# Predicting At-Risk Students for an Introductory Programming Course: A pilot study

**Norman Tiong Seng Lee**
Singapore University of Technology and Design
Singapore

**Oka Kurniawan**
Singapore University of Technology and Design
Singapore

Some novice learners of computer programming are at risk of doing badly in their first programming course. In this pilot study, we develop a logistic regression model to predict at-risk students in our introductory programming course. The model is developed using students' high school grades on mathematics, features calculated from log data, and scores from a programming quiz. The model suggests that students who have lower mathematics grade, who submit their homework assignments late, and who have lower scores in the programming quiz are more likely to be at-risk. We discuss some implications of this result on our teaching and learning strategies in our course.

Keywords: Learning analytics, computer programming, at-risk students

## Introduction

It is commonly accepted that learning programming is difficult for novice learners (Robins et al., 2003). Programming tasks tend to place several cognitive demands on learners, starting with the learning of language features, and ending with developing problem-solving skills (Linn and Dalbey, 1985). Thus it is not surprising that studies have reported that novice programmers lacked skills in reading and tracing code (Lister et al., 2004) and problem-solving ability (McCartney et. al., 2013; McCracken et. al., 2001).

In investigating the factors that determine students' programming ability, many studies have determined that ability in mathematics plays a role. For an introductory programming course (hereafter referred to as "CS1") aimed at Humanities undergraduates, Bryne and Lyons (2001) found that Irish Leaving Certificate Mathematics and Science grades had correlations with programming examination scores, with Mathematics having a weaker correlation compared to Science. In another Irish study, Bergin and Reilly (2005) found strong correlation between Mathematics and programming performance. Predictive models by Quille and Bergin (2018) included a feature on mathematics ability calculated based on high school exit examinations using data collected in Ireland and Denmark. In Botswana, Ayalew et al. (2018) found a moderately strong correlation between high school mathematics scores in and university programming performance.

Other studies have used in-course data such as log data from the learning management system (LMS) or other student-generated data to see what behavioural indicators are linked to programming ability. Behavioural features calculated from log data, such as time of submissions, number of submissions and total elapsed time were also calculated by Edwards et al. (2009). It was shown that students who started and finished their assignments earlier tended to get better grades in programming courses. Using statistical analysis, Willman et al. (2015) used time stamps on programming assignment submissions to derive features on student behaviour, such as the time of day of submission, submission counts and so on, and showed that students with the highest course grades tended to submit their work early and did not work at night.

Studies in other contexts and fields have used similar behavioural features. One feature calculated by Cerezo et al. (2016) from log data from an online course was the time taken by students to submit an assignment from the time it was first released. Results from clustering algorithms then suggested that students who take a longer time to submit their assignments did worse in the course. For an online course offered by a Korean university, a linear regression study by You (2015) showed that late submissions of tasks had a negative effect on the final course score.

Most educators want to identify students at-risk so that suitable interventions can be given in order to reduce the attrition rate and improve learning outcomes. One way of doing so would be to develop predictive models.

Since it has been shown that behavioural indicators can be extracted from log data and LMS data, predictive models have been built using such data to identify at-risk students. Ahadi et al. (2017) logged key presses in online programming exercises, and showed that the number of attempts on an exercise could predict performance on a final exam question. In a CS1 course, Porter et al. (2014), and a subsequent study by Liao et al. (2016) used

answers from classroom response questions at the start of the term and showed that the predictive models built could predict end-of-course performance, thus enabling timely intervention for at-risk students at an early stage. Using data from a first-year engineering course, Pardo et al. (2016) suggested that decision tree models built from log data could be used to give personalized feedback. Log data from LMS have been used to build predictive linear regression and logistic regression in various non-computing courses within the same university, and it was shown that the models generated were course-specific (Connijn et al, 2016).

Some studies have tried to build predictive models on introductory programming by incorporating data from more than one source. Apart from high school mathematics scores, Bergin and Reilly (2005) included data such as class test scores and lab test scores in their linear regression models. Bergin et al. (2015) used data from students' background and instruments such as those measuring self-efficacy, use of learning strategies and so on to build machine learning models. Predictive models by Quille and Bergin (2018) included various features using data collected in Ireland and Denmark, including a feature on mathematics ability based on high school exit examinations, students' self-efficacy and various demographic features.

Undergraduate students in the Singapore University of Technology and Design (SUTD) take compulsory common courses in their first three terms. "The Digital World" (DW) is an introductory Python programming course taken in their third term. Following the third term, students declare their major and choose one out of four "pillars", namely Engineering Product Development, Engineering Systems and Design, Information Systems Technology and Design and Architecture and Sustainable Design. All four pillars have courses that require a good foundation in programming. Hence it would be advantageous to any SUTD undergraduate to acquire programming skills.

As part of the coursework for DW, students have to complete a weekly problem set. Currently, we have a few measures to help students who need assistance in completing their problem sets. These include having an undergraduate teaching assistant (TA) present in regular classes, and weekly consultation sessions manned by senior students. However, the weekly consultation sessions are not well-attended, and the TAs do not get a lot of questions. Both these measures rely on students taking the initiative to make use of them. It would be useful to identify at-risk students so that we may consider other possible interventions for them.

We have no systematic means to identify at-risk students in our introductory programming course. However, we do possess a lot of data on our own students, such as their scores in the assessment components and log data from their interactions with the LMS. Hence, the research question in this study is, is it possible to build predictive models using the data that we have to identify students at-risk?

## Method

### Context

The largest group of students that apply to SUTD do so with the "A-level" qualifications. In the A-levels, students typically have to take Mathematics and three other subjects. The choice of subjects determines what university courses they qualify for. A smaller proportion of students, not considered in this study, enroll with diplomas from the five local polytechnics. Most female students join SUTD after completing the A-levels at age 18, while male students will join at age 20 after completing National Service.

In DW, like many courses in the first year at SUTD, students are divided into cohorts of about fifty students in a classroom setting and meet for five hours per week, divided over three sessions. In each session, a short lesson is given, then students are given time to attempt a weekly problem set by themselves. Instructors and the teaching assistant then circulate around the classrooms to observe students' work and answer questions.

The weekly problem set contains three categories of problems. The "Cohort Problems" (CH) are questions that instructors typically use as lesson examples and students are given time in class to complete them. The "Homework Problems" (HW) are questions that students are typically expected to complete in their own time. Both these problems account for a small percentage of the course overall score. The last category, "Exercises" (EX), are optional problems that are ungraded. These questions are meant for students who want additional practice. This study does not use data on the Exercises. In each category, there are between five to seven questions. For each week, the problem sets are released at 0001 hrs on Sunday and are due 2359 hrs on Tuesday of the following week.

Students submit their programming answers to our online submission platform, Vocareum (2019), which then assigns a score to the answer based on the number of test cases passed. Vocareum is also a platform for students

to receive automated feedback on their solution. After students submit their solution, we programmed it to display a description of the test cases passed and failed, enabling them to revise their solution on their own and then resubmit to obtain a better score. Students have unlimited chances to submit prior to the deadline for each question. Late submissions are possible for one week beyond the deadline with a 50% penalty on the scores. Vocareum also keeps a log of the usage by each student and this is what is used in this study.

Students also have individual assessments that form a larger proportion of their final score. We describe what was used for the student data in this study. The first is programming quizzes throughout the term. For the 2018 run of DW, they are conducted in-class and students are given one question to complete within 30 minutes. The mid-term exam and final exam have the largest weightage and these exams have two components, a written component (20% of the marks) and a problem-solving programming component (80% of the marks). Similarly, during these quizzes and exams, students submit their answers to Vocareum.

## Building the dataset

We gathered data from the students who matriculated in 2017, who then began DW in January 2018. We drew our data from three sources:
1. deriving features from Vocareum log data in week 4
2. prior performance at A-levels for Mathematics and Physics
3. scores of the programming portion of the mid-term and the programming quiz at week 4

*Deriving Features from Log Data*

Students submit their programming assignments to the CH, HW and EX problems in the problem set to our submission platform, Vocareum. The time at which the "Submit" button is last clicked for each question is logged and thus we are able to calculate the following features:

- "CH Submitted", "HW Submitted" – the number of problems submitted in the corresponding category
- "CH Average Days", "HW Average Days" – the number of days between the problem set release and the point of submission, averaged over all the questions in each category
- "Week 4 Programming Quiz Time" – the time taken to complete the programming quiz in Week 4, in minutes
- "Week 4 Programming Quiz Score" – the score for the programming quiz, integers from 0 to 4

We chose the features for CH and HW as they would give us an indication of how diligent students were in completing the problem set.

The time taken for the programming quiz was chosen because we had noticed that some students were able to complete the quiz very quickly, while some students struggled to produce a solution even when the quiz ended. We also included the score from the programming quiz as an indication of the students' performance on this quiz.

As this is a pilot study, we chose data from Week 4 of the course because this is when we introduce advanced programming concepts such as nested for-loops and dictionaries, and we tend to notice more students having difficulties.

*Prior performance at A-levels*

As some studies had reported that students' prior performance in Mathematics and Science had an effect on CS1 performance, we included students' grades in Mathematics and Physics at the A-levels in our dataset. For these subjects, letter grades from 'A' to 'E' are awarded. Students who scored 'A' in Mathematics formed the largest group, so we decided on the following categorical features:

- "IS MATHEMATICS A" – whether the student had scored grade 'A' in Mathematics (coded as 1) or not (coded as 0).
- "IS PHYSICS A" – whether the students had score 'A' for Physics (coded as 1) or not (coded as 0, which includes students who did not take Physics as a subject at the A-levels).

*Scores at mid-term exams*

As an indication of the students' programming ability, we included their scores on the programming portion of the mid-term exam, normalized to between 0 and 1 using min-max normalization. To enable classification algorithms to be applied, we categorized the students as "Weak" or "Not Weak" based on their percentile rank in their mid-term exam scores.

- "MidTerm Part B Score (Normalized)" - their scores on the programming portion of the mid-term exam, normalized to between 0 and 1
- "MidTerm Weak" – coded as 1 ("Weak") if their mid-term part B scores are in the 40$^{th}$ percentile and below, and 0 ("Not Weak") if otherwise. This 40$^{th}$ percentile threshold was decided based on our analysis (described below) and is similar to the value chosen in the study by Liao et al. (2016).

*Missing data and size of dataset*

As this is a pilot study, our dataset is limited to students who joined us with A-level qualifications. There were also students who did not attempt the programming quiz. We excluded these students from the final dataset, thus the dataset has, in its final form, n = 261 records.

### Modelling

To determine the features that best explained the variation in the dependent variable, "MidTerm Part B Score (Normalized)" , we first applied recursive feature elimination to select the best features for a linear regression model.

Following which, we divided our data into two sets, a training set (60%) and a test set (40%). We used the features that were determined earlier to train a logistic regression model, where the dependent variable was "MidTerm Weak". This model was then used to make predictions on the test set. The confusion matrix and its associated metrics were calculated.

The feature selection, model fitting and predictions were done using the "scikit-learn" Python module (n.d.). For each model, the p-values and the 95% confidence interval were obtained using the "statsmodels" Python module (n.d.).

## Results

### Linear Regression

The resulting linear regression model with three features is shown in Table 1. These features are the top three features ranked by the recursive feature elimination, and resulted in p-values of 0.001 or less. The 95% confidence intervals for these features do not include zero. Adding the lower-ranked features produced p-values larger than 0.05 and hence were not included in the model.

The $R^2$ score was 0.39, thus this model explains 39% of the variation in the dependent variable. This $R^2$ score is similar to scores reported in other studies employing linear regressions (Goold, 2000; Wilson et al., 2001; Bergin et al., 2005).

The coefficients are reasonable. The negative coefficient for "HW Average Days" suggest that students who take more time to submit their HW questions will do worse on the mid-term exam. The positive coefficients for the remaining features suggest that students who do better on their programming quiz and obtained A for A-level mathematics tended to do better at the mid-term exams. We visualized the data in Figure 1 which seems to confirm this trend.

Contrary to the results of Bryne and Lyons (2001), the feature "IS PHYSICS A" was not selected to be in the model. Putting this feature in the regression model resulted in a p-value exceeding 0.05. It is also not surprising that the CH features were not included as the cohort questions are used in classroom teaching and may not be indicative of individual student behaviour. Contrary to our expectations, the "Week 4 Programming Quiz Time" was not included in the model. This is probably due to test-taking behaviour where the majority of the students, regardless of ability, would want to make full use of the time that they are given in the test.

**Table 1: Coefficients of the linear regression model**

| Feature | Coefficient | P value | 95% confidence interval |
|---|---|---|---|
| Constant term | 0.5331 | <0.001 | (0.438, 0.628) |
| HW Average Days | -0.0368 | <0.001 | (-0.045, -0.028) |
| Week 4 Programming Quiz Score | 0.0347 | <0.001 | (0.017, 0.052) |
| IS MATHEMATICS A | 0.0697 | 0.001 | (0.030, 0.109) |

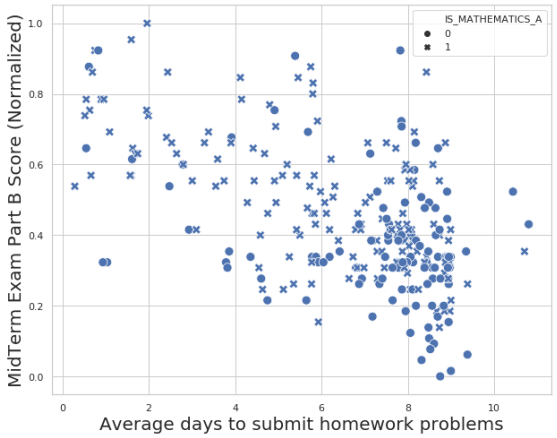


**Figure 1. The distribution of MidTerm Exam Part B score against the average days to submit homework problems . This plot suggests that students who get A at A-level Mathematics tended to do better at the mid-term exams and submit their homework problems earlier.**

### Predictions using Logistic Regression

Using the features selected earlier, the logistic regression model built with the training set is shown in Table 2. The target feature is "MidTerm Weak" with the threshold of the 40$^{th}$ percentile of the "MidTerm Part B Score" separating the "Weak" and "Not Weak" categories. All the coefficients have p-values of less than 0.05, and the 95% confidence intervals for these coefficients do not include 0. We also found that any other features added to the model gave p-values of larger than 0.05, hence they were not included in the model.

We find that the coefficients are reasonable. A positive coefficient for a feature suggests that an increase in that feature results in an increase in the probability of being in the "Weak" category. Hence, the positive coefficient for "HW Average Days" suggests that students who submit their homework problems later have a higher probability of being classified as "Weak". The negative coefficients for the other two features suggests that having a higher quiz score and an A-grade at A-level Mathematics reduces the probability of being classified as "Weak".

**Table 2: Coefficients of the logistic regression model**

| Feature | Coefficient | P value | 95% confidence interval |
|---|---|---|---|
| HW Average | 0.2511 | <0.001 | (0.1427, 0.3594) |
| Week 4 Programming Quiz Score | -0.2983 | 0.014 | (-0.5357, -0.0610) |
| IS MATHEMATICS A | -1.3276 | <0.001 | (-2.0183, -0.6370) |

The logistic regression model was used to make predictions in the test set. The resulting confusion matrix and the associated metrics are shown in Table 3. As we are interested in predicting at-risk students, the precision 0.70, suggesting that the model manages to predict 70% of the at-risk students correctly.

The recall is 0.74, suggesting that, out of the students predicted as "Weak", 74% of them have been identified correctly. Hence, a high recall value suggests that we have few false positives.

**Table 3: Confusion Matrix by the logistic regression model**

| | **Predicted Not Weak** | **Predicted Weak** |
|---|---|---|
| **Actual Not Weak** | 50 | 13 |
| **Actual Weak** | 11 | 31 |
| | **Precision**: 0.70 | **Recall**: 0.74 |

A Naïve Bayes model with the same set of three features obtained a precision of 0.65 and a recall of 0.71 on the test set. This suggests that the logistic regression model has a small improvement over a simpler predictive model like the Naïve Bayes.

Finally, to check that the threshold of the 40$^{th}$ percentile for the target feature "MidTerm Weak" was the best choice, we generated logistic regression models for thresholds of 20$^{th}$ percentile and 30$^{th}$ percentile and summarize the precision and recall obtained from these models in Table 4. It is clear that the 40$^{th}$ percentile threshold gave the best results.

**Table 4: Checking the threshold for "MidTerm Weak"**

| **Threshold for "MidTerm Weak"** | **20$^{th}$ percentile** | **30$^{th}$ percentile** | **40$^{th}$ percentile (as in Table 3)** |
|---|---|---|---|
| **Precision** | 0.56 | 0.57 | 0.70 |
| **Recall** | 0.22 | 0.64 | 0.74 |

## Discussion

The set of three features that have been included in the models seem to suggest that a mixture of pre-course and during-course factors will predict at-risk students. Our study includes the performance of A-level Mathematics as a feature in the models. This is a similar result to other studies that have provided evidence that performance in high-school Mathematics is a factor that influences performance in introductory programming courses (Bergin and Reilly, 2005; Ayalew et al., 2018), and in contrast to the weak correlation reported by Watson et al. (2013).

The "HW Average Days" feature is interesting as a behavioural feature. The models in this study have suggested that the later the student submits the homework problems, the more likely the student will do badly in the mid-term exams. In our course, the homework problems are designed to be more challenging than the cohort problems. Furthermore, at Week 4, we observe that students are beginning to struggle to understand the concepts taught in DW, and have a high academic workload from other courses. Hence, it seems to us that this is an indicator of how well a student is coping with their studies – a student who is struggling with the workload will tend to submit their assignments later compared to one who is coping well. This is similar to other studies that found that students who submit their assignments early tend to do better, both in the introductory computing context (Edwards et. al., 2009; Willman et. al., 2015) and in other contexts (Cerezo et al., 2016; You, 2015). Thus, the time taken to complete an assignment could be an important feature that can be used to predict student achievement in any context.

The fact that the Week 4 Programming Quiz score is included as a feature suggests that simple quizzes earlier on in the course can give an indication on which students are at-risk. However, currently, in DW, it currently takes more than two weeks for the quizzes to be marked. Hence, students do not receive feedback in a timely manner, and students will not be alerted on time that they are at risk. In contrast, using strategies such as Peer Instruction allows feedback in a timely manner and have predictive value (Porter et. al., 2014).

This study has some implications for our instructional strategies. Since the weaker students tend to submit the homework problems later, we would have to consider how we could craft some of those problems to provide scaffolding to slower learners. Since they also tend to have lower A-level mathematics grades, we would have to consider how we might help students develop problem-solving abilities, for example, perhaps by using a sequence of practice tasks that scaffolds students towards a complex task (Denny et al.,2018). We would also have to consider how we might assist students to assess their own abilities, and provide feedback in time. One option for this is to provide questions targeting specific programming concepts (Zingaro et al.,2012) that can be marked automatically by deploying them on learning management systems.

This study has some limitations. The models generated in this study are based on only one week of log data for a particular batch of students. Hence, it is not known if the same conclusions can be drawn if log data is used from

other points during the course. It is also not known if these models are applicable to subsequent batches of students taking this course. Also, we only incorporate one prior feature, students' mathematics grade at 'A'-levels, inside this analysis. However, we have observed that some students who join our course already have some programming ability that they might have picked up on their own. This could have a positive effect on their mid-term exam performance. Hence, future analysis could incorporate data from more than one batch of students, and include information on other pre-course factors such as existing programming ability.

## Conclusions

In this pilot study, from log data in one week of our course, we have extracted students' behaviour features and built a logistic regression model. Included in this model are also students' A-level scores on Mathematics and their score in a programming quiz. We found that this model can predict at-risk students, defined by their performance in a mid-term exam, with a precision of 0.70. The analysis also suggests some improvements in our instructional strategies. This model is done using one week of data in one term of the course, hence, more work is needed if predictions are to be made over a longer duration.

## Acknowledgements

We thank our colleagues at the Office of Admissions, Ms Grace Ang and Ms Lim Su Fang for providing us with the A-level grades of our students. This project is also funded by the Pedagogy Innovation Grant 2018-8048 administered by SUTD.

## References

Ahadi, A., Hellas, A., & Lister, R. (2017). *ACM Transactions on Computing Education,17*(3), 1-19.

Ayalew, Y., Tshukudu, E., & Lefoane, M. (2018). Factors affecting programming performance of first year students at a University in Botswana. *African Journal of Research in Mathematics, Science and Technology Education,22*(3), 363-373.

Bergin, S., & Reilly, R. (2005). Programming: Factors that Influence Success. *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education - SIGCSE 05*, 411-415.

Bergin, S., Mooney, A., Ghent, J., & Quille, K. (2015). Using Machine Learning Techniques to Predict Introductory Programming Performance. *International Journal of Computer Science and Software Engineering,4*(12), 323-328.

Byrne, P., & Lyons, G. (2001). The effect of student attributes on success in programming. *ACM SIGCSE Bulletin,33*(3), 49-52.

Cerezo, R., Sánchez-Santillán, M., Paule-Ruiz, M. P., & Núñez, J. C. (2016). Students LMS interaction patterns and their relationship with achievement: A case study in higher education. *Computers & Education*, 96, 42–54.

Conijn, R., Snijders, C., Kleingeld, A., & Matzat, U. (2017). Predicting Student Performance from LMS Data: A Comparison of 17 Blended Courses Using Moodle LMS. *IEEE Transactions on Learning Technologies,10*(1), 17-29.

Denny, P., Luxton-Reilly, A., Craig, M., & Petersen, A. (2018). Improving complex task performance using a sequence of simple practice tasks. *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE 2018*. 4-9

Edwards, S. H., Snyder, J., Pérez-Quiñones, M. A., Allevato, A., Kim, D., & Tretola, B. (2009). Comparing effective and ineffective behaviors of student programmers.

*Proceedings of the Fifth International Workshop on Computing Education Research Workshop - ICER 09*, 3-14.

Goold, A., & Rimmer, R. (2000). Factors affecting performance in first-year computing. *ACM SIGCSE Bulletin,32*(2), 39-43.

Liao, S. N., Zingaro, D., Laurenzano, M. A., Griswold, W. G., & Porter, L. (2016). Lightweight, Early Identification of At-Risk CS1 Students. *Proceedings of the 2016 ACM Conference on International Computing Education Research - ICER 16*. 123-131.

Lister, R., Seppälä, O., Simon, B., Thomas, L., Adams, E. S., Fitzgerald, S., … Sanders, K. (2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin*, *36*(4), 119–150.

McCartney, R., Boustedt, J., Eckerdal, A., Sanders, K., & Zander, C. (2013). Can first-year students program yet? *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, 91–98.

McCracken, M., Wilusz, T., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., … Utting, I. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin*, *33*(4), 125–180.

Pardo, A., Mirriahi, N., Martinez-Maldonado, R., Jovanovic, J., Dawson, S., & Gašević, D. (2016). Generating actionable predictive models of academic performance. *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge - LAK 16*, 474-478.

Porter, L., Zingaro, D., & Lister, R. (2014). Predicting student success using fine grain clicker data. *Proceedings of the Tenth Annual Conference on International Computing Education Research - ICER 14*. 51-58.

Quille, K., & Bergin, S. (2018). Programming: Predicting student success early in CS1. a re-validation and replication study. *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE 2018*, 15-20.

Robins, A., Rountree, J. & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, 13(2), 137-172.

Scikit-learn: Machine learning in Python. (n.d.). Retrieved from https://scikit-learn.org/

StatsModels: Statistics in Python. (n.d.). Retrieved from https://www.statsmodels.org/

Vocareum. (2019). Retrieved from https://www.vocareum.com

Watson, C., Li, F. W., & Godwin, J. L. (2014). No tests required: Comparing Traditional and Dynamic Predictors of Programming Success. *Proceedings of the 45th ACM Technical Symposium on Computer Science Education - SIGCSE 14*, 469-475.

Willman, S., Lindén, R., Kaila, E., Rajala, T., Laakso, M., & Salakoski, T. (2015). On study habits on an introductory course on programming. *Computer Science Education*,*25*(3), 276-291.

Wilson, B. C., & Shrock, S. (2001). Contributing to success in an introductory computer science course. *Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education - SIGCSE 01*, 184-188.

You, J. W. (2015). Examining the Effect of Academic Procrastination on Achievement using LMS Data in e-Learning. *Educational Technology and Society*, 18(3), 64 - 74.

Zingaro, D., Petersen, A., & Craig, M. (2012). Stepping Up to Integrative Questions on CS1 Exams. *Proceedings of the 43rd ACM Technical Symposium on Computing Science Education - SIGCSE '12*,253 - 258.

**Please cite as:** Lee, N.T.S & Kurniawan, O. (2019). Predicting At-Risk Students for an Introductory Programming Course: A pilot study. In Y. W. Chew, K. M. Chan, and A. Alphonso (Eds.), *Personalised Learning. Diverse Goals. One Heart. ASCILITE 2019 Singapore* (pp.178-185).