Revista Brasileira de Informática na Educação – RBIE
Brazilian Journal of Computers in Education
(ISSN online: 2317-6121; print: 1414-5685)
http://br-ie.org/pub/index.php/rbie

Submission: 19/Apr/2023; 1st round notif.: 13/Sep/2023;
Camera ready: 01/Nov/2023; Edition review: 27/Dec/2023; Available online: 27/Dec/2023; Published: 27/Dec/2023;

Learning Analytics in Introductory Programming Courses: a Showcase from the Federal University of Amazonas

Flávio J. M. Coelho
Computing Research Group
Amazonas State University
ORCID: 0000-0001-7558-9436
fcoelho@uea.edu.br

Elaine H. T. Oliveira
Institute of Computing
Federal University of Amazonas
ORCID: 0000-0003-2884-9359
elaine@icomp.ufam.edu.br

Filipe D. Pereira
Department of Computer Science
Federal University of Roraima
ORCID: 0000-0003-4914-3347
filipe.dwan@ufrr.br

David B. F. Oliveira
Institute of Computing
Federal University of Amazonas
ORCID: 0000-0002-2887-2324
david@icomp.ufam.edu.br

Leandro S. G. Carvalho
Institute of Computing
Federal University of Amazonas
ORCID: 0000-0003-2970-2084
galvao@icomp.ufam.edu.br

Eduardo J. P. Souto
Institute of Computing
Federal University of Amazonas
ORCID: 0000-0003-0003-908X
esouto@icomp.ufam.edu.br

Marcela Pessoa
Computing Research Group
Amazonas State University
ORCID: 0000-0002-3064-5585
msppessoa@uea.edu.br

Rafaela Melo
Institute of Computing
Federal University of Amazonas
ORCID: 0000-0001-6198-8962
rmelo@icomp.ufam.edu.br

Marcos A. P. de Lima
Institute of Computing
Federal University of Amazonas
ORCID: 0000-0002-4560-4946
marcos.lima@icomp.ufam.edu.br

Fabíola G. Nakamura
Institute of Computing
Federal University of Amazonas
ORCID: 0000-0001-8251-3202
fabiola@icomp.ufam.edu.br

Abstract

Introductory computing courses have a high failure rate worldwide. At the Federal University of Amazonas, this also happens and, since 2016 a group of professors decided to reformulate the course at the institution and some learning analytics initiatives have been adopted. The reformulation included a review of the course program and the use of an online judge. After all these years of research, the group has enough material and data and it is a good moment to summarize what has been done and the achievements so far. In this article, the focus will be the learning analytics in three main areas: student performance prediction, classification of difficulty of programming exercises, and gamification. Also, as a contribution, for the first time in a journal, the whole dataset is available to the community.

Keywords: *Failure, Learning Analytics, Courses, Introductory Computing.*

Cite as: Coelho, Flavio J. M., Oliveira, Elaine H. T., Pereira, Filipe D., Oliveira, David B. F., Oliveira, Leandro S. G., Souto, Eduardo J. P., Pessoa, Marcela, Melo, Rafaela, Lima, Marcos A. P., Nakamura & Fabíola G. (2023). Learning Analytics in Introductory Programming Courses: a Showcase from the Federal University of Amazonas. Brazilian Journal of Computers in Education (Revista Brasileira de Informática na Educação – RBIE), 31, 1089-1127. DOI: 10.5753/rbie.2023.3334.

1 Introduction

Computer Science 1 (CS1) is a fundamental curricular component of Science, Technology, Engineering, and Mathematics (STEM) undergraduate degree programs. It introduces students to programming logic and serves as an initial exposure to programming languages and fundamental data structures.

Although it is a crucial course for many science and engineering degree programs, CS1 has worldwide high failure and dropout rates, which is a matter of concern for educational institutions (Bennedsen & Caspersen, 2019; Ihantola et al., 2015). In response, numerous studies have been conducted to comprehend student behavior (Blikstein, 2011), predict student performance (Lacave et al., 2018; Watson et al., 2013), and identify the need for additional support (Castro-Wunsch et al., 2017), among other factors. Additionally, a significant challenge in teaching CS1 is to motivate students who may not intend to pursue programming as a professional career (Echeverría et al., 2017; Norman & Adams, 2015; Santana & Bittencourt, 2018).

In light of these issues, instructors from the Institute of Computing (IComp) at Federal University of Amazonas (Universidade Federal do Amazonas – UFAM) have been continuously improving CS1 teaching practice, since 2015. This initiative was driven by low approval rates between 2012 and 2014, bellow 50%. The implemented changes included the following actions: adoption of a practice-oriented learning methodology, supported by an online judge; frequent formative assessments; update of teaching material; recruitment of teaching assistants; and introduction of gamification in the online judge.

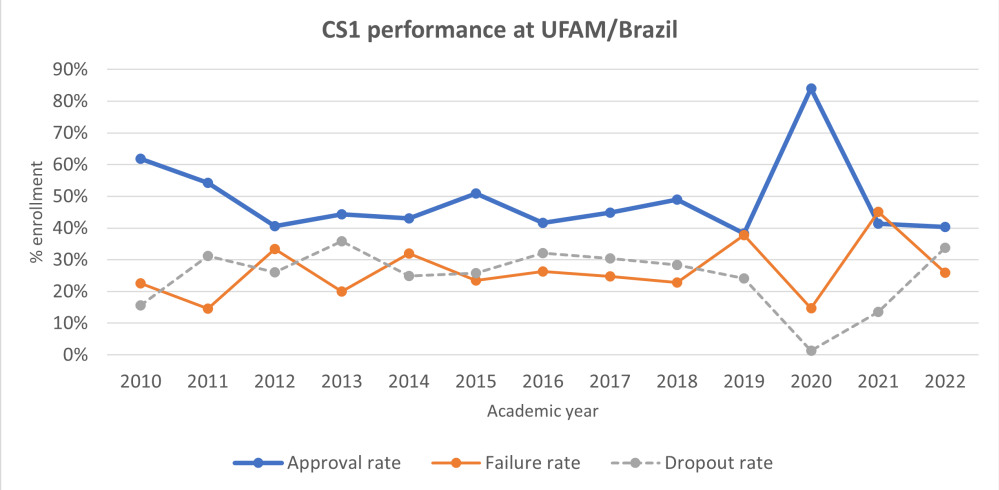

Figure 1: Evolution of CS1 performance over the last academic year at UFAM..

Figure 1 shows the evolution of approval, failure and dropout rates in CS1 at UFAM. Although the aforementioned actions appear to have had no effect on the approval rate, the dropout rate had decreased from 2016 to 2019. In 2020, dropout marks were removed from students' records due to COVID-19 pandemics. However, dropout rate have increased since then, in the last two years, which arouse new concerns among CS1 instructors team.

Thus, recently, they have focused on applying Learning Analytics (LA) techniques to monitor various aspects of the teaching and learning process. To achieve this effectively, they have

adopted an approach that involves using an online judge, collecting educational data, incorporating gamification, and applying LA in various contexts.

This text stands out from most articles as it does not focus on presenting specific research; instead, it serves as a comprehensive showcase of the predominant initiatives and outcomes within the field of Learning Analytics at UFAM, specifically in the context of introductory programming courses. After all these years of research, it is a good moment to summarize what has been done and the achievements so far. Given the importance of understanding the context in which our studies were conducted, we will commence by providing an overview of the current scenario of CS1 at UFAM, which will be presented in the next section. Then, we will delve into the the collection of educational data, the application of LA, and the integration of gamification. We will elaborate on each action using examples from publications and obtained results. Finally, we will provide concluding remarks and discuss potential avenues for future work.

2 Educational Context

Since the studies discussed in this article were conducted within the same educational setting, this section is dedicated to providing a comprehensive overview of the specific details concerning the course, the online judge and the collected data. To enhance the effectiveness of the Computer Science 1 (CS1) course at the UFAM, the team in charge of revamping the methodology recognized the need for a substantial change in the teaching and learning process. As a result, the team chose to prioritize key elements that could facilitate automated activities and evidence-based decisions. To this end, an online judge and LA were integrated to allow tracking of student progress and the adaptation of course content. Furthermore, gamification elements were implemented to engage students and motivate them to participate actively in the course.

This CS1 course is taught by the IComp to 17 undergraduate programs in STEM. The course is divided into seven modules, each one is a topic of introductory programming: (1) sequential structures; (2) simple conditional structures (if-then-else); (3) nested conditional structures (nested if-then-else); (4) while loops; (5) arrays and strings; (6) for loops; and (7) matrices. Each module is composed of: (a) teaching material from the lecture; (b) a formative assignment; and (c) a summative exam. CodeBench, our online judge, provides exercises for assignments and exams that require Python solutions (Carvalho et al., 2016).

CodeBench is an online judge developed at UFAM to automate the correction of programming exercises. In this system, instructors can provide programming exercises for their students, who can then code solutions and submit them through the system's interface (Figure 2). Once a student submits a solution for a given exercise, CodeBench checks if the solution produces the expected outputs for the given inputs and informs the student whether the solution is correct, partially correct, or incorrect.

In addition to automating the evaluation of programming exercises, CodeBench also streamlines the process of creating or selecting such exercises, thus saving instructors time. For this, the system maintains a repository with 14,947 programming exercises that instructors can select when creating assignments. Whenever a new exercise is added to the system, it becomes part of the repository and is immediately available to all other instructors. Therefore, CodeBench can

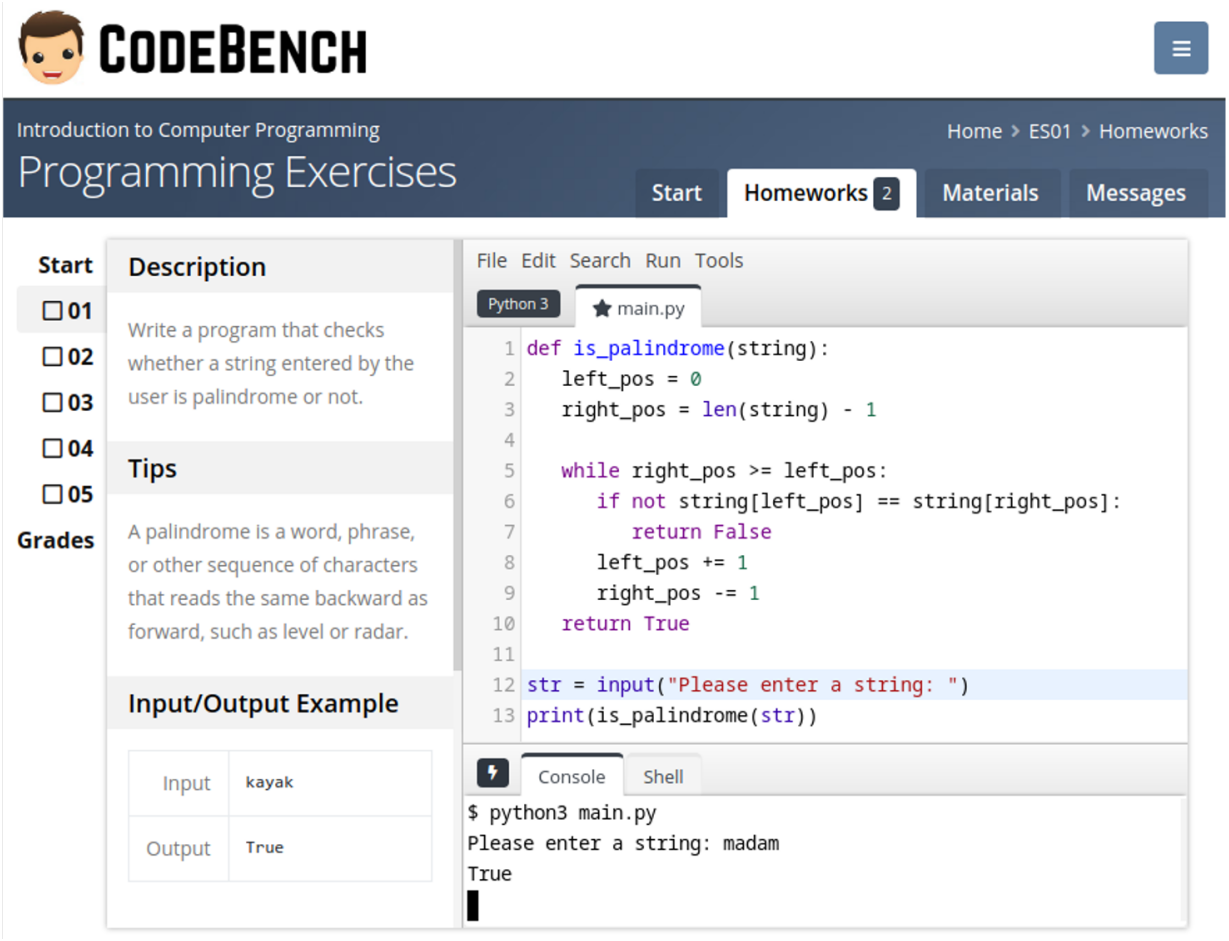

Figure 2: CodeBench screenshot.

also be considered a collaborative environment for creating exercises, with instructors benefiting from each other's work. Currently, the system has 8,085 users, including students and instructors from UFAM and three other partner institutions.

Data collection in online judge tools is essential for monitoring student progress and adapting course content. As mentioned, CodeBench provides its integrated development environment (IDE), designed to be simple and beginner-friendly, as shown in Figure 2. All actions performed by students in the IDE (e.g., keystrokes, submissions, code pasting, mouse clicks, tab or window transitions) are timestamped and recorded in a log file on the server side. Figure 3 shows an example of the log, from which we can extract variables to measure learner behavior. For example, we can identify the number of times a student attempted to solve a problem, the time it took to solve a problem, the number of errors made, the educational resources accessed, the number of logins and logouts made, as well as extract another set of variables related to the use of gamification.

The decision to develop our online judge was crucial to tailor it to our teaching needs and better understand the requirements for such a system. As CodeBench is an in-house tool, it can be modified to meet the needs of both teachers and students, as well as for ongoing research purposes. Furthermore, it can be used as a supportive tool in any teaching modality, whether traditional face-to-face, remote, or hybrid learning environments.

```
2017-3-13 17:28:44.965#viewportChange#0
2017-3-13 17:28:46.733#change#{"from":{"line":2,"ch":0},"to":{"line":2,"ch":0},"text":["p"],"removed":[""],"origin":"+input"}
2017-3-13 17:28:46.885#change#{"from":{"line":2,"ch":1},"to":{"line":2,"ch":1},"text":["r"],"removed":[""],"origin":"+input"}
2017-3-13 17:28:47.109#change#{"from":{"line":2,"ch":2},"to":{"line":2,"ch":2},"text":["i"],"removed":[""],"origin":"+input"}
2017-3-13 17:28:47.373#change#{"from":{"line":2,"ch":3},"to":{"line":2,"ch":3},"text":["n"],"removed":[""],"origin":"+input"}
2017-3-13 17:28:47.605#change#{"from":{"line":2,"ch":4},"to":{"line":2,"ch":4},"text":["t"],"removed":[""],"origin":"+input"}
2017-3-13 17:28:50.369#change#{"from":{"line":2,"ch":5},"to":{"line":2,"ch":5},"text":["()"],"removed":[""],"origin":"+input"}
2017-3-13 17:28:50.369#keyHandled#"'('"
2017-3-13 17:28:52.083#change#{"from":{"line":2,"ch":6},"to":{"line":2,"ch":6},"text":["\"\""],"removed":[""],"origin":"+input"}
2017-3-13 17:28:52.083#keyHandled#"'\"'"
2017-3-13 17:30:31.907#viewportChange#0
2017-3-13 17:30:56.332#mousedown#{"isTrusted":true}
2017-3-13 17:30:56.349#focus#
```

Figure 3: An example of part of CodeBench log.

The unprecedented Covid-19 pandemic brought a sudden shift towards remote learning, making CodeBench even more valuable in facilitating teaching and learning. With its online accessibility, students could continue learning remotely and submitting their solutions to the online judge for instant feedback. As such, CodeBench played an important role in ensuring the continuity of education in the face of the challenges presented by the pandemic.

So, in the next section, we describe an important part of all the researches – the CodeBench dataset. The data is collected since 2016 and enables a longitudinal analysis of CS1 courses in many ways.

3 CodeBench Dataset

In addition to facilitating the teaching and learning experiences of students and professors in programming courses at UFAM, CodeBench provides a public dataset[1] that is presented, for the first time, in a scientific journal so it can be used by any researcher interested in conducting research in the field of computer science education. The dataset contains the data generated from student actions attempting to solve the proposed exercises in the CodeBench IDE. This data is anonymized by the guidelines of the Brazilian General Data Protection Law (LGPD)[2]

Although CodeBench is a supporting tool for other programming courses, the available public dataset only provides data from UFAM's CS1 courses. The latest dataset release, CodeBench Dataset 1.7, contains all logs collected from CS1 students from 2016 to 2022, where each academic year is divided into two semesters. This release covers data from 107 CS1 classes, 4,511 students, 14,947 homework exercises, 1,908 exams, 375,173 student codes, and 3,860,565 code tests and submissions. These data have enabled researchers in computer science education and LA to conduct research on educational online judges (Carvalho et al., 2016), code plagiarism detection (Oliveira et al., 2021), continuous student authentication (Lavareda et al., 2020), prediction of learning and academic performance in programming (Pereira et al., 2017; Fonseca et al., 2019; Pereira et al., 2019; Pereira, Oliveira, et al., 2020; Pereira, Souza, et al., 2020), interpretation of predictive models (Pereira et al., 2021), recommendation systems (Pereira et al., 2022), gamification (Pessoa et al., 2019, 2021; Ribeiro et al., 2020; Pessoa, 2022), among others. In addition,

[1] Available at https://codebench.icomp.ufam.edu.br/dataset/.

[2] Available at https://www.gov.br/cidadania/pt-br/acesso-a-informacao/lgpd.

these research efforts contribute to the formation of new masters and doctors at the IComp at the UFAM.

Figure 4 shows the schema and organization of the dataset. Inside the root folder, there are folders for each academic semester, and each semester folder contains its class folders. For instance, the folder 2022-2 contains all data and logs generated during the second academic semester of 2022 for the classes 438, 439, 440, and so on.

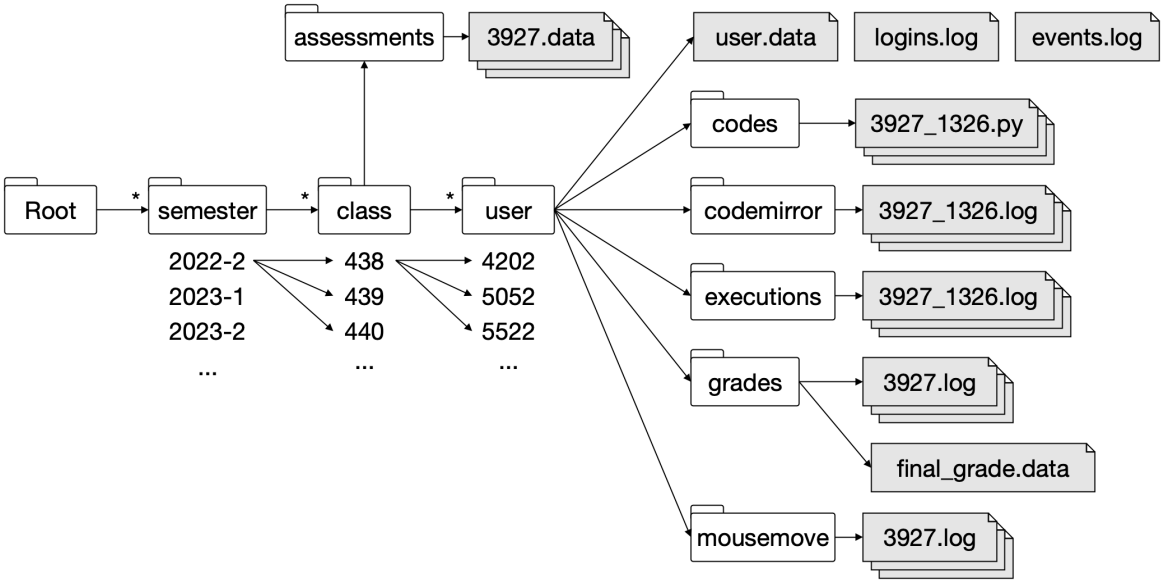

Figure 4: CodeBench dataset schema.

Within a class folder, the assessments folder contains data for all assessments recorded for that class in a given semester. There are two types of assessments: homework and exams. Homework are formative assessments consisting of programming problems aimed at helping the student master the course content. Such assessments have a minimal impact on the student's grade and can be completed in the computer lab or at home. On the other hand, exams are summative assessments administered and supervised by teachers while students solve programming exercises at scheduled dates and times. Unlike homework, CodeBench creates a unique exam for each student by randomly selecting questions from question blocks created by the teachers. Figure 5 illustrates the contents of the 3927.data file, which specifies the data and the list of exercises of the assessment 3927.

In Figure 4, the folders 4202, 5052, 5522, etc. represent students (users) enrolled in class 438. Each folder contains all the data and logs for the corresponding student. For each student, the user.data file stores historical and demographic data, the logins.log file keeps track of the student's login and logout information, and the events.log file records the student's navigation events in the CodeBench UI. Additionally, a student has the following folders:

- **codes**: which contains the final version of all the code (Python 3) developed by the student during the CS1 course. For instance, the file /codes/3927_1326.py is a Python file coded

```
-- ASSESSMENT DATA:
---- assessment title: Lab 0 - Getting Started with Python
---- class name: Introduction to Computer Programming
---- class number: 438
---- start: 2022-10-26 14:55
---- end: 2022-12-23 23:59
---- language: Python 3
---- codemirror mode: python/python.js
---- type: homework
---- weight: 1
---- total_exercises: 16
-- EXERCISES:
---- exercise 01: 994
---- exercise 02: 996
---- exercise 03: 1326
---- exercise 04: 1327
---- exercise 05: 1328
---- exercise 06: 1329
---- exercise 07: 1330
---- exercise 08: 1331
---- exercise 09: 1333
---- exercise 10: 1334
---- exercise 11: 1332
---- exercise 12: 1559
```

Figure 5: File content `3927.data`.

by the student 5052. The numbers 3927 and 1326 indicate the assignment and one of the exercises of this assignment, respectively.

- **codemirror**: the IDE integrated into CodeBench is based on CodeMirror, which is a code editor/component implemented in JavaScript. This component records events of various student activities such as code changes, mouse clicks on the IDE (focus), ctrl+c (copy), ctrl+v (paste), etc. Thus, each file in this folder contains the record of a student's activities while he or she was trying to solve a particular programming exercise. For instance, the file /codemirror/3927_1326.log contains all the keystrokes and mouse clicks made by student 5052 while coding the file /codes/3927_1326.py.

- **executions**: this folder provides the data (codes, errors, outputs, and execution time) of all the code executions students trigger during their CS1 course. CodeBench allows two types of code execution: tests and submissions. In tests, students can run their codes with their inputs to check if the outputs are by the problem requirements. When a student believes that their code is correct, they can submit it for automatic evaluation. The code is evaluated through a series of test cases that check if its output meets the exercise specifications. For example, the file /executions/3927_1326.log provides all the data of tests and submissions executed by student 5052 during the coding of file /codes/3927_1326.py.

- **grades**: contains data on the student's performance in the assignments of their CS1 course. For instance, the file /grades/3927.log contains the performance data of the student 5052 for

each exercise solved in assignment 3927. Unlike the files in this folder, the file /grades/final_grade.data provides the student's final grade, resulting from all the assignments solved.

- **mousemove**: provides data on mouse movements made by the student during the resolution of each exercise of each assignment. For example, the file /mousemove/3927.log contains the mouse movement data of the student 5052, for each exercise solved in assignment 3927.

Note that the logging of the dataset occurs at various levels of granularity: at the keypress level, from the files in the codemirror folder; and at the level of code tests and submissions, from the files in the executions folder. The finer-grained data provide opportunities for highly detailed investigations of the coding behaviors of students during the resolution of programming exercises.

Despite the dataset being limited to logs generated within UFAM, CodeBench is available to professors and researchers from other universities who are interested in conducting research in the field of computer science education. In this case, logs generated by students from those external universities can be shared with such researchers. All the LA conducted by our research group are based on these logs, with a particular focus on the following research topics: student performance prediction, exercise classification and gamification. The next three sections will cover these themes.

4 Methods

In this section, we present the research methodology that encompasses the three mentioned topics of research. It is composed by three main phases: data collection and preprocessing, experiments for selecting the best attributes, and analysis of results, explained next and shown in Figure 6.

1. Data Collection and Preprocessing
   (a) Data Sources: Initially, we collected interaction data from students in programming courses within an online learning environment - CodeBench - used as an educational support tool.
   (b) Data Preparation: The raw data collected was processed for cleaning and organization. This involved removing missing values, standardizing formats, and converting raw data into usable formats.
   (c) Feature Engineering: At that stage, we identified the most relevant attributes for our study. This included student performance metrics, exercise characteristics, and gamification data, as applicable.
2. Experiments for Selecting the Best Attributes
   (a) Data Split: The dataset was be divided into training and testing sets to assess the performance of the classification models.
   (b) Attribute Selection: We used attribute selection techniques such as importance analysis or filter methods to identify the most relevant attributes for the classifications.

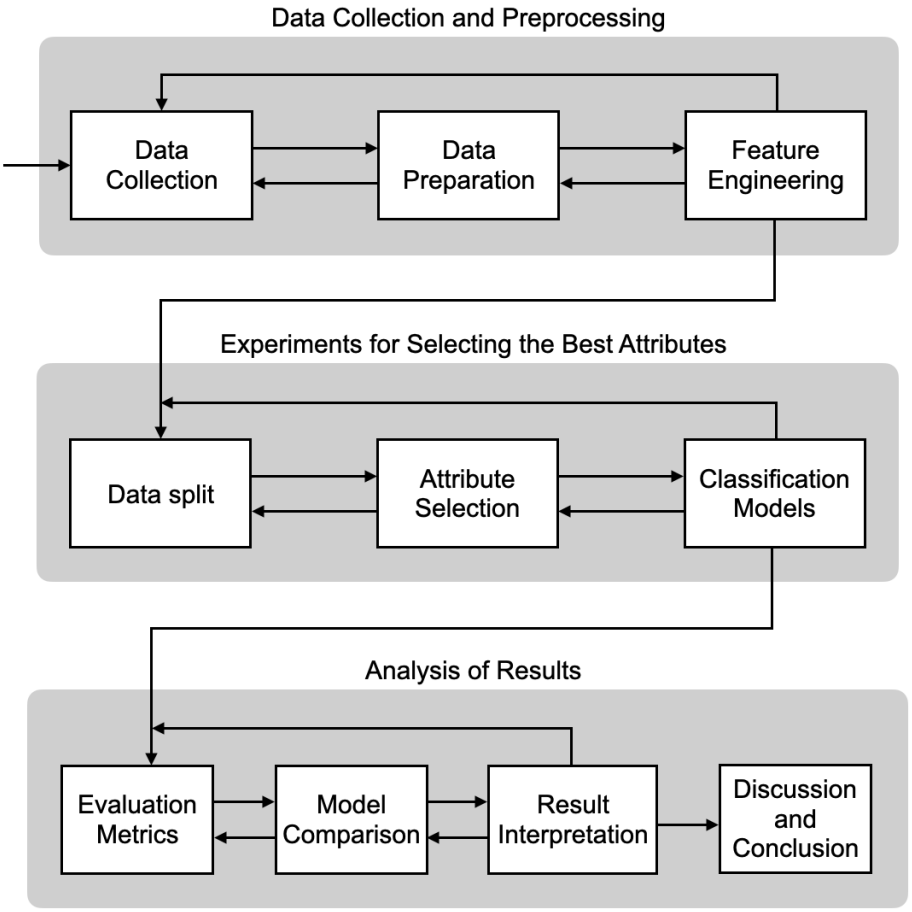

Figure 6: Research methodology.

(c) Classification Models: We implemented various classification models such as decision trees, logistic regression, and neural networks to test the influence of attributes for the classifications.

3. Analysis of Results

(a) Evaluation Metrics: We used evaluation metrics such as accuracy, recall, F1-score, and area under the ROC curve to assess the performance of the models in difficulty classification.

(b) Model Comparison: We compared the performance of different models and attribute selection techniques to identify the most effective approach.

(c) Result Interpretation: We analyzed how the chosen attributes influence the performance prediction, the difficulty classification and the impact of gamification on student performance.

(d) Discussion and Conclusion: Based on the results, we discussed practical and theoretical implications, highlighting the study's limitations and providing directions for future research.

This methodology allowed for a systematic approach to collect, prepare, analyze, and interpret data related to Learning Analytics on the topics of student performance, exercise difficulty classification, and gamification in an educational context.

5 Learning Analytics for Student Performance Prediction

The high rate of failure and dropout in programming courses, particularly in STEM fields, is a serious and recurrent problem that institutions and educators must address. To tackle this issue is crucial to understand the causes and work towards possible solutions that provide information and recommendations to help teachers mitigate the difficulties students face. To achieve this, researchers and educators commonly encounter questions such as:

- Which students are struggling with the course material?
- What topics or exercises do they find most challenging?
- Which exercises are best suited to each student's needs?
- Are students engaging in unacceptable practices, such as plagiarism, when completing their assignments?

To answer these and other important questions, a research group at our institution employs various methods, including predictions, descriptions, interventions, recommendations, and personalization. By analyzing data collected from students in programming courses, such as their performance on assignments and their interaction with our online judge system, we can gain insights into their learning behaviors and tailor our teaching methods accordingly. For instance, if many students struggle with a particular concept, we promptly modify the course material to provide additional support and guidance. Additionally, by identifying students who may be at risk of dropping out, we can intervene early to provide targeted support and resources to help them succeed.

One of the primary research focuses of our group was the utilization of LA techniques for predicting student performance. The existing literature indicates numerous studies on prediction introductory programming course performance (Ihantola et al., 2015; Robins, 2019; Carter et al., 2019). However, many of these studies only use variables based on the number of attempts made by students and the correctness of the submitted questions (Pereira, Souza, et al., 2020). While these two variables are useful, research points to the need for more granular studies (Carter et al., 2019), such as the ones we used in our programming profile. Each student's programming profile was generated based on attributes extracted from the online judge logs. Over 30 attributes were extracted, including:

- attempts: the average number of submissions per exercise;
- changes: the number of code changes between submissions;

- correctness: number of problems correctly solved from programming assignments before the first exam;
- procrastination[3] : the practice of postponing the start or completion of solving exercises of an assignment;
- propPaste: the proportion of pasted characters compared to typed characters;
- qtdEvents: the number of events during development;
- syntaxError: the proportion of submissions with errors.

After the selection of attributes and classification models, we proceed to present the evaluation of metrics and engage in a discussion of the results in the following section.

5.1 Results and Discussion

In fact, by combining the number of attempts and correctness with highly granular variables, our predictive models achieved an F1-Score of approximately 80% (*Accuracy* $\approx$ 81%, *Recall* $\approx$ 82%, *Precision* $\approx$ 79%) in predicting student performance using only data from the first 2 or 3 weeks of class (Pereira et al., 2021). Furthermore, the predictive models become increasingly accurate as more data is accumulated from subsequent weeks (Pereira et al., 2022). To illustrate, by the sixth week, our predictive model had achieved an F1-Score of approximately 90%. It is important to note that the information provided by the predictive model about the likelihood of student approval can be combined with the experience and observations of the teachers in the classroom, further enhancing its predictive power.

Also, as part of our initiatives, we utilized LA techniques to model the fine-grained data and identify early effective programming behaviors that could guide ineffective students (Pereira et al., 2017, 2019; Fonseca et al., 2019; Pereira, Fonseca, et al., 2020). In order to understand the behaviors of CS1 students, we aimed to address the following two goals: (i) detect early effective and ineffective behaviors of CS1 students using data from an online judge; and (ii) understand which effective behaviors of novice students can be utilized to guide students showing ineffective behaviors.

We define "effective students" (Cluster A) as those who progress in learning to program, leading to successful outcomes, while "ineffective students" (Cluster C) are those who struggle to make progress or require excessive effort, resulting in unsuccessful outcomes. We also categorize students who fall between the spectrum of effectiveness and ineffectiveness as "intermediate students" (Cluster B). Figure 7 shows that the student's performance is directly related to their behavior on the online judge (Pereira, Oliveira, et al., 2020).

Furthermore, when drawing parallels with other relevant studies that share the common goal of early performance prediction in introductory programming, we can observe varying levels of success. For instance, (Leinonen et al., 2016) achieved an accuracy of 65.8% using data from 226 students within their first two weeks of the course. In an extension of this work, (Edwards et al.,

[3]Notice that the procrastination can be related to a student's difficulty with the content of the course. Additionally, it may also be associated with factors such as a lack of interest in the subject and/or a lack of organization skills.

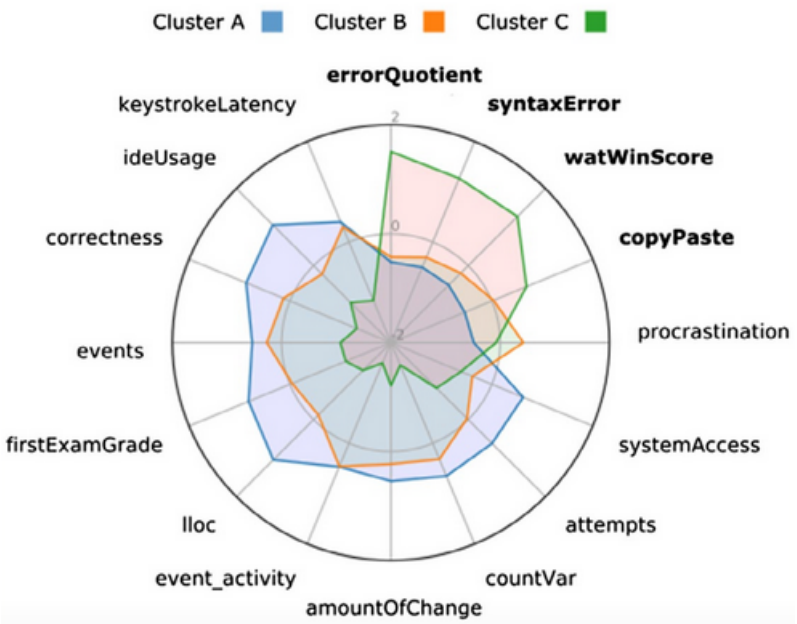

Figure 7: Programming profile in each cluster .

2020) improved the accuracy to 72% with a larger dataset. Meanwhile, (Castro-Wunsch et al., 2017) attained an accuracy of 71.81% using data from 897 learners within their first four weeks, with the first two weeks' performance not reported. (Quille & Bergin, 2019) reached an average accuracy of 71% using early data from 692 students, while (Tomasevic et al., 2020) achieved 78% accuracy.

Notably, our results surpass all related works for early performance prediction. It is worth noting that studies such as (Jadud, 2006) and (Watson et al., 2013), which relied on a single feature, achieved lower accuracy rates (<66%) when using data from the entire course. A comparable result to ours can be seen in (Costa et al., 2017), who achieved a similar result (80%), although they utilized demographic features. While demographic attributes might be reliable predictors, stakeholders have no control over student demographics, making them less practical in the context of our work, for which behavioral attributes prove more valuable, as they allow stakeholders to encourage effective behaviors and guide ineffective ones.

It is noteworthy that some of the proposed predictive models have explainable decisions. This enables the identification of potentially effective programming behaviors, characterized by the variables in the programming profile, that can increase or decrease students' chances of passing the course. To explain the models, we employed the SHAP framework (Lundberg & Lee, 2017), which allows for the explanation of each decision made by the predictive model. Using SHAP in our most accurate predictive model, an important finding of our research was the ability to observe and highlight students' individuality, meaning that different variables may have more or less influence on the prediction for each student. For example, the procrastination variable might be more or less harmful for different student profiles. Some students who leave their assignments until the last minute can still performance in the course, although this is rare. For such students, a different intervention strategy may be possible. We aim to encourage effective learning behaviors and discourage ineffective and harmful ones. Our explainable predictive model (Pereira et al.,

2021) support this.

Although explainable predictive models can provide valuable information for decision-making, it is important to remember that each student is unique, and not all behaviors are easily modifiable. When procrastination is a recurring problem in the class, it may be more effective for the instructor to suggest that students invest more time and attention in the activities. On the other hand, aspects such as keystroke latency may be more challenging to modify, and therefore interventions in this regard may not be as effective.

Furthermore, each student learns to program at their own pace, so ideally they should be challenged to learn as much as possible while considering their limitations and strengths in learning. To illustrate, with our performance prediction model (Pereira, 2022), it is possible to identify students who quickly complete the exercise list with low effort. These students may feel bored or even frustrated. One potential solution would be to recommend more challenging programming activities to keep these students engaged in the course (Pereira et al., 2022).

Observe that a student may begin the course with success but could still fail for various reasons. Conversely, a student may start with difficulty but then succeed. By estimating student performance over time, our model can identify inflection points – moments where a change occurs in performance prediction. For instance, a student may have a 90% chance of passing by the end of the third week but only a 50% chance by the end of the fifth week. This inflection point can be communicated to the instructor so that they can intervene. This concept is also useful for examining the impact of methodological interventions that teachers may implement during the course. For example, a teacher may experiment with a new pedagogical approach from the fourth to the sixth week and then compare predicted student performance at both points.

Additionally, a visualization panel containing easily interpretable graphs on students' chances of passing and their effective and ineffective behaviors can be a valuable resource for instructors and coordinators to more informatively evaluate the process of code production and the learning path of students, not just limiting themselves to the final product delivered. With this tool, it is possible to obtain insights on students' performance and effort throughout the course.

6 Learning Analytics for Classification of Difficulty of Programming Exercises

In general, online judges offer self-correcting programming code-writing exercises. Each exercise consists of three parts:

1. *statement*: this is what is shown to students: text in natural language, equations, images, videos, among other elements (Pelánek et al., 2022). The statement can also be decomposed into: specification, tips, example test cases (inputs and corresponding outputs) and support files;

2. *test cases*: these are ordered pairs <input, output> used to test solution codes submitted by students, but hidden from them;

3. *solution template code*: this is a reference code, valid for all test cases in the question. This

code is developed by the instructor and entered into the system for documentation purposes when creating the question, and is not used to correct the student's solution code. Therefore, it is not accessible to the student.

Once registered in the online judge, the exercise is immediately available to be used in activities for students, who will try to answer it by writing, testing and submitting code, using the Integrated Development Environment. Log data is generated from the student's interaction with the exercise, such as typed and deleted characters, successful and unsuccessful submissions, date and timestamps, and activity and inactivity time in the IDE, among others. This data is stored in the online judge and constitutes a complementary set of information associated with each exercise.

A common task instructors perform within online judges is the creation of assignments. These need to be carefully planned so that they are properly balanced, with questions arranged in order of increasing difficulty, to promote a smoother learning curve (Effenberger, Cechák, & Pelánek, 2019). For this, the online judge should inform the degree of difficulty of the questions already resolved and estimate the difficulty of new questions based on the similarity with others already resolved (Effenberger, Cechák, & Pelánek, 2019). However, how to define "difficulty" in more objective terms? What characteristics of a coding problem (statement, test cases, instructor template code, interaction data) contribute to greater "difficulty" or "complexity"? Can the terms "difficulty" and "complexity" be used interchangeably? It is necessary to clarify these two concepts before formally presenting the objectives of this work.

Liu and Li (2012) found and systematized the variety of understandings about the complexity and difficulty of human tasks, in different contexts, and the educational context. They propose the following distinction: "task complexity involves the objective characteristics of a task, whereas task difficulty involves the interaction among task, task performer, and context characteristics". Beckmann et al. (2017) propose a similar differentiation: "complexity is conceptualized as a quality that is determined by the cognitive demands that the characteristics of the task and the situation impose. Difficulty represents the quantifiable level of a person's success in dealing with such demands".

From these works, Sheard et al. (2013) and Pelánek et al. (2022) instantiate these two concepts in developing learning systems. According to them, the "complexity" of a question is an intrinsic characteristic of the question item, adding aspects of the item that influence how students solve it. For example: extension of the statement, number of programming concepts worked on, number of control structures needed for the solution, cyclomatic complexity, among others. The "difficulty" describes the students' effort to solve a question. In programming questions, some difficulty metrics used are the following: rate of incorrect (or correct) answers, median response time, number of attempts, the proportion of use of hints, and number of code edits to build the answer, among others.

An easy way to discern the concepts of "complexity" and "difficulty" is by looking at the nature of the data used to calculate a specific measure (Pelánek et al., 2022): if only the question description is used, then we have a complexity measure; on the other hand, if data from the record of student interaction actions (log) with the question are used, then we have a measure of difficulty.

In (P. Santos et al., 2019), the authors analyzed only the descriptive text of the utterance programming issues as a source of complexity measures. Considered the following complexity

variables: Flesch index of readability (ease of reading) and the incidence of adjectives, adverbs, pronouns, verbs and logical operators.

As difficulty metrics, the following variables were used:

- **mean number of executions**: the average testing students have done of the solution code, using only the IDE console, without submitting to the judge's automatic correction online.
- **mean number of submissions**: average number of attempts by students to submit your code for online judge autocorrection.
- **success rate**: ratio of the number of students who answered the question correctly to the number of students who received the question in assessment activities.
- **mean solution time**: average interaction times of each student with the IDE during the solution of a given question.

In this analysis, programming questions used only in face-to-face assessments were considered to reduce a plagiarism bias in the sample. Face-to-face assessments are carried out in a controlled environment: only the MAC address of the machines in a given teaching laboratory can access the assessment in the online judge. Furthermore, the presence of an instructor and a tutor inhibits cheating behavior. Questions that at least 30 students had solved were also considered to reduce the bias of inadequate sample size. In this way, it is more certain that the extracted measures corresponded more to a description of the questions than individual student behaviors.

The variable "success rate" was broken down into five categories, according to the "ease index" classification adopted by the Brazilian National Institute of Educational Studies and Research Anísio Teixeira (INEP) in the analysis of the results of the National Student Performance Examination (Enade) (INEP, 2017), according to Table 1.

Table 1: Classification of questions by ease index. Fonte: (INEP, 2017).

Success rate	**Labels**
$\geq$ 0,86	Very easy
0,61 a 0,85	Easy
0,41 a 0,60	Medium
0,16 a 0,40	Difficulty
$\leq$ 0,15	Very difficulty

As a result, in (P. Santos et al., 2019), it was found that the difficulty variable "success rate" was the one that could be classified most accurately (75%). However, low or insignificant correlations were found between the textual intelligibility variables (complexity) and the difficulty variables.

Some more recent works sought to explore metrics extracted directly from the solution code to estimate the difficulty of questions. Therefore, we followed this research trend by exploring the potential of predicting the difficulty of programming questions based on the solution code model registered by the instructor in our online judge.

In this way, the complexity of the questions, expressed in terms of software metrics extracted from the instructor's model solution code, can be used as a predictor of the difficulty of the questions, expressed in terms of the success rate.

In (Lima et al., 2020), we collected a set of 91 attributes of model solution codes from a total of 354 programming questions used in face-to-face exams. Unlike the works mentioned above, the extraction process was automated using the Radon and Tokenize libraries of the Python programming language. In addition to enabling large bases, it mitigates possible errors caused by manual attribute extraction.

The set of attributes, which express part of the complexity of each question, was used to predict the difficulty of the questions, defined again as a function of the success rate. While each set was extracted from the code or derived from attributes extracted from the code, they can be categorized into two groups:

- *attributes based on code size*: describe the macro characteristics of the code, such as: the number of lines of code, number of logical lines, number of blank lines, number of lines with comments and cyclomatic complexity.
- *attributes based on lexical analysis*: describe the code from extracted *tokens*, such as: keywords, identifiers, strings, numeric values and operators.

Again, we employed the "ease index" classification adopted by INEP (see Table 1). However, the variable "success rate" has now been discretized into just two categories, modeling the difficulty predication problem as a binary classification problem, as shown in Table 2.

Table 2: Discretization of success rate for difficulty.

Success rate	**INEP ease index**	**Binary labels**
$\geq$ 0,86	Very easy	Not difficulty
0,61 a 0,85	Easy	
0,41 a 0,60	Medium	
0,16 a 0,40	Difficulty	Difficulty
$\leq$ 0,15	Very difficulty	

Building upon this, we will present the results in the next section, which are based on binary classification, and engage in a discussion about them.

6.1 Results and Discussion

As a result, in (Lima et al., 2020), we estimated the exercises' difficulty expressed by the binary discretization of the "success rate", with 95% accuracy and 81% f1-score. For this, we employed an ensemble of classifiers combined in two levels by means of stacking (Wolpert, 1992).

Although the whole set of questions comes from face-to-face exams of the same course, there is a natural and gradual evolution in the level of difficulty or easiness of the questions. This

occurs because, as the course progresses, new programming concepts are taught, thus increasing the complexity of the solution codes.

As the syllabus of the CS1 course is divided into seven modules, in (Lima et al., 2021b) we sought to explore the classification of questions within each module. For this, the hit rate was discretized regarding the ease of programming questions within each course module, as shown in Table 3.

Table 3: Distribution of Questions for Binary Classification.

Module	**# Features**	**# Problems**	**Labels**	
			Easy	**Not easy**
M01	56	66	84,8%	15,1%
M02	65	59	86,4%	13,5%
M03	72	58	72,4%	27,6%
M04	73	42	23,8%	76,2%
M05	74	44	56,8%	43,2%
M06	83	33	66,7%	33,3%
M07	81	52	69,2%	30,8%

This stratification of the set of questions made it possible to understand that the set of most important attributes for the classifiers varies according to the course's modules. Therefore, trying to classify the whole set of questions is a more arduous task than when we stratify by modules, as this way we group more similar questions, eliminating noise.

In addition, it was possible to identify that as the course progresses, the number of "easy" questions decreases and, on the other hand, the number of "not easy" questions increases. It was also possible to observe that Module 04 (while loops) was the only one that had a lower number of questions labeled "easy" than the number of questions labeled "not easy". Thinking about the success rate, this means that students need some help dealing with questions about repetition structures for the first time.

As a result, it was possible to classify questions stratified by module with a final accuracy of 92% and f1-score of 94% using the same stacking technique. The results obtained exceed those found in (Lima et al., 2020) indicating that the stratification of questions improves the classification process by grouping similar questions. Finally, stratification also allows the use of a smaller amount of features, thus reducing the computational cost.

According to (Effenberger, Čechák, & Pelánek, 2019), estimating the difficulty in programming questions requires a sufficient amount of data. Each question must be presented to a reasonable number of students so that the metrics used to represent the difficulty are valid. This stems from an issue with few solution attempts that could present biased and unrepresentative metrics. The optimal value depends on the chosen metric to represent the difficulty.

In this sense, in (Lima et al., 2021a) we sought to evaluate the behavior of the success rate with different trial filters (thresholds). Analyzing Figure 8, it is observed that by decreasing the threshold value, that is, accepting questions with few attempts, we add noise to the distribution, compromising the representativeness of the success rate values. On the other hand, when we increase the threshold value, we have distributions with less noise, but increasingly reducing the sample size. Therefore, based on the distribution curves, the value 16 was adopted as a threshold for reducing the bias of the correct answer rate, still maintaining 405 questions from the initial

sample.

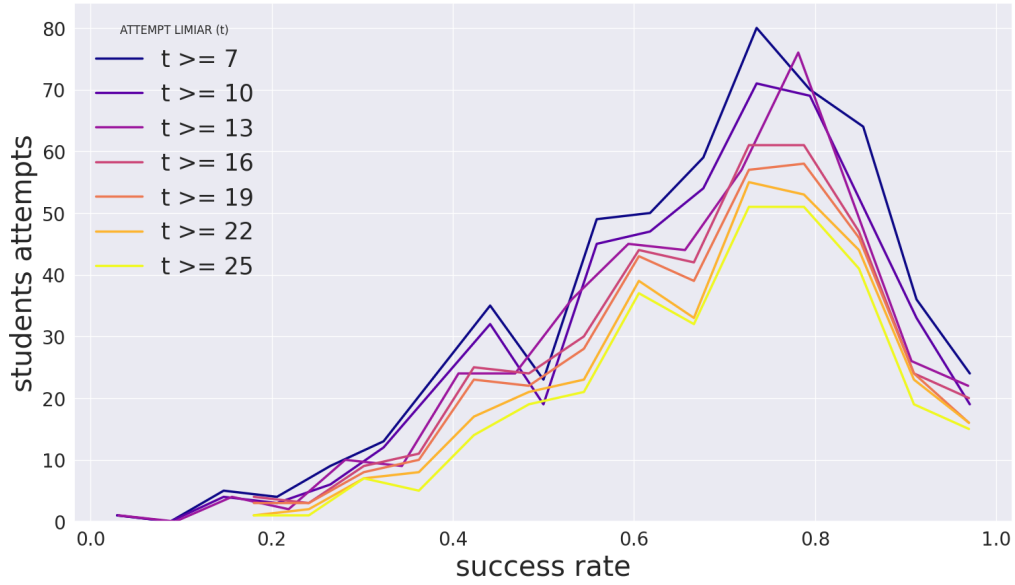

Figure 8: Success rate curves by students attempts.

Although often used to express the difficulty of programming questions, the success rate is not the only viable metric (Elnaffar, 2016; Whalley & Kasto, 2014; Effenberger, Čechák, & Pelánek, 2019). In this sense, in (E. S. Silva et al., 2022) nine metrics (indicators) were explored to express the difficulty of programming issues:

- *success rate*: the percentage of students who answered the question correctly;
- *implementation time*: median time spent by students to solve the question;
- *number of executions*: median number of times students ran their own test solution;
- *number of submissions*: median number of times students submitted their solution code for automatic correction by the online judge;
- *number of queries*: sum of total plays and commits;
- *error query rate*: percentage of queries (executions and submissions) that presented an error (syntactic or logical);
- *interaction events*: median of the total number of events (student interactions) recorded in the log files during question resolution;
- *removal events*: median number of total text removal events in the solution code (delete or backspace keys pressed);
- *change events*: median of total number of student solution code change events;

The same approach as in previous works was used for the prediction of difficulty, discretizing the metrics into two classes, thus transforming the prediction into a binary classification problem. Except for the success rate and the rate of queries with errors, all other metrics use the median as a centrality measure and a discretization threshold. This stems from the fact that the distributions of these metrics tend to be asymmetrical. Although they have a lower limit (zero), there is no defined upper limit. For example, a student can make multiple submissions for a question.

Additionally, an analysis of the Spearman correlation between the metrics was conducted so that it was possible to understand how they behave and also identify metrics that could be discarded. Among the observations, moderate negative correlations were found between the "success rate" and the "number of executions" ($-0.58, \rho \ll 0.001$), which makes it possible to understand that the questions with the highest hit rate also present few executions carried out by the students.

Table 4 presents the results of the classifiers for each difficulty metric sorted by f1-score and accuracy. It is observed that although the "success rate" was still the metric with the best results, we also have good results for the metrics "rate of queries with error", "implementation time" and "interaction events". These results are significant as they show that other variables can express portions of the effort required by the question, thus complementing the instructor's vision and understanding when preparing the question.

Table 4: Classification results by difficulty metric.

Difficulty metric	**Classifier**	**F1-score**	**Accuracy**
Success rate	SVC	.920	.852
Error queries rate	SVC	.751	.783
Implementation time	XGBoost	.732	.741
Interaction events	XGBoost	.725	.746
Remove events	LinearSVC	.690	.717
Change events	XGBoost	.642	.654
Executions	XGBoost	.618	.662
Queries	XGBoost	.608	.630
Submissions	XGBoost	.560	.743

Aiming to explore the metrics of difficulties in regression models, Fernandes et al. (2023) conducted a study addressing the classification of questions into two and three levels and using regression models. The difficulty metrics (dependent variables) set in (E. S. Silva et al., 2022) were expanded to thirteen metrics. For the binary classification models, the metrics were again discretized using the median, except the success rate, which continued to be discretized according to the "INEP ease index". For the ternary classification, the discretization was based on tertiles so that the first, second, and third tertiles correspond to the labels "easy", "medium", and "difficult". For the regression models, three metrics were used: mean absolute error (MAE), relative absolute error (RAD), and the adjusted coefficient of determination (R^2_{adj}).

For regression, the "implementation time" metric showed the best results for the adjusted coefficient of determination ($R^2_{adj} = 0.63$) and also the lowest absolute relative error ($RAE = 0.54$). In contrast, lowest mean absolute error values were obtained with the metrics "correctness rate" ($MAE = 0.08$) and "success rate" ($MAE = 0.09$). For the binary classification, the variable with the best result was the "success rate", with an *f1-score* of 88%. For the three-level classification, the best metric was "implementation time" with an *f1-score* of 67%.

Although the two-level classification obtained better results, analyzing the confusion matrix for the "success rate" in Figure 9, it is observed that there were more false-negatives than true-negatives for the "difficult" class, which shows that the model was not able to generalize well, probably due to the imbalance between classes, thus tending to classify most questions as "easy". On the other hand, the confusion matrix for the "implementation time", although it obtained a lower accuracy, presented a better performance when we observed the "difficult" class.

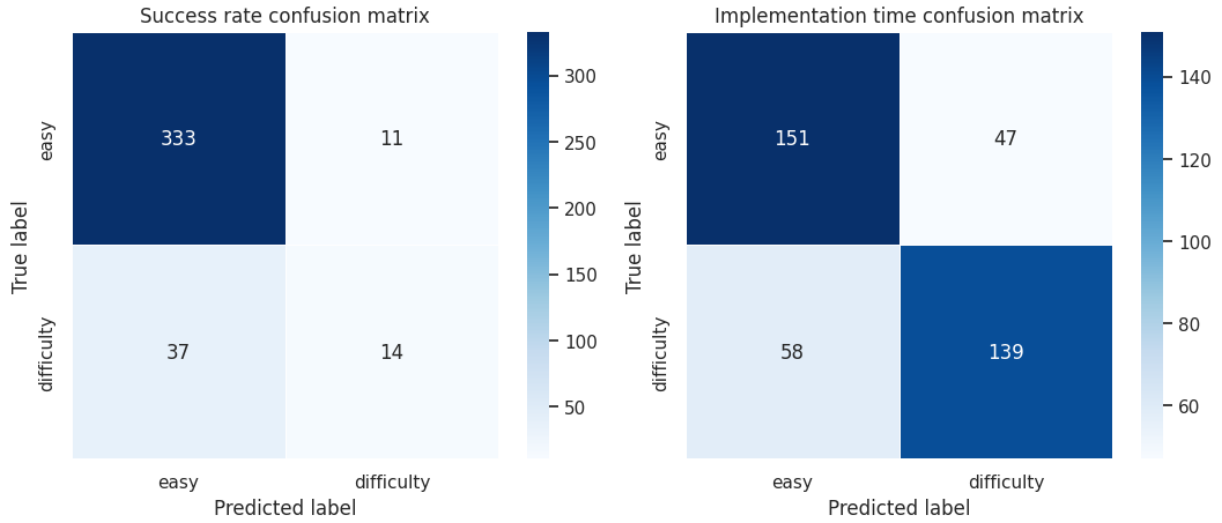

Figure 9: Confusion matrix for success rate and implementation time.

Finally, the evaluated difficulty metrics are adequate for a two-level classification. At the same time, they are still unfeasible for a three-level classification (much more expressive when we think of the instructor), considering the analyzed data set.

7 Learning Analytics for Gamification

Game Learning Analytics (GLA) is a branch of the LA field that analyzes data from educational games, with the main focus being evaluating learning outcomes (Freire et al., 2016). Although GLA is primarily focused on serious games, some researchers (Klock et al., 2018; Maher et al., 2020) have explored the use of data from gamified systems. For example, Klock et al. (2018) presented the use of LA techniques incorporated into a gamified environment based on game elements implemented within the system, such as points, challenges, leaderboards, and donations. The techniques enabled students to visualize their progress and performance to increase user satisfaction and engagement. Following the customization approach, Maher et al. (2020) demonstrated how LA techniques were applied to adapt gamification to the preferences of individual students, i.e., the content was designed and reorganized to meet the individual needs of each student.

One of the reasons for using GLA in useful applications is to measure the effect of interventions on the target audience. For instance, although educational games are widely used tools, there are still challenges regarding their evaluation to determine whether they are genuinely effective and capable of assisting learning. Most game evaluations are based on pre- and post-test ques-

tionnaires, which may need to be revised to understand whether learning has occurred (Calderón & Ruiz, 2015), thus requiring more robust evaluations. In this context, the possibility arises of using interaction data generated by people during game or gamification use to conduct analyses (Alonso-Fernandez et al., 2017).

Among the possibilities for analysis in the GLA field are: player profiles, similar player groups, dropout rates, active players, performance prediction, learning evaluation, engagement, game design validation, feedback effects evaluation, and dashboard creation (Alonso-Fernandez et al., 2017; D. Silva et al., 2021).

Given this context, our research group developed a gamification platform called CodePlay. It incorporated it into our online judge system, CodeBench, to enhance student engagement and improve learning outcomes. However, to ensure the platform is compelling, evaluating its impact on student performance (Pessoa et al., 2019; Pessoa, 2022) is crucial. Data analysis techniques can be implemented to assess how gamification influences student behavior and learning outcomes.

To conduct research involving the CodeBench gamification platform, all student interactions with the platform are captured through a logging system. Some examples of game elements with which students can interact are:

1. Coins: players can earn coins as a reward for answering exercises correctly, or by unlocking locations or missions.
2. Experience Points (XP): players can earn experience points as a reward for answering exercises correctly or unlocking locations or missions.
3. Ranking: players can access the ranking to view the performance order of all students in their class in decreasing order of XP.
4. Infobox: when hovering over a player's avatar, an information box about the player can be displayed, including coins, XP, and weapons.
5. Secret passages: There is at least one secret passage in each game stage. Finding these secret areas is optional for the player, who will receive rewards if they find them.
6. Easter eggs: the game has some Easter eggs (surprise elements in random locations) that the player can find by exploring the environment.
7. Progress bar: the game has progress bars that show completed missions, unlocked locations, secret areas, and found Easter eggs.
8. Minimap: the game has a minimap that users can use to view their location or find other locations on the map.
9. Messages: the game allows users to send messages to others users.
10. NPCs (non-player characters): the gamer has several NPCs with which the player can interact. NPCs provide hints about the game's plot, missions, and objectives.
11. Donations: the game allows users to donate coins, points, and XP to others players.

12. Diary: the game has a diary containing the main and secondary missions of the stage the player is on.
13. Item menu: the game has an item menu (weapons, armor, and potions).
14. Equip menu: the game has an equip menu for the player to equip their character with weapons and armor.
15. Sound: the game can turn the sound on/off.
16. Player menu: the gamer has a player menu where donations or messages can be sent by selecting a colleague.
17. Other options menu: the game has a menu with keyboard shortcuts, teleport, minimap, minimap legend, and help options
18. Glasses: the top three players in the game (who accumulate the most coins and experience points) are rewarded with glasses to stylize their avatars.
19. Anarchist choices: the game has anarchist options (that can harm other players, or just identify flaws in order to help).
20. Secondary missions completed: besides main missions, the game has optional secondary missions in each phase.
21. Group: other game has three honor groups (gold, silver, and bronze), and each player's group is defined by the experience points accumulated during the phases.

The log system provides data that enable research, mainly in two areas: (i) computer education: impact of gamification on the performance and behavior of students in programming courses, and (ii) gamification: user profile analysis, identification of profiles based on usage data of game elements, identification of most and least used elements, in the overall context and by user profile; analysis of game element preferences by user profiles, among others.

Table 5 presents how the data is stored when captured in the gamification integrated with the CodeBench online judge. It can be observed that the log system starts by registering the date and time of each action in the system, the user identifier, the class identifier that the student is enrolled in, the name of the event, and, lastly, details related to the event. The event name is usually associated with using a game element in gamification. For example, in the first row of data in Table 5, the user with identifier number 6994, belonging to class number 387, hovered over their user avatar. The same row has other information about the event, such as the game map, location on the map, and position on the screen. It is important to note that all data is anonymized, thus complying with the LGPD.

As shown in Table 5, having a logging system and capturing all user actions in gamification is essential but insufficient. To conduct meaningful analyses using gamification data, it is necessary to process the captured logs to represent the presented game elements accurately and to uncover hidden patterns. This will enable researchers to conduct experiments of interest to the GLA field. Therefore, data preprocessing and preparation are crucial steps in making gamification data useful for GLA.

Table 5: Examples of logs captured in CodePlay.

Logs do CodePlay
2022-05-19T16:02:43.160 6994 387 infobox {"userId":6994,"posX":10,"posY":51,"map":1,"mapRoot":1}
2022-05-19T16:21:00.444 7004 387 saveMoedas {"amount":5,"type":"gold","switch":5,"map":71,"mapRoot":1}
2022-05-19T16:26:03.503 7004 387 minimapa {"map":1,"mapRoot":1}
2022-05-19T16:27:51.772 7004 387 npc {"npc":"Nashao_Freiheit","eventId":7,"page":2,"map":1,"mapRoot":1}
2022-05-20T12:12:45.611 2841 371 quest {"id":1,"step":1,"end":false,"map":1,"mapRoot":1}

One useful application of GLA is to offer personalization or customization (Klock et al., 2020), allowing the environment to adapt to user profiles or for users to customize it themselves. In the literature, several proposals for classifying users (Bartle, 1996; Barata et al., 2014; Nacke et al., 2014; Tondello et al., 2019) divide players according to their preferences and behaviors within the environment. Specifically in gamification, the Hexad model (Marczewski, 2015) classifies users into six types: philanthropists, socializers, free spirits, achievers, players, and disruptors. These user types are motivated by different aspects of the useful application. For example, achievers enjoy completing challenges and gaining new skills, while socializers prefer to use chats and interaction mechanisms with other participants.

Based on the Hexad user types, an analysis was conducted on using game elements using data captured in CodePlay (Pessoa et al., 2021). The study was conducted during the first semester of 2021 with IPC students at UFAM. Due to the Covid-19 pandemic, classes were held remotely. Therefore, only students who interacted with at least one element of the game's second phase, which students gain access to after a month of class, were considered in the study. The students also answered the Hexad questionnaire to identify their player profiles.

To analyze the usage of game elements, 22 study variables were selected to reflect the elements above, including:

- The total number of times players accessed the progress bar, the diary, the ranking, the equipment menu, the items menu, the player's menu, and the other options menu.
- The number of times players turned on the sound, made the minimap visible and received recognition through their glasses.
- The total sum of experience points (XP) and coins.
- The total number of interactions with NPCs (non-player characters), messages sent to classmates, and messages received from classmates.
- The total number of items donated and received, completed secondary missions and secret passages found, Easter eggs discovered, anarchist choices made, and times players hovered their mouse over another player's avatar to obtain information (infobox).

Since the variables have different value ranges (for example, the total number of times players interact with NPCs is much larger than the number of messages sent and received, the data was normalized using the Z-score normalization strategy, which scales the data to be between zero and one. Building on this, we will now proceed to present the results and provide some discussion regarding them.

7.1 Results and Discussion

To be classified with a single dominant profile, a user must obtain the highest score in one of the six classes (Free Spirit, Achiever, Player, Philanthropist, Socializer, and Disruptor). However, sometimes the user may achieve the highest score in multiple classes. For the study, only students with a single dominant profile were considered, resulting in 72 students.

Among the 72 students, none were identified as Disruptors. The distribution of player types was as follows:

- Free spirit, n = 14 (19%);
- Achiever, n = 23 (32%);
- Player, n = 14 (19%);
- Philanthropist, n = 17 (23%);
- Socialiser, n = 5 (7%).

In order to analyze the usage of the elements, the students were grouped according to their Hexad player type. Figure 10 presents a graph depicting the relationship between the elements and the player types. Visually, it can be observed that there are differences in the behaviors of the player types.

According (Marczewski, 2015), achievers enjoy winning challenges and completing tasks. In CodePlay, this type of player used all elements except for two (donated and received items). Their average completion rate of secondary missions was high, which agrees with the literature regarding the resolution of challenges.

Free spirits also made use of most elements, except visiting secret areas. It is noticeable that these users had high usage averages for most elements, except for donated items, collected coins, and sent/received messages. According to the literature, free spirits like to explore the environment (Marczewski, 2015). The use of various types of elements supports this fact, as accessing the elements requires exploration of the environment. Free spirits also scored high in Easter eggs, which are hidden game elements within the CodePlay environment, meaning that accessing them also requires exploration. The same does not apply to the secret area, as none of the free spirits accessed this element. However, free spirits had the highest average among all user classes in accessing the leaderboard, a competition element that, according to the literature, is more suited for the player type.

Philanthropists are users who enjoy helping others without expecting any reward in return (Marczewski, 2015). Therefore, this type of user highly recommends elements such as donations. Although philanthropists have used most of the elements, except for the secret area, and received donations, they could have had a higher average in donating items, which contradicts the literature's findings.

Players are motivated by extrinsic elements, meaning that they enjoy earning rewards such as coins and prizes, and they are also competitive (Marczewski, 2015). This user group was the only one to use all types of elements, with only a few low scores. Although they did not have a

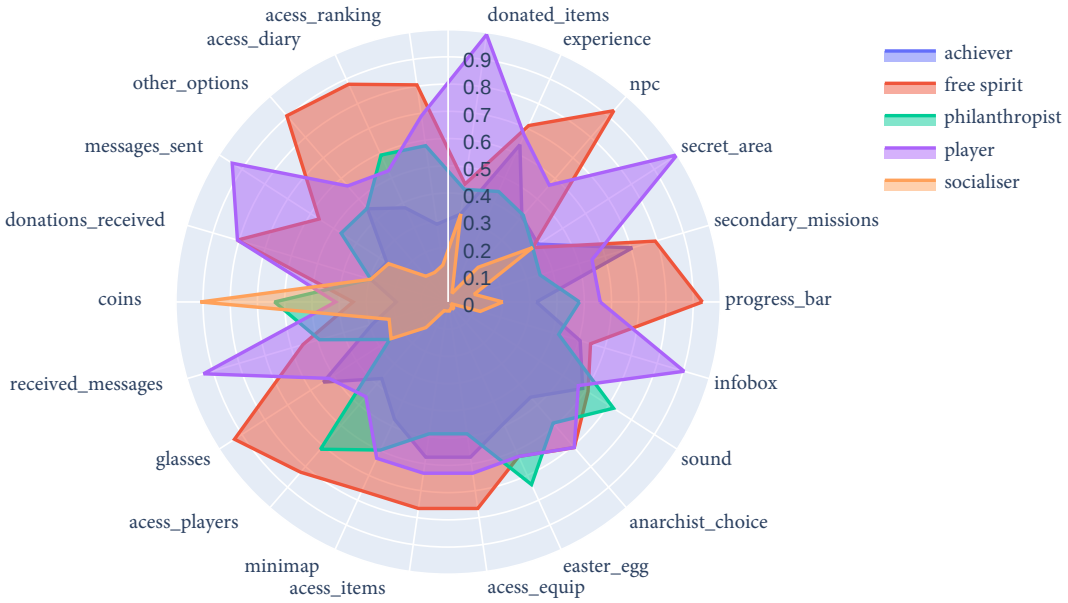

Figure 10: Usage of game elements according to player type.

high average in collecting coins, players earned much experience, which is also recommended for this type of user. Players donated/received many items and sent/received many messages, characteristics that, according to the literature, are more in line with philanthropists and socializers, respectively.

The socializers are primarily motivated by social interaction (Marczewski, 2015). Despite being the smallest group among the students, they were expected to use the messaging feature more to interact with other players. However, they obtained the lowest average among all groups. Socializers were also the users who utilized the fewest game elements, standing out only in collecting coins.

As is the case in other research (Mora et al., 2019; Rogers et al., 2021), there needs to be a consensus when evaluating which game elements are ideal for a given user profile. Depending on the gamification style, users may prefer different game elements (Ferreira et al., 2023). If in (Krath & von Korflesch, 2021) it was discovered that players like to collect things, in our research, we identified that the game currency element was used less when compared to the other elements used by the profile, for example.

Although there is no consensus, Hexad user types were used to classify users in this research because it was the user type classification structure that obtained the best results (Pessoa et al., 2023; Hallifax et al., 2019) and is specific to identify the profile of gamification users (Tondello et al., 2016). However, more in-depth studies are needed: (i) Considering the characteristics inherent to an RPG-style game, evaluate the extent to which the game may have influenced the type of user;

(ii) the human conditions of the answers influence the questionnaire results. Although we have eliminated the answers that were marked in a single answer, we cannot guarantee the quality of the answers; (iii) the questionnaire was answered at the beginning of the semester, and the elements were evaluated over two months. According to the literature, there may be changes in the user profile over time (A. Santos et al., 2021). In this way, our findings reinforce the need to evaluate user usage data to try to identify user profiles in order to offer game elements more aligned with their profiles.

Analyzing the usage of game elements by different user types allows for verifying whether the recommended elements for each profile are accessed by their respective types and identifying which elements are most frequently used by the students. Such analysis of user profiles enables improvements to be made to the Useful Application, such as changes in design, inclusion, and exclusion of elements, or an increase in the frequency of appearance of certain elements. Furthermore, it is possible to identify the combination of game elements that best influence academic performance and correlate the impact of these elements on student learning.

8 Conclusions, limitations and future work

In this article, we presented several initiatives of the Federal University of Amazonas research group that adopted a methodology based on practice and automatic code assessment in CS1 courses. The objective was to showcase the work of a national research group using educational data collection from an online judge to do some LA. Furthermore, the CodeBench dataset is being published for the first time in a scientific journal.

Concerning the topic of **student performance prediction**, the persistent problem of dropout and failure in programming courses, particularly in STEM-related fields, has been a longstanding concern that demands ongoing attention. In this study, we address this issue by applying Learning Analytics techniques, focusing on the prediction of student performance. Our findings underscore the fundamental importance of performance prediction as a valuable tool for identifying at-risk students and providing targeted interventions. This approach can significantly enhance students' chances of success, reduce attrition rates, and improve overall performance.

A significant contribution of this study is the emphasis on data granularity. We observed that simple metrics, such as the number of attempts and correctness of answers, are only part of the equation. Detailed analysis of attributes extracted from online judge logs revealed rich and relevant information for performance prediction. The results of our predictive models are promising, with an F1-Score of approximately 80% using data from the initial weeks of the course. As more data is collected over time, model accuracy improves, reaching approximately 90% by the sixth week. This demonstrates the models' capability for adaptation and continuous learning. Furthermore, our research underscores the importance of model explainability. Using the SHAP framework allowed for a deeper understanding of the impact of different variables on student performance, identifying effective and ineffective behaviors. This not only enhances prediction quality but also provides valuable insights for educators.

Personalization and targeted intervention are essential aspects of our results. Recognizing students' individuality, our models not only predict performance but also suggest personalized

interventions based on student behavior. This approach is critical for tailoring teaching to each student's specific needs.

While our research has achieved significant results, it also highlights the ongoing challenges in the field of education. Not all student behaviors are easily modifiable, and the approach requires a profound understanding of the factors affecting student performance.

Our exploration of **programming exercise difficulty** highlighted the critical distinction between complexity and difficulty. Complexity refers to the intrinsic characteristics of an exercise, such as its description's extent, the number of concepts involved, and its cyclomatic complexity. On the other hand, difficulty pertains to the effort students must invest to solve the exercise.

We discussed a range of metrics to objectively measure programming exercise difficulty, including the success rate, average implementation time, number of executions, number of submissions, and more. Each metric provides a unique perspective on exercise difficulty. Strategies for classifying exercises into different difficulty levels were explored, with module stratification proving effective by grouping similar questions and reducing noise. The application of machine learning techniques, such as classifiers and regressions, has been crucial in this process.

While our study has made significant progress in predicting exercise difficulty, we acknowledge the potential for further research. Exploring alternative metrics and considering a broader dataset can improve prediction accuracy and offer a more comprehensive understanding of exercise difficulty. The practical implications of our work in education cannot be overstated, as instructors can create well-balanced assignments and closely understand student performance. Despite this, the ability to classify exercises into different difficulty levels improves students' learning experience and supports teachers.

In conclusion, this study unveils the potent intersection of Learning Analytics (LA) and **gamification** as exemplified by the CodePlay platform, offering profound implications for educational technology. The amalgamation of LA techniques with gamification, particularly in CodePlay, addresses the age-old challenge of evaluating the effectiveness of educational games and gamified systems. It transcends conventional assessment methods, such as pre- and post-test questionnaires, towards a more robust and comprehensive evaluation paradigm.

CodePlay showcases a rich array of gamification elements thoughtfully integrated into educational platforms, including virtual currency, experience points, competitive rankings, and interactive features. These elements are strategically designed to elevate student engagement, satisfaction, and overall learning outcomes.

A pivotal aspect of our research lies in the meticulous data acquisition and logging system deployed in CodePlay, capturing a wide spectrum of student interactions. This dataset forms the bedrock for in-depth analysis, offering invaluable insights into user behavior within gamified environments. To ensure uniformity in the analysis of data from diverse game elements, we employ Z-score normalization, making the interpretation of user interactions systematic and insightful.

Leveraging the Hexad model for user categorization based on preferences and behaviors introduces a dimension of personalization into gamification. Different user profiles, such as achievers and socializers, interact with game elements in distinct ways, highlighting the potential for tailored educational experiences. Our study delves into a comprehensive analysis of user behavior within CodePlay, harnessing the Hexad model to unveil how various player profiles engage with

the platform's diverse game elements and functionalities.

While this research significantly advances our understanding of the impact of gamification on learning, it also underscores the imperative for ongoing exploration. The field of Learning Analytics for Gamification offers boundless prospects for further refinement of gamified educational paradigms and the discovery of new research avenues.

8.1 Limitations

This study presents several limitations that should be taken into account when interpreting the findings. One of the limitations pertains to the source of the collected data, which is confined to a single educational institution. This limitation may impact the generalizability of the results to a broader context. However, it is worth noting that the data were gathered from CS1 courses spanning multiple years and encompassed students from various academic disciplines. This diversity in the dataset may partially mitigate this limitation, although applying the findings to other institutions should be done cautiously.

Additionally, a limitation of this study is associated with the challenge of addressing the issue of imbalanced problem difficulty levels in the context of introductory computer programming courses. In such courses, a significant proportion of the problems are inherently easy, resulting in an imbalanced distribution of difficulty labels. This imbalance introduces challenges in training and evaluating models for problem difficulty classification.

Furthermore, the integration of gamification elements in an educational context can be highly contextual and may significantly vary between institutions. Therefore, generalizing the results related to gamification should be approached with caution, taking into account the specificities of each educational environment. These limitations underscore areas that warrant attention in subsequent studies and provide a foundation for the development of more robust and generalizable approaches in Learning Analytics, considering the diverse gamification practices in different contexts.

These limitations highlight areas that deserve attention in subsequent studies and provide a foundation for the development of more robust and generalizable approaches in Learning Analytics.

8.2 Future work

As future directions for research, we are working on the development of new metrics for descriptive and diagnostic analytics based on students' behaviors in online judges. The purpose of such metrics is to help teachers identify possible student difficulties, as well as identify topics in their subjects that need to be reinforced in the classroom. We will employ methods of Visual Learning Analytics and Educational Data Storytelling to assist teachers in interpreting the values and dynamics of these metrics.

Additionally, we are also studying techniques to cluster the codes developed to a given exercise according to the strategies adopted by students to solve the exercise. Our idea is to provide teachers, through these clusters, with a comprehensive overview of the codes developed by students for a given exercise, without the need to examine each code individually.

Finally, we are also working on a new method to estimate the difficulty level of programming exercises that have not yet been solved by any student in the online judge system. In the same way as in (Lima et al., 2020), in this method we also intend to use the instructor's model solution code to conduct such task. However, instead of using code metrics as proposed in (Lima et al., 2020), we will search for exercises whose students-developed codes are similar to the model solution code registered by the instructor. Once such exercises are found, the difficulty level of the new exercise will be estimated based on the difficulty level of the exercises found. The intuition behind this technique is that exercises with similar solutions have similar difficulty levels.

Our research contributes to the fields of machine learning and education by demonstrating the effectiveness of data-driven interventions in addressing persistent issues in STEM courses. We hope this study provides a solid foundation for future research and innovations in the field of education.

Acknowledgments

The present work is the result of the Research and Development (R&D) project 001/2020, signed with Federal University of Amazonas and FAEPI, Brazil, which has funding from Samsung, using resources from the Informatics Law for the Western Amazon (Federal Law No. 8.387/1991), and its disclosure is in accordance with article 39 of Decree No. 10.521/2020. This study was financed in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil - CNPq (Process 308513/2020-7) and Fundação de Amparo à Pesquisa do Estado do Amazonas - FAPEAM (Process 01.02.016301.02770/2021-63). Flávio J. M. Coelho thanks the Amazonas State University for the institutional support and incentive for the development of this research.

Special Edition: Practical Applications of Learning Analytics in Educational Institutions in Brazil

This publication constitutes the special edition entitled "Practical Applications of Learning Analytics in Educational Institutions in Brazil", led by guest editors Cristian Cechinel (Federal University of Santa Catarina), Diego Dermeval (Federal University of Alagoas), Elaine H. T. Oliveira (Federal University of Amazonas), Isabela Gasparini (State University of Santa Catarina), and Rafael Ferreira Mello (Federal Rural University of Pernambuco and CESAR School).

References

Alonso-Fernandez, C., Calvo Morata, A., Freire, M., Martínez-Ortiz, I., & Manjón, B. (2017). Systematizing game learning analytics for serious games. In *Proceedings of 2017 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1106–1113). IEEE. doi: 10.1109/EDUCON.2017.7942988 [GS Search]

Barata, G., Gama, S., Jorge, J., & Gonçalves, D. (2014, 10). Relating gaming habits with

student performance in a gamified learning experience. *CHI PLAY 2014 – Proceedings of the 2014 Annual Symposium on Computer-Human Interaction in Play*, 17–25. doi: 10.1145/2658537.2658692 [GS Search]

Bartle, R. (1996). Hearts, clubs, diamonds, spades: Players who suit muds. *Journal of MUD research*, *1*(1), 19. Retrieved from http://www.arise.mae.usp.br/wp-content/uploads/2018/03/Bartle-player-types.pdf [GS Search]

Beckmann, J. F., Birney, D. P., & Goode, N. (2017). Beyond psychometrics: the difference between difficult problem solving and complex problem solving. *Frontiers in psychology*, *8*, 1739. doi: 10.3389/fpsyg.2017.01739 [GS Search]

Bennedsen, J., & Caspersen, M. E. (2019, apr). Failure rates in introductory programming: 12 years later. *ACM Inroads*, *10*(2), 30–36. doi: https://doi.org/10.1145/3324888 [GS Search]

Blikstein, P. (2011). Using learning analytics to assess students' behavior in open-ended programming tasks. In *Proceedings of the 1st International Conference on Learning Analytics and Knowledge* (p. 110-–116). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/2090116.2090132 [GS Search]

Calderón, A., & Ruiz, M. (2015). A systematic literature review on serious games evaluation: An application to software project management. *Computers Education*, *87*, 396–422. doi: 10.1016/j.compedu.2015.07.011 [GS Search]

Carter, A., Hundhausen, C., & Olivares, D. (2019). Leveraging the integrated development environment for learning analytics. In *The Cambridge Handbook of Computing Education Research* (pp. 679–706). Cambridge: Cambridge University Press. doi: 10.1017/9781108654555.024 [GS Search]

Carvalho, L. S. G., Oliveira, D. F., & Gadelha, B. (2016, 11). Juiz online como ferramenta de apoio a uma metodologia de ensino híbrido em programação. In *Proceedings of the XXVII Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE 2016)* (pp. 140–149). SBC. doi: 10.5753/cbie.sbie.2016.140 [GS Search]

Castro-Wunsch, K., Ahadi, A., & Petersen, A. (2017). Evaluating neural networks as a method for identifying students in need of assistance. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (p. 111-–116). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3017680.3017792 [GS Search]

Costa, E. B., Fonseca, B., Santana, M. A., de Araújo, F. F., & Rego, J. (2017). Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses. *Computers in Human Behavior*, *73*, 247–256. doi: 10.1016/j.chb.2017.01.047 [GS Search]

Echeverría, L., Cobos, R., Machuca, L., & Claros, I. (2017). Using collaborative learning scenarios to teach programming to non-cs majors. *Computer Applications in Engineering Education*, *25*(5), 719-731. doi: 10.1002/cae.21832 [GS Search]

Edwards, J., Leinonen, J., & Hellas, A. (2020). A study of keystroke data in two contexts: Written language and programming language influence predictability of learning outcomes. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 413–419). doi: 10.1145/3328778.3366863 [GS Search]

Effenberger, T., Cechák, J., & Pelánek, R. (2019). Difficulty and complexity of introductory programming problems. In *Educational Data Mining in Computer Science Education (CSEDM)*. Retrieved from https://t.ly/Aspf1 [GS Search]

Effenberger, T., Čechák, J., & Pelánek, R. (2019). Measuring difficulty of introductory program-

ming tasks. In *Proceedings of the Sixth (2019) ACM Conference on Learning @ Scale*. New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3330430.3333641 [GS Search]

Elnaffar, S. (2016). Using software metrics to predict the difficulty of code writing questions. In *2016 IEEE Global Engineering Education Conference (EDUCON)* (pp. 513–518). doi: 10.1109/EDUCON.2016.7474601 [GS Search]

Fernandes, J., Carvalho, L. S. G., Oliveira, D. F., Oliveira, E. H. T., Pereira, F. D., & Lauschner, T. (2023). Correlação entre complexidade e dificuldade de questões de programação em juízes online. In *Anais do III Simpósio Brasileiro de Educação em Computação* (pp. 97–107). doi: 10.5753/educomp.2023.228217 [GS Search]

Ferreira, R. M., et al. (2023). *Um estudo sobre a tipologia de usuários Hexad e sua relação com os elementos de jogos de uma plataforma de gamificação baseada em jogos RPG*. Master thesis, Universidade Federal do Amazonas, Manaus, AM. Retrieved from https://tede.ufam.edu.br/handle/tede/9647 [GS Search]

Fonseca, S., Oliveira, E., Pereira, F. D., Oliveira, D. F., & Carvalho, L. S. G. (2019). Adaptação de um método preditivo para inferir o desempenho de alunos de programação. In *Proceedings of the XXX Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE 2019)* (Vol. 30, pp. 1651–1660). doi: 10.5753/cbie.sbie.2019.1651 [GS Search]

Freire, M., Serrano-Laguna, A., Manero, B., Martínez-Ortiz, I., Moreno-Ger, P., & Fernández-Manjón, B. (2016). Game learning analytics: Learning analytics for serious games. In *Learning, Design, and Technology* (pp. 1–29). Springer Nature Switzerland AG. doi: 10.1007/978-3-319-17727-4$_2$1 – 1 [GS Search]

Hallifax, S., Serna, A., Marty, J.-C., Lavoué, G., & Lavoué, E. (2019). Factors to consider for tailored gamification. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play* (pp. 559–572). doi: 10.1145/3311350.3347167 [GS Search]

Ihantola, P., Hellas, A., Butler, M., Börstler, J., Edwards, S., Isohanni, E., ... Toll, D. (2015). Educational data mining and learning analytics in programming: Literature review and case studies. In *Proceedings of the 2015 ITiCSE on Working Group Reports* (p. 41—63). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/2858796.2858798 [GS Search]

INEP (2017). *Relatório Síntese de Área – Ciência da Computação* (Tech. Rep.). Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira. Retrieved from https://download.inep.gov.br/educacao_superior/enade/relatorio_sintese/2017/Ciencia_da_Computacao.pdf

Jadud, M. C. (2006). Methods and tools for exploring novice compilation behaviour. In *Proceedings of the Second International Workshop on Computing Education Research* (p. 73–84). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/1151588.1151600 [GS Search]

Klock, A. C. T., Gasparini, I., Pimenta, M. S., & Hamari, J. (2020). Tailored gamification: A review of literature. *International Journal of Human-Computer Studies*, *144*, 102495. doi: 10.1016/j.ijhcs.2020.102495 [GS Search]

Klock, A. C. T., Ogawa, A. N., Gasparini, I., & Pimenta, M. S. (2018). Integration of learning analytics techniques and gamification: An experimental study. In *2018 IEEE 18th International Conference on Advanced Learning Technologies (ICALT)* (pp. 133–137). doi: 10.1109/ICALT.2018.00039 [GS Search]

Krath, J., & von Korflesch, H. F. (2021). Player types and game element preferences: Investigating the relationship with the gamification user types hexad scale. In *International Conference on Human-Computer Interaction* (pp. 219–238). doi: 10.1007/978-3-030-77277-2$_1$8 [GS Search]

Lacave, C., Molina Díaz, A., & Cruz-Lemus, J. (2018, 06). Learning analytics to identify dropout factors of computer science studies through bayesian networks. *Behaviour & Information Technology*, *37*, 1–15. doi: 10.1080/0144929X.2018.1485053 [GS Search]

Lavareda, R. M., Colonna, J. G., & Oliveira, D. F. (2020). Autenticação contínua de alunos utilizando biometria comportamental em ambiente juiz on-line. In *Proceedings of the XXXI Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE 2020)* (Vol. 31, pp. 1193–1202). Porto Alegre, RS, Brasil: SBC. doi: 10.5753/cbie.sbie.2020.1193 [GS Search]

Leinonen, J., Longi, K., Klami, A., & Vihavainen, A. (2016). Automatic inference of programming performance and experience from typing patterns. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (p. 132–137). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/2839509.2844612 [GS Search]

Lima, M. A. P., Carvalho, L. S. G., Oliveira, E. H. T., Oliveira, D. F., & Pereira, F. D. (2020). Classificação de dificuldade de questões de programação com base em métricas de código. In *Proceedings of the XXXI Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE 2020)* (Vol. 31, pp. 1323–1332). doi: 10.5753/cbie.sbie.2020.1323 [GS Search]

Lima, M. A. P., Carvalho, L. S. G., Oliveira, E. H. T., Oliveira, D. F., & Pereira, F. D. (2021a). Uso de atributos de código para classificar a dificuldade de questões de programação em juízes online. *Revista Brasileira de Informática na Educação*, *29*, 1137–1157. doi: 10.5753/RBIE.2021.29.0.1137 [GS Search]

Lima, M. A. P., Carvalho, L. S. G., Oliveira, E. H. T., Oliveira, D. F., & Pereira, F. D. (2021b). Uso de atributos de código para classificação da facilidade de questões de codificação. In *Anais do Simpósio Brasileiro de Educação em Computação* (pp. 113–122). online: SBC. doi: 10.5753/educomp.2021.14477 [GS Search]

Liu, P., & Li, Z. (2012). Task complexity: A review and conceptualization framework. *International Journal of Industrial Ergonomics*, *42*(6), 553–568. doi: 10.1016/j.ergon.2012.09.001 [GS Search]

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (p. 4768–4777). Red Hook, NY, USA: Curran Associates Inc. Retrieved from https://dl.acm.org/doi/10.5555/3295222.3295230 [GS Search]

Maher, Y., Moussa, S. M., & Khalifa, M. E. (2020). Learners on focus: Visualizing analytics through an integrated model for learning analytics in adaptive gamified e-learning. *IEEE Access*, *8*, 197597–197616. doi: 10.1109/ACCESS.2020.3034284 [GS Search]

Marczewski, A. (2015). *Even ninja monkeys like to play: Gamification, game thinking and motivational design* (First ed.). Gamified UK. Retrieved from https://books.google.com.br/books?id=RoHpjgEACAAJ [GS Search]

Mora, A., Tondello, G. F., Calvet, L., González, C., Arnedo-Moreno, J., & Nacke, L. E. (2019). The quest for a better tailoring of gameful design: An analysis of player type preferences. In *Proceedings of the XX International Conference on Human Computer Interaction* (pp.

1–8). doi: https://doi.org/10.1145/3335595.3335625 [GS Search]

Nacke, L. E., Bateman, C., & Mandryk, R. L. (2014). Brainhex: A neurobiological gamer typology survey. *Entertainment Computing*, *5*(1), 55-62. doi: 10.1016/j.entcom.2013.06.002 [GS Search]

Norman, V. T., & Adams, J. C. (2015). Improving non-CS major performance in CS1. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 558–562). doi: doi.org/10.1145/2676723.2677214 [GS Search]

Oliveira, D. F., Lavareda Filho, R., Oliveira, E. H. T., Carvalho, L. S. G. C., Pereira, F. D., Colonna, J. G., & Menezes, A. (2021). Um Método de Detecção de Plágio para Sistemas Juiz On-line baseado no Comportamento dos Alunos. In *Proceedings of the XXXII Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE 2021)* (Vol. 32, pp. 836–848). Porto Alegre, RS, Brasil: SBC. doi: 10.5753/sbie.2021.21839 [GS Search]

Pelánek, R., Effenberger, T., & Čechák, J. (2022, March). Complexity and difficulty of items in learning systems. *International Journal of Artificial Intelligence in Education*, *32*(1), 196–232. doi: 10.1007/s40593-021-00252-4 [GS Search]

Pereira, F. D. (2022). *Towards AI-human hybrid online judges to support decision making for CS1 instructors and students*. PhD thesis, Universidade Federal do Amazonas, Manaus, AM. Retrieved from https://tede.ufam.edu.br/bitstream/tede/8981/8/Tese_FilipePereira_PPGI.pdf [GS Search]

Pereira, F. D., Fonseca, S., Oliveira, E. H. T., Cristea, A., Bellhäuser, H., Rodrigues, L., ... Carvalho, L. S. (2021). Explaining individual and collective programming students' behavior by interpreting a black-box predictive model. *IEEE Access*, *9*, 117097–117119. doi: 10.1109/ACCESS.2021.3105956 [GS Search]

Pereira, F. D., Fonseca, S., Oliveira, E. H. T., Oliveira, D. F., Cristea, A. I., & Carvalho, L. S. (2020). Deep learning for early performance prediction of introductory programming students: a comparative and explanatory study. *Brazilian journal of computers in education.*, *28*, 723–749. doi: 10.5753/rbie.2020.28.0.723 [GS Search]

Pereira, F. D., Oliveira, E., Oliveira, D. F., Carvalho, L. S. G., & Junior, H. (2019). Otimização e automação da predição precoce do desempenho de alunos que utilizam juízes online: uma abordagem com algoritmo genético. In *Proceedings of the XXX Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE 2019)* (Vol. 30, pp. 1451–1460). doi: 10.5753/cbie.sbie.2019.1451 [GS Search]

Pereira, F. D., Oliveira, E. H. T., & Oliveira, D. F. (2017). Predição de zona de aprendizagem de alunos de introdução à programação em ambientes de correção automática de código. In *Proceedings of the XXVIII Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE 2017)* (Vol. 28, pp. 1507–1516). SBC. doi: 10.5753/cbie.sbie.2017.1507 [GS Search]

Pereira, F. D., Oliveira, E. H. T., Oliveira, D. F., Cristea, A. I., Carvalho, L. S. G., Fonseca, S., ... Isotani, S. (2020). Using learning analytics in the Amazonas: understanding students' behaviour in introductory programming. *British journal of educational technology*, *51*(4), 955–972. doi: 10.1111/bjet.12953 [GS Search]

Pereira, F. D., Rodrigues, L., Henklain, M., Freitas, H., Oliveira, D. F., Cristea, A. I., ... Oliveira, E. H. T. (2022). Towards human-AI collaboration: a recommender system to support CS1 instructors to select problems for assignments and exams. *IEEE Transactions on Learning*

Technologies. doi: 10.1109/TLT.2022.3224121 [GS Search]

Pereira, F. D., Souza, L. M., Oliveira, E. H. T., Oliveira, D. F., & Carvalho, L. S. G. (2020). Predição de desempenho em ambientes computacionais para turmas de programação: um mapeamento sistemático da literatura. In *Proceedings of the XXXI Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE 2020)* (pp. 1673–1682). doi: 10.5753/cbie.sbie.2020.1673 [GS Search]

Pessoa, M. (2022). *Codeplay: uma plataforma que incorpora a ludicidade de jogos de entretenimento a um juiz on-line*. PhD thesis, Universidade Federal do Amazonas, Manaus, AM. Retrieved from https://tede.ufam.edu.br/handle/tede/9150 [GS Search]

Pessoa, M., Lima, M., Pires, F., Haydar, G., Melo, R., Rodrigues, L., ... others (2023). A journey to identify users' classification strategies to customize game-based and gamified learning environments. *IEEE Transactions on Learning Technologies*. doi: 10.1109/TLT.2023.3317396 [GS Search]

Pessoa, M., Melo, R., Haydar, G., Oliveira, D. F., Carvalho, L. S. G., Oliveira, E. H. T., ... Isotani, S. (2021). Uma análise dos tipos de jogadores em uma plataforma de gamificação incorporada a um sistema juiz on-line. In *Proceedings of the XXXII Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE 2022* (pp. 474–486). doi: 10.5753/sbie.2021.218600 [GS Search]

Pessoa, M., Oliveira, D. F., Carvalho, L. S. G., Oliveira, E., Nakamura, W., & Conte, T. (2019). Codeplay: Uma plataforma de gamificação baseada em jogos de RPG multiplayer. In *Proceedings of the XXX Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE 2019)* (Vol. 30, pp. 843–852). doi: 10.5753/cbie.sbie.2019.843 [GS Search]

Quille, K., & Bergin, S. (2019). CS1: how will they do? How can we help? A decade of research and practice. *Computer Science Education*, *29*(2–3), 254–282. doi: 10.1080/08993408.2019.1612679 [GS Search]

Ribeiro, R. B. S., Carvalho, L. S. G., Oliveira, E. H. T., Oliveira, D. B. F., & Pessoa, M. S. P. (2020). Investigação empírica sobre os efeitos da gamificação de um juiz online em uma disciplina de introdução à programação. *Revista Brasileira de Informática na Educação*, *28*, 461–490. doi: 10.5753/RBIE.2020.28.0.461 [GS Search]

Robins, A. V. (2019). Novice programmers and introductory programming. In S. A. Fincher & A. V. Robins (Eds.), *The Cambridge Handbook of Computing Education Research* (pp. 327–376). Cambridge University Press. doi: 10.1017/9781108654555.013 [GS Search]

Rogers, M., Yao, W., Luxton-Reilly, A., Leinonen, J., Lottridge, D., & Denny, P. (2021, March 3). Exploring personalization of gamification in an introductory programming course. In *SIGCSE '21: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (pp. 1121–1127). United States: ACM. doi: 10.1145/3408877.3432402 [GS Search]

Santana, B. L., & Bittencourt, R. A. (2018). Increasing motivation of CS1 non-majors through an approach contextualized by games and media. In *2018 IEEE Frontiers in Education Conference (FIE)* (pp. 1–9). San Jose, CA, USA. doi: 10.1109/FIE.2018.8659011 [GS Search]

Santos, A., Oliveira, W., Hamari, J., & Isotani, S. (2021). Do people's user types change over time? An exploratory study. In M. Bujić, J. Koivisto, & J. Hamari (Eds.), *Proceedings of the 5th International GamiFIN Conference Levi, Finland, April 7-9, 2021* (pp. 90–99).

CEUR-WS. doi: 10.48550/arXiv.2106.10148 [GS Search]

Santos, P., Carvalho, L. S. G., Oliveira, E., & Oliveira, D. F. (2019). Classificação de dificuldade de questões de programação com base na inteligibilidade do enunciado. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE 2019)*, *30*(1), 1886–1895. doi: 10.5753/cbie.sbie.2019.1886 [GS Search]

Sheard, J., Simon, Carbone, A., Chinn, D., Clear, T., Corney, M., ... Teague, D. (2013). How difficult are exams? a framework for assessing the complexity of introductory programming exams. In *Proceedings of the 15th Australasian Computing Education Conference* (Vol. 136, p. 145—154). AUS. Retrieved from https://dl.acm.org/doi/10.5555/2667199.2667215 [GS Search]

Silva, D., Melo, R., Pires, F., & Pessoa, M. (2021, dez.). Avaliacão de objetos digitais de aprendizagem: como os licenciados em computação analisam jogos educacionais? *Revista Novas Tecnologias na Educação*, *19*(2), 111–121. doi: 10.22456/1679-1916.121193 [GS Search]

Silva, E. S., Carvalho, L. S. G., Oliveira, D. F., Oliveira, E. H. T., Lauschner, T., Lima, M. A. P., & Pereira, F. D. (2022). Previsão de indicadores de dificuldade de questões de programação a partir de métricas do código de solução. In *Proceedings of the XXXIII Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação – SBIE 2022)* (Vol. 33, pp. 859–870). Manaus, AM: SBC. doi: 10.5753/sbie.2022.224724 [GS Search]

Tomasevic, N., Gvozdenovic, N., & Vranes, S. (2020). An overview and comparison of supervised data mining techniques for student exam performance prediction. *Computers Education*, *143*, 103676. doi: 10.1016/j.compedu.2019.103676 [GS Search]

Tondello, G. F., Arrambide, K., Ribeiro, G., Cen, A. J.-l., & Nacke, L. E. (2019). "I don't fit into a single type": A trait model and scale of game playing preferences. In D. Lamas, F. Loizides, L. Nacke, H. Petrie, M. Winckler, & P. Zaphiris (Eds.), *Human-Computer Interaction – INTERACT 2019* (pp. 375–395). Cham: Springer International Publishing. doi: 10.1007/978-3-030-29384-$0_2$3 [GS Search]

Tondello, G. F., Wehbe, R. R., Diamond, L., Busch, M., Marczewski, A., & Nacke, L. E. (2016). The gamification user types Hexad scale. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play* (p. 229–243). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/2967934.2968082 [GS Search]

Watson, C., Li, F. W., & Godwin, J. L. (2013). Predicting performance in an introductory programming course by logging and analyzing student programming behavior. In *2013 IEEE 13th International Conference on Advanced Learning Technologies* (pp. 319–323). doi: 10.1109/ICALT.2013.99 [GS Search]

Whalley, J., & Kasto, N. (2014). How difficult are novice code writing tasks? A software metrics approach. In *Proceedings of the Sixteenth Australasian Computing Education Conference - Volume 148* (p. 105–112). AUS: Australian Computer Society, Inc. Retrieved from https://dl.acm.org/doi/abs/10.5555/2667490.2667503 [GS Search]

Wolpert, D. H. (1992, feb). Original contribution: Stacked generalization. *Neural Netw.*, *5*(2), 241–259. doi: 10.1016/S0893-6080(05)80023-1 [GS Search]

Appendix

File sample `/user/user.data` of the student `5052` in Figure 4.

```
-- CURRENT DEGREE COURSE:
---- course id: 8
---- course name: Materials Engineering
---- institution id: 1
---- institution name: Federal University of Amazonas
-- HIGH SCHOOL:
---- high school name:
---- school type: public school
---- shift: afternoon shift
---- graduation year: 2019
-- PERSONAL COMPUTER:
---- has a PC at home: yes
---- share this PC with other people at home: yes
---- this PC has access to Internet: yes (high-speed Internet)
---- previous experience of any computer language: no
-- WORK:
---- worked or interned before the degree: no
---- company name: n/a
---- year started working: n/a
---- year stopped working: n/a
-- PREVIOUS DEGREE:
---- started other degree programmes: no
---- degree course: n/a
---- institution name: n/a
---- year started this degree: n/a
---- year stopped this degree: n/a
-- OTHER INFORMATION:
---- sex: male
---- year of birth: 2002
---- civil status: single
---- have kids: no
```

File sample `/user/logins.log` of the student `5052` in Figure 4.

```
2020-09-08 10:49:28#user/login.
2020-09-08 10:55:20#user/login.
2020-09-09 13:38:59#user/login.
2020-09-09 14:31:33#user/login.
2020-09-17 10:05:41#user/login.
2021-03-30 19:06:40#user/login.
2021-04-02 12:38:36#user/login.
2021-04-02 17:33:59#user/login.
2021-04-03 00:44:39#user/login.
2021-04-05 12:09:02#user/login.
2021-04-10 10:54:20#user/login.
2021-04-12 12:19:25#user/login.
2021-04-19 10:42:46#user/login.
2022-10-26 14:07:32#user/login.
2022-10-26 14:57:18#user/logout.
2022-10-31 14:24:14#user/login.
2022-10-31 15:39:58#user/logout.
2022-11-01 13:29:44#user/login.
2022-11-07 14:05:56#user/login.
2022-11-07 14:07:28#user/login.
```

File sample `/user/events.log` of the student `5052` in Figure 4.


```
{"date":"26/10/2022@14:10:34:79","page":"material/index","turma":"438",
"evento":"acesso"},
{"date":"26/10/2022@14:10:37:667","page":"turma/trabalhos","turma":"438",
"evento":"acesso"},
{"date":"26/10/2022@14:10:43:272","page":"turma/view","turma":"438",
"evento":"acesso"},
{"date":"26/10/2022@14:17:9:528","page":"material/index","turma":"438",
"evento":"acesso"},
{"date":"26/10/2022@14:17:10:874","page":"material/download","turma":"438",
"itemId":"2542","evento":"acesso"},
{"date":"26/10/2022@14:17:35:911","page":"material/download","turma":"438",
"itemId":"2526","evento":"acesso"},
{"date":"26/10/2022@14:51:15:497","page":"turma/view","turma":"438",
"evento":"acesso"},
{"date":"31/10/2022@14:26:17:588","page":"turma/view","turma":"438",
"evento":"acesso"},
{"date":"31/10/2022@14:26:21:26","page":"material/index","turma":"438",
"evento":"acesso"},
{"date":"31/10/2022@14:26:25:925","page":"material/download","turma":"438",
"itemId":"2525","evento":"acesso"},
```


File sample `/codes/3927_1326.py` of the student `5052` in Figure 4.

```
a=40
b=35
print(a+b)
```

File sample `/codemirror/3927_1326.log` of the student `5052` in Figure 4.


```
2022-10-31 14:20:15.907#tab-click:1326#
2022-10-31 14:20:15.941#viewportChange#0
2022-10-31 14:20:19.972#mousedown#{"isTrusted":true}
2022-10-31 14:20:19.976#focus#
2022-10-31 14:20:21.265#mousedown#{"isTrusted":true}
2022-10-31 14:20:37.256#keydown#{"key":"4"}
2022-10-31 14:20:37.257#keypress#{"key":"4"}
2022-10-31 14:20:37.257#change#{"from":{"line":0,"ch":2,"sticky":null},"to":
{"line":0,"ch":2,"sticky":null},"text":["4"],"removed":[""],
"origin":"+input"}
2022-10-31 14:20:37.429#keydown#{"key":"0"}
2022-10-31 14:20:37.429#keypress#{"key":"0"}
2022-10-31 14:20:37.430#change#{"from":{"line":0,"ch":3,"sticky":null},"to":
{"line":0,"ch":3,"sticky":null},"text":["0"],"removed":[""],
"origin":"+input"}
2022-10-31 14:20:37.444#keyup#{"key":"4"}
2022-10-31 14:20:37.508#keyup#{"key":"0"}
2022-10-31 14:20:37.861#keydown#{"key":"Shift"}
2022-10-31 14:20:38.150#keydown#{"key":"Enter"}
2022-10-31 14:20:38.151#keypress#{"key":"Enter"}
2022-10-31 14:20:38.152#change#{"from":{"line":0,"ch":4,"sticky":null},"to":
```

```
{"line":0,"ch":4,"sticky":null},"text":["",""],"removed":[""],
"origin":"+input"}
2022-10-31 14:20:38.158#viewportChange#0
2022-10-31 14:20:38.292#keyup#{"key":"Enter"}
2022-10-31 14:20:38.358#keyup#{"key":"Shift"}
```

File sample `/executions/3927_1326.log` of the student `5052` in Figure 4.

```
== TEST (2022-10-31 14:20:55)
-- CODE:
a=40
b=35
print(a+b)
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
== TEST (2022-10-31 14:21:00)
-- CODE:
a=40
b=35
print(a+b)
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
== SUBMITION (2022-10-31 14:21:06)
-- CODE:
a=40
b=35
print(a+b)
-- EXECUTION TIME:
0.240713
-- TEST CASE 1:
---- input:
xx
---- correct output:
75
---- user output:
75
-- GRADE:
100%
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
```

File sample `/grades/3927.log` of the student `5052` in Figure 4.

```
---- grade (0-10): 10
---- number of exercises: 16
---- correct: 16
---- incorrect: 0
---- blank: 0
```

File sample `/grades/final_grade.data` of the student `5052` in Figure 4.

```
6.48
```

File sample `/mousemove/3927.log` of the student `5052` in Figure 4.

{"date":"31/10/2022@14:18:40:465","e":0,"x":60,"y":225},
{"date":"31/10/2022@14:18:40:482","e":0,"x":60,"y":230},
{"date":"31/10/2022@14:18:40:715","e":"mousedown"},
{"date":"31/10/2022@14:18:40:718","e":0,"x":61,"y":231},
{"date":"31/10/2022@14:18:40:733","e":"mouseup"},
{"date":"31/10/2022@14:18:42:149","e":996,"x":67,"y":230},
{"date":"31/10/2022@14:18:42:182","e":996,"x":75,"y":229},
{"date":"31/10/2022@14:18:42:198","e":996,"x":78,"y":229},
{"date":"31/10/2022@14:18:42:236","e":996,"x":79,"y":229},
{"date":"31/10/2022@14:18:42:617","e":996,"x":70,"y":224},
{"date":"31/10/2022@14:18:42:684","e":996,"x":62,"y":220},
{"date":"31/10/2022@14:18:42:701","e":996,"x":61,"y":220},
{"date":"31/10/2022@14:18:42:717","e":996,"x":60,"y":219},
{"date":"31/10/2022@14:18:42:734","e":996,"x":59,"y":219},
{"date":"31/10/2022@14:18:42:766","e":996,"x":58,"y":219},
{"date":"31/10/2022@14:18:42:815","e":996,"x":57,"y":218},
{"date":"31/10/2022@14:18:42:816","e":"mousedown"},
{"date":"31/10/2022@14:18:42:951","e":"mouseup"},
{"date":"31/10/2022@14:18:43:777","e":994,"x":90,"y":221},
{"date":"31/10/2022@14:18:43:793","e":994,"x":184,"y":235},