G is for Generalisation: Predicting Student Success from Keystrokes

Zac Pullar-Strecker
zpul156@aucklanduni.ac.nz
University of Auckland
Auckland, New Zealand

Filipe Dwan Pereira
filipe.dwan@ufrr.br
Federal University of Roraima
Boa Vista, Brazil

Paul Denny
paul@cs.auckland.ac.nz
University of Auckland
Auckland, New Zealand

Andrew Luxton-Reilly
andrew@cs.auckland.ac.nz
University of Auckland
Auckland, New Zealand

Juho Leinonen
juho.2.leinonen@aalto.fi
Aalto University
Espoo, Finland

ABSTRACT

Student performance prediction aims to build models to help educators identify struggling students so they can be better supported. However, prior work in the space frequently evaluates features and models on data collected from a single semester, of a single course, taught at a single university. Without evaluating these methods in a broader context there is an open question of whether or not performance prediction methods are capable of generalising to new data. We test three methods for evaluating student performance models on data from introductory programming courses from two universities with a total of 3, 323 students. Our results suggest that using cross-validation on one semester is insufficient for gauging model performance in the real world. Instead, we suggest that where possible future work in student performance prediction collects data from multiple semesters and uses one or more as a distinct hold-out set. Failing this, bootstrapped cross-validation should be used to improve confidence in models' performance. By recommending stronger methods for evaluating performance prediction models, we hope to bring them closer to practical use and assist teachers to understand struggling students in novice programming courses.

CCS CONCEPTS

• **Applied computing** → **Education**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

computing education, predicting performance, programming process data, educational data mining, learning analytics

ACM Reference Format:
Zac Pullar-Strecker, Filipe Dwan Pereira, Paul Denny, Andrew Luxton-Reilly, and Juho Leinonen. 2023. G is for Generalisation: Predicting Student Success from Keystrokes. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023), March 15–18, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3545945.3569824

SIGCSE 2023, March 15–18, 2023, Toronto, ON, Canada.

ACM ISBN 978-1-4503-9431-4/23/03…$15.00
https://doi.org/10.1145/3545945.3569824

1 INTRODUCTION

Computer science, and programming in particular, is commonly regarded as a difficult subject to learn [21]. While Simon et al. [22] found little evidence that introductory programming courses had a higher failure rate than other STEM courses, they note that there is clearly room for improvement and that future work improving our understanding of computing education continues to be a valuable direction. One such approach is to build models to predict students' chance at success; these models can then be used to target remedial action such as assigning extra practise on a particular topic [2].

For example, Leinonen et al. [16] used the time students started working on an assignment and the time they submitted to predict their final grade on the course. In general, prior work has used a wide variety of data sources as features, such as integrated development environment (IDE) events, commit logs, Q&A interaction, survey data, past experience, and early assignment grades [4]. In some cases authors only report correlations between these features and performance measures, while others train models such as linear regression, random forests (RF), and Bayesian models [2, 17]. This variety continues in the choice of performance metrics, prior work has attempted to predict categorical and continuous final grades, exam grades, and whether a student will pass or fail [10].

Despite the large body of prior work, few papers have attempted to generalise their findings. Most studies use data collected from a single semester, of a single course, taught at a single university [10]. In these cases, models are commonly evaluated using cross-validation (CV) where a single sample is divided into a series of train and test sets [16]. While CV is a popular method for evaluating machine learning models, it cannot account for the fact that students, and the courses themselves, change between semesters. If a model cannot learn general relationships in the data which persist across these changes, they will not perform as well in real courses as they did while the model was developed. Hence, at minimum a model must be able to generalise to a new semester. There is also the question of whether the modelling approach is generalisable, i.e., whether the features and architecture can be used to train a model that works on a different course or institution.

To evaluate to what extent student performance models and approaches are capable of generalising we train and evaluate them

in three increasingly dissimilar settings using three datasets from two institutions. We investigate three research questions (RQs):

RQ1. Do features and models for student performance prediction appear predictive when tested on a single semester of data?

RQ2. To what extent do models retain their performance when evaluated on a similar course at the same institution?

RQ3. To what extent can models and approaches generalise between institutions?

In Section 2 we describe related work, including proposed features, models, and evaluation methodologies. Next, Section 3 describes our datasets and the context in which they were collected. Then, Section 4 describes our methodology. Section 5 discusses our results, and Section 6 presents our discussion. Finally, Section 7 discusses limitations and Section 8 concludes.

2 RELATED WORK

Prior work on student performance prediction is extremely varied, with studies using different features, models, performance measures, and evaluation methodologies. In this section we summarise the most closely related work. For a more detailed overview of student performance prediction we refer to the 2018 systematic literature review by Hellas et al. [10].

2.1 Features

Time management related features have been popular in prior work as features for predicting student performance. The exact features, and the source of data from which they are collected, varies by study. Edwards et al. used data from an assignment submission platform and found that students whose first and last submissions were early relative to other students performed better than others on average [9]. More recently, other studies have used online IDE events to find the time from the first code edit to the deadline (named *procrastination*); Pereira et al. [19] found procrastination had a small but significant correlation with performance. Leinonen et al. [15] replicated earlier work suggesting that students who start early tend to perform better (Pearson $r = 0.23$, $p = 0.0003$, data from 1st week). Leinonen et al. [16] also explored different representations for the amount of time students spent working. Specifically, they compared *coarse time on task*, the time between the first keystroke and the first submission, and *fine time on task*, the sum of latencies between keystrokes until the first submission with all breaks of longer than 10 minutes removed. They found that fine time on task had stronger correlations with student performance (Pearson $r = 0.51$, $p = 3.7 \times 10^{-10}$ against $r = 0.23$, $p = 0.007$, from whole course).

One of the simplest categories of features used in prior work is counts (or rates) of events. Specifically, the number of submissions, text pastes, and the total number of events have been proposed [7, 19]. Despite their simplicity, prior work has found that they are useful for predicting student performance [19].

A few studies have explored the ability of keystroke latency, or equivalently typing speed, to predict student performance. Leinonen et al. found that the latency distribution of particular digraphs differed between experienced and novice programmers [18]. Leinonen et al. also investigated correlations between typing speed and other process features [17]. Edwards et al. [8] found a weak (Spearman $r = -0.20$) but statistically significant ($p = 0.001$) correlation between latencies and exam score in a Python course, but no significant correlation in a Java course.

A number of other features have been developed that utilise the keystrokes themselves [5, 11, 24]. However, we are primarily focused on exploring event count and time management features in this work.

2.2 Models

Prior work in student performance prediction has utilised a variety of machine learning models. As many papers work with tabular features generated from events, linear regression, Bayesian models, decision trees, and RF are among the most popular models [10]. Of these, RF models tend to perform as good as or better than alternatives [1, 13]. However, as neural networks have become more popular recent work has investigated whether recurrent neural networks, such as long-short term memory models, can be applied to the events themselves [3]. These early results are promising, but have yet to demonstrate success in multiple contexts.

2.3 Evaluation

Student performance prediction models have been used to predict a number of different targets. The most popular targets are predicting a categorical final grade (e.g., A/B/C), predicting a continuous final grade (e.g., 50%), and predicting if a student will pass or fail. A distinct, but closely related task is predicting whether or not a student will drop out of a course.

Evaluations of student performance prediction also vary by how much data they include. Some studies use data from the entirety of the course to predict final exam scores, while others use data from just the first assignment [8, 19]. While there may be some benefit to being able to predict student performance as late as half way through the course, models which can accurately predict student performance from only the first assessment will have the biggest impact. This scenario, termed *early student performance prediction* is substantially more difficult than predicting from all data on a course, and prior works have reported correlations from Pearson $r = 0.23$ as positive results [15].

A substantial proportion of prior work in student performance prediction evaluated features and models on only a single semester of data from a single course taught at a single university [10]. However, some papers have investigated the behaviour of methods across multiple contexts. Castro-Wunsch et al. [6] trained models on submission counts and test correctness to predict whether students would pass or not. This work is of particular interest as they compared the performance they obtained when training and evaluating on a single semester using repeated train and test splits with the performance from using a separate semester as a hold out set. They found that there was only a minimal drop in accuracy between these evaluations, a couple of percent for most of the models they tested. More recently, Pereira et al. [19] explored deep learning methods on a number of semesters of a CS1 course. In total they used data from 2,058 students. To evaluate their models Pereira et al. used stratified 10-fold CV for classification, while for regression they randomly divided the dataset into a train (70%) and

Table 1: The datasets used in our analysis.

	MOOC	LU	CodeBench
Participants	449	291	2, 989
Semesters	1	1	10
Classes	1	1	71
Events	3, 733, 291	1, 986, 501	3, 086, 272

a test (30%) set. The larger number of participants in this study than in prior work provides increased confidence in their evaluation. However, the authors did not investigate the performance of their models when training on a subset of semesters and using the rest as a hold out set, or on similar courses taught at other universities.

Prior machine learning (ML) literature has evaluated trade-offs between evaluation strategies like repeated cross-validation and bootstrapping [12]. However these works explored synthetic experiments which cannot capture effects such as concept drift between semesters, or the extent to which feature distributions differ between contexts.

3 CONTEXT

In this study we evaluate student performance prediction methods on three datasets from two universities. A summary of each dataset is given in Table 1. The difference in the number of events per student is not due to differences in the definitions of the events themselves, and instead is likely due to pedagogical differences.

The first two datasets were collected from variants of an introductory programming course (CS1) taught at a research-oriented university in Finland. The first dataset comes from a massive open online course (MOOC), while the second comes from a local university (LU) version of the course. These datasets are similar to the one described by Leinonen [14]. Both versions were taught in Finnish using the Java programming language and had the primary goal of teaching students object-oriented programming. Students attending the local university version of the course were primarily majoring in computer science, and may have been more motivated than those from the MOOC version of the course.

The final grade for the MOOC version of the course was between 0 and 5 (we rescale to 0 to 1), with 55% of the final grade from coursework and 45% of the final grade from exams. Each of the assignments were worth 8% of the final grade, except the first which was 7%. The exams were worth 10%, 15%, and 20% of the final grade. Additionally, students needed to get at least 50% in the final exam to pass. In the LU version 70% of the final grade came from assignments and 30% from two exams. The assignments were 10% each, and students got the whole 10% if they got more than 90% in that assignment. The exams were worth 10%, and 20% of the final grade respectively.

Students taking this course worked on exercises using an IDE with the TestMyCode plugin [23]. The plugin allows students to download and submit exercises directly in the IDE, and additionally records text changes (insert, remove, paste), focus events (gained, lost), and when students execute their program, or submit an exercise. As we focus on the early student performance prediction case we only include data from the first assignment in the course.

The second dataset we explore is the open CodeBench dataset[1]. CodeBench is a programming online judge developed by UFAM. This data was collected from the Federal University of the Amazonas (UFAM) over 10 semesters covering the period 2016–2021 (excluding 2020 because of COVID-19). At UFAM CS1 is compulsory to 16 degrees outside of computing, and so the dataset contains a number of classes in each semester. To illustrate, in the first semester of 2016 (2016.1), UFAM offered 10 different classes of CS1 for the non-CS students. In each class there were students from one or two different undergraduate courses. In the dataset, they identify the classes using different ids (e.g, 2016.1.102 ... 2016.1.111). Slides, evaluation systems, programming language and pedagogical methodology used to teach the CS1 courses in the Codebench dataset were the same for all classes. In all cases, the classes were taught using Python and covered similar content to the MOOC and LU datasets. Students could submit their source code for automatic correction as many times as they want, without penalty. The code questions were asked in two modalities: list of exercises and face-to-face exams. In both cases, the students used the CodeBench IDE to solve the code problems. CodeBench records the final code, execution snapshots, key presses, and student's interaction with the IDE.

The final grade was calculated based on seven partial exams, seven assignments, and one final exam. The grades from the partial exams had increasing weights (6.1% to 18.2%) towards the final grade. The grades from all assignments had the same weight on the final grade (1.3%). In total, the dataset contains data from 2, 989 students and after selecting data from only the first assignment in each class, a total of 3, 086, 272 events.

4 METHODOLOGY

To enable reproducibility we detail our methodology in this section. All analysis was performed in Python and is available on GitHub[2]. The CodeBench dataset is public, however the MOOC and LU datasets are not.

The CodeBench dataset contains students who did not continue to the end of the course and as such did not receive a final grade, these students we remove. However, we include students who received explicit zeros in the course by sitting but failing required assessments. After this filtering 2, 583 students from this dataset were included in our analysis.

We compute summary features from the events collected by instrumented IDEs in a similar manner to prior work. The first four features we compute are simply counts of each type of event. These are inspired by the counts of paste, run, and submit events which have been used in prior work, but extended to all of the events we have available [7, 19]. The MOOC dataset additionally contains focus gained, focus lost, and text paste events which are not present in the LU dataset. We include these events for computing time management features but do not use their counts as features.

Next, we calculate five time management based features. These are the time to the first event (when the student starts working on the exercise), the time to the last event, the difference between these, and coarse and fine time on task [16]. We make one alteration to the fine time on task proposed by Leinonen et al., instead of removing

[1] https://codebench.icomp.ufam.edu.br/dataset/
[2] github.com/zacps/g-is-for-generalisation

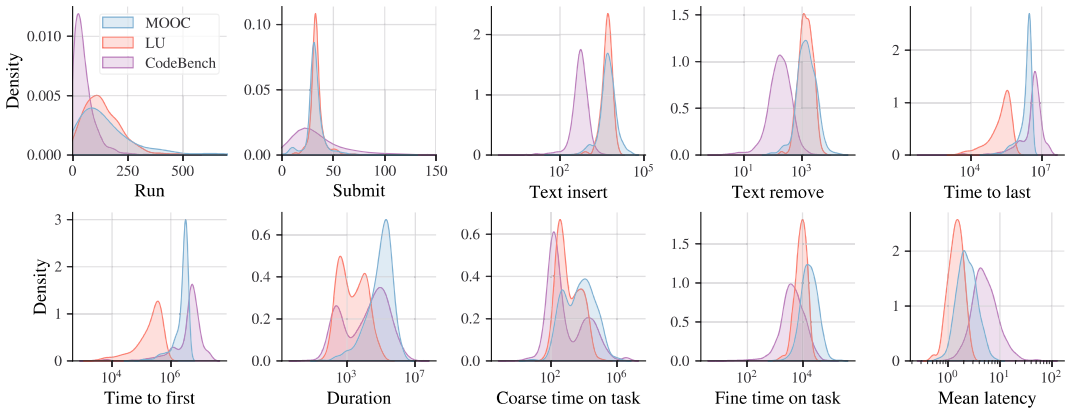

Figure 1: KDE feature distributions of our datasets. Time features are measured in seconds.

Table 2: The features used in our analysis.

Name	Description
Run	the program was run
Submit	the exercise was submitted
Text insert	text was inserted
Text remove	text was removed
Time to last	time from the last event to the deadline
Time to first	time from the first event to the deadline
Duration	duration between first and last events
Coarse time on task	as introduced by [16]
Fine time on task	as introduced by [16]
Mean latency	the average latency between events

latencies longer than 10 minutes we remove latencies longer than the 99.5th percentile. This change was made to be more robust to differences between contexts. Finally, we compute the mean latency between events as our last feature. The features we use are summarised in Table 2.

We average the time-based features over each exercise in order to stop the model from biasing towards particular exercises. While including each feature for each assignment individually may produce better performing models, it also limits the ability of the model to generalise if exercises change between semesters of a course, or are entirely different in a course taken at a different university. Alternative approaches, like producing predictions per exercise and averaging them, or utilising regularisation, are potential avenues for future work.

Figure 1 shows the kernel distribution estimate (KDE) of our features on the datasets we use. Most of the features we use have similar distributions between the LU and MOOC datasets, with the exception of time to the first and last events. We suspect this is because the MOOC course had a longer period between the release and deadline of each assignment than the LU course.

To select the model used in our analysis we trained linear regression, a RF, and a multi-layer perceptron on the MOOC dataset. We found that the RF provided the best results ($r^2 = 0.24$), with linear regression placing second ($r^2 = 0.15$). Because of this, we use RF for the rest of our analysis. Hyper-parameters were left at their default values.

In all of our evaluations we use the coefficient of determination r^2, and its adjusted variant, as our performance metrics. For all statistical tests, we use $\alpha = 0.05$. Spearman's correlation coefficient is used for feature correlations as some are not normally distributed. Wherever multiple significance tests are performed, the Bonferroni correction is used to correct for multiple comparisons.

To address RQ1, we use 5-fold CV. In this method the data is split into five parts and in each of five rounds a classifier is trained on all but one part and evaluated on the remainder. This is one of the most popular methods for evaluating ML models [20].

While CV tends to provide strong performance estimates it does not provide a variance estimate. To address this, we additionally present results from bootstrapping 5-fold CV 1,000 times. In bootstrapping, the dataset is re-sampled with replacement a large number of times and (in our case) 5-fold CV is performed on each of the re-sampled datasets. By bootstrapping CV we gain an insight into the chance that we would obtain a similar result if we performed the same experiment again (though this cannot account for data drift or similar effects).

Lastly, to answer RQ2 and RQ3 we evaluate models using hold-out sets from other semesters or datasets. In this case the model is trained once on a fixed set of data, and evaluated on a disjoint set. This method more closely simulates how a model would be used in the real world as it accounts for changes in the data that could occur between the model's training and use. As such, we hypothesise that this method may produce more realistic evaluations of student performance models than CV.

5 RESULTS

Our first RQ aims to determine whether we can find correlated features and build a predictive model when evaluating on a single semester of data. We first present pairwise Spearman correlations from the MOOC dataset in Figure 2. We can see that some of our summary features produce significant correlations with student's final grade. Specifically, submit, time to last, time to first, and mean latency have significant correlations with student's grades.

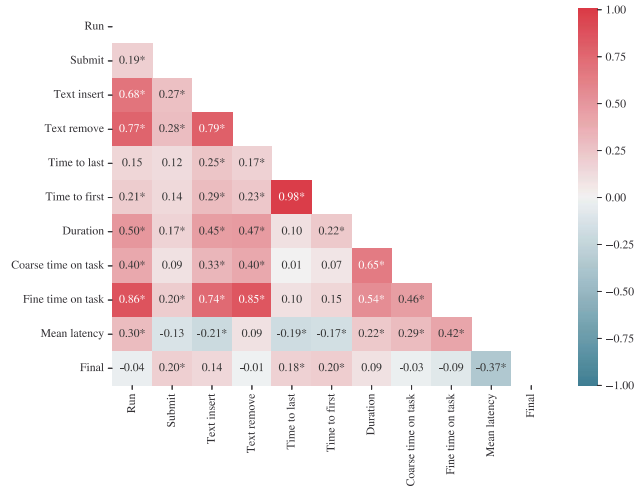

Figure 2: Pairwise Spearman correlation between features and performance metrics on the MOOC dataset. Asterisks denote significance ($p < 0.05$).

To answer the second part of our RQ we train a RF model on the MOOC dataset and evaluate it using 5-fold CV. In order to estimate how reliable this evaluation strategy is, the results from the CV are subsequently bootstrapped using 1,000 repetitions. A single cross-validated evaluation gave an r^2 of 0.24 and adjusted r^2 of 0.15. The bootstrapped distribution had a mean adjusted r^2 value of $\mu = 0.08$ and standard deviation $\sigma = 0.07$, with a 95% confidence interval (CI) of 0.00 to 0.22. The unadjusted r^2 distribution had $\mu = 0.19$, $\sigma = 0.06$ with a CI of 0.08 to 0.31.

Our second RQ seeks to find the difference in a models' estimated performance between using a similar course as a hold-out set and using CV on a single semester of data. We train a RF model on the MOOC dataset, and evaluate its performance on the LU dataset. While the CV evaluation obtained a mean r^2 of 0.24 (adjusted $r^2 = 0.15$), the separate hold out evaluation produced a mean r^2 of 0.12 (adjusted $r^2 = 0.09$), half of the CV result.

Our third RQ considers whether prior models and approaches for student performance prediction are capable of generalising to new contexts. First, we explore whether the features which seemed promising on the MOOC and LU datasets retain any predictive power on the CodeBench dataset. Fig. 3 presents Spearman correlations between our features and students' final grade over each class in the dataset. As we're interested in understanding the conclusions a researcher might draw from seeing a single column of this dataset we apply the Bonferroni correction across each column, but not each row. We find that 29% of the classes have at least one significant correlation, with some of them having correlation coefficients as high as 0.62. When considering all of the data five of the features have significant correlations, though none of these have coefficients greater than $|r| = 0.10$.

Next, we trained a RF model using the same features on the CodeBench dataset, using both 5-fold CV and a hold out set to evaluate performance. For the hold out set we used the classes from 2021 as our test set, and the rest as training data. When using CV we obtained an r^2 of 0.10 (adjusted $r^2 = 0.08$), while the hold-out set evaluation found an r^2 of -4.0 (adjusted $r^2 = -4.2$). The negative r^2 indicates that the model is worse than a model which predicts the mean grade for all students.

6 DISCUSSION

Our first RQ sought to replicate positive results from proposed models. As we discussed in Sec. 2, prior works have reported correlations as small as $|r| = 0.2$ and r^2 as small as 0.23. We found that, using CV, we were able to replicate similar results to prior work in the early student performance prediction setting.

To better understand the properties of the cross-validated estimate we performed bootstrapped 5-fold CV and recorded the distribution of r^2. We found that the cross-validated estimate had a very high variance. A researcher performing a single CV would interpret the lower bound of the confidence interval (0.00) as a clear negative result, while the high bound (0.22) could be interpreted as a positive result. This indicates that CV on a single semester may be insufficient to find a robust measure of the quality of student performance models. While this could be resolved by collecting additional data, this is not always feasible. As an alternative, bootstrapped CV may provide a better understanding of the model's performance when additional data cannot be collected.

Our second RQ aims to determine how well the model generalises to a similar course at the same institution. We found that when our model was trained on the MOOC dataset and evaluated using the LU dataset as a hold-out set the performance dropped substantially ($r^2 = 0.24$ to $r^2 = 0.12$). This indicates that the model had limited ability to generalise to the other variant of the course, even though the variants were similar.

This raises an important question about how far student performance models should be expected to generalise. An ideal model would be suited for any programming course at any university. However, this is unrealistic given the current state of student performance prediction. On the other hand, a model which required all assessments to stay identical is likely not general enough to be useful as course materials, especially in computer science, are frequently updated to stay relevant. Future work in student performance prediction should discuss the extent to which proposed models are intended to generalise, and use appropriate evaluation methodologies to demonstrate that this is achieved.

To answer our third RQ we investigated if the model and approach that showed promising results on a single semester could generalise to another institution. We found that when we evaluated Spearman's correlation coefficient between each feature and students' final grade on the CodeBench dataset across each semester that there was a high degree of variability. Some of the features we tested had correlations as high as 0.67 in one class, and as low as -0.31 in another. In the case of the time to first event feature 20% of the classes had significant correlations with the final grade, yet in the overall data there was no correlation ($r = 0.02$, $p = 1.00$). This indicates that small correlations from a single semester of data are insufficient for determining whether or not a feature could be useful for student performance prediction.

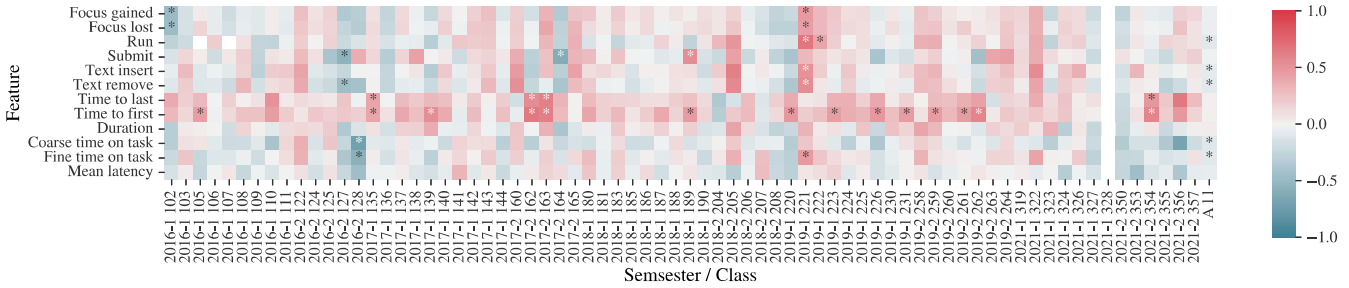

Figure 3: Feature correlations to final grade by class. Empty cells are missing features, asterisks denote significance ($p < 0.05$).

We found that the model and features which seemed promising when evaluated using CV were unable to generalise to a new institution. When trained and evaluated using a hold-out set the model was worse than a model which predicted the mean grade for all students. We initially planned to train a model on the CodeBench dataset and attempt to apply it to other datasets to measure generalisation performance. However, given the poor performance of the model trained and evaluated on the CodeBench dataset it is clear it would not have the ability to generalise to new contexts.

To provide reliable estimates of student performance models we recommend that researchers make use of the open CodeBench dataset, which provides an excellent benchmark on which to compare results. Additionally, we echo the recommendations from prior reviews which suggest that wherever possible papers utilising their own datasets release their data [10]. This way we can build a better understanding of the generalisation performance of proposed methods and step closer to reliable use of performance prediction in the real world.

7 LIMITATIONS

One limitation of our analysis is the relatively small correlations and r^2 values we observed when evaluating features and models on the MOOC dataset. It might be that more sophisticated features or models are capable of obtaining significantly stronger effects which may make more complex evaluation methodologies unnecessary. However, as similar effect sizes have been reported in prior work this is still an issue which needs to be addressed.

Similarly, coarser, non-keystroke based features, may be less affected by differences in assignments or teaching methodologies and hence generalise better to other contexts.

We do not specifically explore the extent to which performance degrades when evaluating models on subsequent semesters of the same course. Performing this comparison for a variety of institutions and proposed models would provide valuable insight on how long relationships in keystroke data remain relevant.

8 CONCLUSION

There is a considerable body of prior work in student performance prediction and these studies have experimented with a wide variety of features, models, and performance measures. Yet, the majority of this work evaluates proposed features and models on data collected from a single semester of a single course taught at a single university [10]. This raises the question of whether or not these features and models are capable of generalising to new contexts.

We compared evaluation strategies for performance prediction methods in order to determine to what extent prior methods can generalise, and whether or not simple approaches like CV are sufficient to obtain accurate performance estimates. To summarise our results, we present the answers to our three RQs:

RQ1. *Do features and models for student performance prediction appear predictive when tested on a single semester of data?*

Yes, we found similar feature correlations and model performance to those found in prior work when evaluating features and models on a single semester of data using CV. Yet, we found that the variance of the cross-validated performance estimate is high enough to produce both positive and negative conclusions from the same data.

RQ2. *To what extent do models retain their performance when evaluated on a similar course at the same institution?*

We found that when using a similar course from the same institution as a hold-out set, compared to CV, the r^2 dropped from 0.24 to 0.12, and the adjusted r^2 from 0.14 to 0.09. While the model did not loose all predictive power, this substantial drop indicates the model may not have captured generalisable relationships in the data.

RQ3. *To what extent can models and approaches generalise between institutions?*

We found that the modelling approach we tested was not able to generalise to a new institution, despite appearing promising when evaluated with CV. Additionally, we found that feature correlations on a single semester of data often appear predictive, yet do not have strong correlations on larger samples.

Our results suggest that evaluating performance prediction models on data collected from a single semester of a single course is insufficient to robustly estimate model performance and generalisation. While developing models in a single context remains a useful first step, we recommend that authors evaluate models on multiple semesters, and, if intended to generalise, on multiple contexts.

To support the effort to generalise student performance prediction models future studies should publish their datasets. In cases where evaluating on more than a single semester is infeasible, bootstrapped CV may provide greater confidence in results.

By recommending stronger methods for evaluating performance prediction models, we hope to bring them closer to practical use and assist teachers to identify struggling students in novice programming courses.

REFERENCES

[1] Alireza Ahadi, Raymond Lister, Heikki Haapala, and Arto Vihavainen. 2015. Exploring Machine Learning Methods to Automatically Identify Students in Need of Assistance. In *Proceedings of the eleventh annual International Conference on International Computing Education Research (ICER '15)*. Association for Computing Machinery, New York, NY, USA, 121–130. https://doi.org/10.1145/2787622.2787717

[2] Balqis Albreiki, Nazar Zaki, and Hany Alashwal. 2021. A Systematic Literature Review of Student' Performance Prediction Using Machine Learning Techniques. *Education Sciences* 11, 9 (Sept. 2021), 552. https://doi.org/10.3390/educsci11090552 Number: 9 Publisher: Multidisciplinary Digital Publishing Institute.

[3] Kai Arakawa, Qiang Hao, Wesley Deneke, Indie Cowan, Steven Wolfman, and Abigayle Peterson. 2022. Early Identification of Student Struggles at the Topic Level Using Context-Agnostic Features. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2022)*. Association for Computing Machinery, New York, NY, USA, 147–153. https://doi.org/10.1145/3478431.3499298

[4] Adam Carter, Christopher Hundhausen, and Daniel Olivares. 2019. Leveraging the Integrated Development Environment for Learning Analytics. In *The Cambridge Handbook of Computing Education Research*, Anthony V. Robins and Sally A. Fincher (Eds.). Cambridge University Press, Cambridge, 679–706. https://doi.org/10.1017/9781108654555.024

[5] Adam S. Carter, Christopher D. Hundhausen, and Olusola Adesope. 2015. The Normalized Programming State Model: Predicting Student Performance in Computing Courses Based on Programming Behavior. In *Proceedings of the eleventh annual International Conference on International Computing Education Research*. ACM, Omaha Nebraska USA, 141–150. https://doi.org/10.1145/2787622.2787710

[6] Karo Castro-Wunsch, Alireza Ahadi, and Andrew Petersen. 2017. Evaluating Neural Networks as a Method for Identifying Students in Need of Assistance. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 111–116. https://doi.org/10.1145/3017680.3017792

[7] John Edwards, Joseph Ditton, Bishal Sainju, and Joshua Dawson. 2020. Different assignments as different contexts: predictors across assignments and outcome measures in CS1. In *2020 Intermountain Engineering, Technology and Computing (IETC)*. IEEE, Williamsburg, VA, USA, 1–6. https://doi.org/10.1109/IETC47856.2020.9249217

[8] John Edwards, Juho Leinonen, and Arto Hellas. 2020. A Study of Keystroke Data in Two Contexts: Written Language and Programming Language Influence Predictability of Learning Outcomes. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. Association for Computing Machinery, New York, NY, USA, 413–419. http://doi.org/10.1145/3328778.3366863

[9] Stephen H. Edwards, Jason Snyder, Manuel A. Pérez-Quiñones, Anthony Allevato, Dongkwan Kim, and Betsy Tretola. 2009. Comparing effective and ineffective behaviors of student programmers. In *Proceedings of the fifth international workshop on Computing education research workshop (ICER '09)*. Association for Computing Machinery, New York, NY, USA, 3–14. https://doi.org/10.1145/1584322.1584325

[10] Arto Hellas, Petri Ihantola, Andrew Petersen, Vangel V. Ajanovski, Mirela Gutica, Timo Hynninen, Antti Knutas, Juho Leinonen, Chris Messom, and Soohyun Nam Liao. 2018. Predicting academic performance: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018 Companion)*. Association for Computing Machinery, New York, NY, USA, 175–199. https://doi.org/10.1145/3293881.3295783

[11] Matthew C. Jadud. 2006. Methods and tools for exploring novice compilation behaviour. In *Proceedings of the second international workshop on Computing education research (ICER '06)*. Association for Computing Machinery, New York, NY, USA, 73–84. https://doi.org/10.1145/1151588.1151600

[12] Ji-Hyun Kim. 2009. Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Computational Statistics & Data Analysis* 53, 11 (Sept. 2009), 3735–3745. https://doi.org/10.1016/j.csda.2009.04.009

[13] Charles Koutcheme, Sami Sarsa, Arto Hellas, Lassi Haaranen, and Juho Leinonen. 2022. Methodological Considerations for Predicting At-risk Students. In *Australasian Computing Education Conference (ACE '22)*. Association for Computing Machinery, New York, NY, USA, 105–113. https://doi.org/10.1145/3511861.3511873

[14] Juho Leinonen. 2022. Open IDE Action Log Dataset from a CS1 MOOC. In *Proceedings of the 6th Educational Data Mining in Computer Science Education (CSEDM) Workshop*. Zenodo, Virtual, 4 pages. https://doi.org/10.5281/zenodo.6983459

[15] Juho Leinonen, Francisco Enrique Vicente Castro, and Arto Hellas. 2021. Does the Early Bird Catch the Worm? Earliness of Students' Work and its Relationship with Course Outcomes. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*. Association for Computing Machinery, New York, NY, USA, 373–379. http://doi.org/10.1145/3430665.3456383

[16] Juho Leinonen, Francisco Enrique Vicente Castro, and Arto Hellas. 2022. Time-on-Task Metrics for Predicting Performance. In *Proceedings of the 53rd ACM Technical Symposium V.1 on Computer Science Education (SIGCSE 2022)*. Association for Computing Machinery, New York, NY, USA, 871–877. https://doi.org/10.1145/3478431.3499359

[17] Juho Leinonen, Leo Leppänen, Petri Ihantola, and Arto Hellas. 2017. Comparison of Time Metrics in Programming. In *Proceedings of the 2017 ACM Conference on International Computing Education Research (ICER '17)*. Association for Computing Machinery, New York, NY, USA, 200–208. https://doi.org/10.1145/3105726.3106181

[18] Juho Leinonen, Krista Longi, Arto Klami, and Arto Vihavainen. 2016. Automatic Inference of Programming Performance and Experience from Typing Patterns. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. Association for Computing Machinery, New York, NY, USA, 132–137. https://doi.org/10.1145/2839509.2844612

[19] Filipe Dwan Pereira, Samuel C. Fonseca, Elaine H. T. Oliveira, David B. F. Oliveira, Alexandra I. Cristea, and Leandro S. G. Carvalho. 2020. Deep learning for early performance prediction of introductory programming students: a comparative and explanatory study. *Revista Brasileira de Informática na Educação* 28, 0 (Oct. 2020), 723–748. https://doi.org/10.5753/rbie.2020.28.0.723 Number: 0.

[20] Sebastian Raschka. 2018. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. https://doi.org/10.48550/ARXIV.1811.12808

[21] Anthony V. Robins. 2019. Novice Programmers and Introductory Programming. In *The Cambridge Handbook of Computing Education Research*, Sally A. Fincher and Anthony V.Editors Robins (Eds.). Cambridge University Press, Cambridge, 327–376. https://doi.org/10.1017/9781108654555.013

[22] Simon, Andrew Luxton-Reilly, Vangel V. Ajanovski, Eric Fouh, Christabel Gonsalvez, Juho Leinonen, Jack Parkinson, Matthew Poole, and Neena Thota. 2019. Pass Rates in Introductory Programming and in other STEM Disciplines. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR '19)*. Association for Computing Machinery, New York, NY, USA, 53–71. https://doi.org/10.1145/3344429.3372502

[23] Arto Vihavainen, Thomas Vikberg, Matti Luukkainen, and Martin Pärtel. 2013. Scaffolding students' learning using test my code. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education (ITiCSE '13)*. Association for Computing Machinery, New York, NY, USA, 117–122. https://doi.org/10.1145/2462476.2462501

[24] Christopher Watson, Frederick W.B. Li, and Jamie L. Godwin. 2013. Predicting Performance in an Introductory Programming Course by Logging and Analyzing Student Programming Behavior. In *2013 IEEE 13th International Conference on Advanced Learning Technologies*. IEEE, San Diego, CA, 319–323. https://doi.org/10.1109/ICALT.2013.99 ISSN: 2161-377X.