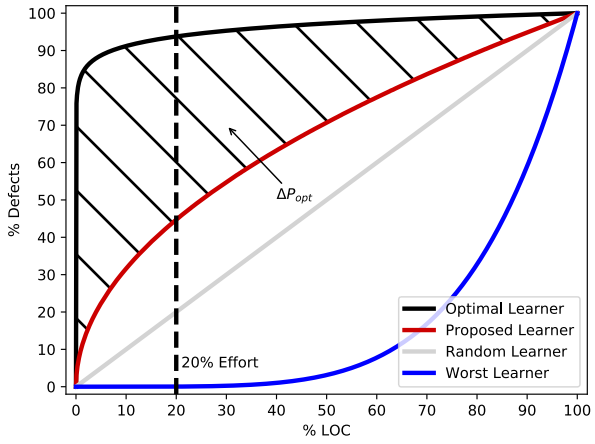


Figure 2: Effort-based cumulative lift chart [73].

$$FAR = \frac{FalsePositive}{FalsePositive + TrueNegative} \quad (2)$$

We combine these two into a "distance to heaven" measure called *dis2heaven* that reports how far a learner falls away from the ideal point of *Recall=1* and *FAR=0*:

$$score_1 = dis2heaven = \sqrt{\frac{(1 - Recall)^2 + FAR^2}{2}} \quad (3)$$

As to P_{opt}, Ostrand et al. [56] report that their quality predictors can find 20% of the files contain on average 80% of all defects in the project. Although there is nothing magical about the number 20%, it has been used as a cutoff value to set the efforts required for the defect inspection when evaluating the defect learners [31, 44, 53, 73]. That is, P_{opt} reports how many defects have been found after (a) the code is sorted by the learner from "most likely to be buggy" to "least likely"; then (b) humans inspect 20% of the code (measured in lines of code), where that code has , how many defects can be detected by the learner. This measure is widely used in defect prediction literature [31, 48, 49, 53, 73, 76].

P_{opt} is defined as $1 - \Delta_{opt}$, where Δ_{opt} is the area between the effort cumulative lift charts of the optimal model and the prediction model (as shown in Figure2). In this chart, the x-axis is the percentage of required effort to inspect the code and the y-axis is the percentage of defects found in the selected code. In the optimal model, all the changes are sorted by the actual defect density in descending order, while for the predicted model, all the changes are sorted by the actual predicted value in descending order. According to Kamei et al. and Xu et al. [31, 53, 73] P_{opt} can be normalized as follows:

$$score_2 = P_{opt}(m) = 1 - \frac{S(optimal) - S(m)}{S(optimal) - S(worst)} \quad (4)$$

where $S(optimal)$, $S(m)$ and $S(worst)$ represent the area of curve under the optimal model, predicted model, and worst model, respectively. This worst model is built by sorting all the changes according to the actual defect density in ascending order.

Note that for our two *score* functions:

- For *dis2heaven*, the *lower* values are *better*.
- For P_{opt}, the *higher* values are *better*.

4 RESULTS

4.1 RQ1: Do FFTs models perform worse than the current state-of-the-art?

Figure 3 compares the performance of FFT versus learners taken from Ghotra et al. In this figure, datasets are sorted left right based on the FFT performance scores. With very few exceptions:

- FFT's *dis2heaven*'s results *lower*, hence *better*, than the other learners.
- FFT's P_{opt} results are much *higher*, hence *better*, than the other learners.

Therefore our answer to **RQ1** is:

> **For defect prediction, FFTs out-perform the state-of-art.**
> When compared to state-of-the-art defect prediction algorithms surveyed by Ghotra et al., FFTs are more effective (where "effective" is measured in terms of a recall/false alarm metric or P_{opt}).

4.2 RQ2: Are FFTs more operational than the current state-of-the-art?

Please recollect from before that a model is operational if its performance is not affected after *avoiding* attributes that are rarely changed by developers.

Figure 4 compares model performance when we learn from all 100% attributes or just the 25% most changed attributes. For this study, these 25% group (of most changed attributes) was computed separately for each data set. Note that:

- The top row of Figure 4 shows the *dis2heaven* results;
- The bottom row of Figure 4 shows the P_{opt} results.

Figure 5 reports the deltas in performance scores seen between using 25% and 100% of the data. These deltas are computed such that *larger* values are *better*; i.e., for (*dist2heaven*, P_{opt}) we report (25%-100%, 100%-25%) since (fewer, more) values are better (respectively).

There are several key features for these results:

- The FFT's red dots for *dis2heaven* are *below* the rest; also, FFT's orange dots for P_{opt} are *above* the rest. This means that, regardless of whether we use all attributes or just the most changed attributes, the FFT results are nearly always better than the other methods.
- As seen in Figure 5, the deltas between using all data and just some of the data is smallest for FFTs and EM (the instance-based clustering algorithm). In P_{opt}, those deltas are very small indeed (the FFT and EM results lie right on the y-axis for most of that plot).
- Also, see in Figure 5, the deltas on the other learners can be highly variable. While for the most part, using just the 25% most changed attributes improves performance, SMO , SL and NB all have large negative results for at least some of the data sets.

In summary, the learners studied here fall into three groups:

Table 8: Frequency heatmap of best exit polices seen for FFT and defect prediction.

Best FFF exit policy		25%		100%	
		D2H	P_{opt}	D2H	P_{opt}
00001	0	0	0	0	0
00010	0	0	0	0	0
00101	0	0	0	0	0
00110	0	0	0	0	0
01001	0	0	0	0	0
01010	0	0	0	0	0
01101	1	0	0	0	1
01110	0	0	0	0	0
10001	14	6	0	7	1
10010	8	4	2	2	0
10101	3	0	1	1	1
10110	5	0	3	0	2
11001	0	0	0	0	0
11010	3	0	1	0	2
11101	2	0	0	0	2
11110	4	0	3	0	1
Totals	40	10	10	10	10

(1) Those that exhibited a wide performance variance after restricting the learning to just the frequently changed data (SL, NB, SMO), and those that are not (FFT, EM);
(2) Those with best performance across the two performance measures studied here (FFT), and the rest (SL, NB, EM, SMO);
(3) Those that generate tiny models (FFT), and the rest (SL, NB, EM, SMO).

Accordingly, FFT is the recommended learner since it both performs well and is unaffected by issues such as whether or not the data is restricted to just the most operational attributes. In summary:

> **When learning from less data, FFTs performance is stabler than some other learners.**
> When data is restricted to attributes that developers often change, then FFTs performance is only slightly changed while the performance of some other learners, can vary by alarmingly large amounts.

4.3 RQ3: Why do FFTs work so well?

To explain the success of FFTs, recall that during training, FFTs explores 2^d models, then selects the models whose exit policies achieves best performances (exit policies were introduced in Section 2.2). The exit policies selected by FFTs are like a trace of the reasoning jumping around the data. For example, a 11110 policy shows a model always jumping towards sections of the data containing most defects. Also, a 00001 policy show another model trying to jump away from defects until, in its last step, it does one final

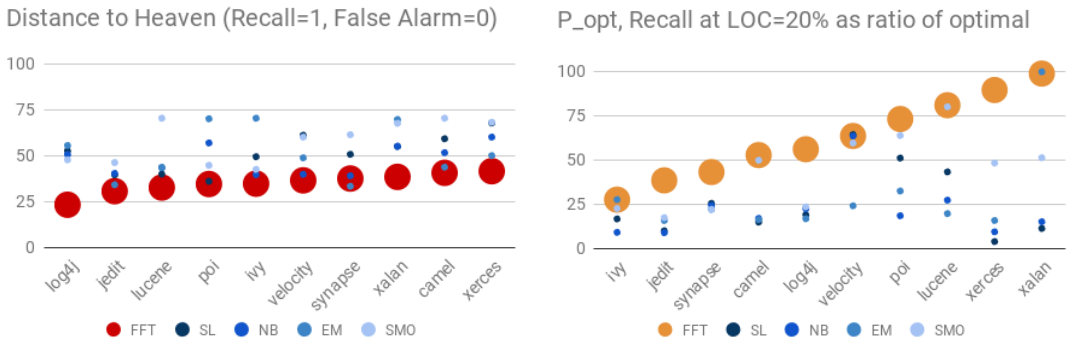


Figure 3: On the left, in the *dis2Heaven* results, *less* is *better*. On the right, in the P_{opt} results, *more* is *better*. On both sides, the FFTs results are better than those from state-of-the-art defect prediction algorithms (as defined by Ghotra et al. [21]).

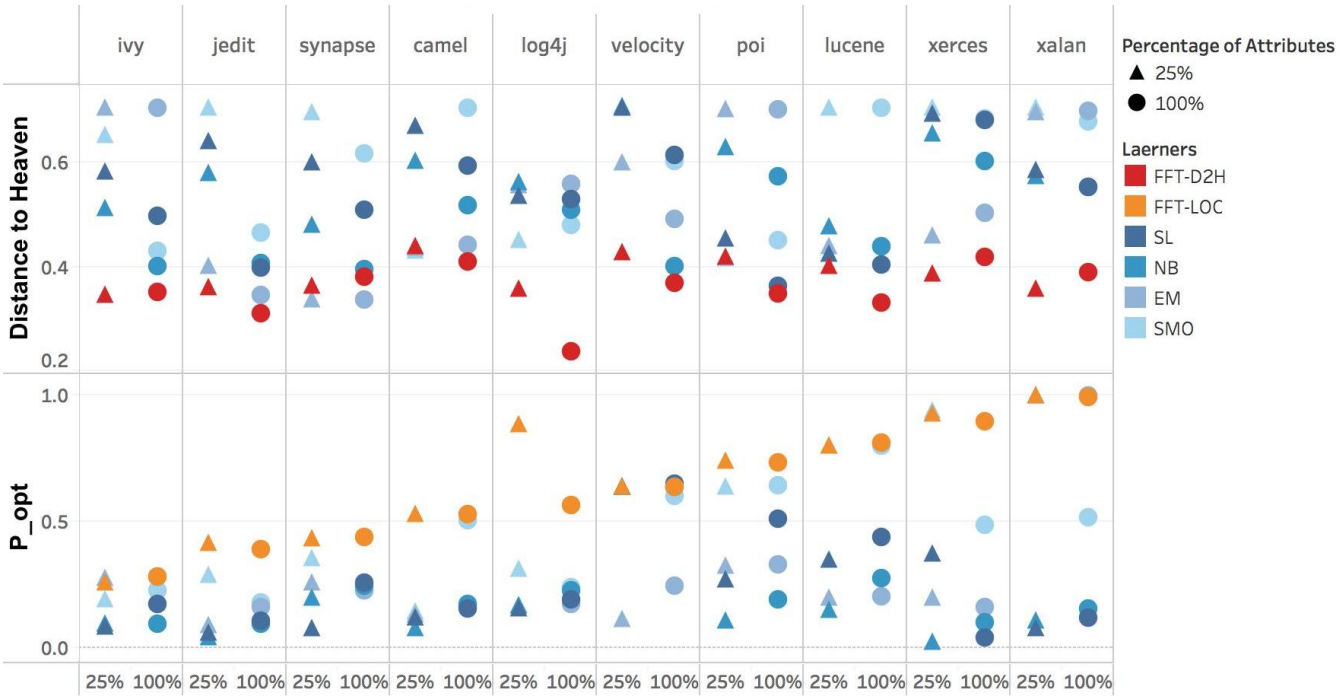


Figure 4: For each learner in Figure 3, this plot shows the difference between the results obtains using the top 25% or all (100%) of attributes. For (*dist2heaven*,P_{opt}), values that are (*lesser*,*greater*) (respectively) are *better*. Note that all the ● 100% results were also shown in Figure 3.

jump towards defects. Table 8 shows what exit policies were seen in the experiments of the last section:

- The 11110 policy was used sometimes.
- A more common policy is 10001 which shows a tree first jumping to some low hanging fruit (see the first "1"), then jumping away from defects three times (see the next "000") before a final jump into defects (see the last "1").
- That said, while 10001 was most common, many other exit policies appear in Table 8. For example, the P_{opt} policies are particularly diverse.

Table 8 suggests that software data is "lumpy"; i.e., it divides into a few separate regions, each with different properties. Further, the number and importance of the "lumps" is specific to the data set and the goal criteria. In such a "lumpy" space, a learning policy like FFT works well since its exit policies let a learner discover how to best jump between the "lumps". Other learners fail in this coarse-grained lumpy space when they:

- Divide the data too much; e.g. like RandomForests, which finely divide the data multiple times down the branches of the trees and across multiple trees;

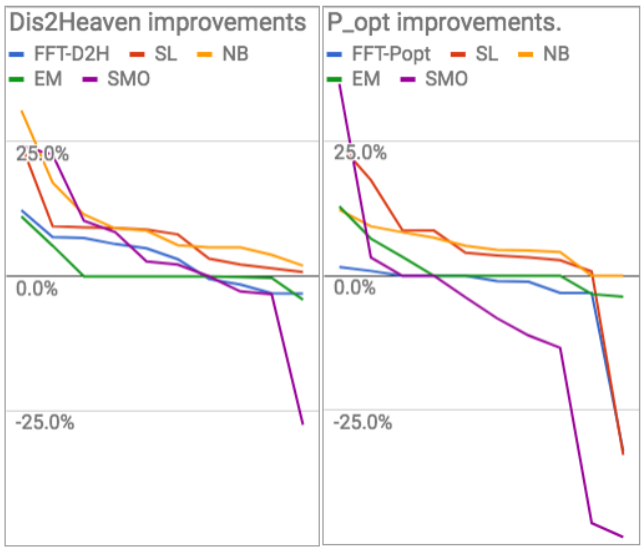


Figure 5: Deltas between results 25% and 100% of the data. Computed from Figure 4. Calculated such that *larger* values are *better*; i.e., for (*dist2heaven*, P_{opt}) we report (25%-100%, 100%-25%) since (less, more) values are better (respectively). All values for each learner are sorted independently.

- Fit some general model across all the different parts of the data; e.g. like simple logistic regression.

In summary, in answer to the question "why do FFTs work so well", we reply:

> **FFTs match the structure of SE data**
> SE data divides into a few regions with very different properties and FFTs are good way to explore such data spaces.

5 THREATS TO VALIDITY

5.1 Sampling Bias

This paper shares the same sampling bias problem as every other data mining paper. Sampling bias threatens any classification experiment; what matters in one case may or may not hold in another case. For example, even though we use 10 open-source datasets in this study which come from several sources, they were all supplied by individuals.

As researchers, we can adopt two tactics to reduce the sampling bias problem. First we can document our tools and methods, then post an executable reproduction package for all the experiments (that package for this paper is available at url_blind_for_review).

Secondly, when new data becomes available, we can test our methods on the new data. For example, Table 9 shows results were FFTs and four different state-of-the-art learners, i.e. Decision Tree, Random Forest, Logistic Regression, K-Nearest Neighbors, were applied to the task of predicting issue close time (the other four learners were used since that was the technology recommended in a recent study in that domain [60, 61]). Unlike the defect prediction data, we did not have multiple versions of the code so, for this domain, we used a 5*10-way cross-validation analysis. White cells show where the FFT results were statistically different and better than all of the state-of-the-art learners' results. Note that, in most cases (43/56 = 77%), FFTs performed better.

Table 9: Which learners performed better (in terms of median *Dis2heaven*) in 5*10 cross-value experiments predicting for different classes of "how long to close an Github issue". Gray areas denote experiments where FFTs were outperformed by other learners. Note that, in (43/56=77%) experiments, FFT performed better than the prior state-of-the-art in this area [60].

	Days till closed						
Data(# of instances)	> 365	< 180	< 90	< 30	< 14	< 7	< 1
cloudstack (1551)	FFT	FFT	FFT	FFT	FFT	DT	LR
node (6207)	FFT	FFT	FFT	FFT	FFT	DT	LR
deeplearning (1434)	FFT	FFT	FFT	FFT	FFT	FFT	RF
cocoon (2045)	FFT	FFT	FFT	FFT	FFT	FFT	FFT
ofbiz (6177)	FFT	FFT	FFT	FFT	FFT	FFT	FFT
camel (5056)	RF/KNN	KNN	FFT/KNN/DT	FFT	FFT	FFT	FFT
hadoop (12191)	KNN	DT	*DT*	FFT	FFT	FFT	FFT
qpid (5475)	DT	DT/RF	DT	FFT	FFT	FFT	FFT

The goal here is to classify an issue according to how long it will take to close; i.e. less than 1 day, less than 7 days, and so on. Values collected via a 5x10 cross-validation procedure.
Cells with a (white, gray) background means FFTs are statistically (better, worse) than (all, any) of the state-of-the-art learners (as determined by a Mann-Whitney test, 95% confidence), respectively. KNN, DT, RF and LR represents K-Nearest Neighbors, Decision Tree, Random Forest and Logistic Regression respectively.

While this result does not prove that FFTs works well in all domains, it does show that there exists more than one domain where this is a useful approach.

5.2 Learner Bias

For building the defect predictors in this study, we elected to use Simple Logistic, Naive Bayes, Expectation Maximization, Support Vector Machine. We chose these learners because past studies shows that, for defect prediction tasks, these four learners represents four different levels of performance among a bunch of different learners [3, 21]. Thus they are selected as the state-of-the-art learns to be compared with FFTs on the defect prediction data. While for Table 9), K-Nearest Neighbors, Decision Tree, Random Forest and Logistic Regression are used to compare against FFTs, because a recent work has summarized all the best learners that were applied on the issue lifetime data.

5.3 Evaluation Bias

This paper uses two performance measures, i.e., P_{opt} and *dist2heaven* as defined in Equation 4 and 3. Other quality measures often used in software engineering to quantify the effectiveness of prediction [29, 47, 51]. A comprehensive analysis using these measures may be performed with our replication package. Additionally, other measures can easily be added to extend this replication package.

5.4 Order Bias

For the performance evaluation part, the order that the data trained and predicted affects the results.

For the defect prediction datasets, we deliberately choose an ordering that mimics how our software projects releases versions

so, for those experiments, we would say that bias was a required and needed.

For the issue close time results of Table 9, to mitigate this order bias, we ran our rig in a the 5-bin cross validation 10 times, randomly changing the order of the data each time.

6 CONCLUSIONS

This paper has shown that a data mining algorithm call Fast-and-Frugal trees (FFTs) developed by psychological scientist is remarkably effective for creating actionable software analytics. Here "actionable" was defined as a combination of comprehensible and operational.

Measured in terms of comprehensibility, the FFT examples of Table 2 show that FFTs satisfy requirements raised by psychological scientists for "easily understandable at an expert level"; i.e., they comprise several short rules and those rules can be quickly applied (recall that each level of an FFT has an exit point which, if used, means humans can ignore the rest of the tree).

Despite their brevity, FFTs are remarkably effective:

- Measured in terms of P_{opt}, FFTs are much better than other standard algorithms (see Figure 3).
- Measured in terms of distance to the "heaven" point of 100% recall and no false alarms, FFTs are either usually better than other standard algorithms used in software analytics (Random Forests, Naive Bayes, EM, Logistic Regression, and SVM). This result holds for at least two SE domains: defect prediction (see Figure 3) issue close time prediction (see Table 9).

As to being operational, we found that if learning is restricted to just the attributes changed most often, then the behavior of other learning algorithms can vary, wildly (see Figure 5). The behaviour of FFTs, on the other hand, remain remarkable stable across that treatment.

From the above, our conclusions is two-fold:

(1) There is much the software analytics community could learn from psychological science. FFTs, based on psychological science principles, out-perform a wide range of learners in widespread use.
(2) Proponents of complex methods should always baseline those methods against simpler alternatives. For example, FFTs could be used as a standard baseline learner against which other software analytics tools are compared.

7 FUTURE WORK

Numerous aspects of the above motivate deserve more attention.

7.1 More Data

This experiment with issue close time shows that FFTs are useful for more just defect prediction data. That said, for future work, it is important to test many other SE domains to learn when FFTs are useful. For example, at this time we are exploring text mining of StackOverflow data.

7.2 More Learners

The above experiments should be repeated, comparing FFTs against more learners. For example, at this time, we are comparing FFTs against deep learning for SE datasets. At this time, there is nothing as yet definitive to report about those results.

7.3 More Algorithm Design

These results may have implications beyond SE. Indeed, it might be insightful to another field– machine learning. For the reader familiar with machine learning literature, we note that FFTs are a decision-list rule-covering model. FFTs restrict the (a) number of conditions per rule to only one comparison and (b) the total number of rules is set to a small number (often often just $d \in \{3, 4, 5\}$). Other decision list approaches such as PRISM [10], INDUCT [70],RIPPER [11] and RIPPLE-DOWN-RULES [20] produce far more complex models since they impose no such restriction. Perhaps the lesson of FFT is that PRISM,INDUCT,RIPPER, etc could be simplified with a few simple restrictions on the models they learn.

Also the success of FFT might be credited to its use on ensemble methods; i.e. train multiple times, then select the best. The comparison between FFTs and other ensemble methods like bagging and boosting [58] could be useful in future work.

7.4 Applications to Delta Debugging

There is a potential connection between the Figure 5 results and the delta debugging results of Zeller [74]. As shown above, we found that, sometimes focusing on the values that change most can sometimes, lead to better defect predictors (though, caveat empty or, sometimes it can actually make matters worse– see the large negative results in Figure 5). Note that this parallels Zeller's approach which he summarizes as "Initially, variable v1 was x1, thus variable v2 became x2, thus variable v3 became x3 ... and thus the program failed". In future work, we will explore further applications of FFTs to delta debugging.

REFERENCES

[1] Behnoush Abdollahi and Olfa Nasraoui. 2016. Explainable restricted Boltzmann machines for collaborative filtering. *arXiv preprint arXiv:1606.07129* (2016).
[2] Amritanshu Agrawal, Wei Fu, and Tim Menzies. 2018. What is Wrong with Topic Modeling?(and How to Fix it Using Search-based Software Engineering). *Information and Software Technology* (2018).
[3] Amritanshu Agrawal and Tim Menzies. 2018. Is "Better Data" Better than "Better Data Miners"? (Benefits of Tuning SMOTE for Defect Prediction). *International Conference on Software Engineering* (2018).
[4] David W Aha, Dennis Kibler, and Marc K Albert. 1991. Instance-based learning algorithms. *Machine learning* 6, 1 (1991), 37–66.
[5] A. Arcuri and L. Briand. 2011. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *2011 33rd International Conference on Software Engineering (ICSE)*. 1–10. DOI: http://dx.doi.org/10.1145/1985793.1985795
[6] Andrew Begel and Thomas Zimmermann. 2014. Analyze this! 145 questions for data scientists in software engineering. In *Proceedings of the 36th International Conference on Software Engineering*. ACM, 12–23.
[7] Alex Berson, Stephen Smith, and Kurt Thearling. 2004. An overview of data mining techniques. *Building Data Mining Application for CRM* (2004).
[8] Nicolas Bettenburg, Meiyappan Nagappan, and Ahmed E Hassan. 2012. Think locally, act globally: Improving defect and effort prediction models. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*. IEEE Press, 60–69.
[9] Henry Brighton. 2006. Robust Inference with Simple Cognitive Models.. In *AAAI spring symposium: Between a rock and a hard place: Cognitive science principles meet AI-hard problems*. 17–22.
[10] Jadzia Cendrowska. 1987. PRISM: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies* 27, 4 (1987), 349–370.
[11] William W. Cohen. 1995. Fast Effective Rule Induction. In *ICML'95*. 115–123.
[12] N. Cowan. 2001. The magical number 4 in short-term memory: a reconsideration of mental storage capacity. *Behav Brain Sci* 24, 1 (Feb 2001), 87–114.

[13] Mark W Craven and Jude W Shavlik. 2014. Learning symbolic rules using artificial neural networks. In *Proceedings of the Tenth International Conference on Machine Learning*. 73–80.

[14] Jean Czerlinski, Gerd Gigerenzer, and Daniel G Goldstein. 1999. How good are simple heuristics? (1999).

[15] J. Czerwonka, R. Das, N. Nagappan, A. Tarvo, and A. Teterev. 2011. CRANE: Failure Prediction, Change Analysis and Test Prioritization in Practice – Experiences from Windows. In *Software Testing, Verification and Validation (ICST), 2011 IEEE Fourth International Conference on*. 357 –366.

[16] Hoa Khanh Dam, Truyen Tran, and Aditya Ghose. 2018. Explainable Software Analytics. *arXiv preprint arXiv:1802.00603* (2018).

[17] Chris Fraley and Adrian E Raftery. 2007. Bayesian regularization for normal mixture estimation and model-based clustering. *Journal of classification* 24, 2 (2007), 155–181.

[18] Mark A Friedl and Carla E Brodley. 1997. Decision tree classification of land cover from remotely sensed data. *Remote sensing of environment* 61, 3 (1997), 399–409.

[19] Wei Fu and Tim Menzies. 2017. Easy over hard: a case study on deep learning. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 49–60.

[20] B. R. Gaines and P. Compton. 1995. Induction of Ripple-down Rules Applied to Modeling Large Databases. *J. Intell. Inf. Syst.* 5, 3 (Nov. 1995), 211–228. DOI: http://dx.doi.org/10.1007/BF00962234

[21] Baljinder Ghotra, Shane McIntosh, and Ahmed E Hassan. 2015. Revisiting the impact of classification techniques on the performance of defect prediction models. In *Proceedings of the 37th International Conference on Software Engineering-Volume 1*. IEEE Press, 789–800.

[22] Gerd Gigerenzer. 2008. Why heuristics work. *Perspectives on psychological science* 3, 1 (2008), 20–29.

[23] Gerd Gigerenzer, Jean Czerlinski, and Laura Martignon. 1999. How good are fast and frugal heuristics. *Decision science and technology: Reflections on the contributions of Ward Edwards* (1999), 81–103.

[24] Gerd Gigerenzer and Wolfgang Gaissmaier. 2011. Heuristic decision making. *Annual review of psychology* 62 (2011), 451–482.

[25] Xiaodong Gu, Hongyu Zhang, Dongmei Zhang, and Sunghun Kim. 2016. Deep API learning. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 631–642.

[26] Khaled Hammouda and Fakhreddine Karray. 2000. A comparative study of data clustering techniques. *University of Waterloo, Ontario, Canada* (2000).

[27] J. Hihn and T. Menzies. 2015. Data Mining Methods and Cost Estimation Models: Why is it So Hard to Infuse New Ideas?. In *2015 30th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)*. 5–9. DOI: http://dx.doi.org/10.1109/ASEW.2015.27

[28] Mirjam A Jenny, Thorsten Pachur, S Lloyd Williams, Eni Becker, and Jürgen Margraf. 2013. Simple rules for detecting depression. *Journal of Applied Research in Memory and Cognition* 2, 3 (2013), 149–157.

[29] Magne Jorgensen. 2004. Realism in assessment of effort estimation uncertainty: It matters how you ask. *IEEE Transactions on Software Engineering* 30, 4 (2004), 209–217.

[30] Marian Jureczko and Lech Madeyski. 2010. Towards identifying software project clusters with regard to defect prediction. In *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*. ACM, 9.

[31] Yasutaka Kamei, Emad Shihab, Bram Adams, Ahmed E Hassan, Audris Mockus, Anand Sinha, and Naoyasu Ubayashi. 2013. A large-scale empirical study of just-in-time quality assurance. *IEEE Transactions on Software Engineering* 39, 6 (2013), 757–773.

[32] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. 2016. The Emerging Role of Data Scientists on Software Development Teams. In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. ACM, New York, NY, USA, 96–107. DOI: http://dx.doi.org/10.1145/2884781.2884783

[33] E. Kocaguneli, T. Menzies, A. Bener, and J. Keung. 2012. Exploiting the Essential Assumptions of Analogy-Based Effort Estimation. *IEEE Transactions on Software Engineering* 28 (2012), 425–438. Issue 2. Available from http://menzies.us/pdf/11teak.pdf.

[34] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. 2007. Supervised machine learning: A review of classification techniques. (2007).

[35] Rahul Krishna and Tim Menzies. 2015. Actionable= Cluster+ Contrast?. In *Automated Software Engineering Workshop (ASEW), 2015 30th IEEE/ACM International Conference on*. IEEE, 14–17.

[36] Jill Larkin, John McDermott, Dorothea P. Simon, and Herbert A. Simon. 1980. Expert and Novice Performance in Solving Physics Problems. *Science* 208, 4450 (1980), 1335–1342. DOI: http://dx.doi.org/10.1126/science.208.4450.1335 arXiv:http://science.sciencemag.org/content/208/4450/1335.full.pdf

[37] K Laskey and Laura Martignon. 2014. Comparing fast and frugal trees and Bayesian networks for risk assessment. In *Proceedings of the 9th International Conference on Teaching Statistics, Flagstaff, Arizona.*

[38] Andy Liaw, Matthew Wiener, and others. 2002. Classification and regression by randomForest. *R news* 2, 3 (2002), 18–22.

[39] Zachary C Lipton. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490* (2016).

[40] Wei Ji Ma, Masud Husain, and Paul M Bays. 2014. Changing concepts of working memory. *Nature neuroscience* 17, 3 (2014), 347–356.

[41] Lech Madeyski and Marian Jureczko. 2015. Which process metrics can significantly improve defect prediction models? An empirical study. *Software Quality Journal* 23, 3 (2015), 393–422.

[42] Laura Martignon, Konstantinos V Katsikopoulos, and Jan K Woike. 2008. Categorization with limited resources: A family of simple heuristics. *Journal of Mathematical Psychology* 52, 6 (2008), 352–361.

[43] Laura Martignon, Oliver Vitouch, Masanori Takezawa, and Malcolm R Forster. 2003. Naive and yet enlightened: From natural frequencies to fast and frugal decision trees. *Thinking: Psychological perspectives on reasoning, judgment and decision making* (2003), 189–211.

[44] Thilo Mende and Rainer Koschke. 2010. Effort-aware defect prediction models. In *Software Maintenance and Reengineering (CSMR), 2010 14th European Conference on*. IEEE, 107–116.

[45] Tim Menzies. 2014. OccamâĂŹs razor and simple software project management. In *Software Project Management in a Changing World*. Springer, 447–472.

[46] Tim Menzies, Alex Dekhtyar, Justin Distefano, and Jeremy Greenwald. 2007. Problems with Precision: A Response to "Comments on 'Data Mining Static Code Attributes to Learn Defect Predictors'". *IEEE Transactions on Software Engineering* 33, 9 (sep 2007), 637–640. DOI: http://dx.doi.org/10.1109/TSE.2007.70721

[47] Tim Menzies, Alex Dekhtyar, Justin Distefano, and Jeremy Greenwald. 2007. Problems with Precision: A Response to" comments on'data mining static code attributes to learn defect predictors'". *IEEE Transactions on Software Engineering* 33, 9 (2007), 637–640.

[48] Tim Menzies, Jeremy Greenwald, and Art Frank. 2007. Data mining static code attributes to learn defect predictors. *IEEE transactions on software engineering* 33, 1 (2007), 2–13.

[49] Tim Menzies, Zach Milton, Burak Turhan, Bojan Cukic, Yue Jiang, and Ayşe Bener. 2010. Defect prediction from static code features: current results, limitations, new approaches. *Automated Software Engineering* 17, 4 (2010), 375–407.

[50] Tim Menzies, Osamu Mizuno, Yasunari Takagi, and Tohru Kikuno. 2009. Explanation vs Performance in Data Mining: A Case Study with Predicting Runaway Projects. *Journal of Software Engineering and Applications* 2 (2009), 221–236.

[51] Tim Menzies, Dan Port, Zhihao Chen, and Jairus Hihn. 2005. Simple software cost analysis: safe or unsafe?. In *ACM SIGSOFT Software Engineering Notes*, Vol. 30. ACM, 1–6.

[52] George A Miller. 1956. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review* 63, 2 (1956), 81.

[53] Akito Monden, Takuma Hayashi, Shoji Shinoda, Kumiko Shirai, Junichi Yoshida, Mike Barker, and Kenichi Matsumoto. 2013. Assessing the cost effectiveness of fault prediction in acceptance testing. *IEEE Transactions on Software Engineering* 39, 10 (2013), 1345–1357.

[54] Hansjörg Neth and Gerd Gigerenzer. 2015. Heuristics: Tools for an uncertain world. *Emerging trends in the social and behavioral sciences: An interdisciplinary, searchable, and linkable resource* (2015).

[55] Thomas J. Ostrand, Elaine J. Weyuker, and Robert M. Bell. 2004. Where the bugs are. In *ISSTA '04: Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing and analysis*. ACM, New York, NY, USA, 86–96.

[56] Thomas J Ostrand, Elaine J Weyuker, and Robert M Bell. 2005. Predicting the location and number of faults in large software systems. *IEEE Transactions on Software Engineering* 31, 4 (2005), 340–355.

[57] Nathaniel D Phillips, Hansjoerg Neth, Jan K Woike, and Wolfgang Gaissmaier. 2017. FFTrees: A toolbox to create, visualize, and evaluate fast-and-frugal decision trees. *Judgment and Decision Making* 12, 4 (2017), 344–368.

[58] J Ross Quinlan and others. 1996. Bagging, boosting, and C4. 5. In *AAAI/IAAI, Vol. 1*. 725–730.

[59] Markus Raab and Gerd Gigerenzer. 2015. The power of simplicity: a fast-and-frugal heuristics approach to performance science. *Frontiers in psychology* 6 (2015).

[60] Tim Menzies Rahul Krishna. 2018. Bellwethers: A Baseline Method For Transfer Learning. *arXiv preprint arXiv:1703.06218v4* (2018).

[61] Mitch Rees-Jones, Matthew Martin, and Tim Menzies. 2018. Better predictors for issue lifetime. *Journal of Software and Systems, submitted. arXiv preprint arXiv:1702.07735* (2018).

[62] Robert Sawyer. 2013. BÍâĂŹs Impact on Analyses and Decision Making Depends on the Development of Less Complex Applications. In *Principles and Applications of Business Intelligence Research*. IGI Global, 83–95.

[63] AI Technology & Industry Review Synced. 2017. LeCun vs Rahimi: Has Machine Learning Become Alchemy? (2017). https://medium.com/@Synced/lecun-vs-rahimi-has-machine-learning-become-alchemy-21cb1557920d

[64] Shiang-Yen Tan and Taizan Chan. 2016. Defining and conceptualizing actionable insight: a conceptual framework for decision-centric analytics. *arXiv preprint arXiv:1606.03510* (2016).

[65] Christopher Theisen, Kim Herzig, Patrick Morrison, Brendan Murphy, and Laurie Williams. 2015. Approximating Attack Surfaces with Stack Traces. In *ICSE'15*.

[66] Burak Turhan, Ayşe Tosun, and Ayşe Bener. 2011. Empirical evaluation of mixed-project defect prediction models. In *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on*. IEEE, 396–403.

[67] András Vargha and Harold D Delaney. 2000. A critique and improvement of the CL common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics* 25, 2 (2000), 101–132.

[68] Martin White, Christopher Vendome, Mario Linares-Vásquez, and Denys Poshyvanyk. 2015. Toward deep learning software repositories. In *Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on*. IEEE, 334–345.

[69] Susan Wiedenbeck, Vikki Fix, and Jean Scholtz. 1993. Characteristics of the mental representations of novice and expert programmers: an empirical study. *International Journal of Man-Machine Studies* 39, 5 (1993), 793–812.

[70] Ian H. Witten and Eibe Frank. 2002. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. *SIGMOD Rec.* 31, 1 (March 2002), 76–77. DOI: http://dx.doi.org/10.1145/507338.507355

[71] Jan K Woike, Ulrich Hoffrage, and Laura Martignon. 2017. Integrating and testing natural frequencies, naïve Bayes, and fast-and-frugal trees. *Decision* 4, 4 (2017), 234.

[72] Xinli Yang, David Lo, Xin Xia, Yun Zhang, and Jianling Sun. 2015. Deep learning for just-in-time defect prediction. In *Software Quality, Reliability and Security (QRS), 2015 IEEE International Conference on*. IEEE, 17–26.

[73] Yibiao Yang, Yuming Zhou, Jinping Liu, Yangyang Zhao, Hongmin Lu, Lei Xu, Baowen Xu, and Hareton Leung. 2016. Effort-aware just-in-time defect prediction: simple unsupervised models could be better than supervised models. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 157–168.

[74] Andreas Zeller. 2002. Isolating Cause-effect Chains from Computer Programs. In *Proceedings of the 10th ACM SIGSOFT Symposium on Foundations of Software Engineering (SIGSOFT '02/FSE-10)*. ACM, New York, NY, USA, 1–10. DOI: http://dx.doi.org/10.1145/587051.587053

[75] Zhi-Qiang Zeng, Hong-Bin Yu, Hua-Rong Xu, Yan-Qi Xie, and Ji Gao. 2008. Fast training support vector machines using parallel sequential minimal optimization. In *Intelligent System and Knowledge Engineering, 2008. ISKE 2008. 3rd International Conference on*, Vol. 1. IEEE, 997–1001.

[76] Thomas Zimmermann, Rahul Premraj, and Andreas Zeller. 2007. Predicting defects for eclipse. In *Proceedings of the third international workshop on predictor models in software engineering*. IEEE Computer Society, 9.

Published in final edited form as:
Nat Mach Intell. 2019 May ; 1(5): 206–215. doi:10.1038/s42256-019-0048-x.

Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead

Cynthia Rudin
Duke University

Abstract

Black box machine learning models are currently being used for high stakes decision-making throughout society, causing problems throughout healthcare, criminal justice, and in other domains. People have hoped that creating methods for explaining these black box models will alleviate some of these problems, but trying to *explain* black box models, rather than creating models that are *interpretable* in the first place, is likely to perpetuate bad practices and can potentially cause catastrophic harm to society. There is a way forward – it is to design models that are inherently interpretable. This manuscript clarifies the chasm between explaining black boxes and using inherently interpretable models, outlines several key reasons why explainable black boxes should be avoided in high-stakes decisions, identifies challenges to interpretable machine learning, and provides several example applications where interpretable models could potentially replace black box models in criminal justice, healthcare, and computer vision.

1 Introduction

There has been an increasing trend in healthcare and criminal justice to leverage machine learning (ML) for high-stakes prediction applications that deeply impact human lives. Many of the ML models are black boxes that do not explain their predictions in a way that humans can understand. The lack of transparency and accountability of predictive models can have (and has already had) severe consequences; there have been cases of people incorrectly denied parole [1], poor bail decisions leading to the release of dangerous criminals, ML-based pollution models stating that highly polluted air was safe to breathe [2], and generally poor use of limited valuable resources in criminal justice, medicine, energy reliability, finance, and in other domains [3].

Rather than trying to create models that are inherently interpretable, there has been a recent explosion of work on "Explainable ML," where a second (posthoc) model is created to explain the first black box model. This is problematic. Explanations are often not reliable, and can be misleading, as we discuss below. If we instead use models that are inherently interpretable, they provide their own explanations, which are faithful to what the model actually computes.

cynthia@cs.duke.edu .

In what follows, we discuss the problems with Explainable ML, followed by the challenges in Interpretable ML. This document is mainly relevant to high-stakes decision making and troubleshooting models, which are the main two reasons one might require an interpretable or explainable model. Interpretability is a domain-specific notion [4, 5, 6, 7], so there cannot be an all-purpose definition. Usually, however, an interpretable machine learning model is *constrained in model form* so that it is either useful to someone, or obeys structural knowledge of the domain, such as monotonicity [e.g., 8], causality, structural (generative) constraints, additivity [9], or physical constraints that come from domain knowledge. Interpretable models could use case-based reasoning for complex domains. Often for structured data, sparsity is a useful measure of interpretability, since humans can handle at most 7±2 cognitive entities at once [10, 11]. Sparse models allow a view of how variables interact *jointly* rather than individually. We will discuss several forms of interpretable machine learning models for different applications below, but there can never be a single definition; e.g., in some domains, sparsity is useful, and in others is it not. There is a spectrum between fully transparent models (where we understand how all the variables are *jointly* related to each other) and models that are lightly constrained in model form (such as models that are forced to increase as one of the variables increases, or models that, all else being equal, prefer variables that domain experts have identified as important, see [12]).

A preliminary version of this manuscript appeared at a workshop, entitled "Please Stop Explaining Black Box Machine Learning Models for High Stakes Decisions" [13].

model types: https://scikit-learn.org/1.5/machine_learning_map.html
https://datasciencedojo.com/wp-content/uploads/ML-algorithm.png

2 Key Issues with Explainable ML

A black box model could be either (i) a function that is too complicated for any human to comprehend, or (ii) a function that is proprietary (see Appendix A). Deep learning models, for instance, tend to be black boxes of the first kind because they are highly recursive. As the term is presently used in its most common form, an explanation is a separate model that is supposed to replicate most of the behavior of a black box (e.g., "the black box says that people who have been delinquent on current credit are more likely to default on a new loan"). Note that the term "explanation" here refers to an understanding of how a model works, as opposed to an explanation of how the world works. The terminology "explanation" will be discussed later; it is misleading.

many moldes are black bx . decision trees be so complex no one can read them; NB complex functions; give some a model and ask them "what to change to acheive some effect"? for linear regression, very hard

I am concerned that the field of interpretability/explainability/comprehensibility/ transparency in machine learning has strayed away from the needs of real problems. This field dates back to the early 90's at least [see 4, 14], and there are a huge number of papers on interpretable ML in various fields (that often do not have the word "interpretable" or "explainable" in the title, as the recent papers do). Recent work on explainability of black boxes – rather than interpretability of models – contains and perpetuates critical misconceptions that have generally gone unnoticed, but that can have a lasting negative impact on the widespread use of machine learning models in society. Let us spend some time discussing this before discussing possible solutions.

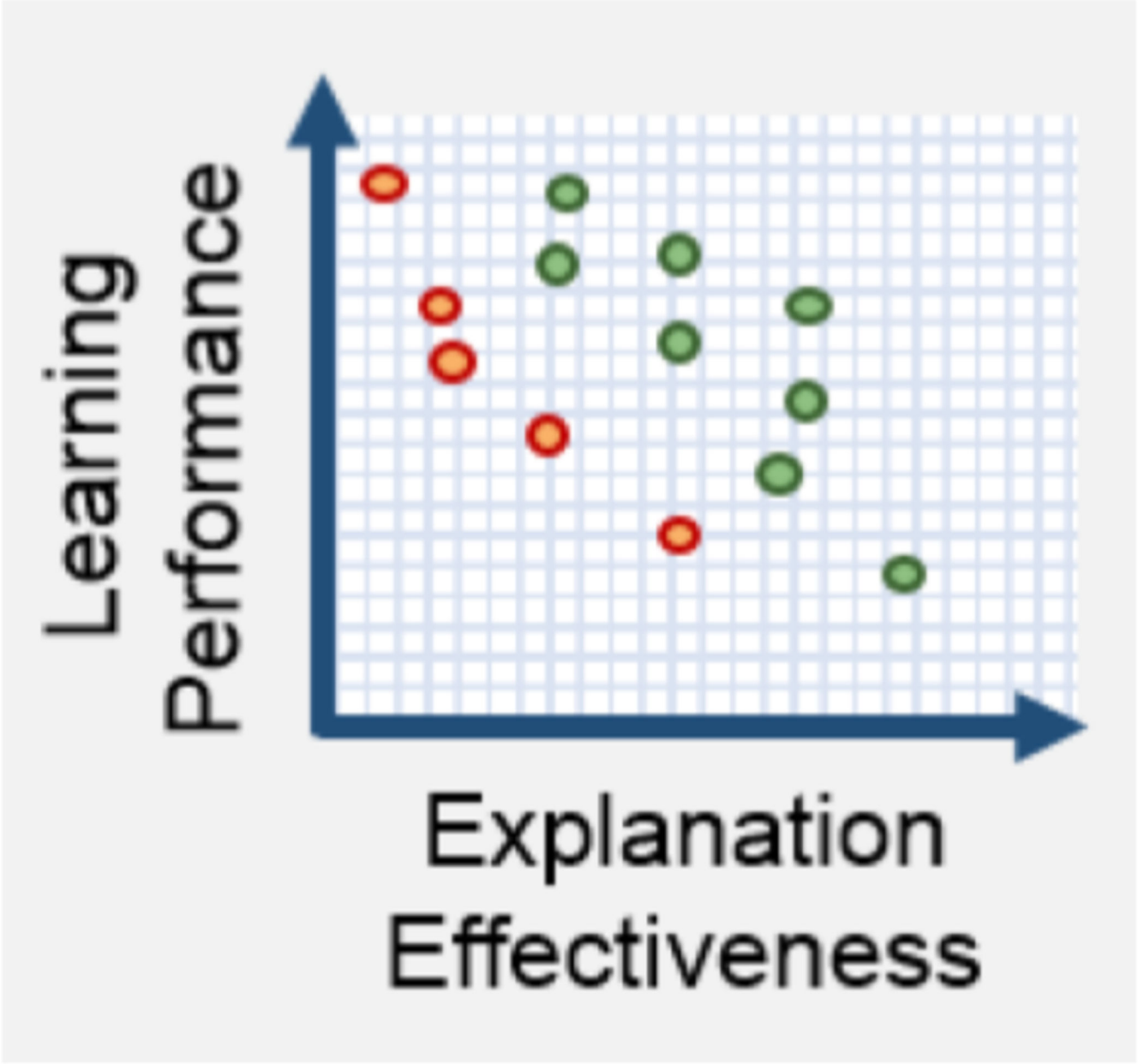


Figure 1:
A fictional depiction of the "accuracy-interpretability trade-off," taken from the DARPA XAI (Explainable Artificial Intelligence) Broad Agency Announcement [18].

(i) It is a myth that there is necessarily a trade-off between accuracy and interpretability.

There is a widespread belief that more complex models are more accurate, meaning that a complicated black box is necessary for top predictive performance. However, this is often not true, particularly when the data are structured, with a good representation in terms of naturally meaningful features. When considering problems that have structured data with meaningful features, there is often no significant difference in performance between more complex classifiers (deep neural networks, boosted decision trees, random forests) and much simpler classifiers (logistic regression, decision lists) after preprocessing. (Appendix B discusses this further.) In data science problems, where structured data with meaningful features are constructed as part of the data science process, there tends to be little difference between algorithms, assuming that the data scientist follows a standard process for knowledge discovery [such as KDD, CRISP-DM, or BigData, see 15, 16, 17].

Even for applications such as computer vision, where deep learning has major performance gains, and where interpretability is much more difficult to define, some forms of interpretability can be imbued directly into the models without losing accuracy. This will be discussed more later in the Challenges section. Uninterpretable algorithms can still be useful in high-stakes decisions as part of the knowledge discovery process, for instance, to obtain baseline levels of performance, but they are not generally the final goal of knowledge discovery.

Figure 1, taken from the DARPA Explainable Artificial Intelligence program's Broad Agency Announcement [18], exemplifies a blind belief in the myth of the accuracy-interpretability trade-off. This not a "real" figure, in that it was not generated by any data. The axes have no quantification (there is no specific meaning to the horizontal or vertical axes). The image appears to illustrate an experiment with a static dataset, where several machine learning algorithms are applied to the same dataset. However, this kind of smooth accuracy/interpretability/explainability trade-off is atypical in data science applications with meaningful features. Even if one were to quantify the interpretability/explainability axis and aim to show that such a trade-off did exist, it is not clear what algorithms would be applied to produce this figure. (Would one actually claim it is fair to compare the 1984 decision tree algorithm CART to a 2018 deep learning model and conclude that interpretable models are not as accurate?) One can always create an artificial trade-off between accuracy and interpretability/explainability by removing parts of a more complex model to reduce accuracy, but this is not representative of the analysis one would perform on a real problem. It is also not clear why the comparison should be performed on a static dataset, because any formal process for defining knowledge from data [15, 16, 17] would require an iterative process, where one refines the data processing after interpreting the results. Generally, in the practice of data science, the small difference in performance between machine learning algorithms can be overwhelmed by the ability to interpret results and process the data better at the next iteration [19]. In those cases, the accuracy/interpretability tradeoff is reversed – more interpretability leads to better overall accuracy, not worse.

Efforts working within a knowledge discovery process led me to work in interpretable machine learning [20]. Specifically, I participated in a large-scale effort to predict electrical grid failures across New York City. The data were messy, including free text documents

(trouble tickets), accounting data about electrical cables from as far back as the 1890's, inspections data from a brand new manhole inspections program; even the structured data were not easily integrated into a database, and there were confounding issues and other problems. Algorithms on a static dataset were at most 1% different in performance, but the ability to interpret and reprocess the data led to significant improvements in performance, including correcting problems with the dataset, and revealing false assumptions about the data generation process. The most accurate predictors we found were sparse models with meaningful features that were constructed through the iterative process.

The belief that there is always a trade-off between accuracy and interpretability has led many researchers to forgo the *attempt* to produce an interpretable model. This problem is compounded by the fact that researchers are now trained in deep learning, but not in interpretable machine learning. Worse, toolkits of machine learning algorithms offer little in the way of useful interfaces for interpretable machine learning methods.

To our knowledge, all recent review and commentary articles on this topic imply (implicitly or explicitly) that the trade-off between interpretability and accuracy generally occurs. It could be possible that there are application domains where a complete black box is required for a high stakes decision. As of yet, I have not encountered such an application, despite having worked on numerous applications in healthcare and criminal justice [e.g., 21], energy reliability [e.g., 20], and financial risk assessment [e.g., 22].

(ii) Explainable ML methods provide explanations that are not faithful to what the original model computes.

Explanations must be wrong. They cannot have perfect fidelity with respect to the original model. If the explanation was completely faithful to what the original model computes, the explanation would equal the original model, and one would not need the original model in the first place, only the explanation. (In other words, this is a case where the original model would be interpretable.) This leads to the danger that any explanation method for a black box model can be an inaccurate representation of the original model in parts of the feature space. [See also for instance, 23, among others.]

An inaccurate (low-fidelity) explanation model limits trust in the explanation, and by extension, trust in the black box that it is trying to explain. An explainable model that has a 90% agreement with the original model indeed explains the original model most of the time. However, an explanation model that is correct 90% of the time is wrong 10% of the time. If a tenth of the explanations are incorrect, one cannot trust the explanations, and thus one cannot trust the original black box. If we cannot know for certain whether our explanation is correct, we cannot know whether to trust either the explanation or the original model.

A more important misconception about explanations stems from the terminology "explanation," which is often used in a misleading way, because explanation models do not always attempt to mimic the calculations made by the original model. Even an explanation model that performs almost identically to a black box model might use completely different features, and is thus not faithful to the computation of the black box. Consider a black box

IF	age between 18-20 and sex is male	THEN predict arrest (within 2 years)
ELSE IF	age between 21-23 and 2-3 prior offenses	THEN predict arrest
ELSE IF	more than three priors	THEN predict arrest
ELSE	predict no arrest.	

Figure 3:
This is a machine learning model from the Certifiably Optimal Rule Lists (CORELS) algorithm [32]. This model is the minimizer of a special case of Equation 1 discussed later in the challenges section. CORELS' code is open source and publicly available at http://corels.eecs.harvard.edu/, along with the data from Florida needed to produce this model.

Table 1:

Comparison of COMPAS and CORELS models. Both models have similar true and false positive rates and true and false negative rates on data from Broward County, Florida.

COMPAS	CORELS
black box 130+ factors might include socio-economic info expensive (software license), within software used in U.S. Justice System	full model is in Figure 3 only age, priors, (optional) gender no other information free, transparent

model for criminal recidivism prediction, where the goal is to predict whether someone will be arrested within a certain time after being released from jail/prison. Most recidivism prediction models depend explicitly on age and criminal history, but do not explicitly depend on race. Since criminal history and age are correlated with race in all of our datasets, a fairly accurate explanation model could construct a rule such as "This person is predicted to be arrested because they are black." This might be an accurate explanation model since it correctly mimics the predictions of the original model, but it would not be faithful to what the original model computes. This is possibly the main flaw identified by criminologists [24] in the ProPublica analysis [25, 26] that accused the proprietary COMPAS recidivism model of being racially biased. COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) is a proprietary model that is used widely in the U.S. Justice system for parole and bail decisions. ProPublica created a linear explanation model for COMPAS that depended on race, and then accused the black box COMPAS model of depending on race, conditioned on age and criminal history. In fact, COMPAS seems to be nonlinear, and it is entirely possible that COMPAS does not depend on race (beyond its correlations with age and criminal history) [27]. ProPublica's linear model was not truly an "explanation" for COMPAS, and they should not have concluded that their explanation model uses the same important features as the black box it was approximating. (There will be a lot more discussion about COMPAS later in this document.)

An easy fix to this problem is to change terminology. Let us stop calling approximations to black box model predictions *explanations*. For a model that does not use race explicitly, an automated explanation "This model predicts you will be arrested because you are black" is not an explanation of what the model is actually doing, and would be confusing to a judge, lawyer or defendant. Recidivism prediction will be discussed more later, as it is a key application where interpretable machine learning is necessary. In any case, it can be much easier to detect and debate possible bias or unfairness with an interpretable model than with a black box. Similarly, it could be easier to detect and avoid data privacy issues with interpretable models than black boxes. Just as in the recidivism example above, many of the methods that claim to produce *explanations* instead compute useful *summary statistics of predictions* made by the original model. Rather than producing explanations that are faithful to the original model, they show trends in how predictions are related to the features. Calling these "summaries of predictions," "summary statistics," or "trends" rather than "explanations" would be less misleading.

(iii) Explanations often do not make sense, or do not provide enough detail to understand what the black box is doing.

Even if both models are correct (the original black box is correct in its prediction and the explanation model is correct in its approximation of the black box's prediction), it is possible that the explanation leaves out so much information that it makes no sense. I will give an example from image processing, for a low-stakes decision (not a high-stakes decision where explanations are needed, but where explanation methods are often demonstrated). Saliency maps are often considered to be explanatory. Saliency maps can be useful to determine what part of the image is being omitted by the classifier, but this leaves out all information about how relevant information *is* being used. Knowing where

the network is looking within the image does not tell the user what it is doing with that part of the image, as illustrated in Figure 2. In fact, the saliency maps for multiple classes could be essentially the same; in that case, the explanation for why the image might contain a Siberian husky would be the same as the explanation for why the image might contain a transverse flute.

An unfortunate trend in recent work is to show explanations only for the observation's *correct* label when demonstrating the method (e.g., Figure 2 would not appear). Demonstrating a method using explanations only for the correct class is misleading. This practice can instill a false sense of confidence in the explanation method and in the black box. Consider, for instance, a case where the explanations for multiple (or all) of the classes are identical. This situation would happen often when saliency maps are the explanations, because they tend to highlight edges, and thus provide similar explanations for each class. These explanations could be identical even if the model is *always* wrong. Then, showing only the explanations for the image's correct class misleads the user into thinking that the explanation is useful, *and* that the black box is useful, even if neither one of them are.

Saliency maps are only one example of explanations that are so incomplete that they might not convey why the black box predicted what it did. Similar arguments can be made with other kinds of explanation methods. Poor explanations can make it very hard to troubleshoot a black box.

(iv) Black box models are often not compatible with situations where information outside the database needs to be combined with a risk assessment.

In high stakes decisions, there are often considerations outside the database that need to be combined with a risk calculation. For instance, what if the circumstances of the crime are much worse than a generic assigned charge? There are often circumstances whose knowledge could either increase or decrease someone's risk. But if the model is a black box, it is very difficult to manually calibrate how much this additional information should raise or lower the estimated risk. This issue arises constantly; for instance, the proprietary COMPAS model used in the U.S. Justice System for recidivism risk prediction does not depend on the seriousness of the current crime [27, 29]. Instead, the judge is instructed to somehow manually combine current crime with COMPAS. Actually, it is possible that many judges do not know this fact. If the model were transparent, the judge could see directly that the seriousness of the current crime is not being considered in the risk assessment.

(v) Black box models with explanations can lead to an overly complicated decision pathway that is ripe for human error.

Typographical errors seem to be common in computing COMPAS, and these typographical errors sometimes determine bail decision outcomes [1, 27]. This exemplifies an important drawback of using overly complicated black box models for recidivism prediction – they may be incorrectly calculated in practice. The computation of COMPAS requires 130+ factors. If typographical errors by humans entering these data into a survey occur at a rate of 1%, then more than 1 out of every 2 surveys on average will have at least one typographical error. The multitude of typographical errors has been argued to be a type of *procedural*

unfairness, whereby two individuals who are identical might be randomly given different parole or bail decisions. These types of errors have the potential to reduce the in-practice accuracy of these complicated models.

On the separate topic of model troubleshooting, an overly complicated black box model may be flawed but we do not know it, because it is difficult to troubleshoot. Having an (incomplete) explanation of it may not help, and now we must troubleshoot two models rather than one (the black box model and the explanation model).

In the next section, we completely switch gears. We will discuss reasons why so many people appear to advocate for black box models with separate explanation models, rather than inherently interpretable models – even for high-stakes decisions.

3 Key Issues with Interpretable ML

There are many cases where black boxes with explanations are preferred over interpretable models, even for high-stakes decisions. However, for most applications, I am hopeful that there are ways around some of these problems, whether they are computational problems, or problems with training of researchers and availability of code. The first problem, however, is currently a major obstacle that I see no way of avoiding other than through policy, as discussed in the next section.

(i) Corporations can make profits from the intellectual property afforded to a black box.

Companies that charge for individual predictions could find their profits obliterated if an interpretable model were used instead.

Consider the COMPAS proprietary recidivism risk prediction tool discussed above that is in widespread use in the U.S. Justice System for predicting the probability that someone will be arrested after their release [29].

The COMPAS model is equally accurate for recidivism prediction as the very simple three rule interpretable machine learning model involving only age and number of past crimes shown in Figure 3 below. However, there is no clear business model that would suggest profiting from the simple transparent model. The simple model in Figure 3 was created from an algorithm called Certifiably Optimal Rule Lists (CORELS) that looks for if-then patterns in data. Even though the model in Figure 3 looks like a rule of thumb that a human may have designed without data, it is instead a full-blown machine learning model. A qualitative comparison of the COMPAS and CORELS models is in Table 1. Standard machine learning tools and interpretable machine learning tools seem to be approximately equally accurate for predicting recidivism, even if we define recidivism in many different ways, for many different crime types [30, 31]. This evidence, however, has not changed the momentum of the justice system towards proprietary models. As of this writing, California has recently eliminated its cash bail system, instead enforcing that decisions be made by algorithms; it is unclear whether COMPAS will be the algorithm used for this, despite the fact that it is not known to be any more accurate than other models, such as the simple CORELS model in Figure 3.

COMPAS is not a machine learning model – it was not created by any standard machine learning algorithm. It was designed by experts based on carefully designed surveys and expertise, and it does not seem to depend heavily on past criminal history [27]. Interestingly, if the COMPAS model were not proprietary, its documentation [29] indicates that it would actually be an interpretable predictive model. (It is a black box of the second type – proprietary – but not the first type – complicated – discussed above.) Revealing this model, however, would be revealing a trade secret.

Let us switch examples to consider the proprietary machine learning model by BreezoMeter, used by Google during the California wildfires of 2018, which predicted air quality as "good – ideal air quality for outdoor activities," when air quality was dangerously bad according to multiple other models [2], and people reported their cars covered in ash. The Environmental Protection Agency's free, vigorously-tested air quality index would have provided a reliable result [33]. How could BreezoMeter's machine learning method be so badly wrong and put so many in danger? We will never find out, but BreezoMeter, who has probably made a profit from making these predictions, may not have developed this new technology if its models were forced to be transparent.

In medicine, there is a trend towards blind acceptance of black box models, which will open the door for companies to sell more models to hospitals. For instance, radiology and in-hospital patient monitoring are areas of medicine that stand to gain tremendously by automation; humans cannot process data fast enough or rapidly enough to compete with machines. However, in trusting these automated systems, we must also trust the full database on which they were trained, the processing of the data, along with the completeness of the database. If the database does not represent the full set of possible situations that can arise, then the model could be making predictions in cases that are very different from anything it was trained on. An example of where this can go wrong is given by Zech et al. [34], who noticed that their neural network was picking up on the word "portable" within an x-ray image, representing the type of x-ray equipment rather than the medical content of the image. If they had used an interpretable model, or even an explainable model, this issue would never have gone unnoticed. Zech et al. [34] pointed out the issue of confounding generally; in fact, the plague of confounding haunts a vast number of datasets, and particularly medical datasets. This means that proprietary models for medicine can have serious errors. These models can also be fragile, in that if the model is used in practice in a slightly different setting than how it was trained (e.g., new x-ray equipment), accuracy can substantially drop.

The examples of COMPAS, Breezometer, and black box medical diagnosis all illustrate a problem with the business model for machine learning. In particular, there is a conflict of responsibility in the use of black box models for high-stakes decisions: *the companies that profit from these models are not necessarily responsible for the quality of individual predictions*. A prisoner serving an excessively long sentence due to a mistake entered in an overly-complicated risk score could suffer for years, whereas the company that constructed this complicated model is unaffected. On the contrary, the fact that the model was complicated and proprietary allowed the company to profit from it. In that sense, the model's designers are not incentivized to be careful in its design, performance, and ease

of use. These are some of the same types of problems affecting the credit rating agencies who priced mortgages in 2008; that is, these are the same problems that contributed to the financial crisis in the United States at that time.

One argument favoring black boxes is that keeping these models hidden prevents them from being gamed or reverse-engineered. It is not clear that this argument generally makes sense. In fact, the reason a system may be gamed is because it most likely was not designed properly in the first place, leading to a form of Goodhart's law if it were revealed. Quoting from Chang et al. [35] about product rating systems: "If the ratings are accurate measures of quality, then making the ratings more transparent could have a uniformly positive impact: it would help companies to make better rated products, it would help consumers to have these higher quality products, and it would encourage rating companies to receive feedback as to whether their rating systems fairly represent quality." Thus, transparency could help improve the quality of the system, whereby attempting to game it would genuinely align with the overall goal of improvement. For instance, improving one's credit score should actually correspond to an improvement in creditworthiness.

Another argument favoring black boxes is the belief that "counterfactual explanations" of black boxes are sufficient. A counterfactual explanation describes a minimal change to the input that would result in the opposite prediction. For instance, a possible counterfactual explanation might be "your loan application was denied, but if you had $1000 less debt, you would have qualified for the loan." This type of explanation can suffer from key issue (iv) discussed above, about combining information outside the database with the black box. In particular, the "minimal" change to the input might be different for different individuals. Appendix C discusses in more depth why counterfactual explanations generally do not suffice for high stakes decisions of black boxes.

(ii) Interpretable models can entail significant effort to construct, in terms of both computation and domain expertise.

As discussed above, interpretability usually translates in practice to a set of application-specific constraints on the model. Solving constrained problems is generally harder than solving unconstrained problems. Domain expertise is needed to construct the definition of interpretability for the domain, and the features for machine learning. For data that are unconfounded, complete, and clean, it is much easier to use a black box machine learning method than to troubleshoot and solve computationally hard problems. However, for high-stakes decisions, analyst time and computational time are less expensive than the cost of having a flawed or overly complicated model. That is, it is worthwhile to devote extra effort and cost into constructing a high-quality model. But even so, many organizations do not have analysts who have the training or expertise to construct interpretable models at all.

Some companies have started to provide interpretable ML solutions using proprietary software. While this is a step in the right direction, it is not clear that the proprietary software is better than publicly available software. For instance, claims made by some companies about performance of their proprietary algorithms are not impressive (e.g., Interpretable AI, whose decision tree performance using mixed integer programming

software in 2017 is reported to be often beaten by or comparable to the 1984 Classification and Regression Tree algorithm, CART).

As discussed earlier, interpretability constraints (like sparsity) lead to optimization problems that have been proven to be computationally hard in the worst case. The theoretical hardness of these problems does not mean we cannot solve them, though in real cases, these optimization problems are often difficult to solve. Major improvements have been made in the last decade, and some are discussed later in the Challenges section. Explanation methods, on the other hand, are usually based on derivatives, which lead to easier gradient-based optimization.

(iii) Black box models seem to uncover "hidden patterns."

The fact that many scientists have difficulty constructing interpretable models may be fueling the belief that black boxes have the ability to uncover subtle hidden patterns in the data that the user was not previously aware of. A transparent model may be able to uncover these same patterns. If the pattern in the data was important enough that a black box model could leverage it to obtain better predictions, an interpretable model might also locate the same pattern and use it. Again, this depends on the machine learning researcher's ability to create accurate-yet-interpretable models. The researcher needs to create a model that has the capability of uncovering the types of patterns that the user would find interpretable, but also the model needs to be flexible enough to fit the data accurately. This, and the optimization challenges discussed above, are where the difficulty lies with constructing interpretable models.

4 Encouraging Responsible ML Governance

Currently the European Union's revolutionary General Data Protection Regulation and other AI regulation plans govern "right to an explanation," where only an explanation is required, not an interpretable model [36], in particular "The data subject shall have the right not to be subject to a decision based solely on automated processing, including profiling, which produces legal effects concerning him or her or similarly significantly affects him or her" (Article 22 of GDPR regulations from http://www.privacy-regulation.eu/en/22.htm). If one were to provide an explanation for an automated decision, it is not clear whether the explanation is required to be accurate, complete, or faithful to the underlying model [e.g., see 37]. Less-than-satisfactory explanations can easily undermine these new policies.

Let us consider a possible mandate that, for certain high-stakes decisions, *no black box should be deployed when there exists an interpretable model with the same level of performance*. If such a mandate were deployed, organizations that produce and sell black box models could then be held accountable if an equally accurate transparent model exists. It could be considered a form of false advertising to sell a black box model if there is an equally-accurate interpretable model. The onus would then fall on organizations to produce black box models only when no transparent model exists for the same task.

This possible mandate could produce a change in the business model for machine learning. Opacity is viewed as essential in protecting intellectual property, but it is at odds with

the requirements of many domains that involve public health or welfare. However, the combination of opacity and explainability is not the only way to incentivize machine learning experts to invest in creating such systems. Compensation for developing an interpretable model could be provided in a lump sum, and the model could be released to the public. The creator of the model would not be able to profit from licensing the model over a period of time, but the fact that the models are useful for public good applications would make these problems appeal to academics and charitable foundations.

This proposal will not solve all problems, but it could at least rule out companies selling recidivism prediction models, possibly credit scoring models, and other kinds of models where we can construct accurate-yet-interpretable alternatives. If applied too broadly, it could reduce industrial participation in cases where machine learning might benefit society.

Consider a second proposal, which is weaker than the one provided above, but which might have a similar effect. Let us consider the possibility that organizations that introduce black box models would be mandated to report the accuracy of interpretable modeling methods. In that case, one could more easily determine whether the accuracy/interpretability trade-off claimed by the organization is worthwhile. This also forces the organization to try using interpretable modeling methods. It also encourages the organization to use these methods carefully, otherwise risking the possibility of criticism.

As mentioned earlier, I have not yet found a high-stakes application where a fully black box model is necessary, despite having worked on many applications. As long as we continue to allow for a broad definition of interpretability that is adapted to the domain, we should be able to improve decision making for serious tasks of societal importance. However, in order for people to design interpretable models, the technology must exist to do so. As discussed earlier, there is a formidable computational hurdle in designing interpretable models, even for standard structured data with already-meaningful features.

5 Algorithmic Challenges in Interpretable ML

What if every black box machine learning model could be replaced with one that was equally accurate but also interpretable? If we could do this, we would identify flaws in our models and data that we could not see before. Perhaps we could prevent some of the poor decisions in criminal justice and medicine that are caused by problems with using black box models. We could also eliminate the need for explanations that are misleading and often wrong.

Since interpretability is domain-specific, a large toolbox of possible techniques can come in handy. Below we expand on three of the challenges for interpretable machine learning that appear often. All three cases have something in common: people have been providing interpretable predictive models for these problems for decades, and the human-designed models look just like the type of model we want to create with machine learning. I also discuss some of our current work on these well-known problems.

Each of these challenges is a representative from a major class of models: modeling that uses logical conditions (Challenge 1), linear modeling (Challenge 2), and case-based reasoning

(Challenge 3). By no means is this set of challenges close to encompassing the large number of domain-specific challenges that exist in creating interpretable models.

Challenge #1: Constructing optimal logical models

A logical model consists of statements involving "or," "and," "if-then," etc. The CORELS model in Figure 3 is a logical model, called a *rule list*. Decision trees are logical models, as well as conjunctions of disjunctions ("or's" of "and's" – for instance, IF condition A is true OR conditions B AND C are true, THEN predict yes, otherwise predict no).

Logical models have been crafted by hand as *expert systems* as far back as the 1970's. Since then, there have been many heuristics for creating logical models; for instance, one might add logical conditions one by one (greedily), and then prune conditions away that are not helpful (again, greedily). These heuristic methods tend to be inaccurate and/or uninterpretable because they do not choose a globally best choice (or approximately best choice) for the logical conditions, and are not designed to be optimally sparse. They might use 200 logical conditions when the same accuracy could be obtained with 5 logical conditions. [C4.5 and CART 38, 39, decision trees suffer from these problems, as well as a vast number of models from the associative classification literature]. An issue with algorithms that do not aim for optimal (or near-optimal) solutions to optimization problems is that it becomes difficult to tell whether poor performance is due to the choice of algorithm or the combination of the choice of model class and constraints. (Did the algorithm perform poorly because it did not optimize its objective, or because we chose constraints that do not allow enough flexibility in the model to fit the data well?) The question of computing optimal logical models has existed since at least the mid 1990's [40].

We would like models that look like they are created by hand, but they need to be accurate, full-blown machine learning models. To this end, let us consider the following optimization problem, which asks us to find a model that minimizes a combination of the fraction of misclassified training points and the size of the model. Training observations are indexed from $i = 1,..., n$, and $\mathcal{F}$ is a family of logical models such as decision trees. The optimization problem is:

$$\min_{f \in \mathcal{F}} \left(\frac{1}{n} \sum_{i=1}^{n} 1_{[\text{training observation } i \text{ is misclassified by } f]} + \lambda \times \text{size}(f) \right). \quad (1)$$

Here, the size of the model can be measured by the number of logical conditions in the model, such as the number of leaves in a decision tree. The parameter λ is the classification error one would sacrifice in order to have one fewer term in the model; if λ is 0.01, it means we would sacrifice 1% training accuracy in order to reduce the size of the model by one. Another way to say this is that the model would contain an additional term only if this additional term reduced the error by at least 1%.

The optimization problem in (1) is generally known to be computationally hard. Versions of this optimization problem are some of the fundamental problems of artificial intelligence. The challenge is whether we can solve (or approximately solve) problems like this in practical ways, by leveraging new theoretical techniques and advances in hardware.

The model in Figure 3 is a machine learning model that comes from the CORELS algorithm [32]. CORELS solves a special case of (1), for the special choice of $\mathcal{F}$ as the set of rule lists, and where the size of the model is measured by the number of rules in the list. Figure 3 has three "if-then" rules so its size is 3. In order to minimize (1), CORELS needs to avoid enumerating all possible models, because this would take an extremely long time (perhaps until the end of the universe on a modern laptop for a fairly small dataset). The technology underlying the CORELS algorithm was able to solve the optimization problem to optimality in under a minute for the Broward County, FL, dataset discussed above. CORELS' backbone is: (i) a set of theorems allowing massive reductions in the search space of rule lists, (ii) a custom fast bit-vector library that allows fast exploration of the search space, so that CORELS does not need to enumerate all rule lists, and (iii) specialized data structures that keep track of intermediate computations and symmetries. This set of ingredients proved to be a powerful cocktail for handling these tough computational problems.

The example of CORELS enforces two points discussed above, which are, first, that interpretable models sometimes entail hard computational problems, and second, that these computational problems can be solved by leveraging a combination of theoretical and systems-level techniques. CORELS creates one type of logical model; however, there are many more. Formally, *the first challenge is to create algorithms that solve logical modeling problems in a reasonable amount of time, for practical datasets.*

We have been extending CORELS to more complex problems, such as Falling Rule Lists [41, 42], and optimal binary-split decision trees, but there is much work to be done on other types of logical models, with various kinds of constraints.

Note that it is possible to construct interpretable logical models for which the global model is large, and yet each explanation is small. This is discussed in Appendix D.

Challenge #2: Construct optimal sparse scoring systems

Scoring systems have been designed by hand since at least the Burgess criminological model of 1928 [43]. The Burgess model was designed to predict whether a criminal would violate bail, where individuals received points for being a "ne'er do well" or a "recent immigrant" that increased their predicted probability of parole violation. (Of course, this model was not created using machine learning, which had not been invented yet.) A scoring system is a sparse linear model with integer coefficients – the coefficients are the point scores. An example of a scoring system for criminal recidivism is shown in Figure 4, which predicts whether someone will be arrested within 3 years of release. Scoring systems are used pervasively throughout medicine; there are hundreds of scoring systems developed by physicians. Again, the challenge is whether scoring systems – which look like they could have been produced by a human in the absence of data – can be produced by a machine learning algorithm, and be as accurate as any other model from any other machine learning algorithm.

There are several ways to formulate the problem of producing a scoring system [see, e.g., 46, 47]. For instance, we could use a special case of (1), where the model size is the number of terms in the model. (Figure 4 is a machine learning model with 5 terms.) Sometimes, one

can round the coefficients of a logistic regression model to produce a scoring system, but that method does not tend to give accurate models, and does not tend to produce models that have particularly nice coefficients (such as 1 and −1 used in Figure 4). However, solving (1) or its variants is computationally hard, because the domain over which we solve the optimization problem is the integer lattice. (To see this, consider an axis for each of $\{b_1, b_2, \ldots, b_p\}$, where each b_j can take on integer values. This is a lattice that defines the feasible region of the optimization problem.)

The model in Figure 4 arose from the solution to a very hard optimization problem. Let us discuss this optimization problem briefly. The goal is to find the coefficients $b_j, j = 1 \ldots p$ for the linear predictive model $f(\mathbf{z}) = \sum_j b_j z_j$ where z_j is the jth covariate of a test observation $\mathbf{z}$. In Figure 4, the b_j's are the point scores, which turned out to be 1, −1, and 0 as a result of optimization, where only the nonzero coefficients are displayed in the figure. In particular, we want to solve:

$$\min_{b_1, b_2, .., b_p \in \{-10, -9, \ldots, 9, 10\}} \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + \exp\left(-\sum_{j=1}^{p} b_j x_{ij}\right)\right) + \lambda \sum_j 1[b_j \neq 0],$$

where the point scores b_j are constrained to be integers between −10 and 10, the training observations are indexed by $i = 1, \ldots, n$, and p is the total number of covariates for our data. Here the model size is the number of non-zero coefficients, and again λ is the trade-off parameter. The first term is the logistic loss used in logistic regression. The problem is hard, specifically it is a mixed-integer-nonlinear program (MINLP) whose domain is the integer lattice.

Despite the hardness of this problem, new cutting plane algorithms have been able to solve this problem to optimality (or near-optimality) for arbitrarily large sample sizes and a moderate number of variables within a few minutes. The latest attempt at solving this problem is the RiskSLIM (Risk-Supersparse-Linear-Integer-Models) algorithm, which is a specialized cutting plane method that adds cutting planes only whenever the solution to a linear program is integer-valued, and otherwise performs branching [44].

This optimization problem is similar to what physicians attempt to solve manually, but without writing the optimization problem down like we did above. Because physicians do not use optimization tools to do this, accurate scoring systems tend to be difficult for physicians to create themselves from data. One of our collaborators spent months trying to construct a scoring system himself by adding and removing variables, rounding, and using other heuristics to decide which variables to add, remove, and round. RiskSLIM was useful for helping him with this task [48]. Formally, *the second challenge is to create algorithms for scoring systems that are computationally efficient*. Ideally we would increase the size of the optimal scoring system problems that current methods can practically handle by an order of magnitude.

Challenge #3 Define interpretability for specific domains and create methods accordingly, including computer vision

Since interpretability needs to be defined in a domain-specific way, some of the most important technical challenges for the future are tied to specific important domains. Let us start with computer vision, for classification of images. There is a vast and growing body of research on posthoc explainability of deep neural networks, but not as much work in designing *interpretable neural networks*. My goal in this section is to demonstrate that even for classic domains of machine learning, where latent representations of data need to be constructed, there could exist interpretable models that are as accurate as black box models.

For computer vision in particular, there is not a clear definition of interpretability, and the sparsity-related models discussed above do not apply – sparsity in pixel space does not make sense. There can be many different ideas of what constitutes interpretability, even between different computer vision applications. However, if we can define interpretability somehow for our particular application, we can embed this definition into our algorithm.

Let us define what constitutes interpretability by considering *how people explain to each other* the reasoning processes behind complicated visual classification tasks. As it turns out, for classification of natural images, domain experts often direct our attention to different parts of the image and explain why these parts of the image were important in their reasoning process. The question is whether we can construct network architectures for deep learning that can also do this. The network must then make decisions by reasoning about parts of the image so that the explanations are real, and not posthoc.

In a recent attempt to do this, Chen, Li, and colleagues have been building architectures that append a special prototype layer to the end of the network [49, 55]. During training, the prototype layer finds parts of training images that act as prototypes for each class. For instance, for bird classification, the prototype layer might pick out a prototypical head of a blue jay, prototypical feathers of a blue jay, etc. The network also learns a similarity metric between parts of images. Thus, during testing, when a new test image needs to be evaluated, the network finds parts of the test image that are similar to the prototypes it learned during training, as shown in Figure 5. The final class prediction of the network is based on the weighted sum of similarities to the prototypes; this is the sum of evidence throughout the image for a particular class. The explanations given by the network are the prototypes (and the weighted similarities to them). These explanations are the actual computations of the model, and these are not posthoc explanations. The network is called “This look like that” because its reasoning process considers whether “this” part of the image looks like “that” prototype.

Training this prototype network is not as easy as training an ordinary neural network; the tricks that have been developed for regular deep learning have not yet been developed for the prototype network. However, so far these prototype networks have been trained to be approximately as accurate as the original black box deep neural networks they were derived from, before the prototype layer was added.

Discussion on Interpretability for Specific Domains

Let us finish this short discussion on challenges to interpretability for specific domains by mentioning that there are vast numbers of papers that have imbued interpretability in their methodology. Interpretability is not mentioned in the title of these papers, and often not in the body of the text. This is why it is almost impossible to create a review article on interpretability in machine learning or statistics without missing the overwhelming majority of it.

It is not clear why review articles for interpretability and explainability make sense to create. We do not normally have reviews of performance/accuracy measures, despite the fact that there are many of them – accuracy, area under the ROC curve, partial AUC, sensitivity, specificity, discounted cumulative gain, F-score, G-means, and many other domain-specific measures. Interpretability/explainability is just as domain-specific as accuracy performance, so it is not clear why reviews of interpretability make any more sense than reviews of accuracy/performance. I have yet to find even a single recent review that recognized the chasm between interpretability and explainability.

Let us discuss very briefly some of examples of work on interpretability that would not have been covered by recent review articles, and yet are valuable contributions to interpretability in their respective domains. Gallagher et al. [56] analyze brain-wide electrical spatiotemporal dynamics to understand depression vulnerability and find interpretable patterns in a low dimensional space. Dimension reduction to interpretable dimensions is an important theme in interpretable machine learning. Problems residing in *applied statistics* are often interpretable because they embed the physics of the domain; e.g., Wang et al. [57] create models for recovery curves for prostatectomy patients whose signal and uncertainty obey specific constraints in order to be realistic. Constraints on the uncertainty of the predictions make these models interpretable.

The setup of the recent 2018 FICO Explainable ML Challenge exemplified the blind belief in the myth of the accuracy/interpretability tradeoff for a specific domain, namely credit scoring. Entrants were instructed to create a black box to predict credit default and explain the model afterwards. *However, there was no performance difference between interpre table models and explainable models for the FICO data.* A globally interpretable model [22] won the FICO Recognition Prize for the competition. This is a case where the organizers and judges had not expected an interpretable model to be able to be constructed and thus did not ask entrants to try to construct such a model. The model of [22] was an additive model, which is a known form of interpretable model [see also 9, 58, where additive models are used for medical data]. Additive models could be optimized using similar techniques to those introduced in Challenge 2 above.

A Technical Reason Why Accurate Interpretable Models Might Exist in Many Domains

Why is it that accurate interpretable models could possibly exist in so many different domains? Is it really possible that many aspects of nature have simple truths that are waiting to be discovered by machine learning? Although that would be intriguing, I will not make this kind of Occham's-Razor-style argument, in favor of a technical argument about function

classes, and in particular, Rashomon Sets. The argument below is fleshed out more formally in [59]. This is related to (but different from) the notation of "flat minima," for which a nice example is given by Hand [19].

Here is the *Rashomon set* argument: Consider that the data permit a large set of reasonably accurate predictive models to exist. Because this set of accurate models is large, it often contains at least one model that is interpretable. This model is thus both interpretable and accurate.

Unpacking this argument slightly, for a given data set, we define the *Rashomon set* as the set of reasonably accurate predictive models (say within a given accuracy from the best model accuracy of boosted decision trees). Because the data are finite, the data could admit many close-to-optimal models that predict differently from each other: a large Rashomon set. I suspect this happens often in practice because sometimes many different machine learning algorithms perform similarly on the same dataset, despite having different functional forms (e.g., random forests, neural networks, support vector machines). As long as the Rashomon set contains a large enough set of models with diverse predictions, it probably contains functions that can be approximated well by simpler functions, and so the Rashomon set can also contain these simpler functions. Said another way, uncertainty arising from the data leads to a Rashomon set, a larger Rashomon set probably contains interpretable models, thus interpretable accurate models often exist.

If this theory holds, we should expect to see interpretable models exist across domains. These interpretable models may be hard to find through optimization, but at least there is a reason we might expect that such models exist.

If there are many diverse yet good models, it means that algorithms may not be *stable*; an algorithm might choose one model, and a small change to that algorithm or to the dataset may yield a completely different (but still accurate) model. This is not necessarily a bad thing, in fact, the availability of diverse good models means that domain experts may have more flexibility in choosing a model that they find interpretable. Appendix E discusses this in slightly more detail.

6 Conclusion

If this commentary can shift the focus even slightly from the basic assumption underlying most work in Explainable ML – which is that a black box is necessary for accurate predictions – we will have considered this document a success.

If this document can encourage policy makers not to accept black box models without significant attempts at interpretable (rather than explainable) models, that would be even better.

If we can make people aware of the current challenges right now in interpretable machine learning, it will allow policy-makers the mechanism to demand that more effort should be made in ensuring safety and trust in our machine learning models for high-stakes decisions.

If we do not succeed at these efforts, it is possible that black box models will continue to be permitted when it is not safe to use them. Since the definition of what constitutes a viable explanation is unclear, even strong regulations such as "right to explanation" can be undermined with less-than-satisfactory explanations. Further, there will continue to be problems combining black box model predictions with information outside the database, and continued miscalculations of black box model inputs. This may continue to lead to poor decisions throughout our criminal justice system, incorrect safety guidance for air quality disasters, incomprehensible loan decisions, and other widespread societal problems.

Acknowledgments

I would like to thank Fulton Wang, Tong Wang, Chaofan Chen, Oscar Li, Alina Barnett, Tom Dietterich, Margo Seltzer, Elaine Angelino, Nicholas Larus-Stone, Elizabeth Mannshart, Maya Gupta, and several others who helped my thought processes in various ways, and particularly Berk Ustun, Ron Parr, Rob Holte, and my father, Stephen Rudin, who went to considerable efforts to provide thoughtful comments and discussion. I would also like to thank two anonymous reviewers for their suggestions that improved the manuscript. I would like to acknowledge funding from the Laura and John Arnold Foundation, NIH, NSF, DARPA, the Lord Foundation of North Carolina, and MIT-Lincoln Laboratory.

A: On the Two Types of Black Box

Black box models of the first type are too complicated for a human to comprehend, and black box models of the second type are proprietary. Some models are of both types. The consequences of these two types of black box are different, but related. For instance, for a black box model that is complicated but not proprietary, we at least know what variables it uses. We also know the model form and could use that to attempt to analyze the different parts of the model. For a black box model that is proprietary but not complicated [we have evidence that COMPAS is such a model, 27], we may not even have access to query it in order to study it. If a proprietary model is too sparse, there is a risk that it could be easily reverse-engineered, thus there is an incentive to make proprietary models complicated in order to preserve their secrecy.

B: Performance Comparisons

For most problems with meaningful structured covariates, machine learning algorithms tend to perform similarly, with no algorithm clearly dominating the others. The variation due to tuning parameters of a single algorithm can often be higher than the variation between algorithms. This lack of single dominating algorithm for structured data is arguably why the field of machine learning focuses on image and speech recognition, whose data are represented in raw features (pixels, sound files); these are fields for which the choice of algorithm impacts performance. Even for complex domains such as medical records, it has been reported in some studies that logistic regression has identical performance to deep neural networks [e.g. 60].

If there is no dominating algorithm, the Rashomon Set argument discussed above would suggest that interpretable models might perform well.

Unfortunately the culture of publication within machine learning favors selective reporting of algorithms on selectively chosen datasets. Papers are often rejected if small or no

performance gains are reported between algorithms. This encourages omission of accurate baselines for comparison, as well as omission of datasets on which the method does not perform well, and encourages authors to poorly tune the parameters of baseline methods, or equivalently, place more effort into tuning the parameters of the author's own method. This creates an illusion of large performance differences between algorithms, even when such performance differences do not truly exist.

C: Counterfactual Explanations

Some have argued that counterfactual explanations [e.g., see 37] are a way for black boxes to provide useful information while preserving secrecy of the global model. Counterfactual explanations, also called inverse classification, state a change in features that is sufficient (but not necessary) for the prediction to switch to another class (e.g., "If you reduced your debt by $5000 and increased your savings by $50% then you would have qualified for the loan you applied for"). This is important for recourse in certain types of decisions, meaning that the user could take an action to reverse a decision [61].

There are several problems with the argument that counterfactual explanations are sufficient. For loan applications, for instance, we would want the counterfactual explanation to provide the *lowest cost* action for the user to take, *according to the user's own cost metric*. [See 35, for an example of lowest-cost counterfactual reasoning in product rankings]. In other words, let us say that there is more than one counterfactual explanation available (e.g., the first explanation is "If you reduced your debt by $5000 and increased your savings by $50% then you would have qualified for the loan you applied for" and the second explanation is "If you had gotten a job that pays $500 more per week, then you would have qualified for the loan"). In that case, the explanation shown to the user should be the easiest one for the user to actually accomplish. However, it is unclear in advance which explanation would be easier for the user to accomplish. In the credit example, perhaps it is easier for the user to save money rather than get a job or vice versa. In order to determine which explanation is the lowest cost for the user, we would need to elicit cost information for the user, and that cost information is generally very difficult to obtain; worse, the cost information could actually change as the user attempts to follow the policy provided by the counterfactual explanation (e.g., it turns out to be harder than the user thought to get a salary increase). For that reason it is unclear that counterfactual explanations would suffice for high stakes decisions. Additionally, counterfactual explanations of black boxes have many of the other pitfalls discussed throughout this paper.

D: Interpretable Models that Provide Smaller-Than-Global Explanations

It is possible to create a global model (perhaps a complicated one) for which explanations for any given individual are very sparse. In other words, even if the global model would take several pages of text to write, the prediction for a given individual can be very simple to calculate (perhaps requiring only 1–2 conditions). Let us consider the case of credit risk prediction. Assume we do not need to justify to the client why we would grant a loan, but we would need to justify why we would deny a loan.

Let us consider a disjunctive normal form model, which is a collection of "or's" of "and's." For instance, the model might deny a loan if "(credit history too short AND at least one bad past trade) OR (at least 4 bad past trades) OR (at least one recent delinquency AND high percentage of delinquent trades)." Even if we had hundreds of conjunctions within the model, only one of these needs to be shown to the client; if any conjunction is true, that conjunction is a defining reason why the client would be denied a loan. In other words, if the client had "at least one recent delinquency AND high percentage of delinquent trades," then regardless of any other aspects of her credit history, she could be shown that simple explanation, and it would be a defining reason why her loan application would be denied.

Disjunctive normal form models are well-studied, and are called by various names, such as "or's of and's," as well as "decision rules," "rule sets" and "associative classifiers." There has been substantial work in being able to generate such models over the past few years so that the models are globally interpretable, not just locally interpretable (meaning that the global model consists of a small number of conjunctions) [e.g., see 62, 63, 64, 65, 66].

There are many other types of models that would provide smaller-than-global explanations. For instance, falling rule lists [41, 42] provide shorter explanations for the decisions that are most important. For instance, a falling rule list for predicting patient mortality would use few logical conditions to categorize whether a patient is in a high-risk group, but use several additional logical conditions to determine which low-risk group a patient falls into.

E: Algorithm Stability

A common criticism of decision trees is that they are not stable, meaning that small changes in the training data lead to completely different trees, giving no guidance as to which tree to choose. In fact, the same problem can happen in *linear* models when there are highly correlated features. This can happen even in basic least squares, where correlations between features can lead to very different models having precisely the same levels of performance. When there are correlated features, the lack of stability happens with most algorithms that are not strongly regularized.

I hypothesize this instability in the learning algorithm could be a side-effect of the Rashomon effect mentioned earlier – that there are many different almost-equally good predictive models. Adding regularization to an algorithm increases stability, but also limits flexibility of the user to choose which element of the Rashomon set would be more desirable.

For applications where the models are purely predictive and not causal (e.g., in criminal recidivism where we use age and prior criminal history to predict future crime), there is no assumption that the model represents how outcomes are actually generated. The importance of the variables in the model does not reflect a causal relationship between the variables and the outcomes. Thus, without additional guidance from the domain expert, there is no way to proceed further to choose a single "best model" among the set of models that perform similarly. As discussed above, regularization can act as this additional input.

I view the lack of algorithmic stability as an advantage rather than a disadvantage. If the lack of stability is indeed caused by a large Rashomon effect, it means that domain experts can add more constraints to the model to customize it without losing accuracy.

In other words, while many people criticize methods such as decision trees for not being stable, I view that as a strength of interpretability for decision trees. If there are many equally accurate trees, the domain expert can pick the one that is the most interpretable.

Note that not all researchers working in interpretability agree with this general sentiment about the advantages of instability [67].

References

[1]. Wexler R When a Computer Program Keeps You in Jail: How Computers are Harming Criminal Justice. New York Times. 2017 June 13;.

[2]. McGough M How bad is Sacramento's air, exactly? Google results appear at odds with reality, some say. Sacramento Bee. 2018 August 7;.

[3]. Varshney KR, Alemzadeh H. On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. Big Data. 2016 10;5.

[4]. Freitas AA. Comprehensible classification models: a position paper. ACM SIGKDD Explorations Newsletter. 2014 Mar;15(1):1–10.

[5]. Kodratoff Y. The comprehensibility manifesto. KDD Nugget Newsletter. 1994;94(9).

[6]. Huysmans J, Dejaeger K, Mues C, Vanthienen J, Baesens B. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. Decision Support Systems. 2011;51(1):141–154.

[7]. Rüping S Learning Interpretable Models. Universität Dortmund; 2006.

[8]. Gupta M, Cotter A, Pfeifer J, Voevodski K, Canini K, Mangylov A, et al. Monotonic calibrated interpolated look-up tables. Journal of Machine Learning Research. 2016;17(109):1–47.

[9]. Lou Y, Caruana R, Gehrke J, Hooker G. Accurate Intelligible Models with Pairwise Interactions. In: Proceedings of 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). ACM; 2013..

[10]. Miller G The magical number seven, plus or minus two: Some limits on our capacity for processing information. The Psychological Review. 1956;63:81–97. [PubMed: 13310704]

[11]. Cowan N The magical mystery four how is working memory capacity limited, and why? Current directions in psychological science. 2010;19(1):51–57. [PubMed: 20445769]

[12]. Wang J, Oh J, Wang H, Wiens J. Learning Credible Models. In: Proceedings of 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). ACM; 2018. p. 2417–2426.

[13]. Rudin C Please Stop Explaining Black Box Models for High Stakes Decisions. In: Proceedings of NeurIPS 2018 Workshop on Critiquing and Correcting Trends in Machine Learning; 2018..

[14]. Holte RC. Very simple classification rules perform well on most commonly used datasets. Machine Learning. 1993;11(1):63–91.

[15]. Fayyad U, Piatetsky-Shapiro G, Smyth P. From data mining to knowledge discovery in databases. AI Magazine. 1996;17:37–54.

[16]. Chapman P, et al. CRISP-DM 1.0 - Step-by-step data mining guide. SPSS; 2000.

[17]. Agrawal D, Bernstein P, Bertino E, Davidson S, Dayal U, Franklin M, et al. Challenges and Opportunities with Big Data: A white paper prepared for the Computing Community Consortium committee of the Computing Research Association; 2012. Available from: http://cra.org/ccc/resources/ccc-led-whitepapers/.

[18]. Defense Advanced Research Projects Agency. Broad Agency Announcement, Explainable Artificial Intelligence (XAI), DARPA-BAA-16-53; 2016. Published August 10. Available from https://www.darpa.mil/attachments/DARPA-BAA-16-53.pdf.

[19]. Hand D Classifier Technology and the Illusion of Progress. Statist Sci. 2006;21(1):1–14.

[20]. Rudin C, Passonneau R, Radeva A, Dutta H, Ierome S, Isaac D. A Process for Predicting Manhole Events In Manhattan. Machine Learning. 2010;80:1–31.

[21]. Rudin C, Ustun B. Optimized Scoring Systems: Toward Trust in Machine Learning for Healthcare and Criminal Justice. Interfaces. 2018;48:399–486. Special Issue: 2017 Daniel H. Wagner Prize for Excellence in Operations Research Practice September-October 2018.

[22]. Chen C, Lin K, Rudin C, Shaposhnik Y, Wang S, Wang T. An Interpretable Model with Globally Consistent Explanations for Credit Risk. In: Proceedings of NeurIPS 2018 Workshop on Challenges and Opportunities for AI in Financial Services: the Impact of Fairness, Explainability, Accuracy, and Privacy; 2018..

[23]. Mittelstadt B, Russell C, Wachter S. Explaining Explanations in AI. In: In Proceedings of Fairness, Accountability, and Transparency (FAT*); 2019..

[24]. Flores AW, Lowenkamp CT, Bechtel K. False Positives, False Negatives, and False Analyses: A Rejoinder to "Machine Bias: There's Software Used Across the Country to Predict Future Criminals". Federal probation. 2016 September;80(2):38–46.

[25]. Angwin J, Larson J, Mattu S, Kirchner L. Machine Bias. ProPublica; 2016. Available from: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

[26]. Larson J, Mattu S, Kirchner L, Angwin J. How We Analyzed the COMPAS Recidivism Algorithm. ProPublica; 2016. https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm.

[27]. Rudin C, Wang C, Coker B. The age of secrecy and unfairness in recidivism prediction. arXiv e-prints 1811 00731 [applied statistics]. 2018 Nov;.

[28]. Checkermallow. Canis lupus winstonii (Siberian Husky); 2016.
Public domain image. https://www.flickr.com/photos/132792051@N06/28302196071/in/photolist-K7Y9RM-utZTV9-QWJmHo-QAEdSE-QAE3pL-TvjNJu-%20tziyrj-EWFwEx-DWb7T4-DTRAWu-CYLBpP-DMUVn2-dUbgLG-ccuabw-57nNvJ-UpDv4D-eNyCQP-q8aWpJ-86gced-QLBwiG-QP7k6v-aNxiRc-rmTdLW-oeTM8i-d1rkCG-ueSwz4-%20dYKwJx-7PxAPF-KFUqKN-TkarEj-7X5FZ2-7WS6Z2-7X5Gwa-7X5GkT-7Z8w5s-s4St8A-%20qsa12b-7X8Vqs-7X8VLy-7X5Gm6-7X5Gjp-PTy69W-7X8VQ3-7X8VEy-7X5GqD-iaMjUN-7X8VgE-odbiWy-TkacgQ-7X5Gk4/

[29]. Brennan T, Dieterich W, Ehret B. Evaluating the Predictive Validity of the COMPAS Risk and Needs Assessment System. Criminal Justice and Behavior. 2009 January;36(1):21–40.

[30]. Zeng J, Ustun B, Rudin C. Interpretable classification models for recidivism prediction. Journal of the Royal Statistical Society: Series A (Statistics in Society). 2017;180(3):689–722.

[31]. Tollenaar N, van der Heijden PGM. Which method predicts recidivism best?: a comparison of statistical, machine learning and data mining predictive models. Journal of the Royal Statistical Society: Series A (Statistics in Society). 2013;176(2):565–584.

[32]. Angelino E, Larus-Stone N, Alabi D, Seltzer M, Rudin C. Certifiably optimal rule lists for categorical data. Journal of Machine Learning Research. 2018;19:1–79.

[33]. Mannshardt E, Naess L. Air quality in the USA. Significance. 2018 Oct;15:24–27.

[34]. Zech JR, et al. Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study. PLoS Med. 2018;15(e1002683).

[35]. Chang A, Rudin C, Cavaretta M, Thomas R, Chou G. How to Reverse-Engineer Quality Rankings. Machine Learning. 2012 September;88:369–398.

[36]. Goodman B, Flaxman S. EU regulations on algorithmic decision-making and a 'right to explanation'. AI Magazine. 2017;38(3).

[37]. Wachter S, Mittelstadt B, Russell C. Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. Harvard Journal of Law & Technology. 2018;1(2).

[38]. Quinlan JR. C4. 5: programs for machine learning. vol. 1. Morgan Kaufmann; 1993.

[39]. Breiman L, Friedman J, Stone CJ, Olshen RA. Classification and regression trees. CRC press; 1984.

[40]. Auer P, Holte RC, Maass W. Theory and Applications of Agnostic PAC-Learning with Small Decision Trees. In: Machine Learning Proceedings 1995. San Francisco (CA): Morgan Kaufmann; 1995. p. 21–29.

[41]. Wang F, Rudin C. Falling Rule Lists. In: Proceedings of Machine Learning Research Vol. 38: Artificial Intelligence and Statistics (AISTATS); 2015. p. 1013–1022.

[42]. Chen C, Rudin C. An optimization approach to learning falling rule lists. In: Proceedings of Machine Learning Research Vol. 84: Artificial Intelligence and Statistics (AISTATS); 2018. p. 604–612.

[43]. Burgess EW. Factors determining success or failure on parole; 1928. Illinois Committee on Indeterminate-Sentence Law and Parole Springfield, IL.

[44]. Ustun B, Rudin C. Optimized Risk Scores. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD); 2017..

[45]. Ustun B, Rudin C. Supersparse linear integer models for optimized medical scoring systems. Machine Learning. 2015;p. 1–43.

[46]. Carrizosa E, Martín-Barragán B, Morales DR. Binarized support vector machines. INFORMS Journal on Computing. 2010;22(1):154–167.

[47]. Sokolovska N, Chevaleyre Y, Zucker JD. A Provable Algorithm for Learning Interpretable Scoring Systems. In: Proceedings of Machine Learning Research Vol. 84: Artificial Intelligence and Statistics (AISTATS); 2018. p. 566–574.

[48]. Ustun B, et al. The World Health Organization Adult Attention-Deficit/Hyperactivity Disorder Self-Report Screening Scale for DSM-5. JAMA Psychiatry. 2017;74(5):520–526. [PubMed: 28384801]

[49]. Chen C, Li O, Tao C, Barnett A, Su J, Rudin C. *This* Looks Like *that:* Deep Learning for Interpretable Image Recognition. In: Neural Information Processing Systems (NeurIPS); 2019..

[50]. O'Malley D Clay-colored Sparrow; 2014. Public domain image. https://www.flickr.com/photos/62798180@N03/11895857625/.

[51]. ksblack99. Clay-colored Sparrow; 2018. Public domain image. https://www.flickr.com/photos/ksblack99/42047311831/.

[52]. Schmierer A Clay-colored Sparrow; 2017. Public domain image. https://flic.kr/p/T6QVkY.

[53]. Schmierer A Clay-colored Sparrow; 2015. Public domain image. https://flic.kr/p/rguC7K.

[54]. Schmierer A Clay-colored Sparrow; 2015. Public domain image. https://www.flickr.com/photos/sloalan/16585472235/.

[55]. Li O, Liu H, Chen C, Rudin C. Deep Learning for Case-based Reasoning through Prototypes: A Neural Network that Explains its Predictions. In: Proceedings of AAAI Conference on Artificial Intelligence (AAAI); 2018. p. 3530–3537.

[56]. Gallagher N, et al. Cross-Spectral Factor Analysis. In: Proceedings of Advances in Neural Information Processing Systems 30 (NeurIPS). Curran Associates, Inc.; 2017. p. 6842–6852.

[57]. Wang F, Rudin C, Mccormick TH, Gore JL. Modeling recovery curves with application to prostatectomy. Biostatistics. 2018;p. kxy002. Available from: 10.1093/biostatistics/kxy002.

[58]. Lou Y, Caruana R, Gehrke J. Intelligible Models for Classification and Regression. In: Proceedings of Knowledge Discovery in Databases (KDD). ACM; 2012..

[59]. Semenova L, Parr R, Rudin C. A study in Rashomon curves and volumes: A new perspective on generalization and model simplicity in machine learning; 2018. In progress.

[60]. Razavian N, et al. Population-Level Prediction of Type 2 Diabetes From Claims Data and Analysis of Risk Factors. Big Data. 2015;3(4).

[61]. Ustun B, Spangher A, Liu Y. Actionable Recourse in Linear Classification. In: ACM Conference on Fairness, Accountability and Transparency (FAT*); 2019..

[62]. Su G, Wei D, Varshney KR, Malioutov DM. Interpretable Two-Level Boolean Rule Learning for Classification. In: Proceedings of ICML Workshop on Human Interpretability in Machine Learning; 2016. p. 66–70.

[63]. Dash S, Gunluk O, Wei D. Boolean Decision Rules via Column Generation. In: 32nd Conference on Neural Information Processing Systems (NeurIPS); 2018..

[64]. Wang T, Rudin C, Doshi-Velez F, Liu Y, Klampfl E, MacNeille P. A Bayesian Framework for Learning Rule Sets for Interpretable Classification. Journal of Machine Learning Research. 2017;18(70):1–37.

[65]. Rijnbeek PR, Kors JA. Finding a Short and Accurate Decision Rule in Disjunctive Normal Form by Exhaustive Search. Machine Learning. 2010 Jul;80(1):33–62.

[66]. Goh ST, Rudin C. Box Drawings for Learning with Imbalanced Data. In: Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD); 2014..

[67]. Murdoch WJ, Singh C, Kumbier K, Abbasi-Asl R, Yu B. Interpretable machine learning: definitions, methods, and applications. arXiv e-prints: 1901 04592 [statistical machine learning]. 2019 Jan;.

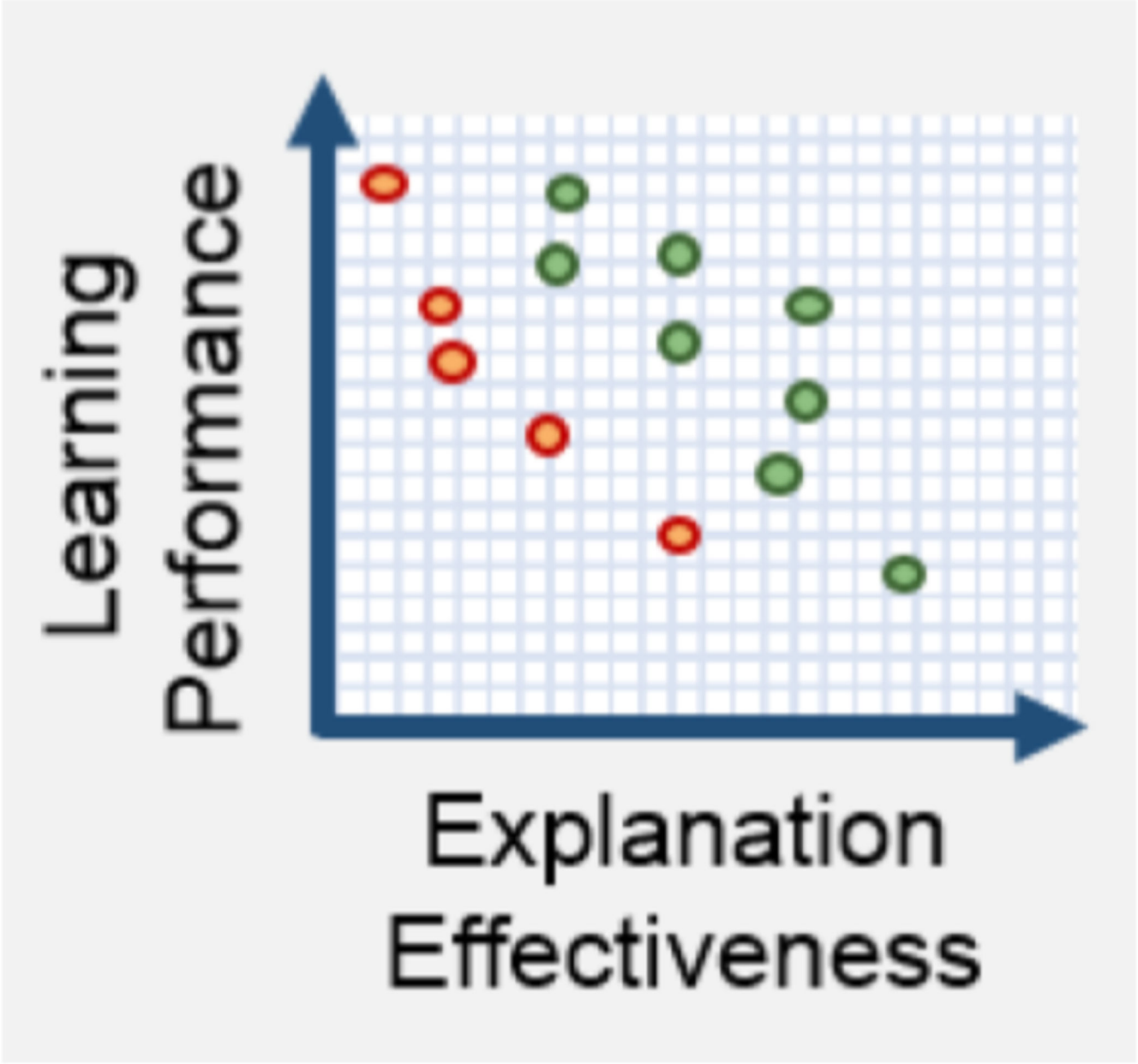


Figure 1:
A fictional depiction of the "accuracy-interpretability trade-off," taken from the DARPA XAI (Explainable Artificial Intelligence) Broad Agency Announcement [18].

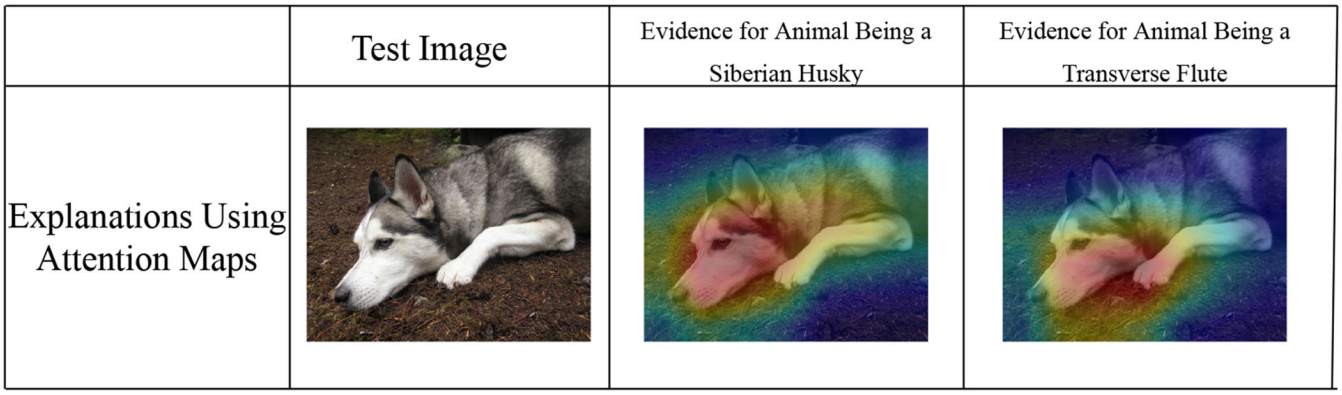


Figure 2:
Saliency does not explain anything except where the network is looking. We have no idea why this image is labeled as either a dog or a musical instrument when considering only saliency. The explanations look essentially the same for both classes. Figure credit: Chaofan Chen and [28].

IF	age between 18-20 and sex is male	THEN predict arrest (within 2 years)
ELSE IF	age between 21-23 and 2-3 prior offenses	THEN predict arrest
ELSE IF	more than three priors	THEN predict arrest
ELSE	predict no arrest.	

Figure 3:
This is a machine learning model from the Certifiably Optimal Rule Lists (CORELS) algorithm [32]. This model is the minimizer of a special case of Equation 1 discussed later in the challenges section. CORELS' code is open source and publicly available at http://corels.eecs.harvard.edu/, along with the data from Florida needed to produce this model.

1.	Prior Arrests $\geq$ 2	1 point		···
2.	Prior Arrests $\geq$ 5	1 point	+	···
3.	Prior Arrests for Local Ordinance	1 point	+	···
4.	Age at Release between 18 to 24	1 point	+	···
5.	Age at Release $\geq$ 40	-1 points	+	···
		SCORE	=	···

SCORE	-1	0	1	2	3	4
RISK	11.9%	26.9%	50.0%	73.1%	88.1%	95.3%

Figure 4:
Scoring system for risk of recidivism from [21] [which grew out of 30, 44, 45]. This model was not created by a human; the selection of numbers and features come from the RiskSLIM machine learning algorithm.

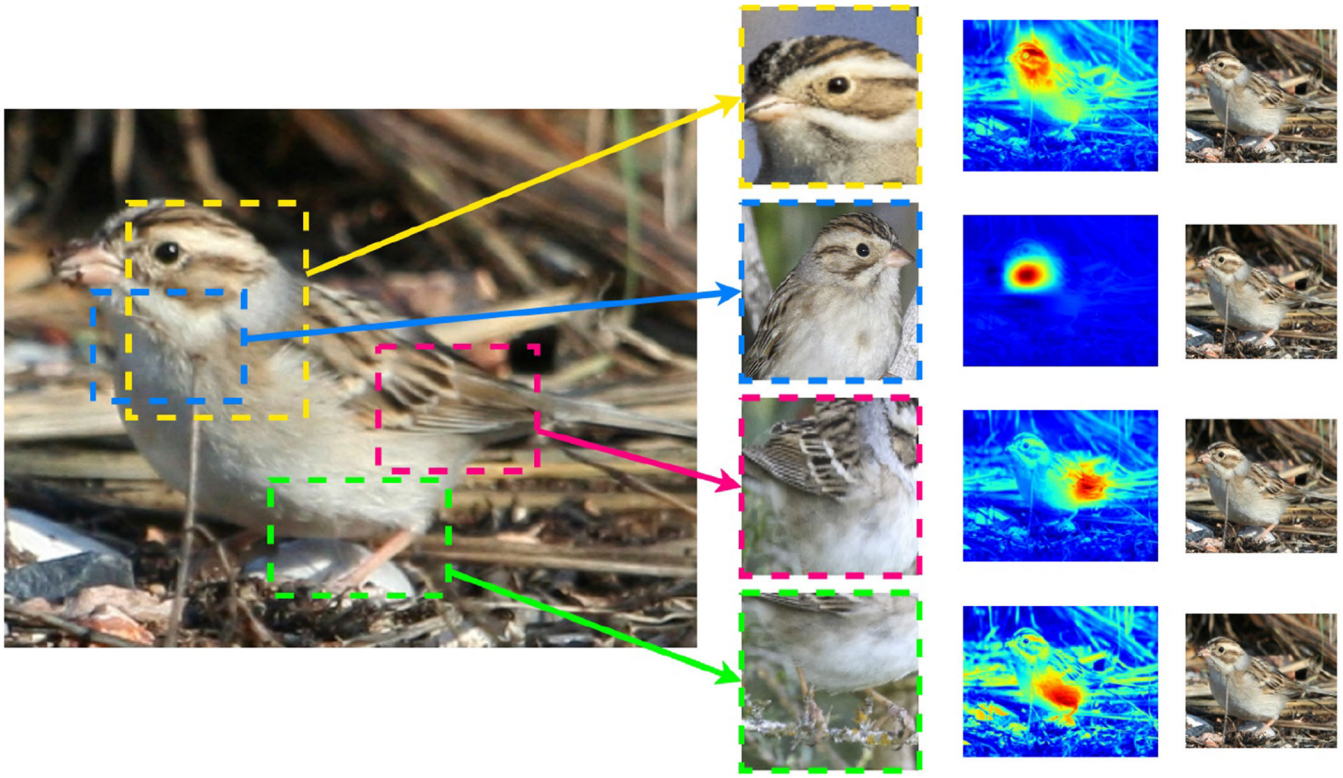


Figure 5:
Image from the authors of [49], indicating that parts of the test image on the left are similar to prototypical parts of training examples. The test image to be classified is on the left, the most similar prototypes are in the middle column, and the heatmaps that show which part of the test image is similar to the prototype are on the right. We included copies of the test image on the right so that it is easier to see what part of the bird the heatmaps are referring to. The similarities of the prototypes to the test image are what determine the predicted class label of the image. Here, the image is predicted to be a clay-colored sparrow. The top prototype seems to be comparing the bird's head to a prototypical head of a clay-colored sparrow, the second prototype considers the throat of the bird, the third looks at feathers, and the last seems to consider the abdomen and leg. Test image from [50]. Prototypes from [51, 52, 53, 54]. Image constructed by Alina Barnett.

Table 1:

Comparison of COMPAS and CORELS models. Both models have similar true and false positive rates and true and false negative rates on data from Broward County, Florida.

COMPAS	CORELS
black box 130+ factors might include socio-economic info expensive (software license), within software used in U.S. Justice System	full model is in Figure 3 only age, priors, (optional) gender no other information free, transparent