Learning Behaviours data in Programming Education: Community Analysis and Outcome Prediction with cleaned data

Tai Tan Mai[a,*], Marija Bezbradica[a] and Martin Crane[a]

[a]School of Computing, Dublin City University

ARTICLE INFO

Keywords:
community detection
learning analytics
random matrix theory
machine learning
educational data mining

ABSTRACT

Due to the COVID19 pandemic, more higher-level education programmes have moved to online channels, raising issues in monitoring students' learning progress. Thanks to advances in online learning systems, however, student data can be automatically collected and used for the investigation and prediction of the students' learning performance. In this article, we present a novel approach to analyse students' learning behaviour, as well as the relationship between these behaviours and learning assessment results, in the context of programming education. A bespoke method has been built based on a combination of Random Matrix Theory, a Community Detection algorithm and statistical hypothesis tests. The datasets contain fine-grained information about students' learning behaviours in two programming courses over two academic years with about 400 first-year students in a Medium-sized Metropolitan University in Dublin. The proposed method is a novel approach to data preprocessing which can improve the analysis and prediction based on learning behavioural datasets. The proposed approach deals with the issues of noise and trend effect in the data and has shown its success in detecting groups of students who have similar learning behaviours and outcomes. The higher performing groups have been found to be more active in practical-related activities throughout the course. Conversely, we found that the lower performing groups engage more with lecture notes instead of doing programming tasks. The learning behaviours data can also be used to predict students' outcomes (i.e. Pass or Fail the exams) at the early stages of the study, using popular machine learning classification techniques.

1. Introduction

Education in Computer Programming and related domains has received increasing attention due to the growth in demand for Information Technology (IT)-related job markets. Furthermore, in recent years, STEM fields (science, technology, engineering and mathematics) also require essential IT skills and knowledge, making these types of skills an integral part of most STEM sub-disciplines such as Artificial Intelligence, Bio-informatics, Statistics etc. One of the pivotal and essential courses in any IT-related degree is a set of programming courses so understanding and improving students engagement and process of learning are of key importance. However, despite the necessity of these skills, there have been considerable drop-out rates in introductory programming courses reported from many studies [6]. The failure rate in introductory programming modules has been reported to be 28% on average, with a huge variation from 0% to 91% [6], according to a recent study using data from 161 universities around the world.

Typically, lecturers can monitor student progress and, if necessary, conduct interventions to ensure the learning quality and progress of students in the class. In addition to practical exercises and formal assessments, interaction in conventional face-to-face classes could help educators to understand the performance of students. However, online courses may restrict the potential for direct communication between educators and students [43]. These difficulties may create more challenges for lecturers to monitor how the students are performing during the courses. In this context, more higher-level education programmes have moved to online channels due to the pandemic, causing the lack of direct communication. Hence, it is necessary to develop novel methods to support educators in monitoring and understanding students' learning behaviour during their online sessions.

Thanks to developments in educational technology, advanced online learning systems, such as Moodle and Blackboard, enable one to capture learning data generated by participants of the courses [58]. These systems provide the ability to automatically record a large amount of interaction data at fine-grained levels, e.g. at the level of mouse and keyboard events on a page. Such log data has the potential to be used to improve the pedagogical value of online teaching and learning [53]. Analysing the massive amount of educational data collected during the learning process also has the potential to help instructors and students to obtain a comprehensive view of a student's learning progress. This insight enables the possibility of evidence-based interventions and recommendations [36] which might have effects on learner perception, learning patterns and learning outcomes [25].

In terms of automatic behavioural log data, there is the potential for the effect of *noise* and *trend* to be present in the automatically collected data. Students can work flexibly when completing their learning paths in the online learning system. For example, they can carry out various learning activities such as reading lecture notes, coding, navigating among course documents in any order, resulting in noise in the logged data, i.e. data heterogeneity and complexity [32]. In addition, we have noticed from the data gathered that students are likely given the same instructions and learn-

*Corresponding author
tai.mai2@mail.dcu.ie (T.T. Mai)
ORCID(s):

ing pathway in the same class. As a consequence, this may create a *trend* effect, i.e. students' learning behaviours can be similar and highly positively correlated with other learners' behaviours in the same course. Hence, it is important to filter noise and clean the trend effect in the event log data before applying further analysis.

This research aims to investigate the relationship between students' learning behaviours on course material items and their performance in the exams while taking programming-related courses in online learning systems. Specifically, the research objective is to answer these research questions:

- RQ1. Do students from different groups, corresponding to different patterns of learning behaviours, perform differently in the exams? If this is the case, how do such groups interactions with items of course material differ?
- RQ2. Is there potential for students' learning behavioural data to be used to predict learning outcomes (Pass or Fail) at the early stages of the study period?

To address the research questions, we investigate about 400 university students participating over the two programming-related courses during the two academic years (2017/2018 and 2018/2019). The courses have been delivered to students in a combination of conventional and online formats. In particular, students have physically attended the lecture sessions in lecture halls, and have conducted all learning activities on a bespoke online system. The learning data were logged and these serve as input datasets for further analysis here. Behavioural data captured automatically from the system is stored in the format of an *event log*. From the input event logs, the concept of a *student-event item data matrix* and a *transition-student data matrix* (described below) have been developed to represent the students' learning behaviour. To deal with the problem of noise and trend effect in the datasets, we utilise the cleaning methods based on Random Matrix Theory (RMT), followed by the construction of Minimum Spanning Trees (MST) to reflect the difference in learning behaviours of all students. Community detection algorithms and statistical tests have also been applied to investigate the students' behaviours on course material items. For the prediction of learning outcome, a set of machine learning algorithms have been applied into every week's original and cleaned data and the predictability has been validated by cross-validation technique.

The rest of the paper is organised as follows: Section 2 discusses the related works; Section 3 describes the context of the study, data and methods; Section 4 provides detail of the experimental results; Section 5 discuss the implications and limitations, followed by the conclusion in Section 6.

2. Related work

2.1. Analysis using Learning Behavioural data

Much research has been carried out to determine the relationship between the learning behaviours and performance of students [16, 17]. In [37], the authors investigated a variety of learning activities such as collaborative activities and giving feedback by using data from 13 participants in an experimental setting class. The effect of the diversity of learning styles on learning scores and satisfaction has also been tested in [57], using the data from an online forum and survey data from 144 students. Although these efforts consider a wide range of learning activities, they have been carried out with small size samples so survey data was still required for the analysis. In the context of this paper, we utilised datasets from a large number of students in two modules over the two academic years, i.e. 112 students in Course#1 2018, 151 students in Course#1 2019, 62 students in Course#2 2018 and 48 students in Course#2 2019. The datasets were automatically collected during the study from our bespoke online learning platform.

Complementing the work above, the analysis of massive learning behavioural log data has been supported with the emergence of Educational Process Mining (EPM), and the application of process mining [62] techniques and algorithms in educational event log data [52]. Recently, EPM appears to be an effective tool to analyse educational data and deliver new insights into the learning and teaching processes. Many of the process mining applications in education have been developed and implemented in various aspects of education [50, 24]. The applications of Massive Open Online Courses (MOOCs) attract the most attention from researchers due to the availability of the input log data [32]. The majority of applications in EPM aim to discover learning patterns from the input data so-called *event log*, resulting in learning process models. However, there may be challenges for the process discovery approach when there are many complicated and noisy event logs. In such cases, the process mining techniques would likely produce 'spaghetti-like' process models which can be incomprehensible. It is also not trivial to combine many process models for the interpretation [15]. This research adopts the notion of 'event-log' in EPM as a storage format of the collected dataset. However, instead of generating complex process models, we extract features from the logs and then propose a method to clean the extracted dataset. We suggest that this cleaning method can separate the information part from the noise in the dataset, and, in the process, improve the performance of community analysis, i.e. to produce more logical coherent communities in terms of their learning performance, as well as to generate better predictive models of learning outcomes.

In terms of the network-based approach in Education, there are a few existing works using Community Analysis and Minimum Spanning Tree, which are most similar to this paper. Both [29] and [51] studied the network structure of undergraduate courses and their contributions to students' learning pathways. However, the authors merely considered the courses' grades from a relatively small number of students. We not only utilise exam results but also consider student learning behaviours by extracting behavioural data features from large automatic collected datasets.

Regarding programming educational context, in [5], the

author found that *practice* is essential for improving students' programming skills and students should be given opportunities to practice and receive constructive feedback. In [9], the author developed metrics for use as formative assessment tools to analyse (successful and unsuccessful) students' learning patterns. However, these approaches have focused mostly on practical activities such as coding and solving programming tasks. Our approach considers comprehensive learning activities that students do in their programming study, i.e. coding, reading lecture notes and labsheets.

2.2. Learning Outcome Prediction

Prediction of student performance has been one of the most popular topics in Learning Analytics in recent years [44, 53]. In general, input data used for the prediction can be classified into two categories: *static* and *dynamic* data [25]. Student demographic information and historical educational records can be classified as static data because these variables and values do not update or change frequently over the study period. On the other hand, online behaviours, textual data and other multimodal data can be considered as dynamic data because they can be continuously generated when students are interacting with the system. In terms of the prediction of learning outcome from the static data, one can rely on features such as personal attributes and cumulative grade point average (CGPA) from previous years [3]. For instance, using learning grades from previous courses can predict the drop-out probabilities of computing students [27]. However, the use of static data to predict learning outcomes has been shown to cause some problems [25], i.e. the student's actual efforts during the learning process can be ignored. It has also been found that previous student results (e.g. the CGPA) are not sufficient to predict drop-outs, and engagement variables also need to be included (e.g., number of accesses to the platform) to achieve good accuracy results [1]. The static data can also be difficult to collect as they may need to be merged from various data sources, which might cause data quality and ethical issues.

On the other hand, dynamic features such as behavioural data can be collected easily due to their availability in the advanced learning platform. It has been shown that it is possible to rely on dynamic data [22] to predict student learning outcomes. For example, learning log data from the Moodle platform has been used for predicting learners' performance [31], using common features in a Learning Management System such as Assignment, Feedback, Course login and Chat. More fine-grained data can be used as predictors such as mouse interactions (e.g. click and drag) [64]. Multimodal features (e.g. eye-tracking, face-video and wristband) have also been demonstrated its predictability of learning performance [56]. Generally, according to the recent surveys, most of the current approaches have a focus on either combining new features collected from learning platforms or new strategies with different machine learning predictive models [44, 25]. However, to the best of our knowledge, none of the existing research directly deals with the issues of *noise* and *trend* in educational data, which, we feel, may have a negative influence on prediction models. One of the most common methods to pre-process data is Principal Component Analysis (PCA) which has been applied in different areas such as education [67], medical [30] and network security [8]. Although PCA supports the selection of the most relevant features, which may help to unintentionally eliminate noise in the data, the trend effect remains. The evaluation of the *predictability* of the behavioural data at early stages also remains limited [25]. Based on Random Matrix Theory, our approach aims to identify and separate the key information part from the noise, which enhances the performance of the prediction models with cleaned datasets in comparison with original and PCA-based processed datasets.

3. Research methodology

3.1. Context of the study

This research has been carried out based on four datasets representing the learning behaviour of students and their performance in two programming-related courses on a Computer Science (CS) program in a Medium-sized Metropolitan University. The first course is a first-year introductory programming module that is delivered to Software Engineering students. These students generally have the aim of targeting programming-related jobs such as software development. The second course is a programming module taken by first-year Business Computing students who are usually looking for non-technological positions in an IT-related field. We denote the two courses as Course#1 and Course#2, respectively.

In both courses, learning material items are provided to the students on a weekly basis. Course items include general course information, lecture notes, labsheets and programming tasks. Students should read lecture notes during a lecturing session. In a lab session, students should follow instructions and examples in labsheets and do given programming tasks. The solutions to the tasks are uploaded and tested automatically by the system. Course items are delivered in the form of web pages on the bespoke online learning system. We formalise the course material items in this context as material type (i.e. *General, Lecture, Labsheet and Practice*) combined with the corresponding week, e.g. *Labsheet_1* means the labsheet used in week 1. For the general documents, we denote them as *General*. Students' interactions with the items (e.g. mouse clicking or scrolling, highlight a piece of text or switching between two items) are logged automatically on the database.

There are three lab exams in Course#1, which take place in weeks 4, 8 and 12 (the final week of the semester) while Course#2 students have to take two lab exams in weeks 6 and 12. On finishing a programming lab examination task, students submit their codes to the systems and get the results as "correct" of "incorrect" submission. A submission is considered "correct" if it passes all the test cases which are pre-defined by the instructors. Each task is given the same mark proportion and the overall mark is given to students after the exam is finished. A student whose grade is fewer

than 40 out of 100 is labelled as "lower-performing', otherwise, that student is considered as "higher-performing". In this research, the overall marks of students have been used for the behavioural analysis while the labelling is used as a target variable, i.e. "higher-performing = 1" and "lower-performing = 0" for the evaluation of the predictability of behavioural data to the students' learning outcomes.

All lab exams are mandatory and carry the same weight in the overall assessment. Students are, therefore, required to take the exams seriously by doing all given programming tasks as much as they can. The last exam in each module is the most challenging one, requiring a comprehensive understanding of the course knowledge to solve the given problems. Therefore, the results of the last exam will be used as a basis for further analysis in this research.

The two courses are expected to provide students with fundamental knowledge and skills in Python programming. Both modules are mandatory and key to the aims of the overall outcomes. Because they are prerequisites for the programmes, students are expect to be equally motivated as they cannot follow curriculum without deep understanding of these modules. Hence, we assume that students are likely to take these modules seriously and participate fully to maximise their learning benefits.

Table 1
Datasets information.

Dataset	Number of students	Number of events	Average events per student
Course#1-2018	112	1,054,394	9414
Course#1-2019	151	1,484,297	9829
Course#2-2018	62	211,855	3417
Course#2-2019	48	200,006	4166

The difference between the two modules relates to the level of knowledge and task requirements. In Course#1, students are taught more advanced concepts in programming and given more challenging exercises, compared to students in Course#2, as that relates to their specific programmes. As a result, students in Course#1 generally have more activities in learning than Course#2 students, as can be seen in Table 1. In other words, while Course#1 can be seen as a typical programming course for CS students, Course#2 represents a programming course for non-IT learners who still need programming skills at a certain level. It is important to note that both courses have the same coordinator and the curriculum had not significantly changed over the two academic years. Therefore, the learning motivation of students is expected to be the same in both courses. However, their behaviours can be distinct due to the differences in the level of course requirements. As a result, these datasets can reflect the diversity of learning characteristics of students' learning behaviours, giving a good quality of data.

3.2. Event logs for learning behaviours

We can consider a real-life scenario of student learning programming on the online learning system as follows. On a day in Week 5, student $s1$ read a labsheet for a task instruction. While reading a labsheet, the student also switched between lecture notes and the labsheet two times, and another two mouse events on the lecture page were logged. Then the student can write code to solve a given task and upload it to the system via the submission portal. All these learning events of the student $s1$ can be recorded and stored as event data structure so-called *event log* which can be seen in Table 2 as an example.

Table 2
Example of event log of student $s1$ on two days in week 5.

Event Item	Timestamps	Student id	
1	Labsheet 5	2018-08-12 14:30:00	s1
1	Labsheet 5	2018-08-12 14:35:00	s1
1	Lecture 5	2018-08-12 14:36:00	s1
1	Labsheet 5	2018-08-12 14:45:00	s1
1	Lecture 5	2018-08-12 14:49:00	s1
1	Labsheet 5	2018-08-12 14:50:00	s1
2	Practice 5	2018-08-13 11:59:00	s1
2	...	...	s1

We adopt the format of *event log* in *Process Mining* [62] to store the students' learning behaviour. An event log includes a collection of events implemented in chronological order. Each event belongs to a learning trace which refers to the sequence of events of a student within a day. Event logs may contain other attributes such as timestamps, participants and results. In the context of this research, a student's learning event log comprises the following information:

- *Trace id*: A trace refers to a sequence of learning events of a student in a day. For example, Table 2 illustrates two learning traces associating with 12 August and 13 August 2018 of the student $s1$.
- *Event Item*: An event item refers to an item of course material of the corresponding week where students' interactions with the system are logged.
- *Timestamps*: Timestamps refer to the date and time when the corresponding event occurred. The timestamp is essential information as it will be used for ordering events and reflecting the behaviour of students.
- *Student id*: *Student id* refers to the identity of students.

3.3. Learning Behavioural Features

We use two types of features to reflect the students' learning behaviour in this research. The first type is the *event item features*, i.e. the number of events that occurred in each course material item. *Event frequency features* extracted from the event log can be arranged as a *student-event item data matrix* where each column refers to the number of events on a material item generated by students and each row is the data for each student. An example of a *student-event item data matrix* can be seen in Table 3.

The second type is the *transition frequency features*, i.e. the number of occurrences that a student moves from an

Table 3
Example of student-event item data matrix

StudentId	Lecture1	Labsheet1	Practice2	...
s1	5	7	6	...
s2	24	14	34	...
s3	12	54	0	...
...	...	...	...	...

event on a course item to another event. Please note that the two events can be on the same item or two different items. We use the term *transition* to denote this phenomenon of moving between consecutive events. The *transition frequency features* can be arranged as a *transition-student data matrix* where rows refer to *transition frequency features* and columns are the data for students. An example of a transition data matrix of an event log can be seen in Table 4. The value of *Lecture1-Labsheet1* for student $s2$ equal to 14 indicates that student $s2$ performed an event 14 times on the *Lecture1* directly before the next event on *Labsheet1*. Please note that if the two materials are the same, e.g. *Lecture1-Lecture1*, the transition reflects a loop in the learning process, i.e. the student keeps working on the same course item *Lecture1*.

In this paper, the *student-event item data matrix* has been used for the learning outcome predictions and the *transition-student data matrix* has been used for analysing the relationship between students' learning behaviour and their assessment performances.

Table 4
Example of transition-student data matrix

Transition	s1	s2	s3	s4	...
Lecture1-Lecture1	4	5	10	23	...
Lecture1-Labsheet1	0	14	9	12	...
Labsheet1-Practice1	12	6	0	21	...
Labsheet1-Lecture1	16	25	0	5	...
...	...	...	...	...	...

3.4. PCA and Random Matrix Theory for Behavioural Features

Given a $m \times n$ data matrix $\mathbf{G}$ extracted from an event log, we can normalise the matrix $\mathbf{G}$ as $\mathbf{G}(n)$ as follow:

$$\mathbf{G}(n)_j = \frac{\mathbf{G}_j - \overline{\mathbf{G}_j}}{\sigma_j} \tag{1}$$

where $\mathbf{G}(n)_j$ is a column of the matrix $\mathbf{G}(n)$; $\mathbf{G}_j$ is a column of matrix $\mathbf{G}$. In case $\mathbf{G}$ is *a transition-student data matrix*, $\mathbf{G}_j$ denote a series of the frequency of transitions a students j. On the other hand, if $\mathbf{G}$ is *a student-event item data matrix*, $\mathbf{G}_j$ denote a series of the frequency of accesses to the corresponding learning item j from all students. $\overline{G_j}$ is the mean value of G_j and σ_j is a standard deviation of $\mathbf{G}_j$. In other words, $\mathbf{G}_j$ and $\mathbf{G}(n)_j$ reflect the learning behaviour of student j. The correlation matrix $\mathbf{C}$ can be expressed in terms of the inner product of $\mathbf{G}(n)_i$ and $\mathbf{G}(n)_j$ as follows: $\mathbf{C}_{ij} = \left\langle \mathbf{G}(n)_i, \mathbf{G}(n)_j \right\rangle$. We note that $\mathbf{C}_{ij} \in [-1; 1]$. It may be noticed that the correlation $\mathbf{C}_{ij}$ can reflect how similarly two students i and j interacted with course material items. If $\mathbf{C}_{ij} > 0$, the transitions of the two students i and j increased together and the students behaved similarly in the course. Conversely, if $\mathbf{C}_{ij} < 0$, the two students tend to behave differently on the learning system.

The characteristic equation of $\mathbf{C}$ can be shown to be given by: $\mathbf{C}V = \Lambda V$ where Λ is a n x n diagonal matrix of eigenvalues λ_i and V is a matrix whose columns refers to the corresponding eigenvectors v_i of $\mathbf{C}$.

Based on PCA theory, the new n variables x_i, forming a new data matrix $\mathbf{X} = [x_1, x_2, ..., x_n]$ can be obtained after Principal Component Analysis of $\mathbf{G}(n)$ as follows:

$$x_i = v_{i1}G(n)_1 + v_{i2}G(n)_2 + ... + v_{in}G(n)_n = v_i^T G(n) \tag{2}$$

where $1 \leq i \leq n$, x_i refers to the scores and v_i refers to the loadings of the principal component i. In other words, each principal component has its own eigenvalue and eigenvector.

We can also reconstruct the original normalised data $\mathbf{G}(n)$ from $\mathbf{X}$ as follows:

$$\mathbf{G}(n) = \sum_{i=1}^{n} v_i x_i \tag{3}$$

In addition, given a random matrix $\mathbf{A}$ where $\mathbf{A}$ is a $m \times n$ matrix with randomly distributed elements with zero mean and unit variance. It has been shown that [66] the properties of $\mathbf{C}$ can be compared to the correlation matrix $\mathbf{R}$ of the random matrix $\mathbf{A}$ as $\mathbf{R} = \frac{1}{m}\mathbf{A}\mathbf{A}^T$ According to RMT, the statistical properties of such a matrix $\mathbf{R}$ are known [26]. In particular, when the sample size $m \to \infty$ and the number of features $n \to \infty$, providing that $Q = \frac{m}{n} \geq 1$ is fixed, the distribution of eigenvalues λ of the random matrix R is given by the Marcenko-Pastur probability density function [49]:

$$P_R(\lambda) = \frac{Q}{2\pi\sigma^2} \frac{\sqrt{(\lambda_+ - \lambda)(\lambda - \lambda_-)}}{\lambda} \tag{4}$$

where $\lambda_- \leq \lambda \leq \lambda_+$, λ_- and λ_+ are lower and upper limits, eigenvalues of R respectively, given by:

$$\lambda_\pm = \sigma^2\left(1 \pm \sqrt{\frac{1}{Q}}\right)^2 \tag{5}$$

where $\sigma = 1$ due to $\mathbf{A}$ having an unit variance.

We note that $\lambda_\pm$ are the upper/lower limits of theoretical eigenvalues distribution. Eigenvalues falling outside of this range are assumed to deviate from the expected values of the Random Matrix Theory [41]. As a result, by comparing this theoretical distribution with the empirical data, we can identify the key eigenvalues containing specific information in the data. This characteristic of the RMT supports the need to clean the effect of noise and trend in the data [49].

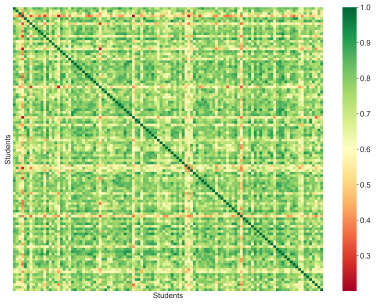

Figure 1: Uncleaned correlation matrix of students' transitions in Course#1 dataset

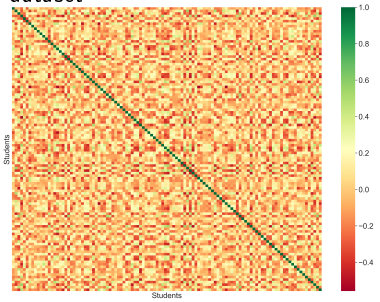

Figure 2: Cleaned correlation matrix of students' transitions in Course#1 dataset

3.5. Noise and trend effect cleaning

We have noticed that, in practical usage of the online learning system, students may interact flexibly with course material items. Although the students can be given the same instructions and learning pathway, they are free to use learning functions in their own way. This phenomenon appears to create noise in the event log data. On the other hand, as all students attended the same lectures, the learning instructions given to them are the same. As a consequence, all students may interact similarly with course material items. We can observe this trend effect in Figure 1. Most transitions among students are highly correlated. This issue may limit the chance of detecting the difference in learning behaviours among groups of students. Therefore, it is necessary to clean the effect of noise and trend in the dataset [49].

We also observe a phenomenon that students' learning behaviours can be affected by a trend factor, i.e. they were asked to follow the same instructions and learning pathway in the class, causing highly positively correlated learning behaviours among the students. (Figure 1). By removing such a trend component in a classroom, the remaining components of the correlation could explain better the characteristics of the students' learning behaviours. In this paper, we adopt, from financial references such as [39, 48], the concept of a "Market Component". This is the largest eigenvalue of a correlation matrix representing a cross-market effect affecting all stocks. Similarly, the trend effect in a classroom can be reflected by the largest eigenvalue of the correlation matrix of students' learning behaviours.

In the following sub-sections, we discuss the methods to clean the correlation matrix of a dataset as well as propose a method to clean the dataset based on Random Matrix Theory.

3.5.1. Cleaning the correlation matrix

Having reviewed a number of correlation cleaning methods (e.g *eigenvalue clipping* [48, 41] and *linear shrinkage* [35]), we utilise the *eigenvalue clipping* because it was found to be the best in terms of its ability of removing the noise while preserving the information part, i.e. the trace of the original correlation matrix, by simply utilising the results of the Marcenko-Pastur equation [13] instead of choosing a parameter during the cleaning process such as *linear shrinkage* and *Rotationally invariant, optimal shrinkage* [14]. The *eigenvalue clipping* provides robust out-of-sample performance [11] and has also been widely adopted [20, 55].

Let $\lambda_1, ..., \lambda_N$ be the set of all eigenvalues of $\mathbf{C}$ and $\lambda_1 > ... > \lambda_N$, and i be the position of the eigenvalue such that $\lambda_i > \lambda_+ and \lambda_{i+1} \leqslant \lambda_+$.

Then we set

$$\lambda_j = 1/(N-i) \sum_{k=i+1}^{N} \lambda_k, \tag{6}$$

where $j = i+1, ..., N$. In other words, we keep all the upper bound eigenvalues, i.e. those with information, and replace all lower bound eigenvalues, i.e. those within bounds predicted by RMT, with the average value of them. Hence, this method can best preserve the trace of the original correlation matrix. The new set of eigenvalues can be used to construct a denoised eigenvalue and spectrum associated correlation matrix $C_{denoised}$. [49].

The effect of the first eigenvalue and eigenvector can be removed from the denoised correlation matrix as follows [49], forming a cleaned correlation matrix:

$$C_{cleaned} = C_{denoised} - W_1 V_1 W_1^T \tag{7}$$

where W_1 and V_1 are the first eigenvector and eigenvalue of C. An example of the effect of the cleaned correlation matrix can be seen in Figure 1 and Figure 2. The two figures illustrate the correlation coefficients of the transition-student data matrix in the Course#1-2018 dataset, i.e. each dot in the figures refers to the correlation of one student to another student. The scale on the right side of the two figures indicates the range value of the correlation coefficients. We can notice that the dots in the diagonal refer to the correlation of transition data of a student to her/himself (i.e. correlation values = 1). It may be seen in Figure 1 that the majority of the dots are in different shades of green. This phenomenon may reflect a "trend effect", i.e. students' learning behaviours can be similar and highly positively correlated with other learners' behaviours in the same class. These issues may negatively influence the construction of prediction models though. After cleaning the data, there are more neutral and dots shades of orange visible in Figure 2, indicating the negative correlation values. That is to say, the $C_{cleaned}$ correlation matrix may contain differences in student learning behaviours, creating more chances to better cluster the students. Similar results are observed in the other three datasets (Course#1-2019, Course#2-2018 and Course#2-2019). The use of cleaned correlation matrix in Community Analysis is discussed in Section 3.6.

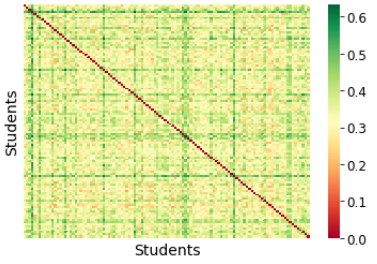

Figure 3: Learning Behavioural Distance matrix of students in Course#1 dataset

3.5.2. Cleaning the Dataset

While the cleaned correlation matrix is expected to be useful in community analysis, the prediction of learning outcomes, however, require a tabular dataset. As a result, it is necessary to clean the original data matrix instead of the correlation matrix in case of improving prediction models. In this section, we propose a method to clean the original data matrix based on Random Matrix Theory.

In terms of the eigenspectrum of the correlation matrix, let $\lambda_1, ..., \lambda_N$ be the set of all eigenvalues of $\mathbf{C}$ and $\lambda_1 \geq ... \geq \lambda_N$, and k be the position of the eigenvalue such that $\lambda_k > \lambda_+$ and $\lambda_{k+1} < \lambda_+$. We note that λ_1 refers to the largest eigenvalues and the first principal component. The clean dataset $\widehat{G}$ can be constructed as follows:

$$\widehat{G} = \sum_{i=1}^{n} v_i x_i - \alpha \sum_{i=k}^{n} v_i x_i - \beta v_1 x_1 \tag{8}$$

where $\sum_{i=k}^{n} v_i x_i$ refers to the noisy part of the dataset based on RMT, and $v_1 x_i$ refers to the first principal component part of data. The parameters $\alpha \in [0, 1]$ and $\beta \in [0, 1]$ control how much we want to remove the noise and trend part from the original data, respectively. If α and β is equal to zero, Equation 8 is similar to Equation 3, i.e. the reconstruction of full original data from the principal component scores dataset. If α and β have a unit value, we have a *fully cleaned dataset*. Otherwise, we have a *partly cleaned dataset* when both α and β are between 0 and 1. The cleaned dataset $\widehat{G}$ can then be used as input for machine learning predictive models. We expect that the performance of the predicting models using $\widehat{G}$, either fully or partly cleaning, will be improved in comparison with the use of the original dataset and simple PCA-based datasets. Results are shown in Section 4.2.1.

3.5.3. Distance matrix of students learning bahaviours

Although the correlation values appear to be useful in reflecting the similarity and difference in students' learning behaviours, they are not appropriate metrics as they do not satisfy non-negativity and triangle inequality conditions [49]. For example, the difference between the correlations (0.8, 1.0) is the same as (0.1, 0.3), but the former tuple illustrates a higher difference regarding co-dependence. Fortunately, it is possible to translate the correlation matrix into a distance matrix $\mathbf{D}$ as follows: [49]

$$\mathbf{D}_{ij} = \sqrt{0.5 * (1 - \mathbf{C}_{ij})} \tag{9}$$

with $D_{ij} \in [0, 1]$ where D_{ij} is a distance value of learning behaviours between the two students i and j. The value closer to 1 refers to where two students interact completely differently with the course material items while the value closer to 0 indicates that two students behave similarly. Figure 3 indicates an example of the learning behavioural distance matrix of students in Course#1-2018 dataset. The diagonal comprises zero values, illustrating the behavioural distance between students and themselves. The other distance values range from 0 to more than 0.6. The distance matrix can be used to construct a community graph which is discussed in the next section.

3.6. Community Analysis

To verify whether students with similar behaviours perform similarly and vice-versa in lab exams, we choose to adopt a network-based approach. Generally, a graph is constructed based on the concept of a distance matrix. In the graph, each node represents a student and the edge weight between two nodes indicate the distance between learning behaviours of the two students. Then, a clustering technique can be applied to the graph to detect the communities where students having similar learning behaviours are grouped.

3.6.1. Construct graph from distance matrix

A graph can be constructed directly using the distance matrix values D_{ij} as edge weights. Unfortunately, such a network is a hardly readable weighted complete graph as each node (student) has a connection to all other nodes in the graph. Additionally, we note that the time complexity of community detection algorithms is proportional to the number of edges and nodes in the graph. For example, the time complexity of the Girvan-Newman algorithm [34] is $O(m^2 n)$ where m is the number of edges and n is the number of nodes (or students). With such a fully connected graph, the number of edges is $m = n(n-1)/2$, which may lead to, in the worst case, the time complexity of the algorithm of $O(n^5)$. To overcome this issue, one possible solution is to reduce the number of edges in such a fully-connected graph. It is important to minimise the number of edges in the constructed graph while preserving the purpose of grouping students having similar behaviours.

In the context of this paper, all values of the distance matrix of each dataset are identical. In other words, all edge weights of the fully connected graph constructed from the corresponding distance matrix are unique. Taking advantage of this characteristic, we adopt the notion of Minimum Spanning Tree (MST) [65], i.e. an MST is constructed for each graph and connects all students in a course without having any loops. With the distance matrix D as the adjacency matrix of a graph, an associated MST is constructed such that the sum of all edges in the graph is minimal for all possible spanning trees. We note that if all edge weights of a graph are unique, then the graph has only one corresponding MST. Hence, in our case, each course dataset can be used to produce a single associating MST. It can be seen that the MST of a set of n students is a graph with $n-1$ edges, reducing the time complexity of the Girvan-Newman algorithm to

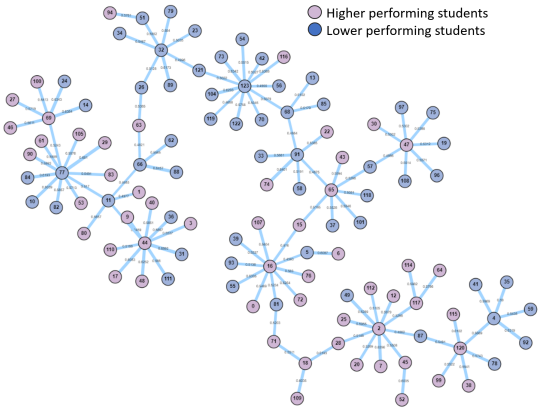

Figure 4: An example of an MST constructed from a distance matrix of the Course#1-2018. Purple nodes refer to the higher performing students while Blue nodes refer to the lower performing students. Each edge corresponds to the distance of learning behaviours between two students, as per Eq.9.

$O((n-1)^2 n) = O(n^3)$. Furthermore, in the MST of a whole course, each student can be connected to one or more other students who have the most similar learning behaviours with that student. Therefore, the clustering purpose is preserved.

3.6.2. Community detection on MST graph

Based on the MST constructed from the distance matrix, it is possible to advance to the further step which is community detection which is supported by several methods [10, 34]. In this research, we utilise the popular detection algorithm from Girvan-Newman [34] which is applied in various domains such as biology [34], finance and cryptocurrencies [18]. The algorithm aims to divide the whole network into smaller communities or groups by progressively removing edges with the highest *edge betweenness* until no edges are remaining. *Betweenness* is the number of the shortest paths between pairs of nodes that run through it from the original network [34]. Please note that we take the weight of edges into account when calculating the edge betweenness. The nodes, i.e. students in a smaller group, are highly connected to each other than the ones outside the group. Figure 4 illustrates an example of the MST constructed from the data for Course#1-2018 in week 12.

The detected groups can be used for further investigation regarding their performance in lab exams. In particular, we can use statistical tests to verify if the lab exam grades are significantly different between the communities. As it is not guaranteed that the data of students in each community will be normally distributed, a non-parametric test is preferred in this case, i.e. Mann-Whitney U Test [46] has been utilised. It is also possible to verify if the two communities interacted differently with each course material item in the system. Further investigations are discussed in Section 4.

3.6.3. Selecting the number of detected communities

We observe that the Girvan-Newman algorithm can be seen as a hierarchical method, i.e. it constructs a dendrogram that shows the hierarchical clustering structure. The number of detected communities can, therefore, range from 1 to the number of nodes in the graph where each community contains only one node. When using Girvan-Newman, it is necessary to determine criteria to decide the cut-off level in the dendrogram to create the resulting communities.

In this research, we define the concept of *mixed community rate*. Let $C = (c_1, c_2, ..., c_n)$ be a community structure. Let $c_i = (h_i, l_i, n_i)$ be a detected community where h_i is the number of higher performing students, l_i be the number of lower performing students in the community c_i. The label n_i of the community c_i is identified as Equation 10 below:

$$n_i = \begin{cases} \text{higher-performing}, & \text{if } h_i/(h_i + l_i) \geq k \\ \text{lower-performing}, & \text{if } l_i/(h_i + l_i) \geq k \\ \text{mixed}, & \text{otherwise} \end{cases} \quad (10)$$

The parameter k can be configured, depending on analysis purposes. In the ideal case of $k = 1$, a community will only be labelled as higher or lower performing if it contains only higher or lower performing students. However, we expect the similarity in learning behaviours between students in practice and it could be difficult to detect such a homogeneous community. Instead, we set $k = 0.7$, i.e. a community is labelled "higher-performing" if there are greater than or equal to 70% of higher performing students in the community and similarly for "lower-performing" communities. Otherwise, the communities are labelled as "mixed". The *mixed community rate* of a community structure can be computed as follows:

$$\textit{mixed community rate} = \frac{\textit{No. of mixed communities}}{\textit{total no. of communities}} \quad (11)$$

The higher/lower performing communities may include key features about student success while mixed communities may contain less information. As a result, we expect a good community structure containing fewer *mixed communities*. Based on the *mixed community rate* indicator, it is possible to investigate each possible community in the resulting dendrogram from the Girvan-Newman algorithm and identify the number of detected communities by considering their *mixed community rates*. We also make a comparison between the original dataset and the cleaned dataset in terms of the community structures detected from them. If cleaned datasets can be used to produce community structure with lower *mixed community rates*, the cleaning method can show its effectiveness in community analysis.

Although we mainly focus on the Girvan-Newman algorithm in the scope of this paper, the Louvain algorithm [10], a commonly-used community detection algorithm [60], is also used as a benchmark to verify if the two algorithms produce significantly different results. Particularly, we utilise *v-measure score* [54], a widely-used clustering metric to measure the agreement of two independent community assignments strategies produced by the two algorithms for each dataset. Furthermore, we also investigate if our cleaning method can support the Louvain method to generate better communities with lower *mixed community rates* for the cleaned data in comparison with the original data.

3.7. Early Prediction of Learning Outcome

To evaluate the predictability of the students' interaction with course material items for the learning outcomes, we use the *student-event item data matrix* as the input variable. Particularly, we combine the *student-event item data matrix* of Course#1 in both academic years into a single tabular dataset. Then, we conduct the cleaning method proposed in Section 3.5.2 on the dataset, forming *fully cleaned data* and *partly cleaned data*. For comparison purposes, we also use the original and PCA transformed datasets as predictors. The target variable is defined based on the student's scores on each lab exam. There are three lab exams for Course#1 on Week 4, 8 and 12. We classify students who achieved more than 40% in the exam as *higher-performing* or passed students and the remaining as *lower-performing* students.

For each week, a *student-event data matrix* has been extracted from the corresponding of the weekly event logs. The data collected in a certain week contains recorded learning events from the beginning of the course to that week. Then, the weekly data was used to predict the student results for the next exam. For example, in Course#1, the data collected in weeks 1, 2, 3, 4 are used to predict the results of the lab exam 1; the data in weeks 5, 6, 7 and 8 are used for predicting lab exam 2 results and the remaining weekly data are used to forecast the last exam result.

In terms of prediction algorithms, Support Vector Machine (SVM) appeared to be the most effective technique for the data captured from MOOCs in many contexts [21, 2]. In addition to SVM, for references and comparison purposes, we also pick four additional classification techniques including XGBoost [19], Logistic Regression [12], Gradient Boosting [28] and K Nearest Neighbours [23] due to their widely applications in Learning Analytics domain [44]. In terms of development tools, we use *sklearn* libraries [47] and Python as the main programming language.

Each dataset serves as input data for all algorithms with the same parameter configuration in each technique. In each dataset, 80% of the data has been used for training the models and the remaining 20% are for validation. The 10-fold cross-validation technique has also been applied, using *ROC_AUC*, *Accuracy* and *F1 scores*, to evaluate the predicting performance of each model.

Table 5
Summary of the features extracted from the four datasets at the end of the courses (after week 12)

Dataset	Number of materials	Number of transitions	Number of students
Course#1-2018	37	819	112
Course#1-2019	37	867	155
Course#2-2018	26	409	62
Course#2-2019	26	496	48

4. Experimental results

In this section, we present the analysis results of the four datasets mentioned in Section 3.1. Learning behavioural features are constructed from the four datasets, as summarised in Table 5. Course#1 has more material items and transitions than Course#2. This is because Course#1 has been delivered to Software Engineering students and was more intensive than Course#2 which targets Business Computing students who may be "less-technical".

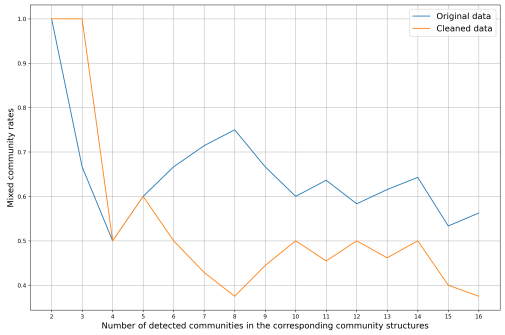

Figure 5: Mixed community rates in community structures for Course#1_2018

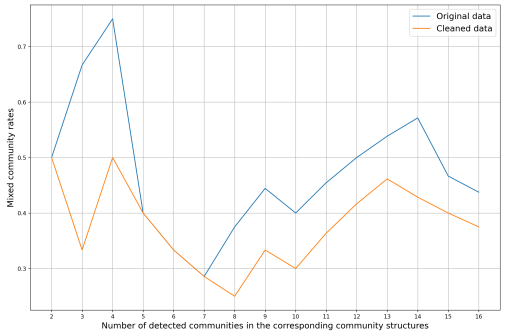

Figure 6: Mixed community rates in community structures for Course#1_2019

4.1. Community Analysis

4.1.1. Selecting community structure

The extracted datasets illustrated in Table 5 have been standardised, and this is followed by the calculation of cross-correlation matrices. The correlation matrices are cleaned before being used to calculate the learning behavioural distance matrices. The distance matrices have been used to construct MSTs. In other words, for each module, we construct a graph as an MST to display the similarity and dissimilarity of the students' learning behaviours. Based on the MSTs, we implement the Girvan-Newman algorithm for community detection. Students in each module can be divided into a smaller number of communities based on the distance between their learning behaviours and other learners' behaviours.

We note that the number of groups to be detected by the Girvan-Newman algorithm can be configurable depending on analysis purposes, forming a community structure. In this research, we rely on *Mixed community rates*, i.e. a good community structure should contain fewer *mixed community rate* and more number of *higher* or *lower performing communities*. Figure 5 and Figure 6 shows the investigation of *mixed community rate* for each possible community structure detect by the algorithm in Course#1 in both academic years. Indeed, the number of detected communities can go up to the

total number of students in the whole graph. However, we do not want a fragmented community structure where each community contains only a few students. Hence, we merely show a part of possible community structures in both figures.

Table 6
Community detection summary for Course#1

Group	Course#1 - 2018		Course#1 - 2019	
	Number of students	Average grade	Number of students	Average grade
Group 1	10	**0.78**	15	**0.76**
Group 2	18	0.56	26	0.64
Group 3	14	0.44	14	0.64
Group 4	16	0.42	24	0.51
Group 5	21	0.35	23	0.36
Group 6	06	0.16	16	0.31
Group 7	17	0.16	18	0.29
Group 8	10	**0.08**	15	**0.20**

Table 7
Community detection summary for Course#2

Group	Course#2 - 2018		Course#2 - 2019	
	Number of students	Average grade	Number of students	Average grade
Group 1	05	**0.75**	06	**0.91**
Group 2	05	0.60	05	0.9
Group 3	09	0.58	06	0.87
Group 4	09	0.52	06	0.75
Group 5	09	0.44	05	0.7
Group 6	04	0.43	07	0.67
Group 7	08	0.43	08	0.56
Group 8	07	0.39	05	**0.55**
Group 9	06	**0.33**	N/A	N/A

Both Figures 5 and 6 show that the cleaned dataset have a better support for community detection in comparison with the original dataset. Overall, the mixed community rate in the community structures detected using the cleaned datasets are lower than the figures for the original datasets for Course#1. We also observed a similar phenomenon for the Course#2 datasets. In addition, based on these figures, it is possible to determine the community structures, using the lowest point of the mixed community rate line. The detected results can be seen in Table 6 for Course#1 and Table 7 for Course#2. In Table 6, eight groups have been detected with the number of students in each group and its average grades of the final lab exam in week 12. Similarly, Table 7 displays nine detected groups for Course#2-2018 and eight groups for Course#2-2019. All groups are ordered from the highest to the lowest average grades in the tables.

4.1.2. Analysing highest vs lowest performing communities

The highest and lowest-performing communities in Table 6 and 7 can be picked for the further investigation of the difference of interactions with course material items among the student cohorts. Table 8 demonstrates the difference in the number of learning activities in using learning material items in Course#1 between the two groups while the results for the Course#2 can be seen in Table 9. For each item, non-parametric statistical tests (i.e. Mann-Whitney U Test [46]) have been used to verify if there is a significant difference between the highest and lowest performing communities in terms of using the item during the courses. The course material items in which the highest and the lowest performing communities have a significant difference in the number of events (p-value < 0.05) are highlighted.

Table 8
Highest vs Lowest performing communities in Course#1. The asterisks indicate the learning items where there is a significant difference between the two communities (p-value < 0.05). In particular, * if there is only a significant difference in Course#1-2018 only, ** if there is only a significant difference in Course#1-2019 only and *** if there are significant differences in both academic years

Items	Course#1 - 2018		Course#1 - 2019	
	Highest performing community	Lowest performing community	Highest performing community	Lowest performing community
General	160.0	159.0	403.0	355.0
Lecture1**	43.0	56.0	57.0	**70.0**
Lecture2*	53.0	**129.0**	121.0	110.0
Lecture3*	15.0	**157.0**	129.0	112.0
Lecture4*	33.0	**72.0**	50.0	47.0
Lecture5***	53.0	**98.0**	76.0	**79.0**
Lecture6***	36.0	**122.0**	148.0	**150.0**
Lecture7***	31.0	**87.0**	82.0	**104.0**
Lecture8**	32.0	38.0	48.0	**59.0**
Lecture9**	128.0	167.0	**140.0**	134.0
Lecture10*	91.0	**102.0**	91.0	76.0
Lecture11***	15.0	**42.0**	81.0	**138.0**
Lecture12**	25.0	2.0	61.0	**106.0**
Labsheet1***	167.0	**245.0**	**164.0**	66.0
Labsheet2**	283.0	352.0	**440.0**	235.0
Labsheet3***	199.0	**263.0**	**313.0**	116.0
Labsheet4	95.0	100.0	206.0	121.0
Labsheet5**	325.0	223.0	**442.0**	208.0
Labsheet6	244.0	213.0	410.0	225.0
Labsheet7***	**245.0**	139.0	**384.0**	194.0
Labsheet8	85.0	85.0	162.0	107.0
Labsheet9**	204.0	132.0	**451.0**	194.0
Labsheet10**	256.0	140.0	**345.0**	137.0
Labsheet11***	**163.0**	100.0	**159.0**	104.0
Labsheet12*	**80.0**	27.0	120.0	133.0
Practice1	286.0	379.0	223.0	52.0
Practice2***	427.0	**613.0**	**417.0**	190.0
Practice3	266.0	348.0	355.0	168.0
Practice4*	173.0	**308.0**	387.0	222.0
Practice5**	254.0	185.0	**382.0**	217.0
Practice6***	**277.0**	185.0	**346.0**	206.0
Practice7***	**246.0**	88.0	**490.0**	390.0
Practice8**	251.0	263.0	**607.0**	232.0
Practice9***	**360.0**	138.0	**493.0**	215.0
Practice10***	**291.0**	89.0	**492.0**	315.0
Practice11***	**328.0**	102.0	**502.0**	426.0
Practice12***	**234.0**	163.0	**639.0**	367.0

Table 9
Highest vs Lowest performing communities in Course#2. Statistical tests were not conducted in this result because there are merely a small number of students in the two communities(i.e. 5 and 6 students).

Items	Course#2 - 2018		Course#2 - 2019	
	Highest performing community	Lowest performing community	Highest of performing community	Lowest performing community
General	211.0	231.0	93.0	107.0
Lecture1	37.0	100.0	93.0	19.0
Lecture3	54.0	86.0	32.0	30.0
Lecture5	42.0	43.0	29.0	31.0
Lecture9	65.0	19.0	59.0	5.0
Labsheet1	168.0	128.0	269.0	163.0
Labsheet2	127.0	56.0	228.0	98.0
Labsheet3	183.0	68.0	256.0	104.0
Labsheet4	203.0	68.0	315.0	123.0
Labsheet5	169.0	60.0	227.0	60.0
Labsheet7	247.0	144.0	151.0	59.0
Labsheet8	196.0	63.0	104.0	8.0
Labsheet9	29.0	9.0	203.0	18.0
Labsheet10	91.0	15.0	145.0	29.0
Labsheet11	40.0	4.0	96.0	12.0
Practice1	54.0	20.0	112.0	60.0
Practice2	115.0	52.0	335.0	35.0
Practice3	180.0	22.0	452.0	43.0
Practice4	171.0	24.0	400.0	57.0
Practice5	129.0	138.0	453.0	281.0
Practice6	109.0	66.0	161.0	31.0
Practice7	675.0	94.0	500.0	1.0
Practice8	247.0	26.0	382.0	20.0
Practice9	67.0	22.0	345.0	4.0
Practice10	81.0	0.0	427.0	0.0
Practice11	84.0	65.0	274.0	440.0

Regarding learning events on practice-related items, students in the highest performing community appeared to be more active than the lowest-performing community, with the higher average number of learning events in all Practice and Labsheet items across all four datasets. These gaps are likely to increase over time. For example, in Course#1-2018, the average number of events in *Practice_11* (i.e. practice items in week 11) of the highest performing community is about three times higher than the figures for the lowest performing community. A similar phenomenon can be observed in the data of other cohorts. Nevertheless, students in the lowest performing community are recorded to create a higher number of events in lecture records for both classes of Course#1 and Course#2-2018. For example, the number of events in lecture notes in weeks 2-7 created by the lowest performing community is about two to three times higher than the figures for the highest performing community in Course#1.

4.1.3. Girvan Newman vs Louvain methods

We compare the Girvan Newman community detection results selected above (Table 6 and 7) and the corresponding results produced by the Louvain method in the four datasets. Table 10 illustrates an investigation of the possible difference between the Louvain and Girvan Newman method when both algorithms are applied to the cleaned data and original data for the four datasets. Overall, the results from the Louvain and Girvan Newman method Overall, the results from the Louvain and Girvan Newman methods appear to be broadly similar. There are strong agreements between the community detection results of both algorithms with *v-measure scores* being greater than 0.82. It can also be seen that the *v-measure scores* for the cleaned data tend to be higher than that of the original data. This may imply that the proposed cleaning method may support the reduction of variation between the two algorithms when they are applied to the same dataset. Additionally, the results in Table 10 indicate the values for *mixed community rates* of the community structure detected by the Louvain method. The rates for the cleaned data are likely to be lower than those for the original data. This is consistent with the application of the Girvan-Newman method, i.e. the cleaned data can also support the Louvain method to deliver better community detection results with the lower number of mixed communities.

Table 10
Comparison community detection results between the Girvan-Newman and Louvain method in both cleaned and orignal datasets.

Dataset	Cleaned data		Original data	
	v-measure score	mixed community rate	v-measure score	mixed community rate
Course#1-2018	0.86	0.41	0.86	0.55
Course#1-2019	0.85	0.25	0.82	0.31
Course#2-2018	0.87	0.5	0.85	0.5
Course#2-2019	1	0.12	0.86	0.57

4.2. Learning Outcome Prediction Results

4.2.1. Comparison of Dataset pre-processing strategies

Figure 8, 9 and 7 demonstrate the *ROC_AUC*, *Accuracy*, and *F1* scores from different models and input datasets, respectively. It can clearly be seen that the three figures illustrate a similar pattern in the difference of the evaluation metrics among the models. Overall, fully and partly cleaned datasets appear to have better-predicting performances in comparison with other data preparation strategies. In particular, regarding the fully cleaned dataset, the Gradient Boosting outperforms other models in all three metrics (i.e. Accuracy: 0.79, F1 score: 0.785 and ROC_AUC: 0.81), followed by KNN models, XGBoost and SVM. In terms of the partly cleaned dataset, the models have shown that they have been well-performing in all algorithms, especially in XGBoost with the highest metric scores (i.e. Accuracy: 0.74, F1 score: 0.735 and ROC_AUC: 0.75) in comparison with other algorithms.

Conversely, models using the original and full PCA datasets appear to have lower performances across all predicting algorithms with the scores roughly around 0.60 and 0.70. Meanwhile, although the PCA dataset with the top largest prin-

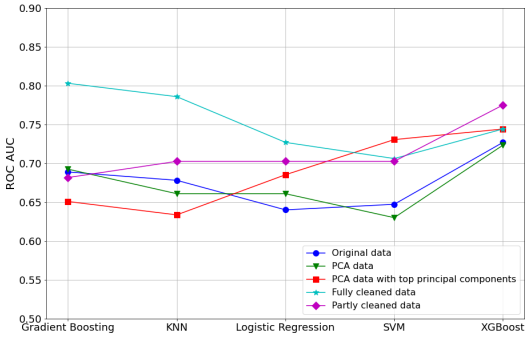

Figure 7: Comparison of the roc_auc scores of predicting models using different data pre-processing strategies.

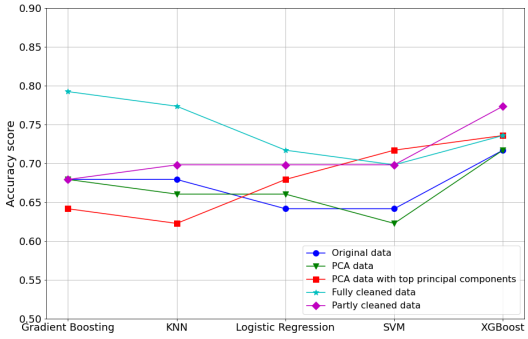

Figure 8: Comparison of the accuracy scores of predicting models using different data pre-processing strategies

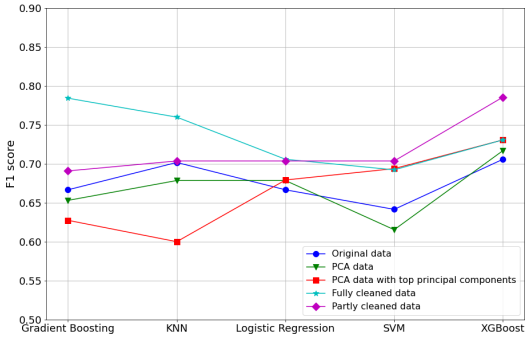

Figure 9: Comparison of the f1 scores of predicting models using different data pre-processing strategies.

cipal components has the lowest performance in Gradient Boosting and KNN, the dataset has shown its predictability in the SVM algorithm with the highest accuracy and roc_auc scores. It is possible that when only top principal components were kept in the dataset, the noise part has been eliminated.

4.2.2. Early prediction investigation

Figure 10 illustrates the mean of the cross-validation on ROC_AUC score of the models using fully cleaned dataset over 12 weeks during the course while Figure 11 shows the results for the partly cleaned dataset. In general, most of the models can produce good predictions for the datasets after week 4. The ability to classify students of the data in the first four weeks is relatively poor, which is expected, probably due to the imbalance of the number of passed and failed students in lab exam 1. In fact, lab exam1 usually comprises the easiest tasks which merely require the understanding of simple concepts in programming, e.g. using variables, operators and inputs. As a result, the majority of students usually pass the first exam. However, the difficulty level increases over lab exams 2 and 3, causing the target variable to become more balanced. Hence, the models can better predict the pass or failure of a student in lab exams 2 and 3.

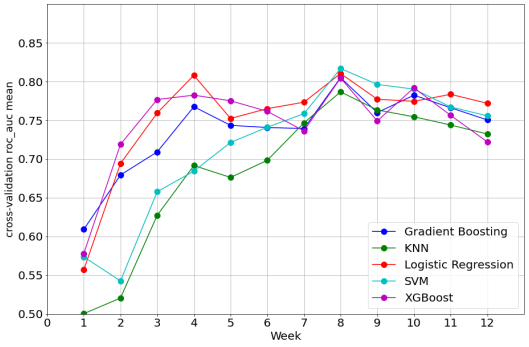

Figure 10: The 10-fold cross validation on roc_auc score of the models with fully cleaned data.

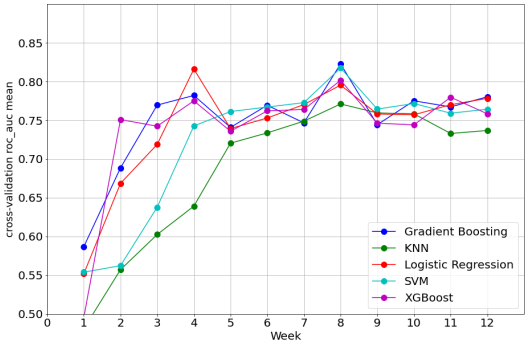

Figure 11: The 10-fold cross validation on roc_auc score of the models with partly cleaned data.

Although the performance of the models increases over time with the growth of the data collected, early data can support relatively good prediction. For example, the XGBoost model for week 5 data, which predicts the students' results of lab exam 2, achieves the ROC_AUC score of 0.78. The SVM model in week 9, which predicts the final exam in week 12, also achieves the acceptable result with a score of 0.80. These models appear to have a better performance than a recent prediction model in a similar computing educational context [27] where the author achieved the ROC_AUC score of 0.73 with the SVM model in the prediction of the student learning outcome in the Data Structure course. Therefore, early learning behaviours data may contain signs of students' learning outcomes [45] and can be good predictors, holding the potential to be a "leading indicator" of "at-risk" students.

5. Implications and Limitations

From the above, we believe that it is possible to say that there is a relationship between the students' learning behaviours and their exam performances. We found that the students who are grouped in the same community were likely to achieve

similar exam results. In other words, students having similar learning behaviours tend to perform similarly in the exam. This finding is in agreement with [40, 63] where the authors have defined and analysed various learning styles with different learners' behaviours in perceiving and responding to learning environments. Moreover, the learning styles appeared to affect students' satisfaction and can also be a useful indicator of learning success [57].

Overall, the learning behaviours of students in Course#1 and Course#2 in both academic years tend to be similar. In both modules, we have found what seem to be differences between lower and higher performing communities. In particular, higher-performing students were found to be more active in practising-related items such as navigating lab sheets and doing exercise. Besides, the higher performing students consistently interacted with course material items and exercises during the courses. The lower performing students, however, appeared to lose their focus and motivation to practice, i.e. actually do programming tasks, in the later stages of the study. This result is consistent with the initial investigation of programming [42] that *practice* is essential for improving students' programming skills. These findings, available so early in the semester, are essential for such core courses especially since it has been found that students should be given opportunities to practice and receive constructive feedback [5]. In [61], the authors indicated that programming skills may be improved if students practice frequently. However in the context of this research, the students from the lower performing group might face challenges during their study progress at the later stage of the course, e.g. the knowledge was becoming more difficult to understand. As a consequence, they might lose their confidence and motivation to actively participate in practical sessions, additionally highlighting the need for early intervention and encouragement. Besides, we noticed that the lower performing students tended only to try to solve only those programming tasks according to the common methods rather than creatively trying different approaches. As a result, they mostly tend to upload solutions once and move to other tasks. In contrast, the higher performing students tend to try various approaches for a given programming task and they submitted them all and once, leading to a higher number of events in practical items logged on the system, in comparison with the practical activities of lower performing communities.

We also found that there seems to be a distinction between the learning behaviours of Course#1 and Course#2 cohorts. In Course#1, the lower performing students appear to focus more on reading lecture notes than higher performing students. However, this phenomenon is not observed in Course#2. Particularly in Course#2, there is almost no difference in reading lecture notes between the two types of students. Even, the higher performing students in Course#2 seems to be becoming more active in reading lecture notes in several weeks during the semester. In fact, the level of knowledge in Course#2 tended to be lower than Course#1 with less advanced concepts and examples. We note that Course#1 was designed for Computer Science students and has a higher level of requirements for acquired knowledge and skills. Perhaps, the lower performing students in Course#1 might be struggling with acquiring new advanced concepts, which would keep them engage more with lecture notes instead of doing programming tasks.

In terms of the learning outcome prediction, using log data collected from online learning systems to predict students' success has been highly developed in the literature. There have been many scientific reports on building an early predicting system in many application contexts, from flagging "at-risk" students [7], to recommending next courses [59] and learning strategies [38, 4]. In our research, we provide a pre-processing data method that has been proven to be effective in improving the performance of widely used machine learning models in our context, i.e. programming education. This method can also be extended to different application contexts above as long as the data satisfies the assumptions of Random Matrix Theory.

We recommend instructors to keep implementing community detection and prediction as students' results come in. Other performance indicators can also be used in addition to lab exam grades, such as weekly exercise results. In practice, community detection can be implemented at any point during the study. Once communities are detected, the instructors can implement promptly interventions. For example, the higher performing groups can be given harder exercises to keep them focused and avoid getting bored of the study. On the other hand, the lower performing groups should be given more basic tasks along with instructions or tutor sessions. Furthermore, the instructors can provide lower performing communities with additional supporting materials or easier tasks with solutions. This would fill the knowledge gap and build up the confidence and motivation for the students as well as re-engage them in the study.

However, although the proposed method appears to be successful in reflecting the relationship between students' learning behaviours and learning performance, there are limitations due to the assumption of Random Matrix Theory which might restrict the method from being applicable to all kinds of learning behavioural data. The distribution of eigenvalues is given by Equation 6 when the sample size (matrix rows) $m \to \infty$ and number of features (matrix column) $n \to \infty$, provided that the ratio of rows and columns is greater than or equal to 1. Hence, in the context of this paper, the number of transitions extracted from event log data is needed to be greater than the number of students. In addition, the application of the RMT could be less effective for small size datasets, i.e. with a small number of students and course material items, although in that case community detection might not be that useful.

There is also a concern in terms of using MST to reduce the size of the graph. When a distance matrix contains duplicate values, the associated graph will have duplicated edges. Consequently, there can be more than one MST being generated from the graph and thus the results of the analysis may not be stable. In such cases, other graph size reduction techniques can be considered to obtain a single reduced graph,

ensuring the stability of results in further analysis. For example, in [33], the authors proposed a network sparsification technique that sparsifies the network while preserving network structures and community properties. The comparison between such techniques is out of the scope of this paper and will be the target for future works in line with this research.

In the future, we will also focus on changing the community structure (currently represented as an MST) of the students during the course. For example, a student may change their group in a different week, which may reveal that his or her learning behaviour also changes accordingly. This analysis can help to understand thoroughly how students are studying and provide better support for educators to improve the curriculum. However, this requires more advanced research approaches to be developed to process more complex data. The time duration on course material items will also be considered on top of the number of events in future works. Additionally, while it has been found that the community analysis results, when using either Girvan-Newman or Louvain method, do not vary significantly, the relationship between community detection techniques and analysis results is also worth further investigation, we believe. We will target this in future works, to further investigate all the insights of learning behaviours among student communities.

6. Conclusion

In this paper, we propose a novel approach to analyse the students' learning behaviours data collected from an online learning system in the context of programming education. This research is one of the first attempts to utilise RMT and Community Detection in the educational domain. The analysis is based on a range of techniques. First, we extract a transition-student data matrix from the event log data. Second, we clean the effect of noise and trend in the correlation matrix of the transition-student data matrix, which is based on the Random Matrix Theory. This cleaning process can help to reveal the underlying meaning of the data. The cleaned correlation matrix is used to construct a distance matrix and the Minimum Spanning Tree. The MST can represent the relationships of students' learning behaviours in using course material items in the form of an MST graph. Students having similar behaviours are closer to each other in the constructed MST graph. The community detection algorithm, i.e. Girvan Newman, has been applied to detect the smaller student groups from the MST. Furthermore, the student-event data matrix is also cleaned and used as input variables to predict the learning outcome of students in the lab exams, using a range of machine learning classification techniques. The findings from the above method have been used to analyse the learning behaviours of students with different learning abilities in programming. The proposed approach in cleaning learning behavioural data also shows its effectiveness in community analysis and building early prediction models. Insights from students' learning behaviours and recommendations are also discussed in the paper.

Acknowledgment

This research is supported by the Irish Research Council under the project number GOIPG/2017/141 and from the ADAPT Centre for Digital Content Technology, funded under the SFI Research Centres Programme (Grant 13/RC/2106_P2), co-funded by the European Regional Development Fund. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission. The authors also acknowledge the support of Dr. Stephen Blott, School of Computing, Dublin City University, on data collection and using the Einstein learning system.

References

[1] Aguiar, E., Chawla, N.V., Brockman, J., Ambrose, G.A., Goodrich, V., 2014. Engagement vs performance: using electronic portfolios to predict first semester engineering student retention , 103–112.

[2] Al-Shabandar, R., Hussain, A., Laws, A., Keight, R., Lunn, J., Radi, N., 2017. Machine learning approaches to predict learning outcomes in massive open online courses , 713–720.

[3] Alsheddy, A., Habib, M., 2017. On the application of data mining algorithms for predicting student performance: A case study. Int. J. Comput. Sci. Netw. Secur 17, 189–197.

[4] Barthakur, A., Kovanovic, V., Joksimovic, S., Siemens, G., Richey, M., Dawson, S., 2021. Assessing program-level learning strategies in moocs. Computers in Human Behavior 117, 106674.

[5] Ben-Ari, M., 2001. Constructivism in computer science education. J. of Computers in Mathematics and Science Teaching 20, 45–73.

[6] Bennedsen, J., Caspersen, M.E., 2019. Failure rates in introductory programming: 12 years later. ACM inroads 10, 30–36.

[7] Berens, J., Schneider, K., Görtz, S., Oster, S., Burghoff, J., 2018. Early detection of students at risk–predicting student dropouts using administrative student data and machine learning methods. Journal of Educational Data Mining .

[8] Bhattacharya, S., Maddikunta, P.K.R., Kaluri, R., Singh, S., Gadekallu, T.R., Alazab, M., Tariq, U., et al., 2020. A novel pca-firefly based xgboost classification model for intrusion detection in networks using gpu. Electronics 9, 219.

[9] Blikstein, P., 2011. Using learning analytics to assess students' behavior in open-ended programming tasks , 110–116.

[10] Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E., 2008. Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment 2008, P10008.

[11] Bouchaud, J.P., Potters, M., . Financial applications of random matrix theory:a short review. The Oxford Handbook of Random Matrix Theory .

[12] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., et al., 2013. Api design for machine learning software: experiences from the scikit-learn project. arXiv preprint arXiv:1309.0238 .

[13] Bun, J., Bouchaud, J.P., Potters, M., 2017. Cleaning large correlation matrices:tools from random matrix theory. Physics Reports 666, 1–109.

[14] Bun, J., Knowles, A., 2018. An optimal rotational invariant estimator for general covariance matrices: The outliers. Preprint .

[15] Cairns, A.H., Gueni, B., Fhima, M., Cairns, A., David, S., Khelifa, N., 2015. Process mining in the education domain. International Journal on Advances in Intelligent Systems 8, 219–232.

[16] Carter, A.S., Hundhausen, C.D., 2017. Using programming process data to detect differences in students' patterns of programming , 105–110.

[17] Carter, A.S., Hundhausen, C.D., Adesope, O., 2017. Blending measures of programming and social behavior into predictive models of student achievement in early computing courses. ACM Transactions on Computing Education (TOCE) 17, 1–20.

[18] Chaudhari, H., Crane, M., 2020. Cross-correlation dynamics and

community structures of cryptocurrencies. Journal of Computational Science 44, 101130.
[19] Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., et al., 2015. Xgboost: extreme gradient boosting. R package version 0.4-2 1.
[20] Conlon, T., Ruskin, H.J., Crane, M., 2007. Random matrix theory and fund of funds portfolio optimisation. Physica A: Statistical Mechanics and its applications 382, 565–576.
[21] Costa, E.B., Fonseca, B., Santana, M.A., de Araújo, F.F., Rego, J., 2017. Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses. Computers in Human Behavior 73, 247–256.
[22] Crossley, S., Liu, R., McNamara, D., 2017. Predicting math performance using natural language processing tools , 339–347.
[23] Cunningham, P., Delany, S.J., 2020. k-nearest neighbour classifiers–. arXiv preprint arXiv:2004.04523 .
[24] Doleck, T., Jarrell, A., Poitras, E.G., Chaouachi, M., Lajoie, S.P., 2016. Examining diagnosis paths: A process mining approach , 663–667.
[25] Du, X., Yang, J., Hung, J.L., Shelton, B., 2020. Educational data mining: a systematic review of research and emerging trends. Information Discovery and Delivery .
[26] Dyson, F.J., 1971. Distribution of eigenvalues for a class of real symmetric matrices. Revista Mexicana de Fisica 20, 231–237.
[27] Erickson, V.L., 2019. Data-driven models to predict student performance and improve advising in computer science , 3–9.
[28] Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Annals of statistics , 1189–1232.
[29] Gajewski, Ł., Chołoniewski, J., Hołyst, J., 2016. Key courses of academic curriculum uncovered by data mining of students' grades. arXiv preprint arXiv:1604.07074 .
[30] Gárate-Escamila, A.K., El Hassani, A.H., Andrès, E., 2020. Classification models for heart disease prediction using feature selection and pca. Informatics in Medicine Unlocked 19, 100330.
[31] Gašević, D., Dawson, S., Rogers, T., Gasevic, D., 2016. Learning analytics should not promote one size fits all: The effects of instructional conditions in predicting academic success. The Internet and Higher Education 28, 68–84.
[32] Ghazal, M.A., Ibrahim, O., Salama, M.A., 2017. Educational process mining: a systematic literature review , 198–203.
[33] Gionis, A., Rozenshtein, P., Tatti, N., Terzi, E., 2017. Community-aware network sparsification, in: Proceedings of the 2017 SIAM International Conference on Data Mining, SIAM. pp. 426–434.
[34] Girvan, M., Newman, M.E., 2002. Community structure in social and biological networks. Proceedings of the national academy of sciences 99, 7821–7826.
[35] Haff, L., 1980. Empirical bayes estimation of the multivariate normal covariance matrix. The Annals of Statistics , 586–597.
[36] Hung, J.L., Wang, M.C., Wang, S., Abdelrasoul, M., Li, Y., He, W., 2015. Identifying at-risk students for early interventions: A time-series clustering approach. IEEE Transactions on Emerging Topics in Computing 5, 45–55.
[37] Hwang, W.Y., Shadiev, R., Wang, C.Y., Huang, Z.H., 2012. A pilot study of cooperative programming learning behavior and its relationship with students' learning performance. Computers & education 58, 1267–1281.
[38] Jovanović, J., Dawson, S., Joksimović, S., Siemens, G., 2020. Supporting actionable intelligence: reframing the analysis of observed study strategies, in: Proceedings of the Tenth International Conference on Learning Analytics & Knowledge, pp. 161–170.
[39] Kim, D.H., Jeong, H., 2005. Systematic analysis of group identification in stock markets. Physical Review E 72, 046133.
[40] Kolb, D.A., 2014. Experiential learning: Experience as the source of learning and development. FT press.
[41] Laloux, L., Cizeau, P., Potters, M., Bouchaud, J.P., 2000. Random matrix theory and financial correlations. International Journal of Theoretical and Applied Finance 3, 391–397.
[42] Mai, T.T., Crane, M., Bezbradica, M., 2021. Students' behaviours in using learning resources in higher education: How do behaviours reflect success in programming education?, in: Proceedings of the 7th International Conference on Higher Education Advances (HEAd'21), pp. 47–55.
[43] Markova, T., Glazkova, I., Zaborova, E., 2017. Quality issues of online distance learning. Procedia-Social and Behavioral Sciences 237, 685–691.
[44] Moreno-Marcos, P.M., Alario-Hoyos, C., Muñoz-Merino, P.J., Kloos, C.D., 2018. Prediction in moocs: A review and future research directions. IEEE Transactions on Learning Technologies 12, 384–401.
[45] Na, K.S., Tasir, Z., 2017. Identifying at-risk students in online learning by analysing learning behaviour: A systematic review , 118–123.
[46] Nachar, N., et al., 2008. The mann-whitney u: A test for assessing whether two independent samples come from the same distribution. Tutorials in quantitative Methods for Psychology 4, 13–20.
[47] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.
[48] Plerou, V., Gopikrishnan, P., Rosenow, B., Amaral, L.A.N., Guhr, T., Stanley, H.E., 2002. Random matrix approach to cross correlations in financial data. Physical Review E 65, 066126.
[49] de Prado, M.M.L., 2020. Machine learning for asset managers. Cambridge University Press.
[50] Reimann, P., Frerejean, J., Thompson, K., 2009. Using process mining to identify models of group decision making in chat data .
[51] Ren, Q., Peng, X., Liu, X., Zheng, Q., He, T., Zhang, L., 2021. Network modelling and visualisation analysis of the undergraduate dental curriculum system in china. Journal of Computer and Communications 9, 38–51.
[52] Romero, C., Cerezo, R., 2016. Educational process mining: a tutorial and case study using moodle data sets 00072.
[53] Romero, C., Ventura, S., 2013. Data mining in education. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 3, 12–27.
[54] Rosenberg, A., Hirschberg, J., 2007. V-measure: A conditional entropy-based external cluster evaluation measure, in: Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL), pp. 410–420.
[55] Sharifi, S., Crane, M., Shamaie, A., Ruskin, H., 2004. Random matrix theory for portfolio optimization: a stability approach. Physica A: Statistical Mechanics and its Applications 335, 629–643.
[56] Sharma, K., Papamitsiou, Z., Olsen, J.K., Giannakos, M., 2020. Predicting learners' effortful behaviour in adaptive assessment using multimodal data, in: Proceedings of the Tenth International Conference on Learning Analytics & Knowledge, pp. 480–489.
[57] Shaw, R.S., 2012. A study of the relationships among learning styles, participation types, and performance in programming language learning supported by online forums. Computers & Education 58, 111–120.
[58] Sin, K., Muthu, L., 2015. Application of big data in education data mining and learning analytics–a literature review. ICTACT journal on soft computing 5.
[59] Sweeney, M., Lester, J., Rangwala, H., Johri, A., et al., 2016. Next-term student performance prediction: A recommender systems approach. Journal of Educational Data Mining 8, 22–51.
[60] Traag, V.A., Waltman, L., Van Eck, N.J., 2019. From louvain to leiden: guaranteeing well-connected communities. Scientific reports 9, 1–12.
[61] Truong, N., Bancroft, P., Roe, P., 2003. A web based environment for learning to program, in: Proceedings of the 26th Australasian computer science conference-Volume 16, pp. 255–264.
[62] Van Der Aalst, W., 2012. Process mining. Communications of the ACM 55, 76–83.
[63] Wang, K.H., Wang, T.H., Wang, W.L., Huang, S.C., 2006. Learning

styles and formative assessment strategy: enhancing student achievement in web-based learning. Journal of computer assisted learning 22, 207–217.

[64] Wei, H., Li, H., Xia, M., Wang, Y., Qu, H., 2020. Predicting student performance in interactive online question pools using mouse interaction features, in: Proceedings of the Tenth International Conference on Learning Analytics & Knowledge, pp. 645–654.

[65] West, D.B., et al., 2001. Introduction to graph theory. volume 2. Prentice hall Upper Saddle River.

[66] Wishart, J., 1928. The generalised product moment distribution in samples from a normal multivariate population. Biometrika , 32–52.

[67] Yang, S.J., Lu, O.H., Huang, A.Y., Huang, J.C., Ogata, H., Lin, A.J., 2018. Predicting students' academic performance using multiple linear regression and principal component analysis. Journal of Information Processing 26, 170–176.