JOURNAL OF LEARNING ANALYTICS

SOLAR SOCIETY for LEARNING ANALYTICS RESEARCH

Volume 11(1), 132–150. https://doi.org/10.18608/jla.2024.7979

# Reproducing Predictive Learning Analytics in CS1: Toward Generalizable and Explainable Models for Enhancing Student Retention

Denis Zhidkikh[1], Ville Heilala[2], Charlotte Van Petegem[3], Peter Dawyndt[4], Miitta Järvinen[5], Sami Viitanen[6], Bram De Wever[7], Bart Mesuere[8], Vesa Lappalainen[9], Lauri Kettunen[10], Raija Hämäläinen[11]

**Abstract**

Predictive learning analytics has been widely explored in educational research to improve student retention and academic success in an introductory programming course in computer science (CS1). General-purpose and interpretable dropout predictions still pose a challenge. Our study aims to reproduce and extend the data analysis of a privacy-first student pass–fail prediction approach proposed by Van Petegem and colleagues (2022) in a different CS1 course. Using student submission and self-report data, we investigated the reproducibility of the original approach, the effect of adding self-reports to the model, and the interpretability of the model features. The results showed that the original approach for student dropout prediction could be successfully reproduced in a different course context and that adding self-report data to the prediction model improved accuracy for the first four weeks. We also identified relevant features associated with dropout in the CS1 course, such as timely submission of tasks and iterative problem solving. When analyzing student behaviour, submission data and self-report data were found to complement each other. The results highlight the importance of transparency and generalizability in learning analytics and the need for future research to identify other factors beyond self-reported aptitude measures and student behaviour that can enhance dropout prediction.

**Notes for Practice**

- Dropout prediction in computer science education aids in detecting struggling students for early support.
- This study verified a privacy-friendly dropout prediction approach's robustness by replicating it in another educational context.
- Combining student behaviour and self-reported data early in a course enhances dropout prediction accuracy, enabling timely support.
- Researchers should find a balance between model accuracy and transparency to promote fair predictive learning analytics.
- Educators should not exclusively depend on classification-based prediction; they should explore alternative structures and pedagogies to facilitate learning.

**Keywords**

Predictive learning analytics, CS1, retention, privacy, self-report data, trace data

**Submitted:** 08/03/2023 — **Accepted:** 15/01/2024 — **Published:** 26/01/2024

*Corresponding author*[1] *Email: denis.d.zhidkikh@jyu.fi Address: The Faculty of Information Technology, PL 35, 40014 Jyväskylän yliopisto, Finland. ORCID iD: https://orcid.org/0000-0003-1335-8346*

[2] *Email: ville.s.heilala@jyu.fi Address: The Faculty of Education and Psychology, PL 35, 40014 Jyväskylän yliopisto, Finland. ORCID iD: https://orcid.org/0000-0003-2068-2777*

[3] *Email: charlotte.vanpetegem@ugent.be Address: Department of Applied Mathematics, Computer Science and Statistics, Krijgslaan 281 - S9, 9000 Ghent, Belgium. ORCID iD: https://orcid.org/0000-0003-0779-4897*

[4] *Email: peter.dawyndt@ugent.be Address: Department of Applied Mathematics, Computer Science and Statistics, Krijgslaan 281 - S9, 9000 Ghent, Belgium. ORCID iD: https://orcid.org/0000-0002-1623-9070*

[5] *Email: miitta.m.jarvinen@jyu.fi Address: The Faculty of Education and Psychology, PL 35, 40014 Jyväskylän yliopisto, Finland. ORCID iD: https://orcid.org/0000-0001-9627-2999*

[6] *Email: sami.a.viitanen@jyu.fi Address: The Faculty of Information Technology, PL 35, 40014 Jyväskylän yliopisto, Finland. ORCID iD: https://orcid.org/0009-0001-0401-4245*

 

[7] Email: bram.dewever@ugent.be Address: Department of Educational Studies, H. Dunantlaan 2, 9000 Ghent, Belgium. ORCID iD: https:// orcid.org/ 0000-0003-4352-4915
[8] Email: bart.mesuere@ugent.be Address: Department of Applied Mathematics, Computer Science and Statistics, Krijgslaan 281 - S9, 9000 Ghent, Belgium. ORCID iD: https:// orcid.org/ 0000-0003-0610-3441
[9] Email: vesa.t.lappalainen@jyu.fi Address: The Faculty of Information Technology, PL 35, 40014 Jyväskylän yliopisto, Finland. ORCID iD: https:// orcid.org/ 0000-0002-3298-0694
[10] Email: lauri.y.o.kettunen@jyu.fi Address: The Faculty of Information Technology, PL 35, 40014 Jyväskylän yliopisto, Finland. ORCID iD: https:// orcid.org/ 0000-0003-0432-2675
[11] Email: raija.h.hamalainen@jyu.fi Address: The Faculty of Education and Psychology, PL 35, 40014 Jyväskylän yliopisto, Finland. ORCID iD: https:// orcid.org/ 0000-0002-3248-9619

## 1. Introduction

One can say that the world runs on code. It is unsurprising, but also remarkable, how a few lines of code can change the world for better or for worse (Bosch, 2022). This underlines the importance of skilled and competent IT professionals. However, computer science (CS) learning and teaching in higher education (HE) worldwide are facing the challenges of interrupted studies and delayed graduation times (e.g., Kori et al., 2017). Furthermore, the World Economic Forum has identified the global shortage of skills and talent as the main obstacle to exploiting the opportunities of advanced technologies ("Markets of Tomorrow Report", 2023). Therefore, it is essential for computer science education (CSE) to consider learners' foundational knowledge, skills, and dispositions (Force, 2020), especially at a time when there is a shortage of competent professionals (e.g., Breaux & Moritz, 2021).

The backbone of CSE is the first introductory programming course—also known as the CS1 course—which introduces students to the world of control constructs, variables, functions, and stacks, among others. CS1 is known to be a course in which students report a wide range of difficulties (Petersen et al., 2016; Heilala et al., 2020), which can lead to problems such as retention (Marco-Galindo et al., 2022; Kinnunen & Malmi, 2006) or plagiarism (Maertens et al., 2022). It is worrying that more than a third of CS1 students worldwide drop out of the course (Watson & Li, 2014). Therefore, teaching and learning CS1 can benefit from, for example, predictive learning analytics, because data-driven pedagogical decision-making can affect teaching practices and intervention strategies to support at-risk students (Herodotou et al., 2019; Sghir et al., 2022; Mangaroska et al., 2020).

Recently, Van Petegem and colleagues (2022) introduced a privacy-friendly method for predicting student pass–fail outcomes using submission metadata. The method allows for early identification of at-risk students, enabling timely intervention. They assessed their approach in the HE context by predicting weekly pass–fail rates for two CS courses and interpreting the resulting models. In their explorative study, they found high prediction quality and verified the models by linking predicted pass odds to known success factors in introductory programming. Overall, the prediction framework of Van Petegem and colleagues (2022) provides an interpretable, privacy-centred approach for predictive learning analytics, with validation conducted in two CS courses.

While novel predictive approaches are being developed and implemented (Sghir et al., 2022; Prenkaj et al., 2020), predictive models of human behaviour and learning processes have become challenging. The critical issues in predictive learning analytics can be related to generalizability, transparency, and ethics (Mathrani et al., 2021). To explore these issues in the framework of Van Petegem and colleagues (2022), verification in a different CS1 context with a larger sample size is needed. On the other hand, some educational researchers have taken the approach of using both student self-reports and activity metadata when measuring and evaluating student behaviour (e.g., Roth et al., 2016; Jansen et al., 2020). So far, there is no clear consensus on how the use of self-report data obtained from the course can affect the prediction quality of the framework. Therefore, there is also a potential for extending and further evaluating the original framework by also considering self-report data.

Our primary research aim for this study is the *reproduction* and *extension* of the approach proposed by Van Petegem and colleagues (2022) in a separate CS1 course. Following Ihantola and colleagues' (2015, p. 48) recommendations and categorization of the research goals in programming education research, the proposed approach is one in which "a different experimenter is analyzing their own new dataset and following a new analysis method designed for the study in order to test the hypotheses in the baseline study." To apply Ihantola and colleagues' (2015) R.A.P. framework, this study (i) analyzes students' dropout risk in a CS1 course in another educational institution and learning environment, (ii) uses a new longitudinal dataset, and (iii) extends the data analysis by considering self-report data as part of the prediction. With the large pool of trace and self-report data from a CS1 course collected over the years 2015–2022, we validate and extend the original framework. Specifically, this study aims to answer the following research questions:

**RQ1** Can the original approach for student dropout prediction be reproduced in a different CS1 course context?

**RQ2** Does the addition of self-reported aptitude measures to the model improve the quality of the prediction?

**RQ3** Does the model consist of relevant and consistent features that can enhance the transparency and comprehensibility of the prediction results?

## 2. Background

### 2.1 Predictive Analytics in CSE

Early identification of students who are academically at risk has been investigated based on different kinds of learning analytics and datasets aiming to predict student dropouts and develop early warning systems and interventions (see Dawson et al., 2017). For instance, Jayaprakash and colleagues (2014) developed an early alert and intervention system for at-risk students by incorporating multiple parameters—student demographics, course-related data, engagement metrics from learning management systems, and grades. Similarly, Foster and Siddle (2020) designed a system focusing on non-engaged students, utilizing analytics about their log-ins to the virtual environments. Prediction in their model relies on learning analytics about students' log-ins to the virtual learning environments indicating students' engagement and progress with their studies.

Analytics aimed at identifying at-risk students, predicting dropout, and improving retention and academic success in CS1 has been an active topic for more than a few decades (Quille & Bergin, 2019; Becker & Quille, 2019). Quille and Bergin (2019) identified 47 articles that used predictive modelling of success in introductory programming courses. Aspects such as programming self-esteem, student prediction of final course grades, and mathematical ability are considered positive predictors of student performance (Quille & Bergin, 2016). Furthermore, the ease with which students grasp programming concepts is directly correlated with their performance (Bergin & Reilly, 2005). Notably, a student's early performance is a telling predictor in CS1 courses (Porter & Zingaro, 2014). Moreover, Hawlitschek and colleagues (2019) found that the time it took to submit an assignment predicted dropouts at an early phase, while factors like error frequency and continuous error streaks were more indicative of later dropouts. In other words, students' assignment submission behaviours can be used to predict academic performance (Kokoç et al., 2021). Broadly, both static and dynamic indicators related to students' backgrounds, behaviours, and performances may be related to course completion (Ifenthaler & Yau, 2020), and different combinations of academic and social data have been used to predict dropouts in CS (see Lacave et al., 2018). From a practical point of view, predictive learning analytics models capable of producing accurate predictions at an early stage can be used as early warning systems that can help teaching staff proactively implement supportive interventions (Olney et al., 2021; Liz-Dominguez et al., 2019).

### 2.2 On Reproducing Research in Predictive Learning Analytics

In Ihantola and colleagues' (2015) R.A.P taxonomy, reproduction studies involve new researchers (R) producing new results (P) and extending the analysis methods (A) of the original study. Reproduction studies aim to analyze the dependence of the hypotheses and results of the original study on the methods and specific procedures (Ihantola et al., 2015). In other words, reproduction thus allows adapting and extending the methods of the original study to fit the context and available data, improving overall generalizability. For example, Quille and Bergin (2018) reproduced their previous studies on factors affecting programming success (Bergin & Reilly, 2005; Quille & Bergin, 2016), enhancing their model by unveiling novel factors, such as social media usage, that improve programming success. Similarly, Castro-Wunsch and colleagues (2017) reproduced a method for identifying at-risk students in a different course and extended the analysis by assessing the efficacy of neural networks in predicting these students. Notably, while reproduction studies are conducted in CSE research, they are still quite scarce (Omer et al., 2023), despite their value in improving the validity of predictive learning analytics methods. Indeed, while the merit of reproduction studies is recognized, conducting them is challenging. Reproduction requires a deep understanding of the original study and approach, the results may fail to be generalizable, and the overall effort of reproduction is often seen as larger than that of conducting new original research (Ihantola et al., 2015).

In this paper, we focus on the predictive framework proposed and evaluated by Van Petegem and colleagues (2022). The original study evaluated a learning analytics approach for predicting whether a student will pass or fail introductory programming courses. The analysis encompassed two distinct introductory programming courses. Both were offered at the same university, albeit by different instructors, spanning from 2016 to 2018. During the courses, students were taught over a 12-week semester with weekly programming assignments. Additionally, both courses had two midterm and one final exam where students had to solve new programming problems. The final exam determined whether the students passed or failed the course. In their analysis, Van Petegem and colleagues (2022) collected student submission attempt data for each unit of tasks and trained pass–fail prediction classifiers based on the features extracted from the data. Among the details collected were timestamps for submission attempts, the accuracy of those attempts, and the nature of errors found in incorrect submissions. Rather than employ a selective approach, Van Petegem and colleagues (2022) included all data and features to explore the correlation between task submissions and course outcomes. They found that if the cohort size was sufficient, a single course edition's historical submission data could accurately predict pass–fail rates, as opposed to models considering students' socio-economic background (e.g., Kovacic, 2012). Further, the researchers explored the explainability of the models by interpreting the feature importances. Their framework provided highly explainable models on par with deep learning

models (e.g., Waheed et al., 2023) by connecting weekly task-solving behaviour to course-passing odds. Their model was verified by confirming known trends in CSE, such as timely completion of programming tasks, while emphasizing systematic problem-solving and solution iteration as primary factors for course success.

The framework and exploratory study undertaken by Van Petegem and colleagues (2022) demonstrated high predictive accuracy and offered preliminary validation of features influencing course-passing outcomes. Such results call for reproduction in another CS1 course context for various reasons. Specifically, while their models and interpretations are valuable, they are at the moment limited to the context of the university. To ascertain the broader validity of the findings, it's pivotal to reproduce the framework in a distinct university environment. Factors such as a larger sample size, varying lecturers, and different course dynamics could play a crucial role in confirming the conclusions about feature importances as observed in similar studies (e.g., Ihantola et al., 2015). Further, the original study exclusively focused analysis only on students who participated in the final exam. Such selection may skew the prediction of at-risk students, since students can drop out early or just not be able to attend the final exam. Because dropout can occur well before the final activity (Prenkaj et al., 2020), it is important to detect and assist potential early dropouts. Ignoring early dropouts or those who couldn't attend the final exam could potentially exclude a critical subset of the student demographic. Hence, pinpointing and aiding potential early dropouts emerges as a pressing research concern. Guided by these insights, this paper seeks to both validate and expand upon the original study. Our objective is to apply the established framework to a new CS1 course, amplify the sample size, and pivot our predictive analysis to encompass all student dropouts.

### 2.3 On Utilizing Self-Report Data in Predictive Learning Analytics

Self-reported data have been widely used in learning analytics applications, often in the form of questionnaires and interviews (e.g., Roth et al., 2016). Nevertheless, the reliance on self-reports in learning analytics is often a subject of debate. Several challenges are associated with self-reports, including their potential intrusiveness, the rapid obsolescence of data, issues with recall, questions that are either too vague or fail to capture the intended phenomenon, and, commonly, low participant response rates (Veenman, 2013; Roth et al., 2016). Further, when it comes to combining self-reports with automatically collected behaviour data, some studies question the benefits of self-reports. For instance, Zhou and Winne (2012) examined the modelling factors that influence academic achievement by comparing self-reported data to behavioural trace data. Their findings indicated a discrepancy between self-reported learning objectives and those identified in the trace data. Crucially, only the trace data effectively predicted academic achievement. In a more recent study, Choi and colleagues (2023) highlighted the challenges of equating survey responses with behaviour traces, especially in the context of analyzing achievement goals and predicting academic outcomes. Given these considerations, the role of self-reports in learning analytics warrants careful discussion, particularly in predictive learning analytics.

While self-reported data alone have potential issues, many studies show the benefits of combining the two data sources. In an empirical study and a critical review of recent studies, Tempelaar and colleagues (2020) proposed that although self-report measures have biases, they still have value in predicting academic performance. Their study suggested that combined with trace data, self-reports can provide a more accurate understanding of student learning and performance. They argue that the biases can add predictive power when explaining performance data and other questionnaire data. Further studies echo this argument. Ellis and colleagues (2017) used a questionnaire to gather self-report data on students' learning approaches. They combined this with observational data from an online learning environment to predict academic performance. They found that the combination of self-report and observational data identified three variables that significantly predicted academic performance: surface approach to study, frequency of accessing an online resource, and number of multiple-choice questions answered. Their results suggested that including self-report measures can improve analysis by providing a comprehensive view of learning experiences and enhancing predictive power. Additionally, Ifenthaler and colleagues (2022) found alignment between self-reported data on self-testing procedures and behaviour data from a learning analytics system. They proposed that combining self-report data with trace data for investigating learner engagement with resources like self-assessments can be more accurate and help validate analytics results. In a recent evaluation of self-report and behaviour data in learning analytics, Zhidkikh and colleagues (2023) analyzed student behaviour patterns and achievement in a small-scale classroom. They concluded that while both data sources yield different information about achievement, the results are not mutually exclusive. Instead, they can model the "why" behind high- and low-achieving students. While self-reports and trace data provide complementary insights, jointly analyzing them gives a more comprehensive understanding of factors influencing student achievement.

Overall, using self-reports in learning analytics is challenging. However, incorporating them could enhance our comprehension of the reasons some students drop out while others persevere. For example, grade expectations reported at the beginning of a CS1 course strongly indicate course achievement (Rountree et al., 2002). In practice, learning analytics research has to draw information from multiple sources, such as clickstreams, trace logs, interviews, observations, and demographic details, to fully understand students' metacognitive and self-regulation processes and their effect on achievement (Omer et al., 2023). When planning interventions and feedback systems with learning analytics, it's imperative to factor in students' individual attributes, such as self-efficacy and academic experiences (see Dawson et al., 2017). Self-reports can shed light on individual aspects, like

expected grades (see Dawson et al., 2017), that impact learning behaviours, and further help personalize feedback. In this study, we have access to a vast self-report data pool. We aim to reproduce and extend the original study by evaluating the benefits of including self-report data.

## 3. Research Context and Methods

### 3.1 CS1 Course Context

The twice-yearly CS1 course at the University of Jyväskylä spans 11 weeks, with our study focusing on the more popular autumn version. Each course week includes two lectures, practical tasks (i.e., "demos" or "homework"), and task review sessions for discussing model answers (Figure 1). Students must also complete a mini-project, typically a small game or console application, which they present to teaching assistants throughout the course. Additionally, students can attend voluntary workshops to get further assistance from teaching assistants on weekly tasks and the mini-project.

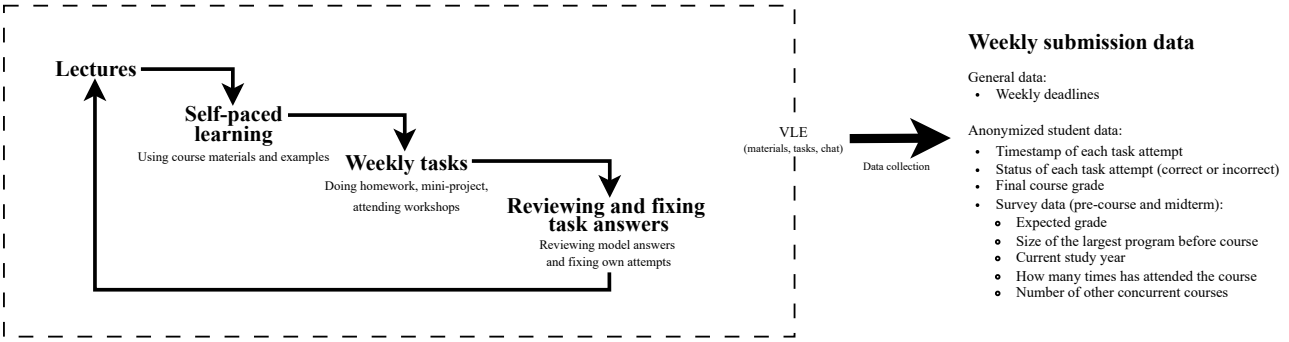


**Figure 1.** Design of the studied CS1 course and data collection procedure used in the present study. Each phase represents a learning phase of a course week cycle; the cycle repeats weekly for 11 weeks. The entire course is situated online in a virtual learning environment (VLE), from which weekly submission data are extracted for dropout prediction.

To pass the course, students must meet specific criteria: securing at least 40% of 66 total homework points, earning a minimum of 2 points per week from mandatory tasks, completing a mini-programming project, demonstrating debugging skills, and passing the final exam. Students can skip the final exam and pass the course with the lowest grade if they obtain 5 points from each weekly task set. The course also provides further flexibility, since students can pass even with a failed final exam as long as they complete extra tasks. Students with fewer than 7 weekly homework points are encouraged to revise their answers based on model answers before going to the next week's tasks. Figure 2 displays the weekly completion or pass rates for the course's autumn version between 2015 and 2021.

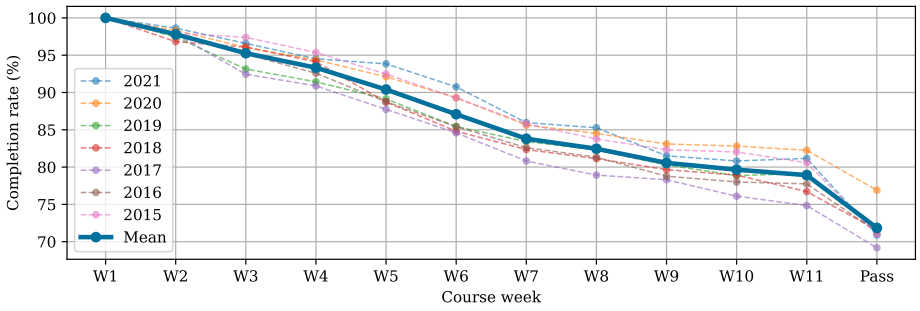


**Figure 2.** Completion rates for each weekly task set for the autumn version of the CS1 course and final pass rate for 2015–2021.

Weekly homework comprises 10–20 exercises, including one or two mandatory tasks per week. Students can choose their preferred tasks, with each task awarding different points. To pass, students must earn at least 40% of the total task points. Exercises are categorized into four difficulty levels: "basic" conceptual tasks usually aided by visual tools, "normal" programming tasks, "bonus" tasks that are doable using the week's knowledge, and "guru" tasks that require students to search for information from sources outside the course (e.g., online documentation). Higher-difficulty tasks yield more points. The homework features various exercise types, such as open-ended questions and programming tasks, with programming tasks becoming increasingly prevalent as the course advances.

### 3.1.1 Difference between Course Contexts

A summary of all the main differences between the CS1 contexts in this study and in the original study is presented in Table 1. The overall goals and topics taught in both courses are similar, but the differences lie in what tasks students complete and what students are graded on. Notably, the original courses included exams as the main measures of course achievement, while the current course is more oriented toward project work and allows students to complete the course in various ways. The larger dataset also means higher variance in students' background and prior skills, ensuring that the contexts are different enough for a meaningful replication of the original framework.

**Table 1.** Comparison of CS1 course study environments between the original article (Van Petegem et al., 2022) and the course used in this study.

| | Present study | Van Petegem et al. (2022) |
|---|---|---|
| **Online learning environment** | TIM (Isomöttönen et al., 2019) | Dodona (Van Petegem et al., 2023) |
| **Course duration** | 11 weeks | 13 weeks |
| **Major target students** | Computer science, information systems | Natural sciences |
| **% elective students** | 38.6%† | n/a |
| **Weekly task types** | Basic tool-assisted programming tasks, normal programming tasks, bonus tasks, "guru" tasks | Normal programming tasks |
| **Weekly task amount** | 10–20; students complete two mandatory tasks and choose the rest; must get 40% of total task points before the course ends | Six mandatory assignments |
| **Exams** | Final exam; not required to pass the course | Final exam; required to pass the course |
| **Other graded work** | Mini-programming project (usually a game) that is programmed throughout the course | Two midterms |
| **Metadata on student submissions** | Timestamp, score, final grade | Timestamp, evaluation status (different options for wrong submissions) |
| **Training data** | 2015–2021, autumn course ($N = 2,615$, avg. 374 students/year), weekly tasks | 2016–2018 ($N = 878$, avg. 293 students/year), weekly tasks and midterm evaluation grades |
| **Dropped-out students**‡ | Included in the analysis | Removed from the analysis |

† Statistics from years 2019–2021.

‡ Student are considered dropouts if they become inactive after a certain course week (see Prenkaj et al., 2020, Definition 2.3).

Besides course context, an important difference is made in what specifically is predicted. As mentioned in Section 2.2, Van Petegem and colleagues (2022) analyzed only students who took the final exam, focusing on pass–fail prediction. In our CS1 context, students may not need to attend the final exam to pass the course, which would affect only pass–fail prediction. To better adapt to the studied CS1 course, we extend our analysis to dropout prediction, since it is a more natural and pressing issue to predict. Specifically, we define a student as a dropout "if they do not have e-activities after the current phase" (Prenkaj et al., 2020). With this definition, our prediction and results better fit the studied CS1 course while still including the notion of the original pass–fail prediction.

## 3.2 Data Collection

The collected data included submission data extracted from the virtual learning environment used in the course, while the submission data included weekly deadlines, students' answer attempts with scores, and the final grade. In total, anonymized behaviour and survey data from 2,615 students from the 2015–2021 course years were included in the study.

For submission trace data, we followed the data collection approach of Van Petegem and colleagues (2022), with modifica-

tions to account for the different educational context[1]. We collected each student's submission data over every course week. The submission data included deadlines for each weekly task set, attempts for each task grouped by the weekly task set, and the student's final grade. The timestamp and the correctness status were recorded for each attempt, depending on the task type. For graded tasks, an attempt was marked correct if the awarded points were at least 50% of the maximum awardable from the tasks; otherwise, the attempt was marked incorrect. This approach is different from Van Petegem and colleagues (2022), and it was used because some tasks were awarded either partial or extra points based on how much of the task the student completed. Furthermore, all submissions to ungraded tasks were marked incorrect because there was no automated way to determine the correctness of these tasks. In contrast to Van Petegem and colleagues (2022), we did not assign different statuses to different kinds of erroneous attempts (e.g., "compile-time error" or "runtime error") because we did not have the data readily available for analysis. We also did not include exam data because the present course had neither a midterm nor a mandatory final exam.

In addition to the trace data, we collected and processed self-report data available from all the course years. The self-report data consisted of pre-course and midterm surveys that collected preliminary information about students' motivation and aptitudes to learn programming. Specifically, we extracted information about expected grades from the course (numerical, 1–5), students' longest written program prior to the course (ordinal, from none to over 5,000 lines of code), current active study year (numerical), how many times the student attended the course (numerical), and how many concurrent courses the student had (numerical). The self-reports do not include any specific cognitive or self-regulation measures because they were not collected during the available years. Nevertheless, we included the few available self-reported measures, since the overall available dataset is large enough to address RQ2. The extracted survey data were automatically merged with the students' trace data before anonymization and feature extraction.

### 3.3 Analysis Procedure

For early student dropout prediction, we applied the classifier-based prediction methodology outlined by Van Petegem and colleagues (2022), with adjustments to account for the available data. All student submission and survey data were processed into feature vectors with the following format: $(f_1, f_2, \ldots, f_n, \text{label})$, where $f_i$ is the numerical value of a single feature derived from submission data and label is the grade coded into the course status value (0 = did not complete the course, 1 = got a passing grade). The features used were derived from the 17 feature types described and used by Van Petegem and colleagues (2022). These types indirectly measure certain behavioural aspects of the students during the course, such as the "number of correct submissions before deadline," the "time between first submission and first correct submission," and the "number of wrong submissions." We derived the final features by measuring them for each course's week and by computing their sum and mean across the entire course. Thus, the same features were computed for each student separately for each weekly task set, allowing for the detection of dropouts based only on weekly behaviour. For the purpose of this study, we selected all feature types presented by Van Petegem and colleagues (2022) except two for which we did not have the necessary data: the number of compile-time errors and the number of runtime errors. Based on the survey data, we added five new self-report feature types mapped directly from the survey. Because the survey data did not change during the course, the features obtained from the surveys were appended to each weekly snapshot without grouping. For replication, we did not apply any feature selection approaches to pre-select features for classification, as was also done by Van Petegem and colleagues (2022). We therefore aimed for the same data-centric discovery of prediction quality and feature importances without feature engineering. Thus, overall, 200 features were computed for each student for each course week. A list of all the features used in this study is presented in Appendix A.

During the CS1 course, the students completed the tasks generally following the weekly cycle (Figure 1). Thus, student behaviour evolved weekly, which could affect prediction (Van Petegem et al., 2022). Instead of training the classifier on all student data at once, we inspected the "weekly snapshots," which included all student behaviour data up to the end of the weekly task set's deadline. Thus, we generated 11 snapshots of each student's feature vectors, one for each course week.

Our primary analysis consisted of training classifier models with feature vectors of weekly snapshots for each year. The classifiers were trained to predict each weekly snapshot's course status label based on the available features $f_i$. Following the analysis of Van Petegem and colleagues (2022), we selected the same four classifiers for evaluation: random forest, stochastic gradient descent, support vector machine, and logistic regression. To answer the research questions, we trained three model types:

- "trace data only" models, which include only the features derived from the task submission trace data;
- "self-report only" models, which include only the five self-report features extracted from student surveys; and
- "trace + self-report" models, which include a combination of trace data and self-report features.

For each weekly snapshot in the "trace data" and "trace + self-report" models, the hyperparameters were searched using 5-, 10-, and 20-fold cross-validation to obtain a more extensive set of models to evaluate. Additionally, the models were trained incrementally (by adding one year at a time to the classifier and evaluating on the year 2021) and yearly (by training only on

[1] All analysis code and trained models are available at https://gitlab.jyu.fi/dezhidki/jyughent-sdp-cs1.

one year and evaluating on the next available year). Finally, "self-report only" data were split into five folds before searching for hyperparameters using the same methodology, as outlined earlier.

## 4. Results

In total, we trained 2,280 classification models on CS1 student data from 2015 to 2021. The trained models can be grouped into two main types:

- incrementally trained models evaluated on year 2021 ($N = 792$), and
- models trained on one year at a time and evaluated on the next available course year ($N = 308$). Year 2021 was also used as training data and evaluated on year 2020.

For each type, the models were trained with two feature sets: one with only features extracted from task performance data and one with both performance features and features extracted from course surveys. Additionally, we trained models on only student survey data to compare prediction quality. However, the number of students who answered the survey ($N = 1,147$) was too low for yearly model training, since under 50% of students answered the surveys yearly. Instead, the "self-report only" models were trained by splitting the survey data 20-fold, resulting in $N = 80$ models. In total, 2,280 models were trained for evaluation, interpretation, and comparison.

### 4.1 Dropout Detection Evaluation

#### 4.1.1 Classification Quality

Following the approach of Van Petegem and colleagues (2022), we evaluated the performance of four classifiers for dropout prediction in 11 weekly snapshots from the CS1 course using only trace data. Figure 3 shows the accuracy measures for all trained models grouped by course week.

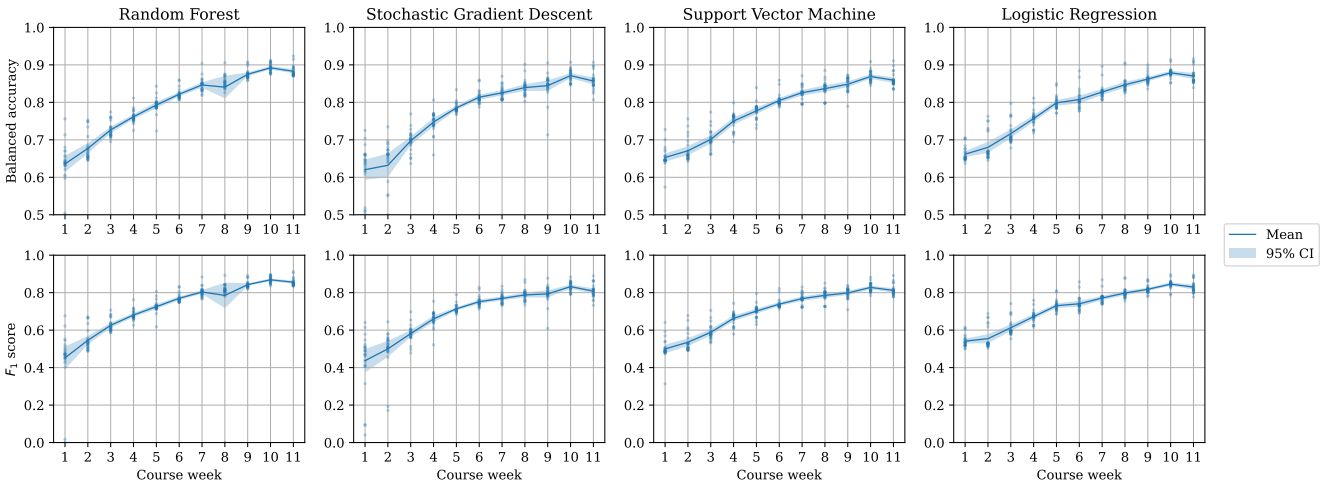


**Figure 3.** Performance of various classifiers for dropout prediction for the time sequence data from CS1 (trained only on submission trace data), as measured by balanced accuracy and the $F_1$ score. Each point represents the performance of a single model ($N_{\text{models}} = 1,100$) trained on a different number of course years and a varying number of cross-validation folds. The solid line represents the mean performance of the models trained on the given weekly snapshot of student data, while the shaded area represents the 95% confidence interval of the mean.

On average, all models provide strong dropout predictions, with an average balanced accuracy of 64%–88% and an $F_1$ score of 0.48–0.84. The prediction performance is on par with that of Van Petegem and colleagues (2022), who reported average balanced accuracies of 60%–80%. Using nonlinear classification models, such as random forest and logistic regression, appears to yield the best and most stable accuracy (i.e., low confidence intervals for balanced accuracy).

When evaluating the year 2021 prediction quality, the mean balanced accuracy for all models trained with the year 2020 is 77% (min: 62%, max: 87%, SD: 8.6%), and the accuracy for models trained with all the available years 2015–2022 is 79% (min: 64%, max: 88%, SD: 8.3%). From a practical standpoint, this means that using more recent course data may be a reasonable choice for dropout prediction despite a smaller dataset, since the training time is lower and the difference in classification results is negligible. Further, Figure 4 shows that the prediction quality of logistic regression appears to decrease slightly when using the oldest available data from 2015.

In contrast to Van Petegem and colleagues (2022), we did not use a midterm evaluation or exam grades as snapshots or prediction labels. Despite the lack of such data, we did not observe any effects on prediction accuracy in our results.

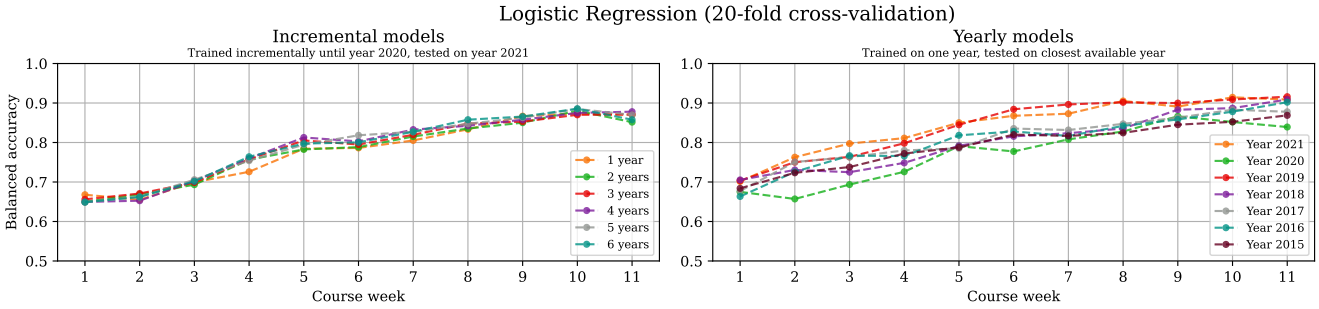


**Figure 4.** Performance of the logistic regression classifier models for dropout detection for the "trace data only" models, grouped by model type. Incremental models were trained by incrementally adding course data from previous years, with balanced accuracies reported for the year 2021. Yearly models were trained only on a specific course year, with the test year being the closest next course year (or year 2020 for the year 2021 model).

Nevertheless, our models display a similar "jump" in prediction accuracy around the midterm of the course (Figure 3) as that reported by Van Petegem and colleagues (2022). The jump in prediction accuracy is situated around the midterm of the course, around the same time when many students drop out (see Figure 2) and when the most difficult parts of the course begin.

#### 4.1.2 Interpreting Student Retention Using Classification Models

Next, we discuss the association of the various features extracted from the submission trace data with the predicted dropout of the models. We specifically focus on the explainability of the yearly models, since feature importance may vary depending on the course year and the surrounding context. In general, the explainability of classification models is based on inspecting the weight the model assigns to each feature.

In the case of the CS1 data studied, random forest and logistic classifiers provide the best prediction. However, logistic regression provides a more straightforward interpretation of the model. In the logistic regression model, each feature is assigned an importance weight that can be positive or negative. The feature importances affect the predicted course-passing probability, since positive features generally increase and negative features, consequently, decrease the predicted passing probability of a student. Therefore, a logistic regression model can be described as a heatmap in which each cell is coloured based on a feature's positive (red) or negative (blue) association with the course-passing prediction. The features can be sorted by various criteria to simplify the comparison. The resulting plot gives the model's "shape," which provides an overview of the model and allows for a quick visual comparison and detection of stable features. In what follows, we compare and interpret the model shapes of all one-year logistic regression "trace-only" models, along with a mean-value model in which each cell is the mean of the feature importance (Figure 5). All trained features and the median of their importance (MD imp.) are listed in Appendix B.

Across all years and course weeks, the most significant features for dropout prediction are related to the general task completion activity during the course. Specifically, the number of correct submissions before the deadline is positively associated with passing the course (MD imp. 0.069; Appendix B), while the number of incomplete tasks is more associated with higher dropout prediction (MD imp. $-0.031$; Appendix B). In other words, the models generally predict lower dropout for students who actively work and complete weekly tasks. We also observed the importance of persevering and iterating one's answer, since the number of incorrect answers (MD imp. 0.007; Appendix B) and the number of submissions after the first correct answer (MD imp. 0.011; Appendix B) are positively associated with continuing the course. Further, the models associated higher dropout with those students who turned in the tasks closer to the deadline (MD imp. $-0.004$; Appendix B) and those who took longer to come up with a correct solution (MD imp. $-0.01$; Appendix B).

While many feature importances are stable throughout the course and between years, the models show that some importances vary from week to week. Based on our observations, a feature may (Figure 5)

- have a positive (or neutral) importance up to a specific week, after which it may turn negative (e.g., "Demo 1: nr of exercises in 5 mins," "Demo 1: time of last submissions before deadline");
- have high variance in importance between years (e.g., "Demo 5: number of submissions after first correct submission"); or
- have occasional "jumps" in the feature importances for specific weeks (e.g., week 7 in year 2021, weeks 2 and 4 in year 2020).

Specifically, we also observed that the features measured for certain weeks were among the most important in terms of dropout prediction. The models suggest that the number of correct submissions in demos 1–4 has the most positive association with course passing (MD imp. $\geq 0.03$; Figure 5, Appendix B). Subsequently, leaving tasks incomplete during demos 1–6 is associated with dropout in the prediction models (MD imp. $\leq -0.01$; Figure 5, Appendix B). Thus, the first course weeks play

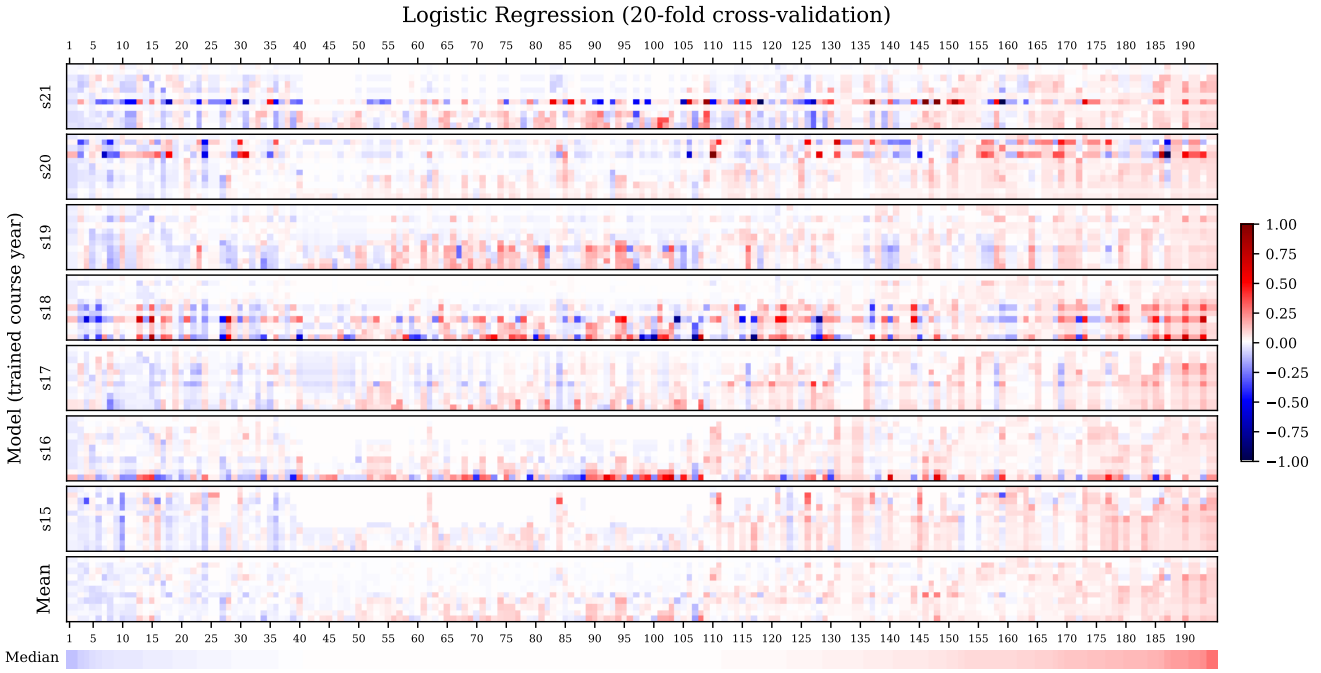


**Figure 5.** Shapes of the models trained using a logistic regression classifier on different course years. Each row represents the feature importances of a single model trained on a specific weekly snapshot (the first row uses only first-week data, while the last row uses data from all weeks). The columns depict the importance of a single feature over multiple models; all feature names are listed in Appendix B. The features are sorted by the median of all the importances, from the most negative to the most positive. A positive feature importance suggests a higher association with students passing the course, and vice versa.

an important role in reducing dropout, consistent with the general course attrition rates in the CS1 course (Figure 2).

### 4.2 Effect of Students' Self-Report Data on Prediction

The accuracy of the weekly snapshots of the "trace + self-report" models is shown in Figure 6. Generally, the trend of increasing accuracy over the number of course weeks remains the same compared with the "trace data only" model (Figure 3). Visually, there is a higher variation in accuracy, with the mean balanced accuracy ranging from 62% to 89% and the confidence intervals being greater for initial weeks than in the "trace data only" models. The higher variance could be attributed to the subjective nature of the self-report data, since self-reporting has higher variation than the behaviour data obtained from the virtual learning environment (Gonyea, 2005; Roth et al., 2016).

To further investigate the effect of including self-report data on dropout detection, we grouped all models by week and compared their accuracies using statistical tests. The resulting comparisons are reported in Tables 2 and 3. The initial accuracy scores were not normally distributed, as verified by the Shapiro–Wilk test. This variation could be due to some classifiers (notably stochastic gradient descent) having very high variability in the accuracy scores. To account for this, outliers were removed using the MAD-median rule (Rousseeuw & van Zomeren, 1990) prior to the statistical tests. Removing the outliers for each week left at least 70% of the scores; thus, parametric tests were used to compare the accuracies. Overall, the tests suggest a statistically significant difference in the accuracies between the "trace data only" and "trace + self-report" models for the first 1–4 weeks (Table 2). Specifically, including self-report data in the model resulted in a slight but notable improvement in prediction performance in the first four course weeks. The results show that students' self-reported data, such as grade expectation, number of times (re)attending the course, and study year, played a statistically significant role during the initial weeks of learning CS. However, for the rest of the weeks, the difference was not statistically significant, since students likely became more accustomed to the course progression and gained sufficient programming knowledge.

We also compared the models trained with self-report measures to identify dropout detection ($N = 80$) for completeness. Because the survey data were not tied to a specific week, we compared all the models with each other without weekly grouping (Table 2, "Total" row). The initial Shapiro–Wilk tests suggested that none of the models' accuracies were normally distributed, and removing the outliers would remove over half of the "self-report only" model accuracies. Therefore, we used a non-parametric Kruskal–Wallis test to test the difference between the "self-report only," "trace data only," and "trace + self-report" models. Because the difference between the model accuracies was statistically significant, a post hoc Mann–Whitney $U$-test was applied to check for pairwise differences (Table 3). The post hoc test clarified that the "self-report only" models performed

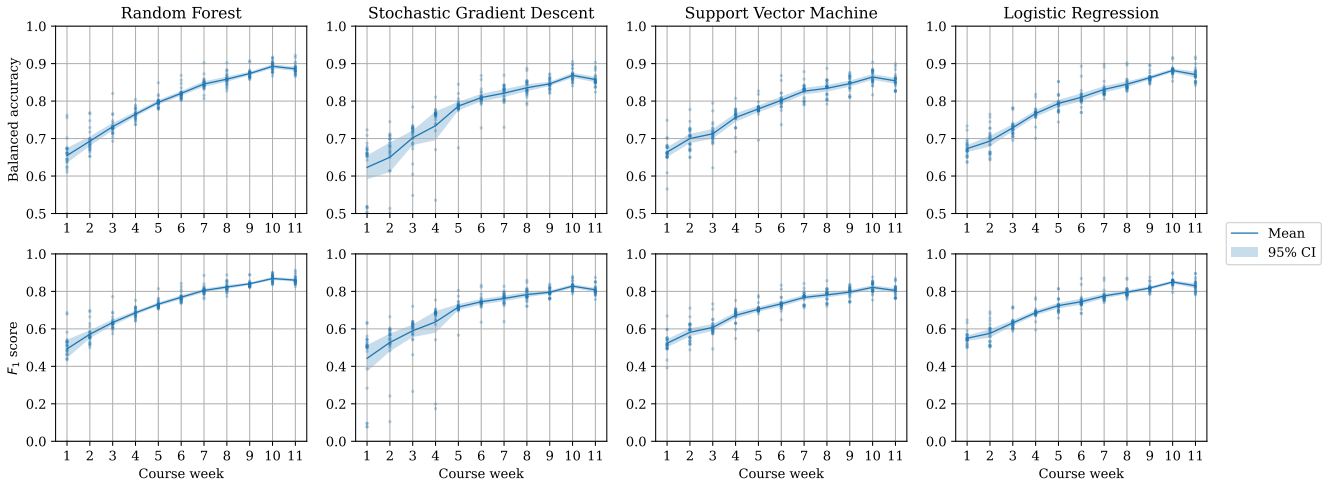


**Figure 6.** Performance of various classifiers for dropout prediction for time sequence data from CS1 (trained on both submission trace data and self-report data) as measured by balanced accuracy and the $F_1$ score. Each point represents the performance of a single model ($N_{\text{models}} = 1,100$) trained on a different number of course years and a varying amount of cross-validation folds. The solid line represents the mean performance of the models trained on the given weekly snapshot of student data, while the shaded area represents the 95% confidence interval of the mean.

worse than the other two model types. Moreover, no statistically significant difference was found in accuracy between the "trace data only" and "trace + self-report" models when comparing all models instead of weekly groups. Therefore, the results emphasize that self-report data serve as a way to enhance dropout prediction during the initial course weeks.

#### 4.2.1 Importance of Self-Report Features

Finally, we examined the effect of students' self-report data on model feature importances. This was conducted by extracting the feature importance values for the five self-reported features in the dataset from all the available "trace + self-report" models. However, because the yearly models contained a very low number of students who answered the survey (usually under 50%), the yearly features of the self-reported measures did not yield easily extendable results. Thus, we focused on interpreting the feature importance values of the models trained on the 2015–2020 data, since these models contained the highest number of students who answered the surveys ($N = 822$).

The weekly feature importances of the self-reported measures over 2015–2020 show how strongly students' initial beliefs about CS1 are associated with dropout. High grade expectations appear to improve course-passing rates throughout all course weeks (Figure 7, "expected grade"). Moreover, the prediction model associated students who repeat CS1 with higher dropout risk, with week 5 being the most critical point where attrition can occur (Figure 7, "how many times the student attended the course"). Thus, students who previously dropped out of the course might likely drop out when retaking the course, likely in week 5, in which the course difficulty increases. Interestingly, the prediction model associates lower course-passing rates with students who claimed to have more extensive previous experience in writing programs (Figure 7, "largest program written before course"). A similar and equally interesting observation is that students who start CS1 later in their studies might drop out more easily, especially in the beginning and toward the end of the course (Figure 7, "current study year"). Finally, the effect of concurrent courses should also be noted, since the feature has positive importance for all weeks (Figure 7).

## 5. Discussion

A common conceptualization describes learning analytics as a progressive process that starts with learners and advances through analytical endeavours to practical interventions that facilitate learning, after which a new cycle emerges (e.g., Clow, 2012). This so-called learning analytics loop consists of several phases. In terms of the prototypical lifecycle for learning analytics work (Wise et al., 2021), our study focused on the analytics part of the cycle by examining the replicability (i.e., Ihantola et al., 2015) of a predictive learning analytics approach in the CS1 context (i.e., Van Petegem et al., 2022). Viewing the global educational landscape, it is important to consider differences in educational settings when examining and developing learning analytics solutions that can be generalized for wider use (Viberg et al., 2023; Vatrapu, 2011). Moreover, valid models are needed to examine and critically evaluate the possible potential of predictive learning analytics approaches, such as dropout prediction. Therefore, the present study reproduced a previous modelling approach presented by Van Petegem and colleagues (2022) in another educational institution and learning environment (RQ1). The study also extended the analytics by complementing the

**Table 2.** Descriptive statistics for all the balanced accuracies of all student dropout prediction models ($N = 1,100$) grouped by weekly snapshot and model type. For each weekly snapshot, the statistical significance between the means of the different model types is reported alongside the effect size. The total contains statistics across all models and includes models trained only on aptitude data ($N = 80$).

| Snapshot | Model type[§] | Balanced accuracy | | | | Difference[*] | | |
|---|---|---|---|---|---|---|---|---|
| | | Mean | $\sigma$ | min | max | $p^{\dagger}$ | 95% CI | $d$ |
| Week 1 | T+S | 0.65 | 0.05 | 0.47 | 0.76 | $< 0.001$ | $[0.01, 0.02]$ | 0.94 |
| | T | 0.64 | 0.05 | 0.48 | 0.76 | | | |
| Week 2 | T+S | 0.68 | 0.06 | 0.28 | 0.78 | $< 0.001$ | $[0.02, 0.03]$ | 1.41 |
| | T | 0.66 | 0.05 | 0.44 | 0.76 | | | |
| Week 3 | T+S | 0.72 | 0.03 | 0.55 | 0.82 | $< 0.001$ | $[0.01, 0.02]$ | 0.79 |
| | T | 0.71 | 0.03 | 0.64 | 0.80 | | | |
| Week 4 | T+S | 0.76 | 0.05 | 0.36 | 0.82 | $< 0.001$ | $[0.01, 0.01]$ | 0.87 |
| | T | 0.75 | 0.02 | 0.66 | 0.81 | | | |
| Week 5 | T+S | 0.79 | 0.02 | 0.68 | 0.85 | 0.50 | | |
| | T | 0.79 | 0.02 | 0.72 | 0.85 | | | |
| Week 6 | T+S | 0.81 | 0.02 | 0.73 | 0.89 | 0.87 | | |
| | T | 0.81 | 0.02 | 0.78 | 0.88 | | | |
| Week 7 | T+S | 0.83 | 0.02 | 0.73 | 0.90 | 0.22 | | |
| | T | 0.83 | 0.02 | 0.80 | 0.90 | | | |
| Week 8 | T+S | 0.84 | 0.02 | 0.79 | 0.90 | 0.63 | | |
| | T | 0.84 | 0.04 | 0.50 | 0.91 | | | |
| Week 9 | T+S | 0.86 | 0.02 | 0.81 | 0.91 | 0.53 | | |
| | T | 0.86 | 0.02 | 0.71 | 0.91 | | | |
| Week 10 | T+S | 0.88 | 0.02 | 0.82 | 0.92 | 0.70 | | |
| | T | 0.88 | 0.02 | 0.85 | 0.91 | | | |
| Week 11 | T+S | 0.87 | 0.02 | 0.82 | 0.92 | 0.75 | | |
| | T | 0.87 | 0.02 | 0.81 | 0.92 | | | |
| | | Mean | $\sigma$ | min | max | $p^{\ddagger}$ | | $\varepsilon^2$ |
| **Total** | T+S | 0.79 | 0.08 | 0.28 | 0.92 | | | |
| | T | 0.79 | 0.08 | 0.44 | 0.92 | $< 0.001$ | | 0.10 |
| | S | 0.52 | 0.05 | 0.39 | 0.73 | | | |

[§] T = trained on trace data only, S = trained on self-report data only, T+S = trained on both trace data and self-report data.
[*] Score distributions are normalized using the MAD-median rule prior to testing for difference.
[†] Using Welch's $t$-test.
[‡] Using the Kruskal–Wallis test.

trace data with self-report measures (RQ2) and by examining the temporal consistency of the model features (RQ3). This study provides evidence of validity in terms of triangulation (see Denzin, 2009) using different data and additional methods within a different context.

### 5.1 RQ1: Can the Original Approach for Student Dropout Prediction Be Reproduced in a Different CS1 Course Context?

In terms of the reproduction of the analytics approach, this study showed that the dropout prediction proposed by Van Petegem and colleagues (2022) yielded similar performance in the CS context of this study. Potential disparities might be due to differences in course design, since the studied CS1 course prioritizes weekly assignments and a mini-project, while Van Petegem and colleagues (2022) assess students predominantly through midterm and final examinations. We also observed that precise prediction accuracy varies slightly depending on what course year was used for training. Such differences can be seen as a natural consequence of course tasks and contents evolving. Moreover, the predictive quality is good even when including all available features without any preliminary feature engineering. In essence, the results validate that the used prediction approach is robust to course implementation details and the selection of initial features before training the predictors.

Building on the original study, our findings indicate that data from the past one or two years are already enough for high-quality prediction. This observation likely comes down to the course design. Given that learning environments, instructional materials, organizational strategies, instructor expertise, and student demographics shift over time, minor yearly variances can

**Table 3.** Pairwise comparison between all weekly models trained with student trace data and those trained with self-report data (T+S, $N = 1,100$), weekly models trained only with student trace data (T, $N = 1,100$), and models trained on only self-report data (S, $N = 80$) using the Mann–Whitney $U$-test. The reported $p$-values are Bonferroni-corrected. For statistically significant differences, confidence intervals ($\alpha = 0.017$) for the difference between the medians and the common language effect size (CLES) are reported.

| | $p$ | 98.3% CI | CLES |
|---|---|---|---|
| T+S vs. T | 1.00 | - | - |
| T+S vs. S | $< 0.001$ | $[0.27, 0.31]$ | 0.99 |
| T vs. S | $< 0.001$ | $[0.26, 0.30]$ | 0.99 |

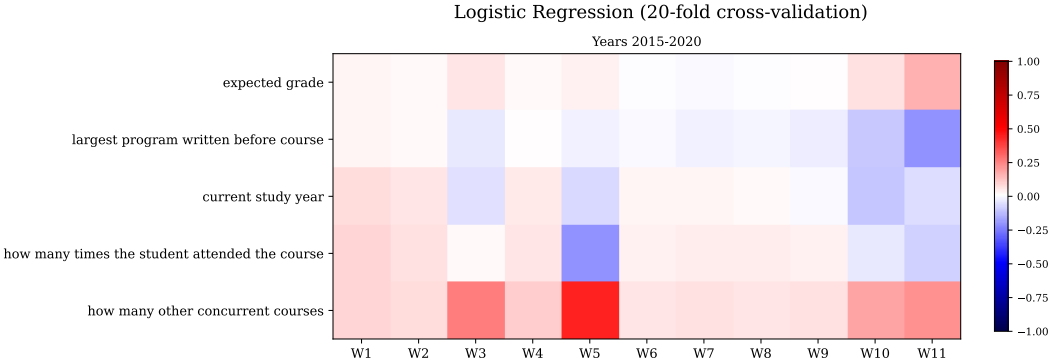


**Figure 7.** Heatmap of the feature importances for the self-reported measures for the logistic regression model (20-fold cross-validation) trained on course data from 2015 to 2020. Each tile represents the importance of a feature in a specific weekly snapshot of the course data. Positive feature importance during a specific week means that the model associated the feature with higher course-passing rates when considering only the task data up to the given week. Conversely, a negative feature importance for a week suggests a more negative association with course passing.

accumulate. This accumulation could potentially account for the larger deviation in accuracy over multiple years compared to just a few recent years. In the context of the studied CS1 course, for example, there has been a gradual transformation in course tasks since 2015, explaining the contrast in predictive quality between models trained on annual data and those trained incrementally.

### 5.2 RQ2: Does the Addition of Self-Reported Measures to the Model Improve the Quality of the Prediction?

Regarding the enrichment of trace data, our comparative analysis showed that the impact of self-reports on the accuracy of prediction models depends on the week of the course. When student behaviour data are available for the entire duration of the course, self-reported data did not significantly enhance the predictive quality. This finding is consistent with numerous similar studies on dropout prediction (e.g., Zhou & Winne, 2012; Choi et al., 2023). However, during the initial four weeks in the studied CS1 course, the self-reported data *did* enhance predictive accuracy. After the fourth week, the students' behavioural patterns and overall performance throughout the course outweighed the importance of self-report data collected at the beginning of the course. As such, self-report data related to motivation, academic goals, and academic aptitude can prove useful in identifying students at risk of dropout (Ifenthaler & Yau, 2020; Tempelaar et al., 2018), especially before the course becomes exceedingly challenging. Therefore, we argue that self-report data are crucial for improving *early* dropout prediction and gaining insights into the underlying causes of dropout.

The availability and quality of self-reports remains an important limitation for their use in learning analytics. In this study, even though there was a substantial volume of self-reports, only about half of the students participated in the questionnaire. In real-world scenarios, educators and researchers might lack access to any self-reported data. Nonetheless, educators can choose what data they can and will collect, and the models presented here offer good prediction quality even in the absence of self-reports. Furthermore, while our study incorporated only five self-reported measures and lacked more comprehensive questionnaires, the results are promising. This aligns with findings by Tempelaar and colleagues (2020), suggesting that even a little self-reported data can help in understanding academic outcomes without diminishing predictive power. A future study could consider collecting self-reports using more specific questionnaires like the MSLQ (Duncan & McKeachie, 2005), especially early on when students are still getting used to the course dynamics. In sum, the role of self-reports within the predictive framework warrants deeper exploration.

### 5.3 RQ3: Does the Model Consist of Relevant and Consistent Features That Can Enhance the Transparency and Comprehensibility of the Prediction Results?

In terms of the comprehensibility of the analytics, we also identified the relevant features associated with dropout in the CS1 course, thus verifying the interpretability of the model. Many of the significant features are consistent with what an educator would expect. For instance, (1) doing tasks is associated with continuing the course, (2) leaving tasks incomplete is associated with dropout, and (3) completing tasks on time and iterating the task after the initial correct submission improves passing. In addition, self-report features reveal trends that can be linked to further CSE research, such that (1) positive grade expectations can reduce dropout (e.g., Tek et al., 2018) and (2) prior programming experience may turn into an "experience trap," whereby a student's high prior experience with programming might increase dropout because they may choose to put less effort into the course basics (e.g., Lakanen & Isomöttönen, 2023). Hence, these features not only align with intuitive reasoning but also corroborate with academic literature, verifying the model's predictions.

Evaluating learning analytics models often revolves around their generalizability and explainability. To obtain generalizable results, the models and approaches used in predictive learning analytics should be assessed in pedagogical settings in different institutions and countries (Quille & Bergin, 2019). Moreover, one must consider the primary audience for these analytic results: diverse visualizations and analytics artifacts will have varying levels of explainability for researchers, educators, and students (Roscher et al., 2020). Our study replicates a prior exploratory study, and, therefore, current findings may necessitate knowledge of the CSE context and learning analytics to be explainable to a user of the prediction models. Thus the results may not be explainable to educators as they are, and future studies are required to build a more general, simple-to-use dashboard. Nonetheless, the framework and the models are transparent—a crucial characteristic in explainability of machine learning models, signifying that a person can contemplate and comprehend how the model works (Lepri et al., 2018; Roscher et al., 2020). Following the original approach of Van Petegem and colleagues (2022), our study used four analysis methods: random forest, stochastic gradient descent, support vector machine, and logistic regression. The results confirmed that, generally, all selected models performed well in predictive learning analytics, suggesting that found trends are genuinely data driven rather than mere "artifacts of the models." Transparency of the model is a prerequisite for fair machine learning models (Abdollahi & Nasraoui, 2018), affecting the practical choice of models to use in the analysis. To promote fair predictive learning analytics, researchers should find a balance between model accuracy and transparency (Deho et al., 2022). For our data, logistic regression provided such balance, as in the case of Van Petegem and colleagues (2022), although it remains unclear whether the same holds for other CS courses. Overall, learning analytics is an intersection of theory building, applying data science analysis, and design of visualizations and practice (Gašević, Kovanović, & Joksimović, 2017). Effective design requires a combination of robust, verifiable theory and methodology. This replication study and the results are the first step toward explainable prediction in CS1.

### 5.4 Evaluation of the Dropout Prediction Approach for CS1

The dropout prediction approach presented in this paper offers preliminary yet significant insights for practical application. The models and features presented here affirm several intuitions held by educators about factors affecting success in CS1: learning programming requires consistent engagement in programming tasks, passing is generally improved when students iterate on their answers and fix their own mistakes, and high course grade expectations often predict good grades. While these findings align with those of Van Petegem and colleagues (2022), our replication in a distinct CS1 course setting further verifies these intuitions with empirical evidence. Currently, this framework can serve as a tool to assess and enhance course design. Based on our results and studied CS1 context, using a game-based introduction to programming, providing flexibility in how the course can be passed, and encouraging students to review their answers before solving new tasks can support students in passing the course. Nonetheless, there is a need for additional research to refine this method, ensuring its utility for educators in identifying students at risk as early as possible.

Current developments in AI systems have the potential to respond better to individual learning needs. For example, Bouvier and colleagues (2021) found that continuous feedback could reduce the number of late submissions of programming assignments. At the same time, there is a growing concern that learning analytics may bring risks that conflict with ethical principles, such as those related to the privacy of students (Viberg et al., 2022). Therefore, it is important to study how this potential can be applied ethically to further improve innovative teaching and learning practices in HE. Our study is one attempt to promote the ethical development of AI in HE. The rationale of this study is that by seeking generalizable predictive models, we may be able to enhance transparency and fairness, which can lead to better ethical practices in predictive learning analytics. We consider that one approach for moving forward with the learning analytics loop is *proactive profiling*, which is the construction of nuanced prototypical student profiles based on the prediction results and semantics of the model features. Unsupervised machine learning and explainable methods can be used to create learner profiles (Heilala, 2022; Saarela et al., 2021), which provide qualitative insight into different learning experiences for teachers to enhance their pedagogical decision-making (Heilala et al., 2022). Similarly, the results of predictive learning analytics could be used as material for profiling what distinguishes individual characteristics in more detail than mere binary prediction probabilities and general averages. Proactive profiling is a

trade-off between fully disclosing the individual students' prediction results to the stakeholders and providing results only at the entire sample level. We argue that proactive profiling can advance the ethical use of predictive learning analytics by protecting individual students' privacy and avoiding the risk of being labelled because analytics deals with sufficiently large subgroups of students. The profiles and their qualitative interpretations based on model semantics can fuel pedagogical decision-making better than general descriptive representations.

Some limitations and broader educational perspectives should be considered for this study. First, the current real-life application of prediction is limited by the training time. The current analysis takes resources and time, since all hyperparameters are exhaustively checked during training. Due to these time constraints, the current analysis was limited to specific combinations of yearly data. The currently available methods for hyperparameter search optimization allow for the speeding up of training with negligible cost in accuracy (e.g., Li et al., 2018). Therefore, simplifying and optimizing dropout prediction should be considered to increase the usability of the prediction model in real-world courses. Additionally, the current interpretation of feature importances is limited primarily to associations with course passing or dropout—it is not known whether some factors outside those considered in the present study could also contribute to dropout. Future studies should thus consider statistically analyzing the features using marginal effects and predicted probabilities (see Niu, 2020). Second, dropout prediction is a type of analytics that complies with traditional educational practices in which students attend a course, complete the required tasks in a timely manner, and are ultimately awarded the final grade and course credits. In a sense, the educational outcome is defined as retaining or completing the course (e.g., Prenkaj et al., 2020). However, completing a course leads only to an official credential; it is not necessarily an indicator of obtained competence, and vice versa (Jarvis, 2010, p. 216). The high dropout rate in CS1 education, which means low achievement in terms of external outcomes, could lead stakeholders to overemphasize it as a problem needing a resolution instead of examining and addressing the underlying causes of students' struggles or critically considering the foundational structures of educational processes. Finally, dropout prediction should not distract educators from considering alternative educational structures and pedagogical approaches in CS1, such as collaborative learning (e.g., Astrachan & Briggs, 2012), specifically in the age of sophisticated language models capable of completing programming assignments on students' behalf (e.g., Biderman & Raff, 2022). For example, struggling students may have to be conceptualized differently than in more traditional educational practices.

## 6. Conclusion

In conclusion, this study successfully reproduced a privacy-friendly student dropout prediction approach in a different CS1 context. Further, combining trace data with limited self-reports on motivation and prior experience significantly improved early detection in the first four course weeks. The study also identified behavioural trends and self-reported factors associated with dropout in CS1. For both trace data and self-reports, the extracted features aligned with known factors like timely submission, task iteration, and positive expectations. Overall, the predictive features were consistent and interpretable, verifying the transparency of the machine learning approach. However, limitations remain, including training time constraints and the basic self-reports collected. Future work should explore additional self-reported and contextual factors beyond behaviour data to enhance the early identification of at-risk students. Researchers must also balance prediction accuracy with model interpretability to ensure ethical, fair use of analytics.

It is important to detect struggling students and support them as early as possible. When students lag behind during the first weeks of a course, it may be hard for them to catch up (e.g., Porter & Zingaro, 2014). Student aptitude has already been shown to be associated with passing/failing (e.g., Tek et al., 2018; Gašević, Jovanović, et al., 2017). Therefore, our results show that combining this information with the logged student performance can enhance predictions in the first few weeks. Future research could further identify other factors beyond self-report aptitude measures and student behaviour that can be helpful for enhancing dropout predictions. Furthermore, despite the importance of identifying struggling students, it may be even more important to obtain detailed insight into their learning processes. This insight is imperative in understanding what specific support can be provided to prevent students from failing/dropping out. In addition to cognitive and metacognitive processes, such as self-regulated learning, self-efficacy, and motivation, evidence has emerged on the importance of emotions for HE students' study processes (Postareff et al., 2017). A better understanding of what students go through—cognitively, metacognitively, and emotionally—and how these processes fluctuate during the first weeks of a course may be important to see the whole picture in terms of dropout. To respond to these needs, physiological data on emotions can be integrated with dropout prediction methods to enable a better understanding of how study processes unfold over time.

## Declaration of Conflicting Interest

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

## Funding

The publication of this article received financial support from the Academy of Finland (grant number 353325) and the Research Foundation—Flanders (FWO) for ELIXIR Belgium (I002819N).

## References

Abdollahi, B., & Nasraoui, O. (2018). Transparency in fair machine learning: The case of explainable recommender systems. In J. Zhou & F. Chen (Eds.), *Human and machine learning: Visible, explainable, trustworthy and transparent* (pp. 21–35). Springer International Publishing. https://doi.org/10.1007/978-3-319-90403-0_2

Astrachan, O., & Briggs, A. (2012). The CS principles project. *ACM Inroads*, *3*(2), 38–42. https://doi.org/10.1145/2189835.2189849

Becker, B. A., & Quille, K. (2019). 50 years of CS1 at SIGCSE: A review of the evolution of introductory programming education research. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (SIGCSE 2019), 27 February–2 March 2019, Minneapolis, Minnesota, USA (pp. 338–344). ACM. https://doi.org/10.1145/3287324.3287432

Bergin, S., & Reilly, R. (2005, February 23). Programming: Factors that influence success. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education* (SIGCSE 2005), 23–27 February 2005, St. Louis, Missouri, USA (pp. 411–415). ACM. https://doi.org/10.1145/1047344.1047480

Biderman, S., & Raff, E. (2022). Fooling MOSS detection with pretrained language models. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (CIKM 2022), 17–21 October 2022, Atlanta, Georgia, USA (pp. 2933–2943). ACM. https://doi.org/10.1145/3511808.3557079

Bosch, T. (Ed.). (2022). *"You are not expected to understand this": How 26 lines of code changed the world*. Princeton University Press. https://www.degruyter.com/document/doi/10.1515/9780691230818/html

Bouvier, D., Lovellette, E., & Matta, J. (2021). Overnight feedback reduces late submissions on programming projects in CS1. In *Proceedings of the 23rd Australasian Computing Education Conference* (ACE 2021), 2–4 February 2021, online (pp. 176–180). ACM. https://doi.org/10.1145/3441636.3442319

Breaux, T., & Moritz, J. (2021). The 2021 software developer shortage is coming. *Communications of the ACM*, *64*(7), 39–41. https://doi.org/10.1145/3440753

Castro-Wunsch, K., Ahadi, A., & Petersen, A. (2017, March 8). 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* (SIGCSE 2017), 8–11 March 2017, Seattle, Washington, USA (pp. 111–116). ACM. https://doi.org/10.1145/3017680.3017792

Choi, H., Winne, P. H., Brooks, C., Li, W., & Shedden, K. (2023). Logs or self-reports? Misalignment between behavioral trace data and surveys when modeling learner achievement goal orientation. In *Proceedings of the 13th International Learning Analytics and Knowledge Conference* (LAK 2023), 13–17 March 2023, Arlington, Texas, USA (pp. 11–21). ACM. https://doi.org/10.1145/3576050.3576052

Clow, D. (2012). The learning analytics cycle: Closing the loop effectively. In *Proceedings of the Second International Conference on Learning Analytics and Knowledge* (LAK 2012), 29 April–2 May 2012, Vancouver, British Columbia, Canada (pp. 134–138). ACM. https://doi.org/10.1145/2330601.2330636

Dawson, S., Jovanovic, J., Gašević, D., & Pardo, A. (2017). From prediction to impact: Evaluation of a learning analytics retention program. In *Proceedings of the Seventh International Conference on Learning Analytics and Knowledge* (LAK 2017), 13–17 March 2017, Vancouver, British Columbia, Canada (pp. 474–478). ACM. https://doi.org/10.1145/3027385.3027405

Deho, O. B., Zhan, C., Li, J., Liu, J., Liu, L., & Duy Le, T. (2022). How do the existing fairness metrics and unfairness mitigation algorithms contribute to ethical learning analytics? *British Journal of Educational Technology: Journal of the Council for Educational Technology*, *53*(4), 822–843. https://doi.org/10.1111/bjet.13217

Denzin, N. K. (2009). *The research act: A theoretical introduction to sociological methods*. Routledge. https://doi.org/10.4324/9781315134543

Duncan, T. G., & McKeachie, W. J. (2005). The making of the Motivated Strategies for Learning Questionnaire. *Educational Psychologist*, *40*(2), 117–128. https://doi.org/10.1207/s15326985ep4002_6

Ellis, R. A., Han, F., & Pardo, A. (2017). Improving learning analytics—Combining observational and self-report data on student learning. *Journal of Educational Technology & Society*, *20*(3), 158–169. https://www.jstor.org/stable/26196127

Force, C. T. (2020). *Computing curricula 2020 CC2020: Paradigms for global computing education*. ACM, IEEE. https://doi.org/10.1145/3467967

Foster, E., & Siddle, R. (2020). The effectiveness of learning analytics for identifying at-risk students in higher education. *Assessment & Evaluation in Higher Education*, *45*(6), 842–854. https://doi.org/10.1080/02602938.2019.1682118

Gašević, D., Jovanović, J., Pardo, A., & Dawson, S. (2017). Detecting learning strategies with analytics: Links with self-reported measures and academic performance. *Journal of Learning Analytics*, *4*(2), 113–128. https://doi.org/10.18608/jla.2017.42.10

Gašević, D., Kovanović, V., & Joksimović, S. (2017). Piecing the learning analytics puzzle: A consolidated model of a field of research and practice. *Learning: Research and Practice*, *3*(1), 63–78. https://doi.org/10.1080/23735082.2017.1286142

Gonyea, R. M. (2005). Self-reported data in institutional research: Review and recommendations. *New Directions for Institutional Research*, *2005*(127), 73–89. https://doi.org/10.1002/ir.156

Hawlitschek, A., Köppen, V., Dietrich, A., & Zug, S. (2019). Drop-out in programming courses—Prediction and prevention. *Journal of Applied Research in Higher Education*, *12*(1), 124–136. https://doi.org/10.1108/JARHE-02-2019-0035

Heilala, V. (2022). *Learning analytics with learning and analytics: Advancing student agency analytics* [Doctoral dissertation, JYU Dissertations 512]. University of Jyvaskyla. https://jyx.jyu.fi/handle/123456789/80877

Heilala, V., Jääskelä, P., Kärkkäinen, T., & Saarela, M. (2020). Understanding the study experiences of students in low agency profile: Towards a smart education approach. In A. E. Moussati, K. Kpalma, M. G. Belkasmi, M. Saber, & S. Guégan (Eds.), *Advances in smart technologies: Applications and case studies* (pp. 498–508). Springer International Publishing. https://doi.org/10.1007/978-3-030-53187-4_54

Heilala, V., Jääskelä, P., Saarela, M., Kuula, A.-S., Eskola, A., & Kärkkäinen, T. (2022). "Sitting at the stern and holding the rudder": Teachers' reflections on action in higher education based on student agency analytics. In L. Chechurin (Ed.), *Digital teaching and learning in higher education: Developing and disseminating skills for blended learning* (pp. 71–91). Springer International Publishing. https://doi.org/10.1007/978-3-031-00801-6_4

Herodotou, C., Rienties, B., Boroowa, A., Zdrahal, Z., & Hlosta, M. (2019). A large-scale implementation of predictive learning analytics in higher education: The teachers' role and perspective. *Educational Technology Research and Development: ETR & D*, *67*(5), 1273–1306. https://doi.org/10.1007/s11423-019-09685-0

Ifenthaler, D., Schumacher, C., & Kuzilek, J. (2022). Investigating students' use of self-assessments in higher education using learning analytics. *Journal of Computer Assisted Learning*, *39*(1), 255–268. https://doi.org/10.1111/jcal.12744

Ifenthaler, D., & Yau, J. Y.-K. (2020). Reflections on different learning analytics indicators for supporting study success. *International Journal of Learning Analytics and Artificial Intelligence for Education (iJAI)*, *2*(2), 4–23. https://doi.org/10.3991/ijai.v2i2.15639

Ihantola, P., Vihavainen, A., Ahadi, A., Butler, M., Börstler, J., Edwards, S. H., Isohanni, E., Korhonen, A., Petersen, A., Rivers, K., Rubio, M. Á., Sheard, J., Skupas, B., Spacco, J., Szabo, C., & Toll, D. (2015, July). Educational data mining and learning analytics in programming: Literature review and case studies. In *Proceedings of the 2015 Innovation and Technology in Computer Science Education Conference on Working Group Reports* (ITiCSE 2015), 4–8 July 2015, Vilnius, Lithuania (pp. 41–63). ACM. https://doi.org/10.1145/2858796.2858798

Isomöttönen, V., Lakanen, A.-J., & Lappalainen, V. (2019). Less is more! Preliminary evaluation of multi-functional document-based online learning environment. In *Proceedings of the 2019 IEEE Frontiers in Education Conference* (FIE 2019), 16–19 October 2019, Cincinnati, Ohio, USA (pp. 1–5). IEEE. https://doi.org/10.1109/FIE43999.2019.9028353

Jansen, R. S., van Leeuwen, A., Janssen, J., & Kester, L. (2020). A mixed method approach to studying self-regulated learning in MOOCs: Combining trace data with interviews. *Frontline Learning Research*, *8*(2), 35–64. https://doi.org/10.14786/flr.v8i2.539

Jarvis, P. (2010). Assessing and evaluating. In *Adult education and lifelong learning* (4th ed., pp. 210–226). Routledge. https://doi.org/10.4324/9780203718100

Jayaprakash, S. M., Moody, E. W., Lauría, E. J., Regan, J. R., & Baron, J. D. (2014). Early alert of academically at-risk students: An open source analytics initiative. *Journal of Learning Analytics*, *1*(1), 6–47. https://doi.org/10.18608/jla.2014.11.3

Kinnunen, P., & Malmi, L. (2006). Why students drop out CS1 course? In *Proceedings of the Second International Workshop on Computing Education Research* (ICER 2006), 9–10 September 2006, Canterbury, UK (pp. 97–108). ACM. https://doi.org/10.1145/1151588.1151604

Kokoç, M., Akçapınar, G., & Hasnine, M. N. (2021). Unfolding students' online assignment submission behavioral patterns using temporal learning analytics. *Educational Technology & Society*, *24*(1), 223–235. https://www.jstor.org/stable/26977869

Kori, K., Pedaste, M., & Must, O. (2017). Integration of Estonian higher education information technology students and its effect on graduation-related self-efficacy. In P. Zaphiris & A. Ioannou (Eds.), *Learning and collaboration technologies. Technology in education* (pp. 435–448). Springer International Publishing. https://doi.org/10.1007/978-3-319-58515-4_33

Kovacic, Z. (2012). Predicting student success by mining enrolment data. *Research in Higher Education*, *15*. http://hdl.handle.net/11072/1486

Lacave, C., Molina, A. I., & Cruz-Lemus, J. A. (2018). Learning analytics to identify dropout factors of computer science studies through Bayesian networks. *Behaviour & Information Technology*, *37*(10-11), 993–1007. https://doi.org/10.1080/0144929X.2018.1485053

Lakanen, A.-J., & Isomöttönen, V. (2023). CS1: Intrinsic motivation, self-efficacy, and effort. *Informatics in Education*, *22*(4), 651–670. https://doi.org/10.15388/infedu.2023.26

Lepri, B., Oliver, N., Letouzé, E., Pentland, A., & Vinck, P. (2018). Fair, transparent, and accountable algorithmic decision-making processes. *Philosophy & Technology*, *31*(4), 611–627. https://doi.org/10.1007/s13347-017-0279-x

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, *18*(185), 1–52. http://jmlr.org/papers/v18/16-558.html

Liz-Dominguez, M., Caeiro-Rodriguez, M., Llamas-Nistal, M., & Mikic-Fonte, F. (2019). Predictors and early warning systems in higher education—A systematic literature review. In M. Caeiro-Rodríguez, Á. Hernández-García, & P. Muñoz-Merino (Eds.), *Learning Analytics Summer Institute Spain 2019: Learning Analytics in Higher Education*. https://ceur-ws.org/Vol-2415/paper08.pdf

Maertens, R., Van Petegem, C., Strijbol, N., Baeyens, T., Jacobs, A. C., Dawyndt, P., & Mesuere, B. (2022). Dolos: Language-agnostic plagiarism detection in source code. *Journal of Computer Assisted Learning*, *38*(4), 1046–1061. https://doi.org/10.1111/jcal.12662

Mangaroska, K., Sharma, K., Gasevic, D., & Giannakos, M. (2020). Multimodal learning analytics to inform learning design: Lessons learned from computing education. *Journal of Learning Analytics*, *7*(3), 79–97. https://doi.org/10.18608/jla.2020.73.7

Marco-Galindo, M.-J., Minguillón, J., García-Solórzano, D., & Sancho-Vinuesa, T. (2022). Why do CS1 students become repeaters? *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, *17*(3), 245–253. https://doi.org/10.1109/RITA.2022.3191288

*Markets of tomorrow report 2023: Turning technologies into new sources of global growth*. (2023). World Economic Forum. Retrieved February 28, 2023, from https://www.weforum.org/reports/markets-of-tomorrow-report-2023-turning-technologies-into-new-sources-of-global-growth/

Mathrani, A., Susnjak, T., Ramaswami, G., & Barczak, A. (2021). Perspectives on the challenges of generalizability, transparency and ethics in predictive learning analytics. *Computers and Education Open*, *2*, 100060. https://doi.org/10.1016/j.caeo.2021.100060

Niu, L. (2020). A review of the application of logistic regression in educational research: Common issues, implications, and suggestions. *Educational Review*, *72*(1), 41–67. https://doi.org/10.1080/00131911.2018.1483892

Olney, T., Walker, S., Wood, C., & Clarke, A. (2021). Are we living in LA (P)LA land? *Journal of Learning Analytics*, *8*(3), 45–59. https://doi.org/10.18608/jla.2021.7261

Omer, U., Tehseen, R., Farooq, M. S., & Abid, A. (2023). Learning analytics in programming courses: Review and implications. *Education and Information Technologies*, *28*(9), 11221–11268. https://doi.org/10.1007/s10639-023-11611-0

Petersen, A., Craig, M., Campbell, J., & Tafliovich, A. (2016). Revisiting why students drop CS1. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, 24–27 November 2016, Koli, Finland (pp. 71–80). ACM. https://doi.org/10.1145/2999541.2999552

Porter, L., & Zingaro, D. (2014). Importance of early performance in CS1: Two conflicting assessment stories. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (SIGCSE 2014), 5–8 March 2014, Atlanta, Georgia, USA (pp. 295–300). ACM. https://doi.org/10.1145/2538862.2538912

Postareff, L., Mattsson, M., Lindblom-Ylänne, S., & Hailikari, T. (2017). The complex relationship between emotions, approaches to learning, study success and study progress during the transition to university. *Higher Education*, *73*(3), 441–457. https://doi.org/10.1007/s10734-016-0096-7

Prenkaj, B., Velardi, P., Stilo, G., Distante, D., & Faralli, S. (2020). A survey of machine learning approaches for student dropout prediction in online courses. *ACM Computing Surveys*, *53*(3), 1–34. https://doi.org/10.1145/3388792

Quille, K., & Bergin, S. (2016). Programming: Further factors that influence success. *PPIG 2016—27th Annual Workshop*, 7–10 September 2016, Cambridge, UK. https://www.ppig.org/papers/2016-ppig-27th-quille/

Quille, K., & Bergin, S. (2018). Programming: Predicting student success early in CS1. A re-validation and replication study. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (ITiCSE 2018), 2–4 July 2018, Larnaca, Cyprus (pp. 15–20). ACM. https://doi.org/10.1145/3197091.3197101

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. https://doi.org/10.1080/08993408.2019.1612679

Roscher, R., Bohn, B., Duarte, M. F., & Garcke, J. (2020). Explainable machine learning for scientific insights and discoveries. *IEEE Access*, *8*, 42200–42216. https://doi.org/10.1109/ACCESS.2020.2976199

  

Roth, A., Ogrin, S., & Schmitz, B. (2016). Assessing self-regulated learning in higher education: A systematic literature review of self-report instruments. *Educational Assessment, Evaluation and Accountability*, *28*(3), 225–250. https://doi.org/10.1007/s11092-015-9229-2

Rountree, N., Rountree, J., & Robins, A. (2002). Predictors of success and failure in a CS1 course. *ACM SIGCSE Bulletin*, *34*(4), 121–124. https://doi.org/10.1145/820127.820182

Rousseeuw, P. J., & van Zomeren, B. C. (1990). Unmasking multivariate outliers and leverage points. *Journal of the American Statistical Association*, *85*(411), 633–639. https://doi.org/10.1080/01621459.1990.10474920

Saarela, M., Heilala, V., Jääskelä, P., Rantakaulio, A., & Kärkkäinen, T. (2021). Explainable student agency analytics. *IEEE Access*, *9*, 137444–137459. https://doi.org/10.1109/ACCESS.2021.3116664

Sghir, N., Adadi, A., & Lahmer, M. (2022). Recent advances in predictive learning analytics: A decade systematic review (2012–2022). *Education and Information Technologies*, *28*, 8299–8333. https://doi.org/10.1007/s10639-022-11536-0

Tek, F. B., Benli, K. S., & Deveci, E. (2018). Implicit theories and self-efficacy in an introductory programming course. *IEEE Transactions on Education*, *61*(3), 218–225. https://doi.org/10.1109/TE.2017.2789183

Tempelaar, D., Rienties, B., Mittelmeier, J., & Nguyen, Q. (2018). Student profiling in a dispositional learning analytics application using formative assessment. *Computers in Human Behavior*, *78*, 408–420. https://doi.org/10.1016/j.chb.2017.08.010

Tempelaar, D., Rienties, B., & Nguyen, Q. (2020). Subjective data, objective data and the role of bias in predictive modelling: Lessons from a dispositional learning analytics application. *PloS one*, *15*(6), e0233977. https://doi.org/10.1371/journal.pone.0233977

Van Petegem, C., Deconinck, L., Mourisse, D., Maertens, R., Strijbol, N., Dhoedt, B., De Wever, B., Dawyndt, P., & Mesuere, B. (2022). Pass/fail prediction in programming courses. *Journal of Educational Computing Research*, *61*(1), 68–95. https://doi.org/10.1177/07356331221085595

Van Petegem, C., Maertens, R., Strijbol, N., Van Renterghem, J., Van der Jeugt, F., De Wever, B., Dawyndt, P., & Mesuere, B. (2023). Dodona: Learn to code with a virtual co-teacher that supports active learning. *SoftwareX*, *24*, 101578. https://doi.org/10.1016/j.softx.2023.101578

Vatrapu, R. (2011). Cultural considerations in learning analytics. In *Proceedings of the First International Conference on Learning Analytics and Knowledge* (LAK 2011), 27 February–1 March 2011, Banff, Alberta, Canada (pp. 127–133). ACM. https://doi.org/10.1145/2090116.2090136

Veenman, M. V. J. (2013). Assessing metacognitive skills in computerized learning environments. In R. Azevedo & V. Aleven (Eds.), *International handbook of metacognition and learning technologies* (pp. 157–168). Springer. https://doi.org/10.1007/978-1-4419-5546-3_11

Viberg, O., Jivet, I., & Scheffel, M. (2023). Designing culturally aware learning analytics: A value sensitive perspective. In O. Viberg & Å. Grönlund (Eds.), *Practicable learning analytics* (pp. 177–192). Springer International Publishing. https://doi.org/10.1007/978-3-031-27646-0_10

Viberg, O., Mutimukwe, C., & Grönlund, Å. (2022). Privacy in LA research. *Journal of Learning Analytics*, *9*(3), 169–182. https://doi.org/10.18608/jla.2022.7751

Waheed, H., Hassan, S.-U., Nawaz, R., Aljohani, N. R., Chen, G., & Gašević, D. (2023). Early prediction of learners at risk in self-paced education: A neural network approach. *Expert Systems with Applications*, *213*, 118868. https://doi.org/10.1016/j.eswa.2022.118868

Watson, C., & Li, F. W. (2014). Failure rates in introductory programming revisited. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education* (ITiCSE 2014), 21–25 June 2014, Uppsala, Sweden (pp. 39–44). ACM. https://doi.org/10.1145/2591708.2591749

Wise, A. F., Knight, S., & Ochoa, X. (2021). What makes learning analytics research matter. *Journal of Learning Analytics*, *8*(3), 1–9. https://doi.org/10.18608/jla.2021.7647

Zhidkikh, D., Saarela, M., & Kärkkäinen, T. (2023). Measuring self-regulated learning in a junior high school mathematics classroom: Combining aptitude and event measures in digital learning materials. *Journal of Computer Assisted Learning*, *39*(6), 1834–1851. https://doi.org/10.1111/jcal.12842

Zhou, M., & Winne, P. H. (2012). Modeling academic achievement by self-reported versus traced goal orientation. *Learning and Instruction*, *22*(6), 413–419. https://doi.org/10.1016/j.learninstruc.2012.03.004

# Appendices

## Appendix A: All Feature Types and Computed Features

**Table 4.** List of all feature types used in this study. For each trace data feature type, 13 features were derived: one for each demo (i.e., course week task set), one containing the sum, and one containing the mean across the course.

| Feature type | No. of derived features | Notes |
|---|---|---|
| **Features from trace data** | | |
| total nr of submissions | 13 | |
| total nr of exercises with no submissions | 13 | |
| nr of submissions wrong | 13 | |
| time of 1st submission before deadline | 13 | |
| time of last submissions before deadline | 13 | |
| number of correct submissions before deadline | 13 | |
| number of correct submissions | 13 | |
| number of submissions after first correct submission | 13 | |
| number of submissions before first correct | 13 | |
| time between first and last submissions of a series | 13 | |
| time between first submission and first correct submission | 13 | |
| nr of exercises in 5 mins | 13 | |
| nr of exercises in 15 mins | 13 | |
| nr of exercises in 2 hours | 13 | |
| nr of exercises in 24 hours | 13 | |
| **Features from self-report data** | | |
| expected grade | 1 | Numerical |
| largest program written before course | 1 | Ordinal: "None," "50 LoC," "500 LoC," "5000 LoC," "Longer," "Missing" (-1) |
| current study year | 1 | Numerical |
| how many times is attending the course | 1 | Numerical |
| how many other concurrent courses | 1 | Numerical |

## Appendix B: Feature Importances for the Trace Data Features

**Table 5.** List of all features collected from the course and used for training. The features are sorted by the median of feature importances (MD imp.) across the logistic regression models trained on yearly data.

| No. | Feature | MD imp. | σ | No. | Feature | MD imp. | σ |
|---|---|---|---|---|---|---|---|
| 1 | Σ total nr of exercises with no submissions | -0.031 | 0.044 | 100 | Demo 9: nr of exercises in 5 mins | 0.000 | 0.094 |
| 2 | Mean: total nr of exercises with no submissions | -0.031 | 0.044 | 101 | Demo 10: time of 1st submission before deadline | 0.000 | 0.096 |
| 3 | Demo 2: number of submissions after first correct submission | -0.022 | 0.072 | 102 | Demo 10: time between first and last submissions of a series | 0.000 | 0.099 |
| 4 | Demo 4: time of last submissions before deadline | -0.020 | 0.095 | 103 | Demo 9: number of correct submissions before deadline | 0.000 | 0.100 |
| 5 | Demo 6: total nr of exercises with no submissions | -0.017 | 0.053 | 104 | Demo 10: nr of exercises in 5 mins | 0.000 | 0.107 |
| 6 | Demo 3: nr of exercises in 5 mins | -0.014 | 0.091 | 105 | Demo 7: time between first submission and first correct submission | 0.000 | 0.108 |
| 7 | Demo 5: time of last submissions before deadline | -0.013 | 0.093 | 106 | Demo 5: time of 1st submission before deadline | 0.000 | 0.112 |
| 8 | Demo 2: time between first submission and first correct submission | -0.013 | 0.093 | 107 | Demo 9: total nr of exercises with no submissions | 0.000 | 0.113 |
| 9 | Demo 5: nr of submissions wrong | -0.011 | 0.057 | 108 | Demo 10: number of correct submissions before deadline | 0.000 | 0.114 |
| 10 | Demo 1: number of submissions before first correct | -0.011 | 0.088 | 109 | Demo 7: time of last submissions before deadline | 0.000 | 0.120 |
| 11 | Σ time between first submission and first correct submission | -0.010 | 0.072 | 110 | Demo 3: time of 1st submission before deadline | 0.000 | 0.149 |
| 12 | Mean: time between first submission and first correct submission | -0.010 | 0.072 | 111 | Demo 2: time of 1st submission before deadline | 0.001 | 0.080 |
| 13 | Demo 2: total nr of exercises with no submissions | -0.010 | 0.105 | 112 | Demo 7: total nr of submissions | 0.002 | 0.036 |
| 14 | Demo 3: total nr of exercises with no submissions | -0.009 | 0.067 | 113 | Demo 7: nr of exercises in 15 mins | 0.002 | 0.045 |
| 15 | Demo 4: total nr of exercises with no submissions | -0.009 | 0.151 | 114 | Demo 6: number of submissions after first correct submission | 0.002 | 0.063 |
| 16 | Demo 4: time of 1st submission before deadline | -0.008 | 0.077 | 115 | Demo 6: nr of exercises in 5 mins | 0.002 | 0.089 |
| 17 | Demo 5: total nr of exercises with no submissions | -0.008 | 0.089 | 116 | Demo 4: number of submissions after first correct submission | 0.002 | 0.103 |
| 18 | Demo 3: time of last submissions before deadline | -0.008 | 0.115 | 117 | Demo 7: number of correct submissions before deadline | 0.002 | 0.121 |
| 19 | Demo 1: total nr of submissions | -0.007 | 0.038 | 118 | Demo 5: number of submissions after first correct submission | 0.002 | 0.124 |
| 20 | Demo 4: nr of submissions wrong | -0.006 | 0.047 | 119 | Demo 6: total nr of submissions | 0.003 | 0.042 |
| 21 | Demo 5: nr of exercises in 5 mins | -0.006 | 0.061 | 120 | Demo 7: nr of exercises in 5 mins | 0.003 | 0.053 |
| 22 | Demo 3: number of correct submissions | -0.006 | 0.061 | 121 | Demo 1: total nr of exercises with no submissions | 0.003 | 0.079 |
| 23 | Demo 1: time between first submission and first correct submission | -0.005 | 0.092 | 122 | Demo 6: time of 1st submission before deadline | 0.003 | 0.096 |
| 24 | Demo 1: nr of submissions wrong | -0.005 | 0.110 | 123 | Demo 3: nr of exercises in 15 mins | 0.004 | 0.055 |
| 25 | Σ time of last submissions before deadline | -0.004 | 0.042 | 124 | Demo 6: number of submissions before first correct | 0.005 | 0.047 |
| 26 | Mean: time of last submissions before deadline | -0.004 | 0.042 | 125 | Demo 5: nr of exercises in 2 hours | 0.005 | 0.062 |
| 27 | Demo 8: total nr of exercises with no submissions | -0.004 | 0.093 | 126 | Demo 2: time of last submissions before deadline | 0.005 | 0.113 |
| 28 | Demo 8: time between first submission and first correct submission | -0.004 | 0.129 | 127 | Demo 7: number of submissions after first correct submission | 0.005 | 0.130 |
| 29 | Demo 5: total nr of submissions | -0.003 | 0.055 | 128 | Demo 4: time between first and last submissions of a series | 0.005 | 0.137 |
| 30 | Demo 1: time of last submissions before deadline | -0.003 | 0.090 | 129 | Demo 7: number of correct submissions | 0.006 | 0.081 |
| 31 | Demo 4: time between first submission and first correct submission | -0.003 | 0.115 | 130 | Demo 5: time between first and last submissions of a series | 0.006 | 0.089 |
| 32 | Demo 4: number of submissions before first correct | -0.002 | 0.042 | 131 | Demo 1: time of 1st submission before deadline | 0.006 | 0.098 |
| 33 | Demo 2: time between first and last submissions of a series | -0.002 | 0.054 | 132 | Σ nr of submissions wrong | 0.007 | 0.032 |
| 34 | Demo 8: time of last submissions before deadline | -0.002 | 0.074 | 133 | Mean: nr of submissions wrong | 0.007 | 0.032 |
| 35 | Demo 5: number of submissions before first correct | -0.002 | 0.078 | 134 | Σ time of 1st submission before deadline | 0.007 | 0.043 |
| 36 | Demo 2: number of correct submissions | -0.002 | 0.098 | 135 | Mean: time of 1st submission before deadline | 0.007 | 0.043 |
| 37 | Demo 6: nr of submissions wrong | -0.001 | 0.034 | 136 | Demo 1: number of submissions after first correct submission | 0.008 | 0.059 |
| 38 | Demo 4: total nr of submissions | -0.001 | 0.038 | 137 | Demo 1: time between first and last submissions of a series | 0.008 | 0.152 |
| 39 | Demo 5: time between first submission and first correct submission | -0.001 | 0.076 | 138 | Demo 2: total nr of submissions | 0.009 | 0.047 |
| 40 | Demo 8: time between first and last submissions of a series | -0.001 | 0.102 | 139 | Demo 2: nr of exercises in 5 mins | 0.009 | 0.070 |
| 41 | Demo 11: nr of exercises in 15 mins | 0.000 | 0.027 | 140 | Demo 4: nr of exercises in 5 mins | 0.009 | 0.116 |
| 42 | Demo 11: nr of exercises in 2 hours | 0.000 | 0.030 | 141 | Demo 2: nr of exercises in 15 mins | 0.010 | 0.066 |
| 43 | Demo 10: total nr of submissions | 0.000 | 0.031 | 142 | Mean: number of submissions after first correct submission | 0.011 | 0.045 |
| 44 | Demo 11: nr of exercises in 24 hours | 0.000 | 0.035 | 143 | Σ number of submissions after first correct submission | 0.011 | 0.045 |
| 45 | Demo 11: number of correct submissions before deadline | 0.000 | 0.038 | 144 | Demo 6: time between first and last submissions of a series | 0.011 | 0.099 |
| 46 | Demo 11: total nr of exercises with no submissions | 0.000 | 0.042 | 145 | Demo 1: nr of exercises in 5 mins | 0.011 | 0.103 |
| 47 | Demo 10: number of submissions after first correct submission | 0.000 | 0.042 | 146 | Demo 7: time of 1st submission before deadline | 0.011 | 0.172 |
| 48 | Demo 11: nr of exercises in 5 mins | 0.000 | 0.043 | 147 | Demo 5: nr of exercises in 24 hours | 0.012 | 0.058 |
| 49 | Demo 10: nr of submissions wrong | 0.000 | 0.043 | 148 | Demo 7: time between first and last submissions of a series | 0.012 | 0.164 |
| 50 | Demo 9: number of submissions after first correct submission | 0.000 | 0.046 | 149 | Demo 3: total nr of submissions | 0.013 | 0.039 |
| 51 | Demo 10: number of correct submissions | 0.000 | 0.046 | 150 | Demo 6: nr of exercises in 15 mins | 0.014 | 0.055 |
| 52 | Demo 8: total nr of submissions | 0.000 | 0.049 | 151 | Demo 5: number of correct submissions | 0.015 | 0.103 |
| 53 | Demo 8: number of submissions after first correct submission | 0.000 | 0.051 | 152 | Demo 3: nr of exercises in 2 hours | 0.017 | 0.071 |
| 54 | Demo 8: nr of submissions wrong | 0.000 | 0.051 | 153 | Σ total nr of submissions | 0.018 | 0.016 |
| 55 | Demo 8: number of submissions before first correct | 0.000 | 0.051 | 154 | Mean: total nr of submissions | 0.018 | 0.016 |
| 56 | Demo 11: number of submissions before first correct | 0.000 | 0.054 | 155 | Demo 3: nr of exercises in 24 hours | 0.018 | 0.053 |
| 57 | Demo 10: nr of exercises in 2 hours | 0.000 | 0.055 | 156 | Demo 2: nr of exercises in 24 hours | 0.018 | 0.066 |
| 58 | Demo 11: time of 1st submission before deadline | 0.000 | 0.055 | 157 | Demo 2: nr of exercises in 2 hours | 0.018 | 0.075 |
| 59 | Demo 7: nr of submissions wrong | 0.000 | 0.056 | 158 | Demo 2: nr of submissions wrong | 0.018 | 0.091 |
| 60 | Demo 10: number of submissions before first correct | 0.000 | 0.057 | 159 | Demo 3: time between first and last submissions of a series | 0.018 | 0.153 |
| 61 | Demo 8: nr of exercises in 15 mins | 0.000 | 0.057 | 160 | Mean: nr of exercises in 5 mins | 0.019 | 0.055 |
| 62 | Demo 3: number of submissions after first correct submission | 0.000 | 0.057 | 161 | Σ nr of exercises in 5 mins | 0.019 | 0.055 |
| 63 | Demo 7: nr of exercises in 24 hours | 0.000 | 0.058 | 162 | Demo 1: nr of exercises in 24 hours | 0.020 | 0.048 |
| 64 | Demo 10: time of last submissions before deadline | 0.000 | 0.059 | 163 | Demo 1: nr of exercises in 2 hours | 0.020 | 0.060 |
| 65 | Demo 9: total nr of submissions | 0.000 | 0.059 | 164 | Demo 4: number of correct submissions | 0.020 | 0.065 |
| 66 | Demo 11: nr of submissions wrong | 0.000 | 0.059 | 165 | Demo 1: number of correct submissions | 0.022 | 0.055 |
| 67 | Demo 11: time between first submission and first correct submission | 0.000 | 0.062 | 166 | Mean: number of submissions before first correct | 0.023 | 0.040 |
| 68 | Demo 8: nr of exercises in 5 mins | 0.000 | 0.062 | 167 | Σ number of submissions before first correct | 0.023 | 0.040 |
| 69 | Demo 11: total nr of submissions | 0.000 | 0.062 | 168 | Demo 4: nr of exercises in 15 mins | 0.025 | 0.068 |
| 70 | Demo 10: total nr of exercises with no submissions | 0.000 | 0.062 | 169 | Demo 2: number of submissions before first correct | 0.027 | 0.099 |
| 71 | Demo 8: number of correct submissions | 0.000 | 0.066 | 170 | Mean: number of correct submissions | 0.029 | 0.052 |
| 72 | Demo 11: number of correct submissions | 0.000 | 0.066 | 171 | Σ number of correct submissions | 0.029 | 0.052 |
| 73 | Demo 11: time between first and last submissions of a series | 0.000 | 0.066 | 172 | Demo 5: number of correct submissions before deadline | 0.029 | 0.092 |
| 74 | Demo 8: nr of exercises in 2 hours | 0.000 | 0.066 | 173 | Demo 1: nr of exercises in 15 mins | 0.029 | 0.097 |
| 75 | Demo 9: nr of submissions wrong | 0.000 | 0.067 | 174 | Σ time between first and last submissions of a series | 0.030 | 0.053 |
| 76 | Demo 9: nr of exercises in 2 hours | 0.000 | 0.067 | 175 | Mean: time between first and last submissions of a series | 0.030 | 0.053 |
| 77 | Demo 10: nr of exercises in 24 hours | 0.000 | 0.068 | 176 | Demo 3: number of submissions before first correct | 0.031 | 0.043 |
| 78 | Demo 8: time of 1st submission before deadline | 0.000 | 0.068 | 177 | Demo 2: number of correct submissions before deadline | 0.032 | 0.091 |
| 79 | Demo 7: nr of exercises in 2 hours | 0.000 | 0.069 | 178 | Demo 3: nr of submissions wrong | 0.033 | 0.052 |
| 80 | Demo 9: time between first and last submissions of a series | 0.000 | 0.069 | 179 | Demo 6: number of correct submissions | 0.033 | 0.081 |
| 81 | Demo 9: number of correct submissions | 0.000 | 0.071 | 180 | Demo 6: nr of exercises in 24 hours | 0.034 | 0.062 |
| 82 | Demo 10: time between first submission and first correct submission | 0.000 | 0.073 | 181 | Σ nr of exercises in 15 mins | 0.035 | 0.026 |
| 83 | Demo 6: time of last submissions before deadline | 0.000 | 0.073 | 182 | Mean: nr of exercises in 15 mins | 0.035 | 0.026 |
| 84 | Demo 3: time between first submission and first correct submission | 0.000 | 0.074 | 183 | Demo 6: number of correct submissions before deadline | 0.035 | 0.065 |
| 85 | Demo 5: nr of exercises in 15 mins | 0.000 | 0.074 | 184 | Demo 6: nr of exercises in 2 hours | 0.036 | 0.057 |
| 86 | Demo 7: number of submissions before first correct | 0.000 | 0.075 | 185 | Demo 4: nr of exercises in 2 hours | 0.037 | 0.114 |
| 87 | Demo 10: nr of exercises in 15 mins | 0.000 | 0.076 | 186 | Demo 3: number of correct submissions before deadline | 0.039 | 0.107 |
| 88 | Demo 6: time between first submission and first correct submission | 0.000 | 0.076 | 187 | Demo 1: number of correct submissions before deadline | 0.044 | 0.172 |
| 89 | Demo 9: nr of exercises in 24 hours | 0.000 | 0.077 | 188 | Σ nr of exercises in 2 hours | 0.047 | 0.027 |
| 90 | Demo 9: number of submissions before first correct | 0.000 | 0.078 | 189 | Mean: nr of exercises in 2 hours | 0.047 | 0.027 |
| 91 | Demo 9: time of last submissions before deadline | 0.000 | 0.078 | 190 | Demo 4: number of correct submissions before deadline | 0.048 | 0.125 |
| 92 | Demo 11: time of last submissions before deadline | 0.000 | 0.079 | 191 | Σ nr of exercises in 24 hours | 0.051 | 0.045 |
| 93 | Demo 7: total nr of exercises with no submissions | 0.000 | 0.082 | 192 | Mean: nr of exercises in 24 hours | 0.051 | 0.045 |
| 94 | Demo 8: nr of exercises in 24 hours | 0.000 | 0.084 | 193 | Demo 4: nr of exercises in 24 hours | 0.056 | 0.129 |
| 95 | Demo 8: number of correct submissions before deadline | 0.000 | 0.085 | 194 | Mean: number of correct submissions before deadline | 0.069 | 0.036 |
| 96 | Demo 11: number of submissions after first correct submission | 0.000 | 0.086 | 195 | Σ number of correct submissions before deadline | 0.069 | 0.036 |
| 97 | Demo 9: time between first submission and first correct submission | 0.000 | 0.088 | | | | |
| 98 | Demo 9: nr of exercises in 15 mins | 0.000 | 0.090 | | | | |
| 99 | Demo 9: time of 1st submission before deadline | 0.000 | 0.092 | | | | |