Assessing the Local Interpretability of Machine Learning Models [⋆]

Sorelle A. Friedler[1], Chitradeep Dutta Roy[2], Carlos Scheidegger[3], and Dylan Slack[1]

[1] Haverford College, Haverford, PA
{sfriedle, dslack}@haverford.edu
[2] University of Utah, Salt Lake City, UT
rahduro@cs.utah.edu
[3] University of Arizona, Tucson, AZ
cscheid@cs.arizona.edu

Abstract. The increasing adoption of machine learning tools has led to calls for accountability via model interpretability. But what does it mean for a machine learning model to be interpretable by humans, and how can this be assessed? We focus on two definitions of interpretability that have been introduced in the machine learning literature: simulatability (a user's ability to run a model on a given input) and "what if" local explainability (a user's ability to correctly indicate the outcome to a model under local changes to the input). Through a user study with 1000 participants, we test whether humans perform well on tasks that mimic the definitions of simulatability and "what if" local explainability on models that are typically considered locally interpretable. We find evidence consistent with the common intuition that decision trees and logistic regression models are interpretable and are more interpretable than neural networks. We propose a metric - the runtime operation count on the simulatability task - to indicate the relative interpretability of models and show that as the number of operations increases the users' accuracy on the local interpretability tasks decreases.

Keywords: Interpretable machine learning; Explainable artificial intelligence; Transparency

1 Introduction

With the rise of machine learning decision-making tools in critical and previously human-driven domains such as criminal justice, hiring, medicine, and scientific discovery, interpretable machine learning has become an important and growing subfield of study. The goal of interpretable machine learning is to allow oversight

[⋆] This research was funded in part by the NSF under grants NSF IIS-1633387, DMR-1709351, as well as through the DARPA Synergistic Discovery and Design (SD2) program. Code and full user study results are available at https://github.com/darkreactions/measuring_interpretability

and understanding of machine-learned decisions by, e.g., giving judges insight into decision-making procedures. Yet, while such interpretable methods have rapidly gained prominence, there has been comparatively little work on assessing whether the definitions of interpretability make sense from a human-performance perspective. In fact, we know of only one study [19] that has directly measured whether the resulting "interpretable" models are in fact interpretable for their end-users. In this work, we study the extent to which three common machine learning models are interpretable, under two definitions of interpretability from the machine learning literature, by translating these definitions into tasks to be performed by people and measuring their accuracy and completion time.

"Interpretability" as a goal can be broadly divided into *global interpretability*, meaning understanding the entirety of a trained model including all decision paths, and *local interpretability*, the goal of understanding the results of a trained model on a specific input and small deviations from that input. In this paper, we focus on local interpretability, and on two specific definitions. We assess *simulatability* [13] - here interpreted as the ability of a person to run a model and get the correct output (model classification) for a given input - and *"what if" local explainability* [20,13] - information that helps a user determine how small changes to a given input affect the model predictions. We will refer to a model as *locally interpretable* if users are able to correctly perform *both* of these tasks when given a model and input.

These definitions present a purposefully narrow view of interpretability, with the goal of studying these definitions in depth so that future work can build on this understanding. Despite the narrow scope of this work, we believe this definition of local interpretability appropriately captures an important aspect of basic interpretability. Consider a defense attorney presented with a risk assessment score and associated model for their client. Simulatability allows the defense attorney and their client to see the factors that went into the assessment and verify for themselves that the risk score was calculated correctly. Having a model that satisfies "what if" local explainability also allows the attorney to understand whether their client would have received a different score if they were another race, sex, etc., which could be critical to their ability to challenge the risk assessment in court.

We examine the simulatability and "what if" local explainability of decision trees, logistic regression, and neural networks via a crowdsourced user study. We asked 1000 users to simulate a model on a given input, produce the expected output of the model, and then determine the output for a slightly different version of that input. We measured user accuracy and completion time over varied datasets, inputs, and model types (described in detail in Section 4). The results are consistent with the folk hypotheses [13] that decision trees and logistic regression models are locally interpretable and are more locally interpretable than neural networks given the particular model representations, datasets, and user inputs used in the study.

But, as has been previously observed [13], it may be the case that a small neural network is more interpretable than a very large decision tree. To begin to answer questions like this, as well as other cross-model comparisons and generalizations of these results to models not studied here, we also investigated a measure for its suitability as a proxy for the users' ability to correctly perform both the simulation and "what if" local explainability tasks. We hypothesized that the number of program operations performed by an execution trace of the model on a given input would be a good proxy for the time and accuracy of users' attempts to locally interpret the model under both definitions; specifically, that as the total number of operations increased, the time taken would increase and the accuracy on the combined task would decrease. We found evidence that as the number of total operations performed by the model increases, the time taken by the user increases and their accuracy on the combined local interpretability task decreases. We see this work as a first step in a more nuanced understanding of the users' experience of interpretable machine learning.

2 Related Work

Work on the human interpretability of machine learning models began as early as Breiman's study of random forests [4]. Since then, many approaches to the interpretability of machine learning models have been considered, including the development of new globally interpretable models [23], post-hoc local explanations [20] and visualizations [18], and post-hoc measurement of the global importance of different features [10,6,2]. For a more detailed discussion of these methods see [15,9].

Some of the recent activity on interpretability has been prompted by Europe's General Data Protection Regulation (GDPR). A legal discussion of the meaning of the regulation with respect to interpretability is ongoing. Initially, the GDPR regulations were described as providing a "right to an explanation" [8], although subsequent work challenges that claim [25], supporting a more nuanced right to "meaningful information" about any automated decision impacting a user [22]. Exactly what is meant by interpretability to support the GDPR and in a broader legal context remains in active discussion [21].

The uncertainty around the meaning of "interpretability" has prompted calls for more precise definitions and carefully delineated goals [13]. One thought-provoking paper makes the case for a research agenda in interpretability driven by user studies and formalized metrics that can serve as validated proxies for user understanding [7]. Doshi-Velez and Kim argue that human evaluation of the interpretability of a method in its specific application context is the pinnacle of an interpretability research hierarchy followed by human evaluation of interpretability on a simplified or synthetic task and analysis of proxy tasks without associated user studies. In order to perform interpretability analysis without user studies, they argue, it is necessary to first assess proxies for user behavior. Here, we propose one such metric and assess its suitability as a proxy for the local interpretability of a model.

Although we are unaware of existing metrics for the local interpretability of a general model, many measures developed by the program analysis community aim at assessing the understandability of a general program, which could be seen as metrics for global interpretability. For example, the *cyclomatic complexity* counts the number of independent paths through a program using its control flow graph [14].

Despite calls for more experimentally grounded assessments of interpretability [7,1], there have so far been few user studies focusing on the interpretability of machine learning models. Perhaps most similar to this paper, Poursabzi-Sangdeh et al. [19] measure how the number of features and model transparency affect trust, simulatability, and mistake detection using randomized user studies on a similar scale to what we will consider here. While we focus on assessing model types, Poursabzi-Sangdeh et al. vary across general model attributes (e.g. black box vs. clear). They find that clear models, meaning the inner calculations of the models are displayed to the user, are best simulated. They also determined that the model type did not affect the trust the user had in the model nor improved users' ability to correct their mistakes.

Other related user studies include Allahyari et al. [3], who measure the *perceived* relative understandability of decision trees and rule-based models and find decision trees are seen as more understandable than rule-based models. Lage et. al. [12] optimize decision trees and neural networks for a variety of proposed interpretability proxies through human subject experiments and find the proxy their method converges on varies by data set. This result suggests that users prefer different aspects of interpretability depending on the data set. Veale et al. [24] surveyed 27 public sector machine learning professionals to determine current challenges facing the successful implementation of their work. They call for a greater focus on transparent machine learning in order to bridge the gap between those implementing algorithms and those overseeing the direction of public projects. We believe that systems that make it possible to compare the relative interpretability models could assist researchers and professionals in making more appropriate choices for their particular contexts.

3 A Metric for Local Interpretability

Motivated by the previous literature and its calls for user-validated metrics that capture aspects of interpretability, we wish to assess whether a candidate metric captures a user's ability to simulate *and* "what if" locally explain a model. The candidate metric we consider here is the *total number of runtime operation counts* performed by the model when run on a given input. We consider two basic variants of operations, arithmetic and boolean, and track their totals separately. Effectively, we seek a proxy for the work that a user must do (in their head or via a calculator) in order to simulate a model on a given input, and will claim that the total number of operations also impacts a user's ability to perform a "what if" local explanation of a model. Even though this is clearly a crude approximation of the process performed by users, we note that even this simple model appears to not have been studied explicitly.

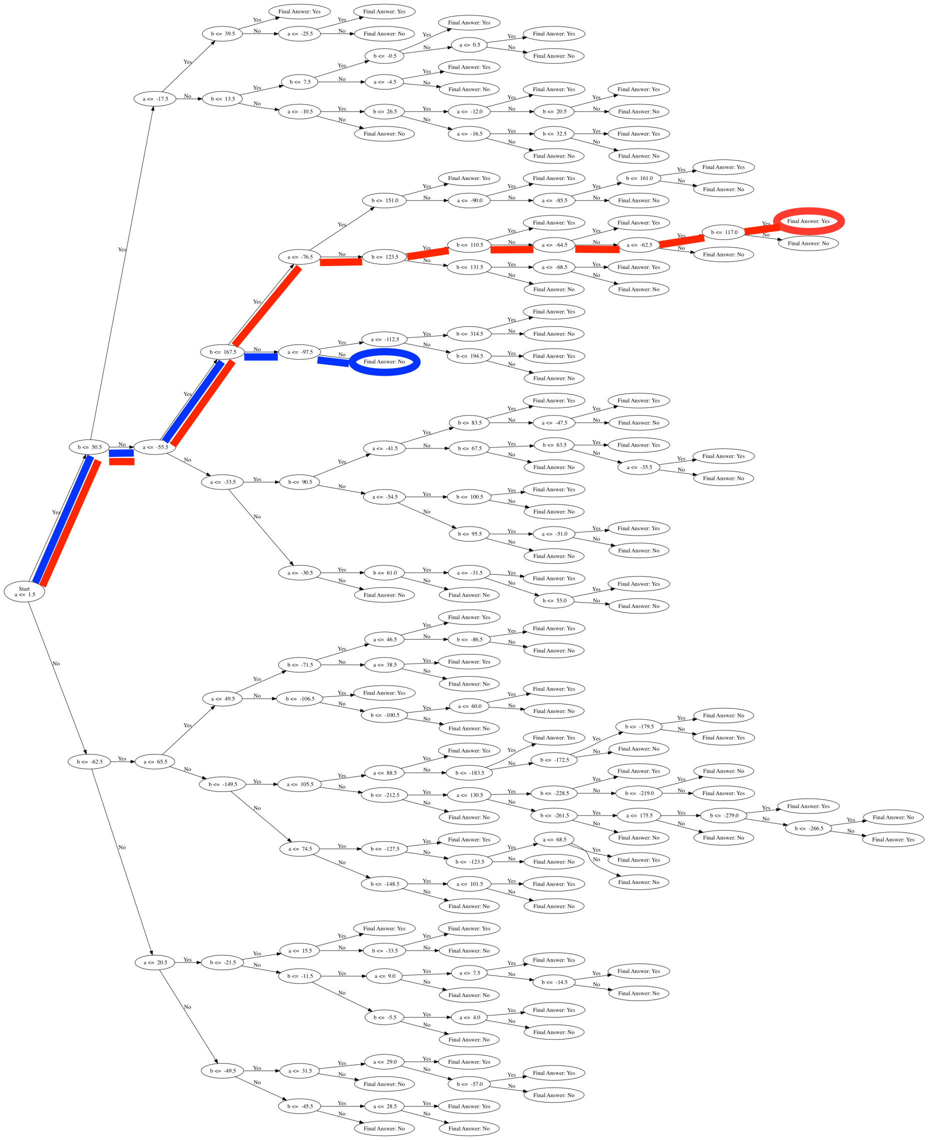


Fig. 1. A decision tree where the answer when run on the input $(a = -80, b = 200)$ is shown circled in blue and the result of running the same model on the input $(a = -64, b = 115)$ is shown circled in red.

3.1 An Example

As an example of how this metric would work, consider the visualization of a decision tree in Figure 1. The result of running the model on the input $(a = -80, b = 200)$ is shown circled in blue and the result of running the same model on the input $(a = -64, b = 115)$ is shown circled in red. The red answer is at a depth of 10 in the decision tree while the blue answer is at a depth of 5. We anticipate that users will take less time and be more accurate in answering the query that leads to the blue decision at the smaller depth. Counting the operations that the model takes to run on the input (including each boolean comparison operation or memory access required, which we count as an arithmetic operation) gives the total number of runtime operations - our candidate metric. Using the below methodology to count these operations,

the blue input is found to require 17 total operations (6 operations are arithmetic and 11 are boolean) while the red input requires 32 total operations (11 arithmetic and 21 boolean). Essentially, at each branch point one arithmetic operation is performed to do a memory access, one boolean operation is performed to check if the node is a leaf node, and one more boolean operation is performed for the branching operation.

3.2 Calculating Runtime Operation Counts

In order to calculate the number of runtime operations for a given input, we instrumented the prediction operation for existing trained models in python's scikit-learn package [5]. The source code for the technique is available at https://github.com/darkreactions/measuring_interpretability. Since most machine learning models in scikit-learn use (indirectly, via other dependencies) cython, Fortran, and C for speed and memory efficiency, we implemented a pure Python version of the `predict` method for the classifiers, and instrumented the Python bytecode directly. We created pure-Python versions of the decision tree, logistic regression, and neural network classifiers in scikit-learn.[4]

Once working only with pure Python code, we used the tracing feature of python's `sys` module and a custom tracer function to count the number of boolean and arithmetic operations. The default behavior of tracer in python is line based, meaning the trace handler method is called for each line of the source code. We used the `dis` module to modify the compiled bytecode objects of useful modules stored in their respective .pyc files. In particular, we modified the line numbering metadata so that every bytecode is given a new line number, ensuring that our tracer function is called for every bytecode instruction [17,16,11]. Inside the tracer function we use the *dis* module to determine when a byte corresponds to a valid operation and count them accordingly for our simplified *predict* method implementations when run on a given input.

[4] Specifically, `sklearn.tree.DecisionTreeClassifier`, `sklearn.linear_model.LogisticRegression`, and `sklearn.neural_network.MLPClassifier`.

4 User Study Design

We have two overall goals in this project: to assess the simulatability and "what if" local explainability of machine learning models, and to study the extent to which the proposed metric works as proxy for local interpretability. To those ends, we designed a crowdsourced experiment that was given to 1000 participants. Participants were asked to run a model on a given input and then evaluate the same model on a locally changed version of the input. We start by describing the many potentially interacting factors that required a careful experimental design.

4.1 Models and Representations

For this study we consider the local interpretability of three models: decision trees, logistic regression, and neural networks. We chose decision trees and logistic regression because they are commonly considered to be interpretable [13]. In contrast, we picked neural networks because they are commonly considered uninterpretable. These models also make for interesting objects of study since they operate in fundamentally different ways. Decision trees rely heavily on boolean operations (branching structure), logistic regression relies heavily on arithmetic operations, and neural networks use both types of operations. The models were trained using the standard package scikit-learn.[5] The neural network used is a fully connected network with 1 input layer, 1 hidden layer with 3 nodes, and 1 output layer. The `relu` (rectified linear unit) activation function was used for the hidden layer.

Perhaps the largest problem in trying to assess whether a user can interpret a model is that the user needs some sort of intelligible representation of the model. While it might be reasonable to say that the code run by the model in the `predict` method is the "purest" representation of the model, it is not likely to be understandable by a lay audience without a programming background, and might additionally suffer from code readability issues. Thus, we sought to create representations that would reveal all of the model's calculations and choices more directly. Our goal was explicitly *not* to create visualizations that might give users higher-level insights into the models' operation, but to keep to an unelaborated form of the model. Still, all our results should be understood as assessing a combination of our representations together with the models, rather than studying the models in isolation.

Our decision tree representation (see, e.g., Figure 1) is a standard node-link diagram representation for a decision tree or flow chart.

Given an input, the trained decision tree model starts at the root node (shown on the far left in the example figure) and proceeds down the tree (towards the

[5] Decision trees were trained using `sklearn.tree.DecisionTreeClassifier` without any depth restrictions and with default parameters. Logistic regression was trained using `sklearn.linear_model.LogisticRegression` with the multi_class argument set to '*multinomial*' and '*sag*'(Stochastic average gradient descent) as the solver. The neural network was implemented using `sklearn.neural_network.MLPClassifier`.

right in the figure), using the given input to answer the node questions and follow the decision path. The leaf nodes indicate the final classification result.

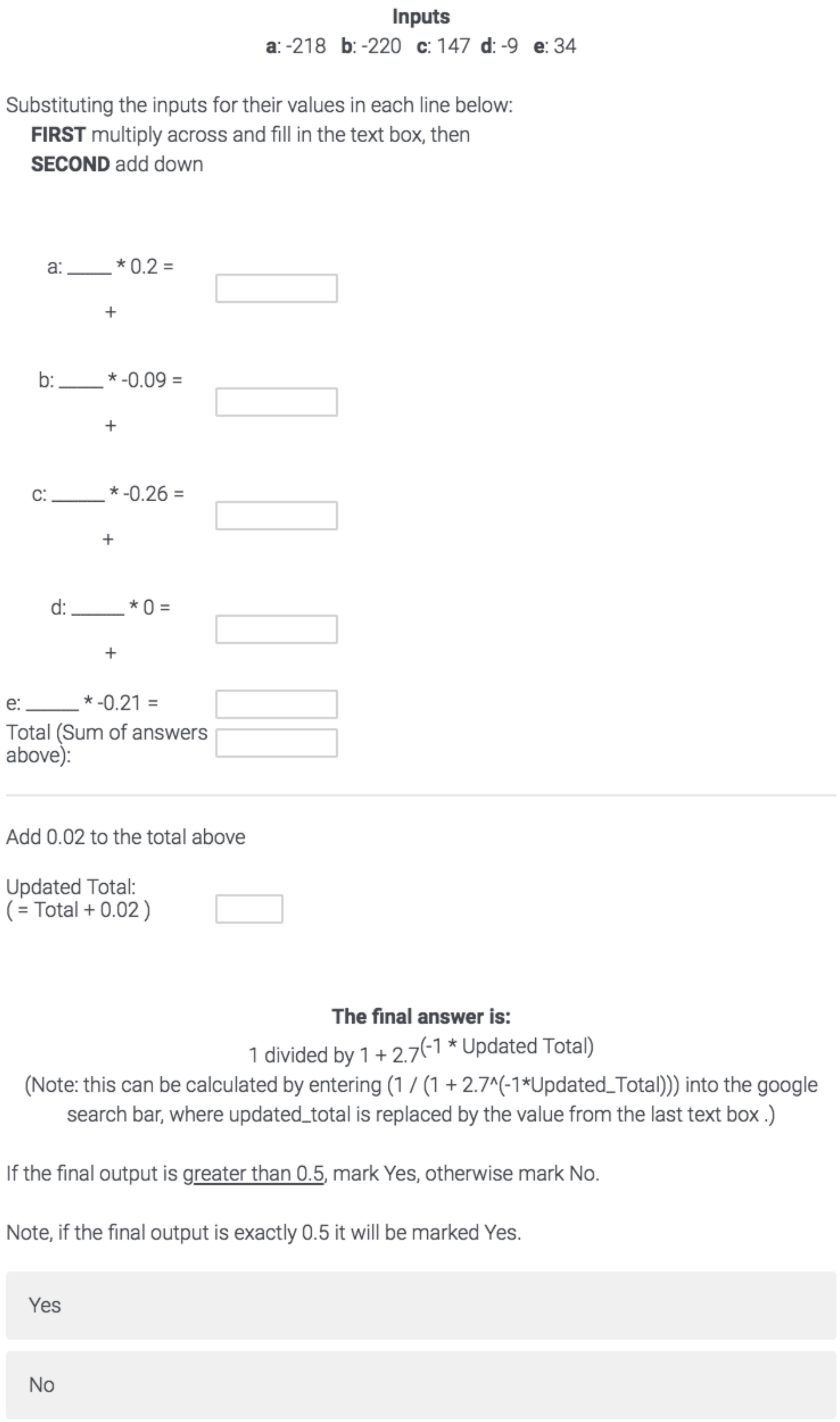
In order to allow users to simulate the logistic regression and neural network classifiers we needed a representation that would walk the users through the calculations without previous training in using the model or any assumed mathematical knowledge beyond arithmetic. The resulting representation for logistic regression is shown in Figure 2. The neural network representation used the same representation as the logistic regression for each node and one page per layer. The activation results from the previous layer were shown to the user as new inputs for the next layer. An example of the resulting representation is shown in Figure 3.

The representations described so far are for the first question a user will be asked about a model - the request to simulate it on a given input. In order to allow users to assess the "what if" local explainability of the model, we will also ask them to determine the output of the model for a perturbed version of the initial input they were shown. The representations used here are the same as the ones above, but a snapshot of the participants' previously filled in answers are shown for the logistic regression and neural network representations (see the neural network perturbed input question in Page 4 of the participant's view in Figure 3).

4.2 Data and Inputs

In order to avoid effects from study participants with domain knowledge, we chose to create synthetic datasets to train the models. We also wanted to ensure that each model would be trained on each dataset, so that, e.g., no model would seem to be more interpretable because participants were only presented with a model of that type trained on a dataset with simple and obvious patterns. Similarly, we wanted the trained models to be highly accurate, so that no model seemed more interpretable on a given dataset because it was a degenerate model that fit the data poorly. Thus, our goal was to create synthetic datasets simple enough that a logistic regression classifier could perform well on them, and ranging in complexity so that we could assess a variety of resulting operation counts.

We created synthetic datasets by choosing a number of dimensions for the dataset (2, 4, or 8) and sampling 10,000 points from a multivariate normal distribution with zero mean and 100 standard deviation. Next, we rounded the data to get integer values, which we believed users could more easily interpret. We randomly selected one coordinate as the "useful" coordinate. We labeled all points with this coordinate negative as false (-1), while all with non-negative values were labeled true (1). In order to create more complicated models that depend on all dimensions, we rotated the data by generating and applying a random rotation matrix. The values were then rounded again to integers. This yielded three datasets of 2, 4, and 8 dimensions. Additionally, we created a 2-dimensional dataset following the same procedure as above, but without the rotation matrix, with the idea that this might create especially interpretable models for users to



Inputs

a: -218 **b**: -220 **c**: 147 **d**: -9 **e**: 34

Substituting the inputs for their values in each line below:
FIRST multiply across and fill in the text box, then
SECOND add down

a: _____ * 0.2 = □

+

b: _____ * -0.09 = □

+

c: ______ * -0.26 = □

+

d: ______ * 0 = □

+

e: ______ * -0.21 = □

Total (Sum of answers above): □

Add 0.02 to the total above

Updated Total:
(= Total + 0.02) □

The final answer is:

1 divided by 1 + $2.7^{(-1 * \text{Updated Total})}$

(Note: this can be calculated by entering (1 / (1 + 2.7^(-1*Updated_Total))) into the google search bar, where updated_total is replaced by the value from the last text box .)

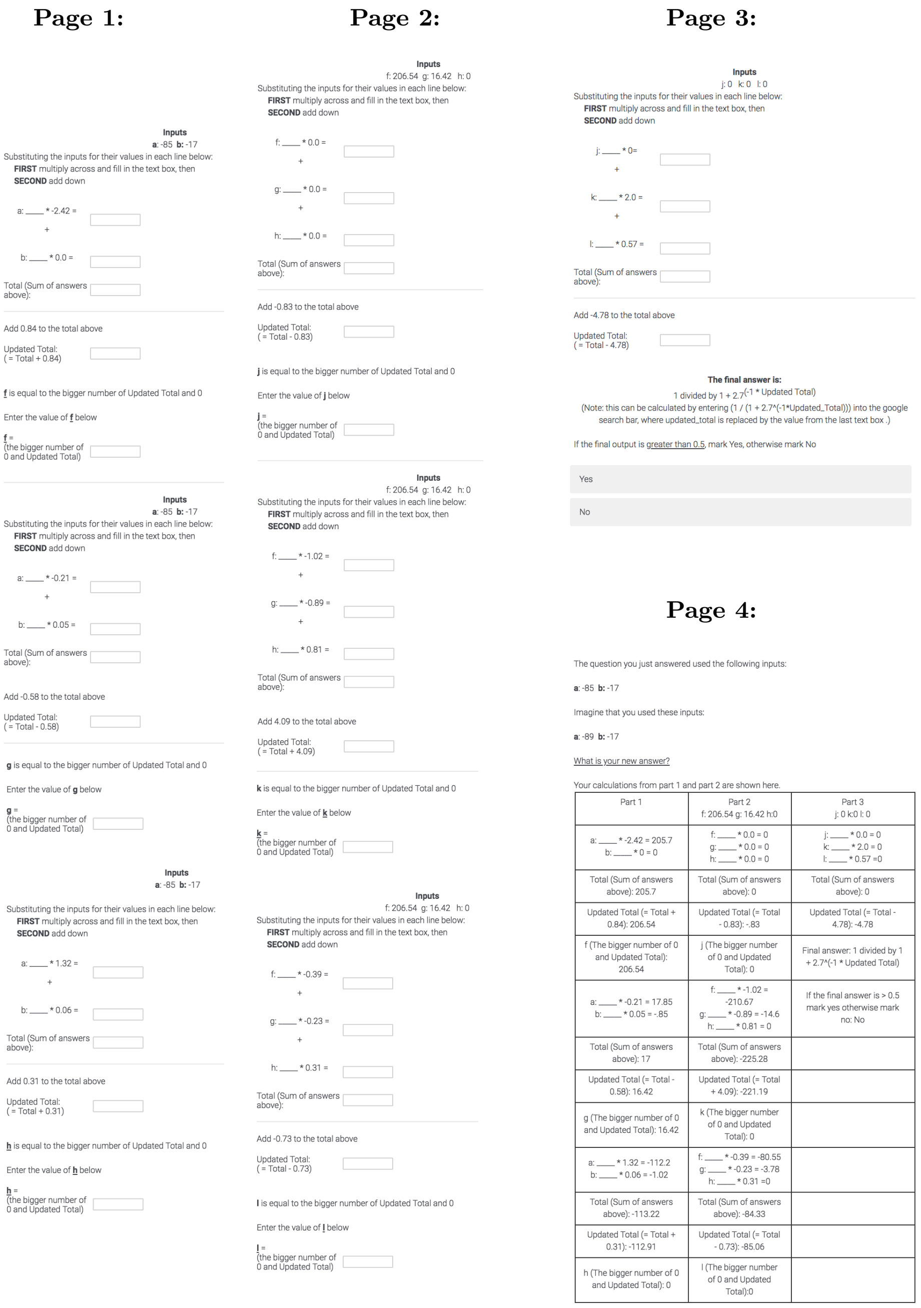
If the final output is greater than 0.5, mark Yes, otherwise mark No.

Note, if the final output is exactly 0.5 it will be marked Yes.

Yes

No

Fig. 2. The logistic regression representation shown to users.



Page 1:

Inputs
a: -85 **b:** -17

Substituting the inputs for their values in each line below:
FIRST multiply across and fill in the text box, then
SECOND add down

a: _____ * -2.42 =
+
b: _____ * 0.0 =

Total (Sum of answers above):

Add 0.84 to the total above

Updated Total:
(= Total + 0.84)

f is equal to the bigger number of Updated Total and 0

Enter the value of **f** below

f =
(the bigger number of 0 and Updated Total)

Inputs
a: -85 **b:** -17

Substituting the inputs for their values in each line below:
FIRST multiply across and fill in the text box, then
SECOND add down

a: _____ * -0.21 =
+
b: _____ * 0.05 =

Total (Sum of answers above):

Add -0.58 to the total above

Updated Total:
(= Total - 0.58)

g is equal to the bigger number of Updated Total and 0

Enter the value of **g** below

g =
(the bigger number of 0 and Updated Total)

Inputs
a: -85 **b:** -17

Substituting the inputs for their values in each line below:
FIRST multiply across and fill in the text box, then
SECOND add down

a: _____ * 1.32 =
+
b: _____ * 0.06 =

Total (Sum of answers above):

Add 0.31 to the total above

Updated Total:
(= Total + 0.31)

h is equal to the bigger number of Updated Total and 0

Enter the value of **h** below

h =
(the bigger number of 0 and Updated Total)

Page 2:

Inputs
f: 206.54 g: 16.42 h: 0

Substituting the inputs for their values in each line below:
FIRST multiply across and fill in the text box, then
SECOND add down

f: _____ * 0.0 =
+
g: _____ * 0.0 =
+
h: _____ * 0.0 =

Total (Sum of answers above):

Add -0.83 to the total above

Updated Total:
(= Total - 0.83)

j is equal to the bigger number of Updated Total and 0

Enter the value of **j** below

j =
(the bigger number of 0 and Updated Total)

Inputs
f: 206.54 g: 16.42 h: 0

Substituting the inputs for their values in each line below:
FIRST multiply across and fill in the text box, then
SECOND add down

f: _____ * -1.02 =
+
g: _____ * -0.89 =
+
h: _____ * 0.81 =

Total (Sum of answers above):

Add 4.09 to the total above

Updated Total:
(= Total + 4.09)

k is equal to the bigger number of Updated Total and 0

Enter the value of **k** below

k =
(the bigger number of 0 and Updated Total)

Inputs
f: 206.54 g: 16.42 h: 0

Substituting the inputs for their values in each line below:
FIRST multiply across and fill in the text box, then
SECOND add down

f: _____ * -0.39 =
+
g: _____ * -0.23 =
+
h: _____ * 0.31 =

Total (Sum of answers above):

Add -0.73 to the total above

Updated Total:
(= Total - 0.73)

l is equal to the bigger number of Updated Total and 0

Enter the value of **l** below

l =
(the bigger number of 0 and Updated Total)

Page 3:

Inputs
j: 0 k: 0 l: 0

Substituting the inputs for their values in each line below:
FIRST multiply across and fill in the text box, then
SECOND add down

j: _____ * 0=
+
k: _____ * 2.0 =
+
l: _____ * 0.57 =

Total (Sum of answers above):

Add -4.78 to the total above

Updated Total:
(= Total - 4.78)

The final answer is:
1 divided by 1 + $2.7^{(-1 * \text{Updated Total})}$
(Note: this can be calculated by entering (1 / (1 + 2.7^(-1*Updated_Total))) into the google search bar, where updated_total is replaced by the value from the last text box .)

If the final output is greater than 0.5, mark Yes, otherwise mark No

Yes

No

Page 4:

The question you just answered used the following inputs:

a: -85 **b:** -17

Imagine that you used these inputs:

a: -89 **b:** -17

What is your new answer?

Your calculations from part 1 and part 2 are shown here.

Part 1	Part 2 f: 206.54 g: 16.42 h:0	Part 3 j: 0 k:0 l: 0
a: _____ * -2.42 = 205.7 b: _____ * 0 = 0	f: _____ * 0.0 = 0 g: _____ * 0.0 = 0 h: _____ * 0.0 = 0	j: _____ * 0.0 = 0 k: _____ * 2.0 = 0 l: _____ * 0.57 =0
Total (Sum of answers above): 205.7	Total (Sum of answers above): 0	Total (Sum of answers above): 0
Updated Total (= Total + 0.84): 206.54	Updated Total (= Total - 0.83): -.83	Updated Total (= Total - 4.78): -4.78
f (The bigger number of 0 and Updated Total): 206.54	j (The bigger number of 0 and Updated Total): 0	Final answer: 1 divided by 1 + 2.7^(-1 * Updated Total)
a: _____ * -0.21 = 17.85 b: _____ * 0.05 = -.85	f: _____ * -1.02 = -210.67 g: _____ * -0.89 = -14.6 h: _____ * 0.81 = 0	If the final answer is > 0.5 mark yes otherwise mark no: No
Total (Sum of answers above): 17	Total (Sum of answers above): -225.28	
Updated Total (= Total - 0.58): 16.42	Updated Total (= Total + 4.09): -221.19	
g (The bigger number of 0 and Updated Total): 16.42	k (The bigger number of 0 and Updated Total): 0	
a: _____ * 1.32 = -112.2 b: _____ * 0.06 = -1.02	f: _____ * -0.39 = -80.55 g: _____ * -0.23 = -3.78 h: _____ * 0.31 =0	
Total (Sum of answers above): -113.22	Total (Sum of answers above): -84.33	
Updated Total (= Total + 0.31): -112.91	Updated Total (= Total - 0.73): -85.06	
h (The bigger number of 0 and Updated Total): 0	l (The bigger number of 0 and Updated Total):0	

Fig. 3. The neural network representation shown to users.

assess. These four datasets were used to train the three considered models via an 80/20 train-test split. All trained models had an accuracy on the test set of at least 99%.

Next, we generated a potential user input set from the test set. For each sample, we randomly chose an index (dimension) and changed the value at that index by sampling from a normal distribution centered at the original feature value with standard deviation set 10. We used a standard deviation $1/10^{th}$ the original standard deviation in order for inputs to vary noticeably but not extremely since these are the *local changes* the participants will be evaluating. The value was then rounded to an integer. Using this value for the chosen dimension and keeping all other values the same, this creates the perturbed version of the original input. From this set of input and perturbed input pairs, we then chose a set of eight pairs for each trained model (i.e., for each model type and dataset combination) to show to the participants. The set was chosen to fit the following conditions: 50% of the classifications of the original inputs are True, 50% of the classifications on the perturbed input are True, and 50% of the time the classification between input and its perturbed input changes. We used this criteria in order to distribute classification patterns evenly across users so that a distribution of random guesses by the participants would lead to 50% correctness on each task, and guessing that the perturbed input had the same outcome as the original input would also be correct 50% of the time.

4.3 Pilot Studies

In order to assess the length of the study and work out any problems with instructions, we conducted three pilot studies. In the first informal study, one of us watched and took notes while a student attempted to simulate an input on each of the three types of models and determine the outcome for a perturbed input for each of those three models. In the second two pilots we recruited about 40 participants through Prolific and gave the study for a few fixed models and inputs with the same setup as we would be using for the full study. The main takeaways from these pilot studies were that we estimated it would take users 20-30 minutes to complete the survey, but that some users would take much longer. Prolific has a timeout after which participants must have completed the survey; in the final survey we set this to 80 minutes. We had originally planned to include a dataset with 10 dimensions, and based on the time taken by users in the pilot survey decreased our largest dataset to 8 dimensions and added the 2-dimensional dataset with no rotation. We also used these pilots to confirm that we were collecting all the data we would want for our future analyses and found that our Qualtrics setup was missing timing information for the perturbed input question; this was added for the full experiment.

4.4 Experimental Setup

As in the pilot studies, we used Prolific to distribute the survey to 1000 users each of whom was paid $3.50 for completing it. Participants were restricted to those with at least a high school education (due to the mathematical nature of the task) and a Prolific rating greater than 75 out of 100. Users who participated in either of the pilot studies were prevented from taking part in this experiment. The full survey information (hosted through Qualtrics) and resulting data is available online https://github.com/darkreactions/measuring_interpretability/.

Each participant was asked to calculate the output of a machine learning model for a given input, and then to determine the output of a perturbed input applied to the same model. We showed each participant three trained models: a logistic regression, a decision tree, and a neural network in a random order. Each participant was shown a model trained on a specific dataset (chosen from the four described earlier) at most once to avoid memory effects across models. Each question began with the initial input and a brief description of the task. The per-model representations as they were shown to participants can be seen in Figures 1, 2, and 3.

Given the potential difficulty of the model simulation and local explanation tasks, we wanted to make sure that we only included users who genuinely attempted to determine the correct classification. In order to encourage effort in advance, we introduced the study to the participants in a personal and friendly way: "Hello! I'm Dylan, a student studying how well people are able to understand machine learning models. ..."

We closed the study by asking participants to self identify whether they had answered to the best of their ability, emphasizing that we would pay them even if they said no. Additionally, we included a question in the survey that asked users to do some basic addition. Based on the emails we got from participants, it appears that even many of the participants who answered "no" to our final question had worked hard to answer the questions but weren't confident in their responses.

Thus, we'll refer to all participants who said "yes" that they answered to the best of their ability and who got the basic addition question right as *confident respondents*. There were 930 confident respondents out of the total 1,000 responses. There were 18 respondents out of the total that got the basic addition question wrong. Of those that got the basic addition question wrong, 1 indicated that they did not answer to the best of their ability.

Based on our interest in evaluating the proposed local interpretability metric of the total number of runtime operations, we preregistered two experimental hypotheses. (Preregistered hypotheses can also be found at the Open Science Framework at: https://osf.io/9dbsn/.)

Hypothesis 1. For users who are able to successfully determine the classification of an instance, time to completion will be related to the total operation count so that as the total operation count increases, the time taken by the user increases.

Hypothesis 2. Users' accuracy in determining the correct classification of instances will be related to the total operation count, so that as the total operation count increases, the users' accuracy decreases.

We also preregistered the following exploratory hypotheses:

> **Exploratory Hypothesis 1.** We will explore the specific relationship of time and accuracy versus the arithmetic, boolean, and total operation counts.
>
> **Exploratory Hypothesis 2.** We will explore if and how the users' ability to determine the classification of the perturbed input is related to time and operation count.

We will additionally evaluate the "folk hypotheses" that decision trees and logistic regression models are interpretable and are more interpretable than neural networks [13] in the next section.

4.5 Ethics

Our Institutional Review Board (IRB) determined that this study was exempt from a full review due to the impersonal nature of the survey and the fully anonymized data collected. In order to treat participants ethically beyond the IRB requirements, we used Prolific to recruit study participants due to their commitment to "ethical rewards," i.e., to paying participants at least $6.50 per hour. We did our best to estimate the time the study would take participants appropriately, via the pilot studies. Based on the time actually spent by participants, Prolific calculated that we paid an average of $6.37 per hour to participants.

Initial Study A first attempt at running the full version of this study revealed further presentation issues that were not caught by the pilot studies. Specifically, since much of the Qualtrics setup for this study was manual, we had incorrectly displayed some of the previous work that users had done when they were shown their previous work on the perturbed input task. We thus fully reran the study. Where we were able to draw conclusions based on the incomplete data from this first study, initial results matched the ones we present here.

Study Setup Issues. After running the user study, we found that an error in the survey setup meant that the survey exited prematurely for users given two of the eight inputs on the decision tree models for the 2-dimensional dataset with rotation. Since we did not receive data from these participants, Prolific recruited other participants who were allocated to other inputs and datasets, so the analyzed dataset does not include data for these two inputs. Users who contacted us to let us know about the problem were still paid.

Multiple Comparison Corrections, Confidence Intervals. In order to mitigate the problem of multiple comparisons, all p-values and confidence intervals we report in the next section include a Bonferroni correction factor of 28. While we included 15 statistical tests in this paper, we considered a total of 28. Reported p-values greater than one arise from these corrections. All confidence intervals are reported at 95% confidence.

5 User Study Results

Based on the results from the described user study, we can now examine the folk hypotheses regarding the local interpretability of different model types, consider the relative local interpretability of these models, and assess our proposed metric.

5.1 Assessing the Local Interpretability of Models

			“What If”
		Simulatability	Local Explainability
	Correct	717 / 930	719 / 930
Decision Tree	p-Value	5.9×10^{-63}	5.16×10^{-64}
	95% CI	$[0.73, 0.81]$	$[0.73, 0.82]$
	Correct	592 / 930	579 / 930
Logistic Regression	p-Value	1.94×10^{-15}	2.07×10^{-12}
	95% CI	$[0.59, 0.69]$	$[0.57, 0.67]$
	Correct	556 / 930	499 / 930
Neural Network	p-Value	7.34×5.5^{-8}	0.78
	95% CI	$[0.55, 0.65]$	$[0.49, 0.59]$

Table 1. Per-model correct responses out of the total confident respondents on the original input (simulatability task) and perturbed inputs (“what if” local explainability task) for decision trees, logistic regression, and neural networks. p-values given are with respect to the null hypothesis that respondents are correct 50% of the time, using exact binomial tests.

In order to assess the local interpretability of different model types, we first separately consider the user success on the task for simulatability (the original input) and the task for “what if” local explainability (the perturbed input). Since inputs were chosen so that 50% of the correct model outputs were “yes” and 50% were “no”, we compare the resulting participant correctness rates to the null hypothesis that respondents are correct 50% of the time. The resulting p-values and confidence intervals are shown in Table 1.

The results indicate strong support for the simulatability of decision trees, logistic regression, and neural networks based on the representations the users were given. It may be surprising that neural networks were found to be simulatable; we'll explore this further in later analyses considering the relative simulatability of these models and considering what happens as the neural networks become larger. The results also indicate strong support for the “what if” local explainability of decision trees and logistic regression models, but neural networks were *not* found to be “what if” locally explainable.

Recall that we consider models to be locally interpretable if they are *both* simulatable and “what if” locally explainable. Based on the results in Table 1, we thus have evidence that decision trees and logistic regression models are locally interpretable and neural networks are not, partially validating the folk hypotheses about the interpretability of these models. Next, we'll consider the relative local interpretability of these models.

5.2 Assessing Relative Local Interpretability

Table 2. Comparative correct / incorrect distributions and *p*-values between model types generated through Fisher Exact Tests for confident responses. Relative correctness is shown for simulatability (correctness on the original input), "what if" local explainability (correctness on the perturbed input), and local interpretability (correctness on both parts). DT stands for Decision Tree, LR stands for Logistic Regression, and NN stands for Neural Network.

Relative Simulatability:

Contingency Table	DT > NN		DT > LR		LR > NN	
Correct	717	556	717	592	592	556
Incorrect	213	374	213	338	338	374
p-value, 95% CI	1.5×10^{-14}	$[1.69, \infty]$	3.7×10^{-9}	$[1.43, \infty]$	1.3	$[0.90, \infty]$

Relative "What If" Local Explainability:

Contingency Table	DT > NN		DT > LR		LR > NN	
Correct	719	499	719	579	579	499
Incorrect	211	431	211	351	351	431
p-value, 95% CI	7.3×10^{-26}	$[2.20, \infty]$	2.6×10^{-11}	$[1.54, \infty]$	2.9×10^{-3}	$[1.09, \infty]$

Relative Local Interpretability:

Contingency Table	DT > NN		DT > LR		LR > NN	
Correct	594	337	594	425	425	337
Incorrect	336	593	336	505	505	593
p-value, 95% CI	9.3×10^{-32}	$[2.36, \infty]$	5.9×10^{-14}	$[1.60, \infty]$	5.7×10^{-4}	$[1.13, \infty]$

In order to assess the relative local interpretability of models — to evaluate the folk hypothesis that decision trees and logistic regression models are more interpretable than neural networks — we compared the distributions of correct and incorrect answers on both tasks across pairs of model types. We applied one-sided Fisher exact tests with the null hypothesis that the models were equally simulatable, "what if" locally explainable, or locally interpretable. The alternative hypotheses were that decision trees and logistic regression models were more interpretable (had a greater number of correct responses) than neural networks and that decision trees were more interpretable than logistic regression models.

The results, presented in Table 2, give strong evidence that decision trees are more locally interpretable than logistic regression or neural network models on both the simulatability and “what if” local explainability tasks. Interestingly, while there was strong evidence that logistic regression is more “what if” locally explainable and more locally interpretable than neural networks, there is not evidence that logistic regression models are more simulatable than neural networks using the given representations. This may be because the logistic regression and neural network representations were very similar. An analysis of the users who got both tasks right, i.e., were able to locally interpret the model, shows that the alternative hypothesis was strongly supported in all three cases, thus supporting the folk hypotheses that decision trees and logistic regression models are more interpretable than neural networks.

5.3 Assessing Runtime Operations as a Metric for Local Interpretability

In order to evaluate our preregistered hypotheses, we considered the relationship between total operation counts, time, and accuracy on the simulatability, “what if” local explainability, and combined local interpretability tasks. The graphs showing these relationships, including ellipses that depict the degree to which the different measurements are linearly related to each other, are shown in Figure 4. The time and accuracy given for the simulatability and “what if” local explainability tasks are separated individually for those tasks in the first two columns of the figure, while the final local interpretability column includes the sum of the time taken by the user on both tasks and credits the user with an accurate answer only if both the simulatability and “what if” local explainability tasks were correctly answered. The accuracies as displayed in the figure are averaged over all users given the same input into the trained model. All total operation counts given are for the simulation task on the specific input. In the case of the “what if” local explainability task for decision trees, this operation count is for the simulatability task on the perturbed input; the logistic regression and neural network simulatability operation counts do not vary based on input. The local interpretability total operation count is the sum of the counts on the simulatability and “what if” local explainability tasks. Additionally, we considered the effect on time and accuracy of just the arithmetic operation counts. The overall trends were the same as are discussed below, so these results are not shown here.

Time Across all three interpretability tasks it appears clear that as the number of operations increases, the total time taken by the user also increases (see the first row of Figure 4). This trend is especially clear for the simulatability task, validating Hypothesis 1. This effect is perhaps not surprising, since the operation count considered is for the simulatability task and the representations given focus on performing each operation. Perhaps more surprisingly, as the total operation count on the simulatability task increases, the total time taken on the “what if” local explainability task also increases; though that pattern is most clear for the decision tree models. When considering the combined local interpretability task, this upward trend in time is also apparent.

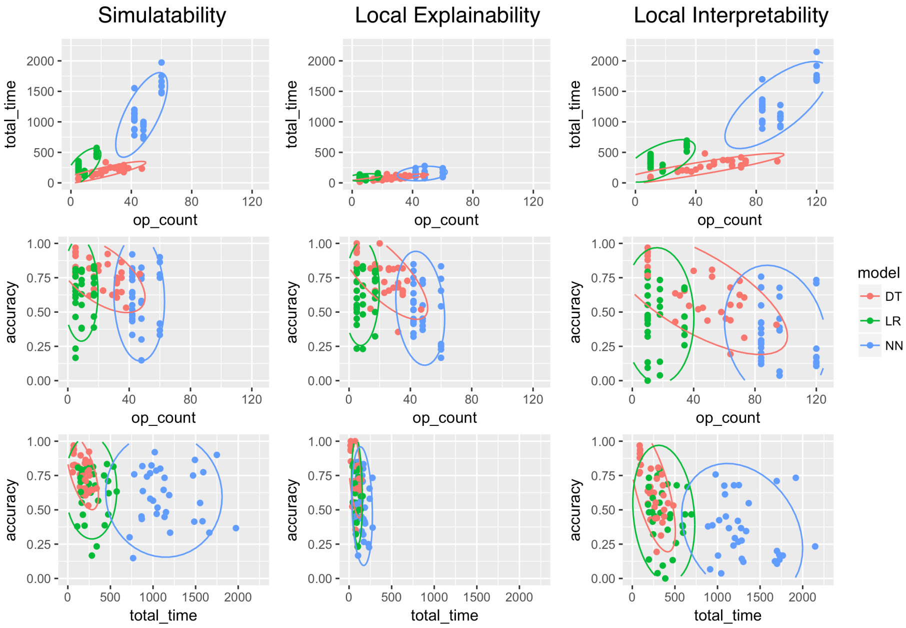


Fig. 4. Comparisons shown are between total operations for a particular trained model and input, the time taken by the user to complete the task, and the accuracy of the users on that task for the simulatability (original input), "what if" local explainability (perturbed input), and the combined local interpretability (getting both tasks correct) tasks. The total time shown is in seconds. The total operation count is for the simulatability task on the specific input; this is the same for both "what if" local explainability and simulatability except for in the case of the decision tree models, where operation counts differ based on input. The local interpretability operation count is the sum for the simulatability and "what if" local explainability task operation counts. Accuracy shown is averaged over all users who were given the same input for that task and trained model. The models considered are decision trees (DT), logistic regression models (LR), and neural networks (NN). The ellipses surrounding each group depict the covariance between the two displayed variables, and capture 95% of the sample variance.

Accuracy As the total number of runtime operations increases, we hypothesized that the accuracy would decrease. In the second row of Figure 4 we can see that this trend appears to hold clearly for all three interpretability tasks for the decision tree models, but there is no clear trend for the logistic regression and neural network models. This lack of effect may be due to the comparatively smaller range of operation counts examined for these two model types, or it may be that the local interpretability of these model types is not as related to operation count as it is for decision trees. The lack of overlap in the ranges for the operation counts of logistic regression and neural networks also makes it hard to separate the effects of the model type on the results.

As can be seen in the bottom row of Figure 4, user accuracy on the tasks varies somewhat with time. Although the accuracy seems in general to decrease as users take more time on the task, it does not appear to be a strong trend except for with decision tree models.

6 Discussion and Conclusion

We investigated the local interpretability of three common model types — decision trees, logistic regression, and neural networks – and showed support via a user study for the folk hypotheses that decision trees and logistic regression models are locally interpretable while neural networks are not. We also found that decision trees are more locally interpretable than logistic regression or neural network models. We introduced a metric for local interpretability — the runtime operation count of the model — and showed that the number of runtime operations has a positive relationship to the time a user takes to locally interpret a model and a negative relationship to the users' accuracy on the local interpretation task (the ability to both simulate and "what if" locally explain a model). The introduction of this metric opens the possibility of analyzing other model types for their local interpretability without running a user study.

However, there are many caveats and limitations to the reach of this work. The domain-agnostic nature of our synthetic dataset has transferability advantages, but also has disadvantages in that it does not study interpretability within its target domain. The definitions of local interpretability that we assess here — simulatability and "what if" local explainability— are limited in their reach and the specific user study setup that we introduce may be limited in capturing the nuance of these definitions. Still, we hope that this work can provide early insight into how user studies can help to validate notions of interpretability in machine learning and encourage others to investigate the growing and important claims of interpretability in critical domains.

References

1. Abdul, A., Vermeulen, J., Wang, D., Lim, B.Y., Kankanhalli, M.: Trends and trajectories for explainable, accountable and intelligible systems: An hci research agenda. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. p. 582. ACM (2018)
2. Adler, P., Falk, C., Friedler, S.A., Nix, T., Rybeck, G., Scheidegger, C., Smith, B., Venkatasubramanian, S.: Auditing black-box models for indirect influence. Knowledge and Information Systems **54**(1), 95–122 (2018)
3. Allahyari, H., Lavesson, N.: User-oriented assessment of classification model understandability. In: 11th scandinavian conference on Artificial intelligence. IOS Press (2011)
4. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
5. Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G.: API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning. pp. 108–122 (2013)
6. Datta, A., Sen, S., Zick, Y.: Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In: Security and Privacy (SP), 2016 IEEE Symposium on. pp. 598–617. IEEE (2016)
7. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608 (2017)
8. Goodman, B., Flaxman, S.: European union regulations on algorithmic decision-making and a" right to explanation". presented at the 2016 ICML Workshop on Human Interpretability in Machine Learning (WHI 2016), New York, NY (2016)
9. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM Computing Surveys (CSUR) **51**(5), 93 (2018)
10. Henelius, A., Puolamäki, K., Boström, H., Asker, L., Papapetrou, P.: A peek into the black box: exploring classifiers by randomization. Data mining and knowledge discovery **28**(5-6), 1503–1529 (2014)
11. Ike-Nwosu, O.: Inside Python Virtual Machine, chap. 5 Code Objects, pp. 68–78. Lean publishing, 1st edn. (2018)
12. Lage, I., Slavin Ross, A., Kim, B., J. Gershman, S., Doshi-Velez, F.: Human-in-the-loop interpretability prior. In: Conference on Neural Information Processing Systems (NeurIPS) (2018)
13. Lipton, Z.C.: The mythos of model interpretability. Queue **16**(3), 30 (2018)
14. McCabe, T.J.: A complexity measure. IEEE Transactions on software Engineering (4), 308–320 (1976)
15. Molnar, C.: Interpretable Machine Learning. https://christophm.github.io/interpretable-ml-book/ (2018), https://christophm.github.io/interpretable-ml-book/
16. Ned, B.: The structure of .pyc files. Blog (April 2008), https://nedbatchelder.com/blog/200804/the_structure_of_pyc_files.html
17. Ned, B.: Wicked hack: Python bytecode tracing. Blog (April 2008), https://nedbatchelder.com/blog/200804/wicked_hack_python_bytecode_tracing.html
18. Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., Mordvintsev, A.: The building blocks of interpretability. Distill (2018). https://doi.org/10.23915/distill.00010, https://distill.pub/2018/building-blocks

19. Poursabzi-Sangdeh, F., Goldstein, D.G., Hofman, J.M., Vaughan, J.W., Wallach, H.: Manipulating and measuring model interpretability. Transparent and Interpretable Machine Learning in Safety Critical Environments Workshop at NIPS (2017)
20. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should i trust you?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1135–1144. ACM (2016)
21. Selbst, A.D., Barocas, S.: The intuitive appeal of explainable machines. Fordham Law Review. Forthcoming. Available at SSRN: https://ssrn.com/abstract=3126971 (2018)
22. Selbst, A.D., Powles, J.: Meaningful information and the right to explanation. International Data Privacy Law **7**(4), 233–242 (2017)
23. Ustun, B., Traca, S., Rudin, C.: Supersparse linear integer models for interpretable classification. arXiv preprint arXiv:1306.6677 (2013)
24. Veale, M., Van Kleek, M., Binns, R.: Fairness and accountability design needs for algorithmic support in high-stakes public sector decision-making. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. p. 440. ACM (2018)
25. Wachter, S., Mittelstadt, B., Floridi, L.: Why a right to explanation of automated decision-making does not exist in the general data protection regulation. International Data Privacy Law **7**(2), 76–99 (2017)