Abstract—Although artificial intelligence (AI) is increasingly involved in everyday technologies, AI literacy amongst the general public remains low. Thus many AI education curricula for people without prior AI experience have emerged, often utilizing graphical programming languages for hands-on projects. However, there are no tools that assist educators in evaluating learners’ AI projects or provide learners with contemporaneous feedback on their work. We developed LevelUp, an automatic code analysis tool to support these educators and learners. LevelUp is built into a block-based programming platform and gives users continuous feedback on their text classification projects. We evaluated the tool with a crossover user study where participants completed two text classification projects, once where they could access LevelUp and once when they could not. To measure the tool’s impact on participants’ understanding of text classification, we used pre-post assessments and graded both of their projects against LevelUp’s rubric. We saw a significant improvement in the quality of participants’ projects after they used the tool. We also used questionnaires to solicit participants’ feedback. Overall, participants said that LevelUp was useful and intuitive. Our investigation of this novel automatic assessment tool can inform the design of future code analysis tools for AI education.

Index Terms—Computer science education, machine learning, automatic assessment tools, visual programming

I. INTRODUCTION

As artificially intelligent (AI) technologies become more integrated into society, citizens will need to possess digital skills in order to thrive [1], [2]. A study done by Juniper Research [3] forecasted that 70 million U.S. households, or 55% of the total households in the U.S., will have installed a smart speaker by the end of 2022. Despite the increasing prevalence of these artificial intelligence (AI) devices, most people lack a baseline knowledge of AI and are therefore unable to grasp how these technologies work [4]. To support universal AI literacy, researchers and educators have created many AI education curricula and initiatives, especially over the last five years [5], [6], targeting the general public and K-12 students.

In the field of computational thinking, internationally-recognized standards equip educators with a yardstick by which to measure student achievement [7], [8]. Plus, automatic assessment tools built on top of these standards have supported both educators and learners in better evaluating their computational thinking skills [9]–[12]. However, assessment remains a significant challenge in AI education research. Although prior AI education work often includes hands-on projects [13], [14], universal project standards and automatic assessment tools have not yet been built for AI education.

In this paper we describe LevelUp, an automatic assessment tool that analyzes block-based, supervised machine learning (ML) projects. We designed LevelUp to be used in conjunction with a curriculum, How to Train Your Robot, that gives students a hands-on AI learning experience [25]. In this curriculum, students learn the AI concepts behind a text classification extension in a block-based programming platform [24]. They then complete coding projects using this extension to further their knowledge. We evaluated LevelUp using a crossover study where participants completed two open-ended text classification projects, once with the tool and once without it. We hypothesized that:

1) The quality of users’ projects would improve when they had access to LevelUp
2) Users would say the tool was helpful and its interface was easy to understand

We believe this tool will be a beneficial addition to ML education curricula and programming interfaces.
II. BACKGROUND

A. Machine Learning Tools for Students

Many tools and platforms have been developed to teach the general public, and particularly K-12 students, about ML. Tools, such as Google’s Teachable Machine [15], Personal Audio Classifier [16], Personal Image Classifier [17], and PlushPal [18] have interfaces that make it easy to create image, text, audio, and pose classification models without writing a line of code. Platforms like Cognimates [19], ML4Kids [20], PoseBlocks [21], AI extensions for Snap! [22], LearningML [23], and the Scratch text classification extension [24] allow students to create ML models and then use them in block-based programming projects. Through hands-on practice with these platforms, students are better able to understand the theory and application of AI.

In previously published ML curricula for K-12 students, researchers evaluated students’ projects using criteria such as the quality of the problem selection [25], the quality of training data [18], [25], [26], the quality of the model after students’ test it [18], [26], and the complexity of students’ programs [25]. Although existing curricula and tools include lessons and tutorials that teach students how to build functional ML models, none provide tools for automatic assessment to support teachers and students in evaluating their projects.

B. Automatic Code Analysis Tools

Automatic assessment is a common approach used to benchmark work. We see this in writing tools, such as Grammarly, integrated development environments, with plug-ins like ESLint, and hint generation for Python programming through ITAP [27]. Many curricula that utilize block-based programming interfaces to teach computational thinking and AI leverage open-ended projects rather than close-ended problems. Since there is no single, correct solution for open-ended projects, graders assess this work by judging the code against a rubric [11]. Tools like Dr. Scratch [9], Ninja Code Village [10], Scrape [12], and Code Master [11] are applications that researchers have used to automatically analyze and evaluate block-based programming projects against custom rubrics. We created LevelUp because, although all of these tools have been proven to effectively support learners, none of them are integrated with programming platforms, provide continuous feedback as students work, or assess competencies related to AI literacy.

LevelUp is most similar to Dr. Scratch’s code analysis tool. Dr. Scratch works by allowing users to upload their Scratch project into their web interface. The tool analyzes the project and outputs a score based on seven categories: Flow Control, Data Representation, Abstraction, User Interactivity, Synchronization, Parallelism, and Logic. Similarly, our tool provides students with a percentage score based on a multitude of factors including the complexity of their code and the quality of their ML models. In their evaluative user study, Moreno-León et al. [9] found that Dr. Scratch had a significant, positive impact on newer programmers. Because of the similarities between Dr. Scratch and our tool, we hypothesize that LevelUp will have a comparable, positive impact on users.

III. SYSTEM DESIGN

Both the programming platform and LevelUp were developed for a specific AI curriculum [24] and intended to be used in a classroom setting. By having LevelUp exist right inside of the platform, we created a streamlined process where students can easily go between receiving feedback and changing their code. The seamlessness of this built-in feedback tool allows users to improve their code without disrupting their creative process.

Users can reach LevelUp by simply clicking on a tab labeled “Progress” near the top of the platform (Fig. 1). After clicking on this tab, the tool calculates their progress score, and they can see what they have done well and how they might improve. There are three main parts to the interface: (1) further suggestions, (2) completed suggestions, and (3) the progress bar.

Progress Bar: The progress bar is displayed at the top of LevelUp and provides a visual representation of the user’s current AI progress. It takes into account the user’s current progress score, calculated based on their code and text classifier data, and updates the progress bar visual accordingly.

Completed Suggestions: To the left of the interface is a box containing formalized improvements that increase the user’s progress score once completed. Suggestions are grayed out until the user fulfills them. Then, that item is bolded, and the progress score updates accordingly. Each improvement adds a certain percentage to the user’s progress score (Table I). The successful completion of all the improvements in bold within Table I is the only way to reach a progress score of 100%. New items are added in the Further Suggestions box as users continue to complete the other formal suggestions. For example, after users have created two class labels, the tool suggests adding a third label.

Further Suggestions: On the right of the interface is a box containing suggestions for improvement users can make to advance their use of the text classification tool. Some suggestions include “You need at least five examples per class to have an accurate classifier,” “Try embedding conditionals to make your code more complex,” and “Try adding a variety of text classification blocks to increase your progress.”

<table>
<thead>
<tr>
<th>Item</th>
<th>Added Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Have more than two text classification classes</td>
<td>30%</td>
</tr>
<tr>
<td>~ Have two or less text classification classes</td>
<td>20%</td>
</tr>
<tr>
<td>Have at least five examples per text classification class</td>
<td>15%</td>
</tr>
<tr>
<td>Have balanced text classification classes</td>
<td>10%</td>
</tr>
<tr>
<td>Use at least four text classification blocks</td>
<td>30%</td>
</tr>
<tr>
<td>~ Use three text classification blocks</td>
<td>20%</td>
</tr>
<tr>
<td>~ Use two text classification blocks</td>
<td>10%</td>
</tr>
<tr>
<td>Use embedded conditionals</td>
<td>15%</td>
</tr>
</tbody>
</table>
TABLE II  
GENDER AND AGE OF USER STUDY PARTICIPANTS

<table>
<thead>
<tr>
<th>Condition</th>
<th>Total</th>
<th>Female</th>
<th>Male</th>
<th>Other</th>
<th>18-24</th>
<th>25+</th>
</tr>
</thead>
<tbody>
<tr>
<td>LevelUp-P1</td>
<td>13</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>LevelUp-P2</td>
<td>12</td>
<td>8</td>
<td>4</td>
<td>0</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>All</td>
<td>25</td>
<td>16</td>
<td>7</td>
<td>2</td>
<td>22</td>
<td>3</td>
</tr>
</tbody>
</table>

IV. METHODOLOGY

We measured the impact that LevelUp had on participants’ developing conceptual understanding and construction of text classifiers. We collected pre-test scores, post-test scores, participants’ self-assessments of their projects, and project rubric scores. We also analyzed how participants used LevelUp and asked for open-ended feedback to better understand how we might improve the tool.

Participants. A total of 25 university affiliates participated in the study. Sixteen (16) identified as female, seven as male, and two as other genders. Their computer science backgrounds ranged from having almost no experience with programming to having taken a few CS courses. We accepted participants who had taken programming courses before but did not accept any who had formally learned about AI because we wanted to study people without prior experience.

We split participants into two conditions, counterbalancing their gender, age, and technical experience (Table II). Participants in the LevelUp-P1 condition completed their first project with the help of LevelUp and the second project without it. In LevelUp-P2, participants completed the first project without LevelUp and the second project with it.

A. Procedure

1. Pre-Test. At the beginning of the study, all participants completed a thirteen question, multiple-choice pre-test about concepts in AI, ML, and text classification. We used this assessment to establish a baseline on how much AI knowledge participants possessed before the study.

2. Presentation. Then, a researcher delivered a presentation introducing participants to word vectors and the KNN algorithm, the algorithm behind the text classification tool. The presentation covered all of the concepts we assessed in the pre-test plus instructions on how to build a text classifier in the block-based programming interface. The presentation included a walkthrough of four key programming practices for building text classifiers (the same ones assessed by LevelUp): Creating at least two labels for the classifier, Using multiple examples for each label, Balancing the number of examples in each label, and Using different kinds of text classification blocks (e.g., using matches blocks, confidence blocks, etc.). Participants had access to the presentation slides, and therefore the list of good programming practices during both projects.

3. Projects. Participants completed two projects during the course of the study. The theme of the first project was a “Spam Filter”. We gave students starter code as well as two (empty) starter classes for their text classifier: “Spam” and “Not Spam” (Fig. 2). The second project had the theme of “Sentiment Analysis” along with starter code and “Positive” and “Negative” as the two text classification classes. Participants had 15 minutes to complete each project by adding additional code and filling out the examples for the classes. After each project, participants completed short reflection questions to describe the goal of their code as well as how much they considered the four text classifier programming practices.

After completing a project with LevelUp, participants answered questions about how usable they found the tool, what they liked about it, and what they would change to make it more useful or easier to understand.

Post-Test. After completing their first project, participants finished a post-test that had the same multiple-choice questions as the pre-test. We predicted that both the presentation and the hands-on project would contribute to participants’ learning. We were not sure if LevelUp would impact participants’ post-test scores, which is why we had them do the post-test after their first project.

V. RESULTS

A. Pre and Post-Test

Participants’ performance on the pre and post-tests of conceptual understanding showed that participants had a better grasp of AI concepts during the post-test and that groups came and left with about the same level of understanding.

Due to the small sample size and non-normality of the data, we used non-parametric statistical tests to analyze participants’ pre and post-test scores. On the pre-test, the average score was 73.2%, sd=0.13, and the median was 76.9%. On the post-test, the average score was 80.06%, sd=0.09, median=84.6%). A Wilcoxon rank test with Pearson’s $r$ determined that the difference between overall pre and post-test scores was significant ($z=-2.5$, $p=0.011$, $r=0.51$).

We used a Mann Whitney U test with Pearson’s $r$ to confirm that the pre-test scores for LevelUp-P1 (Mean=73.4%, sd=0.13, Median=76.9%) were very similar to those from LevelUp-P2 (Mean=73.1%, sd=0.13, Median=76.9%, U=75,
TABLE III
PRE-TEST AND POST-TEST SCORES

<table>
<thead>
<tr>
<th>Condition</th>
<th>Pre-Test</th>
<th></th>
<th></th>
<th>Post-Test</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N</td>
<td>Mean</td>
<td>sd</td>
<td>Mean</td>
<td>sd</td>
<td>Mean</td>
</tr>
<tr>
<td>LevelUp-P1</td>
<td>13</td>
<td>73.4%</td>
<td>0.13</td>
<td>77.5%</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>LevelUp-P2</td>
<td>12</td>
<td>73.1%</td>
<td>0.13</td>
<td>84.0%</td>
<td>0.07</td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>25</td>
<td>73.2%</td>
<td>0.13</td>
<td>80.6%</td>
<td>0.09</td>
<td></td>
</tr>
</tbody>
</table>

p=0.89, r=0.027). As Table III shows, on the post-test there was a slight though not significant difference between LevelUp-P1 (Mean=77.5%, sd=0.10, Median=76.9%) and LevelUp-P2 (Mean=84.0%, sd=0.07, Median=84.6%, U=48, p=0.11, r=0.32). The similarity in learning gains for the two conditions is not surprising since they had access to the same information and the same amount of practice.

B. Final Project Scores

We recorded final project scores using LevelUp’s automatic rubric. Participants’ final project scores revealed that having access to LevelUp did indeed improve the quality of their projects. LevelUp-P1’s first project scores, when they did have access to LevelUp, were much higher than LevelUp-P2’s first projects. When LevelUp-P2 got access to LevelUp on their second project, their scores reached and surpassed the scores of LevelUp-P1.

Since the scores were not normally distributed and the sample size was small, we used a non-parametric Mann Whitney U test with Pearson’s r to determine if there was a significant difference in project scores between conditions. As shown in Table IV, there was a significant difference between LevelUp-P1’s first project scores (with LevelUp, Mean=75.8%, sd=0.29, Median=80.0%) and LevelUp-P2’s first project scores (without LevelUp, Mean=52.1%, sd=0.30, Median=65.0%, U=38.5, p=0.034, r=0.42). There was also a significant difference between LevelUp-P1’s second project scores (without LevelUp, Mean=76.5%, sd=0.23, Median=90.0%) and LevelUp-P2’s scores (with LevelUp, Mean=92.1%, sd=0.16, Median=100.0%, U=38, p=0.032, r=0.42).

We used Wilcoxon Signed-Rank Test and Pearson’s r to compare participants’ first and second projects. There was no significant difference between LevelUp-P1’s first and second project scores (z=-0.20, W=36.5, W critical at 13, r=0.054), while in LevelUp-P2 there was a large significant difference between the first and second project (z=-3.10, W=0, W critical at 13, r=0.88).

TABLE IV
FINAL PROJECT SCORES

<table>
<thead>
<tr>
<th>Condition</th>
<th>Project 1</th>
<th></th>
<th></th>
<th>Project 2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N</td>
<td>Mean</td>
<td>sd</td>
<td>Mean</td>
<td>sd</td>
<td>Mean</td>
</tr>
<tr>
<td>LevelUp-P1</td>
<td>13</td>
<td>75.8%</td>
<td>0.29</td>
<td>80.0%</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>LevelUp-P2</td>
<td>12</td>
<td>52.1%</td>
<td>0.30</td>
<td>65.0%</td>
<td>0.20</td>
<td></td>
</tr>
</tbody>
</table>

B. Final Project Scores

We recorded final project scores using LevelUp’s automatic rubric. Participants’ final project scores revealed that having access to LevelUp did indeed improve the quality of their projects. LevelUp-P1’s first project scores, when they did have access to LevelUp, were much higher than LevelUp-P2’s first projects. When LevelUp-P2 got access to LevelUp on their second project, their scores reached and surpassed the scores of LevelUp-P1.

Since the scores were not normally distributed and the sample size was small, we used a non-parametric Mann Whitney U test with Pearson’s r to determine if there was a significant difference in project scores between conditions. As shown in Table IV, there was a significant difference between LevelUp-P1’s first project scores (with LevelUp, Mean=75.8%, sd=0.29, Median=80.0%) and LevelUp-P2’s first project scores (without LevelUp, Mean=52.1%, sd=0.30, Median=65.0%, U=38.5, p=0.034, r=0.42). There was also a significant difference between LevelUp-P1’s second project scores (without LevelUp, Mean=76.5%, sd=0.23, Median=90.0%) and LevelUp-P2’s scores (with LevelUp, Mean=92.1%, sd=0.16, Median=100.0%, U=38, p=0.032, r=0.42).

We used Wilcoxon Signed-Rank Test and Pearson’s r to compare participants’ first and second projects. There was no significant difference between LevelUp-P1’s first and second project scores (z=-0.20, W=36.5, W critical at 13, r=0.054), while in LevelUp-P2 there was a large significant difference between the first and second project (z=-3.10, W=0, W critical at 13, r=0.88).

C. Self-Assessment of Programming Practices

After each project, we asked participants to self-assess their use of good programming practices. We asked how much they prioritized: “Creating at least two labels for the classifier”, “Using multiple examples for each label”, “Balancing the number of examples in each label”, and “Using different kinds of text classification blocks” (Fig. 3). For the most part, participants seemed to remember the programming practices well.

When working on the first project, participants mainly focused on creating at least two labels for the classifier, using multiple examples for each label, and balancing the examples. LevelUp-P1 also highly considered using a variety of text classification blocks on their first project, but LevelUp-P2 did not (without LevelUp, Mean=1.83, sd=0.72, Median=2). However, on their second project, LevelUp-P2’s consideration of using a variety of blocks significantly increased (with LevelUp, Mean=2.83, sd=0.72, Median=3). This change was deemed significant based on the Wilcoxon Signed-Rank Test.
When was LevelUp most useful?

![Graph showing number of accesses](image)

Fig. 4. A graph showing the number of times participants accessed LevelUp by the amount of time they had spent working on their project. The number of accesses is a total across participants.

\[(z=-2.40, \ W=4, \ p=0.016, \ r=0.69)\]

**D. LevelUp Accesses**

We tracked how often participants navigated to the LevelUp tab to see if that impacted their project scores. There was not a strong correlation between LevelUp accesses and final project scores. Almost all participants in LevelUp-P2 received high scores (90% or 100%) regardless of how much they looked at the tab. In LevelUp-P1, all of the participants who received less than 70% on their final project score looked at LevelUp three or fewer times. Even so, there were three participants who looked at the tool three or fewer times who received nearly perfect scores.

The only difference that we saw between conditions was that participants in LevelUp-P2 used LevelUp more regularly than LevelUp-P1. LevelUp-P1 had an average of 3.58 LevelUp accesses while LevelUp-P2 had an average of 4.10. In LevelUp-P1, the distribution of tool accesses was very diverse. There was one person who did not look at the tab and four people who looked at least six times. This contrasts with LevelUp-P2, where the distribution was more concentrated; the range of LevelUp accesses was from two to six. In both conditions, participants tended to access LevelUp either right at the start or towards the end of the project, with a peak occurring in the 10 to 12 minute range, as shown in Fig. 4.

**E. Participants’ Feedback**

In addition to evaluating the tool by the participants’ performance, we also solicited feedback from them about their experiences. At the end of the study, we asked six Likert scale questions about how participants used the tab and if they found it beneficial and intuitive (Fig. 5). Then, we asked three open-ended questions to understand (1) what they liked best about LevelUp, (2) ways they would improve the tool, and (3) what general feedback they had about the study. We did not see any significant differences between the two conditions in their responses to the Likert scale questions or open-ended questions, so we will explore participants’ responses to these questions all together.

**General Impressions.** Overall, participants responded positively to LevelUp. On the Likert response questions, 19 out of 25 respondents agreed or strongly agreed that the tool was useful and 20 agreed or strongly agreed that having access to the tab changed how they went about their project. Additionally, 21 participants agreed or strongly agreed that they understood how to use LevelUp and that the interface was not difficult to comprehend. In the open-ended response questions, participants elaborated that the tool was useful for helping them improve, expand on, and structure their work.

Participants stated that they used LevelUp as a starting point (2 participants), checklist (4 participants), reminder (3), benchmark (4), and motivator (2) while they worked on their text classification projects. In their responses about what they liked best about the tool, five participants indicated that they found the tool helpful because it helped them add onto and make improvements to their projects. “I liked how it told me what to incorporate into my program. It provided bullet points on how to make my code more complex” P23. Two additional participants spoke of the tool encouraging their creativity. Even though the project working time was short, one of these participants went beyond working with the coding blocks and classifier tool and started playing with the art and music creation functionalities in the programming environment.

Nine participants spoke of the tool as a scaffold that gave them direction and a clear indicator of their overall progress. Two participants said it was helpful for getting started: “[I] gave me a general idea of how to go about doing my project,”
instead of facing an empty canvas without any idea of what to do” P16.

In response to the question about how they would change the tool, nine participants wanted LevelUp to give more suggestions and support, with three specifically expressing that they wanted pictures to go along with the suggestions. “I wasn’t exactly sure what ‘use three text classifiers in block text’ meant at first, so it could be nice to have a little example or explanation or something” P17.

**Usage Patterns.** We observed a wide variety of usage patterns from participants. Nine out of 25 participants agreed or strongly agreed that LevelUp was most useful at the beginning and/or end of the project. Sixteen participants, some overlapping with the first group, stated that looking at the tool multiple times while working on the project was helpful. Finally, there were also two participants who disagreed that LevelUp was useful at all.

**Beginning and end usage.** Participants who used LevelUp at the beginning of their projects did so to know where to get started with their work. P08 is one example of a beginning- and-end LevelUp user. “It gave you a place to get started at the beginning. I got to see what particular things I should add/improve on” P08. They looked at LevelUp several times at the beginning of the project, incrementally completing its suggestions. Then, they left the tab alone for several minutes and did not return until the very end to confirm they had completed all of the suggestions. At that point, P08 got creative and began adding more class labels and programming blocks to deepen the complexity of their project.

**Frequently throughout usage.** It was more common for participants to look at the tool multiple times throughout the entire project. In their open-ended responses, two participants mentioned that the strength of LevelUp was that it gave them feedback about their work. P24 looked at LevelUp six times, all in the last 10 minutes of the project time. “I liked how it automatically refreshed each time I made an update so that I could immediately see my progress even if it wasn’t 100% complete yet” P24. Each time they looked at the tab, they made an update to their project: adding a third class label, then a fourth one, and then adding embedded conditionals. Notably, there was only one participant who made incremental improvements to their program then stopped as soon as they reached 100%.

**Non-users.** When participants did not find LevelUp useful, it was often because they ran into a challenge, either with the tool or the programming project, that they could not overcome. “After I checked it a couple times and it was the same I stopped using it entirely” P11. This problem could have been a malfunction with the tool or, as one participant mentioned, confusion about what the suggestions meant. An additional challenge was that the block-based programming and projects were difficult for some participants. One participant struggled with the interface so much that they did not get the chance to utilize the suggestions in the tool.

**VI. DISCUSSION**

The results of our study confirm both of our hypotheses: having access to LevelUp led to higher project scores and most participants indicated that the tool was intuitive and helpful. On the first project, LevelUp-P1 had access to LevelUp and did better than LevelUp-P2. Then, on the second project, LevelUp-P2 had access to LevelUp and achieved mostly perfect scores. Meanwhile, LevelUp-P1 did not have LevelUp, and their project scores remained about the same as their first projects. On the questionnaire about the usefulness of LevelUp, most participants found the tool helpful and easy to use. In their open-ended feedback, participants described LevelUp as a good starting point, progress checker, and creativity booster.

These results provide promising evidence about the efficacy of using automatic assessment tools in machine learning education, specifically for those not familiar with computer science concepts. However, there are some limitations to the generalizability of our findings due to the study design. For one, this study was conducted in a controlled lab setting where participants spent an hour learning about and implementing machine learning projects. In classrooms where students spend weeks or days learning about AI, we expect that their interaction with the tool would further evolve. Another limitation is that we worked with university-affiliated adults to evaluate our tool. We would want to do further research to understand how K-12 students and other adult learners engage with this tool.

With this in mind, the rest of the discussion considers additional findings from our results that we would consider exploratory. We discuss how getting access to the tool at a particular time impacts its effectiveness, participants’ insights on important features in automatic assessment tools, implications for assessment in ML education, and directions for future work.

**A. The impact of timing on project scores**

By dividing participants into LevelUp-P1 and LevelUp-P2, we were able to observe how the tool supported participants differently depending on when they were given access to it. LevelUp-P1’s scores when they had access to the tool were not nearly as high as the ones LevelUp-P2 achieved when they used LevelUp on their second project.

We may interpret this result through the lens of the Zone of Proximal Development (ZPD) from Vygotsky’s theory of learning and development [28]. The theory describes ZPD as “the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined through problem-solving under adult guidance, or in collaboration with more capable peers”. With the guidance of an automatic assessment tool, participants in LevelUp-P1 achieved more than they would have on their own. However, achieving a perfect project score was outside of the ZPD for them, guided or not. But by the second project, participants had built their skills and had a higher capability. The tool provided the right support to help LevelUp-P2 get mostly perfect scores.
Even though LevelUp was always beneficial, it was more beneficial once learners had experience. It is likely that if participants had access to LevelUp for both projects, even more of them would have reached 100% on the second project. However, given all the new information that learners have to manage when getting started, LevelUp could also be a distraction on the first project. After users gain some experience building text classifiers, the tool is better positioned to be a springboard for creating very high-quality projects.

B. Design considerations for automatic assessment tools

It was important that participants found the interface intuitive and the suggestions easy to understand. Participants who used and enjoyed the tool mentioned “ease of use” as one of its greatest features. Ease of access was also important. We built LevelUp directly into the programming platform and saw most participants use it by frequently visiting the tool and incrementally improving their code. Even so, participants suggested that it should be more present, say in a window on the code tab.

A potential flaw in our design was that users could treat the tool as a crutch. Coding is a subjective process and completion is heavily project-dependent, thus offering a metric for when a project is “good enough” could be misleading. We noticed one participant who stopped improving their project as soon as they reached 100% on the progress bar. Removing the progress bar portion of LevelUp could alleviate the pressure to obtain 100%. Plus, adding additional suggestions for improvement, as some participants requested, could further spark users’ creativity.

C. Automatic assessment tools for AI projects

We believe that more ML education tools should incorporate features that support students’ project creation. Existing ML education tools have made it really easy for non-experts to build ML models - most participants in our study were able to start building ML projects after a few minutes of instruction. But, having LevelUp encouraged participants to add more to their training data sets and write more complex code.

The criteria that LevelUp measured were helpful to participants, plus they aligned with criteria in other ML education curricula: the quantity of training data, balancing classes, and including many different labels [18], [25], [26]. Future tools could go even further by helping users test their models, as one participant requested, “I don’t know how practical this is but it would be nice if there was a way for it to test how good my examples were” P07. Though it might be difficult to implement automatic testing for open-ended ML projects, we saw that even just having a checklist of best practices to scaffold users’ work could be beneficial.

Our tool went further than other ML rubrics by also assessing users’ programming skills. For now, we only gave feedback on programming elements that are relevant to text classification projects (e.g. using confidence level blocks to validate classifier predictions). We imagine that adding additional programming criteria, such as those in similar tools like Dr. Scratch, could help users further improve their computational thinking skills [9], [11].

D. Future work

We would like to further explore failure cases with LevelUp. Some participants encountered difficulties, struggling with the programming interface or errors in the tool, and thus were less engaged. “I am not sure if I was using it correctly as it never seemed to change no matter what I did” P02. Three participants, all from LevelUp-P1 but with a range of prior programming experience, struggled with LevelUp and the programming exercises. It is worth understanding why they struggled and how we could better support them.

It would also be interesting to explore other methods of delivering this information to users. In this study, participants had access to our rubric via a static slideshow and the dynamic LevelUp. We can imagine other form factors such as notifications, which one participant suggested, or a virtual agent to support learners. Different form factors offer different opportunities for scaffolding and student self-regulation, thus it would be interesting to compare different approaches.

VII. Conclusion

This paper describes the design and evaluation of LevelUp, a novel tool we developed to help non-CS majors build block-based machine learning programs. LevelUp provides users with feedback on their projects, showing them what they have done well and how they can improve. In an evaluative study of the tool, users validated our hypothesis that having access to LevelUp would significantly bolster users’ ML projects. Though assessment continues to be a challenge in ML education, we hope the contributions of this tool will be helpful to learners and educators.

Acknowledgements

Thank you to members of the Personal Robots Group for their support in completing this study and revising the paper. This material is based upon work supported by grants from DP World and Microsoft Research. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of either organization.

References


