Explore, Edit, Guess: Understanding Novice Programmers' Use of CodeBlocks for Regression Experiments

Jose Maria Santiago III[1], Giselle Nodalo[1], Jolene Valenzuela[1] and Jordan Aiko Deja[1,2]

[1]*De La Salle University, Center for Complexity and Emerging Technologies, 2401 Taft Avenue, PH-1002 Manila, Philippines*

[2]*University of Primorska, Faculty of Mathematics Natural Sciences and Information Technologies, Glagoljaška 8, 6000 Koper, Slovenia*

Abstract

Machine Learning (ML) suites, while readily available for practical use, are not easily usable. There are several publicly available packaged software and console libraries that are available as well. However, novice programmers tend to avoid these ML suites due to the abstractions with pre-existing tools. Users who know how these models work have limited programming experience and therefore struggle with making practical use of these ML suites. To address this, we present TREX: A Toolbox for Regression Experiments which uses code-blocks. It provides a sandbox environment so novices can freely use and experiment with models and bridge how these models and code work altogether. First, we developed a prototype following an iterative approach, focusing on linear regression tasks. Across several software iterations, multiple usability tests involving at least 33 respondents participated. Second, we observed how they went through their ML tasks and inquired into their pains and struggles in writing code. Lastly, we analysed these findings and provided recommendations for the future design of interactive ML tools.

Keywords

Machine Learning, CodeBlocks Programming, Novice Programmer Practices, Interactive Machine Learning

1. Background and Related Work

Packaged Machine Learning (ML) programs such as WEKA and RapidMiner, were designed to enable novices to implement ML in projects without worrying about the code. However, users do not see the inner workings, often referred to as the BlackBox effect. As a result, these users typically experience some issues when starting. These users are either (1) computing professionals with adequate programming experience but lack the adequate foundation in ML or (2) professionals with preliminary knowledge on ML pipeline activities but lack adequate programming experience. A comparison done by [1] found that visual tools, such as WEKA & Rapidminer, are not entirely interactive. They provide limited user feedback and abstracts the algorithm code from the user. In the work of [2], novice ML students are exposed to ML while utilizing the interactive environment of spreadsheets. These platforms are often limited

Human-Computer Interaction Slovenia 2021, November 11, 2021, Koper, Slovenia

jose_maria_santiago@dlsu.edu.ph (J. M. Santiago III); giselle_nodalo@dlsu.edu.ph (G. Nodalo); jolene_valenzuela@dlsu.edu.ph (J. Valenzuela); jordan.deja@dlsu.edu.ph (J. A. Deja)

to their domain, such as finance and economics. Yet end-users easily understand how minor tweaking of formulas affects the output of these systems due to the familiarity of spreadsheet interactions. They also widen the range of ML practicality, allowing it to be expanded to other fields. The study of [3, 4] investigates the use of an Interactive Machine Learning (iML) system that classified groups of people on social networking sites to reduce the instance of unwanted data leaking to users. These systems allow end-users to work & explore their experiments while considering specific constraints and ethical considerations. The works of [5, 6] attempted to design sandbox environments for regression experiments. They consider steering an inspection of ML but do not investigate the activities and behaviours of the users themselves.

Tensorflow[1], an open-source ML framework that deals with high numerical computations, is designed to be easily accessible and for faster processing of ML algorithms, especially those that can be programmed with Tensors. TensorFlow Playground[2] is an exploratory sandbox that allows users to visualize how a Neural Network learns. The playground displays various components that users can tweak, affecting the output displayed on the interface. Algorithm Visualization (AV) is essential in improving a user's understanding of an algorithm's function [7]. The work of [8] used dynamic AV that exposed novice learners to underlying algorithm logic. Platforms like this help fit the mental model of their users. Also, this balances whatever perceptions they have about the behaviour of the algorithm itself. It has been observed that the programmers in agile settings consider factors that give them flexibility instead [9]. However, the study of [2] observed users that often preferred a specific user flow as a result of using these iML systems, which were discovered from applying user-centred design processes in an agile setting. Ideally, this approach involving end-user innovation on iML systems states that the user-centred design process can be used to improve usability itself for novice users [10] and systems targeted towards them. Several works have investigated on behaviours and productivity of programmers coming from different backgrounds and experiences. Our work explores the underlying design factors that affect these behaviours—attempting to understand the human factors involving novice users operating in an unfamiliar, block-based visual interface. In this paper, we: (1) Observe & understand how programmers implement code when given a linear regression task. We (2) discuss design implications for supporting the interaction needs of a novice user. Lastly, we (3) reflect on designing better interactive ML platforms that help novice-expert transitions in regression experiments.

2. Method

In this study, we subscribed to the iterative user-centric design approach as seen in the works of [1, 11, 12, 13, 14, 15] where we gathered initial user data and performed affinity analysis on the results. The process of extracting insights and deriving guidelines for novices follows the work of [16] where they investigated novice programmers and came up with design guidelines that lead to coming up with prototype features as well. Generally, it had the Exploration phase, Design phase and Test phase. An overview can be seen in Figure 1. On top of these phases, we performed additional steps that involved reviewing and assessing guidelines and heuristics and

[1]https://tensorflow.org
[2]https://playground.tensorflow.org

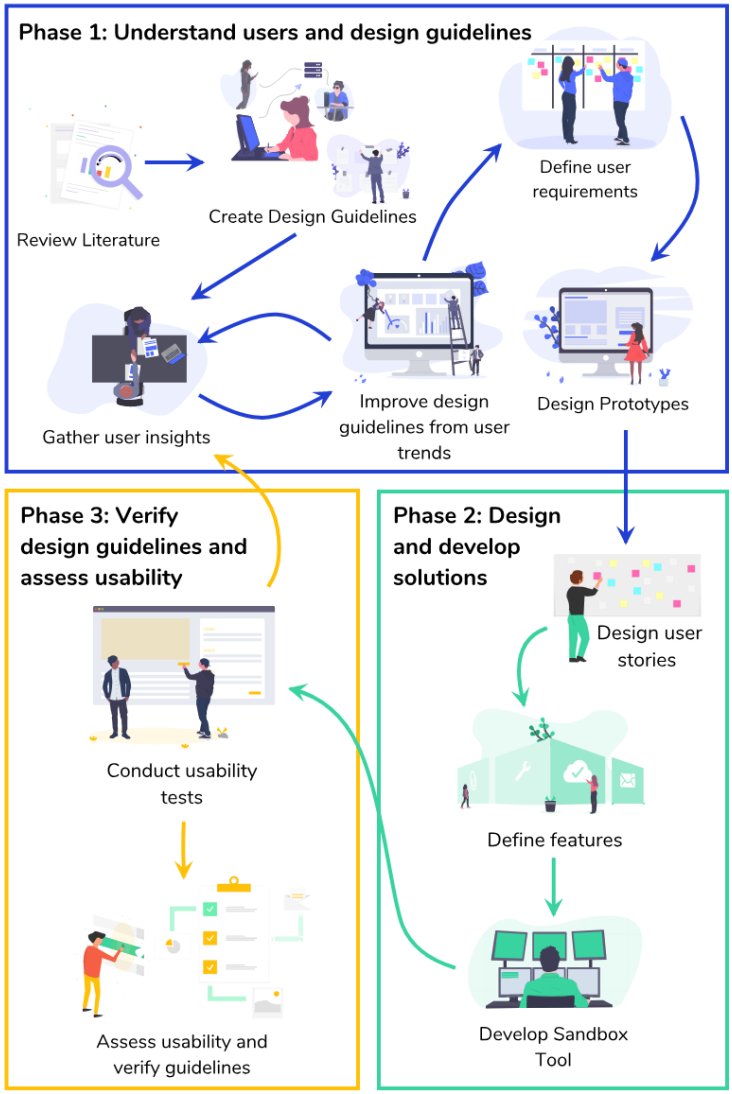

Figure 1: The iterative framework outlining the different phases in the design and development of TREX.

analysing all these insights. These steps are described in the succeeding sections.

2.1. Participants

We recruited, via purposive sampling, undergraduate and graduate computer science students with at least basic knowledge about ML and programming as participants. Participants must have at least a year of programming experience and a surface-level familiarity with ML concepts, especially Linear regression. A total of thirty-three (n = 33) participated on all phases. Since there were many phases in the study, one participant was present during one phase of the study.

Type	*n*	*Age*	*Years exp*	*ML exp*
		User Study Interviewees		
Beginner	3/10 (30.0%)	18-21	1-2y	1-3 m
Novice	5/10 (50.0%)	19-21	2-3y	3-6m
Experienced	2/10 (20.0%)	20-25	3-7y	1-4y
		Iteration 1		
Beginner	1/5 (20.0%)	21	2y	6m
Novice	2/5 (40.0%)	20-21	3y	1y
Experienced	2/5 (40.0%)	22-25	4-8y	2-4y
		Iteration 2		
Novice	7/10 (70.0%)	19-21	3-4y	3m
Experienced	3/10 (30.0%)	20-26	4-9y	1-4y
		Beta Version Testing		
Novice	2/3 (66.6%)	19-21	3-4y	3m-1y
Experienced	1/3 (33.3%)	27	11y	1-2y
		Iteration 3		
Novice	5/5 (100.0%)	19-21	3-4y	3m-1y

Table 1
Demographics of Participants of the User Study and Testing Iterations of the iML tool. The participants are a combination of undergraduate and graduate students from computer science courses with varying years of experience utilizing ML.

We refer to Table 1 for the over-all participant demographic data.

2.2. Exploration Phase

Before conceptualizing a prototype, we conducted a user study with ten (10) students familiar with programming ML models or who actively use them for work and projects. Each student was asked a set of questions relating to their experience with ML, their needs, wants, and frustrations, especially pain points when using tools that have ML capabilities such as RapidMiner, Weka, and Scikit Learn. At the end of the interview, each participant involved in the study were asked about their thoughts regarding a code-block based tool that would assist with creating Linear Regression models with visualizations of the ML pipeline and the code. Participants were then asked to complete a 4-point Likert scale survey —with choices from (1) strongly disagree until four (4) strongly agree —about the guidelines to determine how much they agree with them. After collecting user insights from the interview, we did an affinity mapping session to organize the user insights and resolve the main problem that the tool should help solve.

2.3. Development Phase

For designing the prototype, we used the insights organized from the affinity map to create user stories that will later define the features for the iML tool. Following the interface structure

of Scratch, the tool will have three sections in mind —the code block workspace, the code display, and a graph to help visualize the linear regression model created. Initially, we created a prototype using static code blocks that followed similar naming conventions to Scikit Learn's Linear Regression library. Each category of CodeBlocks was assigned specific colours to represent the phases of the ML pipeline. Three CodeBlocks categories were prepared —Data Preprocessing, Linear Regression, and Test Model. For the interactive prototype, we created it using Google Blockly, which is the same code block technology used for Scratch. We also used Python's Django web framework to allow Python code generated from the tool to be executed by the Python compiler. To visualize the Linear Regression model created, we used Chart.js to create an interactive scatter plot that updates as the model is updated. The scikit-learn Linear Regression library functions were used for making the Linear Regression model. Due to certain limitations with the libraries used, three additional CodeBlocks categories were added to the iML prototype. Libraries used for importing the necessary Python libraries required for the Linear Regression model, Variables contain CodeBlocks of predefined variables created for the iML tool, and Dataset for importing the online datasets recommended for use by Scikit Learn.

2.4. Test Phase

The prototypes created were tested iteratively with three (3) official iterations and a Beta version test. Each iteration was recorded using a screen recorder to record participant screen movement while a separate camera recorded participants' reactions. After each test, post-test surveys such as the System Usability Scale survey and a Guidelines Heuristic Review survey were issued before conducting a short post-test interview to understand the participant's experience using the prototype they were testing. For the first iteration, the static CodeBlocks were tested using a prototyping tool. Each participant was tasked to map CodeBlocks to their understanding of the code samples provided to them. The second iteration tested the first version of the iML tool without any graphical visualization of the Linear Regression model. Some insights from the first iteration were used to improve the CodeBlocks design for this iteration. Participants were tasked to create Linear Regression systems with the two datasets used for the tool. We focused on discoverability for the second iteration to understand user activity flows and determine common interactions participants would follow when using the CodeBlocks and when programming in familiar code environments. Using suggestions for improvement from the second iteration, we tested a Beta version with three (3) testers to determine significant improvements from the second iteration. Further comments and suggestions from this test were then used to improve the third iteration, which focuses on testing specific features of the iML tool. Four tasks were given to the testers to exhaustive explore features the iML tool has. Finally, we used this iteration to observe how users would create Linear Regression systems with a graphical visualization of the model.

2.5. Guidelines Assessment and Heuristic Review

To analyze the adherence of the iML tool's design and functionality to the derived design guidelines, we patterned a Guidelines and Heuristic Review similar to a UX Expert Review where each guideline has criteria to fulfil. The standards were derived from a combination

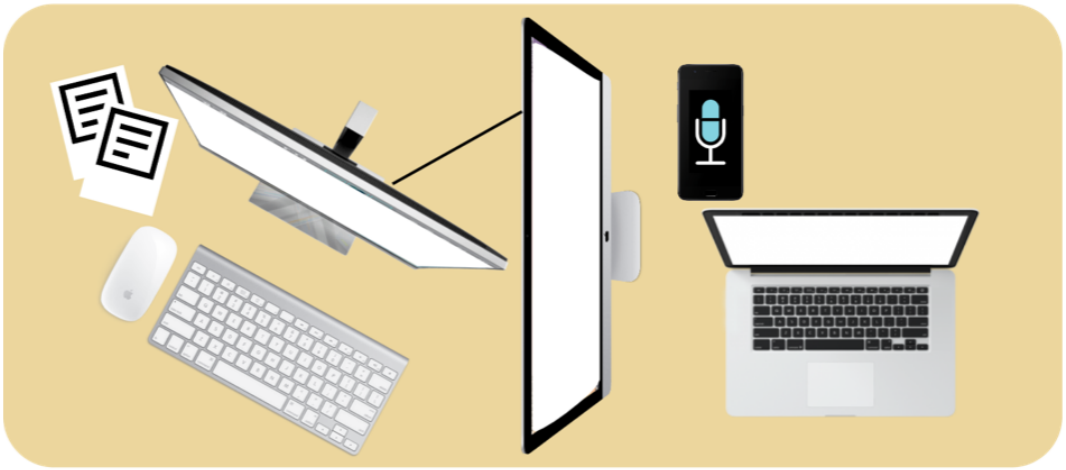

Figure 2: Setup for Usability Testing. Test Setup for all usability testing of prototypes. The participant was positioned in front of a laptop with the prototype available. The Facilitator sits beside the participant. Screen movement was recorded from the external monitor. The web camera of the desktop computer between the external monitor and the laptop was used to record the participant reactions and displayed the test tasks. In addition, a voice recorder was used to record extra audio for the test.

of Amershi's design considerations [3] and Nielsen's Heuristics [17]. The scoring metrics for the guidelines review are: (-1) Does not comply, (0) Kind of complies and (1) Complies. Next, we compute the average for each guideline. This is done by scoring each item over the total number of items per guidelines criteria. An acceptable guideline rating is set to 51%.

2.6. Data Analysis

To further analyse the participant feedback from the test, we issued a 105-word checklist with adjectives that help describe the participant's experience. They were asked to check as many words as they please, and from the selected words, they had to encircle the five most important words. Checked terms received one (1) score, and circled words were scored with a five (5). This was added to the overall word score for all participants. Word clouds were used to display the qualitative feedback from the word checklist to get a general understanding of the feeling of the tools.

3. Results

3.1. Agreement to Design Guidelines

Our guidelines derived from [3] were highly agreed upon by the user study participants as seen in table 2. Some participants disagreed with guideline 5 about error handling as they did not want to be "spoon-fed" when attempting to solve their error. However, after iteration 2 and 3

Design Guidelines Agreement				
Guideline	**1**	**2**	**3**	**4**
G1	0%	0%	10%	90%
G2	0%	0%	10%	90%
G3	0%	0%	70%	30%
G4	0%	0%	10%	90%
G5	0%	10%	30%	60%

Table 2
Agreement of the user study participants with the initial design guidelines derived from [3]. A Design Guidelines Survey —with a 4-point Likert scale: (1) Strongly Disagree; (4) Strongly Agree —was presented to each participant to identify the extent to which they agree with the need of the guidelines when designing iML related tools.

testings were completed, it was discovered that users still wanted explicit help while trying to recover from mistakes. The design guidelines used to aid with designing and developing the iML tool can be seen below:

- G1: The system should indicated to the user its state of change with every interaction.
- G2: The components of the interface should be laid out in an organized manner and explicitly.
- G3: The visualizations should be simple enough to understand. They should be not over-load the user.
- G4: The intent of the system should be clear to the user upon interacting with the system.
- G5: Indicate explicitly when and why errors occur and how users can recover from them.

The current iML tool created mostly adhered to guidelines 1-4 related to system interaction and visualization. Guideline 5 had mixed results from the iteration 3 participants as some participants are still looking for more descriptive errors.

3.2. Features of the TREX Tool

During each iteration, specific limitations and problems were discovered while the user was testing the system. The first limitation found was during the first iteration testing using the iteration of design made using Gravit. Some users mentioned that blocks were missing that was needed to create a proper linear regression system. The blocks that were mainly recommended were the train and test split block and adequate testing blocks required to determine the accuracy and behaviour of the regression system created. These blocks were later added in the second iteration of the system. The train and test split block were responsible for splitting the current dataset into X and Y train and test sets. In addition, respective output blocks were added to check the coefficients, mean squared error, and R squared score. The tool uses the Scikit-learn Metrics library to compute these output values.

The second iteration brought about the first iteration with a functional prototype used during the iteration testing. However, there were several limitations to using the Blockly environment. The limitations were the lack of regression visualization and error handling during this iteration and missing CodeBlocks needed to create or customize the system, with each being addressed in

the next iteration. The regression visualization was added as part of the three (3) main sections of the tool's dashboard. Extensive error handling was also implemented to catch errors using the Python engine, and custom errors are triggered through misuse of CodeBlocks in the Blockly environment. The users also recommended CodeBlocks such as the import libraries block responsible for importing all necessary libraries needed to run the system and implementing a print block that could display specific values while testing.

TREX is designed to be a web-based interactive tool that aims to provide the user with a sandbox environment for testing and creating Linear Regression ML Systems. All functions of the tool can be accessed in the dashboard, which is separated into three sections as seen in figure 3, each with a specific goal in mind. The three main sections are the Sandbox Environment section, Code Translation and Output section, and the Regression Visualization section. TREX uses CodeBlocks, which are connected to code segments similar to functions found in ML systems. We decided to create the system using a full-stack Python setup since Scikit-learn's Linear Regression library runs linear regression functions. This also provides easy access to regression visualization libraries such as `matplotlib` for Python. When the system is compiled and run, the results are displayed on the output section of the dashboard. TREX uses the native error handling of the library along with custom error handling to address CodeBlocks limitations such as incompatibility.

TREX is the result of the final iteration testing and was designed to be a web-based interactive tool that aims to provide the user with a sandbox environment for testing and create Linear Regression Machine Learning Systems that can be used on various datasets. All tool functions can be accessed in the main dashboard that is separated into three main sections, each with a specific goal in mind. The three main sections are the Sandbox Environment section, Code Translation and Output section, and the Regression Visualization section. The sandbox section includes the environment where the user can create their linear regression system using the various CodeBlocks provided by the tool. The environment consists of two major components the toolbox and the workspace. The toolbox holds and stores all CodeBlocks needed to create your linear regression system, while the workspace is used for arranging the blocks that represent the code for the system. The user will be able to drag and drop CodeBlocks from the toolbox onto the workspace, which can then be clipped or snapped to other CodeBlocks and edit the various parameters found on the CodeBlocks themselves. In addition, the user may delete CodeBlocks by dragging them into the bin icon on the lower-left side of the workspace, which can also view previously deleted CodeBlocks when left-clicked.

The code translation and output section are responsible for displaying the translated code from the CodeBlocks arranged in the workspace and the respective outputs when run. The View CodeBlockss button will parse each CodeBlocks in the workspace from top to bottom in a linear fashion and displayed in the code display on the left side of the section. The Run CodeBlocks button will then run the parsed CodeBlocks using a Python compiler with each output display on the output display on the right side of the section. The tool converts the visual CodeBlocks into actual code snippets, which are then arranged based on the order of the blocks. Because of this, all variables are first initialized to a default value to ensure there will be no null values. The code snippets will then be concatenated into a formatted string displayed on the output display section.

The regression visualization section is responsible for displaying the predicted target data

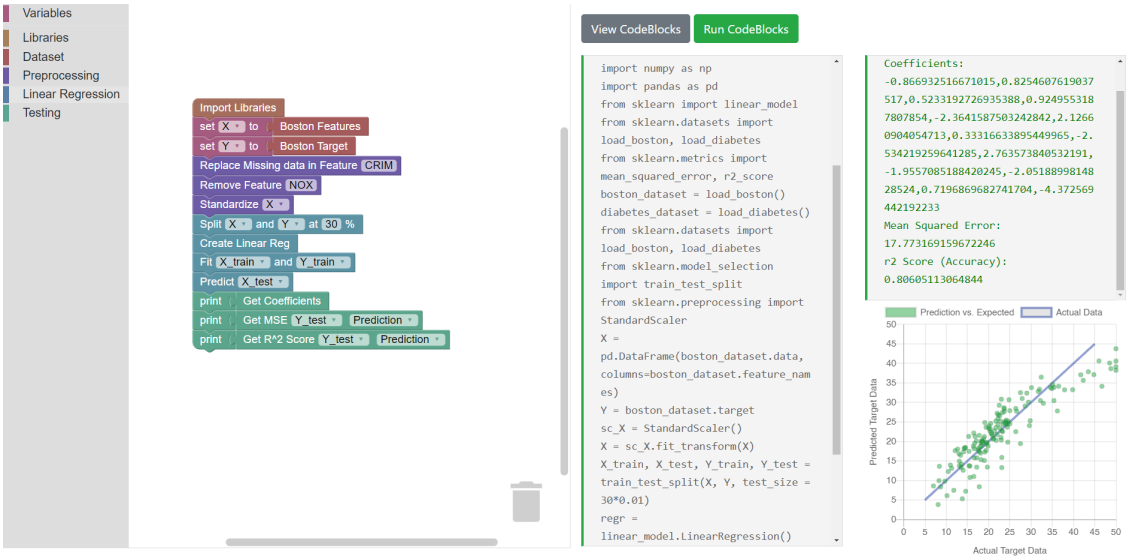

Figure 3: The TREX General Interface. It is composed of three main parts: (i) The Sandbox section, (ii) The Code Translation and Output Section and (iii) The Regression Visualisation section. There is a toolbox on the left where users can pick CodeBlocks from the different categories presented. There is a workspace section where users can drag into the CodeBlocks picked from the toolbox. The CodeBlocks can be rearranged and have values keyed in. A code interface section generates the equivalent python code of the CodeBlocks that they have selected. This helps the user translate the visuals into runnable code they can execute should they wish to port into different platforms. The logs messages are also visible (in green) that shows the possible console output of the code. Lastly, a visualization chart can be seen, which helps the user interpret the recently completed regression experiment.

compared to the actual target data, and the diagonal line descriptor of the system created. The tool first selects the two data arrays it will be using, a combination of either the train or test set of Y and the prediction based on the shape of the prediction and the train or test set(if they match). It then plots a line based on the regression equation of the given target feature. Finally, the tool uses the Chart.js library to create a mixed linear and scatter graph to show the data and the plotted line.

3.3. Success and Completion Rates

For iteration 2, only two (2) tasks that create Linear Regression models were given. One involved coding or sorting code cells in a Google Colaboratory Notebook, and the other used the iML tool. Each of these tasks was counterbalanced. For the notebook task, all participants were able to complete it as the coding environment was something familiar for the participants to use. On the other hand, when using the iML tool, 30% of the participants failed the task since some were unfamiliar with using CodeBlocks and the overall environment was a new approach for them to create Linear Regression models. Thus, the participants who failed the iML tool task mainly received the tool as their first task.

Iteration 3 had varied tasks that involved improving the accuracy of the Linear Regression

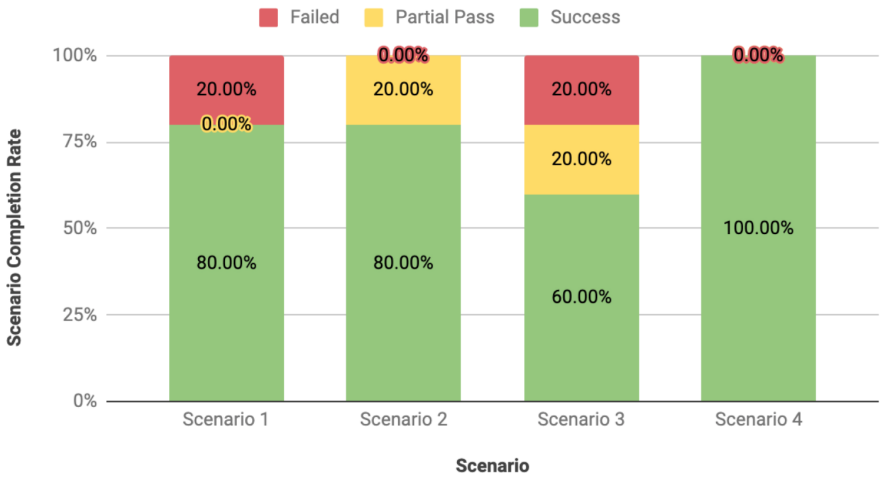

Figure 4: Success rates in tasks used during Iteration 3 testing. There were four (4) scenarios that participants had to perform. Therefore, it can be seen that Scenario 4 was completed, and their apparent difficulties were observed in the other scenarios.

model using two (2) different datasets for scenarios 1 and 4 and tasks that would align the visualization of the tool around the zero-point for scenarios 2 and 3. Overall most participants were able to complete all tasks, as seen in figure 4 with scenario 4 receiving a 100% completion rate. Since all these tasks were counterbalanced, by the time scenario 4 was accomplished, the participants had sufficiently explored the iML tool to complete each task. Scenarios 2 and 3 have partial passes, as indicated by the yellow portions of figure 4, this means that the participants were able to partially complete building their Linear Regression model but decided to skip the task since they could not identify what set of blocks to use next.

3.4. User Experience and Usability Evaluation

We conducted two usability tests with the iML tool where we used a Systems Usability Scale survey (SUS) to measure the overall usability performance of the iML tool. According to the word checklists in the second iteration, most testers found the tool new and innovative. They also thought it was time-saving, simplistic, and satisfying to use. However, they also said the tool seemed confusing, complicated and stressful since it was a new sandbox iML tool that encouraged exploring the environment with minimal assistance. The overall SUS score for Iteration 2 can be seen in table 3. The average SUS score is 63.00, which is rated Grade D - Poor.

Each iteration of the prototype was tested by at least five (5) to ten (10) ML users with varying skill levels, where two (2) of which identified themselves as novice users. For iteration 1, all novices were intuitively following a pattern based on the ML Pipeline: Data Preprocessing, Model Training, and Model Testing. They were also consistent when ordering the Fit Data and Predict blocks. They placed the create-model block outside the model bracket, which was designed to encompass the block. Novice users were observed to be experimental when using data preprocessing & testing blocks. However, they found the tasks manageable. They also noted that organizing different ML phases by colour made it easier but were confused with the

data preprocessing blocks.

For iteration 2, most users enjoyed the block connecting sound. It reassured them of their progress and cited how the blocks and categories followed the ML pipeline they identified. However, some users found the tool stressful, causing them to skip tasks after misunderstanding how to use it, like stacking blocks horizontally rather than top to bottom. Some struggled to figure out the purpose of the "Prediction" block due to its colour being similar to the native Print block. Some users felt that the tool lacked flexibility when displaying outputs and creating & renaming variables, while others wanted line numbers with good error messages similar to traditional IDE's. The SUS survey score from this iteration was an average of 63.00, which falls into Grade D - Poor, seen in table 3. This means the tool failed in terms of usability. This may be due to the errors participants encountered and their time adjusting and exploring the CodeBlocks.

For iteration 3, five (5) users were selected for the final version of the prototype. Most users enjoyed using TREX CodeBlocks to make base linear regression code due to its intuitiveness. However, some participants thought it was vague and frustrating to use due to misunderstanding the intent of error messages, which may lead to confusion. The previous pain-point of connecting the "Prediction" block to the "Print" block led to the design decision of hard-coding the initialization of the Prediction variable instead of maintaining the male jigsaw block shape that is similar in colour to the native print block. The testing blocks for the final version of the prototype, were also recoloured to green to be automatically associate as output value blocks. The SUS survey score from this iteration received an average score of 80.50, which falls into a Grade A - Excellent, seen in table 4. This means the tool is usable and functions towards user expectations. Between Iteration 2 and 3, there is a significant increase in the SUS score from 63 to 80.50.

4. Discussion

As iML platforms are becoming more in demand, it is essential to understand the human factors that influence the design of these systems. We designed and developed TREX: A Toolbox for Regression Experiments following the design methodology from [1]. Through iterative prototyping and testing, we echo the findings of [2]. Novice programmers tend to be less confident than their experienced counterparts. The behaviour they exhibited included: (1) exploring first their environment before building their solution as seen from our regression experiments, (2) editing their existing CodeBlocks based on their understanding of the ML pipeline, and finally (3) testing their assumptions and how it affects their model. With limited participants, our findings are yet to be verified through extensive testing. Nevertheless, these findings emphasize the dynamic demand for better design in iML systems. With further research, we can still validate whether these behaviours are observable in other ML activities beyond regression experiments.

Participant	SUS Score
1	57.50
2	77.50
3	37.50
4	72.50
5	92.50
6	47.50
7	72.50
8	45.00
9	67.50
10	60.00
Average	63.00

Table 3
Average SUS survey score from 10 participants that tested iteration 2 of the iML tool. An average SUS score of 63.00 is below the average industry recognized SUS score of 68.00 giving the overall iteration 2 a D grade according to the SUS grading curve mentioned by [18]. This indicates that the iML tool is not perceived to be highly usable by participants particularly due to the lack of graphical visualization to observe the behavior of their Linear Regression experiment.

Participant	SUS Score
1	85.00
2	85.00
3	85.00
4	72.50
5	75.00
Average	80.50

Table 4
Average SUS survey score from 5 participants that tested iteration 3 of the iML tool. Only 5 participants were tested due to the time constraints and limited availability of students that fit the study's criteria for novice participants. Despite testing with fewer participants, a SUS score is reliable for uncovering differences with smaller samples sizes since reliability and sample size are unrelated [18]. According to the SUS grading curve, an average SUS survey score of 80.5 gives iteration 3 a Grade A indicating this iteration of the iML tool falls in the top 10% of SUS scores where systems at this level are highly perceived to be usable. This score was attained after introducing the graphical visualizations that helped users observe the behavior of their Linear Regression model in response to their adjustments in codeblocks and parameters.

User activity flow patterns were observed in iteration 2 since the tests' focus was on the discoverability of the iML tool's features. The typical user activity flow pattern observed for the novice users was Explore-Edit-Guess, as these users took the time to explore the sandbox iML environment before forming the Linear Regression model. Experienced users were observed to follow Edit-Explore-Guess as they are familiar with the ML terminologies. They would assemble the model and edit it as they explored the tool. The patterns were decided by observing screen

recordings of iteration 2 and selecting the order of activity done. An interrater reliability score was used to determine the habits; 60% was the acceptable agreement level. The agreement of Explore is 63.33%, Edit is 56.67%, and Guess is 73.33%.

5. Limitation and Future Work

One of the areas that can be explored in the future is the potential use of more complex machine learning algorithms with visual programming or the concept of a sandbox environment. With the amount of machine learning libraries found that is being supported in Python, each function can be translated into specific CodeBlocks. However, there are limitations with the current iteration of the tool. One of these limitations is the current implementation of the datasets being used with the tool. The current iteration uses premade datasets retrieved from the Scikit-learn sample datasets library due to how they are structured and the overall cleanliness of the dataset. Some CodeBlocks assume that the dataset being used on the system are structured in a specific way before they are processed and fitted onto the machine learning algorithm. If the dataset does not follow this particular structure, the tool may wrongly interpret the data being given to it. This may result in an unstable system or falsely interpreted data results. In addition, better ways of gaining user activity flow can be explored in the future by using system logs for more accurate tracking of user activity flows. It may make patterns more evident compared to annotation and observation alone. For the guidelines, future work could include exploring if the design guidelines discovered from our prototype can be applied to other machine learning algorithms significantly how they may change depending on the objective and use of the algorithm.

6. Conclusion

As interactive machine learning platforms are becoming more in demand, it is essential to understand the human factors that influence the design of these systems. We echo the findings of [2] and confirm the guidelines we derived following the methodologies of [3] that novice programmers are less more confident than their experienced counterparts. Furthermore, as seen from our regression experiments, we found that they tend to explore their environment before diving into building their solution. From these observations, we validated our guidelines through a heuristic review and software usability scores. These findings and implications emphasize the dynamic and growing demand for better design in interactive machine learning systems. With further research, we can still investigate and validate whether these behaviours are observable in other machine learning activities beyond regression experiments.

References

[1] G. Nodalo, J. M. Santiago III, J. Valenzuela, J. A. Deja, On building design guidelines for an interactive machine learning sandbox application, in: Proceedings of the 5th International ACM In-Cooperation HCI and UX Conference, ACM, 2019, pp. 70–77.

[2] A. Sarkar, M. Jamnik, A. F. Blackwell, M. Spott, Interactive visual machine learning in spreadsheets, in: Visual Languages and Human-Centric Computing (VL/HCC), 2015 IEEE Symposium on, IEEE, 2015, pp. 159–163.
[3] S. Amershi, Designing for effective end-user interaction with machine learning, in: Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology, ACM, 2011, pp. 47–50.
[4] S. Amershi, J. Fogarty, D. Weld, Regroup: Interactive machine learning for on-demand group creation in social networks, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2012, pp. 21–30.
[5] S. Das, D. Cashman, R. Chang, A. Endert, Beames: Interactive multi-model steering, selection, and inspection for regression tasks., IEEE computer graphics and applications (2019).
[6] J. Zhao, M. Karimzadeh, A. Masjedi, T. Wang, X. Zhang, M. M. Crawford, D. S. Ebert, Featureexplorer: Interactive feature selection and exploration of regression models for hyperspectral images, arXiv preprint arXiv:1908.00671 (2019).
[7] C. A. Shaffer, M. L. Cooper, A. J. D. Alon, M. Akbar, M. Stewart, S. Ponce, S. H. Edwards, Algorithm visualization: The state of the field, ACM Transactions on Computing Education (TOCE) 10 (2010) 9.
[8] E. Vrachnos, A. Jimoyiannis, Dave: A dynamic algorithm visualization environment for novice learners, in: Advanced Learning Technologies, 2008. ICALT'08. Eighth IEEE International Conference on, IEEE, 2008, pp. 319–323.
[9] Z. Hussain, W. Slany, A. Holzinger, Current state of agile user-centered design: A survey, in: Symposium of the Austrian HCI and Usability Engineering Group, Springer, 2009, pp. 416–427.
[10] F. Bernardo, M. Zbyszynski, R. Fiebrink, M. Grierson, et al., Interactive machine learning for end-user innovation, American Association for Artificial Intelligence (AAAI), 2016.
[11] K. Peffers, T. Tuunanen, M. A. Rothenberger, S. Chatterjee, A design science research methodology for information systems research, Journal of management information systems 24 (2007) 45–77.
[12] J. A. Deja, K. G. Chan, M. A. Dancel, A. V. Gonzales, J. P. Tobias, Flow: A musical composition tool using gesture interactions and musical metacreation, in: International Conference on Human-Computer Interaction, Springer, 2018, pp. 169–176.
[13] J. A. Deja, P. Arceo, D. G. David, P. L. Gan, R. C. Roque, Myosl: A framework for measuring usability of two-arm gestural electromyography for sign language, in: International Conference on Universal Access in Human-Computer Interaction, Springer, 2018, pp. 146–159.
[14] K. G. Chan, J. A. Deja, J. P. Tobias, A. V. Gonzales, M. A. Dancel, Applying user-centered techniques in the design of a usable mobile musical composition tool, in: Proceedings of the 5th International ACM In-Cooperation HCI and UX Conference, 2019, pp. 152–159.
[15] J. A. Deja, A. Dela Torre, H. J. Lee, J. F. Ciriaco IV, C. M. Eroles, Vitune: A visualizer tool to allow the deaf and hard of hearing to see music with their eyes, in: Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems, 2020, pp. 1–8.
[16] T. J. S. Maria, G. R. Dizon, V. A. Esquivel, J. A. Deja, U. Chua, Designing grit: Discovering features towards supporting novice programmer devops integration, in: Proceedings of

the 2020 Symposium on Emerging Research from Asia and on Asian Contexts and Cultures, 2020, pp. 41–44.
[17] J. Nielsen, Finding usability problems through heuristic evaluation, in: Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, 1992, pp. 373–380.
[18] J. Sauro, Measuring usability with the system usability scale (sus), 2011. URL: https://measuringu.com/sus/.