Investigating Elements of Student Persistence in an Introductory Computer Science Course

Juan D. Pinto
University of Illinois at Urbana-Champaign
jdpinto2@illinois.edu

Yingbin Zhang
University of Illinois at Urbana-Champaign
yingbin2@illinois.edu

Luc Paquette
University of Illinois at Urbana-Champaign
lpaq@illinois.edu

Aysa Xuemo Fan
University of Illinois at Urbana-Champaign
xuemof2@illinois.edu

ABSTRACT

We explore how different elements of student persistence on computer programming problems may be related to learning outcomes and inform us about which elements may distinguish between productive and unproductive persistence. We collected data from an introductory computer science course at a large midwestern university in the U.S. hosted on an open-source, problem-driven learning system. We defined a set of features quantifying various aspect of persistence during problem solving and used a predictive modeling approach to predict student scores on subsequent and related quiz questions. We focused on careful feature engineering and model interpretation to shed light on the intricacies of both productive and unproductive persistence. Feature importance was analyzed using SHapley Additive exPlanations (SHAP) values. We found that the most impactful features were persisting until solving the problem, rapid guessing, and taking a break, while those with the strongest correlation between their values and their impact on prediction were the number of submissions, total time, and (again) taking a break. This suggests that the former are important features for accurate prediction, while the latter are indicative of the differences between productive persistence and wheel spinning in a computer science context.

Keywords

Student modeling, persistence, wheel spinning, predictive modeling, behavior detection

1. INTRODUCTION

Research on student modeling has identified various behaviors and patterns related to learning outcomes and student success. One construct has both a history of research outside of Educational Data Mining (EDM) and is receiving renewed attention in the EDM community. Known by the diverse names of *grit* [8], *perseverance* [25], *academic tenacity* [9], and *persistence*, studies have focused on measuring the trait, identifying when students are exhibiting it, and quantifying its effects on various aspects of student learning. More traditional efforts on this front have focused on measuring persistence using questionnaires and testing its effect based on grades and test scores [8, 17, 41]. Efforts to identify persistence in log data of game-based learning systems [7, 27, 34] or intelligent tutoring systems (ITS) [15] have shown great promise. Many of these efforts have specifically focused on improving persistence detectors for on-the-fly student feedback systems or interventions.

One aspect of persistence that has gained interest in the EDM community in particular is the distinction between productive and unproductive persistence. Persistence is typically characterized by a determination to stick with a problem for long durations despite facing obstacles, and it has often been portrayed as a positive trait. However, researchers have come to question this simplistic stance, noting that there seem to be two related but opposing sides to persistence. On one hand, persistence may produce productive results when it leads to consistent, long-term effort [8] or when students relish the opportunity to overcome challenges [9]. On the other hand, students who are "stuck" may be better off going back to learning more about the subject rather than continuing to spend time working on a problem they don't yet fully understand [3]. In such cases, the student's persistence might be characterized as unproductive.

Given the opposing academic outlook of this dichotomy, understanding what differentiates productive from unproductive persistence is of critical importance. The latter has been termed *wheel spinning* in the literature and has been defined as "a student who spends too much time struggling to learn a topic without achieving mastery" [3]. Recent research has specifically focused on creating and improving automatic detectors of wheel spinning in ITSs [11, 15, 24, 39, 42] and game-based learning systems [27].

In the context of computer science education, [23] have suggested that fostering grit can lead to higher retention among CS students. Other research has identified a weak correlation between grit and measures of academic success [17, 25, 41], especially when focusing on one of the two main components of grit—perseverance of effort—which most closely aligns with definitions of persistence [35].

In this paper, we add to the existing literature by exploring how different elements of persistence on computer programming problems may contribute to learning outcomes. We defined a set of features quantifying various aspects of persistence during problem solving and used predictive modeling approaches to predict student scores on subsequent and related quiz questions. We focus on careful feature engineering and model interpretation to shed light on the intricacies of both productive and unproductive persistence. By investigating these constructs within a computer science course, our study also aims to better understand their application in this context.

2. RELATED WORK

2.1 Modeling Productive Persistence vs. Wheel Spinning

The EDM community's interest in persistence was sparked by [3], who found that students who struggle to master a skill within a certain timeframe are unlikely to do so at all. Besides identifying wheel spinning and describing how it differs from productive persistence, the same study found a clear correlation between wheel spinning and other negative behaviors such as gaming the system and disengagement.

Subsequent studies have devised variations in criteria for differentiating between productive persistence and wheel spinning [42], with many models defining mastery based on the number of correct submissions in a row and others relying heavily on the stability of Bayesian knowledge tracing (BKT) student model predictions [16]. Despite differences in operationalization, however, predictive machine learning models have been found to serve as successful wheel-spinning detectors. Some of the algorithms that have been used include linear regression [3], logistic regression [11, 42], decision trees [15, 27, 39], random forest [27, 42], and neural networks [24]. Most of these studies calculated productive persistence or wheel spinning labels based solely on the data gathered rather than relying on human observers or coders. Two notable exceptions are [24, 27].

The goal of the most recent studies has been to identify wheel spinning in ITSs as early as possible. [42] compared different criteria and feature sets and have shown that it is possible to make predictions with acceptable accuracy as early as step four of a problem. They were also surprised to find that a logistic regression model trained on only one feature ("correct response percentage") resulted in prediction performance that was close to their best models. Relying on hint requests, submission correctness, and time per skill, [39] concluded that models can detect students who will wheel spin after only three questions.

The studies mentioned thus far have focused almost exclusively on ITSs, which are most commonly used to teach math. Detecting and studying persistence on computer programming problems requires first understanding how data from these tasks has been analyzed in past studies.

2.2 Using Action Logs to Study Programming Behaviors

There is growing interest in leveraging data analytic methods to study students' action logs produced during programming activities [13], including to better understand the students' programming processes, behaviors and strategies. Log data have been used to generate visualizations of student behaviors that can be manually inspected to better understand their programming approach [5, 10], explore how students progress through homework assignments [6, 30], understand the learning pathways of novice programmers [4] and analyze problem-solving behavior in a debugging game [20].

Generally, two broad categories of features have been used: 1) frequencies of behaviors and 2) similarity/distance between programs. The first category provides aggregated information related to the quantity of actions performed by the student. This includes the number of blocks used in a Scratch program [10], how often a program was compiled and how many characters it included [5], the number of actions and logic primitives used [4], and the number of lines added, deleted, and modified [6]. [20] leveraged expert judgments to identify meaningful behaviors, such as massive deletion and replacing loops with repetitive code.

Studies have also developed features to evaluate how similar or different two computer programs are. [30] used a combination of the differences in bag of words, abstract syntax tree (AST) edits and similarity in calls to the application programing interface (API) to identify similar program states. [6], in addition to using this same method, considered the frequency of changes in a student's program and the magnitude of those changes.

As our goal was to focus on behaviors related to how students approach solving a problem, rather than investigating the content of the submitted solution, we used an approach in line with the first category to investigate elements of student persistence in a series of computer programming problems. This allowed us to focus specifically on the productive and unproductive behaviors of persistent students.

3. METHODS

3.1 Data Collection and Label Generation

We collected data from an introductory computer science course at a large midwestern university in the U.S. hosted on PrairieLearn, an open-source, web-based problem-driven learning system [40]. Throughout the semester, 733 students used PrairieLearn to submit almost-daily programming homework problems, take weekly quizzes, and complete cumulative exams. In addition, students were free to practice past problems and questions as much as they desired. As our work aims to investigate the relationship between persistence during homework and subsequent assessment, we filtered the data to focus on attempts submitted towards solving a homework problem or a quiz question. After removing practice submissions and other non-credit assignments, our resulting data set consisted of 290,703 individual homework problem attempts and 313,097 quiz question attempts.

All homework assignments were programming problems with checkstyle, compiler, and problem-specific tests that students' code had to pass to receive full credit. Students had one day to successfully complete each homework problem. They were allowed to submit solution attempts as often as required until they successfully passed all the tests. After each submission, the system ran tests to check the correctness of the solution and provided feedback indicating mistakes. First, the system tested whether the solution had any checkstyle and compiler errors. If such error existed, the system showed feedback about these errors and stopped. If there were no checkstyle or compiler errors, the system further used several problem-specific tests to examine whether the solution fulfilled the requirement. For example, given some random input, would the solution generate the correct output? If not, the system would return feedback about the problem-specific test error. Otherwise, the solution was regarded as correct.

We aggregated our dataset at the student-problem level using a series of features specifically related to persistence. While persistence can be studied at various grain sizes, we chose this level due to our interest in how students tackle difficulties within a particular programming problem. Similarly, we only kept instances that demonstrated struggling, as defined in section 3.2.1, since these were the cases that could elicit persistence from students.

Quizzes were conducted weekly as part of regular class activity to assess learning and consisted of both multiple-choice questions and programming tasks. Quizzes were made available at the end of the week and were designed to provide early assessment related to the content of the homework problems assigned earlier that week. We aligned the content of each homework problem to corresponding multiple-choice quiz questions to directly investigate the

relationship between persistence in specific homework problems and outcome on related assessment questions. Once we had these alignments, we calculated for each student-problem instance the total number of points obtained on the relevant quiz questions and the maximum possible points. Using these values, we then calculated the point percentage as the indicator of learning. Only quiz questions that students attempted were considered for these calculations. After these changes and calculations, our aggregated dataset consisted of 7,673 instances of student-problem pairs, submitted by a total of 710 students.

The resulting distribution of the score outcome variable had a strong negative skew, with most instances accumulated at higher scores, as shown in Figure 1. This is because students often managed to obtain a perfect score on their aligned quiz questions. Students were typically given two chances to select the right answer, the second time for half credit.

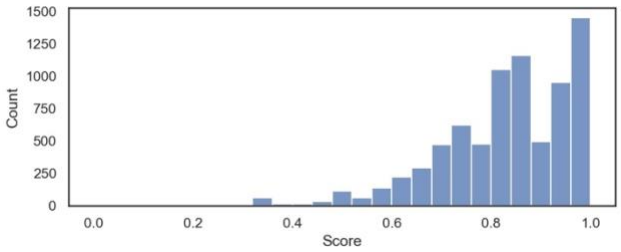

Figure 1. Distribution of score values

3.2 Feature Engineering

Given our goal to study how specific behaviors might be related to persistence, our feature engineering efforts focused on developing features based on an underlying rationale about their relationship to productive or unproductive persistence. Following the Carnegie Foundation for the Advancement of Teaching's definition of productive persistence—"tenacity plus the use of good strategies" [18]—we sought to identify good learning strategies and habits based on the available data. Other features were based on more generalized applications of the aspects of unproductive persistence that have been identified in the wheel-spinning literature. This process resulted in a total of 12 base features. We also standardized most of these at the problem level (by subtracting the problem's mean and dividing by the problem's standard deviation) to create an additional 10 features. The rest of this section describes each feature and our rationale behind it.

3.2.1 Struggling threshold features

Whether the student went beyond a problem's corresponding time or attempt threshold.

We defined students as struggling if they worked on a programming problem for a long time or if they submitted a high number of solutions to a problem. We considered that students could only show persistence in the context of problems for which they struggled.

This operationalization of struggling depends on identifying both a time and attempt threshold, each specifically calculated for that homework problem. Thus, once we calculated the thresholds for each problem, we created two binary struggling threshold features: beyond time threshold and beyond attempt threshold. We only kept instances of students that satisfied at least one of these two criteria. We also created two numerical features that measured a student's deviation from each of these thresholds. Because the thresholds were already calculated at the problem level, standardizing these deviation features would result in perfectly collinear features, so we did not standardize them.

For the time threshold, we used the minimum value between the 75th quantile of students' total time on each problem and 15 minutes. We combined the 75th quantile and 15 minutes to determine the time threshold based on several reasons. First, given that the course is only an introductory CS course, it is reasonable that one fourth of students struggled with difficult programming problems. Second, the proportion of students who struggled with unchallenging problems would be smaller. Using an absolute threshold would be better for these cases. Third, we used 15 minutes as the absolute threshold because 57.56% of problems had a 75th quantile of total time smaller than 15 minutes. It seems reasonable to regard close to half of problems as unchallenging.

Given that the number of attempts is an important indicator of persistence, many attempts on a problem might also be indicative of struggling, even when the total time spent on the problem falls under the time threshold. Analogous to deciding the time threshold, we used the minimum value between the 75% quantile of the number of attempts on a problem and 9 attempts to determine the attempt threshold. If the 75th quantile of the number of attempts on a problem was smaller than 9 attempts, the later became the attempt threshold. We used 9 attempts as the absolute threshold because 56.06% of problems had a 75th quantile of the number of attempts no more than 9 attempts. This number was close to 57.56%, the proportion of problems with a 75th quantile of total time smaller than 15 minutes.

3.2.2 Solved

Whether the student successfully solved the programming problem before the deadline.

This is directly related to wheel spinning as defined by [11]: "problem solving without making progress towards mastery." While PrairieLearn is not suited for measuring mastery the way [11] did with the Cognitive Algebra Tutor and ASSISTments ITSs (three consecutive, correct responses within a specific skill), persistence while struggling that does not lead to an eventual correct solution can be considered a form of wheel spinning or unproductive persistence. Based on this, we hypothesized that solving a challenging problem (productive persistence) would lead to a higher quiz-question score than not solving the problem (wheel spinning).

3.2.3 Number of submissions

The count of how many times the student submitted an attempted solution for the problem.

This is a typical measure used in the persistence literature [15, 39, 42]. Since submissions on PrairieLearn typically end when a student successfully solves a problem, this feature is a count of the number of failed attempts + 1. In essence, this is one way of measuring the level of persistence demonstrated. We reasoned that more unsuccessful attempts would indicate more wheel spinning, resulting in lower quiz scores.

3.2.4 Total time on problem

The total amount of time (in seconds) spent solving the problem.

As with the number of submissions, the time that students spend on a challenging problem might indicate the amount of persistence being demonstrated. We again reasoned that more time (and thus more wheel spinning) may be predictive of more struggling and lower scores on the quiz questions.

Our platform only allowed us to measure the time between submissions, so we had no way of knowing with certainty how much time was spent working on a problem. If the time difference between a student's two consecutive submissions was beyond 15 minutes, we regarded this student as being away from this problem during that interval (see the feature *taking a break* below for the choice of 15 minutes as a threshold). In these cases, we replaced this time difference with the student's mean time difference between other consecutive submissions on this problem so that we could estimate the student's total time on the problem more accurately.

3.2.5 Taking a break

Whether the student spent time away from the problem after passing one of the struggling thresholds.

We defined taking a break as a *struggling* student being away from the problem at least once. When the time between two consecutive submissions on the same problem went beyond 15 minutes, we regarded the student as away from the task. As discussed above, 15 minutes might be sufficient for solving unchallenging problems if students did not struggle. Moreover, 81.57% of pairs of consecutive submissions had a time difference less than 15 minutes. This proportion only increased slightly to 83.77% when increasing this threshold from 15 minutes to 1 hour. Thus, it is reasonable to use 15 minutes as the threshold for being away from the problem. Note that if a student attempted other homework problems between two consecutive submissions on the same problem, we regarded this student as interleaving rather than taking a break.

Our rationale for measuring break taking is based on the idea that a wheel-spinning state may be overcome by time away from task. Some of the cognitive benefits of breaks have been documented [1, 19, 26, 36] and seem to be especially impactful for intensive and prolonged tasks. The term wheel spinning itself was coined in reference to the imagery of a car spinning its wheels but not going anywhere, suggesting that the indiscriminate tactic of subsequent attempts may not always be productive. In their article defining this new construct, [3] suggest devising ways to break up fruitless attempts at solving problems. Our feature tries to capture students who independently choose to break up their homework in this way.

3.2.6 Interleaving

Whether the student switches to a different problem for a time and then comes back to continue attempting the original problem.

Interleaved practice, as opposed to blocked practice, refers to a learning technique that mixes up the order of topics, lessons, or problems presented. Studies have shown that this practice usually improves learning outcomes [32, 38], though—to the best of our knowledge—this has not been explored in a CS context. For the purposes of our study, we measured interleaving as a student attempting a problem without solving it, attempting a different problem, and then returning to continue working on the original problem. We reasoned that such a practice could potentially serve to break up the monotony and potential frustration associated with wheel spinning and thereby lead to better learning. We considered this an alternative to taking a break and did not double count such instances in the features.

3.2.7 Rapid guessing

Whether the student submitted at least three quick submissions in a row.

Quick, consequent submissions may indicate guessing or uncritical attempts to fix problems without much reflection. This behavior has been associated with students trying to game the system [2, 28] and with wheel spinning [3]. Given the nature of programming tasks as opposed to attempts in an ITS, we defined a quick submission as a gap between attempts of less than 15 seconds. If a student's submission stream to a problem contains three or more consecutive quick submissions, we labeled this student as performing rapid guessing on the problem. We hypothesized that rapid guessing would be associated with a lower score on related quiz questions.

3.2.8 Time interval between consecutive submissions

The student's mean and standard deviation of time intervals between consecutive attempts on the problem.

Shorter time between submissions may indicate more unproductive attempts to push through to an answer without stopping to think/work carefully or take breaks [39]. This is also similar to the common practice of cramming, as opposed to the more effective practice of spaced repetition. [15] found both the mean and standard deviation of time differences to be about equally as predictive of wheel spinning. We nevertheless chose to include both features in our initial model to test this claim. We did not count cases of break taking (intervals longer than 15 minutes) towards these features. Because of its association with wheel spinning, we hypothesized that we would find a positive correlation between these features and score.

3.3 Machine Learning and Interpretation

To test the importance of our various features, we created a random forest model using a shuffled 70/30 validation/testing split grouped by student with 5,396 and 2,277 instances respectively. We conducted 500 iterations of Bayesian hyperparameter optimization on the validation set using 10-fold cross validation grouped by student. This hyperparameter tuning was set to optimize the R^2 score.

We originally tested a wide array of models, including various linear, tree-based, and ensemble algorithms, and we further tuned some of the most promising ones. We found that variations of gradient boosting models performed best. However, we chose to focus on random forest for our feature interpretation for two reasons: (1) the performance gained by using the best models over random forest was negligible, and (2) random forest models have been shown to be useful for predictions related to persistence in other EDM research [27, 42].

Once we had constructed our final model, we re-trained it on the entire dataset in preparation for feature interpretation. For the task of interpreting feature importance, we analyzed SHapley Additive exPlanations (SHAP) values. SHAP is a game-theoretic approach that calculates the effect that each value in the feature matrix has on that instance's prediction, relative to the mean prediction [22]. That is, we can output a matrix with the same dimensions as the features data set, each value serving as an explanation of that feature's effect on the prediction made for that particular instance. These SHAP values are in the same unit as the target label—percentage score in our case—further lending themselves for interpretation. Though SHAP values are very resource-intensive to fully and accurately calculate, the nature of tree-based models makes it possible to optimize the process significantly [21].

It is important to note that the mean of the SHAP values for any feature will always be zero. This is because SHAP values are calculated as the difference of each feature-instance from the mean predicted score. However, by finding the mean absolute value of the SHAP values for each feature, we can identify which features have the strongest broad, average impact on prediction.

While mean Gini impurity has been used to interpret features in the persistence literature [42], and permutation feature importance is commonly used as well, numerous studies have identified potential issues with these approaches that can lead to misleading interpretations [12]. This is especially true when using highly correlated features [37], which is the case with our data. Furthermore, SHAP allows for investigation into the interplay between features beyond what these other methods can do.

We conducted all our work using open-source Python packages built on top of Scikit-learn [29]. We tested and tuned a variety of models using PyCaret [31], performed Bayesian optimization [14] with scikit-optimize [33], and investigated feature importance using SHAP [22].

4. RESULTS AND DISCUSSION

4.1 Model Results and Preliminary Analysis

Our tuned random forest model attained an average cross-validated R^2 of 0.133 and an average RMSE of 0.129 on the validation set. On the held-out testing set, the resulting R^2 was 0.145 and the RMSE was 0.130. Our persistence features accounted for roughly 14% of the variation in related quiz scores.

A preliminary analysis of our model uncovered certain important patterns. For one, our least impactful features were all binary measures—such as whether interleaving, rapid guessing, or break-taking were observed—whereas our top features were the standardized measures of those binary features. Figure 2 shows the entire set of feature rankings based on mean absolute SHAP values.

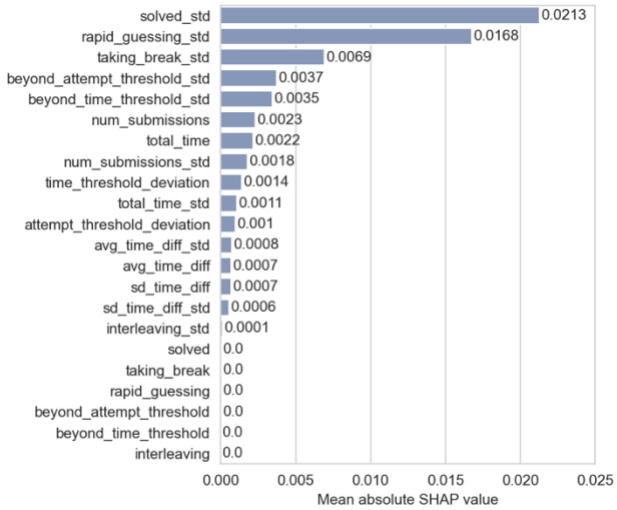

Figure 2. Preliminary model feature rankings

A detailed exploration of these features revealed what appears to be an opposing impact between some binary features and their standardized counterparts. For example, the feature *solved* has a negative correlation between its values and its SHAP values (r = -0.217, $p < 0.0001$), whereas its standardized version, *solved_std*, has a positive correlation (r = 0.33, $p < 0.0001$). Measuring this correlation between feature and SHAP values allows us to better understand how the model is using the feature. Higher correlation, and thus a stronger linear relationship, suggests a more straight-forward interpretation for the feature's role in the model. While the impact of *solved* is very small in the overall model (ranked 17th, mean absolute SHAP = 0.00003), *solved_std* is our top feature in terms of overall impact on the predicted score (mean absolute SHAP = 0.02129). We found this same inverted relationship between many other impactful standardized features and their original, binary, far less impactful counterparts.

Because we standardized features at the problem level, the correlation between each unstandardized and corresponding standardized feature is never quite perfect, but some do come close. Random forest models typically do not suffer from collinear features the way more traditional statistical regression methods do. This is largely because of the way features are randomly sampled for each tree. Even when both collinear features are part of the feature subset, a decision tree will typically ignore one in favor of the other. We suspect that much of our model's preference for the standardized features over unstandardized ones is the added problem-level information they contain, which could be interpreted as information regarding the difficulty of the problem.

However, while the predictive power of a random forest is not affected by collinear features, model interpretability suffers, as we found through our preliminary analysis. Given our goal of better understanding the different aspects of persistence and their relationships, we decided to remove the original non-standardized features. We also removed *time_threshold_deviation* and *attempt_threshold_deviation*, which were very highly correlated with *total_time_std* and *num_submissions_std* respectively. We then re-trained and re-tested our model.

After removing these features, we found that our model's average cross-validated R^2 on the validation set increased slightly, from 0.133 to 0.134, while RMSE remained constant. On the held-out testing set, its R^2 also increased, from 0.145 to 0.147, while RMSE remained constant. We then re-trained our model on the entire dataset in preparation for our in-depth feature analysis.

4.2 Feature Importance and Interpretation

4.2.1 Feature rankings

Our analysis using SHAP values found that the *solved_std* and *rapid_guessing_std* features had the biggest effect, accounting for an average impact of 0.0215 and 0.0172 on the predicted score respectively. The third most important feature, *taking_break_std*, had an average impact less than half as strong at 0.0076. Together, these three features account for 75% of all features' total impact on the predicted score. Figure 3 shows the feature rankings based on mean absolute SHAP values, while Table 1 allows for comparison with other methods such as Gini-impurity-based importance and permutation importance. Rankings based on these three different approaches yielded almost identical results with only minor variations, strengthening the reliability of our findings.

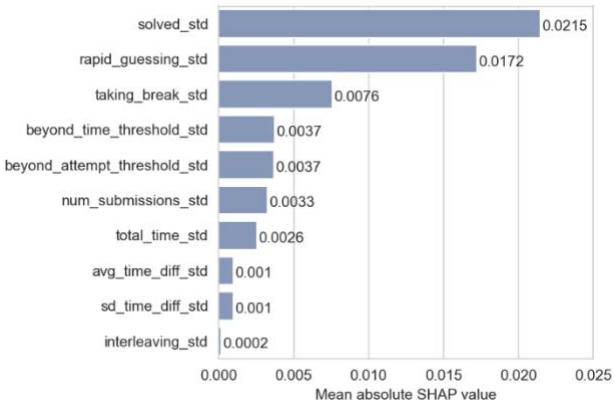

Figure 3. Final model feature rankings

Besides ranking the features by impact on the predicted score, SHAP values allow us to explore the nature of that impact more deeply, as well as the interactions between features. Figure 4 is a beeswarm plot of SHAP values by feature with color indicating the value of each individual instance.

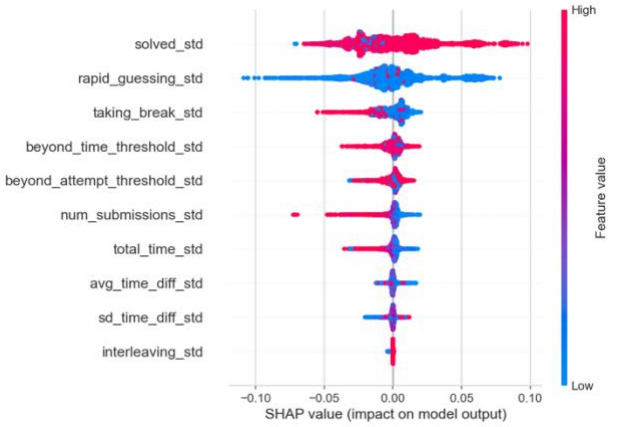

Figure 4. SHAP beeswarm plot

To further aid our interpretation, we also explored which features had the highest absolute correlation between their values and their corresponding SHAP values. In essence, this correlation is a measure of just how linear each feature's effect is on the predicted score. We calculated Pearson's *r* for all features (see Table 1) and found that all *p* values were below 0.0001, except for *sd_time_diff*. Throughout this analysis, we point out when a feature's correlation is indicative of a linear relationship.

4.2.2 Solved

We can see (Figure 4) that the bulk of *solved_std* is composed of high values (red color), indicating that most students managed to solve most homework problems. The long positive skew suggests that small, positive variations in this feature could potentially push the predicted quiz score up by about 0.1. The few lower values in this feature (blue color) are found on the left side of the plot, suggesting that not solving the problem tended to pull the predicted score down. Indeed, we found a moderate positive linear relationship between *solved_std* and its SHAP values (r = 0.34), further confirming our initial analysis.

This confirms our hypothesis. It suggests that solving a challenging problem (productive persistence) may be related to a better understanding of the underlying concepts, whereas not solving the problem (wheel spinning) suggests a lack of understanding.

4.2.3 Rapid guessing

Our models' second most impactful feature, *rapid_guessing_std*, is in many ways the opposite. Most students did not engage in rapid guessing. Those who did, particularly on homework problems where few others did—identified by high *rapid_guessing_std*, or red color in the beeswarm plot (Figure 4)—were more generally affected negatively in their predicted score based on this feature. This effect can more clearly be seen when plotting the SHAP values for the feature against the values of the feature itself (Figure 5). This view allows us to get a better sense of how most instances with a higher *rapid_guessing_std* value impact the predicted score negatively. This aligns with our hypothesis: rapid guessing, with its potential implications of wheel spinning [3] and gaming the system [2, 28], is indicative of lower learning outcomes.

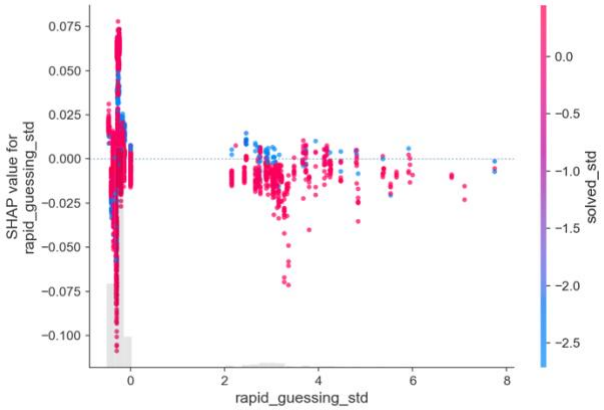

Figure 5. Correlation between *rapid_guessing_std* and its SHAP values, with *solved_std* as color

By adding the values of our top impactful feature, *solved_std*, as the color of the plot illustrated in Figure 5, we can also see an interesting interaction between the two features. It appears that the impact of the high *rapid_guessing_std* values is at least partly dependent on *solved_std*—instances where the student failed to solve the problem (in blue) were less negatively impacted by *rapid_guessing_std* (as indicated by their mostly positive SHAP

Table 1. Feature impact measures (*r* is correlation between feature values and corresponding SHAP values)

feature	mean absolute SHAP	Gini importance	permutation importance	*r*	*p*
solved_std	0.02149	0.26554	0.17083	0.340	< 0.0001
rapid_guessing_std	0.01724	0.20987	0.17059	-0.076	< 0.0001
taking_break_std	0.00758	0.10511	0.04310	-0.608	< 0.0001
beyond_time_threshold_std	0.00373	0.07514	0.02790	-0.349	< 0.0001
beyond_attempt_threshold_std	0.00369	0.07028	0.02760	-0.080	< 0.0001
num_submissions_std	0.00326	0.07751	0.03082	-0.833	< 0.0001
total_time_std	0.00257	0.07871	0.02550	-0.773	< 0.0001
avg_time_diff_std	0.00099	0.05398	0.01558	-0.158	< 0.0001
sd_time_diff_std	0.00098	0.06166	0.01856	0.008	> 0.5
interleaving_std	0.00017	0.00219	0.00028	0.126	< 0.0001

values). One explanation may be that students who rely on rapid guessing and manage to solve the problem may come away with more misguided confidence in their mastery of the material than those who fail to solve the problem and are thus less likely to consider reviewing before a quiz. However, this hypothesis was not investigated further.

4.2.4 *Taking a break*

Our model's third most impactful feature, *taking_break_std*, has a very clear pattern that is easily observable in Figure 4. Lower feature values generally lead to a positive impact on predicted score, whereas taking a break is more likely to have a negative impact on score. We found a negative linear relationship between *taking_break*_std and its SHAP value (Figure 6), with Pearson's *r* of -0.608. The distribution of SHAP values for this feature indicates a potential negative impact about three times as large as the positive one.

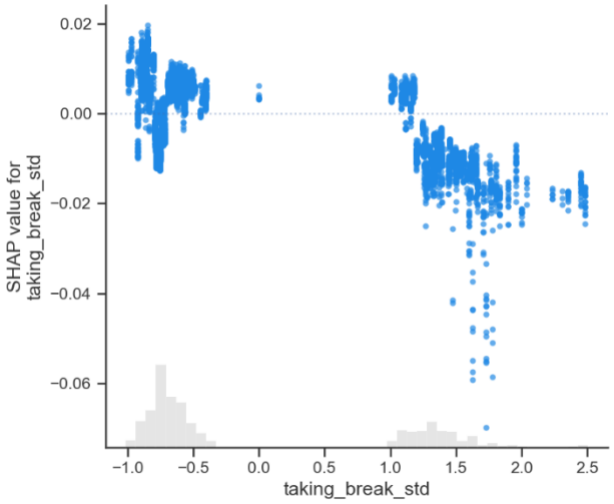

Figure 6. Correlation between *taking_break_std* and its SHAP values

This result is the opposite of what we hypothesized. Since taking breaks during a difficult task has been shown to improve cognition [36], we hypothesized that students who took a break while struggling would ultimately be more productive. We specifically marked a student as taking a break only if there was a large gap between submissions (15 minutes) *after* they had passed one of the two struggling thresholds.

One possible explanation is that students who took a break did, in fact, perform better than they would have otherwise. Since our method does not directly test causation, our model may be using this feature as a proxy for students who struggled more than others. Another possibility is that this feature is not solely capturing intentional break-taking, but also interruptions to students' work, which may serve as distractions—certainly not an ideal learning situation. We did not calculate how many times students took a break, only if there was at least one 15-minute gap between submissions when struggling. Finally, because homework problems were due at midnight on the day they became available, students may simply not have had sufficient time for effective break taking. Without additional information about learning context or calculating additional features, we have no way of knowing which of these explanations, if any, are the most likely.

4.2.5 *Struggling threshold features*

For *beyond_time_threshold_std*, we can see in Figure 4 that lower values generally lead to increases in the predicted score and vice versa. This is indicative of the underlying attribute this feature attempts to capture—going beyond the time threshold yields smaller (generally negative) SHAP values, whereas not going beyond the time threshold yields larger (generally positive) values, the exact value being heavily affected by how much other students crossed the threshold on the same problem. For students who take longer than the norm, this generally has a negative effect on their score. The relationship here is moderately linear with an *r* of -0.349.

The fifth top feature that we identified, *beyond_attempt_threshold_std*, does not have such a clear pattern. The SHAP values seem to be widely spread irrespective of the feature's values. The feature's distribution is bimodal, as is the case with most of the features that standardize a binary variable, and we did find a small distinction in the SHAP values between the two modes (Figure 7). While the mean for each mode is essentially zero, higher instances of *beyond_attempt_threshold_std*, which correspond with student-problem instances that went beyond that problem's attempt threshold, have a moderate negative correlation with their SHAP values (r = -0.409, p < 0.0001) and lower instances, on the other hand, have a positive correlation about equally as strong (r = 0.382, p < 0.0001). This suggests that the impact of this feature on predicted score is highly dependent on how much one's status on the underlying binary variable (*beyond_attempt_threshold*) varies from the norm for that given homework problem.

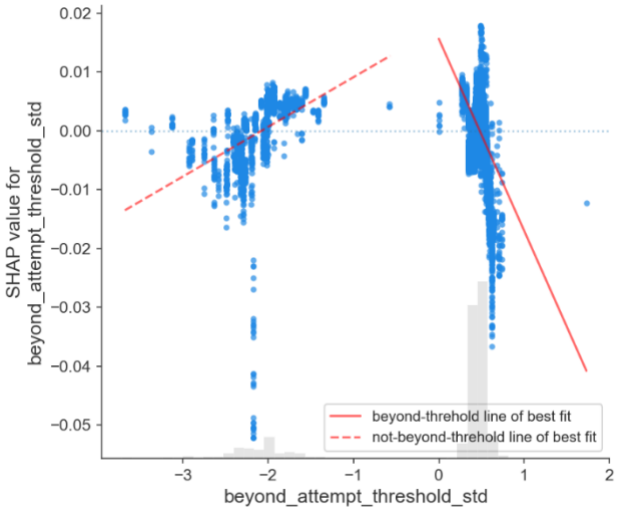

Figure 7. Correlation between *beyond_attempt_threshold_std* and its SHAP values (with annotations)

4.2.6 *Number of submissions*

We found that *num_submissions_std*, our model's sixth top feature in terms of impact, has the strongest correlation between its feature values and SHAP values (r = -0.833). This fits with our hypothesis. The more attempts that students submit, the more likely they are to be struggling, and the less likely they are to perform well when tested on the same skills during their weekly quiz.

4.2.7 *Time features*

We found that our three time-related features—not including *beyond_time_threshold_std*, which is of a very different nature since its non-standardized version is a binary feature—had some of the weakest predictive power in our model. *total_time_std* had a

still moderate mean absolute SHAP at 0.00257 and a very strong correlation between its feature and SHAP values with r = -0.773. *avg_time_diff_std* and *sd_time_diff_std*, by comparison, had a much lower mean absolute SHAP (respectively 0.00099 and 0.00098) and no correlation.

The strong, negative correlation between *total_time_std* and its SHAP values mean that the model is interpreting longer time on a problem as being related to lower learning outcomes, or at the very least as a student struggling enough with a problem to lead to a lower score on the weekly quiz. This latter possibility is in line with our hypothesis and with what we found for *beyond_time_threshold_std*. Interestingly, this pattern is far more pronounced for instances that went beyond the time threshold (red points in Figure 8), whereas the relationship is seemingly reversed for cases where students did not go beyond the time threshold (blue/purple points in Figure 8).

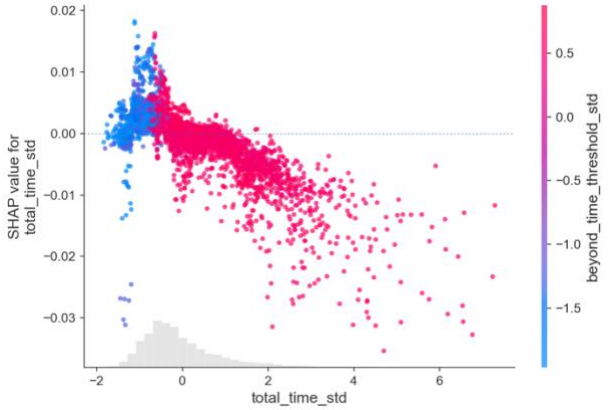

Figure 8. Correlation between *total_time_std* and its SHAP values, with *beyond_time_threshold_std* as color

As for the two features that specifically look at time between submissions (*avg_time_diff_std* and *sd_time_diff_std*), their weakness both in predictive impact and correlation with SHAP values suggest at face value that this factor has little value at predicting learning success (or lack thereof) when students struggle with a problem. These features' impact may also have been affected by the high correlation between them (r = 0.76). Similar information may have also been captured by a combination of *beyond_time_threshold_std* and *beyond_attempt_threshold_std*.

4.2.8 Interleaving

Finally, our model's least impactful feature, *interleaving_std*, had by far the lowest mean absolute SHAP value (0.00017) and a low correlation between its features and its SHAP values (r = 0.126). We originally hypothesized that this feature would play a bigger role in predicting students' scores, considering that the practice of interleaving when struggling is generally considered a good learning practice [32, 38]. However, its low impact in our model is likely because we had so few instances of interleaving—only nine out of 7,673 instances. Most of these nine did lead to an increase in predicted score, but without more examples of the practice, we are unable to make any sound conclusions regarding its role.

4.3 Limitations

Our study suffers from limitations primarily related to the aligned-quiz-question scores we calculated for each student-problem instance. For one, the score distribution was heavily skewed due to the abundance of almost perfect quiz scores. Additionally, while the PrairieLearn platform allowed us to use the course's quizzes without requiring students to take an additional posttest, the scores did not take into account students' prior knowledge and skills. This made it difficult to measure the impact of students' productive vs. unproductive persistence directly.

These factors likely led to our model's limited predictive performance (R^2 = 0.147 on the held-out test set). While we believe that our final model's performance was sufficient for our purposes of interpreting the relationship between elements of persistence and learning outcomes, it should be possible to create a more accurate model without severely sacrificing interpretability.

5. CONCLUSION

The most impactful features were those related to solving the problem, rapid guessing, and taking a break. Those with the most straightforward linear effect were the number of submissions, total time, and (again) taking a break. All three of the latter had a strong negative correlation between their feature values and their impact on prediction. In other words, more attempts, taking a longer time, and taking a break are all correlated with lower scores on related quiz questions. Solving the problem—our most impactful feature—had a moderate positive correlation, highlighting the positive nature of the relationship between successfully completing homework problems and score on subsequent related quiz questions.

This all suggests that solving the problem and rapid guessing are important features for accurate prediction, while the number of submissions and total time are indicative of the differences between productive persistence and wheel spinning in a computer science context. Taking a break fits into both of these categories.

Perhaps most important, we were able to identify features that are directly related to learning strategies. Our findings suggest that students should avoid rapidly submitting subsequent programming attempts without actively trying to address problems in their code (rapid guessing). Taking a break may also be unproductive behavior, though this finding may be an artifact of the specific context in which students were able to submit homework in this course, as well as the particular way in which we calculated this feature. As for interleaving, its predictive strength in our model was low, but its effects nevertheless suggest that a future investigation should study whether it can be an effective practice when struggling on a problem.

In order to address the limitations of our study, we suggest that future research focus on devising a more robust measure of learning that takes into account students' individual starting points. Additionally, for the CS context of this study, a valid measure of programming proficiency that considers the problem-solving process would be superior to the quiz scores we used as proxy.

6. ACKNOWLEDGMENTS

We would like to acknowledge NSF grant #DRL-1942962 for making this work possible.

7. REFERENCES

[1] Ariga, A. and Lleras, A. 2011. Brief and rare mental "breaks" keep you focused: Deactivation and reactivation of task goals preempt vigilance decrements. *Cognition*. 118, 3 (Mar. 2011), 439–443. DOI:https://doi.org/10/b8qg78.

[2] Baker, R.S., Corbett, A.T., Koedinger, K.R. and Wagner, A.Z. 2004. Off-task behavior in the cognitive tutor classroom: When students "game the system." *CHI '04: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2004), 8.

[3] Beck, J.E. and Gong, Y. 2013. Wheel-spinning: Students who fail to master a skill. *Artificial Intelligence in Education* (Berlin, Heidelberg, 2013), 431–440.

[4] Berland, M., Martin, T., Benton, T., Petrick Smith, C. and Davis, D. 2013. Using learning analytics to understand the learning pathways of novice programmers. *Journal of the Learning Sciences*. 22, 4 (Oct. 2013), 564–599. DOI:https://doi.org/10/gg7fkh.

[5] Blikstein, P. 2011. Using learning analytics to assess students' behavior in open-ended programming tasks. *Proceedings of the 1st International Conference on Learning Analytics and Knowledge* (Banff, Alberta, Canada, Feb. 2011), 110–116.

[6] Blikstein, P., Worsley, M., Piech, C., Sahami, M., Cooper, S. and Koller, D. 2014. Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. *Journal of the Learning Sciences*. 23, 4 (Oct. 2014), 561–599. DOI:https://doi.org/10.1080/10508406.2014.954750.

[7] DiCerbo, K.E. 2014. Game-based assessment of persistence. *Journal of Educational Technology & Society*. 17, 1 (2014), 17–28.

[8] Duckworth, A.L., Peterson, C., Matthews, M.D. and Kelly, D.R. 2007. Grit: Perseverance and passion for long-term goals. *Journal of Personality and Social Psychology*. 92, 6 (2007), 1087–1101. DOI:https://doi.org/10.1037/0022-3514.92.6.1087.

[9] Dweck, C.S., Walton, G.M. and Cohen, G.L. 2014. *Academic tenacity: Mindsets and skills that promote long-term learning*. Bill & Melinda Gates Foundation.

[10] Fields, D.A., Quirke, L., Amely, J. and Maughan, J. 2016. Combining big data and thick data analyses for understanding youth learning trajectories in a summer coding camp. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (New York, NY, USA, Feb. 2016), 150–155.

[11] Gong, Y. and Beck, J.E. 2015. Towards detecting wheel-spinning: Future failure in mastery learning. *Proceedings of the Second (2015) ACM Conference on Learning @ Scale* (Vancouver BC Canada, Mar. 2015), 67–74.

[12] Hooker, G. and Mentch, L. 2019. Please stop permuting features: An explanation and alternatives. *preprint arXiv:1905.03151*. (May 2019).

[13] Ihantola, P. et al. 2015. Educational data mining and learning analytics in programming: Literature review and case studies. *Proceedings of the 2015 ITiCSE on Working Group Reports* (Vilnius Lithuania, Jul. 2015), 41–63.

[14] Joy, T.T., Rana, S., Gupta, S. and Venkatesh, S. 2016. Hyperparameter tuning for big data using Bayesian optimisation. *2016 23rd International Conference on Pattern Recognition (ICPR)* (Dec. 2016), 2574–2579.

[15] Kai, S., Almeda, M.V., Baker, R.S., Heffernan, C. and Heffernan, N. 2018. Decision tree modeling of wheel-spinning and productive persistence in skill builders. *JEDM | Journal of Educational Data Mining*. 10, 1 (Jun. 2018), 36–71. DOI:https://doi.org/10.5281/zenodo.3344810.

[16] Käser, T., Klingler, S. and Gross, M. 2016. When to stop? Towards universal instructional policies. *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge - LAK '16* (Edinburgh, United Kingdom, 2016), 289–298.

[17] Kench, D., Hazelhurst, S. and Otulaja, F. 2016. Grit and growth mindset among high school students in a computer programming project: A mixed methods study. *ICT Education* (Cham, 2016), 187–194.

[18] Krumm, A.E., Beattie, R., Takahashi, S., D'Angelo, C., Feng, M. and Cheng, B. 2016. Practical measurement and productive persistence: Strategies for using digital learning system data to drive improvement. *Journal of Learning Analytics*. 3, 2 (Sep. 2016), 116–138. DOI:https://doi.org/10/ggxwxt.

[19] Kühnel, J., Zacher, H., Bloom, J. de and Bledow, R. 2017. Take a break! Benefits of sleep and short breaks for daily work engagement. *European Journal of Work and Organizational Psychology*. 26, 4 (Jul. 2017), 481–491. DOI:https://doi.org/10/gfzk8b.

[20] Liu, Z., Zhi, R., Hicks, A. and Barnes, T. 2017. Understanding problem solving behavior of 6–8 graders in a debugging game. *Computer Science Education*. 27, 1 (Jan. 2017), 1–29. DOI:https://doi.org/10/gftxxk.

[21] Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N. and Lee, S.-I. 2020. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*. 2, 1 (Jan. 2020), 56–67. DOI:https://doi.org/10/ggjtp4.

[22] Lundberg, S.M. and Lee, S.-I. 2017. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*. 30, (2017).

[23] Mahatanankoon, P. and Sikolia, D.W. 2017. Intention to remain in a computing program: Exploring the role of passion and grit. *Twenty-third Americas Conference on Information Systems*. (2017).

[24] Matsuda, N., Chandrasekaran, S. and Stamper, J. 2016. How quickly can wheel spinning be detected? *Proceedings of The 9th International Conference on Educational Data Mining (EDM 2016)* (2016), 607–608.

[25] McDermott, R., Daniels, M. and Cajander, Å. 2015. Perseverance measures and attainment in first year computing science students. *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (Vilnius, Lithuania, Jun. 2015), 302–307.

[26] McGinley, L. 2011. *Test performance and study breaks*. Fort Hays State University.

[27] Owen, V.E., Roy, M.-H., Thai, K.P., Burnett, V., Jacobs, D., Keylor, E. and Baker, R.S. 2019. Detecting wheel-spinning and productive persistence in educational games. *Proceedings of The 12th International Conference on Educational Data Mining (EDM 2019)* (Jul. 2019), 378–383.

[28] Paquette, L., de Carvalho, A.M.J.A. and Baker, R.S. 2014. Towards Understanding Expert Coding of Student Disengagement in Online Learning. *Proceedings of the 36th Annual Cognitive Science Conference* (2014), 1126–1131.

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

[30] Piech, C., Sahami, M., Koller, D., Cooper, S. and Blikstein, P. 2012. Modeling how students learn to program. *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (New York, NY, USA, Feb. 2012), 153–160.

[31] PyCaret: An open source, low-code machine learning library in Python: 2020. *https://www.pycaret.org*.

[32] Rohrer, D., Dedrick, R.F. and Stershic, S. 2015. Interleaved practice improves mathematics learning. *Journal of Educational Psychology*. 107, 3 (2015), 900–908. DOI:https://doi.org/10/gf7dfp.

[33] Scikit-optimize: Sequential model-based optimization in Python: 2020. *https://scikit-optimize.github.io/*.

[34] Shute, V.J., D'Mello, S., Baker, R., Cho, K., Bosch, N., Ocumpaugh, J., Ventura, M. and Almeda, V. 2015. Modeling how incoming knowledge, persistence, affective states, and in-game progress influence student learning from an educational game. *Computers & Education*. 86, (Aug. 2015), 224–235. DOI:https://doi.org/10.1016/j.compedu.2015.08.001.

[35] Sigurdson, N. and Petersen, A. 2018. An exploration of grit in a CS1 context. *Proceedings of the 18th Koli Calling International Conference on Computing Education Research* (Koli, Finland, Nov. 2018).

[36] Steinborn, M.B. and Huestegge, L. 2016. A walk down the lane gives wings to your brain: Restorative benefits of rest breaks on cognition and self-control. *Applied Cognitive Psychology*. 30, 5 (2016), 795–805. DOI:https://doi.org/10/ghcrj3.

[37] Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T. and Zeileis, A. 2008. Conditional variable importance for random forests. *BMC Bioinformatics*. 9, 1 (Dec. 2008). DOI:https://doi.org/10/d7p3rw.

[38] Taylor, K. and Rohrer, D. 2010. The effects of interleaved practice. *Applied Cognitive Psychology*. 24, 6 (2010), 837–848. DOI:https://doi.org/10/fkm7mp.

[39] Wang, Y., Kai, S. and Baker, R.S. 2020. Early detection of wheel-spinning in ASSISTments. *Artificial Intelligence in Education*. I.I. Bittencourt, M. Cukurova, K. Muldner, R. Luckin, and E. Millán, eds. Springer International Publishing. 574–585.

[40] West, M., Herman, G. and Zilles, C. 2015. PrairieLearn: Mastery-based online problem solving with adaptive scoring and recommendations driven by machine learning. *2015 ASEE Annual Conference and Exposition Proceedings* (Seattle, Washington, Jun. 2015), 26.1238.1-26.1238.14.

[41] Wolf, J.R. and Jia, R. 2015. The role of grit in predicting student performance in introductory programming courses: An exploratory study. *SAIS 2015 Proceedings*. 21, (2015).

[42] Zhang, C., Huang, Y., Wang, J., Lu, D., Fang, W., Fancsali, S., Holstein, K. and Aleven, V. 2019. Early detection of wheel spinning: Comparison across tutors, models, features, and operationalizations. *Proceedings of The 12th International Conference on Educational Data Mining (EDM 2019)*. (2019), 468–473.