# Predicting and Improving Performance on Introductory Programming Courses (CS1)

KEITH QUILLE

A thesis submitted for the degree of

*Doctor of Philosophy*

Department of Computer Science

Maynooth University

December 2019

Supervisor: Dr. Susan Bergin

Head of Department: Prof. Joseph Timoney

# CONTENTS

# LIST OF FIGURES

## LIST OF TABLES

# ABSTRACT

This thesis describes a longitudinal study on factors which predict academic success in introductory programming at undergraduate level, including the development of these factors into a fully automated web-based system (which predicts students who are at risk of not succeeding early in the introductory programming module) and interventions to address attrition rates on introductory programming courses (CS1). Numerous studies have developed models for predicting success in CS1, however there is little evidence on their ability to generalise or on their use beyond early investigations. In addition, they are seldom followed up with interventions, after struggling students have been identified. The approach overcomes this by providing a web-based real time system, with a prediction model at its core that has been longitudinally developed and revalidated, with recommendations for interventions which educators could implement to support struggling students that have been identified.

This thesis makes five fundamental contributions. The first is a revalidation of a prediction model named PreSS. The second contribution is the development of a web-based, real time implementation of the PreSS model, named PreSS#. The third contribution is a large longitudinal, multi-variate, multi-institutional study identifying predictors of performance and analysing machine learning techniques (including deep learning and convolutional neural networks) to further develop the PreSS model. This resulted in a prediction model with approximately 71% accuracy, and over 80% sensitivity, using data from 11 institutions with a sample size of 692 students. The fourth contribution is a study on insights on gender differences in CS1; identifying psychological, background, and performance differences between male and female students to better inform the prediction model and the interventions. The final, fifth contribution, is the development of two interventions that can be implemented early in CS1, once identified by PreSS# to potentially improve student outcomes.

The work described in this thesis builds substantially on earlier work, providing valid and reliable insights on gender differences, potential interventions to improve performance and an unsurpassed, generalizable prediction model, developed into a real time web-based system.

# ACKNOWLEDGMENTS

First and foremost, I would like to thank my amazing wife, Faith. A supportive word from Faith in the car one day, started me on this amazing journey, which I never imagined that I was able to do. To that, Faith's patience, support, encouragement and love was never ending throughout this life changing endeavour, and I honestly would never have got to this point without her. I can never thank you enough Faith... I am finally here because of you.

I would really like to extend my sincere gratitude to my supervisor Dr. Susan Bergin. I can not put into words, how professional and amazing Dr. Bergin was. Her support and encouragement, has been inspirational. If in ten years time, I could be a fraction of the supervisor that Dr. Bergin was, I would be happy. I feel humbled and honoured to have had such a great mentor, and I really hope that we get to work together in the future. I have learned and experienced so much in these past five years, I really sincerely thank you for that. I would also like to thank Dr. Aidan Mooney, who has helped me so much over the five years, with support and advice. I met Susan and Aidan for the first time, as my undergraduate lab demonstrators in 2003 and it has been an amazing journey!

I would also like to thank my family, especially my Mam and Dad, Bernie and Austin. For my entire life they have both supported me (and Faith) so much, always been there to help us out when needed, and afforded me so much, I do not know how I could ever fully thank you, I really do not. You have been amazing, and I will never forget that. I also can not forget my brother, Darren, and my family in law, thank you for all your support and encouragement.

For Faith, Mam and Dad....

# DECLARATION

I confirm that this is my own work and the use of all material from other sources has been properly cited and fully acknowledged.

Keith Quille

December 2019

# 1 INTRODUCTION

## 1.1 THE INTRODUCTORY PROGRAMMING LANDSCAPE

Computer Science (CS) non-progression (from year one into year two) 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% (across all higher education disciplines). 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 level 6 to level 8 [77, 82]. In addition, CS is one of only two fields of study, where the non-progression rate has increased since the first report in 2010. It is well acknowledged that a main contributor, is that students struggle to succeed in their initial programming module (CS1), a staple in most first year CS courses. Several studies have investigated CS1 failure rates in particular, reporting 33% [14] and 32.3% [140] failure rates. While these are higher than the overall CS attrition rates in Ireland, Watson noted that pass rates may not be trivial to measure, for example some institutions in their study required a C grade to pass, thus perhaps accounting for the difference.

Early identification of students who are at risk of non-progression is often hindered by the very high student-lecturer ratio (100:1 or greater). Lecturers may not be aware that students are struggling until a considerable time has passed and early problematic threshold concepts have been encountered. Numerous approaches have been trialled with varying degrees of success to improve learning and assessment outcomes. At our institution, numerous initiatives have been implemented to improve outcomes, including the development of automated adaptive assessment systems [133], novel teaching approaches such as problem based learning [64] and more recently a Programming Support Centre run by peer tutors [89]. As computer programming is not currently a traditional Leaving Certificate subject, there are no formal indicators of a student's 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). A number of models exist 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. One of the challenges asserts, that while there are several studies in the literature that predict performance or identify students at risk of not progressing, the studies are seldom revalidated. In addition the models have not been employed in actual interventions where if the findings were positive or of value, they could then be put into practice. The working group also highlighted as a separate grand challenge, the critical need for re-validation of educational data mining models. [62].

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.

## 1.2 THE STARTING POINT - PRESS

### 1.2.1 WHY PRESS?

Over twelve years ago, a detailed study on factors that influence success in CS1 was presented by Bergin at SIGCSE 05, in St. Louis, Missouri, USA [17]. Subsequently this led on to the development of a computational model that could predict student success with 77.5% accuracy (at a very early stage) in a CS1 module, named PreSS (Predict Student Success) [16, 18].

PreSS was developed between 2003 and 2006. It was composed of three studies, one in each year. Multiple institutions at various tertiary levels took part. Several investigations on factors that influence success and on the development of machine learning models to predict performance, were carried out with best practice techniques to improve generalisation, (for example: data stratification, 10 fold cross validation and performance measures). The PreSS model is described in detail in Chapter 2. 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 [17].

PreSS has since been successfully used locally, but the model required paper-based data collection with manual processing and computation. This time consuming process inhibited the uptake and usage of the model on a large scale. A detailed review of other comparable models and related literature on factors that influence success when learning to program, is provided in chapter 2. Given the high performance of PreSS over several studies (with different student cohorts at different tertiary levels) this model was selected for use in this thesis.

Bergin concluded her PhD thesis, with several suggested recommendations for future work. The first recommendation was further research on programming predictors to try to improve the performance of the PreSS model. The second recommendation was to refine the measure of mathematical performance to make it more generalizable to different student cohorts. The third call was for the development of suitable interventions so that once struggling students are identified, timely interventions can be applied. Finally, gender was a strong point of discussion throughout the thesis, and although the dichotomous factor of gender did not make it into the final model, it was believed that it was of significant value. Bergin recommended that further investigation into gender related factors, may lead to a deeper insight and value when developing PreSS further. Between 2006 and the start of this thesis, little progress on these recommendations was made. This thesis describes a substantial body of work to address each recommendation. The goals are described in detail in Section 1.3.1.

### 1.2.2 REFERENCE TERMS

As this work will be referenced throughout this thesis, to avoid any confusion as to what work is being discussed, the following terms will be applied:

**Original PreSS Study** : This refers to the PhD thesis of Bergin, and the development of the PreSS model within.

**Original PreSS Model** : This refers to the final prediction model as discussed in Bergin's PhD thesis.

## 1.3 INTRODUCTION TO THIS RESEARCH

### 1.3.1 RESEARCH GOALS

The research goals (RG) of this thesis are to:

1. *Investigate if the original PreSS model, is still a valid prediction model a decade after it was initially developed*
2. *Develop a web-based real time implementation of the original PreSS model*
3. *Investigate if the original PreSS model can be improved upon using:*
   a) *New factors for the model*
   b) *Alternative machine learning algorithms for the model*
4. *Investigate insights on gender differences in CS1 to further inform the PreSS model and interventions*
5. *Develop and investigate interventions that could reduce attrition rates in CS1*

### 1.3.2 ETHICAL APPROVAL

For all studies conducted in this thesis, ethical approval was sought and granted by Maynooth University. Due to the nature of the data collected (grades, psychometric, demographic), student and institution anonymity was essential. For both ethical and participation reasons (as many institutions cited this as a concern), the study was not a comparison of specific institutions. All data was stored securely, anonymised, using keys and only the person's named on the ethical approval were allowed access to the data.

### 1.3.3 thesis contributions

This thesis makes several fundamental contributions to these objectives outlined in the research goals, as follows:

1. The revalidation of the original PreSS model using a large multi-institutional data set.
2. The development of a web based real time system with PreSS at its core.
3. The further development of PreSS with the identification of several new factors and machine learning algorithms.
4. The identification of several insights into gender differences in CS.
5. The development of two interventions for reducing attrition rates in CS1.

Each of these contributions are discussed in the following section:

*RG1: Revalidation of PreSS*

The revalidation of PreSS was conducted by an initial justification study (and following its positive findings), followed by a large study, referred to in this thesis as the main study. Both are summarized in this section.

**Justification Study:** Initially a small justification study was carried out to examine if PreSS was still an accurate predictor of programming success [97]. The study used two small independent data sets, collected in the academic years 2013-14 and 2014-15. The justification study reported that PreSS was able to predict with a statistically similar accuracy the original Press work that took place over a decade previously. [97]. Although these results were very promising, given the small sample size and the fact that it only represented a single institution, a large study was required to re-validate the model and to examine its generalizability. The justification study is discussed in detail in Chapter 3.1.

**Main Study:** A large scale multi-institutional revalidation study was conducted in the academic year 2015-2016. The study and its purpose was multi faceted and in essence it was the core body of work in this thesis. In total, 692 complete student data sets were collected and used in the main large scale study. This contribution provides evidence that the PreSS model was still a valid model several years after it was developed and is detailed in Chapter 3.2.

*RG2: Web Application: PreSS#*

In 2015 the PreSS model was developed into a web-based educational system named PreSS# [102]. PreSS# is able to predict student success in real time. The system is fully automated allowing institutions to create users, run the prediction and examine outputs. To ensure PreSS# was producing comparable results to that of PreSS, the original data set was used for validation. Both PreSS and PreSS# produced the same results, with no significant differences found [102], thus validating the developed on-line system. PreSS# is discussed in detail in Chapter 4.

*RG3: Model Development*

**Investigating Additional Factors:** Research conducted in 2016 investigated additional factors that may increase the accuracy of PreSS. Two data sources were examined: the first consisted of factors gathered during the original PreSS study that were not used in the final model. The second consisted of additional factors collected in the justification study. The research successfully identified 16 factors that when used in combination or substitution with the original PreSS factors either produced significant increases in model performance or were otherwise worthy of note. The newly identified factors were then included as part of the main study and investigated examining if the PreSS model could be further improved using the multi-institutional large scale data set. This piece of work found several factors that improved the PreSS model. This research is presented in Chapter 5.

**Machine Learning Algorithms:** Using the main study data set, an investigation was conducted of multiple machine learning techniques to predict performance to further improve the PreSS. A multitude of algorithms were compared that include: naïve Bayes, logistic regression, support vector machines, decision tree, k-nearest neighbour, single layer artificial neural network, deep learning artificial neural network, and a convolutional artificial neural network. This contribution reports an increase in performance when using artificial neural networks, and in particular deep learning, over the original PreSS algorithm, and is discussed in detail in Chapter 6.

### *RG4: Insights on Gender Differences*

In the original PreSS study, gender was found to have significance when developing the model and it was noted that future work and a deeper investigation may add prediction value to the model. Thus, using the main data set, a comparative study was carried out in 2017, comparing the profile of male and female students enrolled on CS1, to determine if any significant differences could be identified by gender. The findings contributed to both the development of the PreSS model (RG3) and the development of two interventions (RG5). This study is presented in Chapter 7.

*RG5: Interventions*

This section describes the development of two interventions. With the PreSS model revalidated and further developed, the goal then shifted to interventions to help improve student success in CS1. 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.

**Scratch to improve self-efficacy and performance in CS1:** This study investigated when students were exposed to Scratch, a block type programming language, at the same time as their CS1 module, which was delivered using C#, would their programming self-efficacy increase (a prominent factor identified in the original PreSS study and in this thesis). This study is presented in Chapter 8.1.

**Promoting a Growth Mindset, as an Intervention in CS1:** This study was based on the work of Dweck [46], to promote a growth mindset in an effort to increase performance in CS1. This intervention reported a significant increase in performance over the previous control group where no intervention was deployed and is presented in Chapter 8.2.

### 1.3.4 THESIS VISUAL OVERVIEW

This research is broken into five main sections as presented in Figure 1.1. This figure maps the research goals (RG) to the sections and chapters in this thesis and provides a time-line of the research. It also illustrates where the prior research finishes (vertical division).

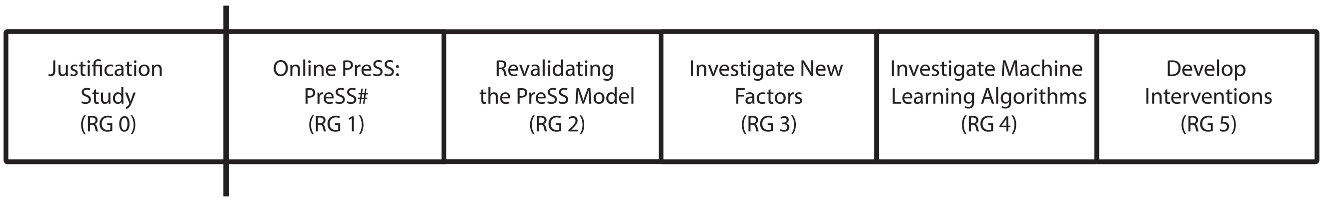

Justification Study (RG 0)
Online PreSS: PreSS# (RG 1)
Revalidating the PreSS Model (RG 2)
Investigate New Factors (RG 3)
Investigate Machine Learning Algorithms (RG 4)
Develop Interventions (RG 5)

Figure 1.1: Visual Overview of Research

As a reference guide, the mappings of RG's to chapters is provided in Table 1.1:

Table 1.1: Reference guide for mapping RG's to chapters.

| **Section** | **Research Goal** | **Chapter(s)** |
|---|---|---|
| Revalidating the PreSS model | 1 | 3 |
| PreSS Web Application (PreSS#) | 2 | 4 |
| Improving the PreSS Model | 3 | 5, 6 |
| Insights on Gender Differences | 4 | 7 |
| Developing Interventions | 5 | 8 |

## 1.4 PUBLICATIONS

The publications that arose as a result of the research contained in this thesis (which are freely accessible at `http://keithquille.com/Publications`) are as follows (peer-reviewed):

[101] Keith Quille and Susan Bergin. "CS1: how will they do? How can we help? A decade of research and practice". In: *Computer Science Education* 29.2-3 (2019), pp. 254–282. DOI: 10.1080/08993408.2019.1612679. URL: https://doi.org/10.1080/08993408.2019.1612679

[104] Keith Quille et al. "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, 2018. DOI: 10.1145/3279720.3279742

[100] Keith Quille and Susan Bergin. "Programming: Predicting Student Success Early in CS1. A Re-validation and Replication Study". In: *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE 2018. Larnaca, Cyprus: ACM, 2018, pp. 15–20. ISBN: 978-1-4503-5707-4. DOI: 10.1145/3197091.3197101. URL: http://doi.acm.org/10.1145/3197091.3197101

[103] Keith Quille, Natalie Culligan, and Susan Bergin. "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*. ITiCSE '17. Bologna, Italy: ACM, 2017, pp. 263–268. ISBN: 978-1-4503-4704-4. DOI: 10.1145/3059009.3059048. URL: http://doi.acm.org/10.1145/3059009.3059048

[98] Keith Quille and Susan Bergin. "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*. 2016. URL: http://www.keithquille.com/PublicationData/ICEP2016.pdf

[99] Keith Quille and Susan Bergin. "Programming: Further Factors that Influence Success". In: *Proceedings of the Psychology of Programming Interest Group (PPIG), 7th to 10th Spetember, University of Cambridge*. 2016. URL: keithquille.com/PublicationData/ppig2016.pdf

[97] K Quille and S Bergin. "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*. 2015. URL: http://www.keithquille.com/PublicationData/ICEP_2015.pdf

[102] Keith Quille, Susan Bergin, and Aidan Mooney. "PreSS#, A Web-Based Educational System to Predict Programming Performance". In: *International Journal of Computer Science and Software Engineering (IJCSSE)* 4.7 (2015), pp. 178–189. URL: http://www.keithquille.com/PublicationData/PreSS.pdf

CONTRIBUTED:

[11] Brett Becker and Keith Quille. "50 Years of CS1 at SIGCSE: A Review of the Evolution of Introductory Programming Education Research". In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. SIGCSE '19. New York, NY, USA: ACM, 2019. DOI: 10.1145/3287324.3287432. URL: http://doi.acm.org/10.1145/3287324.3287432

[35] Natalie Culligan, Keith Quille, and Susan Bergin. "VEAP: A Visualisation Engine and Analyzer for PreSS#". In: *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*. Koli Calling '16. Koli, Finland: ACM, 2016, pp. 130–134. ISBN: 978-1-4503-4770-9. DOI: 10.1145/2999541.2999553. URL: http://doi.acm.org/10.1145/2999541.2999553

[19] Susan Bergin et al. "Using Machine Learning Techniques to Predict Introductory Programming Performance". In: *International Journal of Computer Science and Software Engineering* 4.12 (2015), pp. 323–328. ISSN: 2409-4285. URL: http://mural.maynoothuniversity.ie/8682/

# 2 LITERATURE REVIEW

## 2.1 INTRODUCTION

Over the years there has been a significant amount of research related to CS1 and predicting student success using varying techniques. The literature review only focuses on models/literature for predicting success on introductory programming modules (CS1), where this approach served three purposes. The first was to identify if a more suitable model to PreSS was developed since the original PreSS study in 2006. Second was to investigate how models have been developed further/revalidated after their initial work. Third was to detail the original PreSS model, to make familiar and describe the foundations for this thesis, and to compare PreSS to the findings of the literature review. For the literature review itself, PreSS is omitted, but is described in detail in Section 2.3. The main findings of the literature review are presented in Table 2.1. For this body of research including Research Goal 4 and 5, a multitude of additional literature was considered/reviewed, where these are presented in the relevant sections. Thus allowing the literature review to focus on the core of this thesis, investigating the current landscape of predicting success in CS1 and the prediction model PreSS.

## 2.2 REVIEW OF THE LITERATURE

### 2.2.1 METHODOLOGY

An 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 where then searched in the ACM and IEEE databases with some additional searching in Google Scholar. The search terms were examined in 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 short listed, based on their relevance to CS, factors and/or prediction models. This resulted in 93 articles (when repeating articles were removed due to being returned in multiple searches). After this a detailed analysis of each article and its relevance was conducted. Each of the 93 articles were reviewed in full and were summarized under headings as defined later in this section. This process ensured that the articles were predicting and/or examining factors for CS1 or other introductory programming courses (and not for example a business course). This final selection process resulted in 49 articles that were included in this literature review.

Ideally models to predict student's performance in CS1 would broadly display as many of the following criteria as possible: multi-institutional, longitudinal, large sample size, high prediction accuracy, high sensitivity and early prediction. Many of these criteria were also highlighted in the ITiCSE working group report [62]. Almost all of the 49 articles examined, exhibited one or many (but not all) of these criteria and have been grouped and presented under these headings to give a sense of how the research available spans various criteria. Three of the articles did not satisfy any of the criteria [49, 94, 95]. Table 2.1 presents the criteria per study, and in doing so helps visualize what studies have met more than one criteria, and how the criteria overlap.

### 2.2.2 MULTI-INSTITUTIONAL

To create a generalizable prediction model, it would need to be tested over several institutions, preferably in diverse districts or even countries. From the 49 articles examined, only two studies were conducted in more than a single institution [23, 122]. A study by Bornat, Dehnadi and Simon [23] revisited a predictor of CS1 success, developed by Dehnadi in 2006 [40]. 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. Simon et al. [122] conducted a study across 11 institutions ($n = 177$), exploring issues that influence success in learning to program, using four diagnostic tasks. The study reported findings and general correlations (without reporting the actual values). The study pointed out the challenges and costs associated with a multi-institutional study, and perhaps this is the reason for an average participation of $\sim 16$ students per institution.

### 2.2.3 LONGITUDINAL

Studies are often conducted once, on a single cohort. Given that CS is a constantly evolving area, studies should be repeated over several years, to examine if they stand the test of time. The literature review found that only eight studies were conducted over more than one year or semester [3, 26, 48, 57, 74, 135, 143]. This is concerning, if several prediction models exhibit a strong prediction accuracy and value to the CSEd community, why were they never revalidated? No study that was longitudinal involved more than one institution.

### 2.2.4 GENERALIZABLE SAMPLE SIZE

To test a prediction model, a reasonable sample size is required. A 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 $> 5000$ [33]), a 10% acceptable margin of error was selected as the boundary value for the minimum generalizable sample size [33, 86]. 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 over-fitting. Some models in an effort to combat the small sample size, while trying to minimise over-fitting (a common problem with small data sets), used methods like bootstrapping. 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. A positive finding was that 35 of the articles reported a sample size greater than 96 students [3, 4, 8, 9, 12, 13, 15, 23, 26, 28, 30, 34, 42, 48, 57, 60, 67, 71, 73–76, 88, 109, 110, 118, 122, 131, 135, 136, 141–144, 147]. A positive note from this was that all studies that were longitudinal also included a generalizable sample size.

### 2.2.5 PREDICTION ACCURACY

Several of the articles, reported no significant prediction accuracies or correlations. This was also concerning, as this whittled the list down considerably. A prediction slightly higher than that of chance was selected as a search criteria. A similar correlation coefficient was selected so not to rule out this research. Ten articles reported significant prediction accuracies, although some did not predict early in CS1 [3, 22, 40, 57, 72, 74, 136, 138, 143, 146]. All articles irrespective of time of prediction, were included in the literature review as they may give insight, into factors not considered in this research. Only four studies that were both longitudinal and included generalizable sample sizes, produced a significant prediction accuracy, which included: [3, 57, 74, 143].

### 2.2.6 PREDICTION TIMING

In addition to prediction accuracy, prediction timing examined the point in the course the prediction could be made. The ideal timing is the earliest possible point into CS1, thus allowing educators to implement interventions in a timely manor.

**Early in CS1**

The limit for this sub group of articles was less than approximately 25% of the module completion, with 20 articles meeting this criteria [3, 4, 9, 23, 28–30, 40, 42, 57, 58, 60, 74–76, 96, 109, 110, 118, 122].

**Throughout CS1**

Some models made predictions at multiple stages throughout CS1, with varying levels of prediction accuracy at each stage [12, 45, 48, 63, 71, 106, 131, 135, 138, 141, 143]

**Prior to CS1**

A very positive finding in the literature is that some models were able to predict before the commencement of CS1. [26, 52, 57, 67]

The four studies that met all of the criteria (excluding multi-institutional) that could predict in a timely manner are: Ahadi, Glorfeld, Liao and Wiig [3, 57, 74, 143].

### 2.2.7 PREDICTION SENSITIVITY & SPECIFICITY

Prediction models are often presented with a high accuracy, but that alone does not always reveal the entire story of the model. If sensitivity and specificity are not presented and identified, two concerns can be raised. First, a biased model could have been constructed. For example, if 90% of the students in a study were strong (high performing), the model could predict every student as strong and report an accuracy of 90%. This model presents as very successful, but its ability to identify weak students (the model's main goal) would in-fact be 0% as it predicted every student as strong, thus the 10% of students that are weak, were incorrectly identified. Second, the accu-

racy does not present outcomes for a particular class, therefore making appraisals of models that aim to predict students who are struggling, masked when the study only presents accuracy. This measurement of sensitivity and specificity, was only reported in three studies (in some cases indirectly, but it could be calculated) [22, 57, 74].
Only two studies remained that met all of the criteria (excluding multi-institutional) that also reported sensitivity and specificity: Glorfeld and Liao [57, 74].

### 2.2.8 REVALIDATED 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 presented themselves (some of the authors were even emailed directly to confirm if this was the case, if the article stated a possible follow up study).
**Dehnadi**
In 2006, Dehnadi developed a prediction model, that reported a 100% accuracy (100% sensitivity and specificity) [40]. This work seemed to have made a breakthrough. It was disclosed at the the PPIG (Psychology of Programming Interest Group) workshop in 2006. Dehnadi built a "mental model" that could predict success with 100% accuracy ($n =$ 60 students). Based on this reported accuracy, two follow up studies were completed. In 2007, Caspersen repeated the study using approximately 142 students [30]. It should be noted at this point that Dehnadi completed the study in the UK, whereas Caspersen conducted the study in Denmark. The findings of Caspersen's work is 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 in the following year (2008, co-authored by Dehnadi) examined six experiments, with more than 500 students, across six institutions and three countries [23]. Bornat reported that "the predictive effect of our test has failed to live up to that early promise" with performance, just higher than chance.

**Glorfeld & Fowler**
In 1981, Glorfeld and Fowler, developed a predictive model using a sample size of 151 students in a CS1 course [52]. The model was developed using three pools of data: personal, academic and aptitude. From this data a classification model was developed using the logistic discrimination model. The model produced an accuracy of 80.8% and not only was the accuracy presented but also the sensitivity and specificity. For

identifying weaker students the model was 76.6% accurate. A year later Fowler and Glorfeld, revisited the study with a new cohort [57]. From the 1040 students enrolled in the CS1 course, 150 were randomly selected. The model still performed well, although the accuracy decreased (~ 6%). This is perhaps to be expected when models are exposed to new data sets and being tested for generalizability. Glorfeld and Fowler reported that: "The validation study showed that the model would have a predictive accuracy of approximately 75% in actual application".

The most significant model following the Literature review was developed by Glorfeld and Fowler [52], but as outlined in the following section PreSS performs at a comparably high or higher level and is therefore a good choice of model for further development.

## 2.3 THE ORIGINAL PRESS STUDY AND MODEL

PreSS was developed between 2002 and 2006 based on three studies (a pilot study, a main study and an epilogue study) with four participating institutions, n= 184 [16–18]. These institutions consisted of a University, two Institutes of Technology, and a Community College, thus spanning all levels of Higher Education, from the National Framework of Qualifications (NFQ) level 5 to level 8. The pilot study recorded data using a questionnaire, which in turn lead to the development of the instruments that were used in the main study. The main study recorded a substantial amount of data from 102 students during the academic year of 2004-2005. Four paper-based instruments were used to collect the data in the main study: a background questionnaire, a programming self-esteem questionnaire, a self-efficacy questionnaire and a motivation and learning strategies questionnaire [16, 17]. An epilogue study with 21 participating students was also conducted after the main study and was used in further analysis and validation of findings from the main study.

### 2.3.1 FACTOR SELECTION

In the original PreSS study 25 factors were examined, using four instruments. The original PreSS study used multiple statistical techniques in the development of the prediction model. Stratification was used as the initial step to ensure that all the data was in a homogeneous state. Initially, the original PreSS study examined over 40 logistic regression models using combinations of the 25 factors on the data from the main study [1]. This prediction model used three factors: programming self-esteem [2], mathematical ability (based on a high school mathematics exit examination) and the number of hours per week that a student plays computer games. The programming self-esteem and mathematical performance were found to have a positive relationship with performance while the number of hours a student plays computer games was found to have a negative effect.

### 2.3.2 DATA PREPROCESSING

The pre-processed programming self-esteem data consisted of ten questions and was based on the Rosenberg self-esteem questionnaire but modified to reflect a student's perception of their programming ability [107]. Cronbach alpha values were calculated to compare the internal consistency of the self-esteem questionnaire and were found to be comparable with the Rosenberg self-esteem questionnaire (Appendix A.1). Principle Component Analysis (PCA) was used to reduce the programming self-esteem questionnaire that consisted of multiple data points to one value which accounted for as much of the variance in the multiple data points as possible (Appendix A.1).

### 2.3.3 MACHINE LEARNING ALGORITHM

Six machine learning algorithms were examined in the development of PreSS: logistic regression, k-nearest neighbour, backpropagation (single layer artificial neural network), C4.5 (decision tree), SVMs (support vector machine) and naïve Bayes. Naïve Bayes was selected as it was found to have the highest prediction accuracy. As naïve Bayes is used extensively throughout this paper a detailed description of the machine learning algorithm can be found in Section 6.3.4.

1 This resulted in a single prediction model that produces the highest measurement of programming performance.

2 With respect to programming self-esteem, it should be noted that there are several related terms used within this space and their boundaries are sometimes unclear. These terms have included (from the literature): programming self-esteem, CS confidence, programming self-efficacy, and programming self-confidence. At the time of Bergin's work this physiological phenomena was referred to as programming self-esteem, whereas nowadays its more often than not referred to as programming self-efficacy, with both terms referring to the same physiological phenomena. As the original PreSS study published it as programming self-esteem it will be referenced as this, if referring to the original PreSS study, while in all other cases the term will be referred to as programming self-efficacy.

### 2.3.4 PERFORMANCE MEASURES

For the development of the PreSS model, ten-fold cross validation was implemented to obtain a prediction accuracy. This is a best practice method to avoid over fitting [148] with machine learning algorithms. This method is far superior to a hold-out method as it uses every sample in a data set for both testing and training. A detailed description of this technique is presented in Appendix A.2.

Accuracy, sensitivity and specificity were recorded for each prediction. These recordings were calculated based on the following variables: true positive rate (TP - the correct identification of a weak student), true negative rate (TN - the correct identification of a strong student), false positive rate (FP - type I error) and the false negative rate (FN - type II error). As the overall priority of PreSS is to identify weak students, sensitivity was the main focus. Thus sensitivity in this thesis, was the measure of performance of the model to predict students at risk of failing or dropping out, and specificity was the measure of performance of the model to predict students who would pass or be successful. All three measurements were recorded and presented for every prediction in this study. A detailed description and the formula for each measure is presented in Appendix A.3.

### 2.3.5 RESULTS AND PERFORMANCE

PreSS was able to achieve an accuracy of 77.5% [16] in the main study, with n = 102. In fact using a variation of factors, higher performance could be achieved for a specific sample set, for example, by gender or per institution, however Bergin's goal was to produce the most generic model possible to maximise generalizability. The sensitivity achieved by PreSS (predicting students whom are at risk of failing or dropping out) was 78%.

## 2.4 SUMMARY

In summary the literature review revealed that while many of the 49 articles contribute to the CSEd community, the process confirmed several of the ITiCSE working group concerns [62]. One of the most prominent issues is that the CSEd community find repeatability a challenge, replication studies are rare, and that they may still be as rare a decade from now [62]. 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.

Glorfeld in 1981 and in 1982 [52, 57] developed a strong prediction model. In addition, statistical techniques for generalization were implemented (ten fold cross validation) and even confusion matrices were presented (allowing specificity and sensitivity to be calculated). This really was remarkable work for the early 1980's. 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, to that end attempts were made to contact the authors as no further publications in this space were found. Sadly Fowler has since passed away and it was not possible to contact Glorfeld, however colleagues of his who were familiar with the work were reached, and they confirmed that 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 three years. Thus the PreSS model remains the starting point for this thesis, but factors identified by Glorfeld will be examined in the PreSS model development, in particular the age factor.

The literature did not present (from the search terms and methodology used, as outlined in Section 2.2.1) a prediction model that would offer value over the original PreSS model. In fact when examining the literature, while it has become apparent that there is a large amount of related models, the majority seem to follow the pattern of a one-off study, in a single institution, seldom revisited after the model has been developed. This is concerning for the CSEd community and shapes the overall direction of this thesis. If the PreSS model could address the shortfalls of these educational prediction models, an updated PreSS model may be a model of real value to the CSEd community.

### 2.4.1 ADDITIONAL FINDING: DATE OF STUDIES

While reviewing the literature, and the dates of the studies (ranging back as far as 1975) from the 49 articles, a natural clustering of the dates became apparent. This interesting finding, found three natural clusters of predicting student success in CS1 research. The initial cluster centred around the mid 1980's [9, 26, 29, 49, 52, 57, 60, 67, 72, 88, 141, 143, 144] , the second cluster was centred around the mid 2000's [4, 8, 12, 13, 15, 22, 23, 30, 40, 45, 58, 63, 94, 106, 109, 110, 122, 135, 142, 146] and the third cluster gathers in the past five years or so [3, 28, 34, 48, 71, 73–76, 95, 96, 118, 131, 136, 138] . A noteworthy finding, is that no study attempted to validate from one cluster to another. The studies did not seem to egress past their own cluster. This is disappointing when one's focus is to develop a model that performs well over time.

Table 2.1: Summary of Literature Review References

| Year | Lead Author | Ref | Multi-Institutional | Longitudinal | Large Sample | High Accuracy | Early Timing | Sen and Spec | Revalidated |
|---|---|---|---|---|---|---|---|---|---|
| 2006 | PreSS | [16] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 2015 | Ahadi | [3] |  | ✓ | ✓ | ✓ | ✓ |  |  |
| 2004 | Allert | [4] |  |  | ✓ |  | ✓ |  |  |
| 1983 | Barker | [9] |  |  | ✓ |  | ✓ |  |  |
| 2009 | Barker | [8] |  |  | ✓ |  |  |  |  |
| 2005 | Bennedsen | [12] |  |  | ✓ |  | ✓ |  |  |
| 2006 | Bennedsen | [13] |  |  | ✓ |  |  |  |  |
| 2008 | Bennedsen | [15] |  |  | ✓ |  |  |  |  |
| 2005 | Boetticher | [22] |  |  |  | ✓ |  | ✓ |  |
| 2008 | Bornat | [23] | ✓ |  | ✓ |  | ✓ |  |  |
| 1985 | Butcher | [26] |  | ✓ | ✓ |  | ✓ |  |  |
| 2016 | Campbell | [28] |  |  | ✓ |  | ✓ |  |  |
| 1975 | Capstick | [29] |  |  |  |  | ✓ |  |  |
| 2007 | Caspersen | [30] |  |  | ✓ |  | ✓ |  | ✓ |
| 2015 | Cukierman | [34] |  |  | ✓ |  |  |  |  |
| 2006 | Dehnadi | [40] |  |  |  | ✓ | ✓ |  | ✓ |
| 2010 | Denny | [42] |  |  | ✓ |  | ✓ |  |  |
| 2009 | Doyle | [45] |  |  |  |  | ✓ |  |  |
| 2016 | Estey | [48] |  | ✓ | ✓ |  | ✓ |  |  |

Table 2.1: Summary of Literature Review References

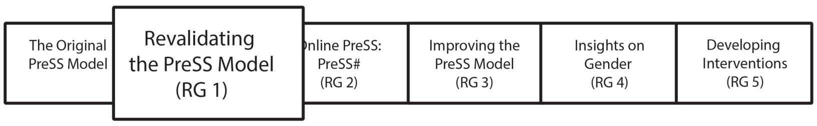
| **Year** | **Lead Author** | **Ref** | **Multi-Institutional** | **Longitudinal** | **Large Sample** | **High Accuracy** | **Early Timing** | **Sen and Spec** | **Revalidated** |
|---|---|---|---|---|---|---|---|---|---|
| 1989 | Evans | [49] | | | | | | | |
| 1981 | Glorfeld | [52] | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 1982 | Glorfeld | [57] | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 2006 | Golding | [58] | | | | | ✓ | | |
| 1983 | Hostetler | [60] | | | ✓ | | ✓ | | |
| 2003 | Katz | [63] | | | | | ✓ | | |
| 1983 | Konvalia | [67] | | | ✓ | | ✓ | | |
| 2015 | Lambert | [71] | | | ✓ | | ✓ | | |
| 1982 | Leeper | [72] | | | | ✓ | | | |
| 2017 | Leinonen | [73] | | | ✓ | | | | |
| 2016 | Liao | [74] | | ✓ | ✓ | ✓ | ✓ | ✓ | |
| 2016 | Lishinski | [75] | | | ✓ | | ✓ | | |
| 2016 | Lishinski | [76] | | | ✓ | | ✓ | | |
| 1975 | Newsted | [88] | | | ✓ | | | | |
| 2004 | Pioro | [94] | | | | | | | |
| 2014 | Porter | [96] | | | | | ✓ | | |
| 2014 | Porter | [95] | | | | | | | |
| 2009 | Rodrigo | [106] | | | | | ✓ | | |
| 2002 | Rountree | [109] | | | ✓ | | ✓ | | |

Table 2.1: Summary of Literature Review References

| **Year** | **Lead Author** | **Ref** | **Multi-Institutional** | **Longitudinal** | **Large Sample** | **High Accuracy** | **Early Timing** | **Sen and Spec** | **Revalidated** |
|---|---|---|---|---|---|---|---|---|---|
| 2004 | Rountree | [110] | | | ✓ | | ✓ | | |
| 2016 | Shell | [118] | | | ✓ | | ✓ | | |
| 2006 | Simon | [122] | ✓ | | ✓ | | ✓ | | |
| 2016 | Tarimo | [131] | | | ✓ | | ✓ | | |
| 2005 | Ventura | [135] | | ✓ | ✓ | | ✓ | | |
| 2013 | Vihavanen | [136] | | | ✓ | ✓ | | | |
| 2013 | Watson | [138] | | | | ✓ | ✓ | | |
| 1986 | Werth | [141] | | | ✓ | | ✓ | | |
| 2007 | Wiedenbeck | [142] | | | ✓ | | | | |
| 1989 | Wiig | [143] | | ✓ | ✓ | ✓ | ✓ | | |
| 1981 | Wileman | [144] | | | ✓ | | | | |
| 2001 | Wilson Shrock | [147] | | | ✓ | ✓ | | | |

3

# REVALIDATING THE PRESS MODEL


The Original PreSS Model
Revalidating the PreSS Model (RG 1)
Online PreSS: PreSS# (RG 2)
Improving the PreSS Model (RG 3)
Insights on Gender (RG 4)
Developing Interventions (RG 5)

## INTRODUCTION

The re-validation of the original PreSS model was conducted over three academic years from 2013 to 2016. This initial body of work was concerned with addressing the first research goal, to investigate if the original PreSS model was still valid almost a decade after it was first developed. Since 2006, there have been multiple significant changes in the computer science landscape, for example, with the advent of mainstream social media such as Facebook (circa late 2006) and smartphones (circa 2007 with the first iPhone). Even forms of more formal communication such as email have migrated to platforms like Microsoft Teams, with success across many levels of education [1]. All of these changes, arguably have changed the landscape significantly, for both student experiences and computer science subject content. In addition, as mentioned in Chapter 2, revalidation attempts often result in lower performance or fail to replicate at all, even over small periods of time. Thus given the changes in landscape and the time lapse between the original PreSS study and this body of work, examining if the PreSS model is still valid is a critical initial Research Goal. This is addressed using two studies, a justification study and the main study of this thesis.

1 https://www.microsoft.com/en-ie/education/stories/default.aspx

## 3.1 JUSTIFICATION STUDY

### 3.1.1 INTRODUCTION

The justification study, as presented in this chapter, consisted of two consecutive independent studies. The two studies were carried out in the academic years of 2013-14 (Initial Study) and 2014-15 (Second Study) respectively and were both hosted by 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 National Framework of Qualifications (NFQ) Level 6 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 that contained additional questions that are later explored, which is presented in Appendix D.

The work described in this chapter was published in the proceedings of the International Conference on Engaging Pedagogy (ICEP), College of Computing Technology, Dublin in 2015 [97].

### 3.1.2 FACTORS EXPLORED

Students completed a questionnaire that contained the original PreSS study questions on self-efficacy, prior maths performance and game playing. Students also provided responses on other factors for separate investigation. These factors were then collected as part of the main study (Chapter 3.2) and examined further in Section 5.3. The additional factors collected are presented in Table 3.1 and Appendix D.

Table 3.1: Justification study, additional survey questions

| | **Factor** |
|---|---|
| 1 | Age Bracket? |
| 2 | Gender? |
| 3 | How many hours per day would you play computer games on a mobile device? |
| 4 | If you play games on a mobile device, what genre of games do you play the most? |
| 5 | How many hours per day would you play computer games on a Console, PC or laptop? |
| 6 | If you play games on a console, PC or laptop, what genre of games do you play the most? |
| 7 | How many hours per day would you use the internet (not including social media or messaging services)? |
| 8 | What would your primary use of the internet consist of (not including social media or messaging services)? |
| 9 | How many hours per day would you use a social networking service? |
| 10 | If you do use social networking, what particular service do you use the most? |
| 11 | How many hours per day would you use a messaging service? |
| 12 | If you do use messaging service, what particular service do you use the most? |

### *Initial Justification Study*

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 study consisted of five females and 29 males. The overall final results showed the ratio of weak to strong students was 16:18 respectively (as developed in Appendix B) and the ratio between mature students (students 23 years of age or older) and non-mature students (students under the age of 23) was 12:22 respectively.

### *Second Justification Study*

The second study consisted of 26 students (all students in the class participated). There was no missing data entries so no student was excluded from the final sample. Four females and 22 males took part in the study. The overall final results showed the ratio of weak to strong students was 9:17 respectively and the ratio between mature students and non-mature students was 3:23 respectively.

### 3.1.3 RESULTS

The goal of the two studies was to determine if they could replicate the results of the original PreSS study, using the original PreSS model. The original PreSS data samples ($n = 102$, from the main study of the original PreSS study) were used as training data. The process was identical to the process used in the development of original PreSS model, using the same machine learning toolbox and methods to compute prediction. Both prediction accuracies using the two justification studies were independently compared to the original PreSS model accuracy. Statistical $t$-tests (using a Welch's $t$-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 (initial study: $Accuracy = 76.5\%, p = 0.86$ second study: $Accuracy = 77\%, p = 0.57$).

This is a significant result as both studies consisted of very different student profiles, for example institution representation (community colleges represented less than 7% of institutions in the original PreSS study, coupled with the lower entry requirements for a community college compared to a university or an institute of technology). This suggested that even though the original PreSS model was developed nearly a decade ago, that similar levels of accuracy may be achievable in a considerably changed landscape.

### 3.1.4 SUMMARY

The positive preliminary findings of the justification work, provide the pretext for this thesis to continue and address RG 2 - 5 (that is to develop a web-based system (RG2), to improve on the model's performance (RG3), to examine gender (RG4), and to evaluate possible interventions (RG5).

## 3.2 MAIN LARGE SCALE STUDY

### 3.2.1 INTRODUCTION

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 work described in this section was published in the proceedings of the 23rd annual ACM conference on Innovation and Technology in Computer Science Education (ITiCSE), 2018 [100].

#### *Additional Positioning*

In 2015 an ITiCSE working group identified a "critical need" for re-validation studies in educational data mining and learning analytics [62]. The working group reported that the majority of the studies reviewed, focused on simplistic metric analysis and were conducted within a single institution and a single course. This highlighted a critical need for validation and replication to better understand the contributing factors and reasons that certain results occur in computer science education [62]. The working group concluded with several Grand Challenges, the second of which was to systematically analyse and verify previous studies using data from multiple contexts to tease out tacit factors that contribute to previously observed outcomes [62]. This chapter contributes to the computer science education (CSEd) community and the working group's call (in particular the second grand challenge), by revalidating the original PreSS model, twelve years after it was developed, on a modern disparate, multi-institutional data set.

### 3.2.2 DATA COLLECTION

During the academic year 2015-16, a large-scale multi institutional study [103] took place in Ireland (ten institutions) and Denmark (one institution) consisting of two universities, five institutes of technology (comparable in academic level to colleges in the US) and four community colleges. The data collected was captured under two categories. The first category captured student data, including background, institution/course and psychological data, which was captured at approximately 4-6 hours into CS1 (this is approximately, 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). In total 45 factors were recorded from the data gathered in the first category. Details of the instruments used and all of the data collected can be found in the references [103].

### 3.2.3 THE MAIN STUDY: THE CORE OF THIS RESEARCH

This study involved a large number of participants, at multiple institutions, at varying tertiary levels. Uptake was very positive, with institutions volunteering to participate, during presentations at conferences of the PreSS model. The integration and set-up needs with each participating group were different and several months of discussion (even addressing ethics boards in person in some cases) was needed. With institutions on-board, log-in credentials (consisting of anonymous keys provided by the institutions) had to be agreed and then the users created on the PreSS# system (Chapter 4). In addition, at the end of the academic year, the final grade was collected and matched with each anonymous user key.

The main study while addressing RG 1 also collected a multitude of additional factors and data so that RG 3 (further development of the original PreSS model) could be addressed. This study also provides a baseline for any model development improvements, to be compared against. In addition, the information collected also provided a rich data source for RG 4 (insights into gender differences in CS1). This study lays the foundations for Chapter 5 and Chapter 6.

### 3.2.4 RESULTS

The PreSS model, data pre-processing techniques, and machine learning algorithm were unchanged from the original PreSS study [16], with the same three factors: programming self-efficacy, mathematical ability based on a high school mathematics exit examination and number of hours per week a student plays computer games [17]. The results and comparison between the original study and this study are presented in Table 3.2.

Table 3.2: The original PreSS study compared to the main study, using the original PreSS Model.

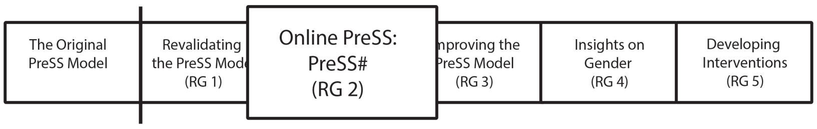
| **Data Set** | **N** | **Accuracy** | **Sensitivity** | **Specificity** |
|---|---|---|---|---|
| 2005 | 102 | 77% | 85% | 66% |
| Current Study | 692 | 67% | 78% | 53% |

The original PreSS model was able to predict with a high accuracy (for two out of every three students), 12 years on, with an $n = 692$, from 11 institutions with diverse academic levels in two different countries. Although the accuracy reduced by 10%, this result is still significant as previous revalidation attempts on other models (Chapter 2) have seen the model rejected or an accuracy reduction of 6% in the span of a single year, within a single institution. More importantly PreSS was able to identify weak students (the main goal of PreSS) with a sensitivity of 78%. It was anticipated that the accuracy would decline, given the substantial increase in student numbers, diversity of institutions/academic levels and change in student profile over time. From the literature it appears that no other model for predicting programming performance has been revisited over ten years after its creation (multi-institutional and longitudinal), thus these results may provide a benchmark for future revalidation studies, predicting student success in CS1.

### 3.2.5 SUMMARY

This chapter describes a multi-institutional and longitudinal revalidation study of the original PreSS model and even though the CS landscape has changed considerably, and the cohort of participants are considerably different, the model was found to predict with high accuracy (correctly predicting 2 out 3 students). This chapter builds upon the work of previous studies and makes a solid contribution to the CSEd community, laying the foundation for future revalidation and replication studies. In doing so, this chapter has addressed the second of the grand challenges as laid out by the ITiCSE working group in 2015. Previous revalidation attempts in the literature (as outlined in Section 2.2.8), reported a reduction in accuracy over a single year in a single institution, or failed to reproduce the findings at all. PreSS was revalidated a decade later using a significantly larger cohort of students from 11 institutions (compared to the four institutions in the original PreSS study). Thus it was anticipated that the accuracy would decline, but most positively the original PreSS model still had a sensitivity of 78%. New factors such as social media usage rather than time spent playing computer games, could be more indicative now [99]. The revalidation of the original PreSS model confirms that the model still performs well, over a decade after its conception.

# 4 PRESS#



## 4.1 introduction

A web-based implementation of the original PreSS model was developed named PreSS#, to increase accessibility to the model and reduce the amount of manual computation required. This chapter describes the development of an on-line version of the original PreSS model, named PreSS#. This chapter contributes to RG2 (*Develop a web-based real time implementation of PreSS*). The goal of this system was to allow the original PreSS model to be used on a large scale, to add the additional factors recorded in the justification study, and to have plug and play machine learning algorithms. Before the system was developed, consideration was given to usability, security and the overall system architecture. Thus before any development began the system requirements were shaped. The work described in this chapter was published in the International Journal of Computer Science and Software Engineering (IJCSSE), Volume 4, Issue 12, 2015 [102].

## 4.2 system requirements

The design process which developed the system requirements was largely taken from Requirements Engineering (RE) [68]. Activity one was the software requirement elicitation which identified four main headings as the pillars for the system. These were: security, user, device and system. These pillars were then (activity two) negotiated and developed in UML diagrams (presented later in this section). Activity three was the development of the software requirements and finally (activity four) validated the requirements using the the main large scale study, using the verbal feedback from institutions who participated in the study.

System requirements rationale:

- **Security:** The system for ethical approval and institution uptake needed to employ the latest security measures.
- **User:** For a large scale study, user roles and ease of bulk creation was vital. It was envisioned that institutions would provide a key, thus PreSS# would have no user identifiable data. When an institution provided the keys and passwords, for log on credentials, a timely user creation was necessary, and with some institutions having over 300 students, automating this task was vital.
- **Device:** With a multitude of devices available to participating students and institutions, the system should function on all commonly used systems, including mobile devices, and be intuitive to use, promoting uptake of the study.
- **System:** The system, required rapid development and deployment techniques. A system that employed a separation of concerns would again complement the security requirement. As additional models and data reduction techniques may be identified and used in the final model, a plug and play model would be beneficial at the core of the system.

The specific software requirements for each key area are referenced by an ID tag which is used throughout this chapter when referring to a specific requirement and are shown in Table 4.1 and 4.2 ; for example if the Software Requirement in Table 4.1 had an ID of 1, the reference will be (SR1).

Table 4.1: Software requirements for the development of PreSS# - Part 1

| ID | Domain | Description | Rational |
|---|---|---|---|
| SR1 | Security | IP address capture of user | Track users of system, in case of a breach |
| SR2 | Security | Encryption of all content | If server is breached, data is unusable |
| SR3 | Security | XXS, CSFR and brute force | Three main cyber-attack methods needing attention |
| SR4 | Security | HTTPS | Encryption during transit |
| SR5 | Security | Authentication | Robust authentication, user details encrypted. |
| SR6 | User | User creation (single & bulk upload) | If server is breached, data is unusable |
| SR7 | User | Reset user password | Reset (not view to change) user password if required |
| SR8 | User | Training data (single & bulk upload) | Allow rapid creation of large training instances |
| SR9 | User | Prediction system | Real time system, with reporting facilities |

Table 4.2: Software requirements for the development of PreSS# - Part 2

| ID | Domain | Description | Rational |
|---|---|---|---|
| SR10 | User | Roles: student, lecturer & admin | Roles used for specific controller access |
| SR11 | User | Student can only take survey once | Survey only required once |
| SR12 | User | Lecturer and admin reset survey | Student can redo survey if needed |
| SR13 | Device | Device compatibility | Phone, tablet, PC and smart TV, HTML5 and CSS3 |
| SR14 | Device | Dynamic controls | Web controls change to be intuitive on native device |
| SR15 | System | Development & deployment | Fast development and deployment required |
| SR16 | System | Separation of concerns | Development in an MVC architecture |
| SR17 | System | PCA | Integration of PCA into the system |
| SR18 | System | Naïve Bayes | Integration of naïve Bayes into the system. |

## 4.3 PLATFORM SELECTION

### 4.3.1 SERVER SELECTION

Selection of an appropriate software platform is vital in developing a secure, robust and reliable system. The two major leading software platforms are Ubuntu Server (Java) and Microsoft Server (.NET). There have been many debates over which platform is superior based on market share, pricing etc. Market share is highly variable based on time and the area of deployment [117] and pricing can be considered less important if the quality of the system is a priority, thus attributes such as security, vulnerabilities and performance are considerably more important here given the nature of the student data being processed and the real time performance requirements. These criteria are paramount if PreSS# as a tool is to be used across multiple institutions with large student numbers. This section discusses security, vulnerabilities and performance in detail and concludes with platform and language selection.

### 4.3.2 SECURITY AND VULNERABILITIES

Security was a major factor in the selection of the overall platform. A recent study of vulnerabilities and errors in .NET and Java found, other than input data validation vulnerabilities (which have been addressed in the latest release of .NET), IIS (Internet Information Services, the .NET web server) was less vulnerable than Apache (Ubuntu Server Web Server) in every other examination of error including access validation, configuration and design errors [149]. Similarly, a study on security focusing on encryption concluded that most Java encryption API's demonstrated relatively poor performances on the Linux platform as compared to those on the Windows platform [53]. One example of the poor encryption performance in Java was the openSSL vulnerability, referred to as "Heart bleed" [137], which was present on the Apache system, but as Windows used a different SSL (SChannel) it was unaffected by this vulnerability.

### 4.3.3 DATABASE MANAGEMENT SYSTEM (DBMS)

Performance relating to data storage and retrieval is key to a real-time system, with large data sets. The Microsoft platform typically uses Microsoft SQL Server whereas Apache typically uses MySQL. These two database management systems (DBMS) are associated with .NET and Java respectively. A study by Bassil [10] analysed the performance of several DBMS, which included SQL Server and MySQL. The database design was identical on each DBMS, which included 15 distinct tables and relationships with 1,000,000 records. The study consisted of running ten queries varying in complexity from a simple select-all query to highly mathematical complex queries, thus completing a thorough analysis of the performance of each DBMS. Bassil measured the length of execution time of each query and found that SQL server was 1% faster over the ten queries and also faster in 80% of the individual queries [10].

### 4.3.4 LANGUAGE SELECTION

The initial short-listing of possible languages was based on the use of the Model-View-Controller (MVC) architecture as shown in the MVC architecture diagram in Figure 4.1. This architecture was selected based on the separation of concerns (SoC) design principle which fulfils the requirements of PreSS# with modular and secure development at the forefront (SR16). In MVC, the architecture is composed of three separate interconnected parts. The controller is the core and is implemented on the server side. The controller returns generated views to the user and the controller gets its data from models which consist of Plain Old CLR objects (POCO). The language short-list consisted of MVC4 .NET and Rails for Ruby, both of which use MVC architecture. Both languages have their own variation of MVC. Microsoft .NET was selected as the development language. In addition .NET easily integrates with the selected platform (Microsoft Server) and DBMS (Microsoft SQL Server) and includes some additional factors not found in Rails for Ruby, which include data annotations and the Entity framework which can increase the speed of development. Both are described further in the following sections.

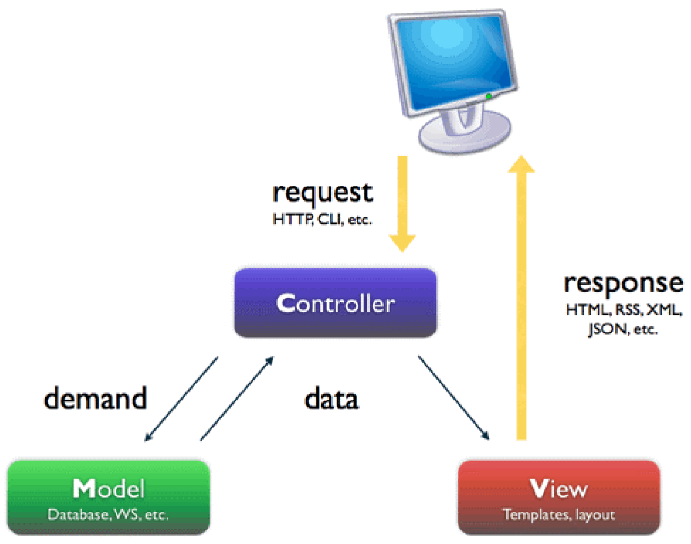


Figure 4.1: MVC architecture diagram

### 4.3.5 PLATFORM SELECTION SUMMARY

Even though there is a lot of public debate over which platform and DBMS to use, evidence that the Microsoft Server is arguably less vulnerable, has faster execution times and has a more robust encryption algorithm than Apache [149]. The Microsoft platform was chosen as a suitable platform for the development of the web-based educational system. Server 2008R2 and SQL Server 2008R2, both the market leaders in the Microsoft platform range were selected. The server used was a VPS server, located in a data centre in Ireland. Additional security consisted of an industry standard hardware firewall (Cisco ASA 5500), which only allows remote access to the server operating system via RDC (Remote Desktop Connection) from valid Irish IP address ranges, thereby narrowing the possibility of geographical attacks. The web system hosted on IIS can be accessed via the HTTPS port 443. Access via the Remote Desktop Services (RDS) was secured with SSL encryption and the latest server updates were automatically installed.

## 4.4 DESIGN

The design specification is shown in Figure 4.2 and Figure 4.3. Figure 4.2 provides the UML use case diagram for all three types of users, outlining each case that the user may access (SR10). Figure 4.3 illustrates the UML diagram of the MVC architecture used with the actors accessing the views from the controller. If the actor is a student they may then complete the form in the view which is returned to the controller. The controller then validates the data in the form (if there are any validation errors, the same view is returned to the student with validation error messages, and this is repeated until the validation passes), and passes the data into the naïve Bayes and PCA models to compute the predictions. The predictions are returned to the controller which may be sent to a lecturer/teacher via a view.

## 4.5 DEVELOPMENT OF PRESS#

The development phase of PreSS# was composed of three parts: PCA and naïve Bayes model development (Section 4.5.1), integration of both models into PreSS# (Section 4.5.2) using the MVC controller, and the development of the web system's interface and front end (MVC views) that are generated by the controller, Section 4.5.3 focusing on the system requirements set out in Table 4.1 and C.3. Although this was a solo development process, the size of the system required it to be developed in phases with testing at each phase using an iterative approach. This allowed for defined development phases, key to the successful development of PreSS#, where a traditional methodology (waterfall), may have not been as efficient. The testing is discussed further in Section 4.6.

### 4.5.1 PCA AND NAÏVE BAYES DEVELOPMENT

Two separate model classes were independently developed and then examined to determine if they could accurately replicate the results of the PCA and naïve Bayes

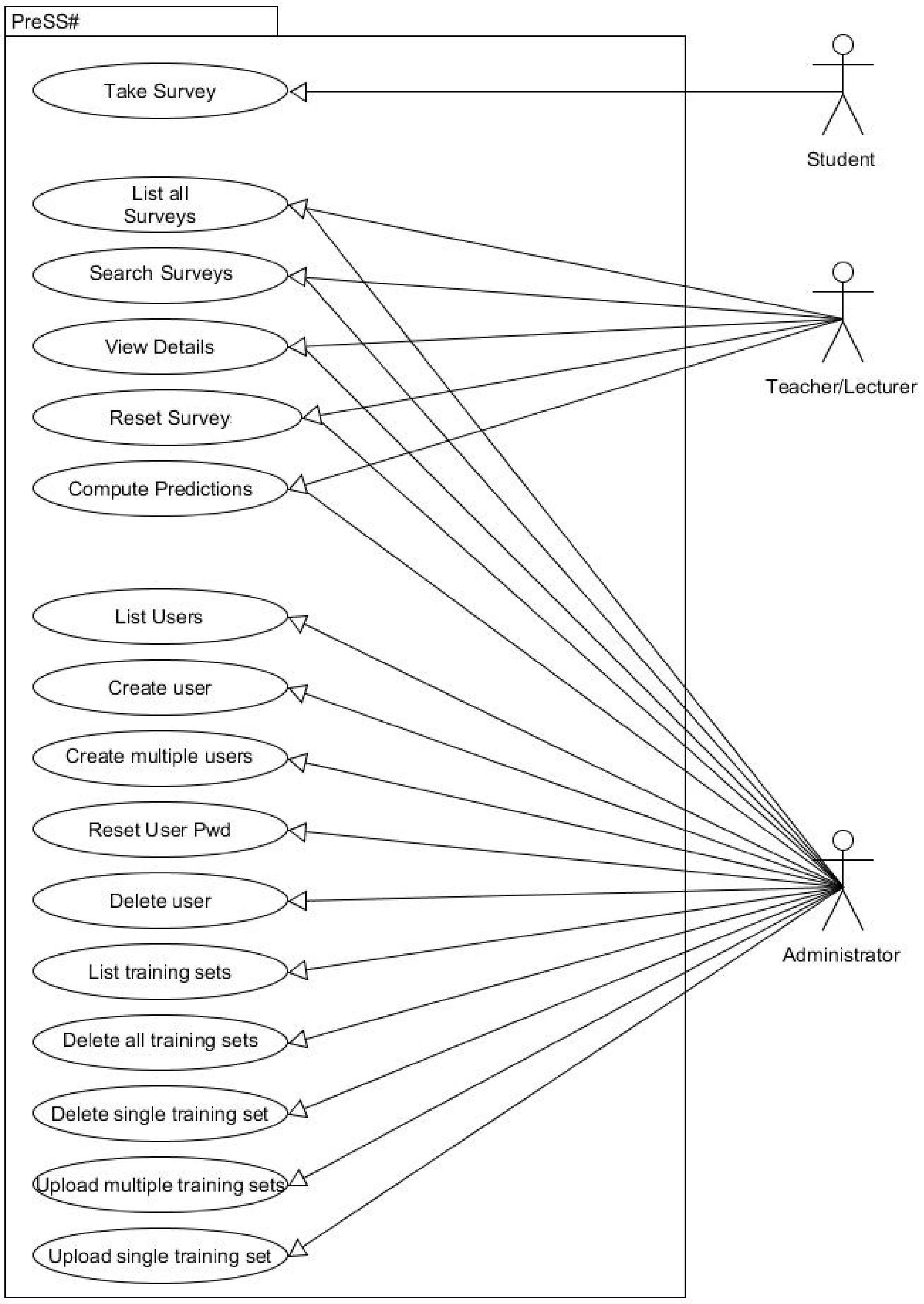


Figure 4.2: UML use case diagram for the three types of users.

implementations used in PreSS. The models were developed using C# (.NET) and the Accord.Net machine learning framework [126]. Two independent software applications were developed to investigate the .NET performance of PCA and naïve Bayes.

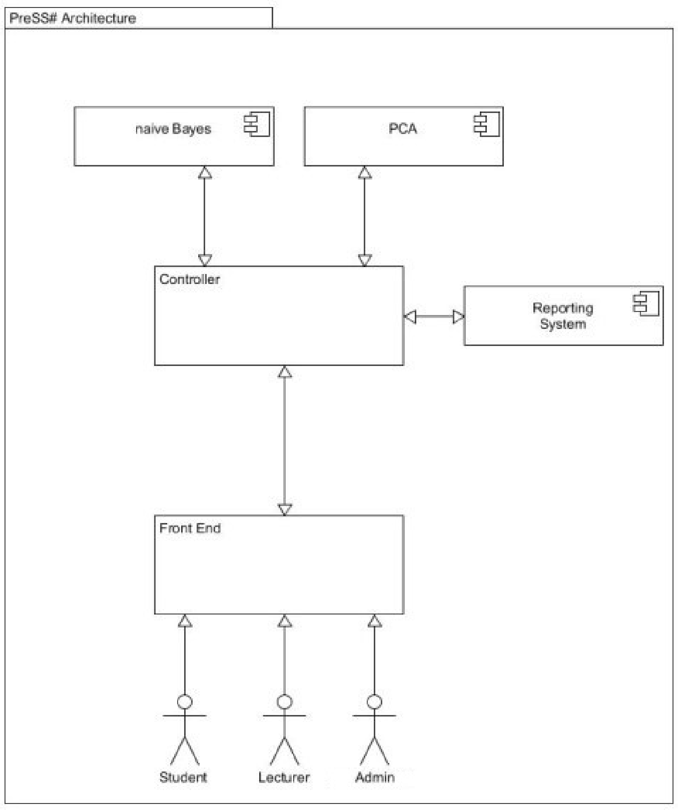


Figure 4.3: UML architecture diagram of PreSS#

These applications were developed using libraries that (once validated independently) could be ported into PreSS#, conforming to the Agile-like approach.

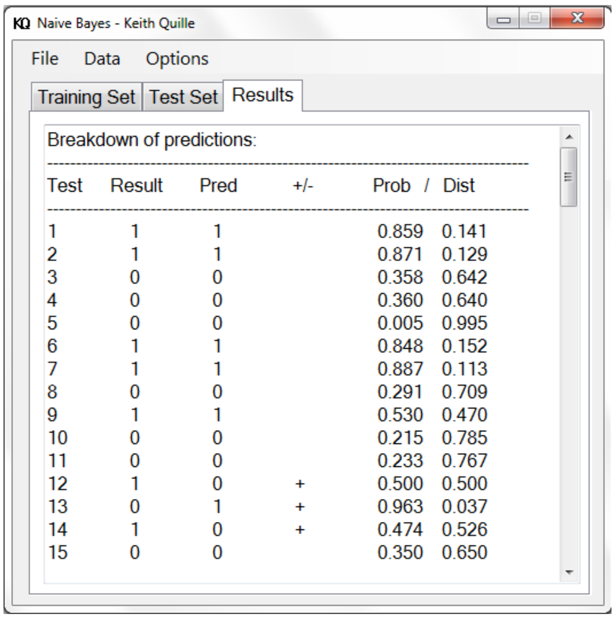
### *PCA Application*

The PCA application needed to be tested to determine that data sets of different size and format would work before it was finally integrated into the web system. This was essential as the data set size in future studies may vary significantly. The original PreSS study only used the first principal component (Appendix A.1), where PreSS# also calculated the second and third principal components for future proofing, even if not currently implemented. The system was able to open files in two formats which allowed for quick loading of different test data, either from WEKA (.ARFF format) or from Comma Separated Values (.CSV) file formats. The results were added to a visual output form and the system allowed for the export of the results either via .CSV format or to print the data. A visual reporting tool was also developed to graphically display the first principal component to aid the user in visualising the first principal components for any post-hoc analysis of the data.

### *Naïve Bayes Application*

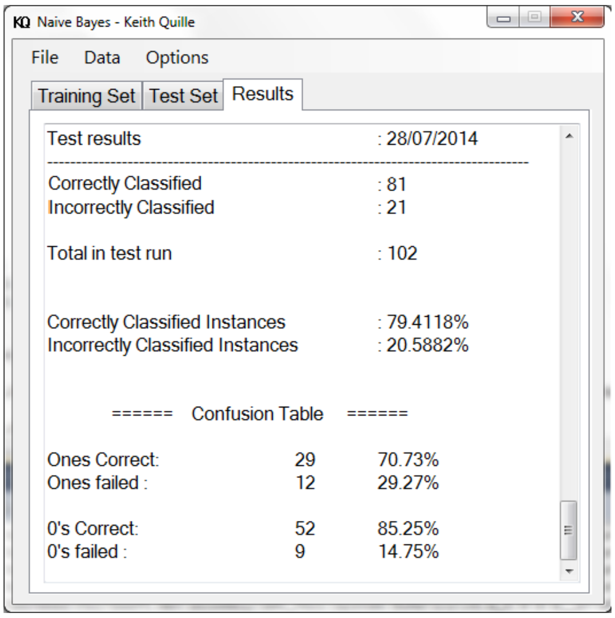
A standalone development system was used to investigate the Accord.Net naïve Bayes ability to replicate the findings of the PreSS study while also validating the PCA of PreSS itself [16][126]. In addition, as the core of this application was library based, other libraries could be used, for example with different machine learning algorithms, allowing PreSS# to implement alternative cores if later work (RG 3) identifies performance improvements using alternate techniques. This system was also developed to test input data sets varying in type and size allowing for the possibility of multiple investigations such as holdout testing and future studies. All data was classified as continuous using the Accord.net <IUnivariateDistribution> composite data type. The system also produced the probability (confidence) of each classification as shown in Figure 4.4. This was also very useful for testing the model as if both .NET and/or WEKA produced a different classification this could be used as a measure of variance between the predictions.

The results table also contained a confusion matrix, as shown in Figure 4.5 and outlined in Appendix A.3. Ten-fold cross validation (10FCV) was also implemented. 10FCV was not part of the Accord.NET framework so it had to be developed separately in this application. 10FCV was required for comparison testing of the models of WEKA and .NET as WEKA has 10FCV as a standard feature and it was the method used in the PreSS study to obtain the overall accuracy of the model [16].



```
Breakdown of predictions:
--------------------------------------------------------------------------------
Test    Result    Pred       +/-        Prob  /  Dist
--------------------------------------------------------------------------------
1       1         1                     0.859   0.141
2       1         1                     0.871   0.129
3       0         0                     0.358   0.642
4       0         0                     0.360   0.640
5       0         0                     0.005   0.995
6       1         1                     0.848   0.152
7       1         1                     0.887   0.113
8       0         0                     0.291   0.709
9       1         1                     0.530   0.470
10      0         0                     0.215   0.785
11      0         0                     0.233   0.767
12      1         0          +          0.500   0.500
13      0         1          +          0.963   0.037
14      1         0          +          0.474   0.526
15      0         0                     0.350   0.650
```

Figure 4.4: A screen shot of the naïve Bayes application developed running on the PreSS study, showing the probability break down of each classification.



```
Test results                               : 28/07/2014
--------------------------------------------------------------------------------
Correctly Classified                       : 81
Incorrectly Classified                     : 21

Total in test run                          : 102


Correctly Classified Instances             : 79.4118%
Incorrectly Classified Instances           : 20.5882%


        ======   Confusion Table   ======

Ones Correct:                    29        70.73%
Ones failed :                    12        29.27%

0's Correct:                     52        85.25%
0's failed :                     9         14.75%
```

Figure 4.5: A screen shot of the naïve Bayes application developed running on the PreSS study, showing the confusion matrix.

### 4.5.2 pca and naïve bayes integration into press#

The naïve Bayes and PCA models were integrated into the overall system as shown in Figure 4.6 (SR17, SR18). The web application can have multiple controllers for separate tasks such as authentication, roles and computation. A controller was developed to handle the interaction with the naïve Bayes and PCA models, named TestingController. A third model (class) named NBResult was also developed which was used as a reporting model for logging the naïve Bayes predictions (SR9).

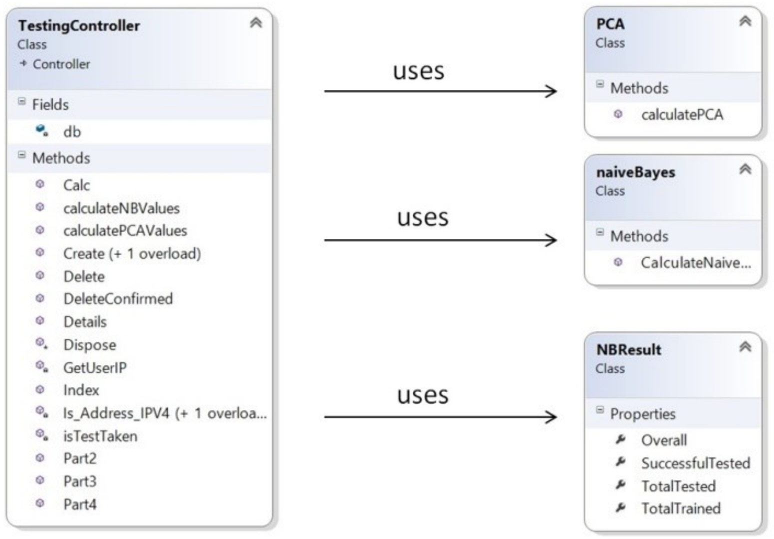


Figure 4.6: UML class diagram showing the PCA and naïve Bayes classes interacting with the Testing Controller.

### 4.5.3 press# front end

The overall development of the front end of PreSS# was broken down into four sub areas, namely: ease of use, rapid development methods, security and services. At each stage references are made to the software requirements as outlined in Table 4.1 and C.3 using the ID.

### *Ease of use*

Ease of use and device compatibility was important for PreSS#. HTML5 was used in conjunction with CSS3. The development of the front end views in HTML5 allowed the package to be dynamic on multiple device platforms. HTML5 is adapted for strong mobile integration which is a key feature for the long term success of the web system, as a large increasing proportion of current internet access is on varying mobile devices (SR13).
Razor was used to enhance HTML5. Razor is a programming syntax used to develop dynamic web pages. Razor is written in C# and is classified as a "view engine". It is written directly into the raw HTML view and is run server side before the view is returned to the client. This can eliminate the need for multiple pages by creating a single page with conditional Razor statements, for example depending on mobile or desktop requests. Razor will select different controls as it is processed on the server side before the HTML is generated allowing dynamic pages to be produced depending on the device requesting it, without having to create multiple pages with similar content. In PreSS# Razor was used extensively (SR14, SR15).

### *Rapid Development Methods*

**Razor**

One feature of models in MVC is the ability to pass a model or a list of models from a Controller into a View. Once passed into the View, Razor was able to parse the model or list of models and create controls, such as, tables using iteration, to display each of the model's properties. Razor also had the use of IntelliSense (context aware automatic code completion) which allowed for faster development (IntelliSense is aware of Models and variables in the scope of that View) in a more abstract manor. Razor was able to significantly reduce the development time needed for views (SR15).

**Data Annotations**

Data annotations are a method of describing validation rules (such as a range for values and error messages) and Database properties (such as a primary Key or Nullable field) which are written into the model as part of its properties. When the view used a model these data annotations were also used for data validation on forms in views; if the data annotation was changed in the model this filtered into every view. This method of validation removed the need for the creation of multiple additional java scripts significantly reducing development time and was a more robust method ensuring consistency for the user with every instance of the model in different views having the same validation (SR15).

**Entity Framework**

The Entity Framework consists of an object-relational mapper that works with models (POCO objects) and translates the models to relational data storage. A class named DB was developed which contained properties (collection of the models), which in turn translate to the tables in the physical database; this is shown in Figure 4.7. The DB class is the Entity access control to the live Microsoft SQL Server database. If no database exists either in development or at release, Entity will create a database matching the schema outlined in the collection of models in the DB class thus allowing the focus to be on the development of the application and models, and not the SQL database development, saving a considerable amount of development time (SR15).

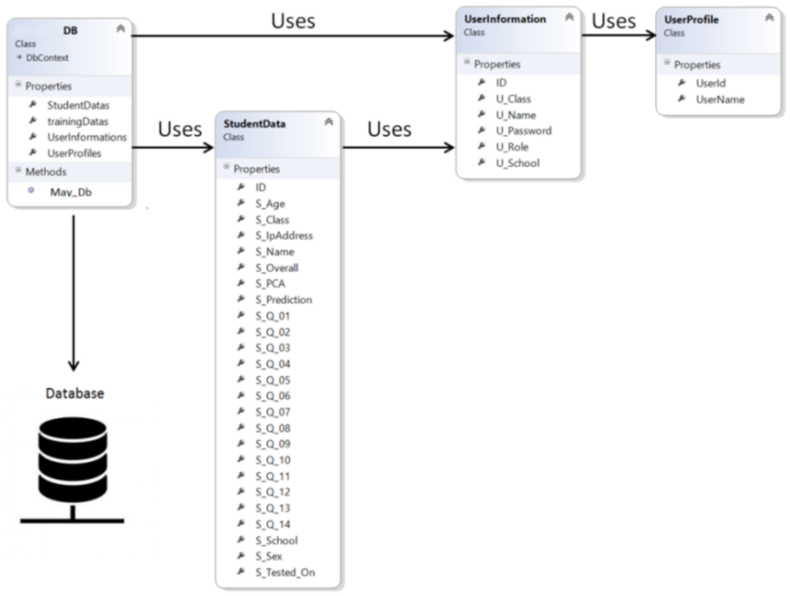


Figure 4.7: Illustration of Entity class and the Models (properties) used.

**LINQ**

In the Controllers LINQ (Language-Integrated Query) was used in preference to using SQL (Structured Query Language). LINQ consists of a native C# data querying language and was able to query the list of models using native C# code replacing the need for SQL statements. LINQ was integrated with IntelliSense and was able to produce run time compile errors, two features that SQL does not exhibit, thus leading to significant gains in terms of development time as well as debugging time (SR15).

### *Security*

**Authentication**

Authentication was handled by the ASP.NET web security model, which validates and manages users and roles. The model was modified to accommodate additional properties required for the web system such as school and institution details. The model was also used for access control to controllers which could be sub-divided by user roles, which allowed for hierarchical access control for users in specific roles. A single view could also be customised for different roles, both contributing to a faster development of a secure and robust system (SR5).

**Encryption**

Password data is hashed by the .NET web security and a cryptography class was developed to encrypt all the data stored in the system using the AES standard with a 256 bit key. The class uses a salt function to prevent standard dictionary attacks on the data. All data stored in the database was encrypted using this class. The data in transit is encrypted by Microsoft IIS7 using HTTPS with SSL. (SR2, SR4).

**Attack Prevention**

To address automated brute force log in attacks, a three attempt system was developed. The user after three failed log in attempts is locked out of the system for ten minutes (SR3). The IP addresses of each student surveyed were also logged (SR1). Cross site scripting attacks (XXS) were addressed with HTML encoding using Razor in all HTML input and retrieval fields, so if a malicious Java Script is injected into the site or database only the raw HTML is displayed making the script redundant when it reaches the browser (SR3). Cross site forgery requests (CSFR) were also addressed by sending two independent random tokens to the view, one as a cookie and the other token was randomly hidden in a field on the form. On the submission of the form both tokens are required by the controller, otherwise the server rejects the request (SR3).

### *Services*

Three roles were created: (i) administrators, (ii) lecturers and (iii) students, each with different levels of functionality (as shown in the use case diagram in Figure 4.2) (SR10)

- **Administrator:** Can create users on the system, delete users, modify a user's details, reset passwords, upload training set(s), run prediction analysis and observe the survey (SR6, SR7, SR8, SR12)
- **Lecturer:** Can see surveys taken, allow students to retake a survey if needed, and can run prediction analysis with a limited report (SR12).
- **Student:** Can only sit a survey once and cannot redo it unless sanctioned by a teacher; as shown in Figure 4.2(SR11)

Administrators can upload users in batch mode via a .CSV file (SR6). The system validates the data upon uploading and returns a log of each entry, containing details of its success or failure (and the cause of any validation error). This bulk upload of users would be very beneficial for large institutions. The training data set can also be bulk uploaded via a similar function.

## 4.6 TESTING

The next phase was to test and validate the PreSS# web-based system itself. For this testing phase the focus was on user interactions and validation of those actions to ensure that the web based system features and actions passed all test cases. Both alpha and beta test cases were performed. The alpha involved white box testing at each phase of development. The beta investigation was a pre-pilot study conducted in the 2013-14 academic year which consisted of 34 users using multiple device types; this will be expanded upon further in the following two sections.

### 4.6.1 alpha testing

White box alpha testing was completed at each stage of the system development. Tables 4.3 and 4.4 show the twelve main test cases which consisted of class, API and method testing. As both the PCA and naïve Bayes classes were previously examined they are not considered in these test cases. As mentioned previously, an agile approach was used, where the test cases were completed iteratively throughout the development to allow each phase to be validated and the next phase built upon this validated phase.

Table 4.3: Results from alpha testing test cases - Part 1

| **ID** | **Description** | **Result** | **Pass / Fail** |
|---|---|---|---|
| 1 | Security – IP address capture | Address stored in DB, even when Proxy used. | Pass |
| 2 | Security – Encryption | All data in DB encrypted, salt functionality working | Pass |
| 3 | Security – XXS, CSFR and brute force | CSFR and brute force failed, XXS raw displayed | Pass |
| 4 | Security – HTTPS (SSL in IIS7) | HTTPS enabled, on all browser types. | Pass |
| 5 | Security – Authentication | Log in successful with all three roles, each action restricted or allowed specific roles. | Pass |
| 6 | User - User creation | User creation with .CSV upload with validation. | Pass |

Table 4.4: Results from alpha testing test cases - Part 2

| ID | Description | Result | Pass / Fail |
|---|---|---|---|
| 7 | User - Reset user password | Reset password for user, admin cannot see current password as per ADDC standard practice. | Pass |
| 8 | User - Training data | Single training data set creation with validation, .CSV bulk upload with validation on all aspects that include file type and missing values. | Pass |
| 9 | User - Prediction output | Predictions completed using PCA and naïve Bayes classes, compared and validated using the two .NET applications. | Pass |
| 10 | User Roles: student, lecturer & admin | Each role was tested; no user had access to any control or action even with URL injection. | Pass |
| 11 | User - Reset a survey | Survey reset for student allowing resit. (admin and lecturer only) | Pass |
| 12 | Device - Submit survey as student on multiple devices | Survey submitted successful in every browser on every device. | Pass |

### 4.6.2 BETA TESTING

Black box beta testing was carried out using a small group of 34 volunteering students. This beta test was completed to test and validate the system using a multitude of devices which included PC's, iPhone, Android phone, iPad and a Surface tablet. Testing was completed in April 2014. The results of this testing are outlined in Table 4.5. Each test case passed successfully.

Table 4.5: Results from beta testing test cases

| **ID** | **Description** | **Result** | **Pass / Fail** |
|---|---|---|---|
| 1 | Security – IP address capture | Address stored in DB, even when Proxy used. | Pass |
| 2 | Security – Encryption | All data in DB encrypted, salt functionality working | Pass |
| 3 | Security – XXS, CSFR and brute force | CSFR and brute force failed, XXS raw displayed | Pass |
| 4 | Security – HTTPS (SSL in IIS7) | HTTPS enabled, on all browser types. | Pass |

## 4.7 REPLICATION OF PRESS

### 4.7.1 OVERVIEW

PCA and naïve Bayes replication testing was carried out to investigate if both could replicate the results of the PreSS study. If the replication was a success the .NET implementation of PreSS could then be confidently integrated into the web-based system.

### 4.7.2 pca replication of results

The PCA implementation was based on that used in the original PreSS model and a tutorial by Souza [127] as described in Smith's tutorial paper [125]. Examples in Smith's paper were used to compare with the PCA class in .NET. In Smith's study, the results were first calculated on paper and then confirmed in Scilab [125], a freeware alternative to MATLAB. Even though the languages differ, the calculations/pre-steps and most importantly the results that Smith computed, would be used to compare the accuracy of the PCA model developed. Souza's tutorial was based on the accord.NET framework and successfully replicated the results of Smith. The aim was to replicate Smith's result and confirm the accuracy of the PCA class.

The .NET implementation results were compared with the results of Smith using the same input data. The results were replicated to the ninth decimal place as shown in Table 4.6. These results were identical to that of both Smith and Souza. The replication of results indicated that the PCA model was statistically the same when using the .NET implementation.

Table 4.6: The Results from Smith's study and from the .NET PCA system

| **PCA Results** | |
|---|---|
| **Lindsay I Smith** | **.NET System** |
| 0.827970186 | 0.827970186 |
| -1.777580325 | -1.777580325 |
| 0.992197494 | 0.992197494 |
| 0.274210416 | 0.274210416 |
| 1.675801419 | 1.675801419 |
| 0.912949103 | 0.912949103 |
| -0.099109438 | -0.099109438 |
| -1.144572164 | -1.144572164 |
| -0.438046137 | -0.438046137 |
| -1.223820555 | -1.223820555 |

### 4.7.3 NAÏVE BAYES REPLICATION OF RESULTS

Table 4.7 shows the naïve Bayes application (labelled .NET) and WEKA's prediction accuracies along with sensitivity (the true positive rate) and specificity (the true negative rate) [148]. Both systems used the dataset from the original PreSS study with all of the 102 data samples as the input and 10FCV to produce the results in Table 4.7.

Table 4.7: Results from naïve Bayes and WEKA applications using 10FCV on the PreSS study.

| **Model** | **Accuracy %** | **Sensitivity %** | **Specificity %** |
|---|---|---|---|
| .NET | 80.39% | 81.53% | 78.38% |
| WEKA | 80.39% | 80.05% | 80.00% |

The results from Table 4.7 show that .NET had an accuracy of 80.4%, identical to the accuracy of WEKA. A two tail *t*-Test for a binomial distribution was run to confirm both applications prediction accuracies were not statistically different. No significant statistical differences were found. The *t*-Test results with a 95% confidence level were: $p_{\text{(value)}}$ = 1.0 and a $t_{\text{(value)}}$ = 0.0, showing that the WEKA results were not statistically different to the .NET results. Sensitivity and specificity were also recorded, but as it was felt that sensitivity was the most important factor for PreSS# (inferring that correctly predicting a weak student is more significant and takes precedence over correctly predicting a strong student) this was the examined measure.

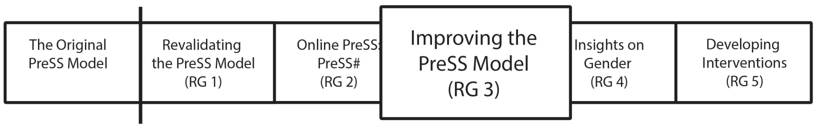
A *t*-Test was completed on sensitivity and specificity in both .NET and WEKA. The sensitivity between .NET and WEKA concluded with no statistically significant difference and the following result: $p_{\text{(value)}}$ = 0.09 and a $t_{\text{(value)}}$ = 1.6. The specificity between .NET and WEKA concluded that there was a statistical difference with the following result: $p_{\text{(value)}}$ = 0.006 and a $t_{\text{(value)}}$ = 2.81. However with the raw sensitivity slightly higher with the .NET application, specificity was going to be lower, as the accuracies were identical. Finally as sensitivity was the valued measure for this project, it was positive to see it perform slightly higher than WEKA.

## 4.8 SUMMARY

PreSS# is a completed functioning tool, that is ready to be integrated into the educational domain and to begin the data collection for the main study. This tool will be invaluable for the early detection of students who may be at risk of failing CS1. The tool may also expedite the development of educational methodologies, both for students who are likely to fail and for students who are likely to be very strong, hence developing methodologies for differentiation. This differentiation and early detection of weak and strong students may lead to effective interventions that may be applied faster than current efforts, thus enabling students to possibly achieve at a higher rate than predicted and increase the progression rates of computer science. In addition, as this research progressed, the plug and play nature of the classifier and PCA classes, allowed for examination and collection of additional data, without full scale redevelopment.

5

# ADDITIONAL FACTORS


The Original PreSS Model
Revalidating the PreSS Model (RG 1)
Online PreSS: PreSS# (RG 2)
Improving the PreSS Model (RG 3)
Insights on Gender (RG 4)
Developing Interventions (RG 5)

## 5.1 introduction

This chapter documents three pieces of work. The first piece of work is an investigation on factors from the original PreSS study dataset, which were not used in the final (2006) 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. This work is presented in Section 5.2.

The second piece of work is an investigation of additional factors collected during the justification study, as presented in Chapter 3.1. This was to determine if incorporating some or all of these new factors could improve the accuracy of PreSS. This work is presented in Section 5.3.

The third piece of work, incorporates the most predictive factors from the two prior pieces of work to examine if the newly identified factors in combination or substitution with the original PreSS model factors could improve the model's performance. This work is presented in Section 5.4.

The first two pieces of work described in this chapter were published within one article in the 27th annual workshop of the Psychology of Programming Interest Group (PPIG, Cambridge University, Cambridge, UK) [99]. The third phase of this work was published at the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE'18, Larnaca, Cyprus) [100].

## 5.2 FACTORS FROM THE ORIGINAL PRESS STUDY

A total of 123 students participated in the original PreSS study (pilot, main and epilogue studies). The study also recorded 113 factors. Five instruments (in the form of questionnaires) were used to collect data: background, comfort-level [145], programming self-efficacy (Appendix E), motivation and learning strategies [93]. These 113 recorded factors consisted of questions (from the five instruments) and normalisation or summations/differences of combinations of questions. To evaluate which factors might be the most useful for predicting performance standard deviation and variance were calculated for each factor.

Not all participants had recorded data for each of the 113 factors (due to the manual data collection technique and missing data). For comparison purposes and to ensure an unbiased statistical comparison, the original PreSS model was re-run only using the students that had provided data for the factors investigated. A Student's t-test was used to examine if a significant difference was found in accuracy between the baseline accuracy and the updated model's accuracy. This piece of work presents the top eight models and their factors that may have value in an updated PreSS model (further models and details are provided in the related publication) [99]. The results are presented in Table 5.1.

Table 5.1: The 8 models that resulted in a significant gain in accuracy with the 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 *** |
| 4 | Mathematical grade, programming self-efficacy, hours spent playing computer games and a student's anxiety, fear and/or uneasiness of examinations and assessment, and a focus on the negative while completing them. | 68 | 2.94 | <0.001 *** |
| 5 | Mathematical grade, programming self-efficacy, hours spent playing computer games and a student's feeling on their own ability, on programming concepts, design of programming logic and completion of lab assignments. | 110 | 1.82 | 0.0010 *** |
| 6 | Mathematical grade, programming self-efficacy and the number of hours a week the student worked in a part time job. | 111 | 1.81 | 0.0024 ** |
| 7 | Mathematical grade, programming self-efficacy, hours spent playing computer games and a student opting for more challenging / relevant course material when given a choice, even at the cost of a higher grade if less challenging material was chosen. | 68 | 1.40 | 0.010 * |
| 8 | Mathematical grade, programming self-efficacy, hours spent playing computer games and a belief the student has, that if they work hard that they will succeed and if they do not work, that they will fail and it will be the student's own fault that they did. | 68 | 1.40 | 0.010 * |

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

## 5.3 FACTORS FROM THE JUSTIFICATION STUDY

This section examines 12 additional factors collected during the justification study as discussed in Chapter 3.1 [99] (Appendix D). From the additional 12 factors in combination and substitution with the original PreSS factors, several new models were developed and investigated. With the use of PreSS# for data collection, no student had missing or dirty data, so all 26 students from the second justification study (2015) were included in each experiment (as the additional factors for the first justification study was collected towards the end of the introductory module, it was not used).

From the developed models, three factors emerged, that added value to the PreSS model: hours spent on social media, gender and age. In the original PreSS study, it was suggested that gender may be of value to the model, thus gender was incorporated into the additional factors recorded in the two justification studies. It was also hypothesised that age may positively affect the accuracy of PreSS. This was based on prior experience that mature students may rate their programming self-efficacy lower than its true value, resulting in an incorrect prediction and the work of Glorfeld who identified age as a significant prediction factor ([52]). Research has shown that there is a positive relationship between age and programming ability/attainment [83]. The use of social media was hypothesised as a new factor for programming prediction. This was based on the significant average time spent on social media as identified in the pre-survey (Appendix D). Distinct differences were found in the social media habits between mature and non-mature students. The same data reduction techniques, and performance evaluation as used in the original PreSS study were implemented. A Student's t-test was used to examine if a significant difference was found in accuracy between the baseline result and the experiment result.

### 5.3.1 RESULTING MODELS FROM THE JUSTIFICATION STUDY

When age was added as a factor (as identified by Glorfeld [52]), represented in years, or dichotomously, it produced statistically significant increases in accuracy over the original PreSS model. Although the addition of gender (in dichotomous form) did not provide an increase in accuracy when introduced into the original PreSS model, it yielded a statistically significant increase in accuracy at ~ 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 interesting outcome. When time spent on social media was substituted for the programming self-efficacy factor, it resulted in an identical accuracy as the original PreSS model, thus showing a correlation or perhaps that both factors may be measuring the same underlying phenomenon. The models incorporating the new factors are presented in Table 5.2.

Table 5.2: 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 *** |
| 4 | Mathematical grade, hours spent playing computer games and the amount of time spent on social media. | 26 | 0 | 1 |

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

## 5.4 INVESTIGATING THE NEW FACTORS USING THE MAIN STUDY

### 5.4.1 INTRODUCTION

From the investigations outlined in 5.2 and 5.3, 16 updated models were identified using a multitude of factors identified in this thesis and in the original PreSS study. These were then added to the PreSS# survey and captured during the main study from this thesis. In total, 39 factors (inclusive of the original PreSS model factors) were recorded in the PreSS# survey during the main study of this thesis. This resulted in (after data reduction) 17 factors that could be examined as part of an updated PreSS model, using the main study dataset. The factors collected as part of the main study are presented in Table 5.3.

Table 5.3: The 17 factors included in the main study (after data reduction)

| **Factor Name** | **Factor Details** |
|---|---|
| Institution type | University, Institute of Technology, Community College {1, 2, 3} |
| Time to complete the survey | Seconds |
| Age | Years as an integer |
| Mature Student | Dichotomous |
| Gender | Dichotomous |
| 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 using Appendix C |
| 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 PCA |

### 5.4.2 FACTOR SELECTION

Two factor selection algorithms were chosen to select the most useful factors for prediction: correlation evaluation and information gain [148]. Both of these algorithms were run on the data set and combinations of the highest ranked factors were examined.

### 5.4.3 UPDATED MODELS

A multitude of models were investigated with the top two presented in this chapter. 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, with performances reported in Table 5.4:

#### RECOMMENDED MODEL 1

Model 1 included: student's age in raw integer form, a student's self-reported 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 and one of the highest sensitivities (an increase of 4%).

#### 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 (as per Appendix C) 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 and the model's sensitivity was higher than model 1 (an increase of 2%).

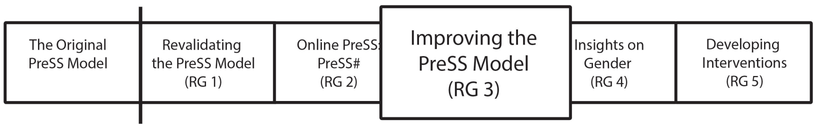
Table 5.4: Recommended PreSS updated models, based on factors developed from recent research

| Model | Factors | Accuracy | Sensitivity | Specificity |
|---|---|---|---|---|
| 1 | Age (years)<br>Expected end of year result<br>Mathematical grade<br>Programming self-efficacy | 71% | 75% | 66% |
| 2 | Institution type<br>Mature student<br>Mathematical grade<br>Programming self-efficacy | 69% | 80% | 54% |

## 5.5 SUMMARY

The results presented in this chapter are very positive. The increases in performance show that PreSS may be able to produce further performance gains. The final recommended models, (model 1 and model 2) are gender, age and institution independent. With updated models developed, this work contributes to RG 3 (further developing the PreSS model). Next, the core machine learning algorithms are explored (Chapter 6), as a capstone to RG3 and improving the model.

# 6 MACHINE LEARNING TECHNIQUES


The Original PreSS Model
Revalidating the PreSS Model (RG 1)
Online PreSS: PreSS# (RG 2)
Improving the PreSS Model (RG 3)
Insights on Gender (RG 4)
Developing Interventions (RG 5)

## 6.1 introduction

Given that new factors have been identified which resulted in higher performance (model 1 and 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 new algorithms, may improve the PreSS model further. This chapter is divided into two investigations, machine learning algorithms and artificial neural networks.

## 6.2 method

### 6.2.1 machine learning algorithms

This section first presents the findings of the original PreSS study, where six machine learning algorithms were compared, as a starting point for this section of work [16]. Then using the updated PreSS models 1 and 2 (Section 5.4.3), this investigation is repeated (using the main study data and newly developed models including updated factors), comparing the performance of six machine learning algorithms.

### 6.2.2 ARTIFICIAL NEURAL NETWORKS

An investigation of deep artificial neural networks was conducted. While the algorithms did exist in 2006 (and the original PreSS study used a single layer artificial neural network), access to the resources and computational power required to conduct deep learning was not as accessible as it is today, thus hampering the use of these algorithms over a decade previous. The techniques employed and the results are presented in detail in Sections 6.5 and 6.5.4.

## 6.3 MACHINE LEARNING ALGORITHMS

The algorithms discussed in this section and in the findings, sought to implement a blend of algorithms using diverse machine learning and artificial neural network techniques to determine if further improvements could be made to PreSS#.

### 6.3.1 LOGISTIC REGRESSION

Logistic regression is a statistical technique to predict a discrete outcome, such as group membership from a set of variables. The dependent variable does not need to be linearly related to the independent variables, homoscedasticity is not required nor do the variables need to be normally distributed. The independent variables can be continuous, discrete or dichotomous. It is a particularly useful technique when there is a non-linear relationship between the dependent variable and one or more of the independent variables [130]. The standard representation for logistic regression is given by Figure 6.1:

$$P_i = \frac{1}{1+e^{-b_0+b_i+X_i}}$$

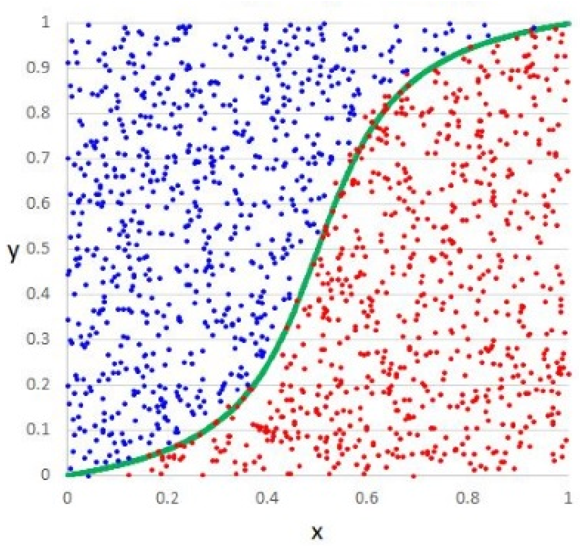


Figure 6.1: Logistic regression formula and visual representation of the algorithm.

### 6.3.2 K-NEAREST NEIGHBOUR

K-Nearest Neighbor (KNN) is an instance-based learning technique. This type of learning is 'lazy' as it defers generalization until the classification stage. The nearest neighbour algorithm is based on the principal that the properties of any particular instance are likely to be similar to those instances within its neighbourhood. Each new instance is compared with existing ones using a distance metric and the new instance is classified based on the majority class of the nearest K neighbours. Typically Euclidean distance is used and is presented in Figure 6.2.

$$E_d = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

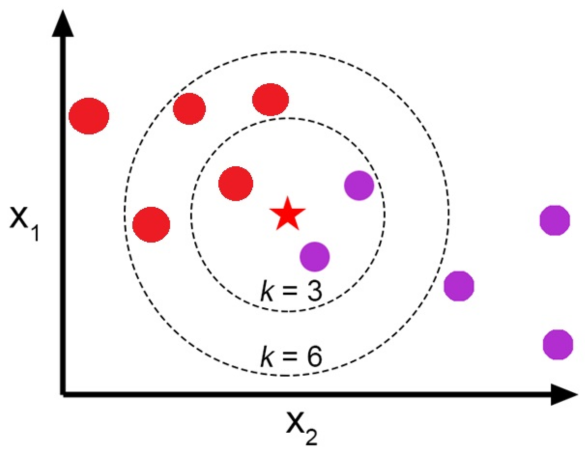


Figure 6.2: KNN formula and visual representation of the algorithm.

### 6.3.3 BACKPROPAGATION

Backpropagation is a learning algorithm that can be used to train multi-layer feed-forward networks. In the backpropagation learning process one of the training instances is applied to the network, and the network produces some output based on the current state of its weights (initially the output will be random). This output is compared to the target output and an error signal is calculated. The total error, $E$, over all of the network output units is defined in equation 6.1:

$$E = \frac{1}{2}\sum_{d \in D}\sum_{k \in outputs}(t_{kd} - o_{kd})^2 \tag{6.1}$$

Where $D$ is the set of training examples, outputs is the set of output units in the network, $t_{kd}$ and $o_k d$ are the target and output values for the $k$ th output unit for training example $d$ . The error value is propagated backwards through the network, and changes are made to the weights in each layer. Weights can be updated after every input-output case and therefore no separate memory is required for the derivatives. Figure 6.3 presents a single perception (neuron) in an Artificial Neural Network (ANN):

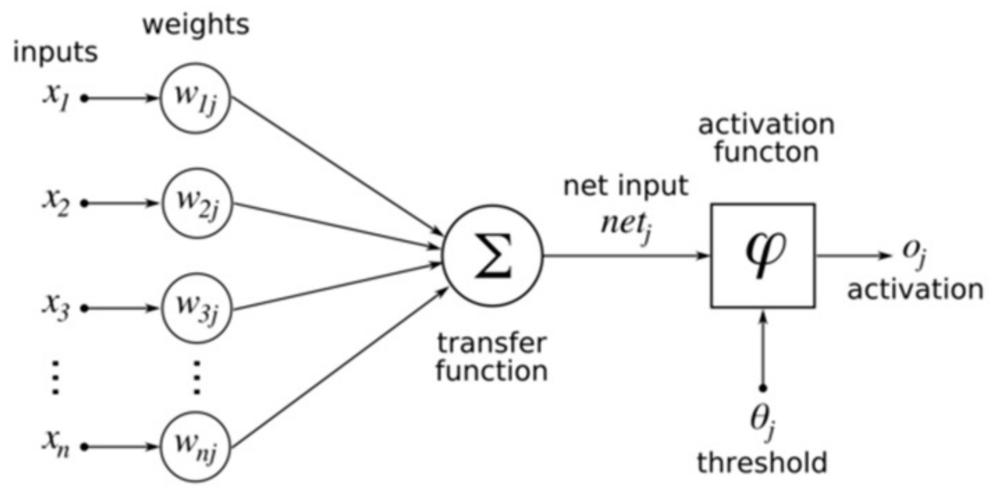


Figure 6.3: Single perceptron in ANN

The whole process is repeated for each of the training instances and the cycle is repeated until the overall error value drops below a pre-determined threshold, in this case using stochastic gradient descent. Section 6.4 presents a further investigation of artificial neural networks, using deep learning and convolutional neural networks.

### 6.3.4 NAÏVE BAYES

Naïve Bayes is a non-parametric probabilistic model based on an assumption of conditional independence among variable attributes. Although this assumption is often violated, naïve Bayes classifiers have been shown to work surprisingly well and often have comparable prediction performance even when compared with some state-of-the art classifiers [81, 148]. A naïve Bayes classifier is denoted by Equation 6.2:

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{p(x)} \tag{6.2}$$

Where $P(x \mid c)$ = posterior probability, $P(c \mid x)$ = likelihood, $P(c)$ = class prior probability and $P(x)$ = predictor prior probability.

### 6.3.5 DECISION TREES

Decision Trees involve the recursive partitioning of a dataset. An attribute is selected to place at the root node and a branch is created for each possible value. This process is repeated recursively for each branch, using only those instances that reach the branch. If all instances at a node belong to the same class, no further partitioning is performed. C4.5 is a popular decision tree algorithm, based on ID3 but contains several improvements, such as handling continuous attributes and measures for choosing an appropriate attribute selection scheme [81, 113, 148]. Figure 6.4, illustrates a decision tree.

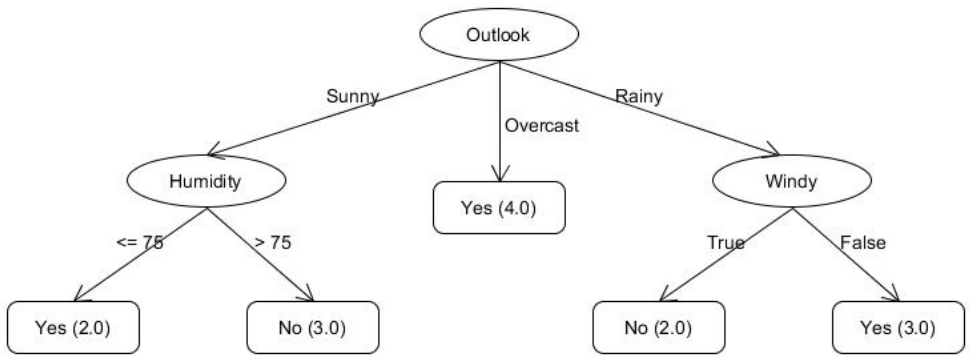


Figure 6.4: A visual example of a decision tree

### 6.3.6 SUPPORT VECTOR MACHINES

Support Vector Machines (SVMs) are a relatively new generation of learning system based on advances in statistical learning theory [116]. The principal idea behind linear SVMs is the optimal hyperplane. During the generation of a discriminant function, standard techniques such as the perceptron will stop as soon as the last sample is classified without error. This provides a quick but potentially poor solution as it leaves the separation surface very close to the last sample classified. This will classify all the data in the training set correctly but may provide poor generalisation. To counteract this problem the linear SVM learning algorithm is modified so that the hyperplane is positioned in an optimal location between the two classes. To do this a conceptual margin is used. The margin is the perpendicular distance between the closest vector to the hyperplane and the hyperplane itself. The optimal hyperplane is the one that maximises the margin [56]. Suppose we have a dataset $(x_1, y_1), .......(x_m, y_m) \in X \times \{\pm 1\}$ where $X$ is some space from which the $X_i$ have been sampled. The optimal hyperplane can be found by solving the dual form Lagrangian (Equation 6.3) and visual hyperplanes are presented in Figure 6.5:

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \bullet x_j) \tag{6.3}$$

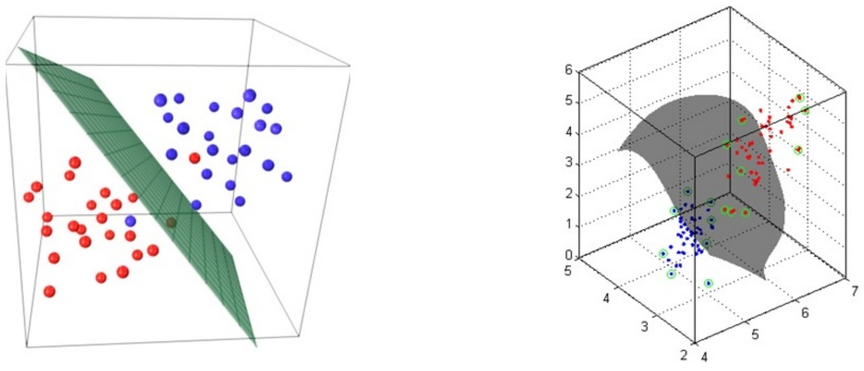


Figure 6.5: SVM example hyperplanes (linear and non-linear)

## 6.4 COMPARING MACHINE LEARNING ALGORITHMS

### 6.4.1 ORIGINAL PRESS STUDY

This section presents the findings of the original PreSS study (and a recent follow up study), which examined the performance of machine learning algorithms, using the original PreSS study data set. Initially, a study in 2004 by Kotsiantis [69], first compared a multitude of algorithms for a distance learning course. Using the original PreSS factors, the performance measures are presented in Table 6.1. Both the original

PreSS study and Kotsiantis found naïve Bayes to be the strongest performer, and k-nearest neighbour to be the weakest [69]. This was later followed up in a study [19], which re-examined the findings and conducted additional analysis (ANOVA tests with Tukey post-hoc analysis, [130]), this study re-affirmed the naïive Bayes as the most suitable algorithm for PreSS.

Table 6.1: Performance of machine learning algorithms from the original PreSS study

| **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 |

## 6.4.2 UPDATED PRESS MODELS

The algorithms were re-examined, but this time using the recommended models as developed in Chapter 5.4. For analysis of both models, ANOVA analysis incorporating a Tukey HSD post-hoc test was implemented.

*Recommended Model 1*

Model 1 as presented in Table 6.2, reported that for accuracy the first three algorithms, naïve Bayes, SVM and Logistic Regression, produced statistically similar performance. Algorithms C4.5 and Backpropagation produced results that were significantly lower than the first three, while k-nearest neighbour was the lowest performing algorithm. For sensitivity however (the prominent measurement of PreSS), naïve Bayes was lower (statistically significant) than SVM and Logistic Regression.

Table 6.2: Performance of machine learning algorithms for Model 1

| **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 |

*Recommended Model 2*

For Model 2, the algorithm that produced statistically poorer performance was k-nearest neighbour. For sensitivity, naïve Bayes was statistically significantly higher than all other algorithms. Thus one could argue that naïve Bayes is still the strongest performing machine learning algorithm. The results are presented in Table 6.3.

Table 6.3: Performance of machine learning algorithms for Model 2

| **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 |

## 6.5 artificial neural networks

### 6.5.1 environment

An investigation of artificial neural networks (ANNs) is presented in this section using the Python programming environment [108]. The machine learning library that was used to develop the neural networks was TensorFlow [1]. The Keras high level neural networks API [31], was also used on top of TensorFlow, the standard in applied deep learning frameworks [24].

### 6.5.2 HYPER-PARAMETER TUNING

ANNs 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 thesis for developing educational data mining (EDM) ANN, which usually consist 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 [148].

- 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 6.4. The optimizers selected were based on their default parameters, such as learning rate and momentum, provided by Keras [31, 66, 132]. The initializers were selected based on distribution: normal and uniform. The epoch selection was based on applied practices [24]. The batch sizes were selected from best practice [24], 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*).

Table 6.4: 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] |

Once the grid search was completed, the strongest performing network was selected and drop-out regularization was added (20% [24]), to each layer of the network. Then, 10 fold-cross validation was used to obtain an accuracy for the model.

## 6.5.3 NETWORK TOPOLOGIES

Three network topologies were selected. There is a large amount of topologies to select from, thus for this initial investigation, three fundamental network configurations were selected [24]. 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 (often referred to as a CNN) were selected as starting

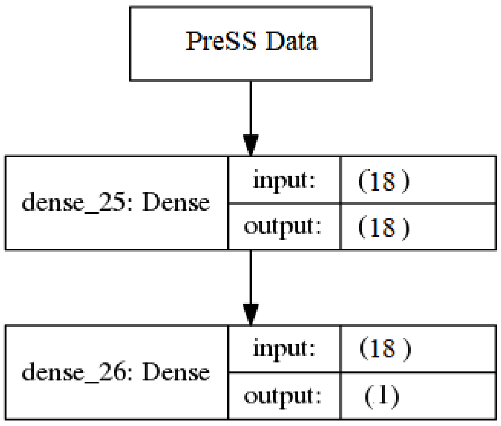


Figure 6.6: Network topology: Single Layer ANN

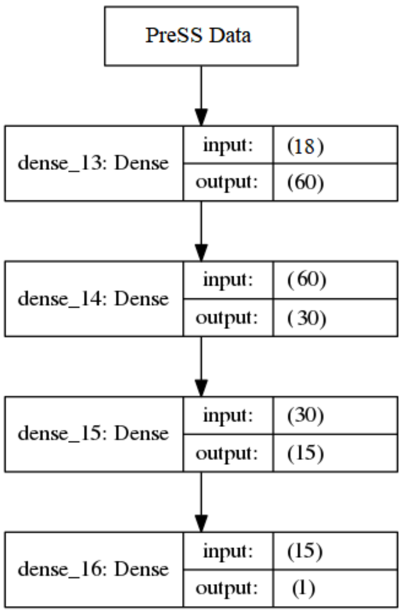


Figure 6.7: Network topology: Deep Learning ANN

points, as this is the initial investigation to justify (if any) future investigations may be worth while. To present the topologies in detail, the Keras environment allowed for the topologies to be visualized and these are presented in Figures 6.6, 6.7 and 6.8. 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. It is acknowledged that the network topology selection is by no means exhaustive and represents an initial starting point to examine if performance gains are plausible using ANNs.

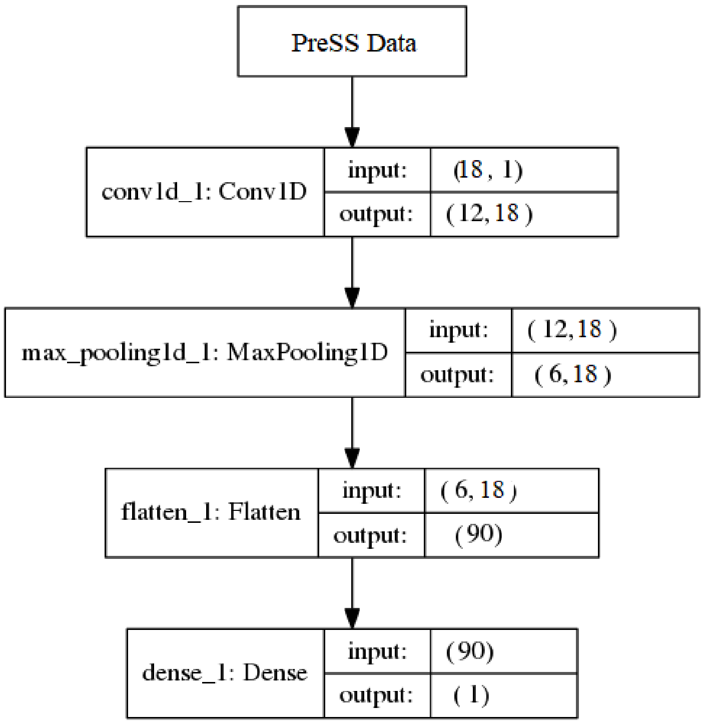


Figure 6.8: Network topology: Convolutional ANN

### 6.5.4 RESULTS

Table 6.5: Performance of deep learning ANN's.

| **Algorithm** | **Acc %** | **Sen %** | **Spec %** |
|---|---|---|---|
| Single layer ANN | 67 | 80 | 51 |
| Deep Learning ANN | 69 | 83 | 51 |
| Convolutional ANN | 66 | 87 | 38 |

Tables 6.2, Table 6.3 & Table 6.5 reveal that the accuracies of the top performing algorithms are very similar (with the exception of naïve Bayes in Table 6.2, 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 per-

formance 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.

## 6.6 summary

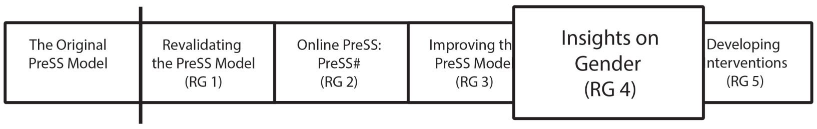
An investigation of machine learning algorithms reported that naïve Bayes was still the strongest performing algorithm in terms of accuracy and sensitivity. However for recommended model 2, SVM and Logistic Regression outperformed naïve Bayes in terms of sensitivity. This is an important finding, as sensitivity is such an important measure in this work.

An investigation using artificial neural networks was also performed. An approach was developed for ANN model development for educational data mining (Section 6.5.2) that differed to usual deep learning practice. This may provide greater generalizability for educational data mining prediction models for this work and in future projects. While the results of the ANNs did not provide an increase in performance (in fact a drop of 2-4%) a significant increase was found in sensitivity. With the deep learning ANN sensitivity at 83% and for the CNN, sensitivity was 87%. This means that the deep learning ANN and the CNN were significantly stronger at predicting students who were likely to fail or drop out. This bodes very well for the use of artificial neural networks in predicting success in CS1 and integration into PreSS.

This chapter concludes along with the previous chapter (Chapter 5), addressing RG 3, recommending two updated PreSS models. This chapter also recommends the same machine learning algorithm (naïve Bayes) as the original PreSS model, but suggests future work with deep artificial neural networks and convolutional neural networks.

7

# INSIGHTS INTO GENDER DIFFERENCES IN CS1


The Original PreSS Model
Revalidating the PreSS Model (RG 1)
Online PreSS: PreSS# (RG 2)
Improving th PreSS Model (RG 3)
Insights on Gender (RG 4)
Developing nterventions (RG 5)

## 7.1 INTRODUCTION

CSEd is often a male dominated discipline, with declining female enrolments [27, 87, 139]. In the late 1970's and early 1980's female enrolment increased but subsequently declined again [27]. On average, Irish enrolment of female students in CS courses is currently around 20% [77] and this can be seen in many other western regions with the U.S. seeing a significant decline from 28.3% in 1993 to 18.2% in 2012 [87]. Increasing the number of female students is vital, given the growing need for CS graduates. Research leading to a better understanding of the factors that influence female students studying CS is timely, necessary and valuable. The work described in this chapter was published as part of the proceedings for the 2017 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE), 2017 [103].

### 7.1.1 MOTIVATION

During the original PreSS study, gender was used in a multitude of models and the study found that models built using gender were more predictive, with some factors more indicative for male or female students. Gender as a factor was not included in the original PreSS model, as the model aimed to be as generalizable as possible (for both male and female students). The investigations outlined in Chapter 5 provided further evidence on the importance of gender (from Table 5.2), increasing prediction accuracy by ~ 4%.

A hypothesis was presented that using gender dichotomously may not be fully representative of underlying gender specific phenomena, as in this thesis and in the original PreSS study, it did not generalize well outside of gender specific models. This chapter aims to address RG 4's goal of examining differences between male and female students in CS1.

### 7.1.2 LITERATURE

Several studies are documented in the literature on gender differences in CS. The most prominent repeated finding appears to be the lower programming self-efficacy of female students [7, 20, 75, 105, 124]. Programming self-efficacy has been shown in multiple studies to significantly correlate to success [7, 16, 18, 97]. The effect of self-efficacy is not just prominent in CS, but also in mathematics where male student's self-reported self-efficacy was significantly higher than that of females [105, 124]. A recent study examined the time spent on social media early in CS1, and found the time in hours negatively correlated to success in introductory programming [99]. Social media for the student's personal use and as an educational tool has become a much-debated topic in education for many years. Several studies have reported correlations between time spent on social media and self-esteem (to which self-efficacy is related), some focusing on gender specific correlations [6]. Some studies have found that male students outperform female students during early stage programming tasks/exams [59, 105]. The early difference in performance could be directly related to programming self-efficacy: female students have been shown to take longer on programming tasks, thus possibly inhibiting a true reflective result on programming ability [55, 115]. This may also contribute to the longer time required to complete programming tasks. A noteworthy finding was that female and male students did not differ in science self-efficacy [41]. This may be due to the high male to female ratio in CS courses, with many studies reporting that female students feel out of place, or that they are minorities within their CS classes [105], which can lead to female students experiencing stereotype-threat [20, 85, 105]. Furthermore, female students have been shown to have far less confidence when it comes to asking questions in a CS class [105].

While examining the literature it became apparent that it is difficult to generalize on the findings. The studies were predominantly based on a single institution [7, 20, 50, 54, 59, 70, 75, 90, 105, 111, 112, 115, 129, 146] (with a single programming language and teaching and assessment methodology). In addition, several of the studies had small sample sizes ($n < 96$) [20, 50, 54, 85, 112, 115]. Where sample sizes were large and/or included multiple institutions, the studies were often conducted towards the end of CS1 or later. This mimics the research literature of prediction models in CS1 (Chapter 2). Timely data is required (early on in CS1) if uptake and attrition rates are to be addressed so that appropriate recruitment strategies and early interventions can be developed. This is particularly important given that some studies have found that gender differences are less temporally stable than expected [21].

No study could be found on gender during the early stages of CS1, that used a large multi-institutional student cohort, while simultaneously examining multiple gender differences. The study presented in this chapter makes two significant contributions to the CS education community. First, the study examines a multitude of gender differences, early in CS1, using the main study from this thesis. Second, it allows educators to generalize on the findings, as they are arguably institution, academic level, programming language and teaching/assessment methodology independent.

## 7.2 DATA COLLECTION

The data was collected as part of the main study of this thesis, as detailed in Chapter 3.2 using PreSS#. In total, 692 complete student data sets were collected in the study, with a male to female ratio of ~ 80:20 respectively. This ratio is in-line with CS courses in Ireland, the US and many other countries [77, 115]. As presented in Chapter 5, the main study collected 17 factors. Many of which were not included in the final two updated PreSS models 5.4.3. This gender study revisited all of these factors and presents the insights on gender differences where the findings were of interest. Thus the 17 factors are presented again in Table 7.1.

When reviewing the literature on gender in CS1, differences in early programming performance was highlighted [59, 105]. In addition, for the main study, a programming test (PT) was added to the PreSS# survey consisting of three questions, of increasing difficulty and was administered without prior notice, which was included after the factors were collected in the main study. Q1 was a single print statement, Q2 was a similar print statement that executed n-times (iteration) and Q3 was similar to Q2 but had a conditional statement for each iteration.

Maynooth University was the institution selected for this preliminary examination of the PT as it had 36% of the student cohort, with a male to female ratio of ~73:27 respectively. The PT recorded the time it took for a student to answer each question, the percentage of students who actually completed a question (by completion it is implied that a student submitted work, not that the work was correct), the percentage of students who ran out of time, and which question that they ran out of time on, and finally a dichotomous result, as a correct or incorrect answer.

Table 7.1: A recap of the 17 factors included in the main study

| Factor Name | Factor Details |
|---|---|
| Institution type | University, Institute of Technology, Community College {1, 2, 3} |
| Time to complete the survey | Seconds |
| Age | Years as an integer |
| Mature Student | Dichotomous |
| Gender | Dichotomous |
| 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 using Appendix C |
| 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 PCA |

## 7.3 ANALYSIS

### 7.3.1 APPROACH

A Welch's *t*-test was used to examine differences between male and female students using the main study factors and the additional PT. This is more suitable than a Student's t-test as the male and female data sets could report unequal variance, where, even if the variance is similar, a Welch's *t* test still performs strongly (unless a very small sample size which is not the case here, typically $n < 5$ [2]). The factors due to their large size (n = 17) have been broken into two instruments best reflecting their measurements, background factors (such as age, prior mathematics grade) and psychological factors (such as programming self-efficacy and test anxiety), this coupled with the PT, results in three instruments recorded and discussed in this chapter.

## 7.4 RESULTS

### 7.4.1 INSTRUMENT 1: BACKGROUND FACTORS

No significant difference was found in the age profile between male and female students, opposing some literature where female students were found to enter the field at a later stage [21].

Female students reported a significantly higher previous achievement in mathematics, in line with other research [20]. A significantly higher percentage of female students also reported that they took the highest level of mathematics in Ireland.

Male students spend significantly more time playing computer games than female students do, while when it comes to social media, the trend inverts where female students spend significantly longer on social media.

For time spent working in a part time job, this study found no significant difference between male and female students. That said this factor could warrant further analysis, given a standard deviation of 7.26 and 7.11 hours respectively. This could suggest that many CS students do not have a part time job, while many others work up to 13 hours a week.

No significant difference was found in the time it took to complete the survey section of this study. The recorded time included both the background survey and the psychological questionnaire, but not the programming test as this was timed separately.

The results to the background factors are presented in Table 7.2:

### *Background Factors Summary Results*

Table 7.2: Background results with Welch's t-test p values.

| **Data Point** | **Male** | **Female** | ***p* value** |
|---|---|---|---|
| Average age | 22.37 | 21.32 | 0.902 |
| Average Mathematics grade prior to CS1 (normalized) | 7.67 | 8.32 | **0.0030** |
| Students who took the higher level mathematics before entry to the course | 25.41% | 38.36% | **< 0.001** |
| Student overall pass rate | 70.75% | 76.71% | **<0.0001** |
| Average hours spent a day playing computer games | 1.68 | 0.66 | <0.0001 |
| Average hours spent a day on social media | 2.57 | 4.19 | <0.0001 |
| Average hours per week spent working at a part time job | 5.16 | 5.63 | 0.4806 |
| Percentage of students who dropped out during the year | 6.95% | 4.11% | <0.0001 |

## 7.4.2 INSTRUMENT 2: PSYCHOLOGICAL FACTORS

### *Programming Self-Efficacy*

As with the original PreSS model, programming self-efficacy was recorded and PCA applied. The recorded values ranged from -3.6 up to +4.6. A student with high programming self-efficacy was in the negative region, while the students with low programming self-efficacy was in the positive region, in both cases approaching the upper and lower ranges suggested the highest or lowest programming self-efficacy respectively. Not only did these results show that there is a significant difference between male and female self-reported programming self-efficacy (Table 7.3), but that male students rate their programming self-efficacy positively whereas female students rate their programming self-efficacy in a negative light, this is also echoed in a large body of literature: [7, 20, 21, 41, 55, 91, 105].

### *Concepts, Design And Completion Of A Program*

This measurement recorded how students felt about the level of difficulty that they associated with three programming areas: understanding programming concepts, designing the logic of a program without help and completing lab assignments. This measure used a normalized 5-point Likert scale, which ranked each area: concepts, design and completion ranging from very difficult (1), to very easy (5), and was summated to give an overall value. Male students reported a significantly larger value than females. This measure could be related to other measurements recorded in this study, such as programming self-efficacy.

### *Intrinsic Goal Orientation*

Intrinsic Goal Orientation is a component of self-regulated learning [92, 93] which measures the curiosity, desire for challenge or an ambition to master a task rather than just getting by. A student with a higher intrinsic goal orientation is likely to promote a higher level of persistence toward learning a subject, in both the short term and long term [134]. The questions used in this instrument were based on the motivated strategies for learning questionnaire (MSLQ) using a seven point Likert scale [92, 93], as investigated in the original PreSS study. Males reported a significantly higher average intrinsic goal orientation value than females, which opposes some literature [38], where female students reported a higher intrinsic goal orientation.

### *Test Anxiety*

Test anxiety is a self-reported measure of one or a combination of tension, fear, worry, nervousness or unease that may occur on or before test situations, based upon the MSLQ using a seven point Likert scale. Female students returned values significantly higher than that of males. This is also examined across the three institution types, which may have value in future research if one institution type has lower test anxiety than another, as presented in Section 7.4.4.

### *Predicted Overall Result Vs Actual Result*

Chapter 5 determined that the grade a student expects to receive in CS1 had value as a factor an updated PreSS model (Model ID 1). In this study male students predicted, that they will achieve significantly higher results compared to their female counterparts. Male students have been found to perceive programming to be easier than females do [91], which could be related to general lower programming self-efficacy of female students, in turn contributing to the lower expected result early on. Contrary to this factor, when the end of year pass rates were examined, female students had a statistically significant higher pass rate than their male counterparts, as presented in Table 7.3. This may align with other research where findings show at least an equal par on performance at the end of the course compared to a male dominated initial period [59, 77].

### *Psychological Summary Results*

Table 7.3: Psychological results with Welch's t-test p values.

| **Data Point** | **Male** | **Female** | ***p* value** |
|---|---|---|---|
| Programming Self-Efficacy | -0.1956 | +0.7328 | <0.0001 |
| Concepts, Design and Completion in Programming | 9.48 | 8.5 | <0.0001 |
| Intrinsic Goal Orientation | 20.83 | 20.00 | <0.0211 |
| Test Anxiety | 21.14 | 25.33 | <0.0001 |
| Average expected end of year grade | 76.89% | 71.95% | 0.0007 |

### 7.4.3 INSTRUMENT 3: PROGRAMMING TEST

#### *Programming Timing Analysis*

The average time it took male students to answer Q1 compared to female students was not significantly different, but it is noteworthy to examine the actual difference in average time. This was 18 seconds, or in other words, it took female students just over 10% longer on average to complete the question. The time difference was substantially less in Q2. Although a significant difference was found in Q3, this must be interpreted with caution as the number of students that attempted Q3 was comparatively small, thus bringing into question its value. The percentage of students who attempted (at least attempted, but not necessarily completed) a question revealed a trend. Other than Q1 (which all students attempted), a significantly higher proportion of male students attempted Q2 and Q3. The percentage of female students who either ran out of time on Q1 or overall was significantly higher than that of their male counterparts.

#### *Programming Performance*

The submitted code of each student for each submission was corrected. This was in dichotomous form, either correct or incorrect. The statistical analysis was completed using a binomial distribution and a Welch's t-test. Male students performed significantly better than female students did, even when individual questions are examined. There could be a strong link in this finding to that of time taken to answer the questions, which perhaps could correlate to programming self-efficacy at this very early stage of tuition. At this point in our study, it is noteworthy to consider that although female students may be performing at a lower standard than males in early programming tests, they perform as good if not better at the end of year examinations. In addition, students did not know in advance that there was a test, thus if students had known in advance, perhaps female students would have prepared more than male students, influencing the result [25]. Table 7.4 presents the results for the programming test.

*Programming Summary Results*

Table 7.4: Programming Test results with Welch's t-test p values.

| Data Point | Male | Female | *p* value |
|---|---|---|---|
| Average time to complete: Q1 | 02:44 | 03:02 | 0.1346 |
| Average time to complete: Q2 | 02:09 | 02:05 | 0.6682 |
| Average time to complete: Q3 | 00:56 | 00:38 | **0.0023** |
| Percentage who attempted: Q1 | 100.00% | 100.00% | NA |
| Percentage who attempted: Q2 | 77.78% | 76.47 | **<0.0285** |
| Percentage who attempted: Q3 | 30.56% | 16.18% | **<0.0001** |
| Percentage who ran out of time overall | 84.44% | 88.24% | **<0.0001** |
| Percentage who ran out of time on Q1 | 17.22% | 19.12% | **0.0006** |
| Percentage who answered Q1 correctly | 64.44% | 61.76% | **<0.0001** |
| Percentage who answered Q2 correctly | 27.86% | 19.23% | **<0.0001** |
| Percentage who answered Q3 correctly | 1.82% | 9.09% | **<0.0001** |

### 7.4.4 ADDITIONAL ANALYSIS

Using the same instruments and recorded data, differences in the genders, across the two countries and then over each of the three types of institutions from the main study were examined. This analysis examined if male and female students exhibit the same characteristics as found in the previous sections, when examined in independent countries or academic institutions.

*Country Specific Differences*

When each of the factors for both female and male students in Ireland and Denmark were compared, only one significant difference was found. This was the average hours spent per week working on a part time job, where Irish male and female students worked significantly longer hours. While this is interesting, it is important to note that in this comparison 95% of students were from Ireland with only 5% of the students from a single institution in Denmark. Further work is required to determine if this finding would hold true on a large sample with other countries or institutions in Denmark.

### *Institution Specific Differences*

The 692 students were compared based on the three institution types in this study, which included two universities, five institutes of technology and four community colleges. First gender specific measurements that were not significantly different across all three of the institution types are presented. These included: the hours spent on social media per day, concepts, design and completion of a program, test anxiety, and intrinsic goal orientation. These findings are of value as they can be used for the development of gender focused retention and recruitment strategies while being institution and academic level independent. Second, gender specific measurements that differed across the three institutions were examined. Differences were expected in some measurements, for example, prior mathematical ability, which may be due to institution entry requirements.

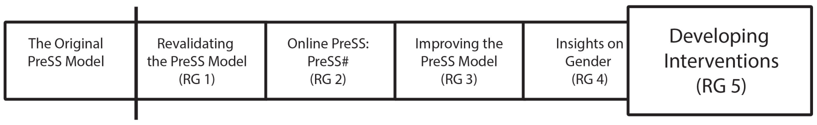
A finding, which was identical for both male and female students, was the average expected end of year grade recorded in the background factors. University students had the lowest average expected end of year grade, followed by the institutes of technology with community colleges reporting the highest average expected grade. A similar result was found when programming self-efficacy was examined across the three institution types, with university students reporting the lowest value (for both male and female students). For female students community colleges reported the highest average programming self-efficacy whereas for male students it was the institute of technology. This finding may be of value when developing institution specific strategies or methodologies.

## 7.5 SUMMARY

The study described in this chapter has found that female students have multiple positively correlated factors for female students learning programming, such as higher mathematical grade prior to CS1, higher end of year pass rates, and lower drop-out rates compared to their male counterparts. This holds true, even though female students underrate their programming self-efficacy compared to their end of year grade. This early, low programming self-efficacy is compounded by several measurements recorded: female students have a significantly lower end of year grade expectation, a significantly lower self-rating on programming concepts, program design and completion and a greater level of anxiety for tests or test situations all at the beginning of CS1 compared to their male counterparts. This low programming self-efficacy and higher test anxiety of female students compared to their male counterparts may be a disadvantage early in CS1 (where female students do not perform as well on average as their male counterparts in early programming tests) but could be advantageous in the later stages. These negative factors combined, may be the catalyst for additional programming practice or study towards the end of the module. Thus resulting in the observed higher performance than their male counterparts in the overall module. The reverse may also be true for males, as their higher self-efficacy and lower test anxiety may result in less study effort towards the end of the module.

Contributing to the development of interventions as discussed in the following two chapters and in addressing RG 4 (*Investigate insights on gender differences in introductory programming courses to further inform the PreSS model and interventions*), this chapter reported some noteworthy insights on gender differences in Cs1. Not only could this be used to help inform interventions, methodologies and pedagogical approaches in the classroom, but may ultimately help in computer science promotion and addressing the concerning low female uptake of computer science, with insights taken from this research [103], and presented in the media: "Girls just as good as Boys at computer science " [43].

# 8 INTERVENTIONS


The Original PreSS Model
Revalidating the PreSS Model (RG 1)
Online PreSS: PreSS# (RG 2)
Improving the PreSS Model (RG 3)
Insights on Gender (RG 4)
Developing Interventions (RG 5)

## INTRODUCTION

As PreSS can predict struggling students with very high levels of accuracy (71% and ~ 80% sensitivity), the next research goal focused on interventions to assist struggling students to improve outcome. The most prominent factor for predicting success, for both the original PreSS study and this body of work has been programming self-efficacy. In addition, the PreSS model could identify students early in CS1 at risk of failing or dropping out, thus improving student programming self-efficacy at this early stage (thus improving performance) is the primary goal of the proposed interventions. While examining the literature, it became very apparent that there was little to no interventions targeting programming self-efficacy in CS1. While some studies cite or mention self-efficacy during their interventions [47], there was no focus on, or quantitative data recorded for self-efficacy, thus making it difficult to evaluate if any positive changes as a product of the intervention (such as performance), could be solely attributed to the interventions affect on self-efficacy, and not other factors.

Developing interventions specifically targeting programming self-efficacy early in CS1, while quantitatively analysing the results, is novel and with little to no literature to establish a foundation, this chapter implements two pilot interventions. The two interventions were conducted over a two year period, with the first investigating the use of Scratch alongside CS1 and the second on the promotion of a growth mindset during the delivery of CS1. With no literature to establish if the interventions are applicable to all students or student sub-groups, the pilot interventions were applied to the entire student cohort. This allowed the interventions to be enacted early in CS1, and using the PreSS survey, factors such as programming self-efficacy was recorded and tracked. This quantitative data enables this work (and future work) to examine the affects of the intervention over time, and its affect on specific student sub-cohorts, for example, by gender or age. This would then inform the CSEd community, if the interventions are of value and if they are applicable to all student cohorts (or specific sub-cohorts). Thus setting the foundation for future works on interventions to promote programming self-efficacy in CS1. This chapter addresses RG 5, which investigates interventions to reduce attrition rates by focusing on increasing student self-efficacy and therefore student performance. The rationale for the selection of each interventions is further discussed within the following two sections, in addition to a description of each intervention and the main findings.

## 8.1 SCRATCH TO IMPROVE SELF-EFFICACY AND PERFORMANCE IN CS1

### 8.1.1 INTRODUCTION

This section describes a study conducted in the 2015-2016 academic year, with consideration to data from the previous three years. The study examined when students learned Scratch, in parallel to their CS1 module, would their programming self-efficacy improve, and as a result, their performance in the CS1 module.

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 [80, 119]) that they have not grasped in their mainstream text based language. This may be due to the fact that in a text based programming language such as Python, Java or C#, if a student could not master the syntax, they may not be able to transverse to the next topic, causing a student to fail or drop-out. This section describes the study, which is also discussed further in the proceedings for International Conference on Engaging Pedagogy (ICEP), Maynooth University, Ireland, 2016 [98].

### *Motivation*

The delivery of a Scratch module in parallel to the staple introductory CS1 programming module was proposed due to its simplicity. 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. Thus Scratch may allow novice programmers to reach threshold concepts before they are reached in the CS1 module or allow the students to access the threshold concept, even if they have not mastered the required CS1 text based code. A threshold concept is a concept that is likely to trouble a student as it is a previously inaccessible way of thinking about something [80].

The exposure to Scratch, in addition to CS1, may help students better comprehend threshold concepts, even if the student has not mastered the text based code. This in-turn could increase student's programming self-efficacy, a critical predictor in the PreSS model. Programming self-efficacy has a positive relationship between a student's programming self-efficacy and programming performance [28, 136, 140]. Recent research has reported that use of Scratch (in a middle school) aided the students in learning more advanced concepts (although the end of year results were not significantly different to those student's who were not exposed to Scratch) [5].

There is little literature on the use of Scratch in third level; the small amount that exists, is usually part of a pre-third level course, summer course or CS0 module and not used or examined in the initial academic year [5, 79]. One study attempted to introduce CS1 via Scratch [39]. The study while finding no improvements in results found an increase in student motivation. Stajkovic reports that self-efficacy makes an important contribution to work motivation [128], thus improvements in motivation due to the inclusion of Scratch, may be the result of increases in self-efficacy. Kereki only introduced Scratch for the initial three weeks of a fifteen week CS1 module, then reverted to Java. A longer period of Scratch use may have value. If Scratch can increase student self-efficacy, it could lead to increased success or results.

### 8.1.2 DATA COLLECTION

This study was conducted at a community college in the academic year 2015-2016 using PreSS# as the data collection tool. The community college delivers an Advanced Certificate in Software Development (ACSD) course (at Level 6 on the National Framework of Qualifications). Results from students who participated on the same course in the three previous years were used for comparison. The CS1 module is comparable to many introductory programming modules delivered at third level. This CS1 module had the same notes, course content and lecturer for the four years, (including the year the study was conducted 2015-16). Entry to the course is based solely on an interview, thus students were not selected competitively based on grades. Table 8.1 presents the student numbers and breakdown of gender and age from each year group. The 2015-2016 year group were the focus for the intervention.

Table 8.1: Cohort data for the four years of the Scratch study

| **Attribute** | **2012-13** | **2013-14** | **2014-15** | **2015-16** |
|---|---|---|---|---|
| Number of students who completed course | 24 | 31 | 28 | 30 |
| Male to Female ratio | 21:3 | 28:3 | 24:4 | 29:1 |
| Mature* to Non-Mature ratio | 4:20 | 13:18 | 1:19 | 2:28 |

** A mature student is any student 23 years of age or older, a non-mature student is under the age of 23*

### 8.1.3 METHODOLOGY

#### *Introductory Programming Module*

The CS1 module was compulsory for each student enrolled in the Advanced Certificate in Software Development course. The CS1 module was taught using C# and the .NET platform. In the initial three years of this study, no other programming language was taught alongside the CS1 module. The module was taught over 10 weeks and consisted of the following structure: four hours of lectures per week and four hours of laboratories a week. Further specifics of the course topics and grading structure can be found in the publication [98].

#### *Scratch - Game development Module*

Scratch was delivered as an introductory standalone games module in semester one alongside CS1 in the academic year 2015-16. The module used the Scratch 2.0 IDE. The aim of this module was to introduce students to computer games, before they undertook the more advanced second semester variant which consisted of Unity and C# scripting. This introductory module ran for the same 10 weeks as CS1 and consisted of one hour of lectures and a one hour and twenty-minute laboratory per week.

### 8.1.4 RESULTS

#### *Programming Self-Efficacy*

Programming self-efficacy was measured using the same method as in the original PreSS study and collected using PreSS#. This study collected programming self-efficacy at $\sim 10\%$ into the module delivery. The study also had access to the previous cohort's programming self-efficacy data, at the same point in the course delivery for comparison, collected from 2013 to the 2015. As per previous studies detailed in this thesis, the same factors and data reduction techniques were applied.

The results presented in Table 8.2, found no significant difference in the student programming self-efficacy between the two control groups and the intervention group. Although no difference was found, it should be noted that the programming self-efficacy questionnaire was administered (as with all studies in this thesis) at approximately 10% of the course delivery. This perhaps may have been too early to see significant differences in programming self-efficacy.

Table 8.2: ANOVA analysis of student's self-efficacy from all three year groups

| **ANOVA Analysis: 2013-2014, 2014-2015, 2015-2016** | |
|---|---|
| Total Students | 88 |
| $f$-ratio | 2.3822 |
| $p$-value | 0.0985 |

### *Programming Performance*

The results from each year group were compared using an ANOVA analysis. Student results consisted of the overall module marks in CS1. The ANOVA analysis investigated if CS1 with and without the Scratch intervention, differed significantly. Table 8.3 presents the performance results for each year group and Table 8.4 presents the ANOVA analysis. Table 8.4 reports no significant differences between any of the four year groups (control and treatment) with a p value = 0.7.

Table 8.3: Student programming performance data

| **Attribute** | **2012-13** | **2013-14** | **2014-15** | **2015-16** |
|---|---|---|---|---|
| Average grade in CS1 | 78.56% | 72.18% | 69.53% | 67.20% |
| Std. Dev. (Grades in CS1) $\sigma$ | 16.81% | 16.73% | 18.00% | 12.91% |

Table 8.4: ANOVA analysis of student's performance from all four year groups

| **ANOVA Analysis - All four year groups** | |
|---|---|
| Total Students | 113 |
| Grade (Mean) | 69.86% |
| Grade (Std. Dev σ) | 16.35% |
| *f*-ratio | 0.4851 |
| *p*-value | 0.6932 |

### 8.1.5 ADDITIONAL FINDINGS

Upon further investigation, a substantial variance in the average module pass rates between the first three year groups (control groups) and the final year intervention group was found. The average module pass rate is calculated across all ten modules delivered in the ACSD course (ten in 2015-2016 which included the additional games 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 8.5.

Table 8.5: Average overall module pass rates from each year group

| | **2012-13** | **2013-14** | **2014-15** | **2015-16** |
|---|---|---|---|---|
| Average pass rate over all modules | 81.51% | 77.83% | 70.45% | 46.37% |
| Std. Dev σ | 13.71% | 19.65% | 17.44% | 18.74% |

Initially an ANOVA analysis was completed on the three control years (2012-2014) concluding that no statistical difference was found between the average pass rates of each delivered module, with a p value of 0.4. Next an ANOVA analysis was completed on all four year groups (control and intervention), resulting in a significant difference found in the average module pass rates with a p value of 0.0004.

A Tukey HSD post-hoc analysis confirmed that it was the intervention cohort results that were significantly statistically lower to the control groups. This suggests that the intervention cohort was significantly weaker overall, than the previous three control cohorts. Given this, the CS1 performance results were statistically similar to that of the previous three years, thus it could be inferred that while the intervention group was weaker, the cohort performed as well as the previous three years in the CS1 module. This may be attributed to the additional Scratch gaming module. Further investigation is required.

## 8.2 PROMOTING A GROWTH MINDSET IN CS1

### 8.2.1 INTRODUCTION

This section describes an intervention conducted in the academic year of 2016-2017. The intervention promoted a growth mindset during a CS1 module in an effort to increase performance. This study was motivated by the work of Dweck [46] on mindset which has received wide spread adoption across primary and more recently, second level education.

### 8.2.2 LITERATURE

Dweck has provided compelling evidence that mindset towards ability can impact upon student performance [46]. She describes two mindset types: a growth mindset and a fixed mindset. Fixed mindset supports the idea that ability is innate, thus your result reflects your innate ability and can not be improved upon. Growth mindset on the other hand supports the idea that ability can be improved, both through effort and learning from failure and constructive feedback.

Although there is very little literature on using mindset interventions to increase performance on introductory programming courses, the studies available have reported promising or mixed results [36, 51, 123]. Cutts reports that although mindset did not change significantly, there was a positive change in performance. This perhaps could be in part due to Cutts reporting (both from their experiences and literature) that students in introductory programming, tend to change towards a fixed mindset during the course, as "that learning to program may foster a fixed mindset due to the very high number of potential error points" [36]. Simon conducted a study spanning three institutions and reported no significant findings while employing their "saying is believing" mindset intervention [123]. However it must be noted that the intervention exposure was very minimal consisting of a single lecture and a one page handout reminder. Flanigan did not employ an intervention, but measured mindset across the semester and reported "for all students, there were significant increases in fixed mindset and significant decreases in growth mindset across the semester. However, results showed that students had higher scores for growth mindset than fixed mindset at both the beginning and end of the semester" [51]. Flanigan also reported that student mindset was a weak predictor of performance.

Overall the literature reports mixed outcomes at third level. This perhaps is due to the limited exposure that students have received. The largest was four consecutive weeks, which reported some success [36]. This study in an effort to address RG 5 (develop and investigate interventions that could reduce attrition rates in introductory programming modules) has implemented an intervention based on the work of Dweck.

### 8.2.3 DATA COLLECTION

During the academic year 2016-17, two institutions participated in this study (with a total $n = 42$ participants), where the data was collected using PreSS#. Both institutions also participated in the main study (Chapter 3.2). This allowed for the comparison of the previous student population (the control group with no intervention) to the student cohort from this study (treatment groupy) to examine the effectiveness of the intervention.

The two institutions consisted of a community college and an institute of technology. The data collection process and analysis techniques were identical to the main study outlined in this thesis. There was also the addition of a mindset survey adopted from the work of Dweck [46], by D'Anca [37] as presented in Appendix F. The only difference in this work to that of the main study in this thesis (other than the mindset survey), was that the same data was collected 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 allowed for the tracking of changes in attributes such as mindset and programming self-efficacy over the entire academic year.

The survey was optional, that coupled with absenteeism meant that not all students, participated in all stages. When analysing one or more stages, only students who were present across all stages examined are investigated. Where this was the case, the sample size is reported.

### 8.2.4 METHODOLOGY

Stage one to stage two, was where the intervention was applied. This consisted of several approaches to promote a growth mindset. The methodology was developed from previous studies [36, 46, 78, 84]. The approaches fall under three headings:

**Lecture:** At the start of each lecture/lab, the lecturer described Dweck's work through a discussion of fixed mindset vs. growth mindset. This approach promoted the fundamentals of growth mindsets and presented case studies from Dweck's work. The methodology also drew upon personal experiences, and relayed the correlation between work ethic (grit) and attainment of ability. In addition, testimonials from students who had completed the course, especially students who initially struggled were presented to the cohort. This was conducted for all 12 weeks from stage one to stage two and typically lasted for the first ten minutes of the lecture.

**Research:** In addition to the lectures, at approximately each quarter of the course delivery, case studies were presented to the students with scientific findings (in contrast to qualitative). This was drawn from Dweck's work, the related literature and neuroscience. This aimed to further target students with a fixed mindset, who believe that they cannot improve and who may disbelieve the findings presented in the lectures. An example of this was research on neuroplasticity [114].

**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 positively, critiquing the process and output, but not the student or ability. This was compounded by promotion, that if feedback was based on a poor result, that the feedback itself is a tool to learn how to improve and if worked upon will lead an increase in performance. This aimed to modify the perception that failure is a negative end point, to a positive starting point with direction on how to succeed.

### 8.2.5 STUDENT COHORT ANALYSIS

First, before any investigations into self-efficacy or performance gains could be conducted, an analysis of differences between the two student cohorts (the control and treatment group) was completed using the same data collected at the same time via PreSS#. This was conducted at stage one, before the intervention was applied to investigate if any differences existed that may account for variance (if any) in the affect of the intervention on performance. It must be noted the both institutions were delivering the exact same course both years, with no additional interventions or teaching methodologies applied. Also the same lecturer, lectured in both institutions to both groups of students (both years), thus ruling out different lecturer delivery methods or the lecturer themselves as a contributor to any difference in performance. A Welch's *t* - test was used to examine if any statistically significant differences existed between both data sets.

The results presented in Table 8.6, show that other than gender balance (*2015-16 = 51% female students compared to this study which only had 13% female representation*), no statistically significant differences existed between the two cohorts. Thus any increases in performances can reasonably be attributed to the intervention, and not pre underlying population differences.

Table 8.6: Comparison of two student cohorts, at stage one,pre intervention.

| **Attribute** | ***p-value*** | Significant |
|---|---|---|
| Programming Self-Efficacy | 0.0521 | N |
| Predicted End of Year Grade | 0.6171 | N |
| Gender Balance | <0.00001 | Y |
| Age | 0.1541 | N |
| Maths Grade Normalized | 0.2655 | N |
| Maths Grade Raw | 0.6624 | N |
| Hours Spent Playing Computer Games | 0.5170 | N |
| Time to Complete Survey | 0.8135 | N |
| Condesco | 0.4622 | N |
| Intrinsic Goal Orientation | 0.3199 | N |
| Test Anxiety | 0.5054 | N |
| Hours Spent on Social Media | 0.3490 | N |

### 8.2.6 RESULTS

The results as presented in Table 8.7 were statistically significant, with the 2015-16 cohort having a pass rate of 41.07%, and this study including the intervention, reporting a pass rate of 66.67%. This was a significant increase in performance given the only underlying population difference was the reduction in ratio of female students in the intervention cohort. It is also significant as female students typically outperform male students.

Table 8.7: Comparison of two student cohorts, at stage two, the end of CS1.

| **Results** | ***p-value*** | Significant |
|---|---|---|
| Raw Results | <0.00001 | Y |
| Dicotomous Results (As per PreSS) | 0.0034 | Y |

### 8.2.7 ADDITIONAL ANALYSIS

The average mindset value in stage one was 41.69 (where higher values correlate to a Growth mindset), with the average mindset value of stage two was 43.03 ($n$ = *24*), thus there was no statistically significant difference between the two stages with, $p$ = *0.0613*. This was interesting as student performance increased with this intervention. Next, the change in programming self-efficacy was examined, from stage one to stage two. The data again reports no statistically significant overall difference in programming self-efficacy from stage one to stage two, with a $p = 0.8$. Both of these findings were unexpected, as with the performance increase, it was envisioned that perhaps either programming self-efficacy or mindset may have been positively affected by the intervention, from stage one to stage two.

To visually examine mindset and programming self-efficacy, the values were graphed to investigate if any patterns or clusters existed that might explain why no significant changes were observed from stage one to stage two. A very interesting pattern emerged, which is presented in Figure 8.1.

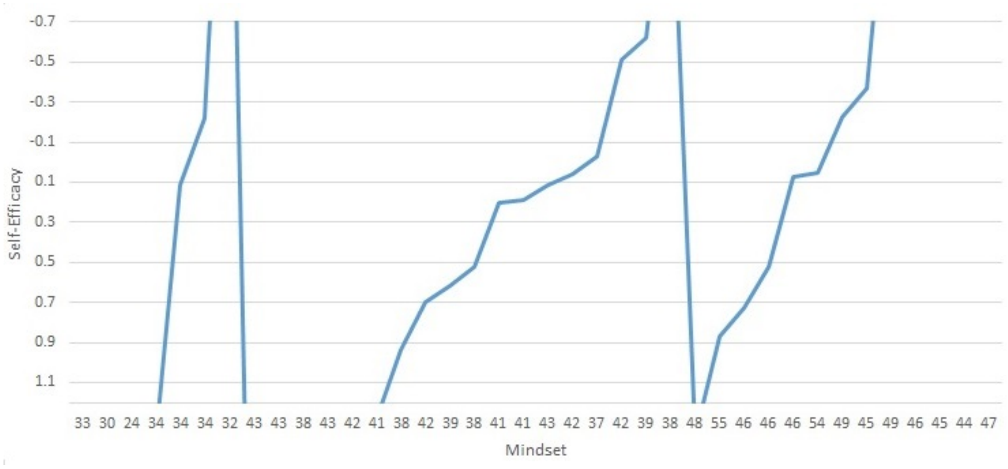


Figure 8.1: Programming self-efficacy and mindset values plotted at stage one.

Figure 8.1 presents a multi-modal distribution with three visual groups. To investigate the change in self-efficacy and mindset from stage one to stage two, the data was grouped in relation to performance ($n = 20$). The three groups consisted of the poor performing students, middle performing students and top performing students. The rationale was to investigate if the intervention was more effective (or not) for a specific cohort of performers in CS1, as Figure 8.1. Figure 8.2 presents the findings.

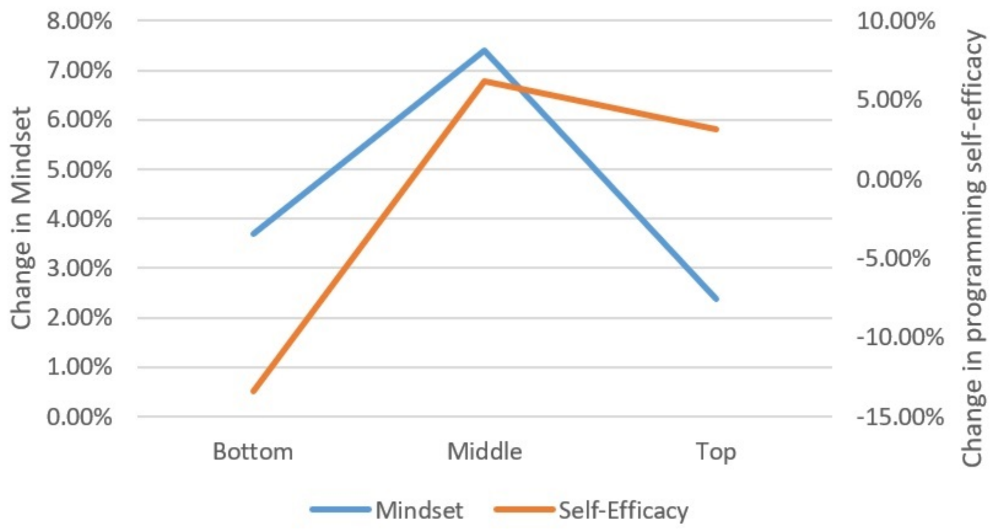


Figure 8.2: Changes in programming self-efficacy and mindset over CS1, grouped by performance

Figure 8.2 shows that changes in programming self-efficacy [1] and mindset from stage one to stage two are somewhat visually correlated ($r = 0.41$). Middle performing students reported the largest positive change in mindset (towards growth), whereas their programming self-efficacy decreased. The weakest students reported the largest positive change in programming self-efficacy while experiencing a positive change in mindset. Top performing students reported the least amount of change in programming self-efficacy and mindset.

### 8.2.8 SUMMARY

Promoting a growth mindset as an intervention in CS1, resulted in a statistically significant increase in performance. Although there is very little literature in this space, studies have reported similar results [36]. Average mindset and programming self-efficacy did not significantly change during the intervention. However this chapter also examined the changes in programming self-efficacy and mindset throughout CS1 for students grouped by performance (stage one, two), showing that programming self-efficacy reduced for high performing students (with the smallest change in mindset) whereas for the lowest performing students self-efficacy improved during the intervention. Students in the middle performing group, had similar self-efficacy and mindset to those students who were top performers, but showed the largest change in mindset.

Perhaps these opposing changes for strong and weak students may be the reason that the overall changes in mindset and programming self-efficacy were not significant (as was reported in the literature). This finding suggests that although the intervention saw a significant increase in performance, that changes in programming self-efficacy and a student's mindset vary depending on the student and their performance. This finding requires a deeper analysis on a significantly larger cohort of students, as this study has a number of limitations due to data set size.

1 Note: due to the data reduction algorithm applied to the programming self-efficacy data, a negative value correlates to a positive change in programming self-efficacy whereas a positive value correlates to a negative change in programming self-efficacy.

# 9 CONCLUSION AND FUTURE DIRECTION

## 9.1 conclusion

This thesis detailed the development of PreSS#, a fully automated web-based system, that can identify (for 2 out of 3 students, and 8 out of 10 struggling students), students at risk of dropping out or failing, early in CS1. In addition, early interventions were developed, that may have real impact when used with the findings of PreSS#. This longitudinal research and development, (extending the work of the original PreSS study and model), documents a multi-institutional, international, multivariate study. This thesis developed the PreSS model further, on a significantly larger disparate cohort of students. This was achieved by multiple revalidation studies, the development of a web-based real time system, factor and machine learning algorithm development and two developed interventions. This thesis presents an overall system that educators could use, to attempt to address attrition rates in CS1. This concluding chapter summarises the contributions made by this thesis and possible future directions of this work, some of which have already begun.

## 9.2 thesis contributions

### 9.2.1 longitudinal revalidation of press

This thesis documents two studies that examined PreSS several years after its original development, to determine if PreSS is still a valid prediction model, a decade after it was developed. The justification study contributed further by: providing additional data/factors that were used as part of a main study.

This then led to the main study in this thesis, which involved 11 institutions and just under 700 students. The institutions and the student cohort differed significantly, with two Universities, five Institutes of Technology and four Community Colleges participating. Not only did the programming languages differ across institutions, so did the assessment criteria and weighting. With that in mind, PreSS was still able to successfully predict 67% of student's end of year result, with sensitivity of 78%, representing students who were likely to fail or drop out. This body of work contributed to and addressed RG 1, to investigate if PreSS, is still a valid prediction model, a decade after it was initially developed. In addition this contribution answered a call from the CSEd community, where revalidation studies were seldom conducted [62].

### 9.2.2 A WEB-BASED REAL TIME IMPLEMENTATION OF PRESS

As the original PreSS model used a paper-based data collection method, it was time consuming to collect the data. If PreSS was to be used across multiple institutions with large student numbers, administration would grow and this could deter institutions from participating. A web based system named PreSS# which has PreSS at its computational core was developed, thus addressing RG 2 (*develop a web-based real time implementation of the original PreSS model*).

PreSS# can be used to allow educators to make informed decisions about methodologies, differentiation, and interventions at an earlier stage. The web-based system was also compared for accuracy to the original PreSS model and no statistically significant difference in prediction performance was found. Additionally this thesis's large scale study used PreSS# as its data collection tool.

### 9.2.3 DEVELOPMENT OF THE PRESS MODEL

Two investigations were conducted, which contributed to the improvement of the PreSS model. First, new factors were identified and an updated PreSS model was developed. Second, several machine learning algorithms were investigated to examine if algorithms not used in the original PreSS model may add value to the updated PreSS model. In doing so this addressed RG 3 (*Investigate if the original PreSS model can be improved upon using: new factors for the model and Alternative machine learning algorithms for the model*)

**New factors:** This thesis examined combinations of factors not used in the original PreSS model. Seventeen additional factors were identified (after data reduction algorithms were applied). Multiple models were developed using subsets and two updated models were produced, with performance gains compared to the original PreSS model.

**Machine Learning Algorithms:** A multitude of machine learning algorithms and its subset, artificial neural networks, were examined next. The goal was to determine if further improvements to the PreSS model could be achieved. The machine learning algorithms included: naïve Bayes, logistic regression, support vector machines, decision tree, k-nearest neighbour, single layer artificial neural network, deep learning artificial neural network and a convolutional artificial neural network. This work reported an increase in performance when using artificial neural networks, and in particular deep learning over the original PreSS algorithm, naïve Bayes.

### 9.2.4 ANALYSIS OF GENDER DIFFERENCES IN CS1

Based on the findings from the original PreSS PhD thesis, gender differences in CS1 were investigated. Although gender specific models performed well they did not generalize when gender was added as a dichotomous factor. Thus a deeper investigation would contribute significantly to the further development of PreSS and in parallel help inform intervention development. This used a combination of instruments from the main study and institution data. This work highlighted multiple areas where male and female CS1 students differ significantly, such as early programming self-efficacy and early performance in contrast to overall CS1 performance. These insights and their implications are important, as some did not conform to literature findings, when developing future interventions and methodologies to help reverse the dwindling female enrolment numbers in western CSEd.

### 9.2.5 INTERVENTIONS TO IMPROVE PERFORMANCE IN CS1

The final contribution that this thesis makes, is the development of two interventions (RG5, *Develop and investigate interventions that could reduce attrition rates in introductory programming modules*). The primary target of the interventions were programming self-efficacy as this was the prominent predictor of performance through-out this thesis and the original PreSS PhD thesis. Two interventions were developed in this thesis: Scratch in parallel to CS1 and promoting a growth mindset in CS1.

**Scratch in Parallel to CS1:** This work investigated when students learned Scratch, a block type programming language, at the same time as CS1, would their performance in the CS1 module increase? Scratch was chosen as arguably, it may help struggling novice programmers to comprehend coding concepts that they have not grasped in their mainstream text-based language. This contribution at first, reported no improvement in performance, however using a deeper analysis of the student cohort, especially in other subjects, this work found that the intervention cohort was overall, significantly weaker than previous cohorts. Given this, the intervention group, their CS1 performance results were statistically similar to that of the previous years, where this may be attributed to Scratch improving the intervention group's programming performance.

**Promoting a Growth Mindset in CS1:** The second intervention developed was based on the work of Dweck, to promote a growth mindset in an effort to increase performance in CS1. This work also examined data from a previous year (as a control group) to compare results. The factors recorded were measured at multiple intervals throughout the course, to monitor changes as the intervention was delivered. This work found a significant increase in performance over the previous control group where no intervention was deployed (with the analysis finding no underlying population differences prior to the intervention). This contribution could help educators with future methodology and intervention development.

## 9.3 FUTURE DIRECTION

The work as outlined in this thesis provides numerous possible directions for future work. This section outlines some potential fruitful areas.

### 9.3.1 ROLL OUT OF PRESS#

PreSS# was successfully rolled out to just under 700 students, in 11 institutions. Going forward it is important to promote usage of the system as broadly as possible. During conference presentations from publications associated with this thesis, significant interest has been expressed in using this system. A follow-up study for improving retention rates and early interventions used by participating institutions would be a very valuable piece of work.

### 9.3.2 IMPROVING PRESS FURTHER

With new factors identified, and an investigation of machine learning techniques, PreSS is able to predict correctly for 2 out of every 3 students. However there are still 33% of the students incorrectly identified. Future work should attempt to investigate this further. A further investigation into varying ANN network topologies would be useful as this thesis reported improvements to the model using deep learning and convolutional neural networks. For ANNs, a large sample size is desired, thus with PreSS# as a ready to use tool, and the level of interest shown in it, a significantly larger study, suitable for ANN's could be considered, to further investigate these algorithms.

### 9.3.3 LEARNING ANALYTICS

A topic gaining traction in recent times, is learning analytics [120]. The literature suggests that models developed in this field can have value in predicting programming success [61, 65, 121]. During the main study, this body of work (as part of the programming test), collected learning analytics data. This data consisted of key strokes, mouse movements, compile attempts, compiler errors, and the final submitted code. Each data point also had timestamps associated with them. Future work would combine the educational data mining data from this study with the learning analytics data collected from the programming test, to investigate new/updated PreSS models to predict success. It is acknowledged that there are several studies in CS1 which use learning analytics data, such as the data collected in the programming test, but from reviewing the literature, to the authors knowledge, there has been no study that has combined the two fields of data. This novel work could perhaps add to the predictive power of PreSS.

### 9.3.4 GENDER RESEARCH

Gender in CSEd is a topic of increasing research, with many interventions and initiatives being deployed nationally and internationally. A popular theory in the literature is imposter syndrome [32]. It would be an interesting investigation to measure, imposter syndrome, mindset [46] and self-efficacy, and examine if there is any multicollinearity between the phenomena. In addition, it would be of value to measure self-efficacy across multiple educational domains, to identify if self-efficacy differences between male and female students report similar values and if not could there be an underlying cause (as there is no gender self-efficacy differences in science).

### 9.3.5 PROMOTING A GROWTH MINDSET IN CS1

This study found a statistically significant increase in performance, promoting a growth mindset to students in CS1. While this finding is of value to the CSEd community, the study also presented the findings (cautiously, as it was on a small data set) that the intervention effected sub groups of students based on performance in varying measures. Future work should involve clustering of the data, to investigate these subgroups further. Also, revalidating the performance increase of the intervention on a large data set would be desirable.

### 9.3.6 migrating press# to second level

PreSS and PreSS# have programming self-efficacy at their core. As coding and computer science subjects in Primary and Second level are currently in their phased (pilot) roll out, this study could be revisited using primary and/or second level students. This could be very valuable in the coming years once the subjects are in mainstream education.

#### *Primary and Second Level - Perceptions and Interventions*

Student uptake of computer science in third level has declined in previous years (where it fell by ~ 8% alone last year [44]), this is compounded by an alarmingly low female uptake [103]. It is unclear in the research and in particular in Ireland, where the cause of these phenomena are rooted. Thus using the factors and insights found in this thesis future work could include two bodies of work. First, a study examining possible root causes and where in the educational pipeline a student is deterred from pursuing an education in computer science (in particular for female students). Second, an intervention aimed at addressing the possible root causes for the declining engagement in computer science. These would be very timely as the introduction of the Leaving Certificate computer science subject is rolled out as phase one (initial pilot year). On a positive note, this work is already begun as outlined in recent research [104].

#### *Primary and Second Level - Teachers*

An important factor for developing student perceptions, may be the educators themselves. A possible research question could examine whether teachers having low computing self-efficacy has a knock-on effect on the students to whom they teach. Or could a teacher who has had a negative experience of coding be deterred from teaching it themselves? Cohorts of computing teachers are not attending professional development training sessions with titles that suggest a specific level of difficulty [104]. This perhaps relates to their self-efficacy (which will be investigated in future work). If teachers are exhibiting low self-efficacy for a topic such as coding, can this have an effect on students and student uptake?

## 9.4 FINAL THOUGHTS

On reflection of this body of work, several notable facets stand out, personally and from feedback from colleagues and at conferences. Perhaps one of the most notable contributions was the revalidation study. This was to the authors knowledge the first of its kind, in predicting programming success. The work's value, was compounded by the ITiCSE working group report [62] which highlighted the lack of revalidation studies in CSEd. The revalidation study, which spanned several studies from 2004 to present, was also of personal value. The author was a student during the original PreSS study, and participated in the one of the original PreSS study's pilot surveys. Thus while the author was not involved academically in the original PreSS study, they have been (at some level) part of the development of PreSS for over a decade.

Another facet that stood out during this body of work was the insights on gender differences in CS1. This investigation was able to (using quantitative data), examine multiple factors over CS1, which was novel work (as many studies typically examined one or two factors). The insights that were presented, proved very positive for female students in CS1 (in some cases challenging stereotypes, such as CS1 performance), and were noticed by national newspapers [43]. It is work and outcomes like this, that on a personal level, makes research such a valuable pursuit.

One of the final pieces of work, the development of the two interventions, was also a notable body of work. It is also a piece of work that perhaps, if more time was available, would have benefited with deeper investigations. Both interventions were developed to target increasing programming self-efficacy to improve student outcomes. As these interventions were novel, no prior foundations were available to build on. For example with promoting a growth mindset, the findings suggest more complex relationships exist between mindset and student sub-cohorts (such as age), that the current limited literature presents. Future work has begun to investigate the data gathered further. The work on the two interventions is very valuable to the CSEd community as the ITiCSE working group report [62] also mentioned: "The fourth Grand Challenge is to adopt results and practices into classroom use to continuously monitor and improve offered education".

The work outlined in this study, satisfied several of the ITiCSE working groups grand challenges [62]. This is very positive for the research outlined in this thesis, as much of the work was identified and started prior to the working group report. The working group consisted of 16 of the most influential CSEd researchers (and role models to the author), thus substantiating the value of this work. In addition, several institutions (US, Germany, England and China), have contacted the team to trial run PreSS# in their CS1 courses. The fact that there is interest in the work presented in this thesis, is very positive as the institutions perhaps see value in the research. This could lead to further work and collaboration on the PreSS model.

A

# APPENDIX A

## A.1 DATA PROCESSING TECHNIQUES

The pre-processed programming self-esteem data consisted of ten questions and was based on the Rosenberg self-esteem questionnaire. The original questionnaire was modified to reflect a student's perception of their programming ability [107].

### A.1.1 INTERNAL CONSISTENCY

Cronbach alpha values can be used to estimate the internal consistency of psychometric questionnaires. The values were calculated here and compared it to the original Rosenberg questionnaire to ensure internal consistency had not been jeopardized through the modifications. The Cronbach alpha values for the original Rosenberg self-esteem questionnaire were in the range of 0.82 to 0.88 and for the modified questionnaire, the alpha value was 0.91 [16].

### A.1.2 PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis (PCA) plots all data points in a multi-dimensional space. PCA was the data reduction algorithm used to reduce the programming self-efficacy questionnaire (consisting of ten questions) to one value to be used in the prediction model.

PCA essentially performs an orthogonal transformation (rotation of data in multi-dimensions) to find the covariance eigenvectors with the largest eigenvalues which represented the largest distribution or effect of the data set, hence selecting the principal component. This was implemented in PreSS, selecting eigenvalues $> 1$, to reduce the 10 questions of the programming self-efficacy attribute to one principal component [16][125]. This can be repeated for subsequent components to find the second principal component etc. The original PreSS model used the first principal component. A graphical illustration of this rotation of data in multi dimensions (one dimension per attribute used) can be seen in Figure A.1, where it can be seen that the data is rotated until the largest eigenvalue is found, which is illustrated by the additional arrows on the figure.

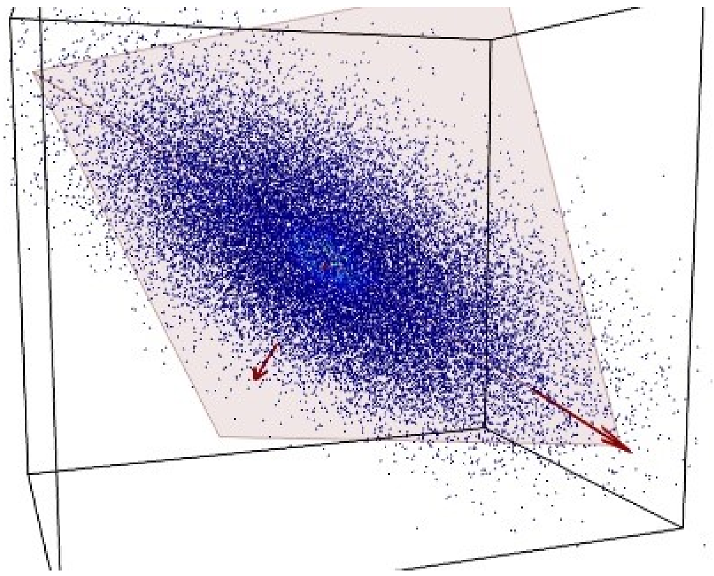


Figure A.1: Principal Component Analysis showing an example of covariance eigenvectors with the largest eigenvalues illustrated by the additional arrows.

## A.2 CROSS VALIDATION

Ten-fold cross validation (10FCV) was implemented, to determine the performance of models. This is a best practice method to avoid over fitting [148] with machine learning algorithms. This method is far superior to a hold-out method (this is where the data set is split into two groups with one group used for training and the other group used for testing) as it uses every sample in a data set for both testing and training. This reduces the chances of over fitting by removing any bias that the holdout method may have introduced. 10FCV starts by first randomising the data set and then the data is split into 10 folds of approximate equal size, this is the most commonly used fold size [148]. If the data set size does not allow for equally divided folds, the sets are created as per Table A.1 on 102 data samples. The data did not divide equally into ten folds so the remaining data is evenly dispersed into the folds starting at fold one, then fold two as needed [148].

Table A.1: Example of 10 folds, how the data is split before training / testing begins with the PreSS study containing 102 samples.

| *Data Set Size = 102, Folds = 10* | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 11 | 11 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| A | B | C | D | E | F | G | H | I | J |
| (*Fold)* | | | | | | | | | |

Once the data is split into the 10 folds, 10FCV holds one fold out for testing and trains on the remaining folds. For example, using the data in Table A.1, fold A is used for testing and the naïve Bayes model is trained on folds B – J. This process is then repeated using the next fold (fold B) for testing and trains on the remaining folds. This is repeated for all ten folds. This allows every fold to be tested on an independent training set, without the same entry being used for both training and testing. The results are then averaged and used as the final predictions as if it was only one test.

## A.3 PERFORMANCE MEASURES

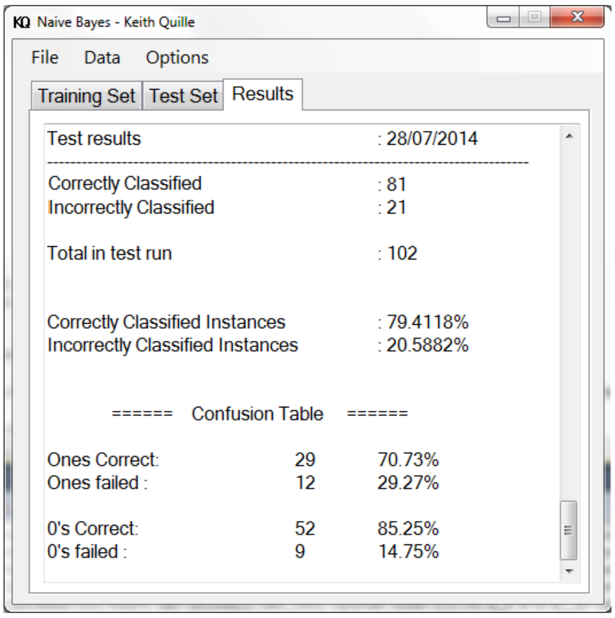
Three measurement techniques were employed in this study, specifically, overall classifier accuracy, sensitivity and specificity. The simplest form of evaluation is classification accuracy: the proportion of instances correctly predicted. Sensitivity is a measure of the proportion of actual positive instances (in the case of all studies in this thesis this infers students whom are at risk of failing or dropping out ) that are correctly classified and Specificity is the proportion of actual negative instances correctly classified (in the case of all studies in this thesis this infers students whom are likely to be successful in the introductory programming module).

The equations for accuracy, sensitivity and specificity are presented in equations A.1, A.2 and A.3.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{A.1}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{A.2}$$

$$Specificity = \frac{TN}{TN + FP} \tag{A.3}$$



```
Test results                                  : 28/07/2014
----------------------------------------------------------------
Correctly Classified                          : 81
Incorrectly Classified                        : 21

Total in test run                             : 102


Correctly Classified Instances                : 79.4118%
Incorrectly Classified Instances              : 20.5882%


        ======   Confusion Table    ======

Ones Correct:                        29       70.73%
Ones failed :                        12       29.27%

0's Correct:                         52       85.25%
0's failed :                         9        14.75%
```

Figure A.2: A screen shot of an application developed running on the PreSS study, showing the confusion matrix.

A confusion matrix is a table representing the breakdown of the predictions into four categories (As presented in Figure A.2)that include, the number of strong students that the system predicted to be strong, the number of strong students that the system predicted to be weak, the number of weak students that the system predicted to be weak and finally the number of weak students that the system predicted to be strong. Thus providing true positive, true negative, false positive and false negative values. This output is then used to calculate the accuracy, sensitivity and specificity.

B

# APPENDIX B

The selection process to label programming performance as weak or strong, was developed using the following criteria: (i) each institution's 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.

Table B.1: Weak and Strong Performance Measures

| **Institution** | **Boundary Value** |
|---|---|
| University | *Strong* > 59% >= *Weak* |
| Institute of Technology | *Strong* > 69% >= *Weak* |
| Community College | *Strong* > 79% >= *Weak* |

# APPENDIX C

## C.1 MATHEMATICAL NORMALIZATION TABLE

As identified in the future work of the original PreSS study, students entering third level had completed varying levels of mathematical education. In Ireland, the highest level of mathematical attainment prior to entry to third level, could have ranged from Junior Certificate, to Post Leaving Certificate courses and even other third level courses. As the original PreSS model assumed only the Leaving Certificate mathematics grade, a method was developed to normalize these differing levels of prior mathematical achievement into a single range of values. This section presents this normalization table, in Table C.2 and Table C.3, with the table keys presented in Table C.1.

Table C.1: Mathematical Normalization Keys

| **Key** | **Value** |
|---|---|
| LCHL | Leaving Certificate Higher Level |
| LCOL | Leaving Certificate Ordinary Level |
| LCFL | Leaving Certificate Foundation Level |
| LCAE | Leaving Certificate Applied |
| JCHL | Junior Certificate Higher Level |
| JCOL | Junior Certificate Ordinary Level |
| CUEX | College / University Mathematics Exam |
| FETA | Further Education Mathematics Exam |

### C.1.1 NORMALIZATION

Table C.2: Mathematics Grade Normalization Table - Part 1

| Exam | Min Grade | Max Grade | Normalization Value |
|---|---|---|---|
| LCHL | 85 | 100 | 12 |
| LCHL | 70 | 84 | 11 |
| LCHL | 55 | 69 | 10 |
| LCHL | 40 | 54 | 9 |
| | | | |
| LCOL | 85 | 100 | 8 |
| LCOL | 70 | 84 | 7 |
| LCOL | 55 | 69 | 6 |
| LCOL | 40 | 54 | 5 |
| | | | |
| LCFL | 85 | 100 | 4 |
| LCFL | 70 | 84 | 3 |
| LCFL | 55 | 69 | 2 |
| LCFL | 40 | 54 | 1 |
| | | | |
| LCAE | 85 | 100 | 8 |
| LCAE | 70 | 84 | 7 |
| LCAE | 55 | 69 | 6 |
| LCAE | 40 | 54 | 5 |

Table C.3: Mathematics Grade Normalization Table - Part 2

| Exam | Min Grade | Max Grade | Normalization Value |
|---|---|---|---|
| JCHL | 85 | 100 | 8 |
| JCHL | 70 | 84 | 7 |
| JCHL | 55 | 69 | 6 |
| JCHL | 40 | 54 | 5 |
| | | | |
| JCOL | 85 | 100 | 4 |
| JCOL | 70 | 84 | 3 |
| JCOL | 55 | 69 | 2 |
| JCOL | 40 | 54 | 1 |
| | | | |
| CUEX | 85 | 100 | 12 |
| CUEX | 70 | 84 | 11 |
| CUEX | 55 | 69 | 10 |
| CUEX | 40 | 54 | 9 |
| | | | |
| FETA | 85 | 100 | 8 |
| FETA | 70 | 84 | 7 |
| FETA | 55 | 69 | 6 |
| FETA | 40 | 54 | 5 |

D

# APPENDIX D

## D.1 PRE JUSTIFICATION STUDY SURVEY

This survey was developed prior to the justification study, and used in the justification study to identify new factors that may improve the performance of PreSS. These factors were then collected as part of the main study (Chapter 3.2) and examined further in Section 5.3.

1. Age Bracket?
2. Gender?
3. How many hours per day would you play computer games on a mobile device?
4. If you play games on a mobile device, what genre of games do you play the most?
5. How many hours per day would you play computer games on a Console, PC or laptop?
6. If you play games on a console, PC or laptop, what genre of games do you play the most?
7. How many hours per day would you use the internet (not including social media or messaging services)?
8. What would your primary use of the internet consist of (not including social media or messaging services)?
9. How many hours per day would you use a social networking service?
10. If you do use social networking, what particular service do you use the most?
11. How many hours per day would you use a messaging service?
12. If you do use messaging service, what particular service do you use the most?

# E

# APPENDIX E

## E.1 PROGRAMMING SELF-EFFICACY QUESTIONNAIRE

Each question had four possible answers:

1. Strongly agree
2. Agree
3. Disagree
4. Strongly disagree

Q1) On a whole I am satisfied with my programming progress?

Q2) At times I think that I am no good at all at programming?

Q3) I feel that I have a number of good programming qualities?

Q4) I am able to complete programming items as well as most other students in my class?

Q5) I feel that I do not have much programming ability to be proud of?

Q6) I certainly feel useless at programming at times?

Q7) I feel that I am a person of worth, at least on a plane with other programmers in my class?

Q8) I wish I could have more respect for my programming ability?

Q9) All in all, I am inclined to feel that I am a failure at programming?

Q10) I take a positive attitude towards my programming ability?

F

# APPENDIX F

## F.1 MINDSET SURVEY

The adopted Mindset survey, from the work of Dweck [46], by D'Anca [37]. Each question had four possible answers:

1. Strongly agree
2. Agree
3. Disagree
4. Strongly disagree

Q1 ) Intelligence is something people are born with that can't be changed.

Q2 ) No matter how intelligent you are, you can always be more intelligent.

Q3 ) You can always substantially change how intelligent you are.

Q4 ) You are a certain kind of person, and there is not much that can be done to really change that.

Q5 ) You can always change basic things about the kind of person you are.

Q6 ) Musical talent can be learned by anyone.

Q7 ) Only a few people will be truly good at sports – you have to be "born with it."

Q8 ) Math is much easier to learn if you are male or maybe come from a culture who values math.

Q9 ) The harder you work at something, the better you will be at it.

Q10) No matter what kind of person you are, you can always change substantially.

Q11) Trying new things is stressful for me and I avoid it.

Q12) Some people are good and kind, and some are not – it's not often that people change.

Q13) I appreciate when people, parents, coaches, teachers give me feedback about my performance.

Q14) I often get angry when I get negative feedback about my performance.

Q15) All human beings are capable of learning.

Q16) You can learn new things, but you can't really change how intelligent you are.

Q17) You can do things differently, but the important parts of who you are can't really be changed.

Q18) Human beings are basically good, but sometimes make terrible decisions.

Q19) An important reason why I do my school work is that I like to learn new things.

Q20) Truly smart people do not need to try hard.

# BIBLIOGRAPHY

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[2] Albert K Adusah and Gordon P Brooks. "Type I Error Inflation of the Separate-Variances Welch t test with Very Small Sample Sizes when Assumptions Are Met". In: *Journal of Modern Applied Statistical Methods* 10.1 (2011), p. 33. DOI: `10.22237/jmasm/1304224320`.

[3] Alireza Ahadi, Raymond Lister, Heikki Haapala, and Arto Vihavainen. "Exploring Machine Learning Methods to Automatically Identify Students in Need of Assistance". In: *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*. ICER '15. Omaha, Nebraska, USA: ACM, 2015, pp. 121–130. ISBN: 978-1-4503-3630-7. URL: `http://doi.acm.org/10.1145/2787622.2787717`.

[4] J Allert. "Learning style and factors contributing to success in an introductory computer science course". In: *IEEE International Conference on Advanced Learning Technologies 2004 Proceedings* 33.1 (2004), pp. 385–389. DOI: `10.1109/ICALT.2004.1357442`.

[5] Michal Armoni, Orni Meerbaum-Salant, and Mordechai Ben-Ari. "From Scratch to "Real"; Programming". In: *Trans. Comput. Educ.* 14.4 (Feb. 2015), 25:1–25:15. ISSN: 1946-6226. DOI: `10.1145/2677087`. URL: `http://doi.acm.org/10.1145/2677087`.

[6] Alison Attrill. *Cyberpsychology*. Ed. by Alison Attrill. 1st. Oxford, United Kingdom: Oxford University Press, Impression 1, 2015. ISBN: 978-0-19-871258-9.

[7] Mustafa Başer. “Attitude , Gender and Achievement in Computer Programming”. In: *Middle-East Journal of Scientific Research* 14.2 (2013), pp. 248–255. ISSN: 1990-9233.

[8] Lecia J. Barker, Charlie McDowell, and Kimberly Kalahar. “Exploring Factors That Influence Computer Science Introductory Course Students to Persist in the Major”. In: *SIGCSE Bull.* 41.1 (Mar. 2009), pp. 153–157. ISSN: 0097-8418. DOI: 10.1145/1539024.1508923. URL: http://doi.acm.org/10.1145/1539024.1508923.

[9] Ricky J. Barker and E. A. Unger. “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*. SIGCSE ’83. Orlando, Florida, USA: ACM, 1983, pp. 154–158. ISBN: 0-89791-091-5. DOI: 10.1145/800038.801037. URL: http://doi.acm.org/10.1145/800038.801037.

[10] Youssef Bassil. “A Comparative Study on the Performance of the Top DBMS Systems”. In: *Journal of Computer Science and Research (CoRR)* abs/1205.2889 (2012). arXiv: 1205.2889. URL: http://arxiv.org/abs/1205.2889.

[11] Brett Becker and Keith Quille. “50 Years of CS1 at SIGCSE: A Review of the Evolution of Introductory Programming Education Research”. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. SIGCSE ’19. New York, NY, USA: ACM, 2019. DOI: 10.1145/3287324.3287432. URL: http://doi.acm.org/10.1145/3287324.3287432.

[12] Jens Bennedsen and Michael E. Caspersen. “An Investigation of Potential Success Factors for an Introductory Model-driven Programming Course”. In: ICER ’05 (2005), pp. 155–163. DOI: 10.1145/1089786.1089801. URL: http://doi.acm.org/10.1145/1089786.1089801.

[13] Jens Bennedsen and Michael E. Caspersen. "Abstraction Ability As an Indicator of Success for Learning Object-oriented Programming?" In: *SIGCSE Bull.* 38.2 (June 2006), pp. 39–43. ISSN: 0097-8418. DOI: 10.1145/1138403.1138430. URL: http://doi.acm.org/10.1145/1138403.1138430.

[14] Jens Bennedsen and Michael E. Caspersen. "Failure Rates in Introductory Programming". In: *SIGCSE Bull.* 39.2 (June 2007), pp. 32–36. ISSN: 0097-8418. DOI: 10.1145/1272848.1272879. URL: http://doi.acm.org/10.1145/1272848.1272879.

[15] Jens Bennedsen and Michael E. Caspersen. "Optimists have more fun, but do they learn better? On the influence of emotional and social factors on learning introductory computer science". In: *Computer Science Education* 18.1 (2008), pp. 1–16. DOI: 10.1080/08993400701791133.

[16] Susan Bergin. "Statistical and Machine Learning Models to Predict Programming Performance". PhD thesis. Maynooth University, 2006. URL: http://mural.maynoothuniversity.ie/5314/.

[17] Susan Bergin and Ronan Reilly. "Programming: Factors That Influence Success". In: *SIGCSE Bull.* 37.1 (2005), pp. 411–415. ISSN: 0097-8418. DOI: 10.1145/1047124.1047480. URL: http://doi.acm.org/10.1145/1047124.1047480.

[18] Susan Bergin and Ronan Reilly. "Predicting introductory programming performance: A multi-institutional multivariate study". In: *Computer Science Education* 16.4 (2006), pp. 303–323. DOI: 10.1080/08993400600997096. URL: https://doi.org/10.1080/08993400600997096.

[19] Susan Bergin, Aidan Mooney, John Ghent, and Keith Quille. "Using Machine Learning Techniques to Predict Introductory Programming Performance". In: *International Journal of Computer Sci-*

*ence and Software Engineering* 4.12 (2015), pp. 323–328. ISSN: 2409-4285. URL: http://mural.maynoothuniversity.ie/8682/.

[20] Sylvia Beyer, Kristina Rynes, Julie Perrault, Kelly Hay, and Susan Haller. "Gender Differences in Computer Science Students". In: *SIGCSE Bull.* 35.1 (Jan. 2003), pp. 49–53. ISSN: 0097-8418. DOI: 10.1145/792548.611930. URL: http://doi.acm.org/10.1145/792548.611930.

[21] Sylvia Beyer, Michelle DeKeuster, Kathleen Walter, Michelle Colar, and Christina Holcomb. "Changes in CS Students' Sttitudes Towards CS over Time: An Examination of Gender Differences". In: *SIGCSE Bull.* 37.1 (Feb. 2005), pp. 392–396. ISSN: 0097-8418. DOI: 10.1145/1047124.1047475. URL: http://doi.acm.org/10.1145/1047124.1047475.

[22] Gary D. Boetticher, Wei Ding, Charles Moen, and Kwok-Bun Yue. "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*. SIGCSE '05. St. Louis, Missouri, USA: ACM, 2005, pp. 500–504. DOI: 10.1145/1047124.1047503. URL: http://doi.acm.org/10.1145/1047124.1047503.

[23] Richard Bornat, Saeed Dehnadi, and Simon. "Mental Models, Consistency and Programming Aptitude". In: *Proceedings of the Tenth Conference on Australasian Computing Education - Volume 78*. ACE '08. Wollongong, NSW, Australia: Australian Computer Society, Inc., 2008, pp. 53–61. ISBN: 978-1-920682-59-0. URL: http://dl.acm.org/citation.cfm?id=1379249.1379253.

[24] Jason Brownlee. "Deep Learning with Python: Develop Deep Learning Models on Theano and TensorFlow Using Keras". In: *Machine Learning Mastery, Melbourne* (2016). URL: https://books.google.ie/books?id=eJw2nQAACAAJ.

[25] C. Buchmann, D. J. Condron, and V. J. Roscigno. "Shadow Education, American Style: Test Preparation, the SAT and College Enrollment". In: *Social Forces* 89.2 (2010), pp. 435–461. ISSN: 0037-7732. DOI: 10.1353/sof.2010.0105.

[26] D. F. Butcher and W. A. Muth. "Predicting Performance in an Introductory Computer Science Course". In: *Commun. ACM* 28.3 (Mar. 1985), pp. 263–268. ISSN: 0001-0782. DOI: 10.1145/3166.3167. URL: http://doi.acm.org/10.1145/3166.3167.

[27] Tracy Camp. "The incredible shrinking pipeline". In: *Communications of the ACM* 40.10 (1997), pp. 103–110. ISSN: 00010782. DOI: 10.1145/262793.262813. arXiv: 9703102v1 [hep-th]. URL: http://portal.acm.org/citation.cfm?doid=262793.262813.

[28] Jennifer Campbell, Diane Horton, and Michelle Craig. "Factors for Success in Online CS1". In: *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE '16. Arequipa, Peru: ACM, 2016, pp. 320–325. ISBN: 978-1-4503-4231-5. DOI: 10.1145/2899415.2899457. URL: http://doi.acm.org/10.1145/2899415.2899457.

[29] C. K. Capstick, J. D. Gordon, and A. Salvadori. "Predicting Performance by University Students in Introductory Computing". In: *SIGCSE Bull.* 7.3 (Sept. 1975), pp. 21–29. ISSN: 0097-8418. DOI: 10.1145/382216.382483. URL: http://doi.acm.org/10.1145/382216.382483.

[30] Michael E. Caspersen, Kasper Dalgaard Larsen, and Jens Bennedsen. "Mental Models and Programming Aptitude". In: *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. ITiCSE '07. Dundee, Scotland: ACM, 2007, pp. 206–210. ISBN: 978-1-59593-610-3. DOI: 10.1145/1268784.1268845. URL: http://doi.acm.org/10.1145/1268784.1268845.

[31] François Chollet et al. *Keras: The Python Deep Learning library*. https://keras.io. 2015.

[32] Pauline Rose Clance and Suzanne Ament Imes. "The imposter phenomenon in high achieving women: Dynamics and therapeutic intervention." In: *Psychotherapy: Theory, Research & Practice* 15.3 (1978), p. 241. DOI: 10.1037/h0086006.

[33] Ronán Conroy. "Sample Size: A rough guide". In: *Beaumontethics* (2015). URL: https://beaumontethics.ie/docs/application/samplesizecalculation.pdf.

[34] Diana Cukierman. "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*. ITiCSE '15. Vilnius, Lithuania: ACM, 2015, pp. 248–253. ISBN: 978-1-4503-3440-2. DOI: 10.1145/2729094.2742623. URL: http://doi.acm.org/10.1145/2729094.2742623.

[35] Natalie Culligan, Keith Quille, and Susan Bergin. "VEAP: A Visualisation Engine and Analyzer for PreSS#". In: *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*. Koli Calling '16. Koli, Finland: ACM, 2016, pp. 130–134. ISBN: 978-1-4503-4770-9. DOI: 10.1145/2999541.2999553. URL: http://doi.acm.org/10.1145/2999541.2999553.

[36] Quintin Cutts, Emily Cutts, Stephen Draper, Patrick O'Donnell, and Peter Saffrey. "Manipulating Mindset to Positively Influence Introductory Programming Performance". In: SIGCSE '10 (2010), pp. 431–435. DOI: 10.1145/1734263.1734409. URL: http://doi.acm.org/10.1145/1734263.1734409.

[37] Joy-Anne D'Anca. "Mindset and Resilience: An Analysis and Intervention for School Administrators". PhD thesis. 2017. ISBN: 978-1-3697-4954-0. URL: `https://eric.ed.gov/?id=ED576901`.

[38] Gabrielle Maria D'Lima, Adam Winsler, and Anastasia Kitsantas. "Ethnic and gender differences in first-year college students' goal orientation, Self-Efficacy, and extrinsic and intrinsic motivation". In: *Journal of Educational Research* 107.5 (2014), pp. 341–356. ISSN: 19400675. DOI: `10.1080/00220671.2013.823366`.

[39] Inés Friss De Kereki. "Scratch: Applications in Computer Science 1". In: *Proceedings - Frontiers in Education Conference, FIE* (2008), pp. 7–11. ISSN: 15394565. DOI: `10.1109/FIE.2008.4720267`.

[40] Saeed Dehnadi. "Testing programming aptitude". In: *Proceedings of the 18th Annual Workshop of the Psychology of Programming Interest Group (PPIG)*. 2006, pp. 22–37. URL: `http://www.ppig.org`.

[41] Jennifer Dempsey, Richard T Snodgrass, and Isabel Kishi. "The Emerging Role of Self-Perception in Student Intentions Categories and Subject Descriptors". In: *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)* (2015), pp. 108–113. DOI: `10.1145/2676723.2677305`.

[42] Paul Denny, Andrew Luxton-Reilly, John Hamer, Dana B. Dahlstrom, and Helen C. Purchase. "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*. ITiCSE '10. Bilkent, Ankara, Turkey: ACM, 2010, pp. 118–122. ISBN: 978-1-60558-820-9. DOI: `10.1145/1822090.1822124`. URL: `http://doi.acm.org/10.1145/1822090.1822124`.

[43] Katherine Donnelly. *Girls 'just as good as boys' at computer science work - Independent.ie*. 2017. URL: `https://www.independent.ie/`

irish-news/education/girls-just-as-good-as-boys-at-computer-science-work-36229848.html (visited on 06/25/2018).

[44] Katherine Donnelly. “Dramatic fall in number of students entering third-level to study computing”. In: *The Irish Independent*. 2018. URL: https://www.independent.ie/irish-news/education/dramatic-fall-in-number-of-students-entering-thirdlevel-to-study-computing-37605169.html.

[45] Maureen Doyle, Dhanuja Kasturiratna, Bartley D. Richardson, and Suzanne W. Soled. “Computer science and computer information technology majors together: Analyzing factors impacting students’ success in introductory programming”. In: *Proceedings - Frontiers in Education Conference, FIE* (2009), pp. 1–6. ISSN: 15394565. DOI: 10.1109/FIE.2009.5350582.

[46] Carol S Dweck. *Mindset: The new psychology of success*. Random House Digital, Inc., 2008. ISBN: 9780345472328.

[47] Stephen Edwards and Zhiyi Li. “Towards progress indicators for measuring student programming effort during solution development”. In: *Proceedings of the 16th Koli Calling International Conference on Computing Education Research - Koli Calling ’16* (2016), pp. 31–40. DOI: 10.1145/2999541.2999561. URL: http://dl.acm.org/citation.cfm?doid=2999541.2999561.

[48] Anthony Estey and Yvonne Coady. “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*. ITiCSE ’16. Arequipa, Peru: ACM, 2016, pp. 236–241. ISBN: 978-1-4503-4231-5. DOI: 10.1145/2899415.2899428. URL: http://doi.acm.org/10.1145/2899415.2899428.

[49] Gerald E. Evans and Mark G. Simkin. “What best predicts computer proficiency?” In: *Communications of the ACM* 32.11 (1989), pp. 1322–1327. ISSN: 00010782. DOI: 10.1145/68814.68817. URL: http://portal.acm.org/citation.cfm?doid=68814.68817.

[50] Allan Fisher, Jane Margolis, and Faye Miller. “Undergraduate women in computer science: experience, motivation and culture”. In: *ACM SIGCSE Bulletin* 29.1 (1997), pp. 106–110. ISSN: 00978418. DOI: 10.1145/268084.268127. URL: http://dl.acm.org/citation.cfm?id=268127.

[51] Abraham E. Flanigan, Markeya S. Peteranetz, Duane F. Shell, and Leen-Kiat Soh. “Exploring Changes in Computer Science Students’ Implicit Theories of Intelligence Across the Semester”. In: *Proceedings of the eleventh annual International Conference on International Computing Education Research - ICER ’15* (2015), pp. 161–168. ISSN: 0361-476X. URL: http://dl.acm.org/citation.cfm?doid=2787622.2787722.

[52] George C. Fowler and Louis W. Glorfeld. “Predicting Aptitude in Introductory Computing: A Classification Model”. In: *AEDS Journal* 14.2 (1981), pp. 96–109. URL: http://dx.doi.org/10.1080/00011037.1981.11008293.

[53] Guillermo A Francia III and Rahjima R Francia. “An Empirical Study on the Performance of Java/. Net Cryptographic APIs”. In: *Information Systems Security* 16.6 (2007), pp. 344–354. DOI: 10.1080/10658980701784602. URL: https://doi.org/10.1080/10658980701784602.

[54] Carol Frieze and Jeria L Quesenberry. “From Difference to Diversity : Including Women in The Changing Face of Computing”. In: *Proceeding of the 44th ACM technical symposium on Computer science education - SIGCSE ’13* (2013), pp. 445–450. DOI: 10.1145/2445196.2445327.

[55] Alexandra Funke, Marc Berges, Andreas Mühling, and Peter Hubwieser. “Gender Differences in Programming : Research Results and Teachers ’ Perception”. In: *Koli Calling Conference on Computing Education Research* (2015), pp. 161–162. DOI: 10.1145/2828959.2828982.

[56] John Ghent and John McDonald. “A computational model of facial expression”. PhD thesis. National University of Ireland, Maynooth, 2005.

[57] Louis W. Glorfeld and George C. Fowler. “Validation of a Model for Predicting Aptitude for Introductory Computing”. In: *Proceedings of the Thirteenth SIGCSE Technical Symposium on Computer Science Education*. SIGCSE ’82. Indianapolis, Indiana, USA: ACM, 1982, pp. 140–143. ISBN: 0-89791-067-2. DOI: 10.1145/800066.801355. URL: http://doi.acm.org/10.1145/800066.801355.

[58] Paul Golding, Lisa Facey-Shaw, and Vanesa Tennant. “Effects of peer tutoring, attitude and personality on academic performance of first year introductory programming students”. In: *Proceedings - Frontiers in Education Conference, FIE* (2006), pp. 7–12. ISSN: 15394565. DOI: 10.1109/FIE.2006.322662.

[59] Virginia Grande and Joachim Parrow. “Motivation and Grade Gap Related to Gender in a Programming Course”. In: *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE ’15)*. ACM, 2015, pp. 349–349. ISBN: 9781450334402. DOI: 10.1145/2729094.2754858.

[60] Terry R. Hostetler. “Predicting Student Success in an Introductory Programming Course”. In: *SIGCSE Bull.* 15.3 (Sept. 1983), pp. 40–43. ISSN: 0097-8418. DOI: 10.1145/382188.382571. URL: http://doi.acm.org/10.1145/382188.382571.

[61] Bowen Hui and Shannon Farvolden. “How can learning analytics improve a course?” In: *Proceedings of the WCCCE 2017: 22nd Western Canadian Conference on Computing Education* (2017). DOI: 10.1145/3085585.3085586.

[62] Petri Ihantola et al. “Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies”. In: *Proceedings of the 2015 ITiCSE on Working Group Reports*. ITICSE-WGR ’15. Vilnius, Lithuania: ACM, 2015, pp. 41–63. ISBN: 978-1-4503-4146-2. DOI: 10.1145/2858796.2858798. URL: http://doi.acm.org/10.1145/2858796.2858798.

[63] Sandra Katz, John Aronis, David Allbritton, Christine Wilson, and Mary Lou Soffa. “A study to identify predictors of achievement in an introductory computer science course”. In: *Proceedings of the 2003 SIGMIS conference on Computer personnel research Freedom in Philadelphia–leveraging differences and diversity in the IT workforce - SIGMIS CPR ’03* (2003), p. 157. DOI: 10.1145/761849.761879. URL: http://portal.acm.org/citation.cfm?doid=761849.761879.

[64] J O Kelly, a Mooney, J Ghent, P Gaughran, S Dunne, and S Bergin. “An Overview of the Integration of Problem Based Learning into an existing Computer Science Programming Module 2 Overview of our PBL Implementation”. In: *Problem-Based Learning International Conference 2004: Pleasure by Learning* (2004). URL: http://mural.maynoothuniversity.ie/726/.

[65] Hassan Khosravi and Kendra M.L. Cooper. “Using Learning Analytics to Investigate Patterns of Performance and Engagement in Large Classes”. In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education - SIGCSE ’17* (2017), pp. 309–314. DOI: 10.1145/3017680.3017711. URL: http://dl.acm.org/citation.cfm?doid=3017680.3017711.

[66] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (2014), pp. 1–15. ISSN: 09252312. DOI: http://doi.acm.org.ezproxy.lib.ucf.edu/10.1145/1830483.1830503. arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

[67] John Konvalina, Stanley A. Wileman, and Larry J. Stephens. "Math Proficiency: A Key to Success for Computer Science Students". In: *Commun. ACM* 26.5 (May 1983), pp. 377–382. ISSN: 0001-0782. DOI: 10.1145/69586.358140. URL: http://doi.acm.org/10.1145/69586.358140.

[68] Gerald Kotonya and Ian Sommerville. *Requirements Engineering: Processes and Techniques*. 1st Edition. Wiley Publishing, 1998. ISBN: 047-197-2088. URL: https://dl.acm.org/citation.cfm?id=552009.

[69] Sotiris Kotsiantis, Christos Pierrakeas, and Panagiotis Pintelas. "Predicting students' performance in distance learning using machine learning techniques". In: *Applied Artificial Intelligence* 18.5 (2004), pp. 411–426. DOI: 10.1080/08839510490442058. URL: https://doi.org/10.1080/08839510490442058.

[70] Samantha Krieger, Meghan Allen, and Catherine Rawn. "Are Females Disinclined to Tinker in Computer Science?" In: SIGCSE '15 (2015), pp. 102–107. DOI: 10.1145/2676723.2677296. URL: http://doi.acm.org/10.1145/2676723.2677296.

[71] Lynn Lambert. "Factors That Predict Success in CS1". In: *J. Comput. Sci. Coll.* 31.2 (Dec. 2015), pp. 165–171. ISSN: 1937-4771. URL: http://dl.acm.org/citation.cfm?id=2831432.2831458.

[72] R. R. Leeper and J. L. Silver. "Predicting Success in a First Programming Course". In: *Proceedings of the Thirteenth SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '82. Indi-

anapolis, Indiana, USA: ACM, 1982, pp. 147–150. ISBN: 0-89791-067-2. DOI: 10.1145/800066.801357. URL: http://doi.acm.org/10.1145/800066.801357.

[73] Juho Leinonen, Leo Leppänen, Petri Ihantola, and Arto Hellas. "Comparison of Time Metrics in Programming". In: *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ICER '17. Tacoma, Washington, USA: ACM, 2017, pp. 200–208. ISBN: 978-1-4503-4968-0. DOI: 10.1145/3105726.3106181. URL: http://doi.acm.org/10.1145/3105726.3106181.

[74] Soohyun Nam Liao, Daniel Zingaro, Michael A. Laurenzano, William G. Griswold, and Leo Porter. "Lightweight, Early Identification of At-Risk CS1 Students". In: *Proceedings of the 2016 ACM Conference on International Computing Education Research*. ICER '16. Melbourne, VIC, Australia: ACM, 2016, pp. 123–131. ISBN: 978-1-4503-4449-4. DOI: 10.1145/2960310.2960315. URL: http://doi.acm.org/10.1145/2960310.2960315.

[75] Alex Lishinski, Aman Yadav, Jon Good, and Richard Enbody. "Learning to Program: Gender Differences and Interactive Effects of Students ' Motivation , Goals and Self-Efficacy on Performance". In: *Proceedings of the 12th International Computing Education Research Conference* (2016), pp. 211–220. DOI: 10.1145/2960310.2960329. URL: http://doi.acm.org/10.1145/2960310.2960329.

[76] Alex Lishinski, Aman Yadav, Richard Enbody, and Jon Good. "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*. SIGCSE '16. Memphis, Tennessee, USA: ACM, 2016, pp. 329–334. ISBN: 978-1-4503-3685-7. DOI: 10.1145/2839509.2844596. URL: http://doi.acm.org/10.1145/2839509.2844596.

[77] Miriam Liston, Denise Frawley, and Vivienne Patterson. "A study of progression in Irish higher education". In: *Higher Education Authority* (2016). URL: http://www.hea.ie/en/publications/2016.

[78] Emily Lovell. "Promoting constructive mindsets for overcoming failure in computer science education". In: *Proceedings of the tenth annual conference on International computing education research - ICER '14* (2014), pp. 159–160. DOI: 10.1145/2632320.2632331. URL: http://dl.acm.org/citation.cfm?doid=2632320.2632331.

[79] David J Malan and Henry H Leitner. "Scratch for Budding Computer Scientists". In: *Sigcse 2007: Proceedings of the Thirty-Eighth Sigcse Technical Symposium on Computer Science Education* (2007), pp. 223–227. ISSN: 00978418. DOI: 10.1145/1227504.1227388.

[80] J.H.F. Meyer and Ray Land. "Threshold concepts and troublesome knowledge: Linkages to ways of thinking and practising within the disciplines". In: *Improving Student Learning – Ten Years On.* 4.1 (2003), pp. 1–16. ISSN: 0018-1560. DOI: 10.1007/978-3-8348-9837-1.

[81] Tom M Mitchell et al. "Machine learning. 1997". In: *Burr Ridge, IL: McGraw Hill* 45.37 (1997), pp. 870–877. URL: https://www.worldcat.org/title/machine-learning/oclc/36417892.

[82] Oliver Mooney, Vivienne Patterson, Muiris OConnor, and Abigail Chantler. "A study of progression in Irish higher education". In: *Higher Education Authority, Dublin* (2010). URL: http://hdl.voced.edu.au/10707/195435.

[83] Patrick Morrison and Emerson Murphy-Hill. "Is Programming Knowledge Related to Age? An Exploration of Stack Overflow". In: *Proceedings of the 10th Working Conference on Mining Software Repositories*. MSR '13. San Francisco, CA, USA: IEEE Press, 2013,

pp. 69–72. ISBN: 978-1-4673-2936-1. URL: http://dl.acm.org/citation.cfm?id=2487085.2487102.

[84] Laurie Murphy and Lynda Thomas. "Dangers of a fixed mindset: implications of self-theories research for computer science education". In: *ACM SIGCSE Bulletin* 40.3 (2008), pp. 271–275. ISSN: 00978418. DOI: 10.1145/1597849.1384344. URL: http://dl.acm.org/citation.cfm?id=1384271.1384344.

[85] Laurie Murphy, Brad Richards, Renée McCauley, Briana B. Morrison, Suzanne Westbrook, and Timothy Fossum. "Women catch up: gender differences in learning programming concepts". In: *Technical Symposium on Computer Science Education* 38 (2006), p. 17. ISSN: 0097-8418. DOI: 10.1145/1121341.1121350. URL: http://portal.acm.org/citation.cfm?id=1121341.1121350.

[86] L Naing, T Winn, and B N Rusli. "Practical Issues in Calculating the Sample Size for Prevalence Studies". In: *Archives of Orofacial Sciences* 1.Ci (2006), pp. 9–14. ISSN: 1823-8602. DOI: 10.1146/annurev.psych.60.110707.163629.

[87] National Center for Science and Engineering Statistics. *Field of Degree: Women - US National Science Foundation (NSF)*. 2012. URL: https://www.nsf.gov/statistics/2015/nsf15311/digest/theme2.cfm/psychology (visited on 12/28/2016).

[88] Peter R. Newsted. "Grade and Ability Predictions in an Introductory Programming Course". In: *SIGCSE Bull.* 7.2 (June 1975), pp. 87–91. ISSN: 0097-8418. DOI: 10.1145/382205.382897. URL: http://doi.acm.org/10.1145/382205.382897.

[89] Keith Nolan and Susan Bergin. "The role of anxiety when learning to program : A Systematic review of the literature". In: *Proceedings of the 16th Koli Calling International Conference on Comput-*

*ing Education Research*. Koli, Finland: ACM, 2016, pp. 61–70. ISBN: 9781450347709. DOI: 10.1145/2999541.2999557.

[90] Aura Paloheimo and Jenni Stenman. "Gender, communication and comfort level in higher level computer science education - Case study". In: *Proceedings - Frontiers in Education Conference, FIE* (2006), pp. 13–18. ISSN: 15394565. DOI: 10.1109/FIE.2006.322486.

[91] Reena Pau, Wendy Hall, Marcus Grace, and John Woollard. "Female Students ' Experiences of Programming : It ' s Not All Bad !" In: *16th annual joint conference on Innovation and technology in computer science education* (2011), pp. 323–327. DOI: 10.1145/1999747.1999837.

[92] Paul R. Pintrich and Elisabeth V. De Groot. "Motivational and Self-Regulated Learning Components of Classroom Academic Performance". In: *Journal of Educational Psychology* 82.1 (1990), pp. 33–40. ISSN: 0022-0663. DOI: 10.1037/0022-0663.82.1.33. arXiv: 90 [0022-0663].

[93] Paul R.; Pintrich, Others, and A. "A Manual for the use of the Motivated Strategies for Learning Questionnaire (MSLQ)". In: *Mediterranean Journal of Social Sciences* 6.1 (1991), pp. 156–164. ISSN: 20399340. DOI: 10.5901/mjss.2015.v6n1p156. arXiv: arXiv:1011.1669v3. URL: http://link.springer.com/10.1007/s10869-013-9342-5.

[94] B T Pioro. "Estimated and Actual Performance Scores on Computer Programming Tasks". In: *SoutheastCon, 2004. Proceedings. IEEE* (2004), pp. 3–7. ISSN: 07347502. DOI: 10.1109/SECON.2004.1287950.

[95] Leo Porter and Daniel Zingaro. "Importance of Early Performance in CS1: Two Conflicting Assessment Stories". In: SIGCSE

'14 (2014), pp. 295–300. DOI: 10.1145/2538862.2538912. URL: http://doi.acm.org/10.1145/2538862.2538912.

[96] Leo Porter, Daniel Zingaro, and Raymond Lister. "Predicting Student Success Using Fine Grain Clicker Data". In: *Proceedings of the Tenth Annual Conference on International Computing Education Research*. ICER '14. Glasgow, Scotland, United Kingdom: ACM, 2014, pp. 51–58. ISBN: 978-1-4503-2755-8. DOI: 10.1145/2632320.2632354. URL: http://doi.acm.org/10.1145/2632320.2632354.

[97] K Quille and S Bergin. "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*. 2015. URL: http://www.keithquille.com/PublicationData/ICEP_2015.pdf.

[98] Keith Quille and Susan Bergin. "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*. 2016. URL: http://www.keithquille.com/PublicationData/ICEP2016.pdf.

[99] Keith Quille and Susan Bergin. "Programming: Further Factors that Influence Success". In: *Proceedings of the Psychology of Programming Interest Group (PPIG), 7th to 10th Spetember, University of Cambridge*. 2016. URL: keithquille.com/PublicationData/ppig2016.pdf.

[100] Keith Quille and Susan Bergin. "Programming: Predicting Student Success Early in CS1. A Re-validation and Replication Study". In: *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE 2018. Larnaca, Cyprus: ACM, 2018, pp. 15–20. ISBN: 978-1-4503-5707-4.

DOI: 10.1145/3197091.3197101. URL: http://doi.acm.org/10.1145/3197091.3197101.

[101] Keith Quille and Susan Bergin. "CS1: how will they do? How can we help? A decade of research and practice". In: *Computer Science Education* 29.2-3 (2019), pp. 254–282. DOI: 10.1080/08993408.2019.1612679. URL: https://doi.org/10.1080/08993408.2019.1612679.

[102] Keith Quille, Susan Bergin, and Aidan Mooney. "PreSS#, A Web-Based Educational System to Predict Programming Performance". In: *International Journal of Computer Science and Software Engineering (IJCSSE)* 4.7 (2015), pp. 178–189. URL: http://www.keithquille.com/PublicationData/PreSS.pdf.

[103] Keith Quille, Natalie Culligan, and Susan Bergin. "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*. ITiCSE '17. Bologna, Italy: ACM, 2017, pp. 263–268. ISBN: 978-1-4503-4704-4. DOI: 10.1145/3059009.3059048. URL: http://doi.acm.org/10.1145/3059009.3059048.

[104] Keith Quille, Roisin Faherty, Susan Bergin, and Brett A Becker. "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, 2018. DOI: 10.1145/3279720.3279742.

[105] Katie Redmond, Sarah Evans, and Mehran Sahami. "A large-scale quantitative study of women in computer science at Stanford University". In: *Proceeding of the 44th ACM technical symposium on Computer science education - SIGCSE '13* (2013), pp. 439–444. DOI: 10.1145/2445196.2445326.

[106] Ma. Mercedes T. Rodrigo, Emily S. Tabanao, Ryan S. Baker, Matthew C. Jadud, Anna Christine M. Amarra, Thomas Dy, Maria Beatriz V. Espejo-Lahoz, Sheryl Ann L. Lim, Sheila A.M.S. Pascua, and Jessica O. Sugay. "Affective and behavioral predictors of novice programmer achievement". In: *ACM SIGCSE Bulletin* 41.3 (2009), p. 156. ISSN: 00978418. DOI: 10.1145/1595496.1562929. URL: http://portal.acm.org/citation.cfm?doid=1595496.1562929.

[107] Morris Rosenberg. *Society and the adolescent self-image*. Vol. 148. 3671. American Association for the Advancement of Science, 1965, pp. 804–804. DOI: 10.1126/science.148.3671.804. URL: https://science.sciencemag.org/content/148/3671/804.

[108] Guido Rossum. *Python Reference Manual*. Tech. rep. Amsterdam, The Netherlands, The Netherlands, 1995. URL: http://www.ncstrl.org:8900/ncstrl/servlet/search?formname=detail\&id=oai%3Ancstrlh%3Aercim_cwi%3Aercim.cwi%2F%2FCS-R9525.

[109] Nathan Rountree, Janet Rountree, and Anthony Robins. "Predictors of Success and Failure in a CS1 Course". In: *SIGCSE Bull.* 34.4 (Dec. 2002), pp. 121–124. ISSN: 0097-8418. DOI: 10.1145/820127.820182. URL: http://doi.acm.org/10.1145/820127.820182.

[110] Nathan Rountree, Janet Rountree, Anthony Robins, and Robert Hannah. "Interacting Factors That Predict Success and Failure in a CS1 Course". In: *SIGCSE Bull.* 36.4 (June 2004), pp. 101–104. ISSN: 0097-8418. DOI: 10.1145/1041624.1041669. URL: http://doi.acm.org/10.1145/1041624.1041669.

[111] Ginger Holmes Rowell, Diane G. Perhac, Judith A. Hankins, Brenda C. Parker, Chrisila C. Pettey, and Judith M. Iriarte-Gross. "Computer-related Gender Differences". In: *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '03. Reno, Navada, USA: ACM, 2003, pp. 54–58. ISBN: 1-58113-648-X.

DOI: 10.1145/611892.611931. URL: http://doi.acm.org/10.1145/611892.611931.

[112] Miguel Angel Rubio, Rocio Romero-Zaliz, Carolina Mañoso, and Angel P. de Madrid. "Closing the gender gap in an introductory programming course". In: *Computers & Education* 82 (2015), pp. 409–420. ISSN: 03601315. DOI: 10.1016/j.compedu.2014.12.003. URL: http://www.sciencedirect.com/science/article/pii/S0360131514002802.

[113] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016. URL: http://thuvien.thanglong.edu.vn:8081/dspace/bitstream/DHTL_123456789/4010/1/CS503-2.pdf.

[114] Jérémie Blanchette Sarrasin, Lucian Nenciovici, Lorie Marlène Brault Foisy, Geneviève Allaire-Duquette, Martin Riopel, and Steve Masson. "Effects of teaching the concept of neuroplasticity to induce a growth mindset on motivation, achievement, and brain activity: A meta-analysis". In: *Trends in Neuroscience and Education* 12.January (2018), pp. 22–31. ISSN: 22119493. DOI: 10.1016/j.tine.2018.07.003. URL: https://doi.org/10.1016/j.tine.2018.07.003.

[115] Donna Saulsberry. "Dwindling Number of Female Students: What Are We Missing?" In: SIGITE '12 (2012), pp. 221–226. DOI: 10.1145/2380552.2380616. URL: http://doi.acm.org/10.1145/2380552.2380616.

[116] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001. ISBN: 0262194759.

[117] NetCraft Internet Services. *A survey of 876,812,666 sites, detailing market share of web servers*. http://news.netcraft.com/

archives/2015/01/15/january-2015-web-server-survey.html. 2015, accessed 22/05/2015.

[118] Duane F. Shell, Leen-Kiat Soh, Abraham E. Flanigan, and Markeya S. Peteranetz. "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*. SIGCSE '16. Memphis, Tennessee, USA: ACM, 2016, pp. 639–644. ISBN: 978-1-4503-3685-7. DOI: 10.1145/2839509.2844606. URL: http://doi.acm.org/10.1145/2839509.2844606.

[119] Dermot Shinners-Kennedy and Sally A. Fincher. "Identifying threshold concepts". In: *Proceedings of the ninth annual international ACM conference on International computing education research - ICER '13* (2013), p. 9. DOI: 10.1145/2493394.2493396. URL: http://dl.acm.org/citation.cfm?doid=2493394.2493396.

[120] George Siemens. "Learning Analytics: The Emergence of a Discipline". In: *American Behavioral Scientist* 57.10 (2013), pp. 1380–1400. ISSN: 00027642. DOI: 10.1177/0002764213498851.

[121] Nikki Sigurdson and Andrew Petersen. "An Exploration of Grit in a CS1 Context". In: Koli Calling '18 (2018), 23:1–23:5. DOI: 10.1145/3279720.3279743. URL: http://doi.acm.org/10.1145/3279720.3279743.

[122] Simon et al. "Predictors of Success in a First Programming Course". In: *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*. ACE '06. Hobart, Australia: Australian Computer Society, Inc., 2006, pp. 189–196. ISBN: 1-920682-34-1. URL: http://dl.acm.org/citation.cfm?id=1151869.1151894.

[123] B. Simon, B. Hanks, L. Murphy, S. Fitzgerald, R. McCauley, L. Thomas, and C. Zander. "Saying isn't necessarily believing: Influencing self-theories in computing". In: *Proceeding of the fourth in-*

*ternational workshop on Computing education research* (2008), pp. 173–184. DOI: 10.1145/1404520.1404537. URL: http://portal.acm.org/citation.cfm?id=1404537.

[124] Jane Sinclair and Sara Kalvala. "Exploring Societal Factors Affecting the Experience and Engagement of First Year Female Computer Science Undergraduates". In: *Proceedings of the 15th Koli Calling Conference on Computing Education Research* (2015), pp. 107–116. DOI: 10.1145/2828959.2828979. URL: http://doi.acm.org/10.1145/2828959.2828979.

[125] Lindsay I Smith. *A tutorial on principal components analysis*. Tech. rep. 2002. URL: https://ourarchive.otago.ac.nz/bitstream/handle/10523/7534/OUCS-2002-12.pdf.

[126] C.R. Souza. *Accord.Net Framework, Open source Machine learning and Data mining Framework,* http://accordframework.net/index.html. 2012.

[127] César Roberto de Souza. "A tutorial on principal component analysis with the accord. net framework". In: *Department of Computing, Federal University of Sao Carlos, Technical Report.* (2012). URL: https://arxiv.org/abs/1210.7463.

[128] Alexander D Stajkovic and Fred Luthans. "Social cognitive theory and self-efficacy: Implications for motivation theory and practice". In: *Motivation and work behavior* 126 (2003), p. 140. URL: https://www.researchgate.net/publication/258995495_Social_cognitive_theory_and_self-efficacy_Implications_for_motivation_theory_and_practice.

[129] Charles Steinfield, Nicole B. Ellison, and Cliff Lampe. "Social capital, self-esteem, and use of online social network sites: A longitudinal analysis". In: *Journal of Applied Developmental Psychology*

29.6 (2008), pp. 434–445. ISSN: 01933973. DOI: 10.1016/j.appdev.2008.07.002.

[130] Barbara G. Tabachnick and Linda S. Fidell. *Using multivariate statistics*. English. 7th. Boston: Pearson Education, 2001. ISBN: 978-0-13-479054-1. URL: https://www.pearsonhighered.com/assets/preface/0/1/3/4/0134790545.pdf.

[131] William T. Tarimo, Fatima Abu Deeb, and Timothy J. Hickey. "Early Detection of At-risk Students in CS1 Using Teachback/Spinoza". In: *J. Comput. Sci. Coll.* 31.6 (June 2016), pp. 105–111. ISSN: 1937-4771. URL: http://dl.acm.org/citation.cfm?id=2904446.2904471.

[132] T. Tieleman and G Hinton. "Lecture 6.5 - RMSProp, COURSERA: Neural Networks for Machine Learning". In: *University of Toronto, Technical Report* (2012). URL: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[133] Des Traynor, Susan Bergin, and J. Paul Gibson. "Automated Assessment in CS1". In: ACE '06 (2006), pp. 223–228. URL: http://dl.acm.org/citation.cfm?id=1151869.1151898.

[134] Maarten Vansteenkiste, Willy Lens, and Edward L. Deci. "Intrinsic Versus Extrinsic Goal Contents in Self-Determination Theory: Another Look at the Quality of Academic Motivation". In: *Educational Psychologist* 41.1 (2006), pp. 19–31. ISSN: 0046-1520. DOI: 10.1207/s15326985ep4101_4.

[135] Philip R. Ventura. "Identifying predictors of success for an objects-first CS1". In: *Computer Science Education* 15.3 (2005), pp. 223–243. ISSN: 0899-3408. DOI: 10.1080/08993400500224419. URL: http://www.tandfonline.com/doi/abs/10.1080/08993400500224419.

[136] Arto Vihavainen. "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*. ICALT '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 498–499. ISBN: 978-0-7695-5009-1. DOI: 10.1109/ICALT.2013.161. URL: http://dx.doi.org/10.1109/ICALT.2013.161.

[137] Common Vulnerabilities and Exposures Database. *CVE-2014-0160: Heart-bleed*. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160. 2014.

[138] Christopher Watson, Frederick W. B. Li, and Jamie L. Godwin. "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*. ICALT '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 319–323. ISBN: 978-0-7695-5009-1. DOI: 10.1109/ICALT.2013.99. URL: http://dx.doi.org/10.1109/ICALT.2013.99.

[139] Christopher Watson and Frederick W.B. Li. "Failure Rates in Introductory Programming Revisited". In: *Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education*. ITiCSE '14. Uppsala, Sweden: ACM, 2014, pp. 39–44. ISBN: 978-1-4503-2833-3. DOI: 10.1145/2591708.2591749. URL: http://doi.acm.org/10.1145/2591708.2591749.

[140] Christopher Watson and Frederick Li. "Failure rates in introductory programming revisited". In: *Proceedings of the 2014 conference on Innovation & technology in computer science education* (2014), pp. 39–44. DOI: 10.1145/2591708.2591749.

[141] L. Werth. "Predicting Student Performance in a Beginning Computer Science Class". In: *SIGCSE '86 Proceedings of the Seventeenth SIGCSE Technical Symposium on Computer Science Education* (1986), pp. 138–143. ISSN: 00978418. DOI: 10.1145/953055.5701.

[142] Susan Wiedenbeck. "Antecedents to End Users' Success in Learning to Program in an Introductory Programming Course". In: *Proceedings - IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2007* (2007), pp. 163–170. DOI: 10.1109/VLHCC.2007.8.

[143] D. M. Wiig. "A Bayesian Probability Approach to Predicting Student Performance in Introductory Computer Science Courses". In: *SIGSMALL/PC Notes* 15.1 (Feb. 1989), pp. 3–19. ISSN: 0893-2875. DOI: 10.1145/66099.66482. URL: http://doi.acm.org/10.1145/66099.66482.

[144] Stanley Wileman, John Konvalina, and Larry J. Stephens. "Factors Influencing Success in Beginning Computer Science Courses". In: *The Journal of Educational Research* 74.4 (1981), pp. 223–226. DOI: 10.1080/00220671.1981.10885313. URL: https://doi.org/10.1080/00220671.1981.10885313.

[145] Brenda C. Wilson. "Contributing factors to success in computer science: A study of gender differences". PhD thesis. 2000, p. 95. ISBN: 9780599886438. URL: https://search-proquest-com.jproxy.nuim.ie/docview/304644388?accountid=12309.

[146] Brenda Cantwell Wilson and Sharon Shrock. "Contributing to Success in an Introductory Computer Science Course: A Study of Twelve Factors". In: *Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '01. Charlotte, North Carolina, USA: ACM, 2001, pp. 184–188. ISBN: 1-58113-329-4. DOI: 10.1145/364447.364581. URL: http://doi.acm.org/10.1145/364447.364581.

[147] Brenda Cantwell Wilson and Sharon Shrock. "Contributing to Success in an Introductory Computer Science Course: A Study of Twelve Factors". In: *SIGCSE Bull.* 33.1 (Feb. 2001), pp. 184–

188. ISSN: 0097-8418. DOI: 10.1145/366413.364581. URL: http://doi.acm.org/10.1145/366413.364581.

[148] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques.* Amsterdam: Morgan Kaufmann, 2016. ISBN: 9780123748560.

[149] Sung-Whan Woo, HyunChul Joh, Omar H Alhazmi, and Yashwant K Malaiya. "Modeling vulnerability discovery process in Apache and IIS HTTP servers". In: *Computers & Security* 30.1 (2011), pp. 50–62. DOI: 10.1016/j.cose.2010.10.007. URL: https://doi.org/10.1016/j.cose.2010.10.007.