2019

CS1: how will they do? How can we help? A decade of research and practice

Keith Quille

Susan Bergin

CS1: how will they do? How can we help? A decade of research and practice

Keith Quille & Susan Bergin

To cite this article: Keith Quille & Susan Bergin (2019) CS1: how will they do? How can we help? A decade of research and practice, Computer Science Education, 29:2-3, 254-282, DOI: 10.1080/08993408.2019.1612679

To link to this article: https://doi.org/10.1080/08993408.2019.1612679

COMPUTER SCIENCE EDUCATION
2019, VOL. 29, NOS. 2–3, 254–282
https://doi.org/10.1080/08993408.2019.1612679

ARTICLE

CS1: how will they do? How can we help? A decade of research and practice

Keith Quille[a,b] and Susan Bergin[b]

[a]TU Dublin, Tallaght Campus, Dublin, Ireland; [b]Maynooth University, Co. Kildare, Ireland

ABSTRACT

Background and Context: Computer Science attrition rates (in the western world) are very concerning, with a large number of students failing to progress each year. It is well acknowledged that a significant factor of this attrition, is the students' difficulty to master the introductory programming module, often referred to as CS1.

Objective: The objective of this article is to describe the evolution of a prediction model named PreSS (**Pre**dict **S**tudent **S**uccess) over a 13-year period (2005–2018).

Method: This article ties together, the PreSS prediction model; pilot studies; a longitudinal, multi-institutional re-validation and replication study; improvements to the model since its inception; and interventions to reduce attrition rates.

Findings: The outcome of this body of work is an end-to-end real-time web-based tool (PreSS#), which can predict student success early in an introductory programming module (CS1), with an accuracy of 71%. This tool is enhanced with interventions that were developed in conjunction with PreSS#, which improved student performance in CS1.

Implications: This work contributes significantly to the computer science education (CSEd) community and the ITiCSE 2015 working group's call (in particular the second grand challenge), by re-validating and developing further the original PreSS model, 13 years after it was developed, on a modern, disparate, multi-institutional data set.

ARTICLE HISTORY
Received 6 August 2018
Accepted 25 April 2019

KEYWORDS
Introductory programming; predicting programming performance; interventions; CS1; attrition rates; programming performance; programming self-efficacy; machine learning; growth mindset; artificial neural networks

1. Introduction

Computer Science (CS) non-progression rates in Ireland are alarming, with a large number of students failing to progress each year. Currently, non-progression rates are 25% in CS, which is significantly higher than the national average of 16%. In two recent reports (2010 and 2016 respectively), CS was found to have one of the largest rates of non-progression across all National Framework of Qualification (NFQ) levels in Ireland, from diploma to degree level courses (Liston, Frawley, & Patterson, 2016; Mooney, Patterson, OConnor, & Chantler, 2010). In addition, CS is one of only two fields of study, where the

CONTACT Keith Quille keith.quille@it-tallaght.ie TU Dublin, Tallaght Campus, Dublin, Ireland

non-progression rate has increased since the first report in 2010. It is well acknowledged that the main contributor is students struggle to succeed in their initial programming module (CS1), a staple in most first-year CS courses.

Early identification of students who are at risk of non-progression is often hindered by very high student-lecturer ratios. Lecturers may not be aware that students are struggling until a considerable time has passed, and early problematic threshold concepts have been encountered. At our institutions numerous approaches have been trialled with varying degrees of success to improve learning and assessment outcomes, for example Traynor, Bergin, and Gibson (2006), Kelly et al. (2004) and Nolan and Bergin (2016). As computer science is not currently a Leaving Certificate subject in Ireland (it is planned to be made optional to schools in 2020, Quille, Faherty, Bergin, and Becker (2018)), there are no formal indicators of a students previous performance available at an early stage to enable the introduction of appropriate interventions. This often renders interventions inadequate, as their introduction may be too late in the course to make a significant difference.

Computer Science Education (CSEd) research is a relatively young field of study ($\approx$ 50 years, Becker and Quille (2019)). A number of models exists to identify students at risk of dropping out or failing; however, most models are only used for a brief period of time and are not developed further. This was highlighted by a call from the ITiCSE 15 working group, which identifies several grand challenges. They noted that while identifying students at risk of dropping out or predicting performance has been investigated, the studies are seldom revisited or tested for generalizability. In addition, the models have not been employed in actual interventions. The working group also highlighted as a separate grand challenge, the critical need for re-validation of educational data mining models (Ihantola et al., 2015). Thus, the development of a complete system that can predict struggling students in a timely manner and act accordingly (using one or multiple interventions), would make a significant contribution to the CSEd community.

2. Literature review

Over the years there has been a significant amount of research related to predicting student success in CS1, using educational data mining techniques. This literature review serves three objectives. The first objective is to identify models to predict success on introductory programming courses. Second, is to investigate how these models have been developed further/revalidated after their initial use. The final objective is to compare the findings of the first two objectives with a local prediction model named PreSS (**Pre**dict **S**tudent **S**uccess).

2.1. Methodology

A robust approach that would ensure the identification of relevant research, given the large quantity available, was required (some searches returned hits in excess of 70,000 results). Search terms were identified that included one or more of predicting, predict, CS1, introductory programming, factors, ability, performance, success, failure and student. Combinations of these terms were then searched in the ACM and IEEE databases with additional searching in Google Scholar. The search terms were examined in the title, abstract and the body of publications. In the case of large search returns, the first 200 results were reviewed, where results were filtered on relevance to the search term. Where the search returned less than 200 results, all the results were reviewed. In total 1,884 articles were reviewed based on search terms appearing in the title, abstract and/or body. From there, articles were shortlisted, based on their relevance to CS, factors and prediction models. This resulted in 94 articles (when repeating articles returned in multiple searches were removed). Following this, a detailed analysis of each article and its relevance to introductory programming courses were conducted, resulting in 47 articles that were included in this literature review. This process ensured that the articles were concerned with factors that influence performance or models to predict performance on introductory programming courses. Ideally, models to predict student's performance, would satisfy some or ideally all of the following criteria: conducted across multiple institutions, longitudinal in nature, of good sample size, and resulted in a high level of performance at an early stage in a CS1 course. Such criteria were also highlighted by an ITiCSE working group report (Ihantola et al., 2015).

Almost all of the 47 articles examined, exhibited one or several (but not all) of these criteria (where three of the articles did not satisfy any of the criteria: Evans and Simkin (1989); Pioro (2004); Porter and Zingaro (2014)). The articles have been grouped and presented under each criterion as headings below. Under the headings, the articles are referenced with a key (not the full citation as some headings have up to 36 citations), which is then indexed in Table 1, with the articles full citation. Table 1 also summarizes which of the criteria/criterion that each article met.

2.2. Multi-institutional

Ideally, to create a generalizable prediction model, it would need to be tested over several institutions, preferably in diverse districts or countries. From the 47 articles examined, only two studies were conducted in more than a single institution [10, 38]. Simon et al. [38] conducted a study across 11 institutions ($n = 177$), exploring issues that influence success in learning to program, using four diagnostic tasks. Although the study was carried out at 11 institutions, the

Table 1. Summary of literature review references.

Key	Reference	Multi-Institutional	Longitudinal	Large Sample	High Accuracy	Early Timing	Sen and Spec	Revalidated
1	Bergin (2006) – PreSS	✓	✓	✓	✓	✓	✓	✓
2	Ahadi, Lister, Haapala, and Vihavainen (2015)		✓	✓	✓	✓		
3	Allert (2004)			✓		✓		
4	R. J. Barker and Unger (1983)			✓		✓		
5	L. J. Barker, Mcdowell, and Kalahar (2009)			✓				
6	Bennedsen and Caspersen (2005)			✓		✓		
7	Bennedsen and Caspersen (2006)			✓				
8	Bennedsen and Caspersen (2008)			✓				
9	Boetticher, Ding, Moen, and Yue (2005)				✓		✓	
10	Bornat et al. (2008)	✓		✓		✓		
11	Butcher and Muth (1985)		✓	✓		✓		
12	Campbell, Horton, and Craig (2016)			✓		✓		
13	Capstick, Gordon, and Salvadori (1975)					✓		
14	Caspersen et al. (2007)			✓		✓		✓
15	Cukierman (2015)			✓				
16	Dehnadi (2006)				✓	✓		✓
17	Denny, Luxton-Reilly, Hamer, Dahlstrom, and Purchase (2010)			✓		✓		
18	Estey and Coady (2016)		✓	✓		✓		
19	Evans and Simkin (1989)							
20	Fowler and Glorfeld (1981)			✓	✓	✓	✓	✓
21	Glorfeld and Fowler (1982)		✓	✓	✓	✓	✓	✓
22	Golding, Facey-Shaw, and Tennant (2006)					✓		
23	Hostetler (1983)			✓		✓		
24	Konvalina, Wileman, and Stephens (1983)			✓		✓		
25	Lambert (2015)			✓		✓		
26	Leeper and Silver (1982)				✓			
27	Leinonen, Leppänen, Ihantola, and Hellas (2017)			✓				
28	Liao, Zingaro, Laurenzano, Griswold, and Porter (2016)		✓	✓	✓	✓	✓	
29	Lishinski, Yadav, Good, and Enbody (2016)			✓		✓		
30	Lishinski, Yadav, Enbody, and Good (2016)			✓		✓		
31	Newsted (1975)			✓				
32	Pioro (2004)							
33	Porter, Zingaro, and Lister (2014)					✓		
34	Porter and Zingaro (2014)							
35	Rountree, Rountree, and Robins (2002)			✓		✓		
36	Rountree, Rountree, Robins, and Hannah (2004)			✓		✓		

(*Continued*)

Table 1. (Continued).

Key	Reference	Multi-Institutional	Longitudinal	Large Sample	High Accuracy	Early Timing	Sen and Spec	Revalidated
37	Shell, Soh, Flanigan, and Peteranetz (2016)			✓		✓		
38	Simon et al. (2006)	✓		✓		✓		
39	Tarimo, Deeb, and Hickey (2016)			✓		✓		
40	Ventura (2005)		✓	✓			✓	
41	Vihavainen (2013)			✓	✓			
42	Watson, Li, and Godwin (2013)				✓	✓		
43	Werth (1986)			✓		✓		
44	Wiedenbeck (2007)			✓				
45	Wiig (1989)		✓	✓	✓	✓		
46	Wileman, Konvalina, and Stephens (1981)			✓				
47	Wilson and Shrock (2001)			✓	✓			

small sample size (∼16 students per institution) makes it difficult to assess how generalizable the model is. A study by Bornat et al. [10] revisited a predictor of CS1 success, developed by Dehnadi [16]. Bornat et al. conducted this revalidation study of Dehnadi's predictor of programming success across six institutions. The study reported that the predictor failed to produce a strong prediction when validated across six institutions.

2.3. *Longitudinal*

Studies are often conducted once, on a single cohort. Given that CS is constantly evolving, studies should be repeated over several years, to examine if they continue to hold valid results. The literature review found that only seven studies were conducted over more than one year or semester [2, 11, 18, 21, 28, 40, 45]. No study that was longitudinal involved more than one institution.

2.4. *Generalizable sample size*

To test a prediction model, reasonable sample size is required. Small sample size can be acceptable if it represents the entire population. As the goal of this model is to generalize across institutions and countries (large populations $n > 5000$ (Conroy, 2016)), a 10% acceptable margin of error was selected as the boundary value for the minimum generalizable sample size (Conroy, 2016; Naing, Winn, & Rusli, 2006). This resulted in a minimum sample size of 96 students. Several studies involved relatively small samples sizes ($n < 96$), where some did not include the sample size at all. This may pose problems with overfitting. In several studies that had a large sample size, there was no model, just correlations reported. These have value, but unless they are developed into a final model, may not serve practitioners in a useful immediate way. We found that 36 of the articles reported a sample size greater than 96 students [2–8, 10–12, 14, 15, 17, 18, 20, 21, 23–25, 27–31, 35–41, 43–47]. A positive note from this was that all studies that were longitudinal also included a generalizable sample size.

2.5. Prediction timing

Prediction timing is the point in the course the prediction could be made, ideally the earlier the better and with a prediction accuracy higher than that of chance, thus allowing educators to implement interventions in a timely manner. Thirty-one articles reported that they could predict performance before a quarter of the module was completed [1–3, 6, 10–14, 16–25, 28–30, 33, 35–40, 42, 43, 45]. A very positive finding in the literature is that some models were able to predict before the commencement of CS1 [11, 20, 21, 24]. In addition, 26 articles that satisfied the prediction timing criteria, also satisfied the generalizable sample size criteria.

2.6. Prediction accuracy

Several of the articles reported no significant prediction accuracies or correlations. A prediction slightly higher than that of chance was selected as search criteria. A similar correlation coefficient was selected so not to rule out this research. Twelve articles reported significant prediction accuracies, although some did not predict early in CS1 [2, 9, 16, 20, 21, 26, 28, 41, 42, 44, 45, 47]. The three studies that met all of the criteria (longitudinal, generalisable sample size, early prediction, excluding multi-institutional) that could predict with significant accuracy are [2, 21, 28].

2.7. Prediction sensitivity & specificity

Prediction models are often presented with high accuracy, but accuracy alone does not highlight outcome per class, for example how well it can predict strong or weak students. Measures such as sensitivity (the ability to predict weak students) and specificity (the ability to predict strong students) are also important measures. Sensitivity and specificity were only reported in four studies (in some cases indirectly, but it could be calculated) [9, 20, 21, 28]. Only three studies thus remained that met all of the criteria (excluding multi-institutional) that also reported sensitivity and specificity: Glorfeld (two consecutive studies) [20, 21] and Liao [28].

2.8. Revalidation of prediction models

From the literature, it appears that models are rarely revisited. In all of the literature reviewed, only two instances where the work was revisited were found:

Dehnadi (2006) [16] developed a prediction model in the UK that reported a 100% accuracy (100% sensitivity and specificity (n = 60 students)). This work seemed to have made a breakthrough. It was disclosed at the PPIG (Psychology of Programming Interest Group) workshop in 2006. Based on this reported accuracy, two follow up studies were completed [10, 14]. Caspersen, Larsen, and Bennedsen (2007) [14] repeated the study using approximately 142 students in Denmark. The findings of Caspersen's work are best described in the abstract: "We have repeated their test in our local context in order to verify and perhaps generalise their findings, but we could not show that the test predicts students success in our introductory programming course". Subsequently, a study by Bornat [10] in the following year (2008, co-authored by Dehnadi) examined six experiments, with more than 500 students, across six institutions and three countries (Bornat et al., 2008). Bornat reported that "the predictive effect of our test has failed to live up to that early promise" with performance, just higher than chance.

Fowler and Glorfeld (1981), developed a predictive model using a sample size of 151 students in a CS1 course [20]. A logistic discrimination model was developed from personal, academic and aptitude data. The model produced an accuracy of 80.8% and could identify weaker students with 76.6% accuracy. A year later Glorfeld and Fowler (1982), revisited the study with a new cohort [21]. From the 1040 students enrolled in the CS1 course, 150 were randomly selected. The model still performed well, although the accuracy decreased ($\sim 6\%$). This is perhaps to be expected when models are exposed to new data sets and being tested for generalizability.

2.9. The original press study and model

Bergin (2006) [1] developed a prediction model named PreSS (Predict Student Success). PreSS was able to identify at an early stage (10% into an introductory programming module), in CS1, students that may be at risk of failing or dropping out. The model was developed in a longitudinal study between 2002 and 2006. The study used a sample size of 102 students (in the main study and 184 in total). The study was multi-institutional consisting of a University, two Institutes of Technology (comparable in academic level to colleges in the US), and a Community College. The PreSS model used three factors to predict student success, specifically, programming self-efficacy, mathematical ability and number of hours per week a student plays computer games. This body of work is well regarded, having the 43rd highest cited publication in any of the ACM SIGCSE sponsored proceedings or publications, from a total of 13,389 (Bergin & Reilly, 2005). Detailed information on the factors, factor selection, data processing and the machine learning algorithm can be found in the references (Bergin & Reilly, 2005). Six machine learning algorithms were examined in the development of PreSS and naïve Bayes was selected as it was found to have the highest prediction accuracy (Bergin, 2006; Bergin, Mooney, Ghent, & Quille, 2015a). Prediction success was significant with a prediction accuracy of 77%, but more importantly, had a sensitivity of 85%.

The selection process to label programming performance as weak (sensitivity value) or strong (specificity value), was developed using the following criteria: (i) each institutions marks and standards (ii) progression rates (after CS1 into CS2 or Semester 3) considering student grades from each institution such as Grade Point Average (GPA) requirements for progression (iii) discussion with instructors at each level/institution determining a minimum grade for progression or success and finally (iv) in the case of the community college, the minimum requirements to enter an institute of technology or a university. Boundary value testing ($\pm$ 10%) was implemented to investigate the confidence of these values, where the differences found in the accuracy were statistically insignificant, with minimal changes in sensitivity and specificity. Thus, providing confidence in the selected border values used to identify strong and weak students. The equations for accuracy, sensitivity and specificity are presented as:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (1)$$

$$Sensitivity = \frac{TP}{TP + FN} \qquad Specificity = \frac{TN}{TN + FP} \quad (2)$$

where TP = True Positive, TN = True Negative, FP = False Positive and FN = False Negative.

2.10. Summary

In summary, the literature review revealed that while many of the 47 articles contribute to the CSEd community, the process confirmed several of the ITiCSE working group concerns (Ihantola et al., 2015). One of the most prominent issues is that the CSEd community finds repeatability a challenge, replication studies are rare, and that they may still be as rare a decade from now. It is also concerning that not only do the studies fall short of acquiring all of the desired model criterion, they struggle to attain multiple criteria, as presented in Table 1. Glorfeld in 1981 and in 1982 [20, 21] developed a strong prediction model, employing statistical techniques for generalization (ten-fold cross-validation) and confusion matrices, allowing specificity and sensitivity to be calculated. The paper concluded with a call for models in other institutions, and comparisons to be investigated. In addition, the paper promised to run this research on a yearly basis, but the work was never repeated.

The original PreSS model shared some factors with the Glorfeld model, namely age and mathematical ability. While the model had a strong prediction accuracy and was able to predict at an early stage in the CS1 module, the initial study and the re-validation were conducted in consecutive years, in the same institution. Thus, it is difficult to assess how this model would perform 35 years after its development and generalize to other institutions, different academic levels and countries. The original PreSS model predicted with marginal increases over the Glorfeld model, but the original PreSS model was validated across multiple institutions, over 3 years. Thus, the PreSS model remains the starting point for this body of work, but factors identified by Glorfeld will be considered in the further development of the PreSS model.

3. Justification study

A justification study consisting of two consecutive independent studies was carried out in the academic years of 2013–14 and 2014–15, respectively. The goal was to revalidate the PreSS model, given the considerable passage of time. The justification study took place in a Community College, this is a similar institution to one of the institutions investigated in the PreSS main study. The participants were studying on a Level 6 (diploma level) Computer Science course where all students completed the same "Introduction to Computer Programming" CS1 module. The language used was C# and this was the first time that this language was used with PreSS. It also appears to be the first time it was used in any documented study to predict programming performance. The grading criteria in the CS1 course consisted of two programming assignments worth a total of 600-grade points and a written examination worth 400-grade points. Each student also completed an online survey of questions on factors that potentially influence success including self-efficacy, prior maths performance and game playing (Quille & Bergin, 2016b).

The first study consisted of 34 students (all students in the class participated). There was no missing data entries, thus no student was excluded from the final sample. The overall final results showed the ratio of weak to strong students was 16:18, respectively. The second study consisted of 26 students (all students in the class participated). Again, there was no missing data entries so no student was excluded from the final sample. The overall final results showed the ratio of weak to strong students was 9:17.

3.1. Results

The original PreSS data samples ($n = 102$, from the main study of the original PreSS study) were used as training data to determine if PreSS could predict performance at a comparable level to the original studies. The process was identical to the process used in the development of the original PreSS model, using the same machine learning toolbox and methods to compute prediction. The results of the two justification studies were independently compared to the original PreSS model accuracy. Statistical *t*-tests (using a Welchs *t*-test and a binomial distribution to calculate the variance) indicated there was no statistically significant difference between the accuracy achieved on the original PreSS study and either of the justification studies (first justification study: $Accuracy = 76.5\%, p = 0.86$, second justification study: $Accuracy = 77\%, p = 0.57$, where the sensitivity was significantly higher than that of PreSS, $p < 0.001$). This is a significant result as both studies consisted of very different student profiles (community colleges represented less than 7% of students in the original PreSS study, and have lower entry requirements than institutes of technology and universities). Even though the original PreSS model was developed over a decade previous, the findings here suggest that similar levels of accuracy may be achievable in a considerably changed landscape and further research of the Press model is justified.

4. Research goals

Following the positive preliminary findings of the Justification study, this body of work aims to revalidate and extend the previous PreSS studies. The research goals are the following (with Figure 1 as a visual representation):

4.1. The research goals

RG-1: Develop a web-based real time implementation of PreSS – PreSS#

RG-2: Revalidate the PreSS model on a large multi-institutional data set

RG-3: Investigate new factors that may improve the performance of the PreSS model

RG-4: Investigate additional algorithms that may improve the performance of the PreSS model

RG-5: Develop interventions based on the findings of the previous research goals

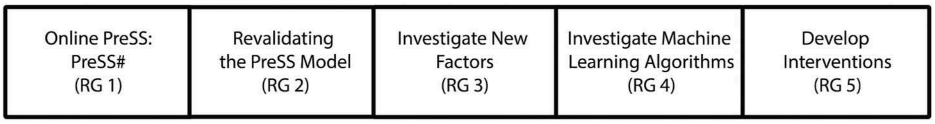
Online PreSS: PreSS# (RG 1)
Revalidating the PreSS Model (RG 2)
Investigate New Factors (RG 3)
Investigate Machine Learning Algorithms (RG 4)
Develop Interventions (RG 5)

Figure 1. Timeline and overview of the body of research and corresponding research goals.

5. PreSS#: RG-1

Research goal (RG-1) was to eliminate the labour-intensive paper-based data collection, manual processing and computation that the original PreSS model required, thus allowing PreSS to be used on significantly larger student cohorts across multiple institutions. This section provides a high-level outline of the PreSS# system, a detailed description of the software development process can be found in the references (Quille, Bergin, & Mooney, 2015). PreSS# (Predict Student Success Sharpe) is a web-based educational system that was completed in 2015 (Quille et al., 2015). PreSS# accurately replicates the performance of PreSS, is fully functional and can compute predictions in real time with cross-browser (mobile and desktop) compatibility. PreSS# was developed using a modular approach, and used the same process as PreSS, with the same data processing techniques and prediction algorithm, with the ability to include/remove factors and exchange prediction algorithms.

The system was developed using a Software as a Service (SaaS) model incorporating Model, View Controller (MVC) architecture. Security and scalability were key components. The user interface and institution/instructor interaction were further developed into a visualisation engine and analyser thus adding to PreSS# a streamlined user interface, an easy acquisition process, automatic modelling and reporting tools (Culligan, Quille, & Bergin, 2016; Quille et al., 2015). The PreSS# system was investigated to examine if it could accurately replicate the results of the original PreSS study using the same techniques as the original PreSS study and the 2006 main study dataset, $n = 102$. Bergin used Java implementations of the machine learning algorithms from the Waikato Environment for Knowledge Analysis toolbox, WEKA (Witten, Frank, Hall, & Pal, 2016), when developing models and for the final PreSS model. Thus, PreSS# was benchmarked against the WEKA toolbox, as it was developed using a C# .NET implementation. The investigation found that PreSS# had an accuracy identical to the accuracy of WEKA. A two-tailed t-test for a binomial distribution was run that confirmed that the prediction accuracies of each model were not statistically different ($p = 1.0$).

6. Main study: RG-2

Given the positive findings from the justification study, a large-scale revalidation was undertaken to determine if PreSS generalized on a substantially larger data set (the ease of collection and analysis of the study were facilitated through the development of PreSS#). This section contributes significantly to the computer science education (CSEd) community and the ITiCSE 2015 working group's call (in particular the second grand challenge), by revalidating the original PreSS model, 12 years after it was developed, on a modern, disparate, multi-institutional dataset.

During the academic year, 2015–16, a large-scale multi-institutional study (Quille, Culligan, & Bergin, 2017; Quille& Bergin 2018) took place in Ireland (10 institutions) and Denmark (one institution). This included: two universities, five institutes of technology and four community colleges. The data collected (using PreSS#) was classified under two categories. The first category captured student data, including background, institution, course and psychological data. This data was captured at approximately 4–6 hours into CS1 (this is when students are ~ 10% of the way through the module). The second category captured final CS1 performance data, such as grade. In total, 692 complete student data sets were used in the study. Six programming languages were used which included: Java (n = 553), C# (n = 75), Python (n = 33), Processing (n = 24), Visual Basic (n = 4) and C++ (n = 3). The main study while addressing RG 2 also collected a multitude of additional factors and data so that RG 3 (further development of the factors of the original PreSS model) could be addressed. In total 17 factors were collected using the online survey (after data reduction techniques were applied), and are presented in Table 2. The same three factors from the original PreSS study were used

Table 2. The 17 factors included in the main study (after data reduction).

Factor Name	Factor Details
Institution type	University, Institute of Technology, Community College
Time to complete the survey	Seconds
Age	Years as an integer
Mature Student	Dichotomous, under or over 23 years of age
Gender	Dichotomous, male of female
Social Media	Average time in hours per day spent on social media
Part Time Job	Average time in hours per week spent working in a part time job
Expected end of year result	Percentage Grade {0 − 100%}
Concepts, Design and Completion of a program	Likert Scale – Normalized
Intrinsic Goal Orientation	Likert Scale – Normalized
Intrinsic Questioning	Categorized based on Intrinsic goal Orientation {1, 2, 3}
Control of Learning Beliefs	Likert Scale – Normalized
Test Anxiety	Likert Scale – Normalized
Mathematical Grade	Normalized, accounting for various exam types/levels
Playing Games During	Average time in hours per day spent playing computer games during the course
Playing Games Before	Average time in hours per day spent playing computer games before taking the course
Programming Self-Efficacy	Ten questions reduced to one value using principle component analysis

Table 3. The original PreSS study compared to the main study, using the original PreSS model.

Data Set	N	Accuracy	Sensitivity	Specificity
2005	102	77%	85%	66%
Main Large Scale Study	692	67%	78%	53%

for prediction: programming self-efficacy, mathematical ability based on a high school mathematics exit examination and number of hours per week a student plays computer games (Bergin & Reilly, 2005). The results and comparison between the original study and this study are presented in Table 3.

The study was able to predict with high accuracy (67% – for two out of every three students, weak or strong, *n* =692), from 11 institutions with diverse academic levels in two different countries. Although the accuracy reduced by 10% given that the study was conducted 12 years after the original model was developed, this result is significant, especially given the changing landscape in CS education (student profiles, technologies, etc.) and given that previous revalidation attempts on other models (Section 2) have seen the model rejected or an accuracy reduction of 6% in the span of a single year, within in a single institution. More importantly, PreSS was able to identify weak students (the main goal of PreSS) with a sensitivity of 78%. In addition, the size of the study was significantly larger than the original PreSS study, so perhaps the difference with a comparable number of students and institutions may be smaller. From the literature, it appears that no other model for predicting programming performance has been revisited over 10 years after its creation (multi-institutional and longitudinal). The results here can provide a benchmark for future revalidation studies for predicting student success in CS1. This works satisfy-the second research goal of this article and furthermore contributes significantly to the call of the ITiCSE 15 working group for revalidation studies.

7. Investigating new factors: RG-3

This section documents three pieces of work. The first piece of work (referred to as Factor Selection 1) is an investigation on factors collected during the original PreSS study, but not used in the PreSS model. This is important as several factors were found to be significant at the time but were excluded as their associated sample size was small.

The second piece of work (referred to as Factor Selection 2) is an investigation of additional factors collected during the justification study, as presented in Section 3. This was to determine if incorporating some or all of these newly identified factors could improve the accuracy of PreSS.

The third piece of work incorporates the most predictive factors from Factor selection 1 and Factor Selection 2. The objective is to determine if a higher performing model can be achieved using some or all of these factors.

7.1. Factor selection 1: from the original press study

This section investigates factors collected during the original PreSS study, but not used in the PreSS model. As noted a paper-based collected data collection method was used in the original study and thus missing data was an issue with some instances (student data points) not having any data collected for some factors, thus reducing sample size (initially 123 students took part, with 102 students having all three of the original PreSS Factors). In some cases, the small sample size precluded factors for selection in the final model. Two-factor selection algorithms were used: correlation evaluation and information gain (Witten et al., 2016), where the top three models are presented in Table 4. For comparison purposes and to ensure an unbiased statistical comparison, the original PreSS model was re-run using samples with complete data sets for each of the factors under investigation. A Student's *t*-test was used to examine if a significant difference was found in accuracy between the baseline and the updated model. Model 2 is particularly noteworthy with almost 8% of an increase in accuracy. In all cases, the new model factors showed potential.

7.2. Factor selection 2: from the justification study

This section examines additional factors collected during the justification study (the second of the two justification studies with further details in Quille & Bergin (2015, 2016b)). From the additional factors, several new models were developed and investigated. With the use of PreSS# for data collection, there was no missing data, so all students were included in each experiment.

When age was added as a factor (as identified as potentially valuable by Fowler and Glorfeld (1981)), represented in years, or dichotomously, it produced statistically significant increases in accuracy over the original PreSS model. The addition of gender (in dichotomous form), yielded a statistically significant increase in accuracy at $\sim$ 4% on the justification study dataset. The time spent on social media when added to the PreSS model did not increase the accuracy; however, it did produce an

Table 4. The 3 models that resulted in a significant gain in accuracy over the original PreSS model.

ID	New Model Attributes	Sample Size (N)	Accuracy Increase %	P value
1	Mathematical grade, hours spent playing computer games and what a student believed their final overall grade would be at the beginning of the module.	56	8.93	< 0.001***
2	Mathematical grade, programming self-efficacy and the difference in hours spent playing computer games before the commencement of the course with that of the hours spent playing computer games during the course.	107	7.5	< 0.001***
3	Mathematical grade, programming self-efficacy, hours spent playing computer games and what a student believed their final overall grade would be at the beginning of the module.	56	7.2	< 0.001***

** Significant at p < 0.05; ** Significant at p < 0.005; *** significant at p < 0.001;*

Table 5. Justification study additional survey factors that may have shown to have value.

ID	New Model Attributes	Sample Size (N)	Accuracy Increase %	P value
1	Mathematical grade, programming self-efficacy, hours spent playing computer games and age (actual age in years).	26	7.7	< 0.001***
2	Mathematical grade, programming self-efficacy and gender.	26	3.85	< 0.001***
3	Mathematical grade, programming self-efficacy, hours spent playing computer games and age (dichotomous age value for mature and non-mature students)	26	3.85	< 0.001***

** Significant at $p < 0.05$; ** Significant at $p < 0.005$; *** significant at $p < 0.001$;*

interesting outcome. The top three models incorporating the new factors are presented in Table 5.

7.3. Investigating the new factors using the main study

The 17 factors collected in the main study, included factors identified as important, from Factor Selection 1 and Factor Selection 2, along with additional factors that were hypothesised to add value to the model. Using the main study dataset ($n = 692$), two-factor selection algorithms were used: correlation evaluation and information gain (as applied in Factor Selection 1). Both of these algorithms were run on the data set, and combinations of the highest ranked factors were examined. A multitude of models was investigated with the top two presented in this section. These two models produced either the strongest performance or added value to the model. The following models are proposed as preferable to the original PreSS model.

7.3.1. Recommended model 1

Model 1 included: a student's age in raw integer form, a student's expected end of module result, a student's mathematical ability (normalized) and a student's programming self-efficacy. This model was selected as a primary candidate for the updated PreSS model as it reported the highest accuracy (71%) and one of the highest sensitivities (75%).

7.3.2. Recommended model 2

Model 2 included: Institution type (University, College or Community College), mature student, over 23 years of age, in dichotomous form, a student's end of school mathematical result normalized and a student's programming self-efficacy. This model was also selected as a primary candidate for the updated PreSS model. Its accuracy was among the highest recorded (69%) and the model's sensitivity was higher than model 1 (an increase of 2% at 80%).

The results presented in this Section are very positive. The increases in performance show that PreSS may be able to produce further performance gains. This satisfies RG-3, to examine additional factors for the PreSS model (further developing the PreSS model). Next, an evaluation of different types of machine learning algorithms is explored (Section 8).

8. Machine learning algorithms: RG-4

Given the new models resulted in higher performance (recommended model 1 and recommended model 2), a different machine learning algorithm could potentially improve the PreSS model's performance even further. In addition, as some algorithms were not readily accessible a decade previous (such as deep learning), critiquing these algorithms, may improve the PreSS model further again. This section is divided into two investigations, machine learning algorithms and artificial neural networks.

8.1. Comparing machine learning algorithms

8.1.1. Original press study

In the original work by Bergin (2006) and revisited by Bergin, Mooney, Ghent, and Quille (2015b), six machine learning algorithms were investigated for use in the PreSS model. In both studies, the original PreSS dataset was used ($n = 102$). The results of this work are presented as an overview in Table 6.

8.1.2. Updated PreSS models

The algorithms were re-examined, but this time using the recommended models as developed in Section 5. The models were examined on the main large-scale study data set ($n = 692$). For analysis of both models, a one-way ANOVA was implemented.

Recommended model 1. Table 7 presents the performance of each machine learning algorithm using model 1. There was a statistically significant difference between groups accuracies as determined by one-way ANOVA ($F(5, 4146) = 63.7321, p = 0.0000$). Table 8 presents further analysis using a Tukey HSD post-hoc test, presenting the p values on the algorithm's accuracies. The first three algorithms' accuracy differences (Naïve Bayes, SVM and

Table 6. Performance of machine learning algorithms from the original PreSS study, using the same factors and data processing techniques and 10 fold cross-validation to examine performance, n = 102.

Algorithm	Acc %	Sen %	Spec %
Naïve Bayes	78.28	87	66
Logistic regression	76.47	84	65
Backpropagation	75.46	84	63
SVM	77.49	87	63
C4.5	74.49	85	63
K-nearest neighbour	74.49	85	63

Table 7. Performance of machine learning algorithms for Model 1, n = 692.

Algorithm	Acc %	Sen %	Spec %
Naïve Bayes	71	75	66
SVM	70	79	57
Logistic regression	70	78	59
C4.5	68	72	62
Backpropagation	66	72	58
K-nearest neighbour	61	67	54

Table 8. Tukey HSD post-hoc *p* values of machine learning algorithms for Model 1, with accuracy values from Table 7, and a confidence interval of 95%.

	Naïve Bayes	SVM	Logistic regression	C4.5	Backpropagation	K-nearest neighbour
Naïve Bayes	-	0.6540	0.6540	0.0001	0.0000	0.0000
SVM	0.6504	-	1.0000	0.0293	0.0000	0.0000
Logistic regression	0.6540	1.0000	-	0.0293	0.0000	0.0000
C4.	0.0001	0.0293	0.0293	-	0.0293	0.0000
Backpropagation	0.0000	0.0000	0.0000	0.0293	-	0.0000
K-nearest neighbour	0.0000	0.0000	0.0000	0.0000	0.0000	-

Logistic Regression) are not statistically significant. C4.5, Back propagation and K-nearest neighbour produced results that were statistically significant (lower) when compared to each of the first three algorithms.

The difference in sensitivity was not statistically significant between the SVM and Logistic Regression algorithms as determined by one-way ANOVA ($F(3, 2764) = 1.5447, p = 0.2009$); however, when naïve Bayes was included, the difference was statistically significant ($F(2, 2073) = 24.7656, p = 0.0000$). Thus concluding, that although the top three algorithms performed with statistically similar accuracy, SVM and Logistic Regression outperform naïve Bayes for sensitivity.

Recommended model 2. For Model 2, the difference in accuracy was not statistically significant between the top four performing algorithms as determined by one-way ANOVA ($F(3, 2764) = 1.5447, p = 0.2009$). When the top five algorithms are investigated the difference is statistically significant ($F(4, 3445) = 3.2279, p = 0.0118$) and when all six are included the difference is very statistically significant ($F(5, 4146) = 66.3511, p = 0.0000$). Thus, concluding that C4.5 and Back propagation perform significantly lower than the top four algorithms for model 2. For sensitivity, naïve Bayes was statistically significantly higher than the nearest performing algorithm, Logistic Regression (using a Student's *t*-test, $p = 0.0005$). The results are presented in Table 9.

Using the updated models, and the main large-scale study dataset (Tables 7 and 9), this work concludes that naïve Bayes is still the most suitable machine learning algorithm for PreSS. Other algorithms with similar levels of accuracy and sensitivity could be used if preferred. In addition, for model 2, Back

Table 9. Performance of machine learning algorithms for Model 2, n = 692.

Algorithm	Acc %	Sen %	Spec %
Naïve Bayes	69	80	54
Backpropagation	69	74	62
SVM	68	77	55
Logistic regression	68	78	55
C4.5	67	70	62
K-nearest neighbour	59	63	54

propagation (a single layer artificial neural network) performed as well asnaïve Bayes (accuracy), where Section 8.2 will investigate artificial neural networks further.

8.2. Artificial neural networks

8.2.1. Environment and hyper-parameter tuning

An investigation of the use of artificial neural networks (ANN) to predict introductory programming performance is presented in this section. The machine learning library that was used to develop the neural networks was TensorFlow (Abadi et al., 2015), using the Python programming environment (Rossum, 1995). The Keras high-level neural networks API (Chollet et al., 2015), was also used on top of TensorFlow, the standard in applied deep learning frameworks (Brownlee, 2016).

ANN's are reasonably simplistic to write in code (for example in Python with the use of API's and libraries, such as Keras and TensorFlow), but the difficulty arises when ensuring strong performance while being able to stand over the generalizability of the final model. An approach has been developed in this body of work for developing educational data mining (EDM) ANN, which usually consists of a significantly lower amount of instances and attributes (compared, for example to image recognition). Thus, steps are included, which are not common practice in the deep learning community due to this computation constraint but may add value to the EDM model. As work in this space is only commencing and there is limited literature, especially in computer science education, it is important to establish a white box approach for repeatability and generalizability. The approach is as follows:

- First, set out the parameters that the network may use. This is in four forms: optimizers, network initializers, the number of epochs and finally the batch size. Use pre-defined acknowledged initializers for initial weight selection, and optimizers as that reduces the complexity in selecting learning rate, momentum and decay rates.
- Second, perform a grid search using the above parameters. This is very time consuming, thus the network parameters should be carefully chosen.

The grid search is performed using the entire dataset. The results of the grid search return an optimum set of parameters for the ANN.

- The third part of the approach is not usually conducted in deep learning experiments due to the learning computation associated with significantly large datasets. Ten fold-cross validation should be applied, which is the gold standard for performance measurements on traditional machine learning models (Witten et al., 2016).
- To add to the generalizability of the model, drop-out regularization will be applied to both the visible and to the hidden layers of the network. This is the fourth step of developing the EDM ANN.

For the following experiments, the ANN parameters are presented in Table 10. The optimizers selected were based on their default parameters, such as learning rate and momentum, provided by Keras (Chollet et al., 2015; Kingma & Ba, 2014; Tieleman & Hinton, 2012). The initializers were selected based on distribution: normal and uniform. The epoch selection was based on applied practices (Brownlee, 2016). The batch sizes were selected from best practice (Brownlee, 2016), but as the instance size was comparably small (compared to image recognition), this batch size option was also included in the grid search parameters ($n = 692$).

Once the grid search was completed, the strongest performing network was selected and drop-out regularization was added (20% Brownlee (2017)), to each layer of the network. Then, 10 fold-cross validation was used to obtain an accuracy for the model.

8.2.2. Network topologies

There is a large amount of topologies to select from, thus for this initial investigation, three fundamental network configurations were selected (Brownlee, 2016). This sought to implement a blend of topologies to determine their effectiveness at predicting performance in CS1. A simple single layer ANN, a deep ANN and a convolutional ANN were selected. To present the topologies in detail, the Keras environment allowed for the topologies to be visualized and these are presented in Figure 2. In addition, the input and output dimensions are presented for each layer. This is again to avoid the black box paradigm that is prevalent in machine learning and artificial intelligence studies. The network topologies selected, is by no means exhaustive and represents an initial starting point to examine if performance gains are plausible using ANN's.

Table 10. ANN grid search parameters.

Parameter	Details
Optimizers	rmsprop & adam & stochastic gradient descent
Initializers	normal & uniform
Epochs	[10, 50, 100, 150, 500, 1000]
Batch Size	[5, 10, 20, 50, 100, 150, 250, 500, 692]

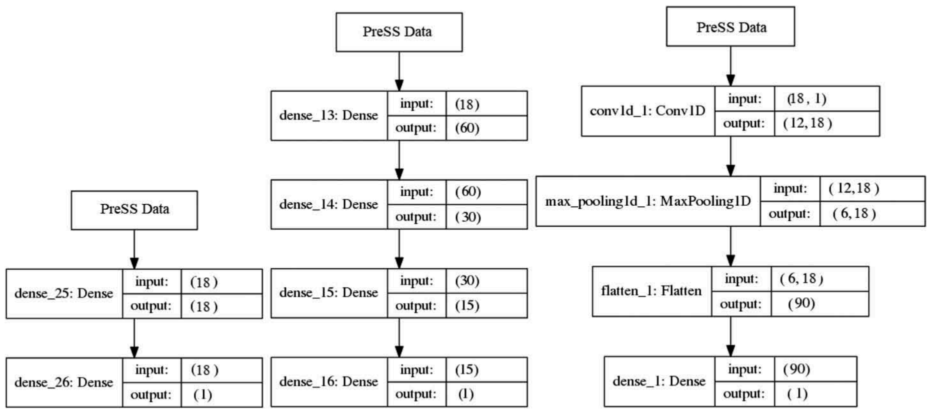

Figure 2. Network topology: single layer ANN, deep learning ANN and convolutional ANN.

Table 11. Performance of deep learning ANN's, where all 17 factors from the main study were inputs to the networks ($n = 692$), using 20% drop-out regularization and 10 fold cross-validation to obtain performance metrics.

Algorithm	Acc %	Sen %	Spec %
Single layer ANN	67	80	51
Deep Learning Fully Connected ANN	69	83	51
Deep Learning Convolutional ANN	66	87	38

8.2.3. Results

Tables 7, 9 and 11 reveal that the accuracies of the top performing algorithms are very similar (with the exception of naïve Bayes in Table 7, where the difference in naïve Bayes performance compared to the other algorithms was statistically significantly higher). However, the deep neural network and the CNN both achieve sensitivity levels that are statistically higher than the other algorithms. As the main goal of PreSS is to predict students at risk of failing, both may provide performance gains over the previous machine learning models. Future research should involve a deeper grid search with multiple network topologies and perhaps custom optimizers with learning rate schedules.

9. Developing interventions: RG-5

This section describes the development of two interventions (RG-5) based on a body of previously related research (Bergin and Reilly (2005, 2006); Quille and Bergin (2015, 2016b, 2018); Quille et al. (2017)). The main focus of both interventions was to positively influence the main predictor of success, programming self-efficacy, in the hope of improving programming performance.

9.1. Scratch alongside CS1

This intervention was developed using a study that was conducted in the 2015–2016 academic year while examining three previous years of data (Quille and Bergin (2016a)). The study investigated when students learnt Scratch, at the same time as their introductory programming module, would their programming self-efficacy increase. Scratch was chosen as students do not need to learn code syntax, rather it is a programming by discovery language. Arguably it may help struggling novice programmers to comprehend coding concepts (even threshold concepts) that they have not grasped in their mainstream text-based language.

9.1.1. Methodology

The Scratch module was delivered in parallel to the staple introductory CS1 programming module. Programming in Scratch consists of dragging and snapping blocks of code together to construct a program. The blocks are predefined and available through a sorted visual display thus allowing a programmer with no previous knowledge to explore and find and code block required. This form of interaction alleviates syntax and compilation errors or the requirement to know the code before a programmer may begin to build. The additional Scratch module is in addition to the hours allocated to the CS1 module.

9.1.2. Results and conclusions

The study first compared student programming self-efficacy and performance of the previous cohorts to the intervention cohort. This was to examine if any underlying population difference may account for any performance variance identified (if any). The difference was not statistically significant between the previous cohorts as determined by one-way ANOVA for programming self-efficacy ($F(1, 58) = 2.0506, p = 0.1575$) or for programming performance ($F(2, 80) = 0.1716, p = 0.8426$).

Next, a one-way ANOVA analysis was conducted on all four-year groups, to examine if the intervention had a positive affect on programming self-efficacy or performance. The difference was not statistically significant between all of the groups for programming self-efficacy ($F(2, 85) = 2.3914, p = 0.0976$) or for programming performance ($F(3, 109) = 0.48516, p = 0.6932$).

9.1.3. Scratch intervention conclusions

The intervention initially concluded that Scratch delivered in parallel to CS1 did not increase student performance. Upon further investigation, a very significant finding showed a substantial variance in the average module pass rates between the first three-year groups and the final year group which was examined in this study. The average module pass rate is calculated across all modules delivered in the academic year (ten in 2015–2016 which included the additional Scratch module and nine in the three previous years 2013 2015 where Scratch was not included). The average module pass rates for each year group is presented in Table 12. The difference was not statistically significant between the previous cohorts (pre-intervention) as determined by one-way ANOVA ($F(2, 80) = 2.7884, p = 0.0675$). Further ANOVA analysis between all cohorts (including the intervention) indicated a significant difference in the average module pass rates with the 2015–16 group compared to the previous cohorts ($F(3, 109) = 22.8345, p = 0.0000$). A Tukey HSD post-hoc test confirmed that for the intervention groups' performance (2015–16), the difference was statistically significant compared to the three pre-intervention group's ($p = 0.0000$ in each case). This suggests that the 2015–2016 student cohort was overall, significantly weaker than the previous three cohorts. Given that, the CS1 performance results were statistically similar to that of the previous 3 years, it could reasonably be hypothesised that the comparable performance on CS1 was due to the additional Scratch module. Further research is warranted to evaluate the efficacy of this intervention.

9.2. Promoting a growth mindset

This section describes an intervention conducted in the academic year of 2016–2017. The intervention was based on the work of Dweck, to promote a growth mindset in an effort to increase performance in introductory programming courses. As described programming self-efficacy is the most significant (positive) factor in predicting performance, this study investigates if self-efficacy and therefore performance can be improved by promoting a growth mindset. This study also considers performance data from a - previous year (as a pre-intervention group) to compare results.

9.2.1. Data collection

During the academic year 2016–17, two institutions participated in this study (with a total *n* = *46* participants). Both institutions also participated in a study outlined in Section 6, which allowed for the comparison with a previous student population (with no intervention) to examine the effectiveness of the intervention. The two populations were also compared (pre-intervention) to investigate if any differences existed that may account for variance (if any) in the affect of the intervention. The two institutions consisted of a community college and an institute of technology. A mindset survey adopted from the work of Dweck was

Table 12. Average overall module pass rates from each year group.

	2012–13	2013–14	2014–15	2015–16
N	24	31	28	30
Average pass rate over all modules in the course	81.51%	77.83%	70.45%	46.37%

used (Dweck (2008)), by D'Anca (D'Anca (2017)). The only difference to the previous studies (other than the mindset survey), was that the surveys were conducted at three stages through out the academic year. Initially before the intervention was deployed (stage one, at approximately 10% into the delivery of CS1), at the end of CS1 (stage two, in semester 1 before the examinations) and at the end of the academic year (stage three, at the end of CS2 in semester 2, before the examinations). This is useful to track changes in attributes such as mindset and programming self-efficacy over the entire academic year and is planned for future research, as this intervention is ongoing.

9.2.2. Methodology

The intervention was applied, from stage one to stage two. This consisted of several approaches to promote a growth mindset. The methodology was developed from previous work (Cutts, Cutts, Draper, O'Donnell, and Saffrey (2010); Dweck (2008); Lovell (2014); Murphy and Thomas (2008)). The approaches fall under three headings:

Lecture: This was delivered at the start of each session (4 hour session) and lasted generally for five to ten minutes. This approach promoted the fundamentals of growth mindsets, and it's a success story, with respect to increases in performance in other domains (such as kindergarten and at the second level). The lecturer used their own personal experiences and relayed the correlation between work ethic (grit) and attainment of ability. The lecturer also presented testimonials from students who had completed the course, especially students who initially struggled. This was conducted for all 12 weeks from stage one to stage two.

Research: Research was presented formally (approximately at each quarter of the course delivery), from that of Dweck, the related literature and neuroscience. This was to identify measurable changes that growth mindset has, and the successful outcomes it has produced. For example in brain activity, where time on task showed an increase in activity, thus the brain was changing to adopt to the new ability.

Feedback: Feedback was delivered regularly during the programming labs, but also formally after assessment. The main goal of the feedback was to praise the process, not the person. In addition, if the feedback was on a poor result, it was delivered in a constructive manor identifying the processes that the student needed to do, to achieve a stronger result.

9.2.3. Student background

Before the intervention data results were analysed, the previous cohort of students from 2015–16 were compared to the intervention cohort. This was conducted at stage one before the intervention was applied. This was to examine if any affects of the intervention could be attributed to underlying population differences, for example a particular group may have had higher initial self-efficacy. Both institutions were delivering the same course, teacher and syllabus both years, with no additional interventions or teaching methodologies applied. The results reported that other than gender balance (*2015–16 = 4% female students* (*n* $= 2$) *compared to this study which had 13% female representation* (*n* $= 6$)), no statistically significant differences existed between the two cohorts for any measured attribute.

9.2.4. Results and conclusions

The 2015–16 cohort had an average CS1 grade of 66.71%, where the intervention group (2016–17), reported an average grade in CS1 of 75.39%. There was a statistically significant difference between groups as determined by a Welch's *t*-test ($p = 0.0194$). This increase in performance was also significant, given the only underlying population difference, was the increase in the ratio of female students in the intervention cohort. Given the very small number of female students ($n = 2$ and $n = 6$ respectively) it is reasonable to suggest that the significant differences in performance is not due to the gender balance increase, but perhaps due to the intervention. Future work will involve an investigation on different sub cohorts (such as gender and on performance) and re-running the study on a significantly larger cohort.

10. Conclusions

Over a decade after its conception, validation, and presentation at SIGCSE'05, PreSS was once again been put under the microscope. It is believed that seldom before has a prediction model been subjected to such longitudinal rigour and developmental processes. Thus, making a valuable contribution to the CSEd community and hopefully encouraging future revalidation and replication studies. PreSS is able to predict with an accuracy of $\approx$ 70%, 13 years after it was first developed and validated. More importantly, PreSS was able to identify weak students (the main goal of PreSS) with a sensitivity of 80–89%. The 13 years of research and development have examined factors and algorithms ensuring the model is generalizable while answering a call from the ITiCSE 2015 working group (Ihantola et al., 2015). In addition to the PreSS computational model, PreSS is now integrated into an online toolbox ready to use. With PreSS# fully developed and deployed, CSEd educators, can start using this tool, to try and address attrition rates in introductory programming courses.

The objective of this article is to provide an overview (along with additional novel research) to CSEd educators and researchers, in the shape of the complete journey thus far. This not only links the research together, but presents follow up research (revalidation, research improvements) allowing researchers to examine the complete suite of tools and research, to enable them to use the tools in the classroom, and inform the foundations of future research in this space. It is our hope that by bringing this large body of work together, it will encourage others to do the same, accelerating the progression of future research in this space. We welcome opportunities to share and corroborate with interested researchers as we continue this work.

Disclosure statement

No potential conflict of interest was reported by the authors.

ORCID

Keith Quille http://orcid.org/0000-0002-1414-5142

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. Retrieved from https://www.tensorflow.org/(Software available from tensorflow.org)

Ahadi, A., Lister, R., Haapala, H., & Vihavainen, A. (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* (pp. 121–130). New York, NY: ACM. doi: 10.1145/2787622.2787717

Allert, J. (2004). Learning style and factors contributing to success in an introductory computer science course. *IEEE International Conference on Advanced Learning Technologies 2004 Proceedings*, *33*(1), 184–188. Retrieved from http://portal.acm.org/citation.cfm?id=364581

Barker, L. J., Mcdowell, C., & Kalahar, K. (2009). Exploring factors that influence computer science introductory course students to persist in the major. *ACM SIGCSE Bulletin*, *41*(2), 282–286.

Barker, R. J., & Unger, E. A. (1983). A predictor for success in an introductory programming class based upon abstract reasoning development. In *Proceedings of the fourteenth sigcse technical symposium on computer science education* (pp. 154–158). New York, NY: ACM. doi: 10.1145/800038.801037

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* (pp. 338–344). New York, NY: ACM. doi: 10.1145/3287324.3287432

Bennedsen, J., & Caspersen, M. E. (2005). An investigation of potential success factors for an introductory model-driven programming course. *Proceedings of the First International Workshop on Computing Education Research*, 155–163. doi: 10.1145/1089786.1089801

Bennedsen, J., & Caspersen, M. E. (2006). Abstraction ability as an indicator of success for learning object-oriented programming? *SIGCSE Bulletin*, *38*(2), 39–43.

Bennedsen, J., & Caspersen, M. E. (2008). Optimists have more fun, but do they learn better? On the influence of emotional and social factors on learning introductory computer science. *Computer Science Education*, *18*(1), 1–16.

Bergin, S. (2006). *A computational model to predict programming performance* (Unpublished doctoral dissertation). Department of Computer Science, Maynooth University.

Bergin, S., Mooney, A., Ghent, J., & Quille, K. (2015a). Using machine learning techniques to predict introductory programming performance. *International Journal of Computer Science and Software Engineering*, *4*(12), 323–328.

Bergin, S., Mooney, A., Ghent, J., & Quille, K. (2015b). Using machine learning techniques to predict introductory programming performance. *International Journal of Computer Science and Software Engineering*, *4*(12), 323–328.

Bergin, S., & Reilly, R. (2005, February). Programming: Factors that influence success. *SIGCSE Bulletin*, *37*(1), 411–415.

Bergin, S., & Reilly, R. (2006). Predicting introductory programming performance: A multi-institutional multivariate study. *Computer Science Education*, *16*(4), 303–323.

Boetticher, G. D., Ding, W., Moen, C., & Yue, K.-B. (2005). Using a pre- assessment exam to construct an effective concept-based genetic program for predicting course success. In *Proceedings of the 36th sigcse technical symposium on computer science education* (pp. 500–504). New York, NY, USA: ACM. doi: 10.1145/1047344.1047503

Bornat, R., Dehnadi, S., & Simon. (2008). Mental models, consistency and programming aptitude. In *Proceedings of the tenth conference on australasian computing education - volume 78* (pp. 53–61). Darlinghurst, Australia: Australian Computer Society, Inc. Retrieved from http://dl.acm.org/citation.cfm?id=1379249.1379253

Brownlee, J. (2016). *Deep learning with python: Develop deep learning models on theano and tensorflow using keras*. Melbourne: Machine Learning Mastery.

Brownlee, J. (2017). Deep learning with python. Retrieved from *https://machinelearningmastery.com/deep-learning-with-python/*.

Butcher, D. F., & Muth, W. A. (1985, March). Predicting performance in an introductory computer science course. *Communications ACM*, *28*(3), 263–268.

Campbell, J., Horton, D., & Craig, M. (2016). Factors for success in online cs1. In *Proceedings of the 2016 acm conference on innovation and technology in computer science education* (pp. 320–325). New York, NY: ACM. doi: 10.1145/2899415.2899457

Capstick, C. K., Gordon, J. D., & Salvadori, A. (1975, September). Predicting performance by university students in introductory computing courses. *SIGCSE Bulletin*, *7*(3), 21–29.

Caspersen, M. E., Larsen, K. D., & Bennedsen, J. (2007). Mental models and programming aptitude. In *Proceedings of the 12th annual sigcse conference on innovation and technology in computer science education* (pp. 206–210). New York, NY, USA: ACM. doi: 10.1145/1268784.1268845

Chollet, F., et al. (2015). Keras. Retrieved from https://keras.io.

Conroy, R. (2016). Sample Size: Introduction 1 Sample size A rough guide How to use this guide (Tech. Rep.). Retrieved from https://beaumontethics.ie/docs/application/samplesizecalculation.pdf

Cukierman, D. (2015). Predicting success in university first year computing science courses: The role of student participation in reflective learning activities and in i-clicker activities. In *Proceedings of the 2015 acm conference on innovation and technology in computer science education* (pp. 248–253). New York, NY: ACM. doi: 10.1145/2729094.2742623

Culligan, N., Quille, K., & Bergin, S. (2016). Veap: A visualisation engine and analyzer for press#. In *Proceedings of the 16th koli calling international conference on computing education research* (pp. 130–134). New York, NY: ACM. doi: 10.1145/2999541.2999553

Cutts, Q., Cutts, E., Draper, S., O'Donnell, P., & Saffrey, P. (2010). Manipulating mindset to positively influence introductory programming performance. *Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE '10*, 431. Retrieved from http://portal.acm.org/citation.cfm?doid=1734263.1734409

D'Anca, J.-A. (2017). *MINDSET AND RESILIENCE: AN ANALYSIS AND INTERVENTION FOR SCHOOL ADMINISTRATORS* (Unpublished doctoral dissertation).

Dehnadi, S. (2006). Testing programming aptitude. In *Proceedings of the 18th annual workshop of the psychology of programming interest group* (pp. 22–37).

Denny, P., Luxton-Reilly, A., Hamer, J., Dahlstrom, D. B., & Purchase, H. C. (2010). Self-predicted and actual performance in an introductory programming course. In *Proceedings of the fifteenth annual conference on innovation and technology in computer science education* (pp. 118–122). New York, NY: ACM. doi: 10.1145/1822090.1822124

Dweck, C. S. (2008). *Mindset: The new psychology of success*. Random House Digital, Inc.

Estey, A., & Coady, Y. (2016). Can interaction patterns with supplemental study tools predict outcomes in cs1? In *Proceedings of the 2016 acm conference on innovation and technology in computer science education* (pp. 236–241). New York, NY, USA: ACM. doi: 10.1145/2899415.2899428

Evans, G. E., & Simkin, M. G. (1989). What best predicts computer proficiency? *Communications of the ACM*, *32*(11), 1322–1327. Retrieved from http://portal.acm.org/citation.cfm?doid=68814.68817

Fowler, G. C., & Glorfeld, L. W. (1981). Predicting aptitude in introductory computing: A classification model. *AEDS Journal*, *14*(2), 96–109.

Glorfeld, L. W., & Fowler, G. C. (1982). Validation of a model for predicting aptitude for introductory computing. In *Proceedings of the thirteenth sigcse technical symposium on computer science education* (pp. 140–143). New York, NY: ACM. doi: 10.1145/800066.801355

Golding, P., Facey-Shaw, L., & Tennant, V. (2006). Effects of peer tutoring, attitude and personality on academic performance of first year introductory programming students. *Proceedings - Frontiers in Education Conference, FIE*, 7–12.

Hostetler, T. R. (1983, September). Predicting student success in an introductory programming course. *SIGCSE Bulletin*, *15*(3), 40–43.

Ihantola, P., Vihavainen, A., Ahadi, A., Butler, M., Börstler, J., Edwards, S. H., … Toll, D. (2015). Educational data mining and learning analytics in programming: Literature review and case studies. In *Proceedings of the 2015 iticse on working group reports* (pp. 41–63). New York, NY: ACM. doi: 10.1145/2858796.2858798

Kelly, J. O., Mooney, A., Ghent, J., Gaughran, P., Dunne, S., & Bergin, S. (2004). An overview of the integration of problem based learning into an existing computer science programming module 2 overview of our PBL implementation. *Problem-Based Learning International Conference 2004: Pleasure by Learning*.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization, 1–15. Retrieved from http://arxiv.org/abs/1412.6980

Konvalina, J., Wileman, S. A., & Stephens, L. J. (1983, May). Math proficiency: A key to success for computer science students. *Communications ACM*, *26*(5), 377–382.

Lambert, L. (2015, December). Factors that predict success in cs1. *Journal of Computing Sciences in Colleges*, *31*(2), 165–171. Retrieved from http://dl.acm.org/citation.cfm?id=2831432.2831458

Leeper, R. R., & Silver, J. L. (1982). Predicting success in a first programming course. In *Proceedings of the thirteenth sigcse technical symposium on computer science education* (pp. 147–150). New York, NY: ACM. doi: 10.1145/800066.801357

Leinonen, J., Leppänen, L., Ihantola, P., & Hellas, A. (2017). Comparison of time metrics in programming. In *Proceedings of the 2017 acm conference on international computing education research* (pp. 200–208). New York, NY: ACM. doi: 10.1145/3105726.3106181

Liao, S. N., Zingaro, D., Laurenzano, M. A., Griswold, W. G., & Porter, L. (2016). Lightweight, early identification of at-risk cs1 students. In *Proceedings of the 2016 acm conference on international computing education research* (pp. 123–131). New York, NY: ACM. doi: 10.1145/2960310.2960315

Lishinski, A., Yadav, A., Enbody, R., & Good, J. (2016). The influence of problem solving abilities on students' performance on different assessment tasks in cs1. In *Proceedings of the 47th acm technical symposium on computing science education* (pp. 329–334). New York, NY: ACM. doi: 10.1145/2839509.2844596

Lishinski, A., Yadav, A., Good, J., & Enbody, R. (2016). Learning to program: Gender differences and interactive effects of students ' motivation, goals and self-efficacy on performance. *Proceedings of the 12th International Computing Education Research Conference*, 211–220.

Liston, M., Frawley, D., & Patterson, V. (2016). *A study of progression in irish higher education*. Higher Education Authority. Retrieved from http://www.hea.ie/en/publications/2016

Lovell, E. (2014). Promoting constructive mindsets for overcoming failure in computer science education. *Proceedings of the tenth annual conference on International computing education research - ICER '14*, 159–160. Retrieved from http://dl.acm.org/citation.cfm?doid=2632320.2632331

Mooney, O., Patterson, V., OConnor, M., & Chantler, A. (2010). *A study of progression in irish higher education*. Dublin: Higher Education Authority.

Murphy, L., & Thomas, L. (2008). Dangers of a fixed mindset: Implications of self-theories research for computer science education. *ACM SIGCSE Bulletin*, (3), 271–275.

Naing, L., Winn, T., & Rusli, B. N. (2006). Practical issues in calculating the sample size for prevalence studies. *Archives of Orofacial Sciences*, *1*(Ci), 9–14.

Newsted, P. R. (1975). Grade and ability predictions in an introductory programming course. *ACM SIGCSE Bulletin*, *7*, 87–91.

Nolan, K., & Bergin, S. (2016). The role of anxiety when learning to program: A Systematic review of the literature. In *Proceedings of the 16th koli calling international conference on computing education research* (pp. 61–70). Koli, Finland: ACM.

Pioro, B. T. (2004). Estimated and actual performance scores on computer programming tasks. *SoutheastCon, 2004. Proceedings. IEEE*, 3–7.

Porter, L., & Zingaro, D. (2014). Importance of early performance in CS1: Two conflicting assessment stories. *Sigcse '14*, 295–300.

Porter, L., Zingaro, D., & Lister, R. (2014). Predicting student success using fine grain clicker data. In *Proceedings of the tenth annual conference on international computing education research* (pp. 51–58). New York, NY, USA: ACM. doi: 10.1145/2632320.2632354

Quille, K., & Bergin, S. (2015). Programming: Factors that influence success revisited and expanded. In *Proceedings of the international conference on enguaging pedagogy (icep), 3rd and 4th december, college of computing technology*, dublin, ireland.

Quille, K., & Bergin, S. (2016a). Does Scratch improve self-efficacy and performance when learning to program in C#? An empirical study. In *Proceedings of the international conference on enguaging pedagogy (icep), maynooth university*, Maynooth, ireland.

Quille, K., & Bergin, S. (2016b). Programming: Further factors that influence success. In *Proceedings of the psychology of programming interest group (ppig), 7th to 10th spetember*, University of Cambridge, United Kingdom.

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'18)*. New York, NY, USA: ACM. doi:10.1145/3197091

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'18)*. New York, NY: ACM.

Quille, K., Bergin, S., & Mooney, A. (2015). Press#, a web-based educational system to predict programming performance. *International Journal of Computer Science and Software Engineering (IJCSSE)*, *4*(7), 178–189. Retrieved from http://eprints.maynoothuniversity.ie/6503/

Quille, K., Culligan, N., & Bergin, S. (2017). Insights on gender differences in cs1: A multi-institutional, multi-variate study. In *Proceedings of the 2017 acm conference on innovation and technology in computer science education* (pp. 263–268). New York, NY: ACM. doi: 10.1145/3059009.3059048

Quille, K., Faherty, R., Bergin, S., & Becker, B. A. (2018). Second Level Computer Science: The Irish K-12 Journey Begins. In *Proceedings of the 18th koli calling international conference on computing education research (koli calling '18)*. Koli, Finland ACM.

Rossum, G. (1995). *Python reference manual (Tech. Rep.)*. The Netherlands: Amsterdam.

Rountree, N., Rountree, J., & Robins, A. (2002, December). Predictors of success and failure in a cs1 course. *SIGCSE Bulletin*, *34*(4), 121–124.

Rountree, N., Rountree, J., Robins, A., & Hannah, R. (2004). Interacting factors that predict success and failure in a CS1 course. *ACM SIGCSE Bulletin*, *36*(4), 101.

Shell, D. F., Soh, L.-K., Flanigan, A. E., & Peteranetz, M. S. (2016). Students' initial course motivation and their achievement and retention in college cs1 courses. In *Proceedings of the 47th acm technical symposium on computing science education* (pp. 639–644). New York, NY: ACM doi: 10.1145/2839509.2844606

Simon, F., Robins, S., Baker, A., Box, B., Cutts, I., & Tutty, J. (2006). Predictors of success in a first programming course. *Proceedings of the 8th Austalian conference on Computing education - Volume 52*, 189–196.

Tarimo, W. T., Deeb, F. A., & Hickey, T. J. (2016, June). Early detection of at-risk students in cs1 using teachback/spinoza. *Journal of Computing Sciences in Colleges*, *31*(6), 105–111. Retrieved from http://dl.acm.org/citation.cfm?id=2904446.2904471

Tieleman, T., & Hinton, G. (2012). *Lecture 6.5 - rmsprop, coursera: Neural networks for machine learning*.

Traynor, D., Bergin, S., & Gibson, J. P. (2006). *Automated Assessment in CS1* (Ace '06), 223–228.

Ventura, P. R. (2005). Identifying predictors of success for an objects- first CS1. *Computer Science Education*, *15*(3), 223–243.

Vihavainen, A. (2013). Predicting students' performance in an introductory programming course using data from students' own programming process. In *Proceedings of the 2013 IEEE 13th international conference on advanced learning technologies* (pp. 498–499). Washington, DC: IEEE Computer Society. doi: 10.1109/ICALT.2013.161

Watson, C., Li, F. W. B., & Godwin, J. L. (2013). Predicting performance in an introductory programming course by logging and analyzing student programming behavior. In *Proceedings of the 2013 IEEE 13th international conference on advanced learning technologies* (pp. 319–323). Washington, DC, USA: IEEE Computer Society. doi: 10.1109/ICALT.2013.99

Werth, L. (1986). Predicting student performance in a beginning computer science class. *SIGCSE '86 Proceedings of the Seventeenth SIGCSE Technical Symposium on Computer Science Education*, 138–143.

Wiedenbeck, S. (2007). Antecedents to end users' success in learning to program in an introductory programming course. *Proceedings - IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2007*, 163–170.

Wiig, D. M. (1989, February). A bayesian probability approach to predicting student performance in introductory computer science courses. *SIGSMALL/PC Notes*, *15*(1), 3–19.

Wileman, S., Konvalina, J., & Stephens, L. (1981). Factors influencing success in beginning computer science courses. *The Journal of Educational Research*, (March), 223–226.

Wilson, B. C., & Shrock, S. (2001). Contributing to success in an introductory computer science course: A study of twelve factors. *ACM SIGCSE Bulletin*, (1), 184–188.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data mining: Practical machine learning tools and techniques*. Amsterdam: Morgan Kaufmann.