SAPIENZA
UNIVERSITÀ DI ROMA

Latent Deep Sequential Learning of Behavioural Sequences

Computer Science Department

PhD in Computer Science – XXXIV Cycle

Candidate

Bardh Prenkaj
ID number 1602894

Thesis Advisors	Co-Advisors
Prof. Paola Velardi	Prof. Giovanni Stilo
Prof. Damiano Distante	Prof. Stefano Faralli

October 31st 2021

Thesis not yet defended

Latent Deep Sequential Learning of Behavioural Sequences
Ph.D. thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: prenkaj@di.uniroma1.it

Abstract

The growing use of asynchronous online education (MOOCs and e-courses) in recent years has resulted in increased economic and scientific productivity, which has worsened during the coronavirus epidemic. The widespread usage of OLEs has increased enrolment, including previously excluded students, resulting in a far higher dropout rate than in conventional classrooms. Dropouts are a significant problem, especially considering the rising proliferation of online courses, from individual MOOCs to whole academic programmes due to the pandemic. Increased efficiency in dropout prevention techniques is vital for institutions, students, and faculty members and must be prioritised. In response to the resurgence of interest in the student dropout prediction (SDP) issue, there has been a significant rise in contributions to the literature on this topic. An in-depth review of the current state of the art literature on SDP is provided, with a special emphasis on Machine Learning prediction approaches; however, this is not the only focus of the thesis. We propose a complete hierarchical categorisation of the current literature that correlates to the process of design decisions in the SDP, and we demonstrate how it may be implemented. In order to enable comparative analysis, we develop a formal notation for universally defining the multiple dropout models examined by scholars in the area, including online degrees and their attributes. We look at several other important factors that have received less attention in the literature, such as evaluation metrics, acquired data, and privacy concerns. We emphasise deep sequential machine learning approaches and are considered to be one of the most successful solutions available in this field of study.

Most importantly, we present a novel technique - namely GRU-AE - for tackling the SDP problem using hidden spatial information and time-related data from student trajectories. Our method is capable of dealing with data imbalances and time-series sparsity challenges. The proposed technique outperforms current methods in various situations, including the complex scenario of full-length courses (such as online degrees). This situation was thought to be less common before the outbreak, but it is now deemed important.

Finally, we extend our findings to different contexts with a similar characterisation (temporal sequences of behavioural labels). Specifically, we show that our technique can be used in real-world circumstances where the unbalanced nature of the data can be mitigated by using class balancement technique (i.e. ADASYN), e.g., survival prediction in critical care telehealth systems where balancement technique alleviates the problem of inter-activity reliance and sparsity, resulting in an overall improvement in performance.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

The novel coronavirus (COVID-19) outbreak forced many nations to massively push their education system towards online learning environments (OLEs) that comprise fast- and slow-paced online courses and online degrees. Institutions are also evaluating this new trend of digital adoption of OLEs as an opportunity for them to exploit e-learning technologies to teach more broadly and target students worldwide. Nevertheless, effectively teaching and assessing student performances is a task of paramount importance yet to be stabilised. Current means of delivering courses at any educational level are still in their initial version, and crucial social-economic factors are not covered. The literature of the Learning Analytics (LA) field [67] has given a considerable contribution to the study of online courses and online degrees (hereafter e-degrees) in various contexts ranging from psychological and social-economic indicators of student behaviour to machine learning predictions of future student interactions with the e-learning platforms. Online learning can be synchronous [37] or asynchronous [122]. When the OLEs are synchronous, students participate in real-time classes with other peers and instructors, and get immediate feedback and clarifications. Contrarily, when the OLEs are asynchronous, students can attend classes according to their own schedule, work at their own pace, and review lectures to clarify concepts. Here, we explore the asynchronous aspect of the OLEs and develop our insights based on their peculiarities.

One of the most critical aspects of OLEs is the high dropout rate of students w.r.t. their counterparts in traditional education institutions. Although online institutions have adopted initial countermeasures to lower dropout rates, a mitigation strategy is still missing. Academic researchers have also contributed to designing novel approaches to cope with the student dropout phenomenon, but a sophisticated tracking system for student activities is necessary. Online institutions (e.g. Coursera, edX, and Udemy) have proprietary software only applicable to their online courses scenario. Moreover, the complexity that e-degrees entail is higher than that of simple online courses and predicting whether students drop out is a challenging task.

In the literature, Student Dropout Prediction (SDP) is a research topic in the multidisciplinary field of Learning Analytics. In detail, it belongs to the area of Educational Data Mining (EDM) (see [8, 40, 104] for an overview of this field). SDP's specific objective is to analyse student dropout in OLEs by modelling student behaviour when interacting with e-learning platforms. Students interact with the e-learning platform of the course they are enrolled in by generating different activities traced in several log files. These activities - hereafter e-tivities - comprise data about how a student consumes course material such as videos, quizzes, homework assignments and forum questions. E-tivities help profile students according to their interactions, and by having automatic procedures analyse them, one can tackle the dropout phenomenon via machine/deep learning strategies.

As anticipated, OLEs suffer from a significant amount of dropout rate with regards to traditional institutions. Therefore, SDP represents a new research line in e-learning in general, and researchers should give particular attention to it. Online courses have been around for more than twenty years[1], but they achieved a boom in usage with the diffusion of Massive Open Online Courses (MOOCs). Notwithstanding their lasting presence, academia has posed little attention to the difficulties that "online students" experience during their studies in these institutions. The high number of enrolled students in MOOCs has given the literature the possibility to tackle the dropout problem thoroughly.

Consequently, a growing number of online institutions have considered adopting automated systems to predict their students' dropout decisions. Automated strategies, in turn, have boosted the attention of researchers, particularly those in the machine learning area. Nevertheless, real-time and intelligent prediction methodologies remain unexplored; whereas, simple statistical strategies based on ad-hoc surveys remain the most common approach to solve the SDP problem.

1.1 Motivation

Online education has been established as one of the wealthiest industries in the world. At the brisk of the pandemic, e-learning solutions have received an immense amount of incentives from governmental institutions, and private entities [11]. Notwithstanding the plethora of benefits that OLEs bring (e.g. ubiquity of learning), public and private institutions have a growing concern for low retention rates. The swift shift towards full-online education methods needs to be backed up by new teaching regulations to conform to students' needs during this period. Moreover, studies have shown that students enrolled in OLEs have a higher probability of abandoning their studies prematurely than their counterparts attending traditional classroom environments [19, 32, 38, 56]. To promote self-paced and ubiquitous learning, students in OLEs, especially MOOCs, can follow lectures according to their timetables and occupations (i.e. working students or professionals seeking an upgrade in their careers). With the lack of prerequisite courses and bureaucratic regulations, as in in-presence institutions, students can drop out at any time without severe financial repercussions besides not obtaining a degree/certification. Thus, the lack of penalties for dropout students induces online institutions to have a retention rate of 10-20% lower than that of traditional universities [55, 121].

Although e-learning institutions have managed the dropout students by employing ad-hoc manual strategies, more efficient and less time-consuming methodologies are required to mitigate the dropout phenomenon. SDP strategies can help institutions

[1]One of the pioneers in online education was Universitat Oberta de Catalunya - `https://www.uoc.edu/portal/en/index.html`.

to increase their retention rates substantially. Identifying beforehand which students are likely to abandon their studies is beneficial for distance learning institutions to provide bespoke intervention strategies to at-risk students. Table 1.1 shows online institutions with the highest graduation rates in Italy[2,3,4]. These institutions require to maximise their retention rates and minimise their economic losses - the dropout phenomenon is one of the most significant factors in economic wastes [81] - when a particular student decides to abandon their studies preemptively.

Table 1.1. Comparison of online institutions with the highest graduation rate in Italy. The number of online institutions in the Italy in 2020/2021 is 9. The overall number of students enrolled in online universities in Italy as of 2020/21 is $130,457$. The average number of enrolled students in 2020/21 per university is the ratio between the overall number of enrolled students and the number of colleges (i.e. ~ 1495).

Institution	Graduation rate	Number of enrolled students (2020/2021)	Ratio w.r.t. the average enrolled students (2020)
Unimarconi	16.1%	10,249	~ 6.85
Unifortunato	16.1%	1,800	~ 1.20
Pegaso	14.8%	65,611	~ 43.88
Online University "San Raffaele"	14.7%	8,008	~ 5.36
University of Rome Unitelma Sapienza	12.4%	2,727	~ 1.82

According to recent studies of the market investments and future projections, the adoption of online education technologies has received a large amount of financial support from private entities. Due to the pandemic diffusion, governments have supported the transition from traditional to online education with outstanding non-repayable loans. Although the bulk of the capital goes to furnishing teaching staff and students with necessary digital devices to continue their education at a distance, noticeable finances pour into novel EDM strategies and thus SDP. In 2019, the size of the global online e-learning market was approximately $101 billion. During that same period, the learning management system market generated roughly $18 billion. By 2026, the total market for e-learning worldwide is forecasted to grow exponentially (see Figure 1.1), reaching $370 billion by 2026. Statista[5] performed the projection study before the novel COVID-19 pandemic. Hence, it might get a higher investment budget during the period of study. Besides having an immense social and economic impact, SDP is a novel and interdisciplinary research problem comprising social and computer sciences. Therefore, SDP has increased academia's contribution presenting an increment of a 2.5 factor in the time interval from 2010 to 2020 (see Figure 1.2).

[2]According to the Italian Ministry of Education, Research, and University, the Online University "Leonardo Da Vinci" has the highest graduation rate (i.e. $\sim 41\%$) among all the online institutions in Italy in the 2020/2021 academic year. We do not include this university as the first entry in Table 1.1 since it had only 23 matriculated and 16 graduated students as of 2020/2021.

[3]We derived the statistics from the telematic institutions found at `http://ustat.miur.it/dati/didattica/italia/atenei#tabistituti`.

[4]The number of online universities in Italy is 11, but we do not take into consideration two of them (i.e. Online University of Rome "Niccolò Cusano" and Universitas Mercatorum) because they do not have complete statistics for 2020/2021.

[5]`https://www.statista.com/statistics/1130331/e-learning-market-size-segment-worldwide/`

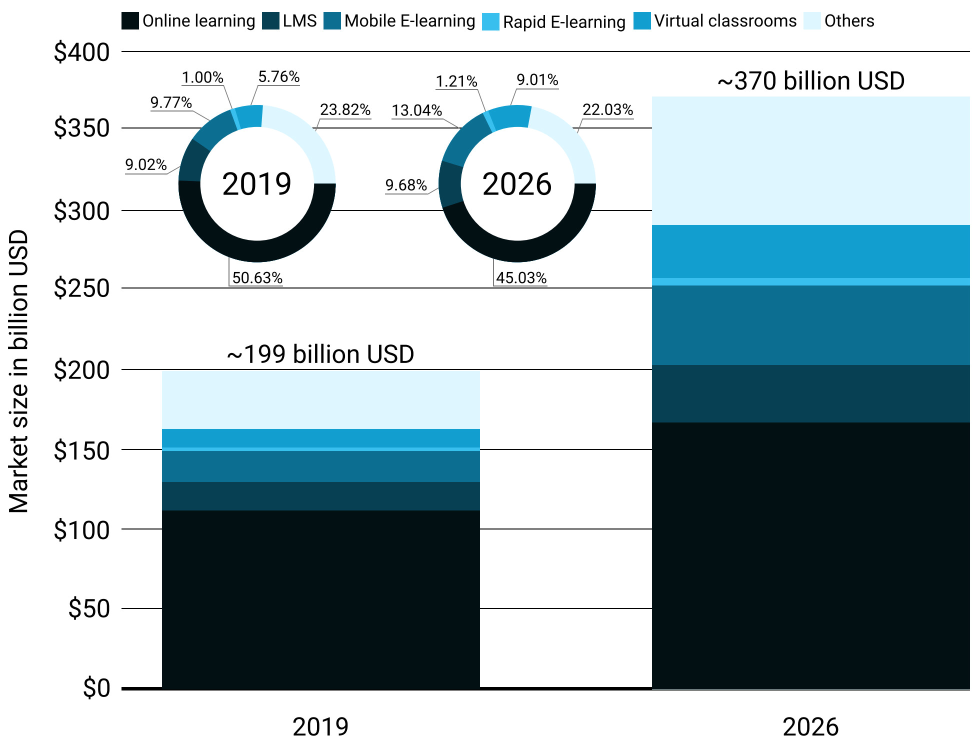


Figure 1.1. Size of the global e-learning (in billion USD) market in 2019 and 2026 by segment.

Student e-tivities are multiple and belong to many possible interactions, thus rendering the SDP problem challenging. These e-tivities are sequential when students interact with the resources of a single course. Simultaneously, they can be parallel since students attend more than one course in the scenario of e-degrees. E-tivities can also be interactive among peers through course forums and social networks integrated within the e-platforms. Therefore, building a model that considers these e-tivities and understanding their influence on students' performance is still an open issue. As mentioned before, academia has tackled the SDP problem by handling ad-hoc surveys and building complex rule systems based on them. Although the current availability of deep learning strategies offers the possibility to integrate sequential and parallel data, coping with student modelling complexity in OLEs is yet to be explored.

1.2 Objectives

The principal objectives of this work are two: i.e. theoretical and experimental. We group the main theoretical contributions of the work as follows:

- To provide a cornerstone in Learning Analytics, we formally define the student dropout phenomenon in asynchronous OLEs and present the first input modelling strategy following the adopted definition.

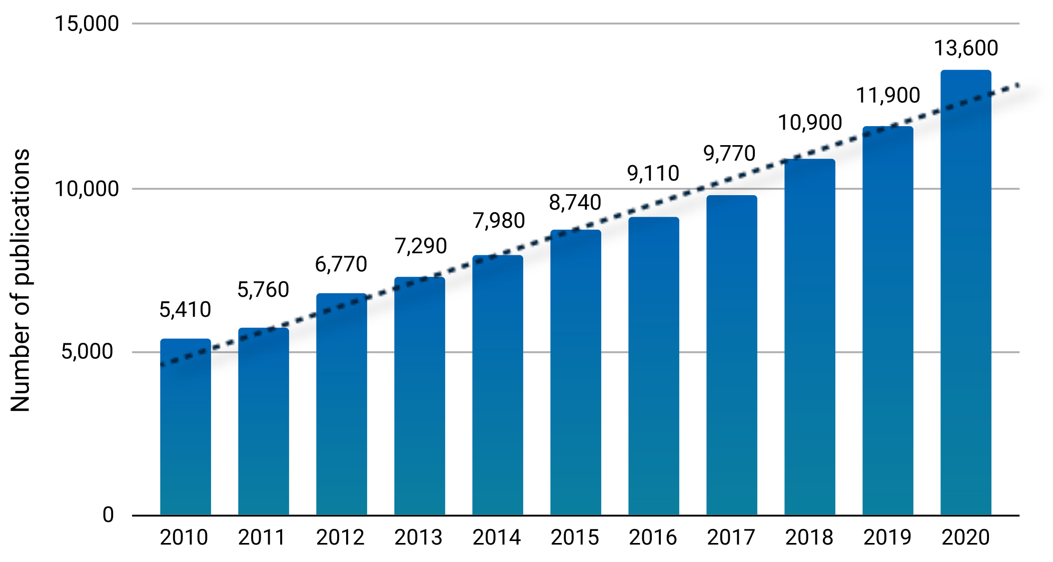


Figure 1.2. Yearly trend of publications in SDP. For each year 2010-2020, we calculate the number of publications that treat SDP. According to a search on `scholar.google.com`, we set the number of publications equal to the size of the result set for the keyword *student dropout in distance learning.*

- We provide a detailed classification of state-of-the-art approaches, both simple machine learning and deep learning, according to the chosen input modelling technique.
- We give a comprehensive evaluation framework of the proposed models on benchmarking datasets that contain student trajectories formed of e-tivities.

Furthermore, in the experimental contribution, we devise a novel strategy based on latent information and time-related data generated from students' interaction with e-tivities. Moreover, we demonstrate that the proposed method surpasses the state-of-the-art in the case of slow-paced degrees and fast-paced MOOCs. Finally, we test our model's performances in a real-case risk-prediction scenario on patient health trajectories.

1.3 Thesis Structure

To ease the reader into the thesis, Figure 1.3 depicts the structure of this work. Chapter 1 illustrates the phenomenon of student dropout in traditional and OLEs and its financial impact in learning institutions. Additionally, following a growing trend to be amplified in the near future, we describe different reasons to care about the student dropout prediction problem. Chapter 2 describes the methodologies presented in the literature exploited to solve the SDP problem. According to the works surveyed, we also identify open issues that we tackle throughout this thesis.

The main contribution of the thesis is divided into two main branches. The first comprises the theoretical background and input modelling definitions alongside

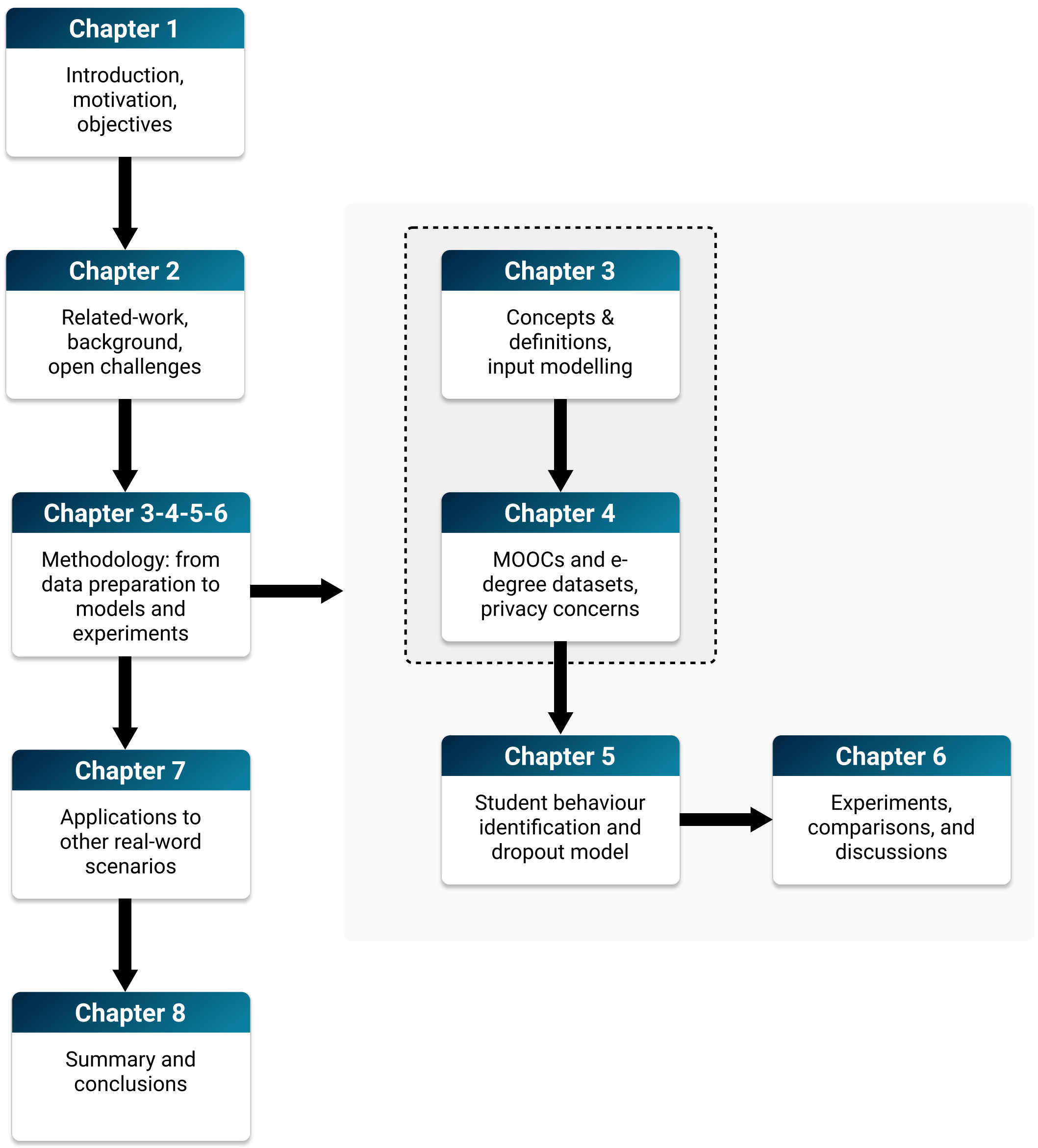

Chapter 1
Introduction, motivation, objectives
Chapter 2
Related-work, background, open challenges
Chapter 3-4-5-6
Methodology: from data preparation to models and experiments
Chapter 7
Applications to other real-word scenarios
Chapter 8
Summary and conclusions
Chapter 3
Concepts & definitions, input modelling
Chapter 4
MOOCs and e-degree datasets, privacy concerns
Chapter 5
Student behaviour identification and dropout model
Chapter 6
Experiments, comparisons, and discussions

Figure 1.3. The structure of the thesis.

the datasets used (Chapters 3 and 4), and the second encompasses the dropout prediction model and the experimentation phase (Chapters 5 and 6).

Chapter 7 introduces a real-world risk-prediction application of the proposed model. Here, ill older people are monitored during their hospitalisation in intense care units. While we empirically show that our proposed method is superior to the state-of-the-art methods in SDP, we conclude that it has better performances than the others in a risk-prediction scenario where missing alarms have severe repercussions to the health of the patients.

Lastly, Chapter 8 closes the thesis with final observations about the challenging SDP task and how we provide a time-dependent deep learning methodology to solve it. Additionally, we give an important deduction for future researchers to focus on selecting significant evaluation metrics to demonstrate the effectiveness of a particular system w.r.t. the state-of-the-art. The contribution with a new dataset containing long-term e-degree time-series provides a new branch of student dropout prediction to stabilise new milestones in this promising research area.

Chapter 2

Related Work

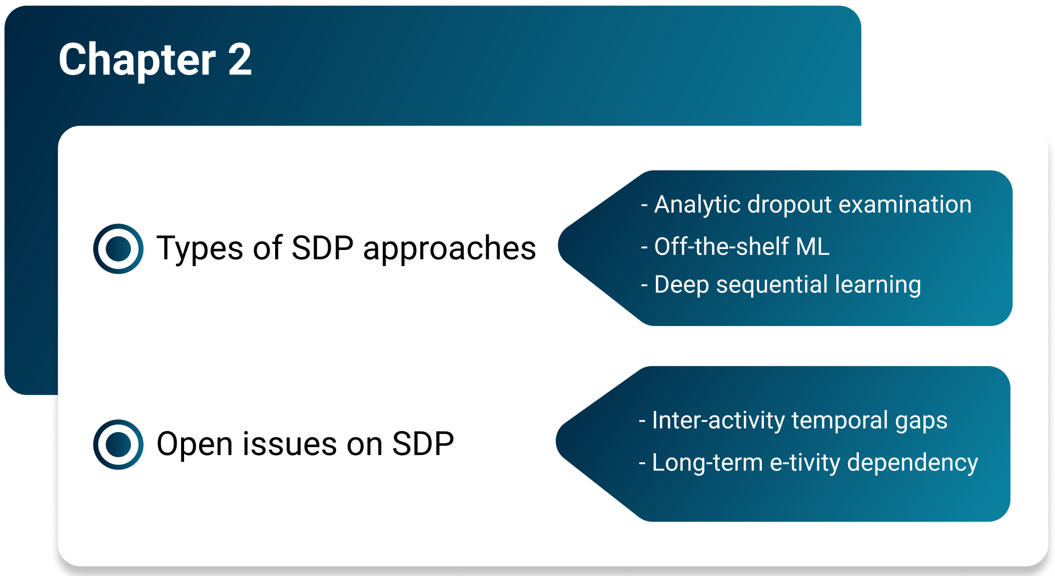

Chapter 2
Types of SDP approaches
- Analytic dropout examination
- Off-the-shelf ML
- Deep sequential learning
Open issues on SDP
- Inter-activity temporal gaps
- Long-term e-tivity dependency

Figure 2.1. Academia has approached the SDP since the diffusion of MOOCs by adopting various strategies. This chapter describes three types of SDP approaches (i.e. statistical analysis, simple ML, and deep learning strategies). Additionally, we provide future researchers with open issues in the dropout phenomenon that comprise temporal gaps between e-tivities and long-term dependencies.

Besides dividing the research area into two different simple categories, which encompass statistical analysis and off-the-shelf machine learning algorithms, we also study deep learning methods that consider the intrinsic temporal relationship between student e-tivities. Moreover, taking into consideration the lack of studies based on e-degrees, we tackle two important aspects neglected in research:

- *Inter-activity temporal gaps* - The contribution of inactive periods from students enrolled in MOOCs and those in e-degrees is different. Being MOOCs fast-paced and short-term courses, idle periods might indicate a potential dropout status. Whereas, because e-degrees are prolonged in time (e.g. from three to five years), inactive periods might not be as important.

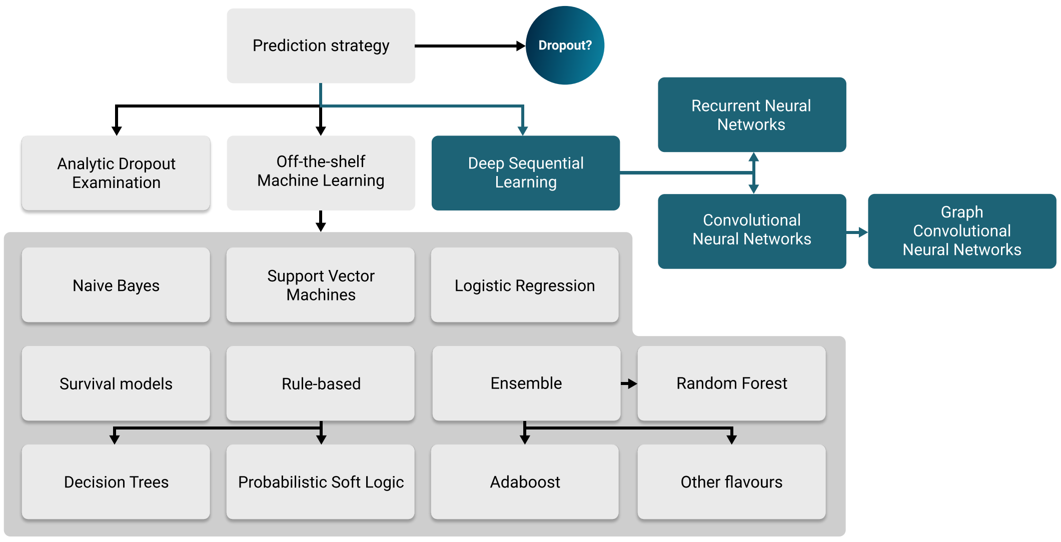


Figure 2.2. A taxonomy of prediction strategy approaches. The greyscale illustrates the strategy branch's complexity (i.e. the darker it is, the more sophisticated the models are).

- *Long-term e-tivity dependency* - E-tivities generated in a prerequisite course form an important context for students' progress in successive and more complex courses in an online degree. By employing sequential methodologies that share information in time, the e-tivity dependency might alter the dropout status. Meanwhile, this aspect of dependencies can be absent in MOOCs.

2.1 Types of SDP Strategies

During the three decades in which OLEs have been present, different artificial intelligence branches ranging from statistical analyses and naive rule-based systems to complex deep sequential methodologies resolved the SDP problem [108].

This section illustrates a taxonomy of the prediction strategies to address the SDP problem, as depicted in Figure 2.2. *Analytic dropout examination* (ref. Section 2.1.1) analyses the data and attempts to provide insight based on pure statistical metrics. *Off-the-shelf machine learning* (ref. Section 2.1.2) exploits machine learning algorithms. *Deep sequential learning* (ref. Section 2.1.3) handles strategies based on recurrent and convolutional neural networks. Because of the high performances of deep learning strategies when the data quantity increases[1], we discuss their peculiarities.

2.1.1 Analytic Dropout Examination

In this branch of dropout prediction, pure statistical methodologies are used to forecast a student's status. Analytical studies often collect data from various sources and then conduct correlation analyses between the extracted features and the dropout

[1]In SDP, we log every student interaction with the course platform. Therefore, we produce an excessive amount of information.

status. Additionally, because statistical approaches are highly interpretable, they provide additional insight into the selected features' distribution: the distribution of dropouts by assignment submission rate. Because the analytic examination is descriptive rather than predictive of future student status or e-tivity creation behaviour, academics and online institutions have used it as a tool for interpreting data in order to develop appropriate intervention methods. Additionally, analytic inspection approaches are insensitive to time constraints. As a result, they cannot guarantee sound conclusions, as dropout patterns alter over time.

Studies based on analytic dropout examination. The authors in [134] examine the Hellenic Open University's Informatics bachelor's degree (HOU). They gather computer science-related data in order to identify students who are at risk of dropping out. The information comprises computer literacy and previous computer usage training. Furthermore, they compile a dump of student responses to questionnaires and phone interviews on the reasons for their dropout decision. As a result, they examine dropout motivations for each student to develop rule-based intervention mechanisms to assist freshly enrolled students along their academic path. In particular, HOU permits students to repeat a failed course module - that is, if a course spans more than one semester, HOU breaks it into separate consecutive modules. As a result, the authors investigate four distinct student groups, only two of which are genuine dropouts. The authors perform association studies on the student profiles according to the two established dropout groups. They provide statistically significant information regarding the relationship between submitting assignments and past schooling and the decision to drop out. They conclude that dropout students overestimate the burden of learning while working by further analysing the reasons behind the dropout decision using data acquired from phone interviews. Furthermore, a small percentage of dropouts believe that their tutor did not ease them into the course by assisting them in understanding the material or completing their projects.

Unlike the previous study, the authors in [133] exploit an automatic method of hypotheses based on empirically gathered data. They use General Unary Hypotheses Automaton (GUHA) to generate data based on three initial parameters:

- *Confidence* depicts the probability that a generated hypothesis correctly classifies students as dropouts or not.
- *Support* is the minimum percentage of the hypothesis to fit the generated rule.
- *The maximum number of antecedents* corresponds to the number of literals occurring on the left part of the rule implication.

The authors demonstrate that failing to submit the penultimate assessment leads to a complete dropout decision for the course module at hand. Nevertheless, the insight does not hold when applied across different course modules or related courses. Hence, the varying nature of modules within the same online course[2] makes it hard for GUHA rules to hold in an inter-module (inter-course) scenario.

[2]The observation can be generalised for e-degrees in an online course where a module can be considered a prerequisite course.

2.1.2 Off-the-shelf Machine Learning

The bulk of the state-of-the-art methods address the SDP problem by relying on simple machine learning algorithms. The literature uses classic ML models because of the extensive support from modern programming languages and related frameworks (e.g. Python and its libraries fully support data engineering and machine learning predictions).

This section follows the same format as Section 2.1.1. In particular, for each type of model in Figure 2.2 under the *Off-the-shelf Machine Learning* category, we provide its formal description and subsequently enlist the literature studies that exploit it. Although research confronts several prediction models, we report only the best performing strategy (and combinations). Notice that we group methods based on Decision Trees and Probabilistic Soft Logic under the *Rule-based* class. All the models above identify rules which help to classify new instances. Instead, we situate methods based on Random Forests and AdaBoost under the *Ensemble* category.

Logistic Regression

It models the relationship between two variables by fitting a linear equation to the data $x_1, ..., x_n$ to predict an outcome y [14, 129]. The literature denotes the variable to predict y as the dependent variable and the input data features as independent variables. Unlike the analogous linear regression model, the dependent variable y can be categorical or continuous. The model does not require strictly continuous data, and, for the sake of simplicity, we assume that y is binary. Moreover, the logistic regression uses a log odds ratio rather than probabilities and an iterative maximum likelihood method rather than the least squares to fit the final model. The log odds ratio helps researchers use logistic regression since it is more appropriate for non-normally distributed data or when the samples have unequal covariance matrices.

The logistic function is a sigmoid function, which takes any real input y, and outputs a value in [0,1]. For the logit, this is interpreted as taking input log-odds and having an output probability. The literature defines the standard logistic function $\sigma : \mathbb{R} \to (0, 1)$ as follows:

$$\sigma(y) = \frac{e^y}{e^y + 1} = \frac{1}{1 + e^{-y}} \tag{2.1}$$

Assuming that y is a linear function of a single independent variable x, we can express y as follows:

$$y = \beta_0 + \beta_1 x \tag{2.2}$$

According to Equation 2.2, we can write the general logistic function $p : \mathbb{R} \to (0, 1)$ as follows:

$$p(x) = \sigma(y) = \frac{1}{1 + e^{-\beta_0 - \beta_1 x}} \tag{2.3}$$

Here, $p(x)$ can be interpreted as the probability of the dependent variable y being a positive case (i.e. dropout student) or a negative one (i.e. persisting student).

Works based on Logistic Regression - The authors in [42] use decision trees to select features corresponding to maximum information gain. The authors split courses into weekly phasic views. Subsequently, they use a logistic regression model to distinguish between dropouts and persisters. LR-SEQ and LR-SIM, sequentially and simultaneously smoothed logistic regression, respectively, exploit transfer learning to use the previous week's knowledge to help learn smoother probabilities for the current week of the student learning behaviour [53]. Together with data derived from user e-tivities based on clicks, this work utilises homework submissions and grades. In detail, they compute statistics at the end of a particular phase (e.g. average attempts on each assignment done by week i) to train their weekly-based model.

Furthermore, LR-SIM improves LR-SEQ because, in the latter, early inaccurate predictions cannot benefit from the knowledge learned in subsequent weeks, hence sabotaging later models. Therefore, LR-SIM learns models for all weeks simultaneously. The work proposed in [113], jointly with the student responses to ad-hoc questionnaires for several HarvardX courses, uses general student demographic statistics. Here, the authors exploit a lasso-regularised variant of the logistic regression, which operates as a penalty for complexity whose size the authors determine empirically. Logistic regression relies on two temporal concepts to structure their prediction strategy (i.e. lead and lag) to predict the dropout status of students [127]. Lead represents how many weeks in advance we predict dropout; whereas, lag represents how many weeks of historical data the classifier has available to make the prediction. For instance, when considering a lead of five and a lag of three, the first three weeks of data are used to predict the upcoming five weeks. Therefore, the data corresponding to the first three weeks of a course becomes the training set and the dropout value for the eighth week becomes the label to predict.

Support Vector Machines (SVM)

When performing a binary data classification task, the goal is to assign one of the two classes to a new data point, given that we begin with a set of instances in a particular vector space $\mathbb{R}^n$. The authors in [26] represent each data point x as a p-dimensional vector. Hence, the classification goal is to know whether we can separate the instances with a (p-1)-dimensional hyperplane. Although there are many ways to divide the data points into two classes, SVM aims to learn the hyperplane that maximises the separation. In other words, the hyperplane is considered an optimal solution if the distance from it to the nearest data point of both classes is maximal.

Suppose that we have a dataset of n points $(x_1, y_1), ..., (x_n, y_n)$ where $y_i \in \{-1, +1\}$ and $x_i = \{f_1, ..., f_m\}$ s.t. $f_j \in \mathbb{R}\ \forall i \in [1, n] \ \wedge \ j \in [1, m]$. Any class-separating hyperplane can be written as the set of points that satisfy the equation $w^T x - b = 0$ where w is the normal vector to the hyperplane. Let us distinguish between two scenarios of the data. Figure 2.3 depicts an example of SVM where the dimensionality of each data point $x_i \in \mathbb{R}^2\ \forall i \in [1, n]$ and, more importantly, the data is linearly separable. However, there are cases where the data is nonlinearly separable.

In the first case, as shown in the Figure, we can select two parallel hyperplanes that separate the data. We call *margin* the region within these two hyperplanes. Moreover, the maximum-margin hyperplane is the hyperplane that divides the margin into two halves. Generally, assuming that the dataset is normalised and standardised, the hyperplanes can be described by $w^T x - b = 1$ and $w^T x - b = -1$,

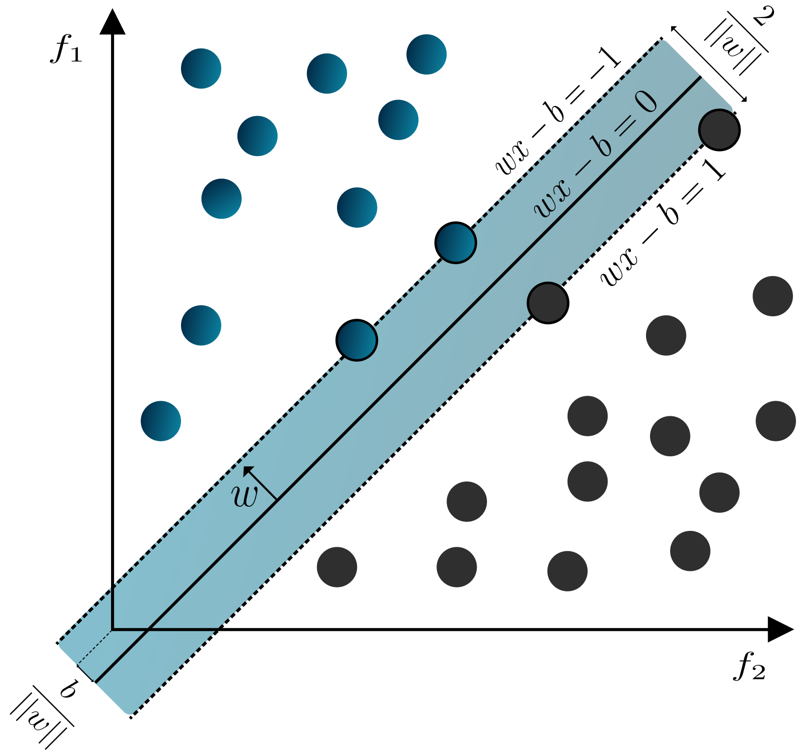


Figure 2.3. An example of the maximum-margin hyperplane on a binary classification problem. The data points (circles) on the dotted lines are denoted as support vectors; meanwhile, the line separating the instances is SVM's solution. Besides, notice that the distance between the support vectors is $\frac{2}{||w||}$ and that the distance of the linear function from the origin of the data vector space - in this scenario $\mathbb{R}^2$ - is $\frac{b}{||w||}$.

respectively. Because the distance between the hyperplanes is $\frac{2}{||w||}$, we need to minimise $||w||$ for the distance to be maximal. Finally, we need to prevent data points from being inside the margin. Hence, for each $i \in [1, n]$ we add the following constraints:

- $w^T x_i - b \geq 1$ if $y_i = 1$;
- $w^T x_i - b \leq -1$ if $y_i = -1$;

The optimisation problem minimises $||w||$ subject to $y_i(w^T x_i - b) \geq 1 \; \forall i \in [1, n]$, which is the merged version of the previous two constraints. The solution to the optimisation problem depends from the data points x_i (also called *support vectors*) that lie nearest to the max-margin hyperplane.

In the second case, as happens with real-life applications, data is not linearly separable. Therefore, the authors in [13] applied the kernel trick and transformed the input data points from a source to a target vector space in which the points are linearly separable. The proposed algorithm is similar to the original SVM, but nonlinear kernel functions replace the dot products. It allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be nonlinear in the transformed space high-dimensional. In other words, although the classifier is a hyperplane in the transformed feature space, it may be nonlinear in the original input space. Some of the most famous kernel functions are the polynomial function, the Gaussian radial basis function (RBF) [119], and the hyperbolic tangent.

Works based on SVM - The authors in [72] trained SVM with a polynomial kernel function based on page-view and video-view logs aggregated (e.g. summing or averaging) week by week. They extract numerical features by capturing the user activity levels, such as the number of requests and technical features coming from the browser information. Moreover, the authors state that historical features from the previous week are useful only until a certain point in time, after which the contribution of the e-tivities fades away. Similarly, a specific and a general case dropout SVM model has been trained relying on the RBF kernel [3]. Here, the authors distinguish between dropout and inactive student cases. In detail, dropouts comprise students' set without any generated e-tivities during the entire time-span of the online course. Whereas inactive students are those students that attend the course, but their sequence of e-tivities is parsimonious. The authors use quiz- and activity-related information where, for each week, they transform the e-tivities of a student in a string identified with letters corresponding to the type of e-tivities.

Naive Bayes classifier

Naive Bayes classifiers [48] are a family of simple probabilistic classifiers based on Bayes' theorem. In detail, they assign probabilities of class labels $p(C_k|f_1, ..., f_m)$, drawn from a finite set $\{C_1, ..., C_k, ..., C_K\}$, to the input instances $x_i = \{f_1, ..., f_m\}$ $\forall i \in [1, n]$ assuming that all features f_j $\forall j \in [1, m]$ are independent and identically distributed (iid) variables given the class variable. The problem with $p(C_k|x_i = \{f_1, ..., f_m\})$ is that if m is large or if a feature f_j can take on a large number of values, then its calculation becomes infeasible. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k|x_i) = \frac{p(C_k) \times p(x_i|C_k)}{p(x_i)}.$$

Because the denominator of the fraction does not depend on C_k, then we can eliminate it and calculate only the joint probability $p(C_k) \times p(x_i|C_k) = p(C_k \cap x_i) = p(C_k \cap \bigcap_{j=1}^{m} f_j)$[3]. Using the chain rule, we can exploit the rule of conditional probability to write the previous equation as follows:

$$\begin{aligned} p(C_k) \times p(x_i|C_k) &= p(f_1, ..., f_m, C_k) \\ &= p(f_1|f_2, \dots, f_m, C_k) \times p(f_2, \dots, f_m, C_k) \\ &= p(f_1|f_2, \dots, f_m, C_k) \times p(f_2|f_2, \dots, f_m, C_k) \times p(f_3, \dots, f_m, C_k) \\ &= p(f_1|f_2, \dots, f_m, C_k) \times ... \times p(f_{m-1}|f_m, C_k) \times p(f_m|C_k) \times p(C_k) \end{aligned}$$

[3]For readability purposes, we replace the joint probability symbol $\cap$ with a comma.

We can exploit the iid assumption to write $p(f_j|f_{j+1},\dots,f_m,C_k) = p(f_j|C_k)\ \forall j \in [1, m-1]$. Hence, the joint model written above can be expressed as follows:

$$\begin{aligned} p(f_1,\dots,f_m,C_k) &\propto p(C_k,f_1,\dots,f_m) \\ &\propto p(C_k) \times p(f_1|C_k) \times p(f_2|C_k) \times ... \times p(f_m|C_k) \\ &\propto p(C_k) \times \prod_{j=1}^{m} p(f_j|C_k) \end{aligned}$$

where $\propto$ denotes proportionality.

Now, the Naive Bayes classifier combines the previous probability calculation in a rule-based system. A common approach is choosing the most probable hypothesis, the maximum a posteriori (MAP) decision rule. More formally, the MAP decision is a function as follows:

$$\hat{y} = \underset{k\in\{1,\dots,K\}}{\operatorname{argmax}}\ p(C_k) \times \prod_{j=1}^{m} p(f_j|C_k)$$

Works based on Naive Bayes - By combining demographic data with information derived from the aggregation of student e-tivities (e.g. number of visits, number of sessions, percentage usage of each of the learning resources), the authors in [41] employ several rule-based decision trees algorithms. However, they conclude that the most accurate model is the Naive Bayes classifier. Similarly, the Naive Bayes classifier is the best performing and statistically most significant classification model in [76]. This classifier works best in all the dataset variants the authors have considered (i.e. demographics data only and demographics together with the first homework results). The authors in [91] propose WAVE, a monitoring architecture for student academic progress. Alongside other student-performance metrics, WAVE exploits exam information (e.g. GPA) to determine whether a student attending the first semester persists in the second one. The authors rely on the Naive Bayes classifier because of its superior performances according to their testbeds and its time efficiency in training and predicting the dropout status of new students. Similarly, the authors in [77] perform predictions via a Naive Bayes classifier on input data from student demographic information and their academic performance in the Computer Science course offered at HOU. Lastly, the authors in [84] consider behavioural patterns such as accessing, viewing and closing course resources according to their time of occurrence in a certain course phase. To resolve the SDP problem, they exploit n Naive Bayes classifier to train a multi-view semi-supervised learning (SSL) architecture [20]. They follow two steps in multi-training. First, they train the components on a subset of the dataset. Afterwards, they transfer the unlabelled examples, which have high confidence upon prediction from the n classifiers, into the labelled set of instances. This two-phase process continues until there are no more unlabelled data.

Decision Trees

In data mining, a decision tree is a predictive model representing both regression and classifier models. When considering classification problems, a decision tree is called a classification tree; we use the former nomenclature throughout this work (often, we use the shorthand DTree). According to Rokach et al. [114], a decision tree is a classifier delineated as a recursive partition of the feature space. It consists of nodes that, but the root node, have exactly one incoming edge. Nodes are divided into two categories: internal nodes, which split the feature space into two or more sub-spaces, and leaves are also called deciders. Moreover, each leaf corresponds to a single class which represents the most appropriate outcome label. An instance is classified by tracking its features throughout the tree's nodes down to the leaf nodes. Specifically, the search starts from the root, and, according to the features observed, a branch is chosen. Subsequently, the node corresponding to the previously selected branch is chosen, and the same procedure of electing a branch occurs. This goes on until a terminal node (leaf) is reached.

The literature has proposed various algorithms for generating decision trees. Generally, algorithms that build decision trees work in a top-down fashion where they choose a feature value that best splits the feature space at each step. Different algorithms use different metrics to select the best split of the feature space into two sub-spaces. Here, we provide a brief description of each tree-generation algorithm with its corresponding splitting metric:

- *CART* [16] exploits the Gini impurity score, which measures the probability of misclassifying an instance. More formally, let K be the number of different class labels $\{C_1, ..., C_K\}$ and $p(k)$ be the probability of getting any data point $x = (f_1, ...f_m)$ with class C_k. The Gini impurity is calculated as $G = \sum_{k=1}^{K} p(k) \times (1 - p(k))$. A Gini impurity of 0 is the lowest and best possible impurity. It can only be achieved when every input example has the same class.

- *ID3*, *C4.5*, and *C5.0* [111] use the Information Gain[4] (IG) measurement based on the concepts of entropy and information content. First, the entropy of a particular target feature T is expressed as $H(T)^5 = -\sum_{k=1}^{K} p(k) \times log_b p(k)$ where, as previously mentioned, $p(k)$ is the probability of getting any data point $x = (f_1, ..., f_m)$ with class C_k from the set of class labels $\{C_1, ..., C_K\}$. Now, the information gain of T according to the splitting feature f_i is expressed as follows
$$IG(T, f_i) = H(T) - \sum_{v \in f_i} \frac{|T^{f_i=v}|}{|T|} \times H(T^{f_i=v})$$
where v are the values that feature f_i can assume, and $T^{f_i=v}$ are the instances in the input dataset that have v assigned as value of f_i.

- *CHAID* [65] is a decision tree generation algorithm that, depending on the continuity of the target variable, uses either the Chi-Square (categorical case) or the F (continuous case) test to determine the next optimal split. Because the dropout phenomenon has been modelled as a binary classification problem (i.e. 0s are persisting student and 1s are dropouts), the Chi-Square is used. We refer the reader to [102] for more details on the Pearson Chi-Square test.

[4]We refer the reader to [80] for more information on the conditional expected value of the Kullback-Leibler divergence as a synonym of IG in decision trees.

[5]The base b of the logarithm is usually set to 2 since the information gain is measured in bits.

Table 2.1. Representation of the cost matrix for cost-sensitive learning.

	classified as negative	classified as positive
actual negative	$C(-;-) = 0$	$C(-;+) = 1$
actual positive	$C(+;-) = 4$	$C(+;+) = 0$

Works based on Decision Trees - The authors in [1] use previous education information and demographic data to shape their two decision tree models. In detail, they test ID3 and C4.5, where the first performs better when executing a hold-one-out experiment, whereas the second outperforms the other in a 10-fold cross-validation method. By completing an exhaustive study of the dropout phenomenon on exploring socio-demographic and study-environment variables, Kovavcic [78] exploits CHAID and CART. To distinguish between different misclassification types on dropout labels according to data related to previous studies, Dekker et al. [30] boost the accuracy of CART via cost-sensitive learning. They use a cost matrix, shown in Table 2.1, as input to the CART meta-classifier whose goal is to increase the weight of false negatives over false positives. The authors in [73] adopt an SSL approach over the C4.5 algorithm. They use the Tri-Training [138] strategy to produce the best performing variant for their dataset. Lastly, the work presented in [95] confronts interpretable methods such as decision trees against non-interpretable strategies (e.g. random forests and gradient boosted trees). The chosen model has the best performance and the greatest interpretability. Therefore, decision trees emerge as the model satisfying both criteria. Ortigosa et al. [99] exploit the C5.0 decision tree variant algorithm using data from more than 11,000 students collected over five years in a Spanish OLE. The authors stretch themselves to produce more meaningful results to provide some interpretability criterion for the end-user (the OLE staff). They divide first-year students from successive-year students because the former have a higher dropout rate. Thus, they consider these two types of students as coming from two distinct populations with different problems. Besides outputting a binary outcome of the student dropout status, the authors also provide risk scores to prioritise support for those students standing at the high-end of the risk area. Although they split the academic year into ten separate periods from September to June, they train different C5.0 models for each period. The information passed from one period to the other is limited, and models trained with data further in time are agnostic of the models' outcomes with data in earlier periods.

Similarly, Isidro et al. [63] use a decision tree classification to distinguish between persisters and dropouts. They study the dropout phenomenon according to two time windows: i.e. global and weekly student trajectory view. Both the decision tree with the entropy and the Gini splitting function outperforms the other compared methods.

Probabilistic Soft Logic

Probabilistic Soft Logic (PSL) is a framework for collective and probabilistic reasoning in relational domains. PSL uses first-order logic rules over random variables with soft truth values from the interval [0,1]. A PSL program has a set of first-order logic (FOL) rules with conjunctive bodies and single literals. Rules are labelled with non-negative weights. While PSL shares the syntax of its rules with FOL, it uses soft truth values from the interval $[0,1]$ instead of the true/false binary labels. Given a set of atoms $l = \{l_1, ..., l_n\}$, the mapping $I : l \rightarrow [0,1]^n$ from atoms to soft truth values is called an interpretation. Hence, PSL defines a probability distribution over interpretations that satisfy more ground-rule instances more probable. For more details on how the rule satisfaction and the inference and learning method in PSL work, we point the reader to [68].

Works based on Probabilistic Soft Logic - By exploiting forum intervention data incorporated with clickstream information, the authors in [112] use the modelisation schema described above to construct relevant PSL rules using logical connectives. The produced model associates these rules with student survival. The authors build two PSL models to resolve the SDP problem. The first - denoted as *direct* - infers the student survival solely from observable features; whereas, the second - denoted as *latent* - infers the survival as a hidden variable. The rules produced vary from one model to the other. In more detail, the direct model exploits one or more observable behaviour features to predict student survival. Hence, observable features directly imply survival. Contrarily, the latent model includes hidden variables based on patterns of student engagement. Because these variables cannot be directly measured, the authors treat student engagement as a latent variable by associating observable variables (features) to a certain engagement form. The authors demonstrate that the latent model outperforms the direct one.

Neural Networks

Although the architectures of neural networks have been expanded in recent years, we provide the reader with the theory of simple and deep networks (i.e. multilayered architectures) as inspired by [51]. An artificial neural network (ANN), often shortened to a neural network (NN), comprises simple processing units called neurons. A neuron is the fundamental processing unit in a neural network, and it consists of four elements, as depicted in Figure 2.4.

- A set of weighted connecting links (synapses). The input signal $x_k = (f_1, ..., f_m)$ $k \in [1, n]$ when transmitting is multiplied by its corresponding weight (real number) in the connection link $w_{j,k}$ $\forall j \in [1, m]$.
- A linear combiner $\sum_{j=1}^{m} w_{j,k} \times f_j$ which sums the weighted input.
- A bias term b_k can be applied to the final summing junction. This element is optional, thus can be set to 0.
- A differentiable activation function Φ (i.e. squashing function) which is applied to the output of the neuron. It limits the output of the neural network. The most used activation functions in literature are sigmoid [47], hyperbolic tangent, and rectified linear unit [96].

The simplest form of a neural network is a single feedforward network (perceptron). This type of network has three layers: input, hidden and output. Generally, the input and output layers are not counted; thus, this network is a single layer network.

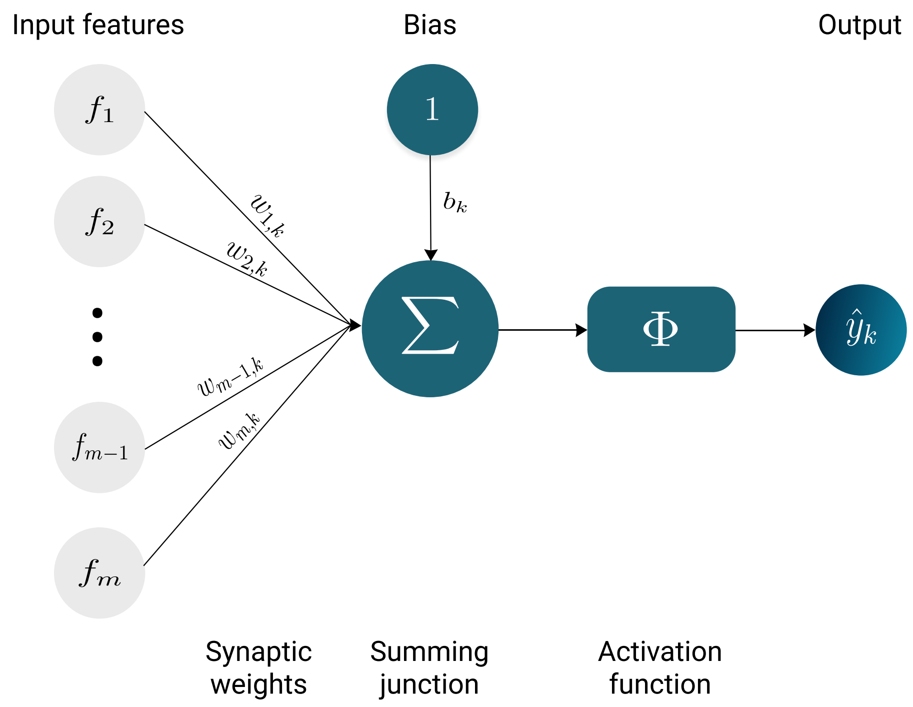


Figure 2.4. The basic elements of a neuron.

A generalisation of the single feedforward is a multilayer feedforward network (multilayer perceptron). In this case, there is more than one hidden layer. The neurons belonging to the hidden layers - denoted as hidden neurons - can acquire more global information and solve complex tasks. The input signals of a layer consist of the output signals of the preceding layer only. Additionally, all layers are fully connected.

A neural network is learned by adjusting the selected activation function's weights, edges, and parameters. The algorithm used for training is backpropagation [54]. The weights are randomly initialised between a specific range of values. In a supervised learning environment, when predicting the outcome label of a certain training instance, the neural network engenders an error: the difference between the actual label y_k and the predicted one $\hat{y}_k$. The loss function (continuously differentiable) is the sum of squares of the generated errors $\mathcal{L} = \sum_i (\hat{y}_k - y_k)^2$. The backpropagation algorithm optimises the weights of the network. Hence, the network learns how to assign inputs to outputs by minimising the loss function at each training step. Specifically, backpropagation uses the error values to calculate the loss function's gradient to find the minimum. This is the reason why the activation function must be differentiable to update the weights. The weights of a neural network are analogous to the coefficients in a linear regression model. However, because the number of weights compared to the number of coefficients in a regression model is high, it is difficult to interpret the weights in a neural network.

Works based on Neural Networks - Chen et al. [21] base their work on a novel combination of decision trees with an extreme learning machine (ELM). They transform results obtained from the decision tree employed into a neural network.

They perform feature selection using decision trees based on maximal information gain and weigh them based on their impact on the leaf nodes. Because the root node has the best classification ability to be connected to all nodes, it has the most significant impact. They call this procedure the *enhancement layer*. The subsequent layer (i.e. the *mapping layer*) transforms the decision tree into an ELM, a single hidden layer neural network. The input neurons are the root and internal nodes of the tree, and the hidden neurons correspond to the leaf nodes. They connect an input neuron to the hidden neurons if it impacts its corresponding leaf in the tree. Therefore, they link the input neuron to the root node to each hidden neuron, while other input neurons might be connected only to some hidden neurons. Finally, the authors set to zero the weights of a connectionless neuron pair.

Ensembles

Instead of relying on the predictive power of a single model, an ensemble strategy combines more models to obtain better results [98]. The models that build the ensemble method are denoted as components. Notice that the components can be heterogeneous: for instance, assuming that we have an ensemble of two components, one might be a neural network and the other a decision tree. Each component is trained separately. When predicting the outcome label of an instance x_i, each component j gives its prediction $\hat{y}_{j,i}$ and a consensus method is employed to select the overall prediction label $\hat{y}_i$. The simplest way of merging the different $\hat{y}_{j,i}$ is by selecting the label that the components have produced the most. This is also known as the majority voting consensus. The most popular ensembles consist of bagging (see Figure 2.5) and boosting methods (see Figure 2.6). The former engages in training each classifier on a random redistribution of the training set. The classifiers' training set is generated by randomly selecting as many examples as with replacement in the original training set. Contrarily, in the boosting strategy, each classifier's training set is based on the performance of the previous classifiers. Examples incorrectly predicted by previous classifiers occur more often in the training set of the current component. Hence, boosting produces new classifiers specialising in predicting the outcome label of those difficult to classify instances. We have identified Random Forests[6] and AdaBoost as the primary ensemble methods in the literature. Nevertheless, some works also concentrate on alternative and custom made ensemble methods.

Random Forests

According to Breiman et al. [15], random forests are a combination of decision trees such that each one depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. In other words, random forests are a technique of bagging with bootstrap samples. Specifically, a random forest is a classifier consisting of a collection of tree-structured classifiers where each tree casts a unit vote for the most popular class at a particular input x. In a random forest, each tree is fully grown, meaning that pruning methods are not employed. Although different studies [94, 101] sustain that pruning affects

[6]Random Forests can be considered as a bagging method with a slight tweak.

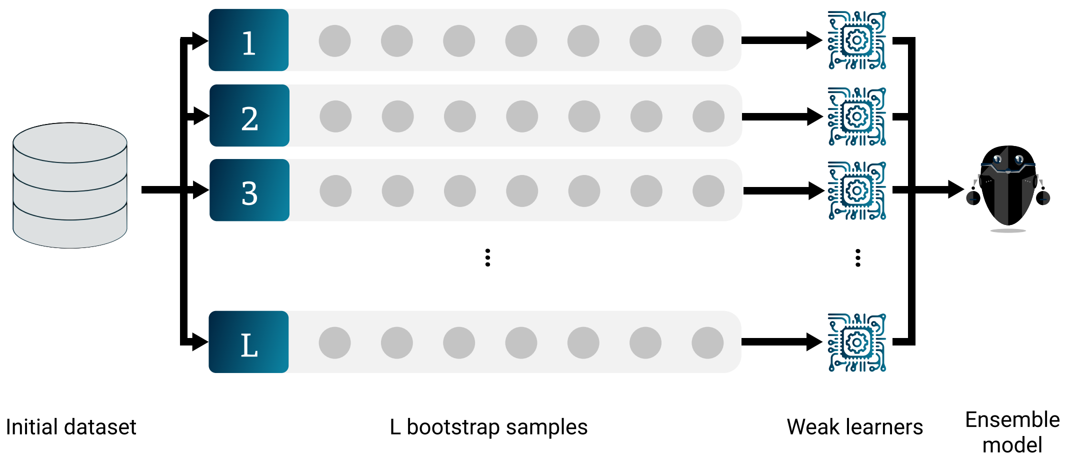


Figure 2.5. An example of bagging (with bootstrap) ensemble learning. Beginning from the initial dataset, we create L bootstrap samples so that each of them acts as another dataset whose instances are drawn independently with the same probability. Then, we can fit a weak learner for each of the sample datasets and finally aggregate them to obtain an ensemble model with less variance that its components.

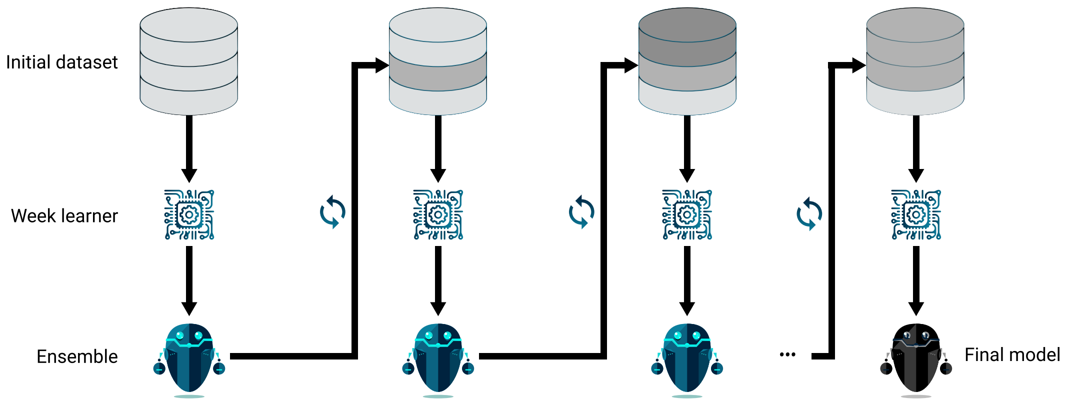


Figure 2.6. An example of boosting ensemble learning. Boosting consists in, iteratively, fitting a weak learner, aggregate it to the ensemble model and update the training dataset to better take into account the strengths and weakness of the current ensemble model when fitting the next base model. In other words, each model in the sequence is fitted giving more importance to observations in the dataset that were badly handled by the previous models in the sequence. The colour of each strip in the dataset indicates which instance the week learners need to focus on in each iteration (i.e. the darker the shade the more important that instance is in the current iteration).

performances, Breiman [15] suggests that the generalisation error always converges even without pruning the tree. The user selects the number of features to consider at each node and the number of trees to grow. Therefore, at each node, only the selected features are searched for the best split. In testing, each new instance is passed down to each of the L trees. The forest chooses a class having the most out of L votes for that particular instance [100]. Although random forests generally perform better than a simple decision tree, their classification decision cannot be interpreted. Because interpretability is important in the dropout scenario since tutors and course administrators want to devise intervention strategies, the literature proposes compression techniques. Among these solutions, we can transform a random forest into a *born-again* decision tree that reproduces the same decision function [89, 130].

Works based on Random Forests - Gray et al. [45] model a student as a set of η boolean observations indicating the attendance of a student at each point in time: i.e. $z = \{z_1, ..., z_\eta\}$. They utilise an ad-hoc function that combines attendances and non-attendances of a student from the η observations to calculate engagement behaviour of a student: i.e. $BEM_\eta = \sum_{i=1}^{\eta} (-1)^{1-z_i}$. They choose the BEM scores for the first three weeks and demographic features to aid the information gain of their model. Next, the authors use random forests, which rely on the degree program as the discriminatory cohort and the BEM scores to determine the dropout cases. Haiyang et al. [46] generate three time-series based on forum, video-lecture and textual resource clicks. Afterwards, they sum up the number of clicks from each module every day and align students' total clicks. They exploit Time-series Forests (TSF) [59], which employs a combination of entropy gain and distance measure to evaluate the different splits. Besides capturing important temporal characteristics, it reveals which course module (or period) most affects the learning progress of a particular student s, thus enabling them to determine the exact time interval in which s drops out.

AdaBoost

Adaboost is the first practical boosting algorithm [39] as illustrated in Figure 2.6. Boosting is a machine learning approach that creates a highly accurate model by combining many weak and naive models. For AdaBoost to be effective, its component classifiers must be simple with low training error. Given l training examples $(x_1, y_1), ..., (x_l, y_l)$ where $x_i \in R^n$ and $y_i \in \{-1, +1\}$. For each component $j = 1, ..., N$, a distribution D_j is computed over the l training examples. Thus, a weak learner is applied to find a weak hypothesis $h_j : R^n \rightarrow \{-1, +1\}$ with a low error ϵ_j over D_j. The final hypothesis computes the sign of the weighted combination of the weak hypotheses $\sum_{j=1}^{N} \alpha_j h_j(x)$. This signifies that the final hypothesis performs a majority vote from the weak hypothesis according to the weights α_j. Following the presentation of [117], we provide the pseudocode of the AdaBoost algorithm.

Works based on AdaBoost - Berens et al. [12] combine the prediction powers of linear regression, neural networks and random forests to distinguish at-risk students and persisters. The authors use a simple multivariate model for the linear regression $y_{s,j} = \beta_0 + \beta_1 \vec{d_s} + \beta_2 z_{s,j} + \epsilon_{s,j}$ where s and j denote the students and the semester,

```
Data: $(x_1, y_1), ..., (x_l, y_l)$
Initialise: $D_1(i) = \frac{1}{l}\ \forall i \in [1, l]$;
for $j = 1, ..., N$ do
    Train weak learner using distribution $D_j$;
    Get weak hypothesis $h_j : R^n \to \{-1, +1\}$;
    Select $h_j$ with low weighted error $\epsilon_j = P_{i \sim D_j}[h_j(x_i) \neq y_i]$;
    Let $\alpha_j = \frac{1}{2}ln(\frac{1-\epsilon_j}{\epsilon_j})$;
    for $i = 1, ..., l$ do
        Update $D_{j+1}(i) = \frac{D_j(i) \times e^{-\alpha_j y_i h_j(x_i)}}{Z_j}$;
        where $Z_j$ is a normalisation factor
    end
end
Result: $H(x) = sgn(\sum_{j=1}^{N} \alpha_j h_j(x))$
```

Algorithm 1: The pseudocode of AdaBoost algorithm

respectively, and $\vec{d_s}$ is the vector comprising demographic data. In this equation $z_{s,j}$ denotes the time-varying performance data of student s in semester j. The neural network component comprises three layers (including the input) with a final sigmoid output layer. The input layer has thirty-one neurons. The first hidden fully connected layer has sixteen, whereas the second fully connected has eight. The hidden layer weights are randomly initialised in the range $[-1, +1]$. The activation of each layer is the sigmoid function. The last component is a random forest because decision trees tend to overfit data being a non-parametric ML technique. Hu et al. [61] use two variants of boosting with decision trees. In other words, the authors rely on the CART and C4.5 algorithms to build the components of the boosting algorithm. According to the experimentation phase, the authors use the CART base algorithm because it has a lower false-positive rate.

Other ensemble methods

Kotsiantis et al. [75] use an ensemble of WINNOW [85], 1-Nearest Neighbour and Naive Bayes. WINNOW is a linear online algorithm similar to a perceptron. The only difference stands on the weight updating rule. In WINNOW, the weights of relevant features increase exponentially; meanwhile, those of irrelevant features decrease exponentially. On the other hand, 1-Nearest Neighbour assigns an instance to the class of its closest neighbour in the feature space.

Lykourentzou et al. [88] use a neural network, SVM, and a probabilistic ensemble simplified fuzzy ARTMAP (PESFAM) [86] as the components of the ensemble model. Having covered enough content about the first two strategies, we provide the reader with information on how a PESFAM works. PESFAM combines simplified fuzzy ARTMAP (SFAM) modules whose merging policy is the probabilistic plurality voting strategy. Inspired by the PESFAM illustration in [88], we delineate in Figure 2.7 the same architecture comprised of n SFAM components in the student dropout scenario. The left part of the Figure depicts the PESFAM ensemble. As with other ensembles, the dataset instances are randomly selected with replacement to generate

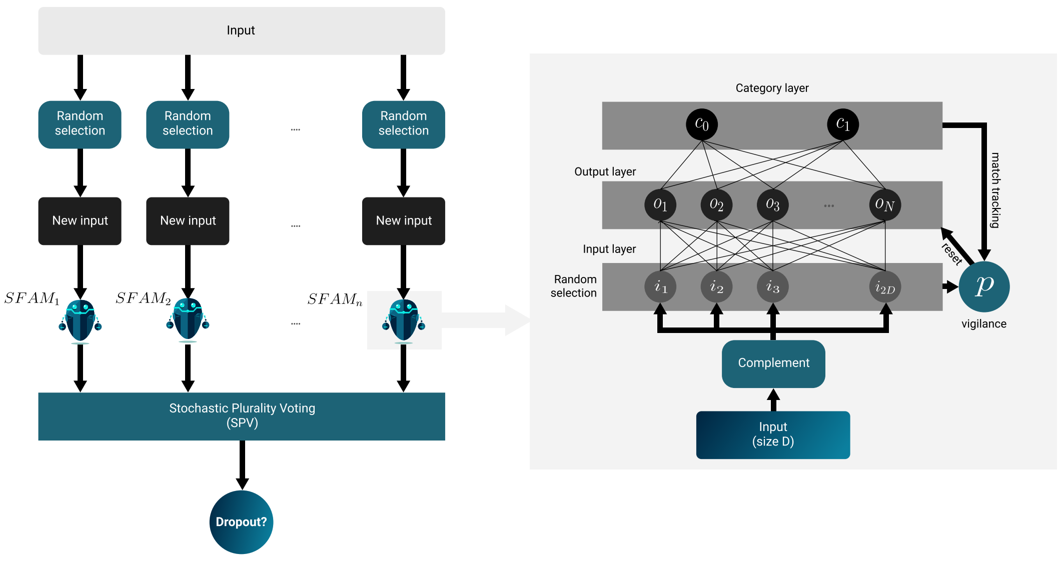


Figure 2.7. The PESFAM architecture composed of n components.

a new dataset represented the *New input* rectangles. They represent different "views" of the original input dataset. Each component then produces prediction labels for each input instance. A majority voting strategy stabilises the final labels for the instances. The right part of the Figure represents the detailed view of an SFAM module of the ensemble. An SFAM is a three-layered neural network: input, output, category. There is a connection between the input nodes and those in the output layer. Every output layer node points to a single category node, depicting one of the two classes (dropout or not). For further details about the training and testing phases of an SFAM network, we refer the reader to [66, 88].

Survival Models

As described in [64], survival analysis consists of tracing and examining time-to-event data. For instance, in the SDP scenario, a student might be monitored until the event of their preemptive dropout. More formally, let $T \geq 0$ be a random variable representing the survival time. The survival function is the probability that an individual survives beyond time t:[7]

$$S(t) = p(T > t),\ 0 < t < \infty$$

Additionally, in survival models, not all trajectories need to share the exact starting point (e.g. $t = 0$). Because students can enrol in a certain course, especially MOOCs, having a model that generalises the start time of the time-series is of utter importance. Besides being robust to starting times' heterogeneity, survival analysis permits input trajectories to have different durations. This characteristic reflects the time series in the dropout scenario because two distinct students have different behaviours, thus elongating/shrinking their studies' duration (e.g. time to graduation from an

[7]In SDP, the time t might vary according to the needs of the system. In other words, t might have several meanings such as next semester, next week, or next course module.

e-degree, or time to complete an online course). Having misaligned input trajectories, some students in the dataset do not experience the dropout event within the chosen time frame (i.e. their survival time is longer than t). Hence, survival models allow us to *censor* these temporal series. In other words, censorship enables us to measure "lifetimes" for a population that has not experienced the chosen event by time t. We refer the reader to [58] for the different censorship strategies employed in the literature. Notwithstanding the utility of the survival function, the literature often relies on the Hazard Function to build a prediction model. This function is defined as the probability that the student experiences an event within a short time interval if they survived until the beginning of that interval [128]. In other words, it represents the risk of experiencing the event of interest at time t.

To include the most used survival technique in the literature, we provide a brief description of the Cox Proportional Hazards Regression model [27]. It considers the effect of several variables and examines the relationship of the survival distribution on these variables. It is similar to Multiple Regression Analysis, but the difference is that the dependent variable is the Hazard Function at a given time t. It is based on very small intervals of time, called time-clicks, which contains at most one event of interest. For more detail[8] on how this model works, we refer the reader to [49].

Works based on Survival Models - Gitinabard et al. [42] consider features based on student interactions in course fora. However, they do not employ the intrinsic structural and temporal aspects of forum-based characteristics. They perform an initial survival analysis that finds that both behavioural and social features have a high hazard ratio, meaning that the lack of interaction in fora induces students to feel discouraged and drop out of MOOCs. Because survival models provide less biased estimates than least-squares linear regression, Yang et al. [136] analyse the determinant dropout factors with the highest hazard rate at the end of every course module. They exploit the hazard rate to describe the impact of the input features on student attrition[9]. Lastly, it is worth mentioning several studies that, although concerned with physical rather than online degrees, employ a modelisation strategy similar to those adopted for online courses/degrees [2, 23]. Ameri et al. [2] use demographics and pre-enrolment information. Additionally, they collect semester-wise attributes (e.g. GPA, credits passed/failed/dropped) from study-related data such as exams and other curricular activities. Chen et al. [23] exploit the same information as the previous work, but they use a lower quantity of features as input to their forecasting method (i.e. thirteen [23] against thirty-one [2]). Both [2] and [23] use Cox proportional hazard regression as their survival model to predict dropout labels in their degrees.

2.1.3 Deep Sequential Learning

With the advent of deep learning, the dropout research area has benefited with multiple proposals of sophisticated strategies to solve the SDP problem. We can adopt deep learning approaches because of the size of available datasets, and because some

[8]This model has been supported via a programmatic framework available at `https://lifelines.readthedocs.io/en/latest/`

[9]Here, the authors use the concept of attrition that represents a significant drop in student interactions. Notice that attrition is a general case of dropout.

algorithms have been shown to perform particularly well on sequential information. Models proposed in the literature exploit the raw logs that the interaction of students with the e-platform generates. Moreover, deep strategies are convenient because, unlike classic learning techniques, they avoid most manual feature engineering problems, thus facilitating the task of inexperts in the e-learning domain.

As depicted in Figure 2.2 and, specifically, in *Deep Learning* sub-classes, we have identified two families of deep learning approaches used in the literature: i.e. recurrent and convolutional neural networks. SDP studies have not adopted deep strategies until recently. So, the works we enlist in the following sections are far less numerous than those belonging to the classic learning methods discussed above. To analyse these works, we follow the same structure as the previous section.

Recurrent Neural Networks (RNNs)

Unlike traditional neural networks, an RNN remembers history. An RNN can be thought of as n copies of the same network put in the sequence $1, ..., n$ s.t. the i-th copy passes information to the $(i+1)$-th copy. They allow previous outputs to be used as inputs while having hidden states. Hence, RNNs are highly correlated to sequences and, in our case, adaptable to the SDP problem.

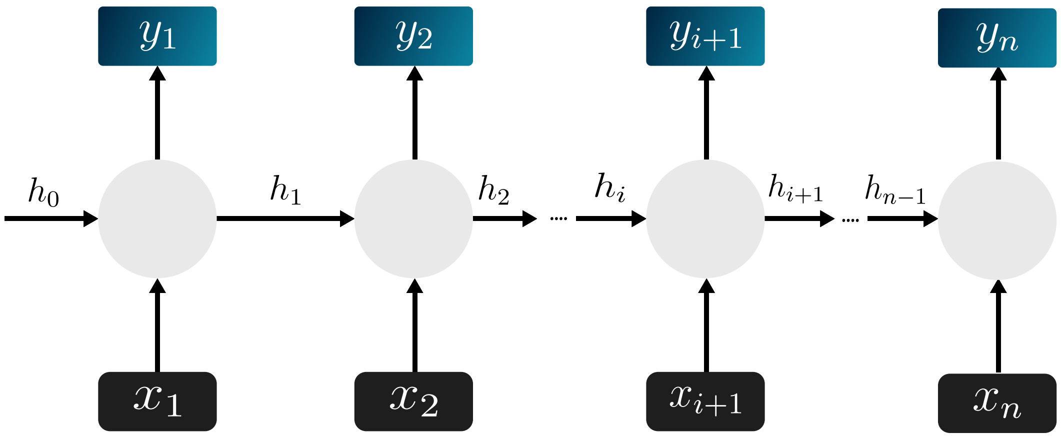


Figure 2.8. Unrolled representation of an RNN.

Figure 2.8 illustrates the unrolled representation of an RNN running for n time steps. At each time step i, the output y_i and the hidden state h_i are calculated as follows[10]:

$$h_i = a_1(W_{hh}h_{i-1} + W_{hx}x_i + b_h)\ \forall i \in [1, n]$$

$$y_i = a_2(W_{yh}h_i + b_y)\ \forall i \in [1, n]$$

where x and y represent the input and output vectors, respectively. h is the hidden state, represented as a vector that the network passes to the next time step. W_{hh}, W_{hx} and W_{yh} are weight matrices whose dimensionality is equal to the dimensions

[10]We take inspiration from the equations in `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks`.

of its subscripts. b_h and b_y are the bias terms added, respectively, to h and y. The dimensions of these vectors hold the same reasoning as those for the weight matrices. Finally, a_1 and a_2 are activation functions. The activation functions are set according to the problem at hand, but, generally, sigmoid and ReLU are used.

Because it is difficult to capture long term dependencies, an RNN suffers from the vanishing/exploding gradient phenomena. In particular, when updating the weights through backpropagation, gradients can exponentially decrease/increase w.r.t. the number of layers. There are two methods to resolve these two anomalies. One consists of capping the gradient's maximum value during backpropagation - known as *gradient clipping*. The other exploit specific gates Γ to mitigate the vanishing/exploding gradient problem. There are four types of gates: update Γ_u, relevance Γ_r, forget Γ_f, and output Γ_o gates. RNN flavours that deal with the vanishing/exploding gradients use these gates to cope with the problems provided by these phenomena. In other words, gates control the way information is propagated in a network. There are two well-known RNN cells: the Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU). Because the literature works rely only on vanilla RNNs and LSTMs, we provide details for the LSTM cell and the characteristic equations of its gates.

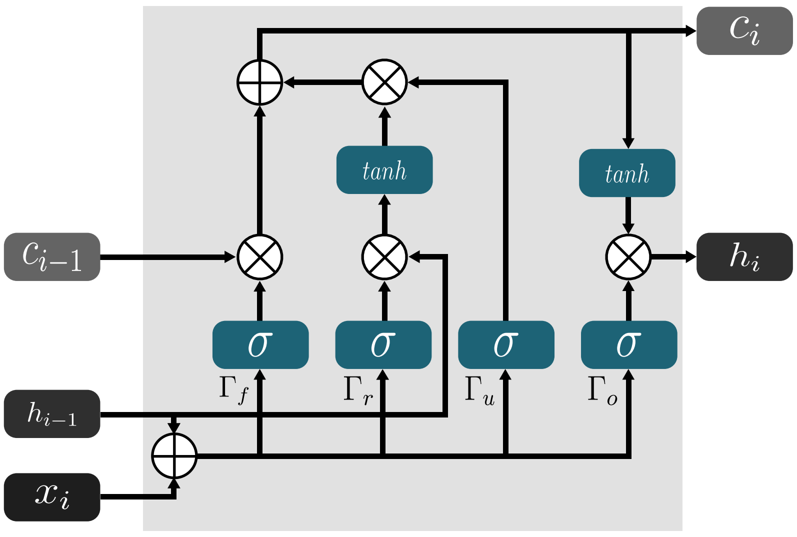


Figure 2.9. Detailed representation of an LSTM unit.

LSTMs are capable of learning long-term dependencies [57]. The key to an LSTM is the cell state c_i. The LSTM can add/remove (operations regulated by gates) information to the cell state. Notice that only the update of the hidden state h_i differs from a plain RNN. Meanwhile, the output y_i remains untouched. Therefore, we describe only the steps necessary for the update of the hidden state. Figure 2.9 illustrates the different components to update the context and hidden states of the LSTM relying on the aforementioned gates. Firstly, an LSTM needs to decide what information to throw away from the cell state. This decision is made by the forget gate which looks at h_{i-1} and x_i in order to output 0 (forget) or 1 (keep). Formally, $\Gamma_f = \sigma(W_f \times (h_{i-1} \oplus x_i) + b_f)$ where $\oplus$ represents the concatenation operator. Second, it decides what new information to store in the cell state which is regulated by the relevance gate. The LSTM needs a vector of new candidate values c_i to update the previous cell state. In other words, $c_i = tanh(W_c \times ((\Gamma_r \times h_{i-1}) \oplus x_i) + b_c)$ where $\Gamma_r = \sigma(W_r \times (h_{i-1} \oplus x_i) + b_r)$. Subsequently, the current cell state is updated $c_i = \Gamma_u \times c_i + \Gamma_f \times c_{i-1}$ where $\Gamma_u = \sigma(W_u \times (h_{i-1} \oplus x_i) + b_u)$. Finally, the output gate decides what to produce by exploiting a sigmoid function, $\Gamma_o = \sigma(W_o \times (h_{i-1} \oplus x_i) + b_o)$. The current hidden state is updated according to the current cell state, $h_i = \Gamma_o \times tanh(c_i)$.

Works based on RNNs Although the majority of the works in the literature base their prediction models on LSTM cells, Wang et al. [132] exploit a recurrent network as the last predictive layer of their architecture. They use the predictive power of an RNN for the dropout decision because past e-tivities affect current student interactions. In other words, e-tivities that are closer in time have a more dominant influence than those that are distant. The authors exploit this characteristic by using an RNN among different time slices. The activation function for the hidden state is the ReLU function except for the last one, a sigmoid function.

Works based on LSTMs The authors in [35] propose two different Input-Output Hidden Markov Models (IOHMMs) approaches. Finally, they demonstrate that a variant of an RNN with twenty LSTM cells is the best performing model. Ding et al. [33] use a combination of LSTMs and auto-encoders (AEs) to learn compact and effective representations from raw data. They choose an unsupervised approach to extract underlying representations of the input sequences they use. Then, an LSTM intertwined with AEs combines the sequence-to-sequence learning framework adopted in the SDP scenario [123]. As an AE requires, the authors use two LSTMs: an encoder and a decoder LSTM. The input sequence $(x_1, x_2, ..., x_{i-1})$ is fed into the encoder which learns a hidden representation z forwarded to the decoder. The decoder LSTM reads the original sequence - with the exception of x_1 - in reverse order and it outputs $(\hat{x}_{i-1}, \hat{x}_{i-2}, ..., \hat{x}_1)$ as a reconstruction of the original input. The authors try to predict student performance with only behavioural patterns from the first i course phases. For this, the predictive power of the initial model is not optimal. Therefore, they exploit transfer learning from finished courses to cope with ongoing courses and real-time dropout identification. In this way, they can use the features after phase i to the decoder part. The prediction happens at the encoder level only using the first i course phases. The authors, hence, add a predicting decoder parallel to the reconstruction decoder. The newly added component to the overall architecture aims to predict features' sequences in the later phases. Finally, the authors use the learnt representations as new feature inputs to a fully connected one hidden layer neural network to distinguish between dropout students and persisting students. Differently from the previous works that predict the dropout status of a particular student s, the authors in [125] use LSTMs to predict the next probable learning resource[11] that s will visit according to their personalised learning path. The proposed method outputs also the estimated time s spends in the predicted learning resource, thus giving sprout to a time-augmented LSTM.

[11]The authors represent each student as a sequence of URLs corresponding to the learning resources visited in a specific timestamp.

Convolution Neural Networks (CNNs)

CNNs are deep learning models that, taken in a two-dimensional[12] input, assign importance to objects identified in them[13]. Additionally, they can differentiate between two images by the same concept of giving priority to the various things in the input. A CNN is capable of capturing spatial and temporal dependencies in an image by applying specific filters. Without loss of generality, a CNN's objective is to reduce images into a better-processing format without losing important features. Three principal components constitute a CNN: i.e. convolution kernel, pooling layer and fully connected layer. The last one is a neural network used as a final layer for prediction purposes.

As described, CNNs are best suited for volume data $h \times w \times d$ where h, w, and d represent, respectively, height, width and depth. Matrices are also volume data with $d = 1$. An RGB image is a three-dimensional matrix, each corresponding to one of the colour channels. However, we assume that there is only one channel to describe the kernel and pooling layers, hence $d = 1$. When $d > 1$, the kernel and pooling layers are applied to each dimension independently. In each of the following, we assume that the input image is illustrated in Figure 2.10. We give a detailed explanation of only the first two components of a CNN architecture.

0	2	1	0	4
1	1	1	2	0
2	5	2	4	0
5	2	3	6	0
0	0	3	1	1

Figure 2.10. A $5 \times 5 \times 1$ matrix as image example.

Convolution kernel/layer A kernel - oftentimes denoted as a filter - is a three dimensional matrix $h_k \times w_k \times d_k$. In the scenario of CNNs, the depth of the kernel, d_k matches the channel number of the image; thus, $d_k = d$. Notice that the number of kernels n_k may vary from application to application. Therefore, each of the $1, ..., n_k$ kernels is applied to the same input volume.

Suppose that we have a $3 \times 3 \times 1$ kernel as the matrix in the upper-left corner of Figure 2.11. In this case, we are applying a stride $n_s = 1$, which denotes the number of cells to slide right/down. The kernel matrix performs a summation of the element-wise multiplication with the portion of the image it collides with. This operation is called a convolution operation[14]. Subsequently, it shifts one cell to the

[12]Here, we use two dimensions for clarity purposes. CNNs can also take multi-dimensional inputs.

[13]Notice that CNNs can also be used in other contexts in which data has a spatial relationship such as document classification and sentiment analysis.

[14]If the kernel and input image have depths different from 1, then the element-wise multiplication is executed singularly for each dimension, and the summation operation is done throughout all of them.

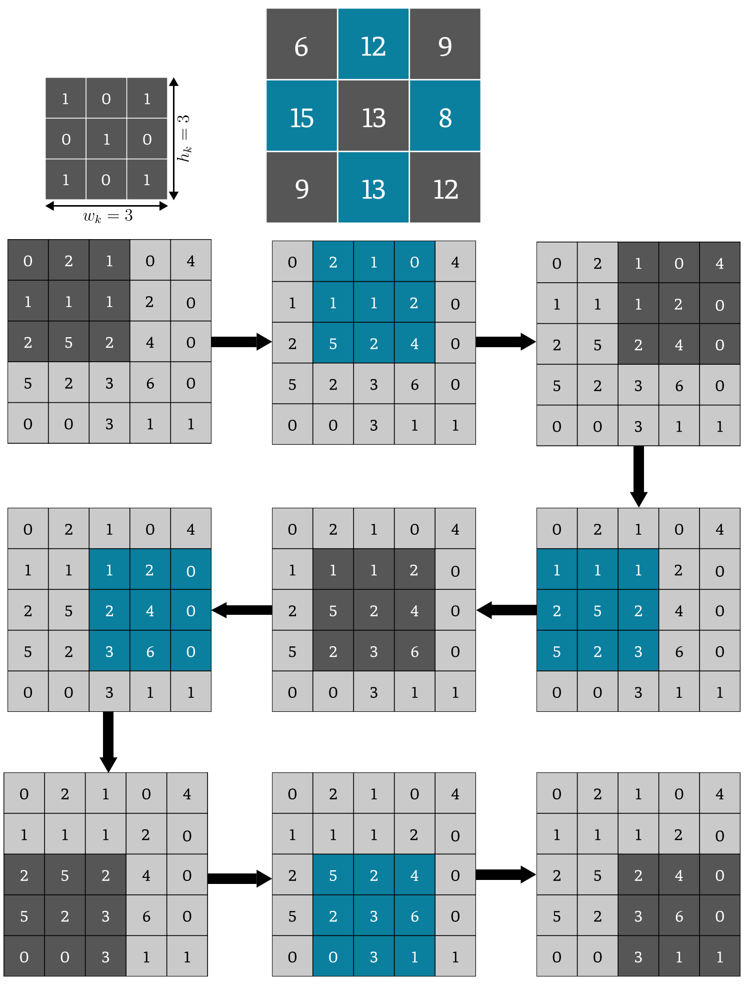


Figure 2.11. Application of a 3×3 kernel (top-left corner) on the input matrix. The output matrix is depicted at the top of the image. The colours represent the different slices where the kernel is applied. Each cell of the output matrix is calculated as the sum of the Hadamard product of the kernel and the highlighted section of the input matrix.

right. After performing the convolution operation on the first row, it moves down one cell and proceeds as before[15]. The multiplication is depicted with different colours, and the resulting matrix is the one situated at the top-centre part of the illustration. In this way, the original matrix is reduced into a condensed representation version. Notice that the elements of the kernel in this example form an X. In computer vision, filters are used to detect geometric features such as edges, blobs, vertical stripes. Hence, in this particular illustration, we are searching for X shaped elements.

Often, the input image and the kernel size are not proportionate. In this case, a padding operation - usually the input is zero padded - is performed in the original input in both the height and width dimensions. Hence if we have a $h \times w \times d$ tensor and we pad with size n_p, then the new tensor's dimensions are $(h+2n_p)\times(w+2n_p)\times d$. Therefore, in order to calculate the dimensions of the convoluted volume we need to

[15]Notice that if the stride n_s were different from 1, then the shifting would happen every n_s cells to the right and then n_s cells down.

consider also the striding factor. A striding of n_s shrinks the image n_s times. Hence, if we apply n_k kernels of dimensions $h_k \times w_k \times d$ with stride n_s to a n_p-padded volume $h \times w \times d$, it transforms this volume into another one with dimensions $(\frac{h+2n_p-h_k}{n_s}+1) \times (\frac{w+2n_p-w_k}{n_s}+1) \times n_k$. Notice that the new volume's depth is equal to the number of different filters we apply to the input image in order to detect distinct high-level features.

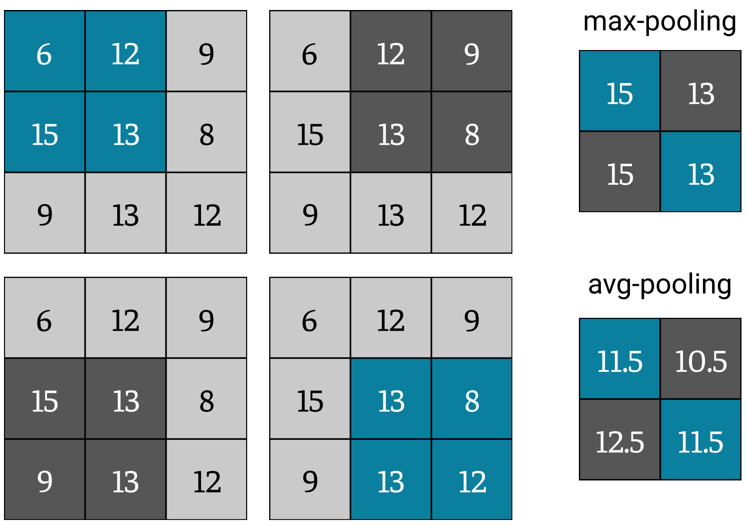


Figure 2.12. Max and average pooling for the previous convolutional layer.

Pooling layer This layer is similar to the convolutional layer. It helps reduce the spatial size of the volume to make it less computationally intensive. Additionally, it is effective to extract dominant features. There are two fundamental types of pooling layers: i.e. max and average pooling. The former returns the maximum value from the portion of the image covered by the pooling matrix. The latter returns the average. However, max-pooling also performs as a noise suppressant. It eliminates the noisy activations and performs de-noising with dimensionality reduction. Generally, the pooling layer comes after a convolutional layer. The pooling layer can also have a stride n'_s and a padding n'_p option, which works in the same way as described in the previous paragraph. For the sake of argument, suppose that $n'_s = 1$, $n'_p = 1$ and the dimension of the pooling kernel is $2 \times 2 \times 1$ as in Figure 2.12. The matrix taken into consideration in this example is the one that the previous convolution layer has produced. The pooling kernel returns the maximum value for each of the 2×2 portions of the matrix. The same reasoning holds with the average pooling operation. In general, the pooling kernel is a volume $h_p \times w_p \times n_k$ where its depth matches the number of filters applied in the convolution layer. Unlike the convolution layer, the input depth is maintained in this layer, whereas the height and width can change. The new volume produced after the pooling operation is of shape $(\frac{h_k+2n'_p-h_p}{n'_s}+1) \times (\frac{w_k+2n'_p-w_p}{n'_s}+1) \times n_k$.

Works based on CNNs. The core of the CFIN system presented in [36] relies on a one dimensional CNN. Moreover, the authors boost their model's performances by depending on XGBoost [22]. They incorporate context information, including user and course data, into a single prediction schema. Thus, because contextual data correlates with learning activities, the authors rely on CNN to learn a context-aware representation for each activity feature - process called context-smoothing. Three crucial steps compose CFIN: i.e. context-smoothing, attention mechanism, and prediction component. Feature augmentation, embedding and fusion form the first step. These procedures come before the convolution part, which learns a context-aware representation for each selected e-tivity feature. The authors exploit an attention mechanism[16] to model attention-based interactions for e-tivity features using contextual information. Then, a dense neural network receives the produced weighted-sum vector from the attention mechanism, and it predicts the outcome label. Qiu et al. [110] propose DP-CNN to tackle the SDP problem. The convolution stage follows an end-to-end approach. It comprises several convolutional layers and a last fully connected layers with a sigmoid function. Precisely, DP-CNN consists of a four-layer network with two convolutional layers and two fully connected ones. To preserve all the feature information, the authors do not use pooling layers. Because the input that the authors consider is two-dimensional, the convolution kernel is also two-dimensional. The kernel activation function is a ReLU. The third layer of the architecture also uses a ReLU activation function, whereas the last one - the predictive layer - relies on a sigmoid function.

As mentioned above, Wang et al. [132] utilise a simple RNN as the last layer of their prediction strategy, ConRec. The authors use a CNN to extract features for the e-tivity matrices in each time slice. Therefore, the first four layers of the architecture are convolution and pooling layers. The fifth layer is a fully connected neural network that fuses features that the convolutional and pooling layers extract. This layer generates one vector for each time slice. The RNN takes these vectors and engages in determining the instances' dropout class labels.

Graph Convolutional Neural Networks (GCNs)

Many important real-world datasets come in the form of graphs or networks: social networks, knowledge graphs, protein-interaction networks, the World Wide Web. Yet, until recently, very little attention has been devoted to the generalisation of neural network models to such structured datasets. As of recent, a number of papers re-visited this problem of generalising neural networks to work on arbitrarily structured graphs ([29, 34, 69]) in domains that have previously been dominated by kernel-based methods and graph-based regularisation techniques.

A GCN learns a function of features on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ taken in input having the following characteristics:

- Every node $v_i \in \mathcal{V}$ has a feature description x_i. Therefore, $\mathcal{V}$ can be represented as a feature matrix $X \in \mathbb{R}^{|\mathcal{V}| \times D}$ where D is the number of input features.

- The graph structure contains a representative description as an adjacency matrix A[17].

[16]We invite the reader to check the work of Bahdanau et al. [7] for more detail on the attentive networks in the context of neural machine translation.

[17]Notice that the graph structure can also be represented as a function derived upon the adjacency matrix A.

and produces a node-level output $Z \in R^{|\mathcal{V}| \times F}$ where F is the number of output features per node. Graph-level outputs can be obtained via pooling operations as shown in [34]. Thus, each layer in a GCN can be expressed as a non-linear function $h_{i+1} = f(h_i, A)$ with $h_0 = X$ and $h_L = Z$ where L is the number of layers.

For the sake of argument, let us consider the following form of layer-wise propagation rule:

$$f(h_i, A) = \sigma(A h_i W_i)$$

where W_i is the weight matrix of the i-th layer and σ is an arbitrary non-linear activation function. According to the literature [69], the previous equation has two limitations:

1. Multiplying by A means that, for each node, we have to sum up all the feature vectors of all neighbouring nodes, but not the node itself in case there are no self-loops in $\mathcal{G}$. To avoid this first pitfall, we can force $\mathcal{G}$'s structure to have self-loops by adding an identity matrix to A (i.e. $\hat{A} = A + I$).

2. A is not normalised in general leading to a change of scales in feature vectors when multiplying by the matrix[18]. Hence, we need to normalise A such that all rows sum to one (i.e. $D^{-1}A$ where D is the diagonal matrix containing node degrees). Now, the multiplication $D^{-1}A$ corresponds to taking the average of neighbouring node features.

Nevertheless, the average of neighbouring node features has led to a more practical normalisation (i.e the symmetric normalisation $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$) which was exploited in [69] to express the following propagation rule in GCNs:

$$f(h_i, A) = \sigma(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} h_i W_i)$$

where $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$.

Now that we have introduced a solid propagation rule in GCNs, we provide the reader with the under-the-hood details of learning node embeddings with GCNs [105]. We interpret the GCN model as a generalised and differential version of the Weisfeiler-Lehman algorithm on graphs. For completeness purposes, we provide the reader with the 1-dimensional Weisfeiler-Lehman algorithm in 2. It essentially assigns a feature that uniquely describes each node v_i in the graph $\mathcal{G}$. We can adapt this algorithm in the scenario of a GCN such that $h_{i+1}^{v_i} = \sigma(\sum_j \frac{1}{c_{i,j}} h_i^{v_j} W_i)$ where v_j are the neighbouring nodes of v_i, $c_{i,j}$ is a normalisation constant for the edge (v_i, v_j) originating from the symmetrically normalised adjacency matrix $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. Notice that this propagation rule is a differentiable and parametrised variant of the hash function in algorithm 2. By choosing an appropriate non-linear activation function and an orthogonal initialisation of the weight matrices of the layers [43], the update rule becomes stable in practice reaching a convergence. The smoothed embeddings produced entail a relationship between node distance as similarity measures of local graph structures.

Works based on GCNs Hu and Rangwala [60] propose an attention-based GCN to determine the academic success of students by exploiting their performances

[18]The change on the scales of the feature vectors can be verified by looking at the eigenvalues of A.

```
Data: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $k$ is the number of steps to repeat
$\mathcal{V} = \{v_1, v_2, \dots, v_i, \dots, v_{|\mathcal{V}|}\}$;
while $t < k$ or until convergence do
    for $v_i \in \mathcal{V}$ do
        Let $\{h^{v_j}\}$ be the feature set of neighbouring nodes $\{v_j\}$ of $v_i$;
        Update node features:;
        $h^{v_j} = \text{hash}(\sum_j h^{v_j})$, where hash($*$) is an injective hash function;
    end
    $t = t + 1$;
end
```

Algorithm 2: The pseudocode of the 1-dimensional Weisfeiler-Lehman algorithm

in prerequisite courses. They take into consideration the grades of a particular student s in each of the courses taken as prior knowledge as well as the sequence in which s has finished these courses and the intrinsic relationships thereof. The authors use two GCN layers to aggregate information from a node's second-hop neighbours. In other words, the authors exploit information from courses taken up to two past semesters. The output of the GCN is a graph embedding comprising of the information about the knowledge and skills acquired in prior courses that s has taken. Nevertheless, some prerequisite courses have different importance for a particular target course. Thus, the authors capture the importance differences of the prior courses by integrating an attention layer in the original two-layered GCN model. As a consequence, the graph embedding matrix is weighted by the attention scores and given in input to a pooling layer. The produced vector goes into a multilayered neural network whose output is the estimated grade of s for that specific course. Notice that the prediction of this specific work is a real number (regression) instead of a binary label (classification). However, the system is capable of identifying failing students whose grade results lower than the passing threshold.

2.2 Open Issues in SDP

Considering that SDP is a new research field with much attention in the coronavirus era, the literature contains issues yet addressed. Here, we identify three of the major criticalities in SDP. The first two issues are related to datasets and the student trajectories structure, whereas the third one encompasses the evaluation criteria and benchmark standardisation. For more detail on the challenges and solutions in this research field, we refer the reader to [107, 108].

2.2.1 Inter-activity Temporal Gaps

Student trajectories are usually modelled as a sequence of events ordered in time. Generally, this sequence is discrete, meaning that each event's time is drawn from a finite set of times. Hence, at any time, students can interact with the e-platform and engender a variety of e-tivities. The interaction behaviour varies from student to student. Therefore, some students might have a steady interaction pattern while others check the e-platform parsimoniously. Moreover, two consecutive e-tivities have a temporal gap between them: i.e. the beginning of the successive e-tivity comes later than the end of the previous one.

To the best of our knowledge, the literature does not emphasise temporal gaps between e-tivities. Nevertheless, this phenomenon is important since it entails how the students' behaviours mutate over time. The trend shift in generating e-tivities provides insight into the direction of the student education path. An increase in temporal gaps signifies that a particular student has learning difficulties transitioning into a dropout decision. Meanwhile, a decrease in gaps means that a student is completing the online course/degree successfully.

Additionally, the importance of inter-activity gaps depends on the scenario. While fast-paced MOOCs might experience fewer delays between e-tivities, the phenomenon is critical in e-degrees. MOOCs' characteristics permit students to be more interactive, thus having shorter delays in completing course modules. Contrarily, online degrees, being long-term and slow-paced, induce students to take more breaks between one e-tivity and the other. Furthermore, interpreting these inter-activity gaps is difficult because they depend on the way students interact with the e-platform and course material. In other words, long breaks do not produce a grave alarm if a particular student is consistently interacting in a more coarse-grained fashion. Differently, if a student is active and does not take long to finish individual courses in an e-degree, a single long break should raise a dropout alarm.

2.2.2 Long-term E-tivity Dependency and E-degrees

As mentioned before, MOOCs usually span from four to twelve weeks. Students enrolled in e-degrees can take up to five years, depending on the period of the programme, to graduate. Therefore, generated e-tivities by students enrolled in fast-paced courses do not have an immense impact on shaping future interaction behaviours with the e-platform. Meanwhile, e-degrees, being extended more vastly w.r.t. MOOCs, consider the importance of past e-tivities that impact future decisions. Moreover, online degrees consist of multiple courses interlaced with one another, having different intertwined dependencies. Hence, students not participating actively in one course might drop out when they tackle the second module of the same course (e.g. one cannot successfully finish the course of Machine Learning if they have not completed Linear Algebra or Basic Statistics).

The literature has not sufficiently explored strategies that cope with the student trajectories' long-term sequentiality and feature dependency. Deep sequential learning strategies have been proved to improve performances when analysing time-related information. Additionally, as mentioned above, it is necessary to address the more complex case of e-degrees to tackle the problem of long-term e-tivity dependency. As emphasised from Table 2.3, the literature does not focus on online degrees, but it copes with short-term MOOCs. Modelling e-degrees and handling the dependency of different e-tivities is a challenging task in the long run. The decision to abandon the studies for a particular student might arise from complex interactions between the sequentiality of the generated e-tivities according to a non-deterministic student interaction behaviour. In general, modelling e-degrees is not a simple extension of the characteristics of MOOCs. Moreover, according to the Table, studies based on e-degrees do not release their datasets online because their institutions' student

Table 2.2. The list of evaluation measurements used in the literature. Notice that only a small percentage of the works in the literature rely on AUCPR, the stablest evaluation metric in a class imbalance scenario.

	Measurements	Works
Rates	False Negative	[61, 91]
	False Positive	[45, 61, 91]
	True Negative	[12, 91]
	True Positive (Recall)	[3, 12, 30, 45, 84, 88, 91, 110, 132, 133]
	Precision	[3, 12, 30, 45, 84, 88, 110, 132, 133]
	Accuracy	[1, 2, 3, 12, 30, 46, 61, 72, 73, 75, 77, 76, 78, 88, 91]
F_β	$\beta = 1$	[2, 3, 23, 21, 36, 42, 45, 84, 110, 132, 133]
	$\beta = 1.5$	[30]
	AUCROC	[2, 23, 21, 35, 36, 42, 45, 95, 110, 112, 113, 127, 132]
	AUCPR	[23, 112]
	Cohen's Kappa	[3]
	MSE	[33]
	MAE	[2]
	OPER/UPER	[2]

information is proprietary. Therefore, another issue in SDP is having common datasets used for benchmarking and easier confrontation of state-of-the-art model performances.

2.2.3 Absence of Standardised Evaluation Benchmark

The literature uses variegated evaluation metrics to assess the performances of the predictive models on future unseen cases. Generally, several well-known evaluation metrics are used in statistics, machine learning and deep learning. However, given the unbalanced nature[19] of the data in the SDP scenario, only a fraction of the proposed evaluation strategies are useful. Here, we tackle the lack of a standardised evaluation benchmark due to the spurious usage of metrics and, most importantly, of the tendency not to consider common benchmarking datasets (see Table 2.3).

The works in the literature consider SDP as a binary classification problem. Note that even a probabilistic output must be eventually converted into a binary decision, i.e., whether a student is at the risk of dropout. We have identified several measurements that the literature exploits based on classification problems.[20]. Hereafter, we report a description of these metrics, and we summarise their usage in Table 2.2:

1. *Accuracy.* Pioneer works [73, 75, 77, 76, 78] relying on simple off-the-shelf machine learning algorithms employ only accuracy as a means to assess the

[19]The number of persisting and dropout students is different, and some courses might have more students that obtain certificates of completion rather than cutting their studies short. Others might have the opposite scenario.

[20]Some works exploit also regression-related metrics adopting them into a classification scenario.

Table 2.3. The types of OLEs used in the literature with links to the datasets available online. Most studies concentrate on fast-paced MOOCs or single unit online courses. Only a handful of studies tackle the higher complex scenario of e-degrees.

Type	Resource	Study	Available link
MOOC	HarvardX	[113]	None
	edX	[33]	`https://www.edx.org/course/introduction-to-java-programming-part-1-2` `https://www.edx.org/course/introduction-to-java-programming-part-2-2`
		[35]	`https://www.edx.org/course/introduction-to-java-programming-part-1-2` `https://www.edx.org/course/introduction-to-java-programming-part-2-2`
		[42]	`https://www.edx.org/course/big-data-and-education`
		[95]	`https://www.edx.org/course/i-heart-stats-learning-love-statistics-notredamex-soc120x`
	MITx	[127]	`https://www.edx.org/xseries/mitx-circuits-and-electronics`
	Coursera	[35]	`https://www.coursera.org/learn/gastronomy`
		[42]	None
		[112]	`https://www.coursera.org/learn/surviving-disruptive-technologies` `https://www.coursera.org/learn/genes`
		[136]	None
	KDDCup15	[21, 36, 84, 110, 132]	`http://data-mining.philippe-fournier-viger.com/the-kddcup-2015-dataset-download-link/`
	EMNLP 2014	[3, 72]	None
e-course	CS in HOU	[76, 77, 75, 73, 134]	`https://www.eap.gr/en/courses/216-computer-science/course-structure/1556-pli10-introduction-to-informatics`
	Programming I in YU	[1]	None
	Information Systems in OP	[78]	None
	Information Literacy & Information Ethics	[61]	None
	Computer Networks & Communications NTUA	[88]	None
	Web Design NTUA	[88]	None
Online degree	Electrical ENG in TU/e	[30]	None
	ENG in UFRJ	[91]	None
	State University in NRW	[12]	None
	PUAS in NRW	[12]	None
	UK Bangor University	[45]	None
	Open University	[46, 133]	`https://analyse.kmi.open.ac.uk/open_dataset`

performances. Accuracy solely can be misleading. In SDP, considering the class imbalance problem, a model can predict the value of the majority class for all predictions and achieve a high classification accuracy. Nevertheless, this model is not useful in the problem domain. Therefore, it is mandatory to apply additional measurements to evaluate the classifier properly. Other papers [2, 3, 12, 30, 46, 61, 72, 78, 88, 91] also use different metrics to obtain a more realistic view of the achieved performances.

2. *Confusion Matrix and F-measure.* To tackle the inefficiency of accuracy of capturing good classifiers, the literature relies on rates calculated from the confusion matrix [3, 12, 30, 45, 61, 84, 88, 91, 110, 132, 133] (e.g. Precision, Recall and their derivatives such as the F scores). Notice that the majority of the studies that rely on precision and rate-based metrics exploit the F-score measurement. However, the F-score does not provide any intuitive explanation of the performances. On the other hand, many samples - as in SDP's case - do not affect precision. Precision measures the number of true positives from the samples predicted as positives (i.e. true positives and false positives together), and it does not depend on the number of samples. A mixture of recall, precision and F-score gives an intuition of the overall performances of a classifier. Because the general formula of the F-score is $F_\beta = (1+\beta^2) \times \frac{P \times R}{\beta^2 \times P + R}$ where P and R denote, respectively, the precision and recall, one can exploit several choices of β to measure the relationship between P and R. The most common choice of β is 1.

3. *AUCROC and AUCPR.* Recall and false positive rate (FPR) constitute the building blocks of the ROC curve. They measure the ability to separate dropouts from persisting students. The ROC is a probability curve, and AUCROC tells how much a model can distinguish between classes. The higher the AUCROC, the better a model predicts dropouts as 1s and persisters as 0s. Therefore, the works that rely on AUCROC combine the true positive and false positive rates to assess the model's predicting power [2, 23, 21, 35, 36, 42, 45, 95, 110, 112, 113, 127, 132]. However, AUCPR, used by [23, 112], is a more suitable measure since it is more resistant to the class imbalance problem w.r.t. AUCROC [28]. Unlike the ROC curve, the Precision-Recall (PR) curve does not account for true negatives because they are not components of either the recall or the precision formula[21].

4. *Cohen's Kappa.* Only a minority of research exploits the Cohen's Kappa score [3]. This measure manages both multi-class and imbalanced class problems. Cohen's Kappa represents the inter-rater reliability score (taking into account an agreement by chance) of two models (classifiers). Therefore, researchers could exploit this metric to verify the alliance between their classifier and that corresponding to the ground truth. In this way, one determines how close the selected model's predictions are with the original labels. Although -1 indicates no agreement and $+1$ perfect agreement, the reader can consult the work in [92] when choosing an acceptable range of the kappa score.

[21]`http://www.chioka.in/differences-between-roc-auc-and-pr-auc/`

5. *Mean Squared Error (MSE).* It measures the average of the squares of the errors represented as the difference between the predicted values and the ground truth: i.e. MSE $= \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$ where $\hat{y}_i$ represents the i-th predicted value and y_i is the i-th true label. In this context, it determines how well the model fits the data. A perfect MSE score is zero, but since algorithms contain some randomness, MSE scores are strictly positive. We consider as optimal those models whose MSE is close to zero. Ding et al. [33] employs this metric for their performances.

6. *Mean Absolute Error (MAE).* It is the average of absolute errors: i.e. MAE $= \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$ where $\hat{y}_i$ and y_i have the same meaning as those mentioned in MSE. It treats both underestimating and overestimating of the true value in the same manner. As with MSE, MAE is also a negatively oriented metric, meaning that lower scores are better than those tending to infinity. Ameri et al. [2] use MAE.

7. *Underestimated/Overestimated Prediction Error Rate (UPER/OPER).* To mitigate the underestimating and overestimating effect that MAE lacks on the predicted values, Ameri et al. [2] introduce UPER and OPER. The authors define UPER as the fraction of the underestimated prediction output over the entire prediction error. In other words,

$$UPER = \frac{\sum_{i=1}^{n} I(\hat{y}_i < y_i)}{\sum_{i=1}^{n} I(\hat{y}_i < y_i) + \sum_{i=1}^{n} I(\hat{y}_i > y_i)}$$

   where $I(\hat{y}_i < y_i) = 1$ if $\hat{y}_i < y_i$ and $I(\hat{y}_i > y_i) = 1$ if $\hat{y}_i > y_i$. The indicator functions $I(\hat{y}_i < y_i)$ and $I(\hat{y}_i > y_i)$ are equal to zero if their conditions are not satisfied. Therefore, $\sum_{i=1}^{n} I(\hat{y}_i < y_i)$ is the number of underestimated instances and $\sum_{i=1}^{n} I(\hat{y}_i > y_i)$ is the number of overestimated instances. OPER is the opposite of UPER: i.e. $OPER = 1 - UPER$. The authors exploit these two metrics to estimate the semester in which a student drops out. Nevertheless, researchers might consider UPER and OPER if they treat SDP as a regression problem. i.e. the label of each student now represents the probability that they drops out.

Considering the increase in popularity of the SDP research field, most of the OLEs made available parts of their course data and information of enrolled students. Table 2.3 summarises the adopted datasets in the literature. We group the resources according to their type (i.e. MOOCs, online courses, and e-degrees). The publication of these datasets could aid researchers to compare their prediction results with available baselines.

The KDD Cup competition[22] is one of the few common benchmarks adopted in the literature. In this competition, systems have been compared on well-known online courses provided by famous MOOCs such as Coursera and edX. The goal was to predict the probability of a student dropping out of a course in ten days. The competition used AUCROC to evaluate the submissions. The authors in [21, 36, 84, 110, 132] exploit the mentioned benchmark dataset. As reported in

[22]`https://biendata.com/competition/kddcup2015/`

Table 2.2, we can compare [21, 36, 132] as they use AUCROC for the evaluation. Instead, we can also compare [36, 84, 110, 132] because they all rely on the F_1 score. From the first triple of papers, Feng et al. [36] is the winner with an AUCROC of 0.9093. In contrast, the most performing strategy from the second quadruple is Li et al. [84] with an F_1 score of 0.9241. However, we note that predicting a student dropout with ten-day anticipation might not allow in practice the adoption of any retention strategy by the institution.

For completeness, we mention two studies concerned with traditional degrees that used data on first-year students enrolled at Wayne State University [2], and George Mason University [23]. The former does not distinguish between students of different degrees and uses all the enrolled freshmen for training. Instead, Chen et al. [23] study the dropout phenomenon by discriminating between newcomers registered in nine major STEM degrees (e.g. Mathematics, Physics, Chemistry).

Besides KDDCup15 and the Open University, all the other datasets are not publicly available and require that researchers contact data providers to access them. These studies often devise prediction strategies tailored for their institutions, and thus, they do not release sensitive data about their enrolled students. We note that the publication of sensitive data is to be strictly compliant with privacy regulations. For this reason, in Section 4.3, we analyse critical aspects related to privacy infringement mitigation.

Chapter 3

Student Trajectory Modelling

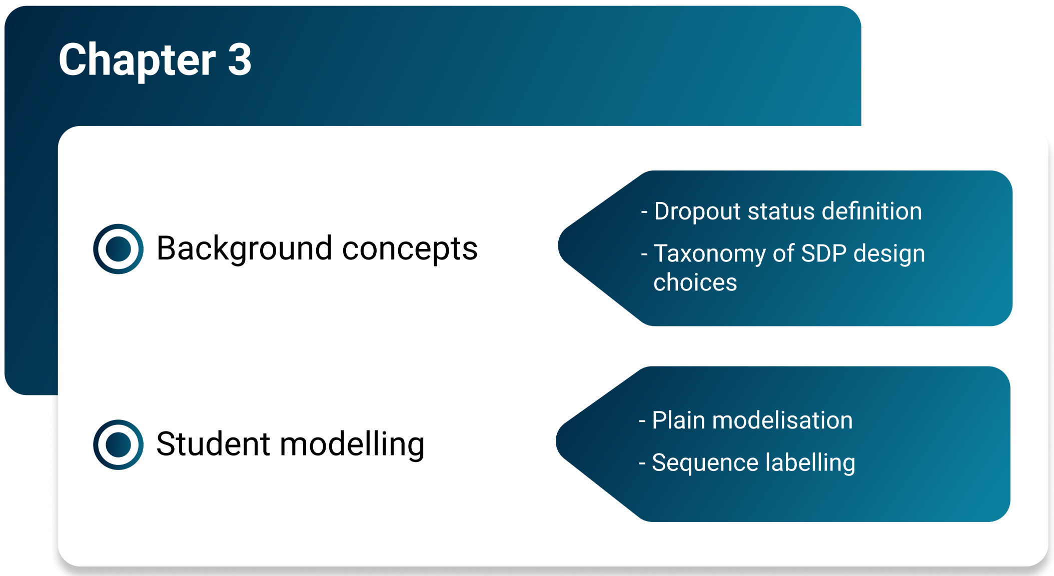


Figure 3.1. This chapter encompasses the background of the SDP research field, including background concepts and formalisation of the dropout definitions. Because the literature uses two different input modelling strategies, we divide them into plain modelisation and sequence labelling with time-series and temporal networks.

Being SDP a novel research field, the literature lacks the structure of providing a mathematical formalisation of the dropout definitions. Therefore, besides giving background concepts and formalising the input modelling strategy employed in the literature, we give also a taxonomy of different approaches utilised on the student information (e.g. student e-tivities, demographic background and previous education data). Figure 3.1 summarises the structure of this Chapter. Notice that we also provide the reader with a detailed description of the differences between the two identified student modelling techniques. On the one hand, plain modelisation comprises all time-agnostic student information such as background, data from their previous education, and current grades. On the other, sequence labelling is a more complicated task since it includes time-related data engendered from the interaction of students with the e-platforms.

3.1 Background Concepts

As summarised in Chapter 2, researchers model student e-tivities adopting either a recurrent (sequence labelling) or plain modelisation scheme. While a conventional classification problem assumes that the inputs are *independent and identically distributed (iid)*, the input features in a sequence labelling task are dependent. Without loss of generality, we introduce time-aware definitions of the relevant entities and e-tivities within online courses and e-degrees. Moreover, we extend the formalisation presented in [108] to the scenario of e-degrees.

An e-degree generally contains one or more courses. We indicate with $\mathcal{C} = \{c_1, ..., c_{|\mathcal{C}|}\}$ the set of these courses[1]. A course $c \in \mathcal{C}$ is composed of k_c sequential phases - also known as modules in MOOCs - p_i^c $(1 \leq i \leq k_c)$ that begin at time $b(p_i^c)$ and finish at time $f(p_i^c)$. We indicate with $b'(c) = b(p_1^c)$ and with $f'(c) = f(p_{k_c}^c)$ the beginning and the end of course c, respectively.

Upon enrolment, the students have to complete all the courses in an e-degree to graduate[2]. Hence, $\mathcal{S}$ represents the set of all students in an e-degree, whereas $\mathcal{S}_c \subseteq \mathcal{S}$ is the set[3] of those students that attend course c. Interacting with the e-platform of a course c consists of performing some actions from the set of all permissible ones $\mathcal{E}$. Since e-tivities constitute the basis of SDP, we annotate the student $s \in \mathcal{S}$ as a temporal sequence of length m_i of its e-tivities throughout the i phases of course c: i.e.

$$s = \left\{ e_{p_i^c}^{t_1}, ..., e_{p_i^c}^{t_{m_i}} \right\}$$

where $e_{p_i^c}^{t_j}$ $1 \leq j \leq m_i$ is the e-tivity $e \in \mathcal{E}$ performed in the phase p_i^c at time t_j. We assume[4] that a student s has completed the course phase p_i^c if their sequence of interaction has a duration equal to the course phase time span: i.e. $m_i = f(p_i^c) - b(p_i^c)$.

For readability purposes, we propose an aggregation function of e-tivities

$$w_s(t_b, t_f, c_i) = \bigcup_{t_b \leq t_j \leq t_f} e_{p_i^c}^{t_j} \tag{3.1}$$

that a student s performs in phase p_i^c within a specific time interval $[t_b, t_f]$. According to this formulation, we can express all the e-tivities that s has performed in phase p_i^c as $w_s(p_i^c) = w_s(b(p_i^c), f(p_i^c), p_i^c)$. Furthermore, we extend Equation 3.1 to describe e-tivities of student s in the entire course:

$$w_s(t_b, t_f, c) = \bigcup_{1 \leq i \leq k_c} w_s(t_b, t_f, p_i^c) \tag{3.2}$$

Hence, $w_s(c) = w_s(b'(c), f'(c), c)$. This is useful when collecting the e-tivities that stretch to more than one course phase. To cover the scenario of e-degrees, we extend

[1]The cardinality of the set of courses is equal to one when taking into consideration a single MOOC or online course.

[2]If we consider the scenarios of MOOCs and single online courses, then students have to finish a single course to obtain a completion certificate.

[3]$\mathcal{S}_c = \mathcal{S}$ in MOOCs and single online courses.

[4]Notice that this assumption is strong and it is based only on the time factor without considering if a student successfully completes assignments/homework.

Equation 3.2 to obtain the student e-tivities in the period spanning from t_b to t_f:

$$w_s(t_b, t_f, \mathcal{C}) = \bigcup_{c \in \mathcal{C}} w_s(t_b, t_f, c) = \bigcup_{c \in \mathcal{C}} \left[\bigcup_{1 \le i \le k_c} w_s(t_b, t_f, p_i^c) \right] \tag{3.3}$$

Table 3.1. Notation used with its corresponding meaning.

Relation	Notation	Meaning
	$\mathcal{C} = \{c_1, ..., c_{\vert\mathcal{C}\vert}\}$	The set of all courses in an e-degree
	c	A single course belonging to $\mathcal{C}$
	p_i^c	The i-th phase of course c ($1 \le i \le k_c$)
Course	$b(p_i^c)$	The time when phase p_i^c begins
	$f(p_i^c)$	The time when phase p_i^c finishes
	$b'(c)$	The time when course c begins. $b'(c) = b(p_1^c)$
	$f'(c)$	The time when course c finishes. $f'(c) = f(p_{k_c}^c)$
	$\mathcal{S}$	The set of all enrolled students in an online degree
Student	$\mathcal{S}_c$	The set of students enrolled in course c
	s	A single student belonging to $\mathcal{S}$
	$\mathcal{E}$	The set of all possible e-tivities
	$e_{p_i^c}^{t_j}$	A single e-tivity performed at time t_j in course phase p_i^c
	$w_s(t_b, t_f, p_i^c)$	The set of e-tivities that s performs in phase p_i^c in the time interval $[t_b, t_f]$
	$w_s(p_i^c)$	The set of e-tivities that s performs in phase p_i^c in the time interval $[b(c_i), f(c_i)]$
E-tivity	$w_s(t_b, t_f, c)$	The set of e-tivities that s performs in course c in the time interval $[t_b, t_f]$
	$w_s(c)$	The set of e-tivities that s performs in course c in the time interval $[b'(c), f'(c)]$
	$w_s(t_b, t_f, \mathcal{C})$	The set of e-tivities that s performs in the e-degree $\mathcal{C}$ in the time interval $[t_b, t_f]$
	$w_s(\mathcal{C})$	The set of e-tivities that s performs in the e-degree $\mathcal{C}$ in the time interval $[b'(c_1), f'(c_{\vert\mathcal{C}\vert})]$

Let n_s be the length of the temporal sequence of student s:
i.e. $n_s = \sum_{c \in \mathcal{C}} \sum_{1 \le i \le k_c} m_i$. Notice that two students s' and s'' might have different lengths $n_{s'} \neq n_{s''}$ because their interactions with the online degree have different durations. We annotate a student $s \in \mathcal{S}$ as a sequence that spans over the entire duration of the e-degree denoted as $d(\mathcal{C})$ where $d(*)$ represents the sum of the duration of all courses $c \in \mathcal{C}$: i.e. $d(\mathcal{C}) = \sum_{c \in \mathcal{C}} f'(c) - b'(c)$. In other words, a student $s \in \mathcal{S}$ is denoted as a time series of e-tivities throughout the entire degree

$$s = w_s(\mathcal{C}) = w_s(b'(c_1), f'(c_{|\mathcal{C}|}), \mathcal{C}) \tag{3.4}$$

According to the definition of the duration of an e-degree, $d(\mathcal{C})$, and that of the length n_s of a student trajectory over $\mathcal{C}$, there are three scenarios to consider assuming that students enrol at the beginning of the e-degree:

- $n_s < d(\mathcal{C})$ - the student has stopped interacting with the courses $c \in \mathcal{C}$ and lacks the potential to graduate, thus renouncing their studies completely. Notice that, in the e-degree scenario, this observation holds because students are not allowed to graduate before the legal length of the online degree[5]. Contrarily, students enrolled in MOOCs or online courses can obtain a certificate even if the course they are enrolled in has not finished yet.

- $n_s = d(\mathcal{C})$ - the student has finished their studies and graduated from the e-degree.
- $n_s > d(\mathcal{C})$ - the student has difficulties completing the e-degree, thus taking extra time to complete the compulsory courses for graduation. This case refers to students that have not finished their exams within the prescribed period of time of the e-degree.

Table 3.1 summarises the used notation and its corresponding meaning.

3.2 Asynchronous Dropout Status Definitions

In the literature, SDP is a binary classification problem. In other words, zero indicates a persisting student and one represents a dropout student. We formulate the dropout problem both in a recurrent (sequential labelling) and plain (without temporal dependence) scenario. Additionally, we discuss the benefits and possible drawbacks of each formulation.

1. *Plain dropout formulation*: The student-platform interactions are independent in time, while in reality, student behaviour may change over time. Given the e-tivities $w_s(t_b, t_f, c)$ that $s \in \mathcal{S}_c$ performs, plain dropout determines whether s drops out or not regardless of how the e-tivities are sequenced in $[t_b, t_f]$. The formulation is easily extendable to the scenario of e-degrees.

2. *Recurrent dropout formulation*: The recurrent dropout formulation consists of using information from previous course phases to decide a student's dropout status. When transiting from phase p_{i-1}^c to p_i^c, we share hidden information to efficiently monitor student behaviours in course c in real-time. Therefore, the label of s in phase p_i^c depends on the activities performed in the preceding phases p_{i-r}^c, $r \in \{1, ..., \chi \leq i-1\}$. The length χ of the window used to consider previous phases determines how much of a student's past behaviour we want to consider. All this information passes as a hidden state to p_i^c.

   The recurrent dropout adaptation to the e-degree scenario is more challenging than the plain dropout formulation. In e-degrees, two courses can happen in parallel. Therefore, there is no clear relationship between two courses unless one is a prerequisite to the other. Moreover, a student s may decide to withdraw from a specific course but not from the whole degree. Nevertheless, without loss of generality, we assume that an e-degree is a giant course divided into semesters ψ_i $\forall i \in [1, N]$ that play the role of the phases introduced before. Thus, the dropout status of s is decided on the entire e-degree instead of per single course.

[5]E-degrees might allow excellent students to graduate before the legal length of the course in some cases.

The works in the literature adopt the previous two dropout formulations according to two different student modelisation strategies. Below we provide their formal definitions. Notice that we abuse the notation in the following definitions by indicating e-degrees, MOOCs, and single courses with c.

Definition 3.2.1 ***Plain dropout:*** *Given a time interval $[t_b, t_f]$, a student s is a dropout from course c if they do not survive until the end of the time span. In other words, s is a dropout if $\exists t_u \in [t_b, t_f]$ s.t. $w_s(t_u, t_f, c) = \emptyset$.*

According to Def. 3.2.1, $w_s(t_b, t_f, c)$ return a portion of the e-tivities of student s. One can monitor the engagement of student s during the entire time span $[t_b, t_f]$. Nevertheless, this definition does not consider a phasic view of the single course nor the inter-course relationship in an e-degree. The dropout label is based on all the e-tivities $w_s(t_u, t_f, c)$ that s does after a certain point in time t_u.

Fei et al. [35] introduce three recurrent dropout definitions that we reformulate according to the abovementioned notation. While originally employed in MOOC contexts, we adapt them to other OLEs such as e-degrees. Moreover, we generalise them by considering a phasic course unit instead of the original weekly time frame adopted. Recall that k_c sequential phases $p^c_1, ..., p^c_{k_c}$ form a course c. The following definitions assume that the e-tivities of s are grouped according to $[b(p^c_i), f(p^c_i)]$ $\forall i \in [1, k_c]$ in course c.

Definition 3.2.2 ***Participation in the final course phase:*** *A student s is a dropout if they do not persist until the last phase, $p^c_{k_c}$, of course c; otherwise, they are a persister. In other words, s is a dropout if $w_s(p^c_{k_c}) = \emptyset$.*

Definition 3.2.3 ***Last phase of engagement:*** *A student s is a dropout if they do not produce any e-tivities after the current phase p^c_i: i.e. s is a dropout if $w_s(p^c_i) \neq \emptyset \wedge \forall j \in [i+1, k_c]$ $w_s(p^c_j) = \emptyset$. Notice that this definition is a generalisation of the previous one: i.e. we emulate Def. 3.2.2 by setting $i = k_c - 1$.*

Definition 3.2.4 ***Participation in the next phase:*** *A student s is a dropout if they do not have e-tivities in the next phase, p^c_{i+1}. Hence, s is a dropout if $w_s(p^c_{i+1}) = \emptyset \wedge i \neq k_c$.*

Definitions 3.2.2 and 3.2.3 predict the final condition of the students. Contrarily, Def. 3.2.4 does not refer to the dropout status for the entire course c. Instead, it has a phasic view of the dropout status[6]. Thus, a student, which does not finally dropout but has $w_s(p^c_{i+1}) = \emptyset$, is a dropout in phase p^c_i. Recall that, in the scenario of e-degrees, phases have the role of semesters.

Table 3.2 summarises the advantages and disadvantages of the above definitions. Def. 3.2.2 disregards any fluctuations in student behaviour during the course phases. The works in the literature use it to design a model that predicts the student survivability at the course end. Def. 3.2.3 requires only a partial view of the performed e-tivities to foretell the dropout label of student s. Nevertheless, because it corresponds to the final status of dropout, it does not preemptively identify

[6]For more clarification, check the example in [35].

Table 3.2. Advantages and disadvantages of the three dropout definitions.

Definition	Advantages	Disadvantages
Def. 3.2.2	- Simple student survivability model	- Final dropout status prediction - No preemptive identification
Def. 3.2.3	- Prediction using a partial view of student e-tivities	- Final dropout status prediction - No preemptive identification
Def. 3.2.4	- Ongoing dropout status prediction - Prediction using a partial view of student e-tivities - Real-time prediction - Temporarily inactive student recognition - Monitoring of student behaviour in each phase	- Complex prediction mechanism required

dropout cases. The last week of engagement might be too late to recover at-risk students. Finally, Def. 3.2.4 is suitable to perform real-time dropout identification and consistent monitoring of the behaviour of s at each phase. Nevertheless, its adoption requires implementing complex predictive strategies.

Contrarily, the adaptation of the definitions in [35] to the scenario of e-degrees implies utilising semesters $p_1^{\mathcal{C}}, ...p_{k_{\mathcal{C}}}^{\mathcal{C}}$ as the phases introduced in Table 3.1. Therefore, all the above definitions now consider coarse-grained student trajectories w.r.t. fast-paced MOOCs. Moreover, since e-degrees contain courses happening simultaneously throughout a particular semester $p_i^{\mathcal{C}}$, the student trajectory corresponding to $p_i^{\mathcal{C}}$ illustrates e-tivities coming from several courses. To the best of our knowledge, not only the literature has not experimented on e-degrees but has also disregarded the intrinsic complexities of complying with e-tivities engendered by the interaction of students with multiple parallel courses.

3.3 Student Modelling

When designing a student dropout predictor, we must select a suitable student model to account for the dependencies between two consecutive phases/semesters. Figure 3.2 illustrates the taxonomy of designing an input modelling strategy for SDP. Specifically, it shows *Plain Modelisation* and *Sequence Labelling*. The former mostly consider demographics, current study-related information, and their data derivations. The latter adopts two different structural approaches that, respectively, exploit temporal series and networks. Additionally, plain modelisation refers to time-invariant characteristics of the raw logs on student e-tivities; meanwhile, sequence labelling considers only time-variant features. Sequence labelling on temporal series is modelled in the form of clickstreams or forum interventions.

Here, we present student modelling features following the taxonomy introduced in Figure 3.2. We begin by analysing plain modelisation methodologies (Section 3.3.1) and next we consider the vaster class of sequence labelling methods, in Sections

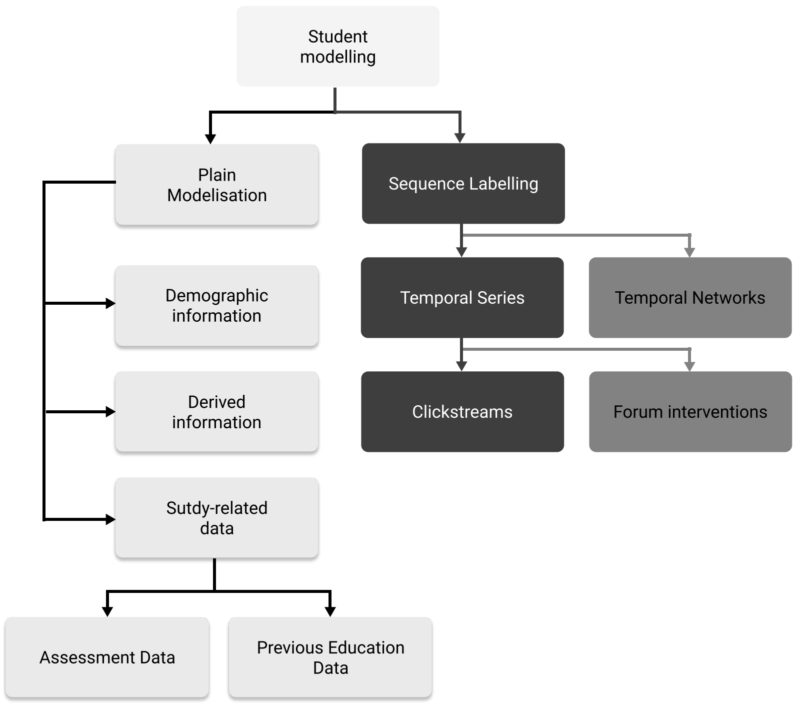


Figure 3.2. A taxonomy of student modelling approaches. The highlighted rectangles depict harder approaches suitable for time-related strategies. The darkest path on the right illustrates the typology of our proposed method.

3.3.2 and 3.3.3.

3.3.1 Plain Modelisation

Plain modelisation exploits student demographic information and data related to the scores of tests/homework that students achieve in each course. Figure 3.3 shows a conceptual workflow for transforming raw data into flattened input features. The two vectors in the upper-left corner represent the demographic and pre-university data, respectively. The matrices depict the interactions of a student in phase $p_i^c \ \forall i \in [1, k_c]$. Data obtained from the e-tivity matrix concatenation of all phases are flattened into the vector f_c containing grades, exam failures, and various cumulative statistical features (steps 1.c and 2.b). Finally, all the vectors are merged (steps 3.a and 3.b) into the input feature vector used for the prediction strategy (step 4). Notice that the majority of works engage in plain modelisation since it requires less complex prediction strategies. To have an idea of the features used when considering this particular input modelling strategy, we provide the reader with some of the most interesting works. Most of the works rely on cumulative statistical features generated from the e-tivities. However, some of them exploit previous education information and current student performance metrics.

SDP studies based on plain modelisation. The authors in [73, 77, 134] use student demography and academic performance. Xenos et al. [134] also exploit student interviews and ad-hoc questionnaires to identify other factors that lead to dropout. The collected data is used to provide additional information regarding a certain

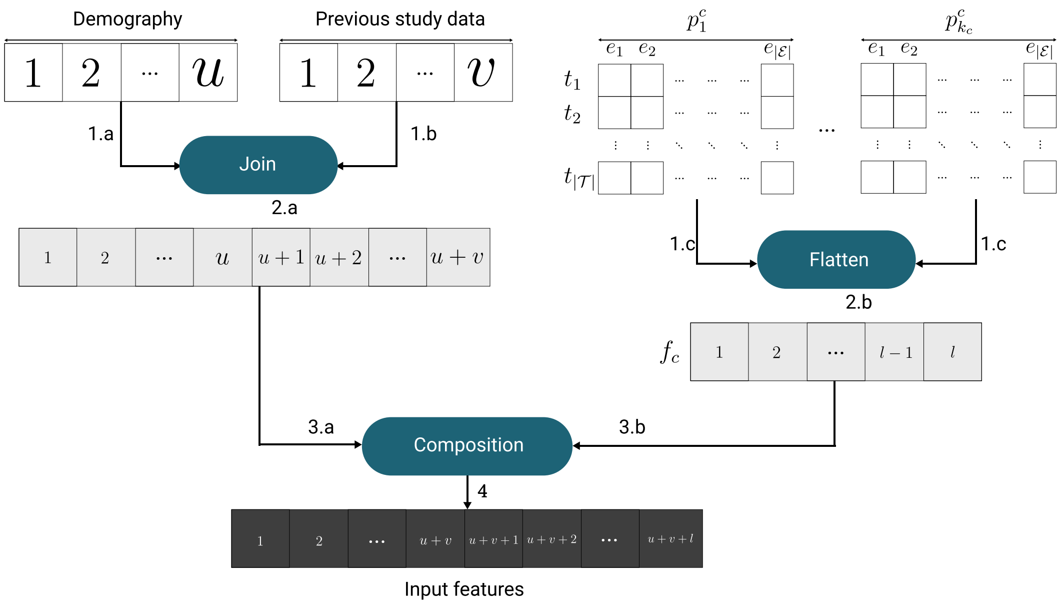


Figure 3.3. Conceptual workflow illustrating the transformation of raw data in a plain modelisation format for student s. Time-dependent e-tivities related to each course phase p_i^c (the matrices in the upper left part of the Figure) are flattened into a vector of study-related features f_c (i.e. statistics derived from all phases). These features are then composed with demographic information and other information related to previous studies (upper left part of the Figure). The result is a flat feature vector which acts as a condensed representation of the interaction history of students.

dropout decision rather than to recognise a persisting student. Kotsiantis et al. [76] extract only demographic data (e.g. background history, ethnicity, first-generation in college) from the course of Informatics in the Hellenic Open University (HOU) to identify struggling students before the midterm of the academic year. The authors in [75] use assessment scores to classify dropout HOU students. They mark a student as failing if they have not submitted one of the two required homework. In this scenario, the authors use the e-tivity matrix on the course phases, and they extrapolate data regarding the performance of students in assignments without using other information from the interaction of students with the OLE. Lykourentzou et al. [88] consider time-varying and invariant attributes by examining demographic and previous study information. Nevertheless, they flatten the time-varying data (i.e. e-tivities) to derive grades, GPA, and the number of activities per course phase. Dekker et al. [30] look at only pre-university data ignoring current performance and interaction patterns in their OLE. Although most traditional universities predict different success outcomes for their students based on demographic data, here, the authors do not have any substantial correlation between background information and dropout probability. Berens et al. [12] build a self-adjusting early detection system targeted for German universities. They employ personal, previous education, and current matriculation information to identify dropouts as soon as the first semester. Kovavcic [78] performs an exhaustive study of the dropout phenomenon by exploring socio-demographic and study-environment variables. Similarly, Al-Radaideh et al. [1] use previous education information and some demographic data to shape their model. Jointly with the student responses to ad-hoc questionnaires for several HarvardX courses, Robinson et al. [113] use general student demographic statistics.

Manhaes et al. [91] propose WAVE, a monitoring architecture for student academic progress. Alongside other student-performance metrics, WAVE exploits exam information (e.g. GPA) to determine whether a student attending the first semester persists in the second one. Although Hu et al. [61] keep track of time-dependent information, they flatten the e-tivities into plain data for statistical analysis. Similarly, Gaudioso et al. [41] combine demographic data with information derived from the student interaction with the e-platform (e.g. total number of visits, number of sessions, percentage use of each of the learning resources). Wolf et al. [133] study the Virtual Learning Environment (VLE) clicks for each student enrolled in the Open University (OU). The authors enrich their dataset with tutor assessment grades, final exam marks, and demographic information.

Amnueypornsakul et al. [3] use quiz- and activity-related information. For each week, they transform the activity-related data in a string consisting of e-tivities, identified with a letter. Gray et al. [45] model a student as a set of η boolean observations indicating the attendance of a student at each point in time: i.e. $z = \{z_1, ..., z_\eta\}$. They utilise an ad-hoc function that combines attendances and non-attendances of a student from the η observations to calculate engagement behaviour of a student: i.e. $BEM_\eta = \sum_{i=1}^{\eta} (-1)^{1-z_i}$. They choose the BEM scores for the first three weeks and demographic features to aid the information gain of their model. Feng et al. [36] extract student behaviour patterns. They engender statistically aggregated data from student clickstreams while simultaneously modelling user and course contextual information. They refer to demographic data for the student context; whereas, they represent the course context with its category (e.g. maths, physics, and arts). Ortigosa et al. [99] integrate user data coming from two different systems to obtain background (socio-demographic data) and static academic information jointly with details about student e-tivities throughout their course intake. Nevertheless, the authors aggregate and flatten this information, thus reducing the time complexity. Isidro et al. [63] extract video-related information such as the number of videos played and time spent watching them. Additionally, they collect assessment data and aggregated information about their interaction in the OLE fora (i.e. number of comments in specific topics).

Finally, it is worth mentioning several studies that, although concerned with physical rather than online degrees, employ a modelisation strategy similar to those adopted for online courses/degrees [2, 23]. Ameri et al. [2] use demographics and pre-enrolment information. Additionally, they collect semester-wise attributes (e.g. GPA, credits passed/failed/dropped) from study-related data such as exams and other curricular activities. Chen et al. [23] exploit the same information as the previous work, but they use a lower quantity of features as input to their forecasting method.

3.3.2 Sequence Labelling with Temporal Series

Sequence labelling consists of shaping the raw data logs into a discrete temporal series consisting of the student e-tivities. A discrete-time temporal series [17] is a set

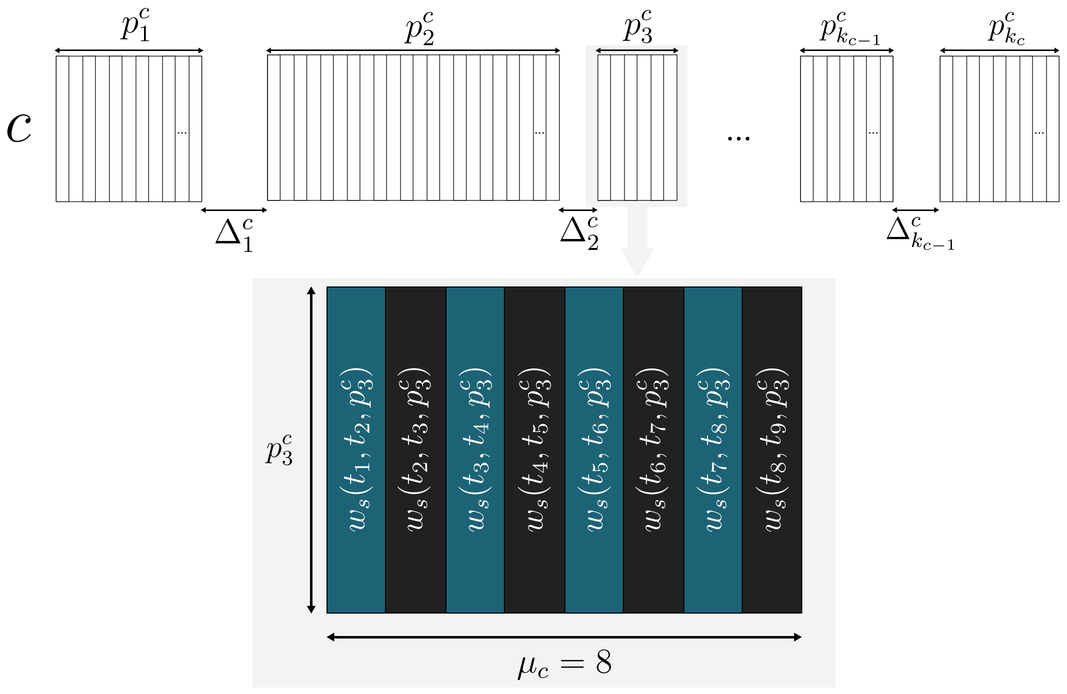


Figure 3.4. Conceptual modelling of a sequential scheme of course c, with an example of consecutive and same-width time-slices in phase p_3^c. Note that periods of non-activity (the $\Delta_i^c \; \forall \, i \in [1, k_c)$) may have a different length.

of observations, each one being recorded at a specific time t coming from a discrete set $\mathcal{T}$ of times. Essentially, we divide each phase p_i^c into μ consecutive temporal slices generally of the same duration where the end of the last slice coincides with the beginning of the current one. Figure 3.4 illustrates the segmentation of the phases of course c in time-slices. Notice that there can be gaps between two consecutive phases p_i^c and p_{i+1}^c, here denoted as inter-phase delay Δ_i^c. Additionally, we portray the course phases with different duration to represent heterogeneity among them. In this example, we assume that phase p_3^c has eight consecutive time-slices ($\mu_c = 8$). Based on these information, we can observe $w_s(t_b^j, t_f^j, p_i^c)$ for each student s to predict whether they continues with their studies in that particular time frame $[t_b^j, t_f^j]$, based on previous history. Afterwards, one can scale up and generalise, in a hierarchical manner, for the persistence of s in course c by relying on the information obtained during their persistence in the different phases.

Notice that the model presented in [108] generalises over what the current research studies do. In the literature, the courses have the same duration and no inter-phasic gaps, which are a useful indicator of future dropouts. Moreover, researchers adopt a time window of the same length of the duration of p_i^c. Referring to the notation of the previous example, t_b of the first slice and t_f of the last one correspond, respectively, to $b(p_i^c)$ and $f(p_i^c)$.

Moreover, we provide the reader with a scaled-up version of the student trajectory in a single online course to a long-term e-degree. Recall that an e-degree $\mathcal{C}$ is a collection of courses $\mathcal{C} = \{c_1, ..., c_{|\mathcal{C}|}\}$. We extend Figure 3.4 to represent a more complex scenario by taking into account several courses that can begin simultaneously.

Without loss of generality, we suppose that the courses of an e-degree belonging to the same semester begin in the first day of the semester[7]. Moreover, some courses might span over several semesters (e.g. yearly courses). In this case, we alleviate this constraint by considering the course spanning over more semesters as a chunk of per-semester sub-courses. Figure 3.5 illustrates an e-degree composed of $|\mathcal{C}|$ courses belonging to different semesters $\psi_1, ..., \psi_N$. Notice that the student interaction differs not only from one course to the other but also within the same course if it spans over more semesters (e.g. c_1 over the first two semesters of $\mathcal{C}$). As mentioned above, here c_1 is considered as two sub-courses c_1' and c_1'' where the completion of both implies successfully passing c_1. Now, consider course c_1. Notice how, in the first semester, it ends before the finish date of the semester - i.e. $f'(c_1') < end(\psi_1)$ - and it starts after and ends before the allotted time in ψ_2 - i.e. $b(c_1'') > begin(\psi_2) \ \wedge \ f(c_1'') < end(\psi_2)$. To tackle "spurious" courses (i.e. those that begin after the start of the semester or end before the finish of the semester), we pad the student time series with leading/trailing zeros to indicate inactivities. Although this time-series alignment simplifies the overall input modelling strategy, it introduces a bias in the prediction model. Nevertheless, because all students will have leading/trailing holes in their time series, temporal deep learning models are resilient enough to understand this manually introduced bias as it does not change the prediction outcome of the dropout status.

Without loss of generality, we provide Figure 3.6 as a simple representation of the input model. Here, we refer to a single online course, but the model can be easily extended to e-degrees by replacing the course phase notion with semesters $\psi_1, ..., \psi_N$. The squares in the Figure depict the course phases, p_i^c. They contain the e-tivities of student s, $w_s(p_i^c)$. Notice that they can also include aggregation features calculated upon the e-tivities instead of just $w_s(p_i^c)$. Differently from Figure 3.4 that considers time-frames within a specific course phase p_i^c, we exploit the whole engagement patterns for s in p_i^c to engender their dropout probability.

We have identified two main approaches to temporal modelisation: i.e. *clickstream-based* and *forum intervention-based*. However, a minority of works in the literature uses other approaches derivations of the previous two. As the name suggests, a clickstream-based schema is based on e-tivities related to clicking resources such as viewing learning resources and videos. In each course phase $p_i^c \ \forall i \in [1, k_c]$, the student trajectory s comprises of same-type click e-tivities grouped together according to an aggregation function (e.g. average, variance, count). Notice that clickstreams do not include forum discussions, such as commenting, starting threads, or assignment/project grades.

Nevertheless, they incorporate forum click-events, such as liking or disliking forum comments, and data related to assignment submissions. Contrarily, a forum intervention-based schema depends on information gathered from student discussions with their peers/tutors. Data used in this schema comprises both structural (e.g., thread starters, thread comments and replies) and temporal information. Generally, time series from the forum-derived data of s include statistics or NLP-derived metrics

[7]In the real world, although courses belong to the same semester, they might begin in different days, thus not permitting the students to interact with the course material until the effective begin date.

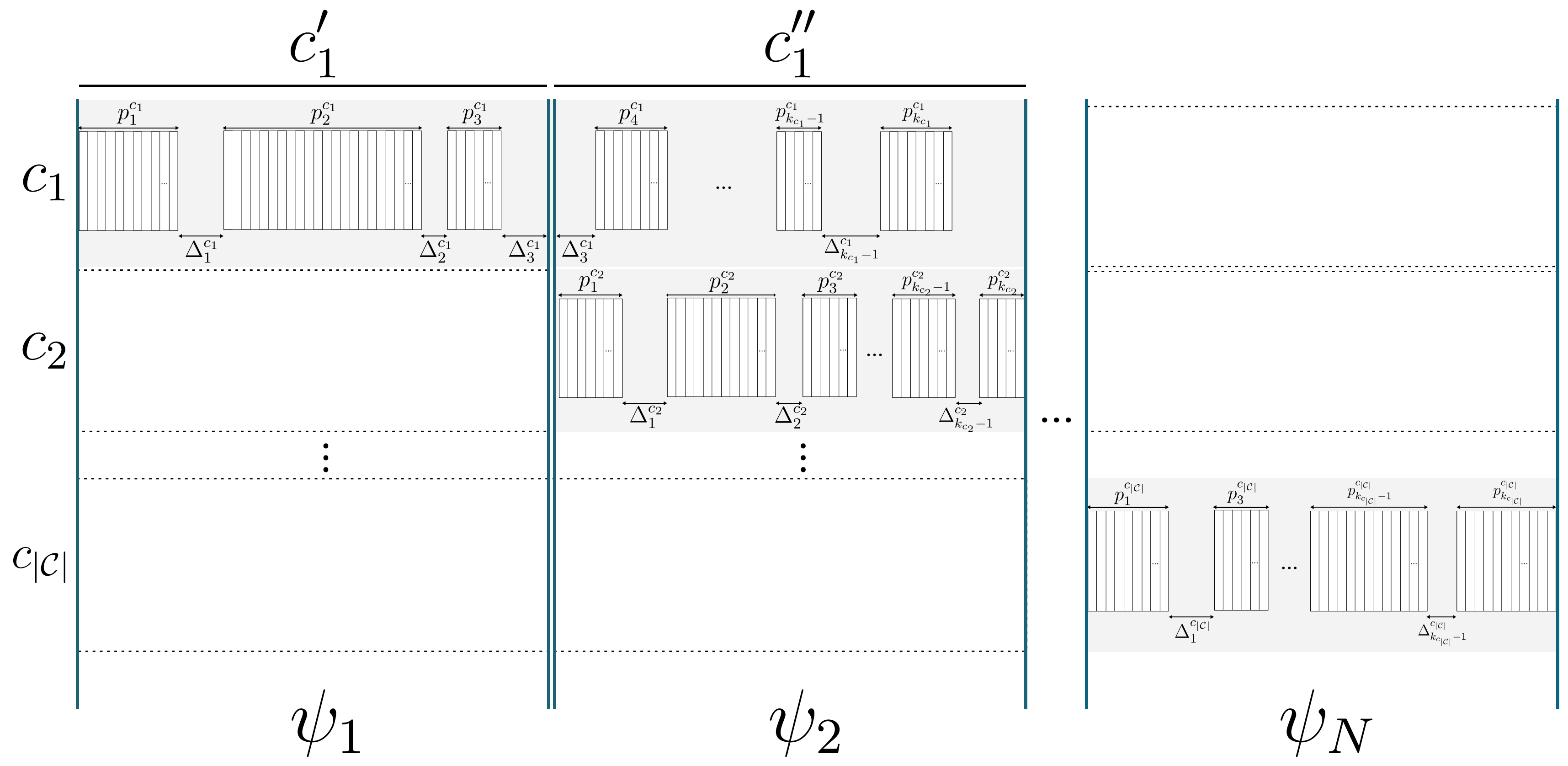


Figure 3.5. Conceptual modelling of a sequential scheme of e-degree $\mathcal{C} = \{c_1, ..., c_{|\mathcal{C}|}\}$, with a spurious course c_1 spanning over two semesters, hence, divided into two sub-courses c_1' and c_1''. Note that periods of non-activity (the $\Delta_j^{c_i}$ $\forall\ i \in [1, |\mathcal{C}|] \ \wedge \ j \in [1, k_{c_i}))$ may have a different length. Moreover, the duration of each semester ψ_i $\forall i \in [1, N]$ can be different and it is not bound to the real-world six-month time-frame.

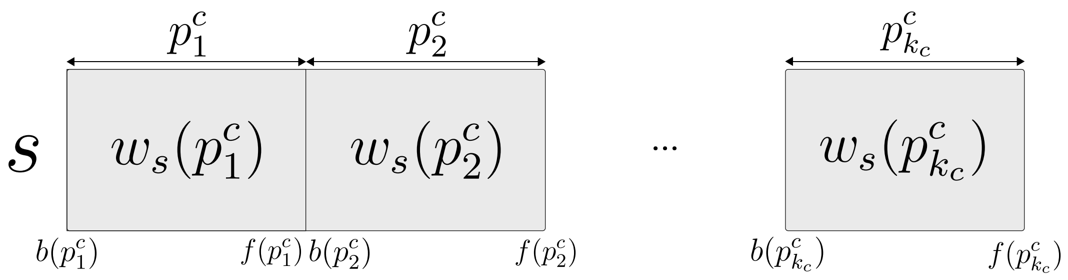


Figure 3.6. A discrete-time temporal series where student s performs $|w_s(p_i^c)|$ e-tivities in phase p_i^c. In the scenario of e-degrees, the phase p_i^c is represented by a semester ψ_j, where $j \in [1, N]$.

(e.g., total number of responses, text length/density) in each course phase. Only a minority of works exploits the structural connections engendered by the interaction of peers in OLE fora.

SDP studies based on clickstream-based schema. Kloft et al. [72] exploit page-view and video-view logs by performing a weekly aggregation (e.g. summing or averaging) of the identified features. Their strategy relies on extracting numerical features by capturing the user activity levels (e.g. number of requests) as well as some technical features coming from the used browser information. Ramesh et al. [112] use lecture views and quiz answers to model two forms of student engagement (i.e. active and passive) derived from user clicks. According to the authors, active engagement occurs when a student s attends lectures and submits quizzes/assignments. Meanwhile, passive engagement occurs when s scarcely attends lessons and does not submit the assigned homework. Hence, by using student commitment as a latent variable for their predictive model, the authors use student submissions to gauge the student survival rate.

Similarly, Nagrecha et al. [95] use clickstream information from video viewing patterns. Each of the data characteristics included in their schema captures latent factors in the consumption of video contents which is the main interaction feature in most OLEs. Jointly with data derived from user clickstreams, the authors in [53, 127] utilise homework submissions and grades. In detail, He et al. [53] compute statistics at the end of a particular phase (e.g. average attempts on each assignment done by week i) to train their weekly-based model. Taylor et al. [127] derive input statistics in the same way as presented in [53], but they use the notions of lag and lead to train their model. Fei et al. [35] exploit lecture views, downloads, and quiz attempts. Li et al. [84] consider behavioural patterns such as accessing, viewing and closing course resources according to their time of occurrence in a certain course phase. Qiu et al. [109] include a list of statistical features such as the number of chapters a student browses and the total time they spend on watching videos. The authors in [21] extract characteristics from weekly learning behaviour records to provide prompt analyses of the student performances. Their performance is illustrated as a metric based on video-viewing patterns (i.e. pause, play, stop) in important lectures before the weekly assignments. Wang et al. [132] distinguish e-tivities coming from clickstream data according to click-source and click-type in a course phase. They transform these data into one-hot vectors and bit-wise sum the vectors belonging to the same time unit feeding their result into the prediction model. Haiyang et al. [46] generate three time-series based on forum, video-lecture and textual resource clicks. Afterwards, they sum up the numbers of clicks from each module on each day and align the total clicks of students. Ding et al. [33] use clickstream data corresponding to navigation and video lessons such as playing, pausing and stopping. Qiu et al. [110] model a temporal series as a one-dimensional grid data sampled at fixed time intervals. They transform the raw input according to a time window algorithm (DTTW), obtaining a time-behaviour dimensional matrix. The time dimension describes the time sequence relationship. In comparison, the behaviour dimension describes the various behaviours in the corresponding period. Finally, Tang et al. [125] exploit time-augmented sequences of learning resources. In this scenario, the authors have a collection of unique URLs corresponding to the learning resources (e-tivities) of their OLEs and the time accessed by each student.

SDP studies based on forum intervention-based schema. Models based on forum-derived features are suitable for distinguishing hugely engaging students from those that exhibit a less determined engagement. However, according to [42], since only a tiny percentage of students engages in OLE fora, the literature lacks studies that purely base their strategies on forum intervention. Because temporal series cannot model the structural patterns of forum discussions, research exploits forum-derived features to enrich clickstream-based models. Below we report some studies based on clickstreams that use forum features to boost their performances.

Ramesh et al. [112] use sentiment analysis of the forum threads as linguistic characteristics and the structural typology of the network formed by the student discussion (i.e. thread openings, comments, and replies) to capture forum information. They include the subjectivity and polarity of each thread post to their clickstream-based feature vector. Fei et al. [35] investigate forum views, thread initiations, posts, and comments. Finally, the authors in [109] derive features from forum activities. They also consider the homophily correlation of a student expressed as the number of replies received from well-performing students. In this way, the students who received the replies are less prone to drop out.

3.3.3 Sequence Labelling with Temporal Networks

As stated in [127], commenting and replying to threads may suggest a lower dropout susceptibility. To better model student engagement in fora, a few studies have exploited temporal networks, and they capture both structural and temporal information of students that interact with their peers. There are three types of interactions in the forum:

- *Thread initialisation* characterises the action of creating a new argument. A thread θ can have $[0, n]$ comments. We denote with $OP(\theta)$ the original poster of θ as indicated in [139].

- *Comments* are posts directly correlated to the original thread message. A comment can have $[0, n]$ replies.

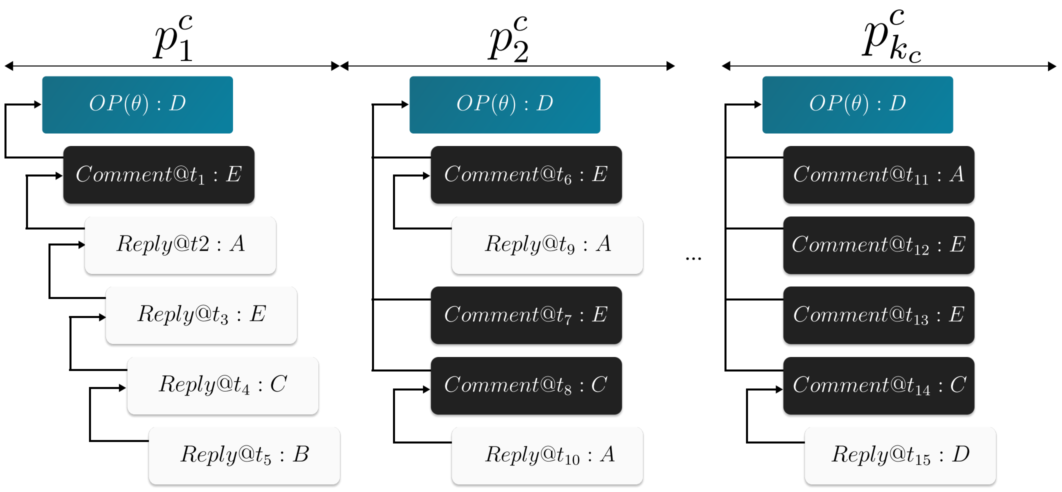


Figure 3.7. Hierarchical view of the forum discussion in each course phase p_i^c. Here we assume that only one thread θ has been opened, and we trace the student interactions throughout the phases. The indentations denote the interactions between two students. In other words, the student in the innermost layer interacts with that in the layer immediately above it. We distinguish between the three interaction types of students; the dark grey represents a thread starter, and the brightest grey a replier. Lastly, the mid-dark grey illustrates a commenter.

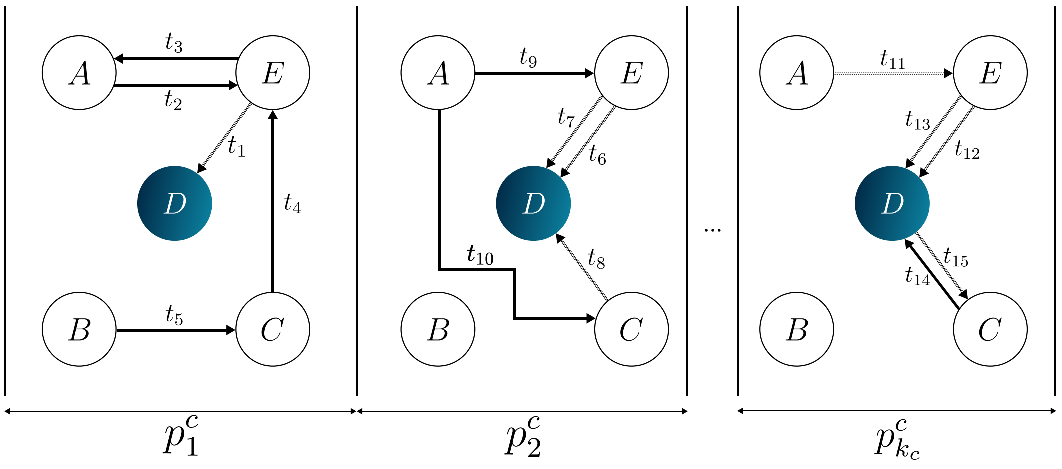


Figure 3.8

- *Reply messages* are responses to the comment messages. A reply message can have other nested replies.

We specialise the temporal-network definitions in [9, 93] for the student interactions in fora to cope with the description of the methodologies that use this particular form of modelisation. We denote with $\Theta_c = \{\theta^1, \theta^2, .., \theta^x\}$ the set of all threads in course c. Recall that $\mathcal{S}_c$ is the set of students enrolled in course c. A forum-based social network for a thread $\theta^r \in \Theta_c$ during a time interval $[t_b, t_f]$ is a labelled multidigraph (i.e. directed graph with parallel edges [9]) $\mathcal{G}^{\theta^r}_{[t_b,t_f]} = (V, A, \ell_A)$ where $V \subseteq S_c$, A is the set of arcs (s', s'') derived from the following interactions:

- s' comments to thread θ^r that s'' has generated, or
- s' replies to a comment of s'' in thread θ^r

and $\ell_A : A \to \mathcal{T}$ is a function that maps the arcs (s', s'') to their relative interaction timestamp t. Clearly the social network interactions of phase p^c_i is

$$\mathcal{G}_{p^c_i} = \bigcup_{\theta^r \in \Theta_c} \mathcal{G}^{\theta^r}_{[b(p^c_i), f(p^c_i)]}$$

and the interactions related to thread θ^r are captured by

$$\mathcal{G}^{\theta^r} = \mathcal{G}^{\theta^r}_{[b'(c), f'(c)]}$$

For completeness purposes, we denote with

$$\mathcal{G}^{\theta^r}_{p^c_i} = \mathcal{G}^{\theta^r}_{[b(p^c_i), f(p^c_i)]}$$

The same formalisation can be extended to the scenario of e-degrees. In detail, the social interactions in e-degree $\mathcal{C}$ can be expressed as $\mathcal{G}_{\mathcal{C}} = \bigcup_{c \in \mathcal{C}} \bigcup_{1 \le i \le k_c} \mathcal{G}_{p^c_i}$. Whereas, the tracing of θ^r in e-degree $\mathcal{C}$ is equal to $\mathcal{G}^{\theta^r}_{\mathcal{C}} = \mathcal{G}^{\theta_r}_{[b'(c_1), f'(c_{|\mathcal{C}|})]}$. Since a particular thread $\theta^r \in \Theta_c$ begins and finishes in course c, then $\mathcal{G}^{\theta^r} = \mathcal{G}^{\theta^r}_{\mathcal{C}}$ $\forall c \in \mathcal{C}$. In other words, it is sufficient to trace θ^r in its belonging course rather than in the entire e-degree.

Figures 3.7 and 3.8 illustrate an example of a temporal social network showing the timed interaction patterns between students. Here, we suppose that the students initiate only a single forum thread in each phase p^c_i. Figure 3.7 depicts the hierarchical structural representation of the interaction of the students (A, B, C, D, E). Notice that, for each phase p^c_i, we depict only the threads opened in p^c_i. We do not want to trace the evolution of the thread influence in each phase but rather check students' engagement in time in different threads. To distinguish the three forum interaction types, we highlight them. We indicate the thread's original poster with a dark shade, a comment with a lighter shade, and a reply with the lightest shade. Figure 3.8 depicts the hierarchical representation transformed into a labelled multidigraph. The labels in the edges depict the time of interaction between its endpoints. Notice that, by examining the phases $p^c_1, ..., p^c_k$, we can model the difference in the density of discussions among peers. For instance, student B is inactive in phase p^c_2. Hence, if B does not participate in the discussion in the successive phases $p^c_3, ..., p^c_{k_c}$, then

B is a potential dropout from the whole course. On the contrary, the other students maintain a linear pattern of interaction with one another; hence, they are likely to persevere.

In conclusion, temporal network models make it possible to exploit network-related metrics (e.g. betweenness centrality [97], PageRank, and HITS [71]) to determine the departure decision for each student in each phase. Furthermore, it is possible to refine the criterion of considering an engaging student in the forum by discriminating their participation in the same thread in different time phases. Accordingly, a student that does not get involved in new discussions in the successive course phases can be considered a dropout. Additionally, by analysing the interaction between at-risk students and highly engaging peers, works exploiting this particular input modelling strategy can fine-tune the predictive model to minimise the false alarm rate.

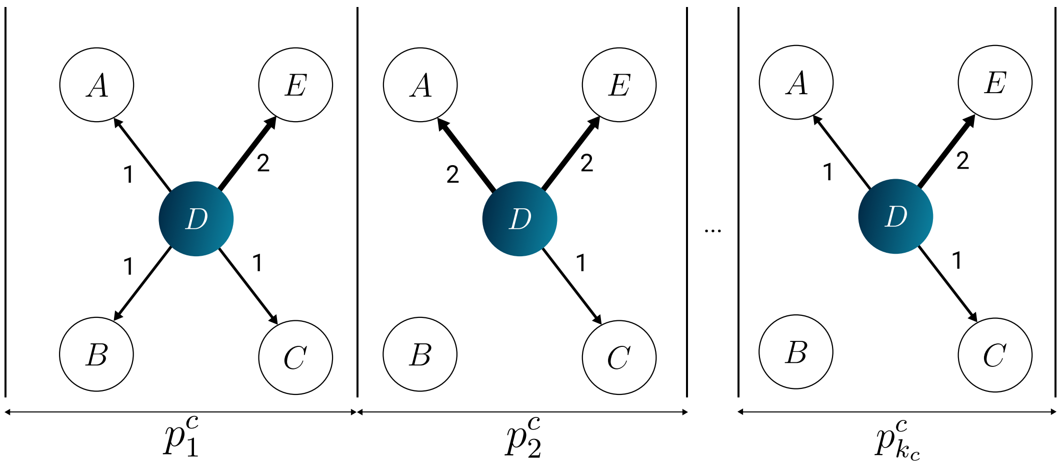


Figure 3.9. Temporal network model highlighting the network structure in each course phase p_i^c according to [136]. Only thread starters (bold circled nodes) have outgoing arcs towards the other students. The edge labels indicate the number of times a thread starter has influenced another student into interacting in that particular thread in the course phase p_i^c. Here, we do not distinguish between comment and reply edges. For visualisation purposes, the thickness of the direct edges depends on the number of interactions between its incident nodes.

SDP studies based on temporal networks. As mentioned above, only a handful of works in SDP base their predictive model on features derived exclusively on peer-to-peer interactions in fora. Yang et al. [136] partition the enrolled students into cohorts according to their registration period. They reveal that students in later cohorts are more isolated while those from earlier ones have a steady interaction rate. They model the relationships between thread starters and other students. In their work, the thread initiators have an outgoing connection to all students participating in the discussion. Therefore, we need to transpose our network structure in each phase p_i^c. Furthermore, the authors do not model parallel edges to indicate multiple interactions between students. Instead, they label the edges with the number of interactions between their endpoints. Essentially, the network that the authors

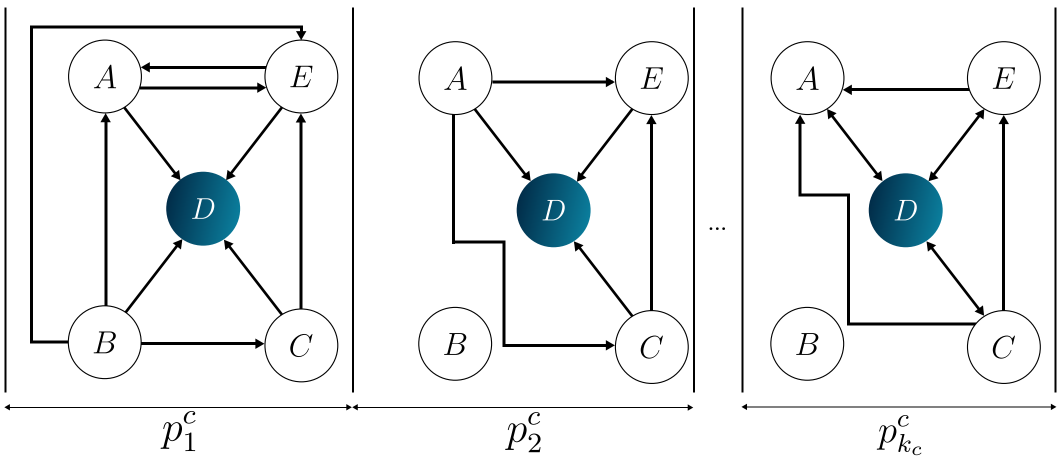


Figure 3.10. Temporal graph in each phase p_i^c according to [18]. We connect each node with all the others that have previously participated in the discussion. For instance, consider phase p_2^c. According to the hierarchical interaction in Figure 3.7, firstly E comments to the initial thread message of D, thus $G_{p_2^c}$ contains the edge (E, D). Then, E comments twice, but since we do not model parallel edges we do not $G_{p_2^c}$ with arc (E, D) again. The comment of C to D induces the edge (C, D). Lastly, we model both replies of A to E and C, respectively, as two edges (A, C) and $(A, E)$the reply of C to the comment of D. The same reasoning stands behind the other phases. The representation of $OP(\theta)$ remains unaltered from the other figures.

generate is equivalent to a transformation from $\mathcal{G}_{c_i}^{\theta^r}$ to a simple directed graph $G_{c_i}^{\theta^r} = (V', E)$ where $E = \{(s, s') | \exists s' \in V \land s = OP(\theta^r) \land s \neq s' \land deg_{out}^{\mathcal{G}}(s) \neq 0\}$ where $deg_{out}^{\mathcal{G}}(s)$ is the out-degree of s in $\mathcal{G}_{c_i}^{\theta^r}$ and the weighted function $\ell_E : E \to \mathbb{Z}^+$ depicts the out-degree, in the original multigraph, of the second element in $e \in E$.

Figure 3.9 illustrates the formalisation adopted in [136], with reference to the hierarchical model introduced in Figure 3.8. The authors rely on social network analysis metrics, each providing different insights into the student engagement patterns.

Gitinabard et al. [42] consider two types of graph generation strategies. Based on [18], the first type connects the students to all those that have previously participated in the discussion of the same thread. In this way, each user has an outgoing edge towards all the others except themselves. This method assumes that every participant in a thread has read all of the preceding posts first and is responding to all of them. The network that the authors generate is equivalent to a transformation from $\mathcal{G}_{c_i}^{\theta^r}$ to a simple directed graph $G_{c_i}^{\theta^r} = (V', E)$ where $E = \{(s, s') | \exists s, s' \in V \land s \neq s' \land min(\ell_A, s).t \geq min(\ell_A, s').t\}$ and $min(\ell_A, s) = \{t | \exists s' \in V \land (s, s') \in A \land t = \ell_A((s, s')) \land \forall s'' \in V \land (s, s'') \in A \to t \leq \ell_A((s, s''))\}$ returns the first timestamp of interaction of student s in thread θ^r. Figure 3.10 illustrates the interaction of users with all the previous participants in the thread discussion as described above. In this case, the authors do not model parallel edges nor they label arcs according to a weight function. Nevertheless, one can extend this network structure by transforming it into a weighted graph.

The second strategy adopted in [42] relies on thread-tracing networks. Based on the observation in [139] that students use commenting and replies interchangeably,

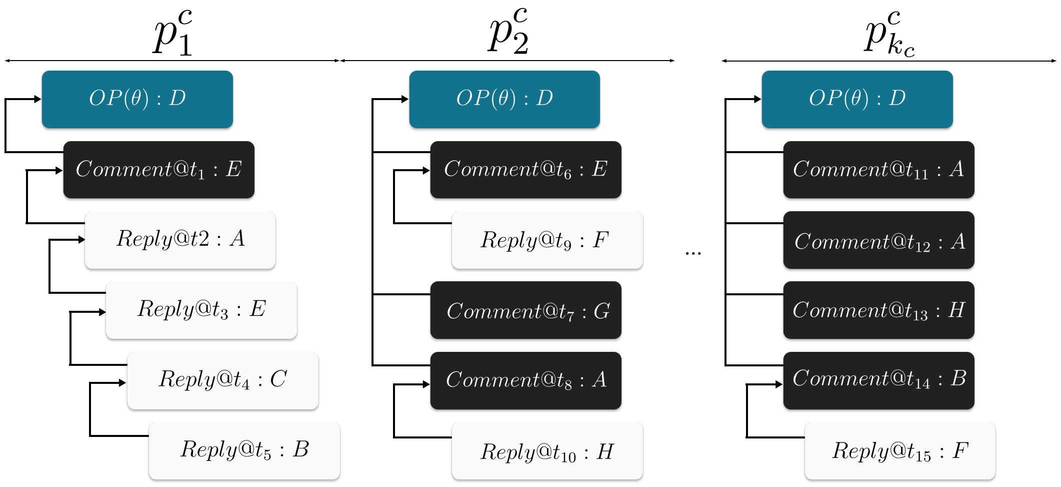


Figure 3.11. Interaction of students in the same thread in different phases. The meaning of the rectangles is the same as in 3.7. To show how the discussion of the same thread differs in the various phases, we assume that the number of students enrolled in c increases (from five to eight in the example) .

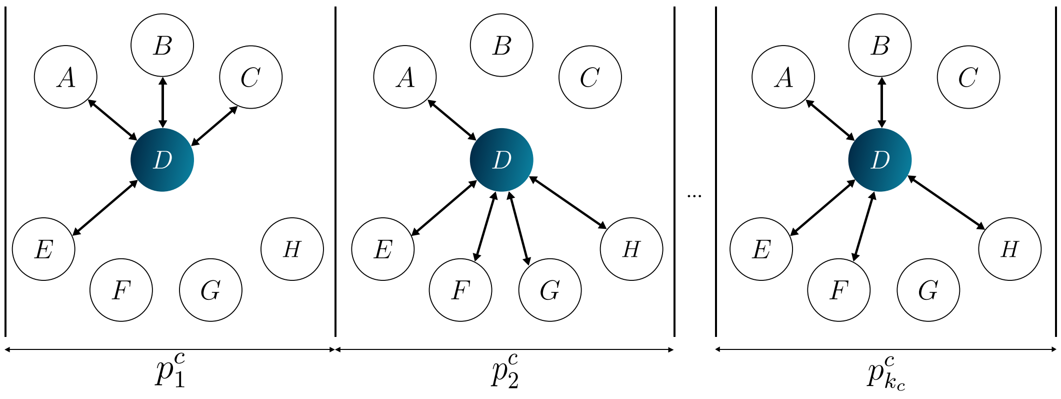
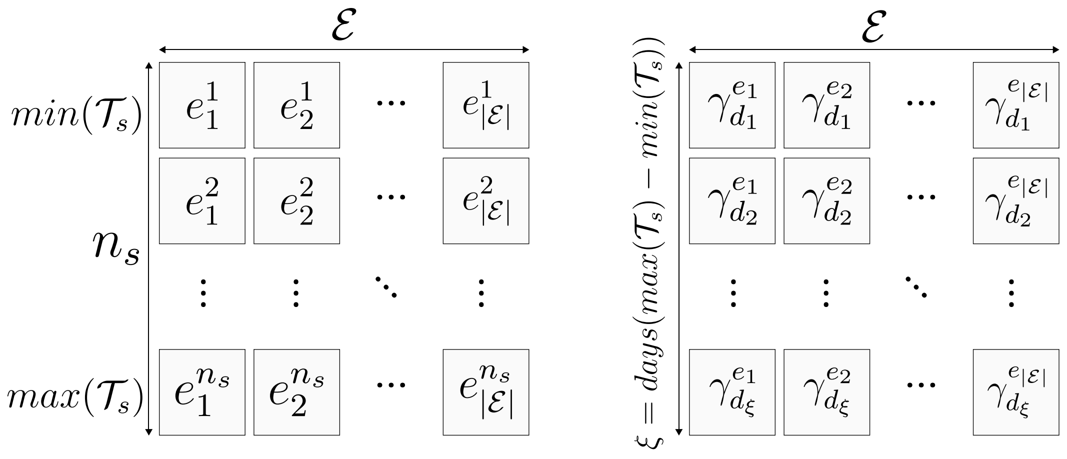


Figure 3.12. An example of time-evolving social network for the same thread θ in all the course phases. The temporal graphs in each phase p_i^c are based on the structure proposed in [139]. Note that the nodes connected to the $OP(\theta)$ - bold border node - form a strongly connected component. The cardinality of the strongly connected component with its centre in $OP(\theta^r)$ indicates the popularity of θ^r. We maintain the bidirectional edges for consistency with our modelisation of $\mathcal{G}_{c_i}^{\theta^r}$.

the authors in [42] connect all the students in a thread to its original poster (initiator). Notice that the network built in this way has undirected edges, but for consistency purposes with $\mathcal{G}_{c_i}^{\theta^r}$ we employ bidirectional arcs. In fact, this network can be seen as a transformation from multidigraph $\mathcal{G}_{c_i}^{\theta^r}$ to a simple graph $G_{c_i}^{\theta_r} = (V', E)$ where $E = \{(s, s')|\exists s, s' \in V \wedge s \neq s' \wedge (s = OP(\theta^r) \vee s' = OP(\theta^r)) \wedge vis(s, s', \mathcal{G}) = 1\}$ and $vis(s, s', \mathcal{G})$ returns $(0, 1)$ according to a graph search (either breadth or depth search) from s to s' in $\mathcal{G}$. In other words, the transformed graph has bidirectional edges between $OP(\theta^r)$ and all the students that have participated in its discussion. According to this strategy, thread starters are at the centre of a star network. Although this approach traces a thread θ in each c_i to assess how the neighbourhood



$\mathcal{E}$
$min(\mathcal{T}_s)$
n_s
$max(\mathcal{T}_s)$
e_1^1 e_2^1 $\cdots$ $e_{|\mathcal{E}|}^1$
e_1^2 e_2^2 $\cdots$ $e_{|\mathcal{E}|}^2$
$e_1^{n_s}$ $e_2^{n_s}$ $\cdots$ $e_{|\mathcal{E}|}^{n_s}$
$\xi = days(max(\mathcal{T}_s) - min(\mathcal{T}_s))$
$\gamma_{d_1}^{e_1}$ $\gamma_{d_1}^{e_2}$ $\cdots$ $\gamma_{d_1}^{e_{|\mathcal{E}|}}$
$\gamma_{d_2}^{e_1}$ $\gamma_{d_2}^{e_2}$ $\cdots$ $\gamma_{d_2}^{e_{|\mathcal{E}|}}$
$\gamma_{d_\xi}^{e_1}$ $\gamma_{d_\xi}^{e_2}$ $\cdots$ $\gamma_{d_\xi}^{e_{|\mathcal{E}|}}$

Figure 3.13. (left) Time-matrix $\mathcal{M}_s^{n_s,|\mathcal{E}|}$ for student s with the e-tivities in the designated cells. For instance, e-tivity e_1^1 corresponds to the first e-tivity type being generated at time $t = 1$. Notice that certain e-tivities cannot be generated in the same time instant, hence, some e-tivities in the same row are $\emptyset$. (right) Time-matrix $\mathcal{M}_s^{\xi,|\mathcal{E}|}$. Here we reference the subscript of every cell with d_i to indicate the i-th day. Each cell (i, j) contains the number of times e-tivity e_j has been generated in day d_i from student s: i.e. $\gamma_{d_i}^{e_j}$. Notice that different students might have different interaction lengths ξ.

of $OP(\theta)$ changes, it can be applied as an commitment signal for the students therein. To show how the neighbourhood of the thread starter changes in time, we base the thread discussion on Figure 3.11. Additionally, Figure 3.12 depicts $\mathcal{G}_{c_i}^{\theta^r}$ based on the hierarchical interaction of Fig. 3.11. Notice how the star network changes structure in each phase. This is useful to verify per-thread student interactions in different course phases. Finally, the authors in [42] use similar metrics as Yang et al. [136] to distinguish persevering students from dropouts.

3.4 Modelisation for Time Gaps and Sparsity Phenomena

Having provided the reading with the necessary background of student dropout definitions and student trajectory modelisation, we exploit the time factor of e-tivities engendered by student interaction with the OLEs. Specifically, with the additional complexity that e-degrees have, we analyse ordered and finite sequences of e-tivities of a particular student s: i.e. $w_s(\mathcal{C})$. Depending on the number of e-tivities that s can perform in an OLE, we distinguish between univariate and multivariate trajectories. The datasets (ref. Section 4) taken into consideration contain multiple interaction ways. Thus, we delve into multivariate e-tivity time-series.

Recall that n_s represents the length of the temporal sequence of student s. For the sake of clarity, instead of having a generic set of timestamp $\mathcal{T}$, here we have a set of timestamps for every student s: i.e. $\mathcal{T}_s\ \forall s \in \bigcup_{c\in\mathcal{C}} \mathcal{S}_c$. Therefore, $min(\mathcal{T}_s)$ and $max(\mathcal{T}_s)$ represent, respectively, the first and last timestamp of the e-tivities that s performs in $\mathcal{C}$. Formally, a multi-variate time-series of student s can be represented as a time-matrix $\mathcal{M}_s^{n_s,|\mathcal{E}|}$. Notice that two distinct time-matrices $\mathcal{M}_{s'}^{n_{s'},|\mathcal{E}|}$ and $\mathcal{M}_{s''}^{n_{s''},|\mathcal{E}|}$

corresponding to student s' and s'', respectively, share the same dimensionality for the e-tivities but not the interaction time-span. To account for problems such as feature sparsity and the curse of dimensionality, we aggregate e-tivities of the same type that occur in the same time-slot. The aggregation function can be an arbitrary function such as the mean time spent of performing a specific e-tivity or the percentage of the total interaction time spent on performing a particular e-tivity. In this study, we choose a simple counting function in a specific time-interval. Additionally, we select a day as the time-interval of aggregation to cope with fine-grained trajectories for extended courses and online degrees. Hence, the previous time-matrix $\mathcal{M}_s^{n_s,|\mathcal{E}|}$ becomes $\mathcal{M}_s^{\xi,|\mathcal{E}|}$ where $\xi = days(max(\mathcal{T}_s) - min(\mathcal{T}_s))$, $max(*)$ and $min(*)$ take the last/first timestamp of interaction for student s and $days(*)$ extracts the number of days of the input time-interval.

Figure 3.13 illustrates both definitions of the time-matrices. Specifically, the left-hand side shows a matrix of dimensions $n_s \times |\mathcal{E}|$ where the cells are the types of e-tivities done at a certain timestamp. Notice that a student s cannot perform some e-tivities at the same time. In order to obtain a more condensed view of the time-matrix $\mathcal{M}_s^{n_s|,|\mathcal{E}|}$, the right-hand side groups together all the e-tivities generated in the same time slot (i.e. according to a daily time-frame) such that the resulting compressed matrix contains ξ rows. We refer to $\gamma_{d_i}^{e_j}$ to the count of all e-tivities of type j generated in day d_i. Here $\gamma_{d_i}^{e_j}$ can be any statistical metric on a daily time-frame based on the generation of e-tivity e_j.

Although the aggregation function γ on a specific time-frame aids in reducing the time gap between two different events, it does not provide a solution for the sparsity in the feature space (i.e. the columns of $\mathcal{M}_s^{\xi,|\mathcal{E}|}$). In other words, students that generate only a few e-tivities per day (e.g. only watch videos) have time-matrices that contain a vast amount of zeros. Therefore, space-reduction (i.e. feature-projection) approaches are useful to cope with the curse of dimensionality. Notice, however, that linear feature-projection techniques are not useful in our context since there is no clear linear dependency among the e-tivity generation. Moreover, the complexity of the feature space arises by introducing the time component, which has a substantial role in the behaviour of the student interaction with the OLEs. Because we do not delve into pre-processing the input data and then applying commonly-known reduction strategies[8], we rely on end-to-end learning to generate a low-dimensional embedding layer that copes with the sparsity of e-tivities in a single day (see Section 5 for more details).

[8]Principal Component Analysis [103], Non-linear Matrix Factorisation [31], and Generalised Discriminant Analysis [10]

Chapter 4

Datasets

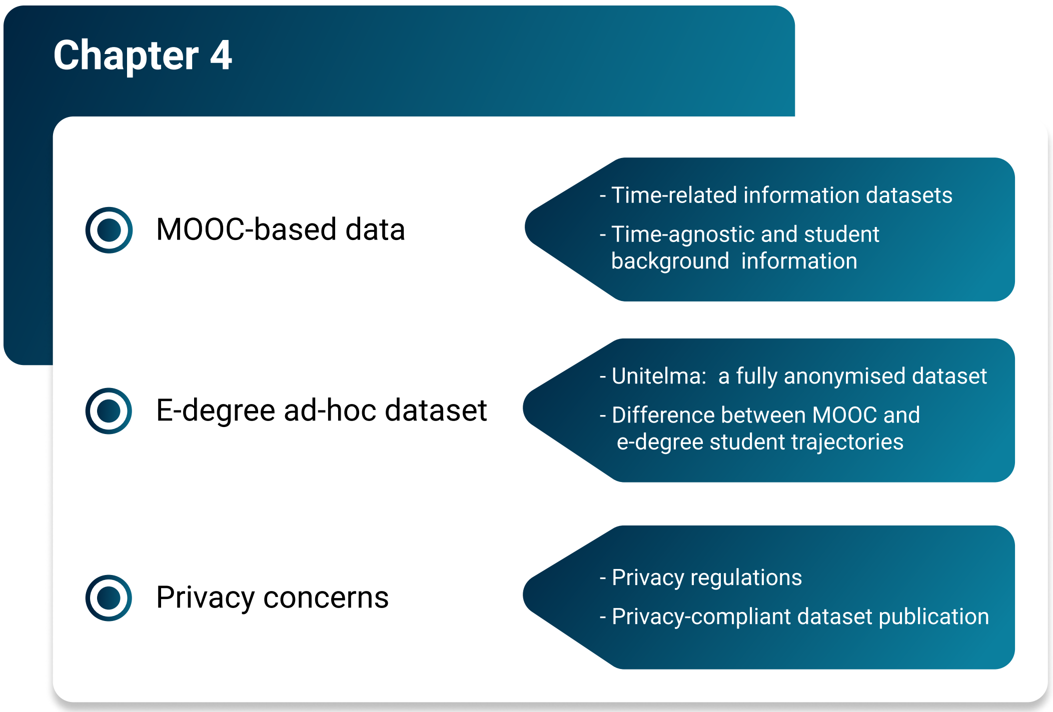

Chapter 4
MOOC-based data
- Time-related information datasets
- Time-agnostic and student background information
E-degree ad-hoc dataset
- Unitelma: a fully anonymised dataset
- Difference between MOOC and e-degree student trajectories
Privacy concerns
- Privacy regulations
- Privacy-compliant dataset publication

Figure 4.1. The Learning Analytics research field employs several datasets, of which only a handful made public. This Chapter gives insight into the available MOOC-based datasets published from well-known OLEs (e.g. edX, Coursera, XuetangX). Furthermore, we delve into analysing an ad-hoc dataset comprising student trajectories in e-degrees. Finally, to bridge the gap between scientific works and available benchmarking datasets, we provide the reader with a data publication schema compliant with privacy preservation regulations such as the GDPR.

In the last decade, the literature of Learning Analytics has received a plethora of contributions with novel strategies to resolve the SDP problem. Nevertheless, only a handful has contributed by publishing their data online and helping the research area build common testing benchmarks[1]. Because academia relies on

[1]In Table 2.3, we provide a full list of datasets used in academia with links to the resources

	Time-related			Time-agnostic
	XuetangX	KDDCup15	Unitelma	HarvardX
Number of e-tivities $\|\mathcal{E}\|$	22	7	97	-
Maximum window length ℓ	30	25	180	-
Number of time-series $\mathcal{S}$	∼24k	∼120k	∼5k	∼301k
Number of courses	19	39	13	6
Mean daily sparsity	90.02%	90.8%	96.92%	-
Mean day gap between e-tivities	10.65 ± 2.07	18.06 ± 1.53	34.6 ± 11.5	-
Class distribution (0:1)	38.7% : 61.3%	20.7% : 79.3%	20.4% : 79.6%	2% : 98%

Table 4.1. Dataset characteristics.

fast-paced MOOCs, here we analyse three different benchmarking datasets. The first two have time-related information included within that permits to construct student time-series of e-tivities. We provide the reader with a third time-agnostic solution that contains only student demography and background information for completeness purposes. Additionally, we analyse an entirely anonymised dataset collected from the interaction of students with e-degrees in the University of Rome Unitelma Sapienza. Here we highlight the differences between the Unitelma dataset and the MOOC-based ones.

Finally, to bridge the gap between a large number of scientific articles and the publication of proprietary datasets, we provide the reader with a step-by-step guide on how to publish sensitive student data while complying with privacy regulations such as the GDPR.

4.1 MOOC-based Data

4.1.1 Time-related Information Datasets

The literature exploits only two datasets that are available online. Both of them encompass fast-paced courses. The first contains anonymised courses from the XuetangX learning platform. XuetangX offers online courses in multiple disciplines and also certificate and degree programs. It was launched in 2013 as the first Chinese MOOC platform initiated by Tsinghua University and MOE Research Centre for Online Education. With more than 2,300 courses covering over 13 different categories, it has more than 5.3 million subscribed learners. To expand their reach to non-Chinese territories, recently, it has integrated full support in English of their OLE. The second covers more OLEs from edX, a MOOC created by Harvard and MIT. Additionally, it was part of the annual Data Mining and Knowledge Discovery competition organised by ACM Special Interest Group on Knowledge Discovery and Data Mining, the leading professional organisation of data miners. Hereafter, we denote with XuetangX[2] and KDDCup15[3], respectively, the first and second dataset.

where applicable. Notice that the majority of the works does not publish their proprietary student information due to institutional policies.

[2]`http://moocdata.cn/data/user-activity`

[3]`https://data-mining.philippe-fournier-viger.com/the-kddcup-2015-dataset-download-link/`

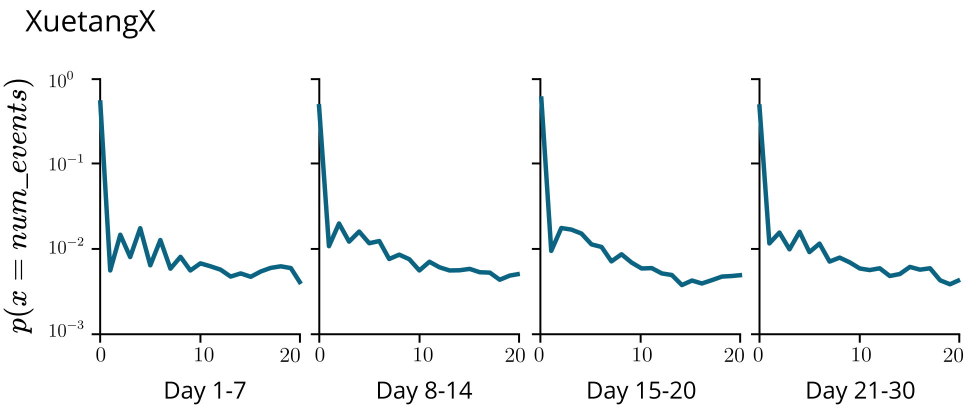


Figure 4.2. The probability distribution of e-tivities in XuetangX in different time-slices of the entire time-window $\ell = 30$. The y-axis is log-scaled. The x-axis represents the exact number of e-tivities present in all the time-series of the dataset.

For XuetangX, we prune all e-courses with less than 350 student trajectories[4], which leaves us with 19 courses overall, whereas in KDDCup15, all courses are sufficiently populated. Additionally, notice that we extract the student trajectories of the datasets from raw log files where every timestamp corresponds to a particular student interaction engendering a specific e-tivity. Hence, the student trajectories are globally aligned according to their interaction time. MOOCs have a simpler nature w.r.t. e-degrees since students are enrolled from the first day that the course commences and, similarly, they are granted a certificate - if they are persisters - at the end date of the course[5]. In other words, since we have a short-term scenario where the start and end dates are globally shared among the students, we can align each student in the time-axis and monitor their e-tivities.

Table 4.1 summarises the characteristics of the datasets in terms of e-tivities, the number of time-series (i.e. the number of students enrolled) and the maximum window length ℓ considered as interaction history for prediction purposes. Notice that ℓ is shorter than the total duration - in days - of the students with the OLEs. We stop our history time window before the end of the courses because we want to predict the dropout status as soon as possible. Notice that the daily sparsity in the time-matrix $\mathcal{M}$ is high. This phenomenon is expected due to a large number of dropout students within the first few course phases [55, 121]. Additionally, students can get a certificate by completing certain tasks and using only a small fraction of e-tivities. Hence, the high mean daily sparsity on persisters. Furthermore, in MOOCs, students have immediate access to the course materials at the beginning of each phase $p_i^c \; \forall i \in [1, k_c]$ inducing them to complete the assignments in the first few days, leaving their trajectories empty for the remaining duration of p_i^c. Thus, besides having a mean day gap between e-tivities of ~ 11 and ~ 18 days, respectively, for

[4]The minimum number of trajectories has been experimentally selected to reflect the best performances on the overall dataset.

[5]In e-degrees this scenario is optimum. Nevertheless, as stated in Section 3.1, students can enrol and graduate at any point in time, thus having fewer time constraints as in MOOCs.

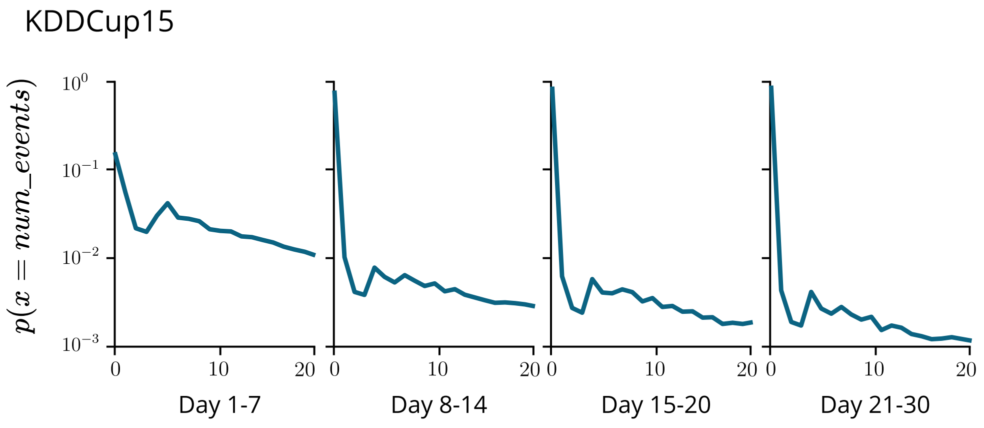


Figure 4.3. The probability distribution of e-tivities in KDDCup15 in different time-slices in the entire time-window $\ell = 25$. The y-axis is log-scaled. The x-axis represents the exact number of e-tivities present in all the time-series of the dataset.

XuetangX and KDDCup15, persisting students have sporadic engagements with the OLEs with, respectively, ~ 8 and ~ 16 days between interactions. As expected, both XuetangX and KDDCup15 are composed of a majority of null values in the time-matrices $\mathcal{M}_s^{\xi,|\mathcal{E}|}$. Finally, Figures 4.2 and 4.3 illustrate the probability distribution of e-tivities in specific time-slices of the time-window ℓ for XuetangX and KDDCup15, respectively. Notice that the distribution of e-tivities remains unaltered in time because most students decide to drop out around the end of the first week for various reasons (e.g., course complexity rises, time management, underestimation of workload). The first week of both datasets consists of a more stable interaction of students with the OLE, with only a small percentage of inactive students from the beginning of the courses. However, the course first week is more determinant in discriminating between dropouts and persisters in KDDCup15 than XuetangX. In the latter, students who hardly ever interact with the OLE constitute more than 80% of the entire population, and, since this percentage has a non-decreasing trend, it is safe to state that the decision to drop out shrinks to the first two days. This phenomenon is expected since most of them enrol to check the arguments treated in the course. Finally, the class imbalance present in both datasets certifies that fast-paced courses need to restrict their enrolment policies or place regulations and consequences on their dropout students besides not refunding the minuscule enrolment fee[6].

4.1.2 Time-agnostic and Student Background Information

Although we focus on time-related and sequential datasets, for completeness purposes, we provide the reader with a time-agnostic dataset that contains student background information and current student performance (e.g. GPA and assignment submissions).

[6]In some OLEs, the initial enrolment fee is reimbursed within a certain period if students do not enjoy or find the course useful for their career/further education.

HarvardX [50] [7] is comprised of de-identified data from the first year (autumn 2012 - summer 2013) of Harvard courses on the edX platform.

Each row of the dataset represents a specific student s enrolled into a particular course c. Every row comprises aggregated data (e.g. last interaction date, the total number of events, number of active days) that s has performed in c. Moreover, the dataset provides some personal information such as gender, GPA, world region of origin, and previous education level if applicable. Recall Table 4.1. Because HarvardX does not contain time-related information and data related to the generation of e-tivities and types of student-OLE interactions, the e-tivity set $\mathcal{E}$ is empty. For the same reason, we cannot consider a time-window length as historical data to predict the dropout status. Moreover, since each row represents a particular student s, we cannot calculate the daily sparsity nor the temporal gap between two e-tivities. Finally, having the largest population of enrolled students (i.e. $\sim$301k) spread only on six courses, HarvardX has the highest dropout rate (i.e. $\sim$98%) among all the datasets[8]. Notice that HarvardX already comes with a plain input modelling strategy that permits researchers to assess whether demographic information and cumulative e-tivity data impact the probability of dropping out.

4.2 E-degree Ad-hoc Dataset

Recall from Section 2.2.2 that one of the most challenging aspects of SDP is the nature of the student trajectories. To the best of our knowledge, no work has treated the long-term dependency among e-tivities, especially in slow-paced and temporally elongated online degrees. This section provides a full understanding of an ad-hoc dataset encompassing e-tivities generated from different e-degrees at the University of Rome Unitelma Sapienza. The dataset is fully anonymised to comply with the institution's policies following a privacy preservation schema described in Section 4.3.

4.2.1 Unitelma: A Fully Anonymised Dataset

Unitelma includes data collected from 13 different e-degrees held from 2010 to 2018 with $\sim$5k students enrolled therein. To adhere to the restrictive regulations that the proprietary institution has, we publish the dataset online in an aggregated form containing only necessary information about the temporal series for each student[9]. Table 4.1 shows that Unitelma has the highest diversity of e-tivities with the lowest number of trajectories. This aspect is expected since MOOCs, for their agile structure, limited time commitment and low cost are attended by at least

[7]Researchers interested in this dataset need to fill a form at `https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/26147` to access the data. We hereby declare that we filled this form as of 07/06/2021 and own the dataset. The observations made in this Section come from our insight and statistical analyses generated on our server.

[8]We refer the reader to `https://colab.research.google.com/drive/1bJp5_GCgcug66aawwBIU5JJBLdyGep02?usp=sharing` for further analyses on HarvardX. Notice that we do not include the dataset in the directory of the shared notebook file. Researchers interested in the dataset need to download it from the previous link at Harvard's institution.

[9]The anonymised time-series of the dataset can be downloaded at `https://figshare.com/articles/dataset/UnitelmaSapienza_1_0_zip/14554137`.

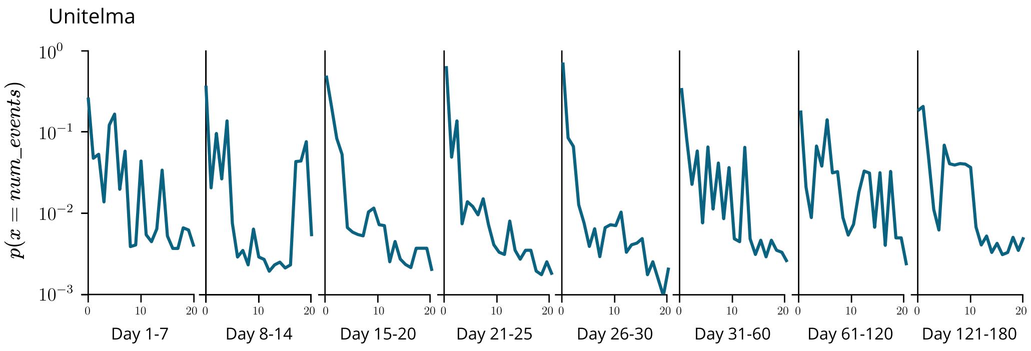


Figure 4.4. The probability distribution of e-tivities in Unitelma in different time-slices in the entire time-window $\ell = 180$. The y-axis is log-scaled. The x-axis represents the exact number of e-tivities present in all the time-series of the dataset.

one order of magnitude higher number of students w.r.t. online degree programs. Because the majority of the students enrolled in an OLE - may that be a single course or an e-degree - exploit only a fraction of the entire e-tivity set to advance throughout the course phases, we expect Unitelma to have the largest sparsity in the feature dimension $\mathcal{E}$. Additionally, having to cope with extended temporal series, the maximum window length to consider as historical data for the predictive models enlarges to one semester. In many institutions, a semester is considered the least amount of time before labelling students as dropouts. According to the previous observation, the average daily gap between two e-tivities in a particular student time-series is larger w.r.t. MOOC OLEs (i.e. 30 days of inactivity on average).

Similarly to Figures 4.2 and 4.3, we illustrate the temporal probability distribution of finding u daily events, no matter their type, in a randomly selected trajectory $M_s^{\xi,|\mathcal{E}|}$ in all the 13 e-degrees in Unitelma (see Figure 4.4). Unitelma has a bell-shaped temporal probability distribution (i.e., a specular trend in e-tivity generation as time passes). Hence, unlike the MOOC datasets, students seem to have a period of limited interaction at the beginning of the course, probably due to the difficulty of adapting to the rules of the degree course and the use of teaching tools, after which they present a somewhat more constant interaction behaviour. We expect this shift in the generation of e-tivities improves the performances of sequential methods, able to capture long-term dependencies, while it worsens standard machine-learning strategies.

4.2.2 Difference between MOOC and E-degree Student Trajectories

Because the raw input of student trajectories between MOOCs and e-degrees is different, we provide details on coping with these challenges and have a unique input representation for the predictive models. Figure 4.5 depicts the difference between the modelisation in MOOCs (up) and e-degrees (down). Here, we concentrate on the upper part of the Figure. The interaction of students stretches from the beginning $b(c)$ to the end $f(c)$ of course c. As mentioned before, in the scenario of MOOCs, we assume that each student enrols at the beginning of the course (i.e. $min(\mathcal{T}_s) = b(c)$)

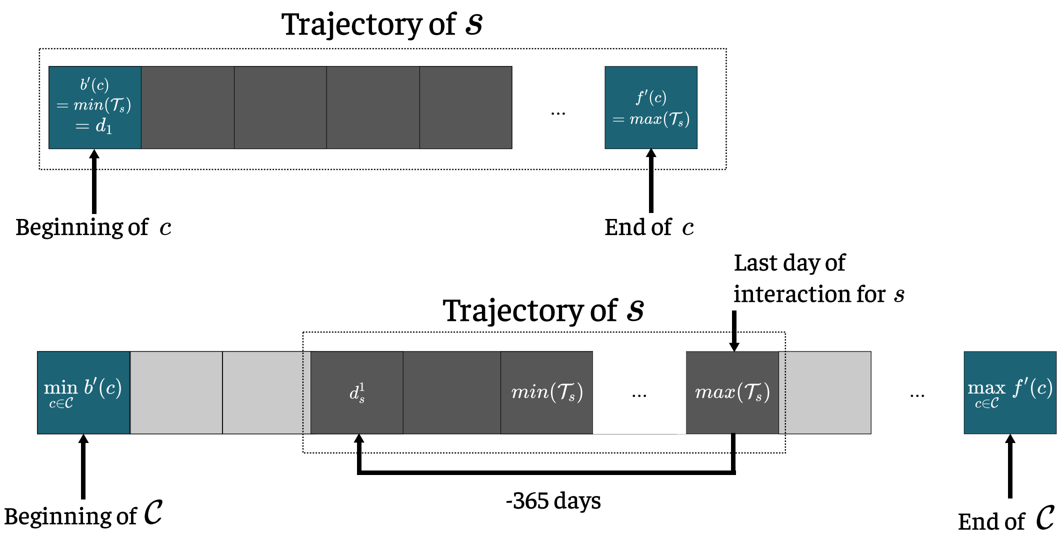

Trajectory of s
$b'(c) = min(\mathcal{T}_s) = d_1$
$f'(c) = max(\mathcal{T}_s)$
Beginning of c
End of c
Last day of interaction for s
Trajectory of s
$\min_{c \in \mathcal{C}} b'(c)$
d_s^1
$min(\mathcal{T}_s)$
$max(\mathcal{T}_s)$
$\max_{c \in \mathcal{C}} f'(c)$
Beginning of $\mathcal{C}$
-365 days
End of $\mathcal{C}$

Figure 4.5. Differences between the student trajectories in the MOOC scenario (up) and in Unitelma e-degrees (down). The rectangles represent the different days in the e-degree $\mathcal{C}$ and MOOC c. Notice that the trajectory of student s in $\mathcal{C}$ is aligned locally according to the last day of interaction of s.

and their last e-tivity is recorded at the end of the course (i.e. $max(\mathcal{T}_s) = f(c)$). Therefore, the ξ dimension of the time-matrix $\mathcal{M}_s^{\xi,|\mathcal{E}|}$ for a student s belonging to the dataset XuetangX/KDDCup15 is equal to the duration of the course in days, $days(f(c) - b(c))$. While the series of MOOCs are global (i.e. the start of the e-tivity data for all students matches the course start date), those for an e-degree - in our case, Unitelma - are local. In other words, students may enrol in an e-degree and start/pause/restart their studies at any time. Therefore their time series are not aligned, a common scenario in entirely online degrees, mainly attended by working students who can devote themselves to study depending on their work commitments. We consider the date of their last observed e-tivity and collect one year in the past for these students. In this way, we align the trajectories locally. The lower part of the Figure depicts the method to generate the student time-series for Unitelma. For each student s, we find the timestamp of the last e-tivity generated, $max(\mathcal{T}_s)$, and fetch 365 days of the history of s. Thus, $d_s^1 = max(\mathcal{T}_s) - 365$ represents the first day of the series. In this scenario, a student might generate their first e-tivity at any given time after the beginning of the e-degree (i.e. $min(\mathcal{T}_s) \geq \min_{c \in \mathcal{C}} b'(c)$) and might record their last e-tivity at any time before the end (i.e. $max(\mathcal{T}_s) \leq \max_{c \in \mathcal{C}} f'(c)$). Hence, we need to take care of two different scenarios:

- The first day in the temporal series is prior to the first generated e-tivity by s (i.e. $d_s^1 < min(\mathcal{T}_s)$) - In this case, we need to prepend $min(\mathcal{T}_s) - d_s^1$ null vectors to $\mathcal{M}_s^{\xi,|\mathcal{E}|}$. Formally, $\mathcal{M}_s^{365,|\mathcal{E}|} = [O^{min(\mathcal{T}_s) - d_s^1, |\mathcal{E}|}; \mathcal{M}_s^{\xi,|\mathcal{E}|}]$ where $O^{x,y}$ is the zero matrix of $x \times y$ dimensions, and $[*;*]$ concatenates its parameters on the first dimension.

- The first day in the temporal series is posterior to the first generated e-tivity by s (i.e. $d_s^1 \geq min(\mathcal{T}_s)$) - Here, we need to eliminate the first rows up to d_s^1 from

$\mathcal{M}_s^{\xi,|\mathcal{E}|}$. Formally, $\mathcal{M}_s^{365,|\mathcal{E}|} = del(\mathcal{M}_s^{\xi,|\mathcal{E}|}, 0, d_s^1)$ where $del(\mathcal{M}, x, y)$ eliminates the rows from x to y in matrix $\mathcal{M}$.

4.3 Privacy Concerns

Nowadays, every transaction that involves sharing/exchanging personal and sensitive data has to be protected following well-known privacy regulations, especially in the EU community. Because data is a primary concern of OLEs, and preserving the anonymity of e-tivity generations by students is important, disclosing proprietary information online is a cumbersome task. This challenge becomes even more problematic since the majority of the works in the literature copes with sensitive data (e.g. student biography and demography information). Managing this kind of data is complementary but a crucial aspect of the SDP, specifically if the datasets need to be published online for further experiments. Hence, in this Section, we provide insight on concerns and policies that help avoid the violation of students' privacy according to regulations of the operating territory.

The works in the literature exploit data either coming from their institutions or well-established and notorious MOOCs (e.g. Coursera, edX, and Udemy). Hence, they defer the legal responsibilities to the body of the data property. The US lacks a comprehensive federal law that regulates the collection and use of personal information. Instead, the government has approached privacy and security by adjusting only specific sectors and types of sensitive information, creating overlapping and contradictory protections[10]. Meanwhile, the European Union (EU) General Data Protection Regulation[11] (GDPR) intends to[12]:

- Harmonise data privacy laws across Europe.
- Protect and empower all EU citizens data privacy.
- Reshape the way organisations across the region approach data privacy.

The GDPR is a legal framework that sets guidelines for collecting and processing personal information from the EU. Since the regulation applies regardless of where websites are based, it must be heeded by all sites that attract European visitors, even if not targeted to market goods or services to them. Additionally, it mandates that EU visitors be given several data disclosures [131]. The site must also facilitate such EU consumer rights as a timely notification if personal data is breached.

To provide a reliable schema that preserves the integrity and anonymity of the student information, in Section 4.3.1, we provide some hints in case researchers would like to share their collected data with the SDP community.

4.3.1 Privacy-compliant Dataset Schema

Having to deal with vast quantities of transactions (i.e. e-tivity generations) every particular time instant for every student enrolled in OLEs, storing and, then,

[10] `https://www.cfr.org/report/reforming-us-approach-data-protection`
[11] `https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en`
[12] `https://eugdpr.org/`

processing this information is arduous. Therefore, institutions (see as an example HarvardX) need to select what information to disclose to the public and leave unpublished. The following are the fundamental entities in SDP that should be protected:

- *Students* register in e-platforms with their personal (e.g. demographic data) information generally anonymised. They can interact in the course forum, giving sprout to discussions of a specific argument. Notice that their interaction should be protected as well.

- *Courses* are allotted in phases, each containing several resources (e.g. videos, quizzes, reading lectures). Therefore, e-degrees and the courses therein must be anonymised as well. Additionally, relationships such as prerequisites between different courses should be maintained ciphered to prevent information leakage about the syllabus of the e-degree such that malicious users are not facilitated to extract sensitive information from other available resources.

- *E-tivities* are the activities produced by events that students cause during the interaction with the OLE. The event that generates an e-tivity should be protected since it can leak information such as the resource interacted with, the grade in the assignment, and the browser/OS/IP data. According to Tankard et al. [126], session and navigation information is bound to the same regulation as other sensitive data. Moreover, when treating forum discussions, researchers need to cypher the thread creations and arguments conducted therein. In other words, the content of each of the interactions (i.e. comments and replies to a thread θ) needs to be embedded into a latent space such that one cannot reverse engineer the information about the course at hand.

To publish an SDP dataset, one should examine whether privacy policy infringement has happened or not. Authors should protect as much data as possible to avoid leaking sensitive student information. Nevertheless, some information should not be ciphered for interpretability purposes. For instance, the type of e-tivities[13] provides tremendous interpretability power. Hence, one can stabilise a correlation between the e-tivity type and the dropout decision. Student personal information such as age and prior education could be left unprotected since they do not provide access to the identity of a student. This particular type of feature can only allow others to categorise students into cohorts without infringing their privacy. Nevertheless, as Sweeney et al. [124] suggest, grouping continuous attributes into several bins (e.g. instead of age, one could have bins that fall into categories such as youth, middle-aged, elderly) reduces the risk of identification. Finally, a fully anonymised dataset can enrich the literature with a standard benchmark for evaluation purposes, but it would not efficiently explain why students fail at persisting throughout a particular course/e-degree.

[13]There are several types of e-tivities among which forum-related, video-related and navigational ones.

4.3.2 Case Study of a Privacy-Preserving Dataset for Data Mining

Privacy-Preserving Data Mining (PPDM) is a multi-disciplinary research area that consists of prohibiting/minimising the leakage of sensitive information in data mining. PPDM transforms the original data so that the data mining processes applied to it do not violate privacy constraints. While protecting sensitive information, PPDM allows data mining techniques to achieve promising results. Saranya et al. [116] categorise PPDM techniques into three classes:

- *Randomisation* uses data distortion strategies to create hidden representations of the original data - in this case, records of e-tivities. Here, the individual interaction events cannot be recovered, but their aggregate distributions are recoverable.

- *Anonymisation* protects an individual's direct identity by using obfuscation techniques. Although there is no cypher method employed at this stage, anonymisation is a procedure that helps select the information to disclose. According to Malik et al. [90], there are several hierarchical levels to the importance of an attribute: i.e. explicit identifiers, quasi-identifiers, sensitive attributes, and non-sensitive attributes. Explicit identifiers and sensitive attributes should be retained from the publication of the dataset. However, as mentioned above, impostors can perform privacy-intrusion attacks when quasi-identifiers can be linked to publicly available data.

- *Encryption* encompasses several cryptography algorithms that cypher the raw data and shares them online. Those interested can obtain the raw data once the proprietary data share the decryption key via a secure channel. Although this privacy-preservation technique is the most supported one, it lacks the flexibility of the previous two. In other words, once the data is published, the proprietary institution must complete download requests one by one, which is a tedious job. This kind of privacy-preservation technique is useful when data needs to be shared only with a few stakeholders.

We can use them to engender a privacy-conform dataset. For a detailed review of the previous strategies, we point the reader to [120]. By adopting privacy-based methods of PPDM, the produced dataset does not contain sensitive information that violates privacy regulations such as the European GDPR. Although the US and other non-European countries do not possess well-established privacy laws, it is advisable to cleave to the ethics of protecting the user profile data.

By following the different privacy-preservation techniques for dataset processing presented in [116], we present a simple solution to the e-degrees in Unitelma and fully anonymise them. Because we want to publish the dataset to academia, we do not rely on encryption algorithms that provide interested researchers with a decryption key. Nevertheless, we satisfy the other two classes (i.e. randomisation and anonymisation) to cope with sensitive student data. The information that we publicise consists of aggregated interaction trajectories of students. In other words, for each e-tivity type in a particular day d, we calculate the number of times that that e-tivity has happened in d. In this way, malicious users can only recuperate the

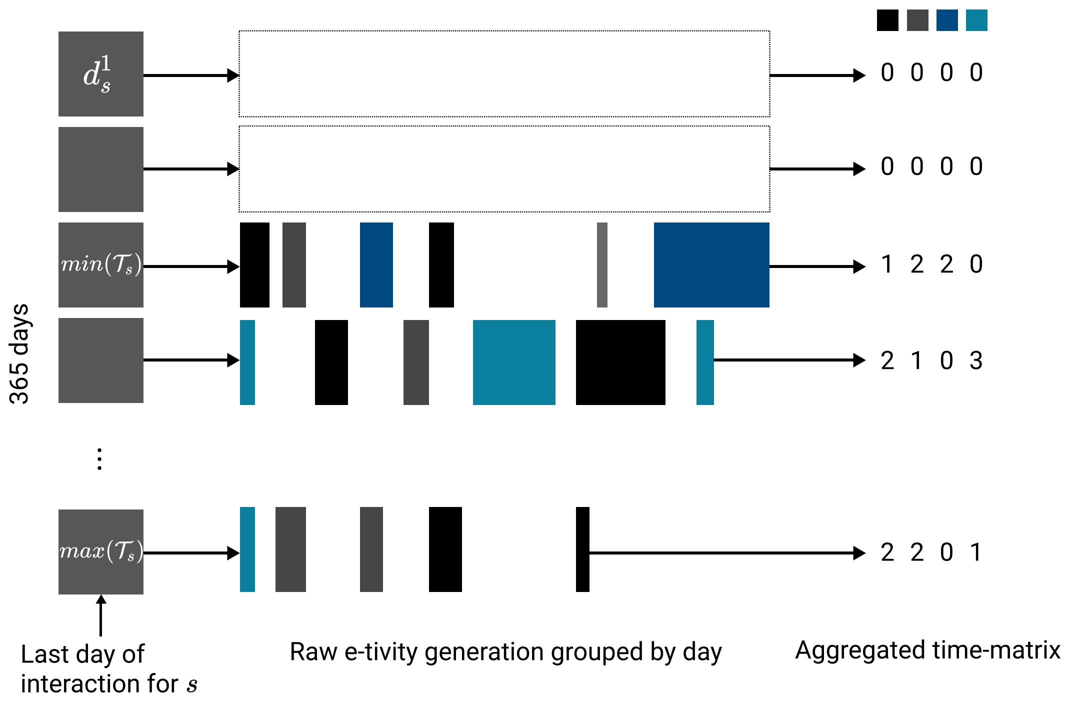


Figure 4.6. An overview of the schema adopted from the publication of the Unitelma dataset under a privacy-compliant point-of-view. Each student in the Unitelma dataset contains 365 days of interaction backwards of their last day of interaction. The colours of the rectangles illustrate the different types of e-tivities. Meanwhile, their width represents the duration of each e-tivity. In this example, we exploit a simple counting per e-tivity type aggregation function to randomise the original distribution of e-tivities.

aggregated distribution of e-tivities but not the exact generation moment. Moreover, the resource involved in the interaction and the course at hand is removed from the time-series information. The student identity numbers have been anonymised by following a one-to-one mapping with the set of natural numbers. The original identity is then discarded from the modified dataset. The same reasoning is applied to de-identify the e-degrees in the dataset as well as the e-tivity type. Therefore, according to Malik et al. [90] we only deal with explicit identifiers since we do not distribute information about the interaction of students in fora, nor do we share student demographic data. Referring to the lower part of Figure 4.5 and to the matrix notation in Section 3.4, for each e-degree $\mathcal{C}_i$ s.t. $i \in [1, 13]$ in the Unitelma dataset we only disclose the time-matrix $\mathcal{M}_s^{365,|\mathcal{E}|}$ for each student $s \in \bigcup_{c \in \mathcal{C}_i} \mathcal{S}_c$.

Figure 4.6 summarises the schema that we followed to publish the Unitelma dataset. Recall that, to align student trajectories locally, we obtain the last interaction day for each student s and travel backwards up to 365 days to establish d_s^1. We group the e-tivities coming from the raw logs by days and order them in the time axis. Finally, we obtain summarised data by exploiting a particular aggregation function - a counting function - per e-tivity type (i.e. the different colours for the rectangles) and report the outputs per day. Notice that the aggregation function can be complex to account for more information. However, we chose not to disclose more information because of the stringent privacy policies at the University of Rome

Unitelma Sapienza institution. Hence, as mentioned above the reported data for each student s has a dimension of $365 \times |\mathcal{E}| = 365 \times 97$. As a consequence, each e-degree $\mathcal{C}_i$ s.t. $i \in [1, 13]$ in Unitelma is a tensor of dimensions $|\bigcup_{c \in \mathcal{C}_i} \mathcal{S}_c| \times 365 \times 97$.

Chapter 5

A Novel Hidden Space Deep Sequential Method for Student Behaviour Analysis

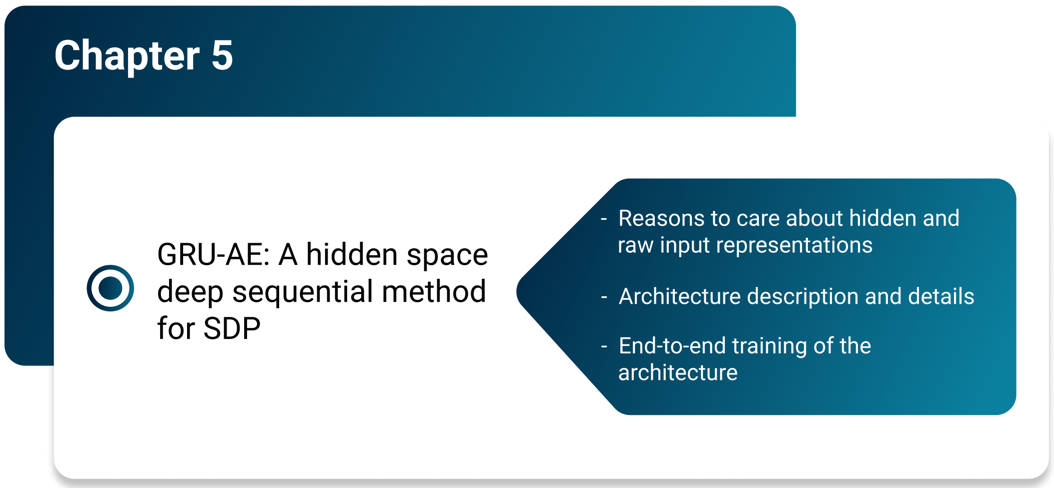

Chapter 5
GRU-AE: A hidden space deep sequential method for SDP
- Reasons to care about hidden and raw input representations
- Architecture description and details
- End-to-end training of the architecture

Figure 5.1. Simple solutions fail to extrapolate long-term dependencies between e-tivities because they have to cope with critical issues such as feature-space sparsity and inter-activity gaps. This Chapter provides details on the proposed architecture to distinguish dropouts from persisting students in highly imbalanced datasets. Moreover, we identify the main benefits of the combination of hidden and raw input features to the SDP problem. Finally, we provide details on how we train the model end-to-end by choosing a composite loss function to optimise.

According to Prenkaj et al. [108], one of the open challenges that the SDP research field has is the lack of deep sequential methods that prioritise the identification of at-risk students as early as possible in the course/e-degree timeline. One of the main reasons academia is hesitant to contribute with novel deep learning solutions is the poor interpretability power of so-called black-box models or lack thereof. Although knowing the reasons behind a dropout decision for a particular student is of paramount importance, we noticed that the models presented in the literature do not exploit key predictive factors to improve their performances. In other words, the literature does not exploit the latent and intrinsic long-term relationship between e-tivities that constitute student behavioural patterns. This Chapter addresses the phenomenon of combining hidden representations with the raw input features to obtain an overall better-performing model. Additionally, we provide the reader with the details about the architecture of the proposed model - hereafter GRU-AE. Finally, to have a functioning end-to-end model, we rely on a composite loss function that we optimise for the SDP problem.

5.1 Gated Recurrent Unit with Autoencoding (GRU-AE)

We propose a novel sequential strategy based on the latent and raw input feature space to cope with the last research gap. We name our strategy GRU-AE, as introduced in [106], since it combines autoencoders with stacked layers of GRUs. Before delving into the description of the architecture, we provide some of the major reasons to consider latent information as well as the raw feature space as crucial parts of the proposed prediction strategy (see Section 5.1.1). Finally, we give a detailed representation of the architecture of GRU-AE in Section 5.1.2.

5.1.1 Reasons to Care about Hidden and Raw Representations

Recurrent neural networks (RNNs) and their specialisations (i.e. LSTMs [57] and its GRU [24] variation) are the most prominent approaches for learning sequential patterns. In RNNs, at each time step t, the recurrent cell obtains hidden information on the history of a trajectory from the hidden state of the previous step $t-1$. Despite their popularity, naive LSTM/GRU approaches may fail to capture temporal dependencies under the following constraints:

- *The feature sparsity* - Since most students do not perform all the possible e-tivities, but rather they concentrate on some of them[1], features tend to be very sparse. This phenomenon induces models to overfit.

- *The gap between e-tivities* - In addition to feature sparsity, often e-tivities are interleaved by temporal gaps of variable lengths. When gaps between e-tivities increase, sequential models may forget the latent information extracted in the blocks that are further away. According to a pure statistical point-of-view, the authors in [137] prove that the recurrent process of propagating information from one hidden state to the other of RNNs and its variations is geometrically ergodic and has a short memory. Additionally, LSTMs suffer from catastrophic forgetting when proceeding with transfer learning to fine-tune the model weights [6].

We know that the gap between e-tivities induces recurrent networks to forget the information extracted that are distant in time. Nevertheless, the temporal gaps are useful to discriminate between highly- and sporadically-engaging students. The

[1]This is motivated both by the students' commitments and by the course schedule.

number of gaps and their temporal extensiveness is clear indications of unmotivated and at-risk students, although they do not necessarily indicate a certain dropout student. Moreover, the temporal gaps in e-degrees should provide the predictive model with a higher information gain w.r.t. MOOCs. Online degrees should have fewer gaps between e-tivities since several courses are held simultaneously in a specific semester.

Additionally, one can specify and choose to analyse the gap between the same e-tivity types. In this way, we can measure the dropout probability and behavioural pattern shifts per e-tivity type. Although interpretability is out of scope for this work, researchers can rely on gaps to assess whether the distribution of e-tivities of an at-risk student diverges from the average temporal distribution of e-tivities for persisting students. Therefore, gaps in an e-degree are useful to distinguish between failing and persisting students in a particular time frame.

On the contrary, the vast sparsity in the feature dimension induces the model not to generalise and have a lower prediction power. According to the literature, students interact with only a fraction of resources, thus neglecting the possible interaction methods in an OLE. This phenomenon of sparsity has also been shown in Table 4.1. Sparse time-matrices, especially when the feature dimension is vast, might degrade the performances of the predictive model. Hence, projection/densification strategies can embed the features mapped into a denser vector space by combining the significance of temporal gaps between e-tivities in the input student trajectories with the condensed representation of same-type e-tivities.

5.1.2 Architecture Description and Details

Figure 5.2 shows the four components of the GRU-AE architecture: the *Autoencoder*, the *Stacked Embedded GRU*, the *Stacked raw GRU*, and the *Fully Connected layer*. The input is represented by the time-matrix $\mathcal{M}_s^{\xi,|\mathcal{E}|}$, where rows are the vectors of e-tivities performed by a student s at day $d_j, j = 1 \dots \xi$. Note that the length ξ of the observed trajectory varies for each student. Additionally, notice that, for a given student, a prediction can be made about their future dropout status at different times in their career (e.g., after 30 days, three months, six months), which is a meaningful strategy, especially for long trajectories as in e-degrees.

Autoencoder The upper component of GRU-AE is an autoencoder [44, 79, 118] - hereafter AE - comprised of two modules: an encoder and a decoder. The task of the autoencoder, in our scenario, is to learn a compact and denoised representation of the features to mitigate the sparsity phenomenon. As illustrated in Figure 5.2, the AE component takes a vectorial representation $\mathcal{M}_s^{\xi\times|\mathcal{E}|}$ of the time-matrix $\mathcal{M}_s^{\xi,|\mathcal{E}|}$ by exploiting a row-major order flattening procedure. It then projects the original feature into a denser vector space $\vec{h}$. In detail, the encoder learns a hidden representation $\vec{h}$ of the input $M_s^{\xi\times|\mathcal{E}|}$ and the decoder extracts the information from $\vec{h}$ to produce a reconstruction of the input $\widetilde{M_s^{\xi\times|\mathcal{E}|}}$. The lower the reconstruction error

$$\left\| M_s^{\xi\times|\mathcal{E}|} - \widetilde{M_s^{\xi\times|\mathcal{E}|}} \right\|_b^a = \sqrt[a]{\sum_{i\in[1,\xi\times|\mathcal{E}|]} M[i]^b - \widetilde{M[i]^b}}$$

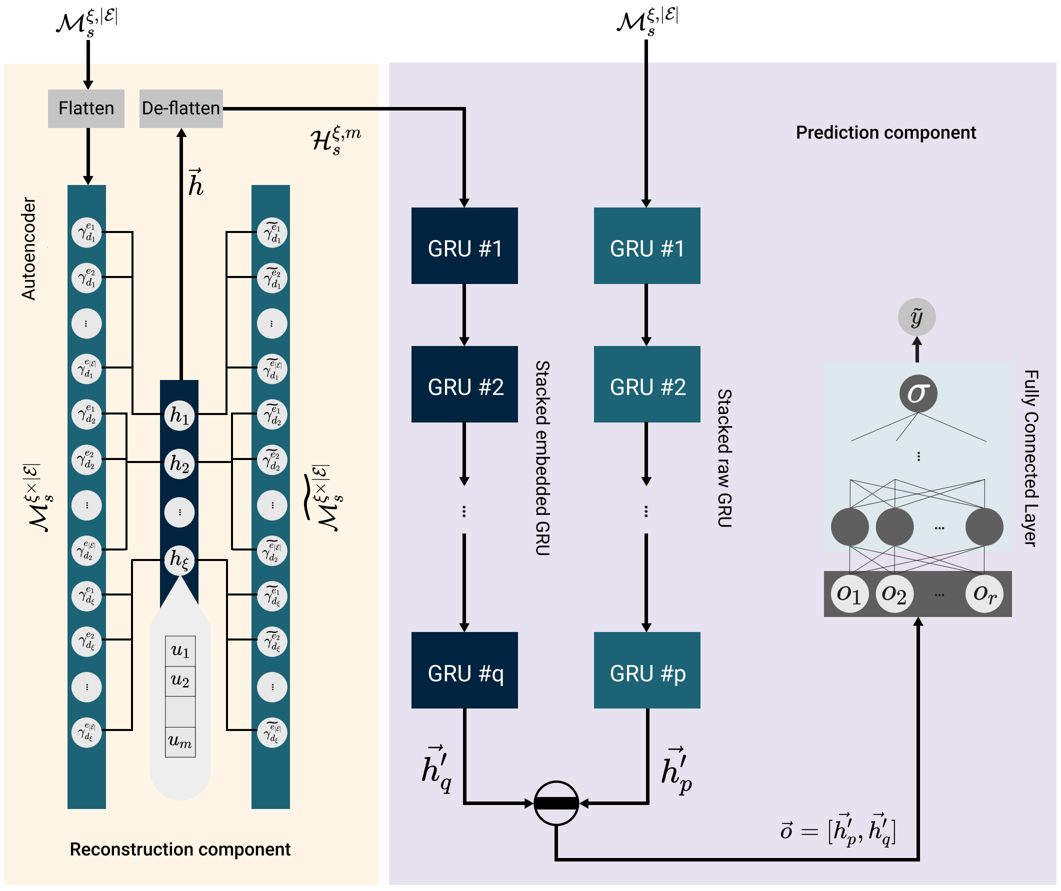


Figure 5.2. The architecture and components of GRU-AE. Rather than using a recurrent autoencoder, we opt to embed the temporal information in the connections of the network. Notice how the aggregated data, for each day d_j $\forall i \in [1, \xi]$, is connected only to its corresponding neuron h_j which, in turn, is a vector of m cells. The reconstructed matrix is then de-flattened into an embedded matrix $\mathcal{H}_s^{\xi,m}$ and fed to the stacked embedded GRU. The raw time-matrix $\mathcal{M}_s^{\xi,|\mathcal{E}|}$ is fed to the stacked raw GRU. The output of both GRUs, respectively, $\vec{h}'_p$ and $\vec{h}'_q$ is concatenated into a single vector $\vec{o}$ of r cells. Finally, o is the input layer to a fully connected layer which predicts the dropout probability $\widetilde{y}$ according to a sigmoid function σ.

the better the latent representation of the input. Besides using a reconstruction error which measures the difference between the original input and the reconstructed output - e.g. mean squared and root mean squared errors, Kullbach-Leibner divergence - one can also use similarity metrics (i.e. cosine similarity, normalised mutual information) to measure how much the reconstructed trajectories are related to the original ones.

Notice that recurrent AEs can be used when treating sequential data, as in our case of student trajectories. Nevertheless, we modify a fully connected AE, where the input neurons $\gamma_{d_j}^{e_i}$ $\forall i \in [1, |\mathcal{E}|]$ are only connected to h_j $\forall j \in [1, \xi]$. We use the same logic to link the latent layer to the output one in a specular way. Notice that each h_j is represented as a single neuron for illustration purposes, but it can have m dimensions. Hence, the vector $\vec{h}$ has a length of $\xi \times m$, where m is the embedding dimension within each temporal interval d_j, as represented by the vector $u_1, ..., u_m$ in the upper-right side of Figure 5.2. The idea behind the proposed architecture is to build an autoencoder made of smaller autoencoders that operate on feature counts *in any time slot* d_j and learn which e-tivities occur together in the original feature space. In this way, we mitigate the problem of the sparsity of e-tivities as well as the gaps between e-tivity generation in different days, since null e-tivities are transformed in real (negative or positive) numbers and compressed into a latent dimension.

Stacked embedded GRU The usage of stacked neural networks was reported to be beneficial in performance gain [62, 123] and in obtaining a higher model complexity capable of extracting latent information. The rationale of using a recurrent network on the hidden representation of the original time-matrices is to obtain a dense representation in both the feature space (thanks to the AE) and the daily gap between e-tivities performed by the students.

The purpose of the stacked embedded GRU component is to further compress the hidden representation of $\mathcal{M}_s^{\xi,|\mathcal{E}|}$, built by the AE according to the encoding function $E : \mathbb{N}^{\xi \times |\mathcal{E}|} \rightarrow \mathbb{R}^{\xi \times m}$ where $\xi \times m$ is the dimension of the hidden vector $\vec{h}$. However, because $\vec{h}$ does not contain the temporal dimension as the original input $\mathcal{M}_s^{\xi,|\mathcal{E}|}$, we need to reconstruct the temporal dimension in order for the GRU to work. As shown in Figure 5.2, we perform the inverse operation of the previous flattening function on $\vec{h}$. This process consists of reversing the row-major order operation performed by the flattening procedure on $\mathcal{M}_s^{\xi,|\mathcal{E}|}$. By employing a de-flattening operation, we obtain a hidden matrix $\mathcal{H}_s^{\xi,m}$ that restores the original time dimension ξ but has a denser feature space, as described above. This new matrix is the input of the first component of the stacked embedded GRU network. At each time step $d_j \in [1, \xi]$, this GRU is fed with the j-th row (i.e. j-th day) of $\mathcal{H}_s^{\xi \times m}$. At the end of the sequence, the first block produces an output of sequences represented as a matrix. The output matrix of the previous GRU block becomes the input of the subsequent block. Finally, instead of computing the output of the last GRU block, we use only its latent state h'_q.

For the sake of completeness, we also provide a brief description of a GRU unit. A GRU unit resembles an LSTM unit with a forget gate but has fewer parameters because of the lack of the output gate. In this work, we use a fully gated unit as represented in Figure 5.3. We refer the reader to [25] for the equation details of the

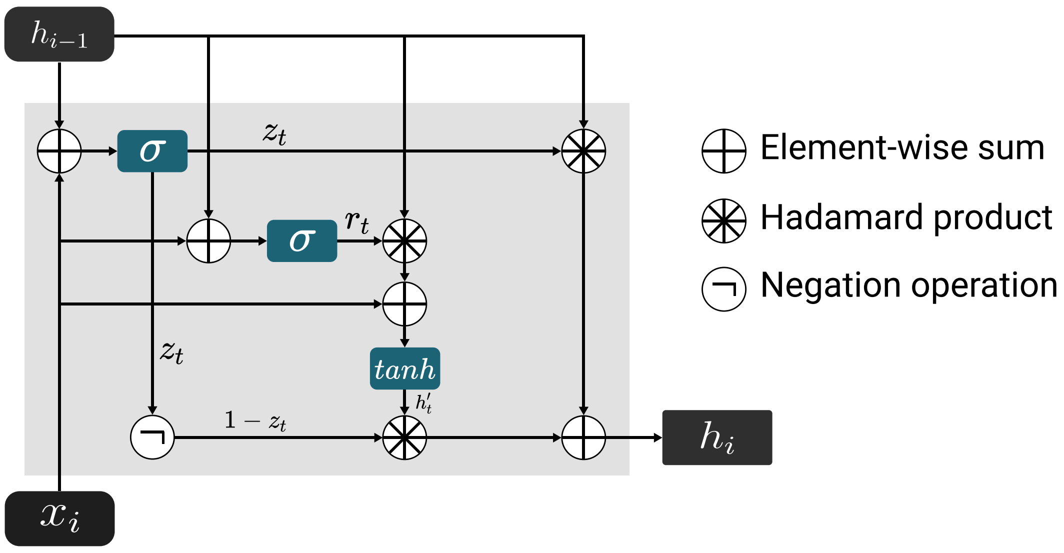


Figure 5.3. Gated Recurrent Unit block.

update and reset gates and that of the current memory content.

Stacked raw GRU Similarly to the component described above, we use a parallel sequence of stacked raw GRUs to learn time-related information from the original student trajectory $\mathcal{M}_s^{\xi,|\mathcal{E}|}$. Although the original time matrix contains much sparsity and daily gaps (see Section 4), it contains time-related information that captures the behaviour of the student throughout the days of the course. Like for the stacked embedded GRU, the output of this stack is the last hidden state h'_p. Notice that the length p of the sequence of GRU networks adopted in this component can be different from that of the stacked embedded GRU q. The activation function of both the recurrent components is a Rectified Linear unit (ReLU).

Fully connected neural network Finally, we concatenate $\vec{h'_q}$ and $\vec{h'_p}$ (i.e. $\vec{o} = [\vec{h'_p}; \vec{h'_q}]$ where $|\vec{o}| = r = |\vec{h'_q}| + |\vec{h'_p}|$) thus forming the input vector to the fully connected neural network layer. Notice that this layer can be a deep neural network formed of multiple layers of neurons. Its objective is to leverage the joint information of the two stacked GRUs and to use the time-related information and the time-unrelated embedded representation. The contribution of $\vec{h'_q}$ and $\vec{h'_p}$ is boosted by updating the weights of the final component via backpropagation. Finally, we use a sigmoid activation function for the first layer and the output layer. For all the other layers, we use the ReLU activation function. The reason for this choice is twofold. Since the sigmoid function is prone to the vanishing gradient problem, we use the ReLU unit to mitigate this phenomenon. Furthermore, the ReLU function suffers from the dying ReLU problem as analysed in [87]; this causes neurons to become stale when dealing with large gradients. To avoid this phenomenon, we use the sigmoid function in the first and output layers. The output of the architecture is the estimated probability that s is a dropout student. In other words, the closer $\widetilde{y}$ is to 1, the more s is to be considered a future dropout; otherwise, s is a persisting student.

5.1.3 End-to-End Architecture Training

Notice - see the shaded areas in Figure 5.2 - that the architecture of the model contains two separate logic components: i.e. the reconstruction component that generates the latent embedding of the student trajectories and the predictive component encompassing both stacked GRUs and the fully connected layer. The training of an autoencoder is executed differently from that of a recurrent neural network with a final dense predictive layer. Nevertheless, because we need to train the entire architecture end-to-end to be useful, we combine two separate loss functions into one and minimise it.

The loss function of the reconstruction component The first loss function encompasses the minimisation of the reconstruction error. Notice that training an autoencoder is a special case of solving a regression problem. There are several loss functions related to regression problems, among whom the most important is the mean squared error (MSE) and the mean absolute error (MAE). Both of them have advantages and drawbacks over the other, considering the characteristics of the datasets. Here, we briefly introduce both of them and describe their strengths and flaws.

Mean Square Error (MSE) is the most commonly used regression loss function. MSE is the sum of squared distances between the target variable and predicted values.

$$\mathcal{L}_{MSE} = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|}(y_i - \tilde{y}_i)^2$$

where $|\mathcal{S}|$ is the number of students (i.e. trajectories) enrolled in an online degree $\mathcal{C}$. Similarly, MAE is the sum of absolute differences between the target and predicted variables:

$$\mathcal{L}_{MAE} = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|}|y_i - \tilde{y}_i|$$

Therefore, it measures the average magnitude of errors in a set of predictions without considering their directions. The domain of MAE/MSE ranges from 0 (i.e. there is no difference between the target and the predicted values) to ∞.

For instance, let us analyse both errors in two different scenarios, where the first contains predictions close to the true values and the second has some variations due to the presence of outliers (see Table 5.1). As expected, the outlier instances produce higher errors since the predictive models tend to generalise instead of memorising the instances one by one, leading the model to learn a representation that suffers from data instances skewed from the normal data distribution. Since MSE squares the error $y_i - \tilde{y}_i$, its value increases a lot if $y_i - \tilde{y}_i > 1$. If we have an outlier in our data, the value of $y_i - \tilde{y}_i$ will be high and $(y_i - \tilde{y}_i)^2 >> |y_i - \tilde{y}_i|$. Outliers make the model with MSE loss give more weight to them w.r.t. a model with MAE loss. In

Table 5.1. MSE and MAE for two toy examples with the outlier instance underlined. Every row is a new instance in the dataset. Each column represents the error between the predicted and the true values.

	Dataset without outliers			Dataset with outliers		
	$y_i - \tilde{y}_i$	$\lvert y_i - \tilde{y}_i \rvert$	$(y_i - \tilde{y}_i)^2$	$y_i - \tilde{y}_i$	$\lvert y_i - \tilde{y}_i \rvert$	$(y_i - \tilde{y}_i)^2$
	0	0	0	0	0	0
	1	1	1	1	1	1
	-2	2	4	1	1	1
	-0.5	0.5	0.25	-2	2	4
	1.5	1.5	2.25	$\underline{15}$	$\underline{15}$	$\underline{225}$
$\mathcal{L}_{MSE}$		1.4884			46.1041	
$\mathcal{L}_{MAE}$		1			3.8	

the second case in the Table, the model with the MSE as the loss will be adjusted to minimise that single outlier case at the expense of other common examples, which reduces its overall performance. MAE loss is useful if the training data is corrupted with outliers. In other words, MAE is better than MSE if we receive unrealistically large negative/positive values in our training set [2]. Nevertheless, the MAE loss has the same gradient throughout the entire dataset, which entails a large number even for small loss values. The gradient for MSE is high for larger loss values and decreases as it approaches 0, making it more stable and precise during training.

Because our scenario encompasses dropout students (outliers) that we want to identify, the MSE loss is more suitable than MAE since, as mentioned above, it is more sensitive to outlier instances. However, the high sensitivity of MSE towards outlier data makes it impracticable in our case. Therefore, we rely on the Huber loss function, which is less sensitive to outliers than MSE. Moreover, differently from MAE, the Huber loss is differentiable at 0 being a continuous function. The Huber loss function incorporates the positive aspects of both MAE and MSE:

$$\mathcal{L}_{\Omega,Huber}(y_i, \tilde{y}_i) = \begin{cases} \frac{(y_i - \tilde{y}_i)^2}{2} & \text{if } |y_i - \tilde{y}_i| \leq \Omega \\ \Omega|y_i - \tilde{y}_i| - \frac{\Omega^2}{2} & \text{otherwise} \end{cases} \tag{5.1}$$

where Ω represents the amount for which the residuals (i.e. errors) need to be considered as outliers or as normal instances. Residuals larger than Ω are minimised with MAE (see the *otherwise* condition), while those smaller than Ω are minimised with MSE. Notice that we specify the input to the loss function as it operates on each predictive value separately, and it does not have an accumulated form as in the cases of $\mathcal{L}_{MSE}$ and $\mathcal{L}_{MAE}$. The only drawback the adoption of the Huber loss brings to our overall training mechanism is the optimisation of the Ω hyperparameter.

The loss function of the predictive component Recall that the literature considers the dropout prediction problem a binary classification problem. Therefore, dropout students are labelled with a 0, and the persisting ones with a 1. Moreover,

[2]Notice that we cannot make any observations regarding the content of the test set since it is considered unseen data. Moreover, the overall performances of a particular model cannot be disclosed only by referring to the data distribution in the test set.

recall that the activation function of the output layer in the fully connected component is a sigmoid function, which allows us to produce dropout probabilities. Generally, the literature exploits binary cross-entropy loss, also known as the log loss, to learn a model in a binary classification scenario.

Binary cross-entropy compares each of the predicted probabilities to the actual class. Then, it calculates the score that penalises the probabilities based on the distance from the expected value. We describe the binary cross-entropy for one single student. The overall loss function is the summation of all the cross-entropies divided by the number of the population $|\mathcal{S}|$:

$$\mathcal{L}_{BCE}(\tilde{y}_i) = \tilde{y}_i \times log(p(\tilde{y}_i)) + (1 - \tilde{y}_i) \times log(p(1 - \tilde{y}_i))$$

where $\tilde{y}_i$ is the label corresponding to the i-th student in $\mathcal{S}$ and $p(\tilde{y}_i)$ is the predicted probability of that student to be a dropout. Besides using this particular loss function in our scenario, the binary cross-entropy loss is beneficial to undo any exponential behaviour - due to the gradient descent - given in output by the activation units in the intermediate layers of the architecture. The logarithm avoids that the gradient saturates for extreme values since large gradients are useful to make progress in learning in each iteration [44]. Finally, cross-entropy is used because it is equivalent to fitting the model using maximum likelihood estimation, which can be interpreted as minimising the Kullback-Leibner (KL) divergence between the distribution of the training data and that produced by the model.

Combined loss function To train the architecture end-to-end, we need to minimise both the loss functions introduced above. Notice that the sole hyperparameter induced to the loss function is the Ω parameter of the Huber loss, hence the subscript in the minimisation. Moreover, because the binary cross-entropy and the Huber losses are continuous and differentiable, their sum is also differentiable, equal to the sum of the derivatives of the losses.

$$\operatorname{argmin}_{\Omega}\left\{ -\frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|}\mathcal{L}_{BCE}(\tilde{y}_i) + \sum_{i=1}^{|\mathcal{S}|}\mathcal{L}_{\Omega,Huber}(y_i,\tilde{y}_i)\right\} =$$
$$\operatorname{argmin}_{\Omega}\left\{ \sum_{i=1}^{|\mathcal{S}|}\left(|\mathcal{S}|\mathcal{L}_{\Omega,Huber}(y_i,\tilde{y}_i) - \frac{1}{|\mathcal{S}|}\mathcal{L}_{BCE}(\tilde{y}_i)\right)\right\}$$

To prove that the proposed combined loss function is differentiable, let $\mathcal{L}_{BCE}$ be function f and $\mathcal{L}_{\Omega,Huber}$ be function g. Now, if f is differentiable at point c, then by definition $f'(c) = \lim_{h\to 0}\frac{f(c+h)-f(c)}{h}$. Similarly, if g is differentiable at c, then $g'(c) = \lim_{h\to 0}\frac{g(c+h)-g(c)}{h}$. Now, because $(f+g)(x) = f(x) + g(x)$ we need to prove that

$$\lim_{h\to 0}\frac{[f(c+h)+g(c+h)] - [f(h)+g(h)]}{h} = f'(c) + g'(c)$$

By applying simple mathematical manipulations on the limit above we obtain

$$
\begin{aligned}
&\lim_{h\to 0}\frac{[f(c+h)+g(c+h)]-[f(h)+g(h)]}{h}\\
&=\lim_{h\to 0}\left[\frac{f(c+h)-f(h)}{h}+\frac{g(c+h)-g(h)}{h}\right]\\
&=\lim_{h\to 0}\frac{f(c+h)-f(h)}{h}+\lim_{h\to 0}\frac{g(c+h)-g(h)}{h}\\
&=f'(c)+g'(c)
\end{aligned}
$$

An interesting property of the proposed composite loss function is minimising both its components that go hand-in-hand with the modules of the overall architecture. In other words, to obtain a small loss, we need to minimise the overall sum of the losses. Notice that we do not rely on the product between the Huber and the binary cross-entropy losses because it would be sufficient to minimise only one of them to have an overall small loss. Hence, we obtain promising performances, but we also show that the reconstruction error of the student trajectories has a decreasing trend as the interaction history becomes more extensive (see Section 6.2 for further details). For completeness purposes, we provide the reader with an ablation study of the performances of GRU-AE according to the Ω hyperparameter of the Huber loss function (see Section 6.2.1).

Chapter 6

Experiments

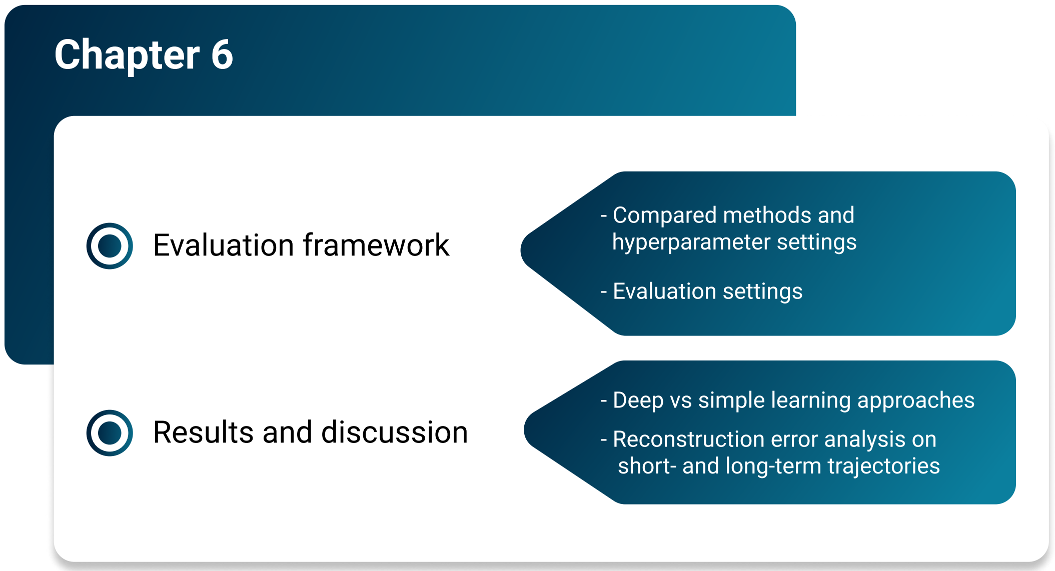

Chapter 6
Evaluation framework
- Compared methods and hyperparameter settings
- Evaluation settings
Results and discussion
- Deep vs simple learning approaches
- Reconstruction error analysis on short- and long-term trajectories

Figure 6.1. While scrutinising the most suitable evaluation metrics under a high-class imbalance issue and the presence of a large portion of inter-e-tivity gaps, we compare state-of-the-art methods - both simple machine learning and deep learning approaches - against GRU-AE. We illustrate how our proposed method has better performance than the other strategies while demonstrating how long-term trajectories (i.e. e-degrees) benefit our methodology more. Conclusively, we show how a lower reconstruction error benefits higher predictive power for our model.

Since the SDP literature does not have a common benchmarking and evaluation framework, we follow the guidelines provided in [108] while taking into consideration issues such as high-class imbalance and the daily gap between e-tivities as illustrated in Table 4.1 and Section 5.1.1. In this Chapter, we determine the most suitable metrics for evaluation in our scenario (see Section 6.1.2) and compare GRU-AE with ten state-of-the-art simple machine learning and deep learning strategies (see Section 6.2.2). In conclusion, we discuss the relationship of the trajectories' reconstruction error with the overall performances of the architecture.

6.1 Evaluation Framework

Recall from Section 2.2.3 that the literature has little to no contribution for a standard evaluation benchmark regarding the optimal metric to measure performances under a high-class imbalance problem and the training characteristics according to a stabilised time window. Here, we give a detailed description of the compared state-of-the-art methods. Notice that most of the strategies enlisted below are easily implemented from modern programmatic libraries. Moreover, we provide the reader with the choices of the evaluation metric and the selection of the time window we use to determine the amount of history of the student trajectories for training.

6.1.1 Compared Methods and Hyperparameter Settings

To comparatively assess the performances of GRU-AE, we select five off-the-shelf machine learning models, two ensemble and five deep learning techniques[1] grouped as follows:

Off-the-shelf machine learning approaches. Because the majority of the literature concentrates on simple machine learning models and simple modelisation strategies - see Table 4 in [108] - we select four of the most widespread prediction approaches. Then, to assess the performances of unsupervised methods in a binary classification problem, we provide the reader with a simple unsupervised strategy suitable for the SDP problem. The last one in the list depicts the simplest strategy possible in this scenario that illustrates the lower bound in terms of performances (i.e. for a method to be considered "fit" to solve the SDP problem, then it needs to surpass the lower bound performances according to the baseline mentioned above strategy).

- Logistic Regression (LR) - We use logistic regression with lasso regularisation as in [42, 113]. We also specify the maximum number of iterations to help the model converge.
- Gaussian Naive Bayes (GNB) - We use Naive Bayes [41, 76, 91] but we do not rely on prior probabilities for the target labels.
- Decision Tree (DT) - We exploit the optimised version of the CART algorithm [83] with the Gini impurity to measure the quality of the splits per level [78, 95]. Furthermore, we noticed that a cost matrix, used in [30], does not support performance boosting.
- Support Vector Machines (SVM) - We use the radial basis function as the kernel for this method. Only the authors in [3] adopt the same structure as reported here.
- K-Nearest Neighbours (KNN) - Inspired by the nature of unsupervised learning, we use a 2-nearest neighbours algorithm to assess whether the cluster formations

[1]The implementation of all methods and the benchmark datasets can be found at `http://iim.di.uniroma1.it/projects/hsdsrpst/`. The Unitelma dataset can be found at `https://doi.org/10.6084/m9.figshare.14554137`.

speculate the binary target labels. Little to no contribution has been made in the literature using unsupervised learning methods [74, 135].

- Cox Proportional Hazard Model (CoxProp) [70] - The Cox (proportional hazard) model is one of the most popular model combining the covariates and the survival function. It models the hazard function

$$h^s(t|X = \mathcal{M}_s^{\xi,|\mathcal{E}|}) = h_0^s(t)e^{\mathcal{M}_s^{\xi,|\mathcal{E}|}[:,j]\beta}$$

  where β is the vector of coefficients of each covariate, $h_0^s(t)$ is the baseline hazard function for student s, and $\mathcal{M}_s^{\xi,|\mathcal{E}|}[:,j]$ represents a random covariate (e-tivity) at any row (day). From the previous equation, we can derive the cumulative conditional hazard function

$$H^s(t|\mathcal{M}_s^{\xi,|\mathcal{E}|}) = e^{\mathcal{M}_s^{\xi,|\mathcal{E}|}[:,j]\beta}\int_0^j h_0^s(w)dw = e^{\mathcal{M}_s^{\xi,|\mathcal{E}|}[:,j]\beta}H_0^s(j)$$

  which helps us get the survival function for each student. Since a survival analysis model needs to have a duration at which the dropout status gets verified, we condense the matrix $\mathcal{M}_s^{\xi,|\mathcal{E}|}$ until that specific time according to the summation aggregation function (see how the selection of the time-window is performed in Section 6.1.2).

- Majority Class (MC) - Finally, we use a baseline that always predicts the majority class of the training data.

Ensembles. We use a Random Forest (RF) with bootstrapped samples and the Gini impurity to evaluate the splits [45, 46]. Next, as in [75], we construct an ad-hoc ensemble mechanism - hereafter *KENS* - with majority voting as the consensus function. The base components are WINNOW [85], 1-Nearest Neighbour and Naive Bayes without priors.

Deep feed-forward neural networks. We implement a three- and five-layered neural network as in [36], respectively DNN-3 and DNN-5. We use a shrinking factor α to calculate the number of neurons: i.e., $n_i = \lfloor \alpha \times n_{i-1} \rfloor$ where n_i is the number of neurons in layer i.

Inspired by the network architecture in [82], we implement a CNN, namely *Simple CNN*. It consists of two convolutions (with max-pooling) and three dense layers. The activation of each layer is the ReLU function. The last layer has a sigmoid function. The number of output channels is 6 and 16 for the first and second convolution layers, respectively. The convolutional layers have a kernel of size 5 and a padding of 2. The pooling layers have both the kernel size and stride equal to 2. The first dense layer receives the flattened volume produced by the last pooling layer and passes it through the other layers. The output neurons of the dense layers are 20, 10, and 1, respectively.

Deep sequential methods. We implement an LSTM [35] - namely *Simple LSTM* - and a combination of CNNs and RNNs - namely *ConRec* [132]. Simple LSTM has m LSTM cells where m is the portion of the sequence length that we consider. We specify a hidden vector size of 100. Lastly, the output of the LSTM network is flattened and goes into a dense layer with a sigmoid activation function. We implement *ConRec* by stacking two convolutional layers before passing the output to an RNN. The number of kernels in the convolution layers are 20 and 50, respectively, with kernel sizes of 2. We set the hidden vector of the recurrent neural network to 50. Finally, we feed the hidden vector to a dense layer with a sigmoid activation function.

Deep sequential with attention. First, we incorporate an attention mechanism to a 1d CNN as in *CFIN* [36] that infers the kernel size. Because the datasets presented in Section 4 do not contain user-specific data (e.g. grades, exams taken, homework submitted), we skip the augmentation and smoothing processes introduced in the paper. Instead, we use an attention context that is initialised according to a uniform distribution with $\mu = 0$ and $\sigma = 0.05$. Finally, we have a fully connected layer that outputs the probability of a dropout student relying on a sigmoid activation function.

GRU-AE. First, we need to set the hyperparameters of all the four architecture components described in Chapter 5. We performed a grid search on all hyperparameters to find the best combination. Instead, the number of observed days ξ for each student depends on the characteristics of the considered dataset. We decided to predict the dropout probability of each student at fixed intervals of time. We refer the reader to Section 6.1.2 for additional details on the setting of this hyperparameter. We use $\Omega = 10^{-2}$ as the Huber loss function threshold parameter (see Section 6.2.1 for further details on the selection of Ω). For the remaining hyperparameters, they have been set as follows:

- *AE* - To set the size of the embedding dimension m we use a shrinking factor γ on the dimension of e-tivities $|\mathcal{E}|$, thus $m = \lfloor \gamma \times |\mathcal{E}| \rfloor$. We used the range $[0.05, 0.5]$ with a step of 0.05 to search for the best performing γ. We obtained the best performances when $\gamma = 0.15$.

- *Stacked embedded GRU* - We set the stacking level q according to the values of the set $\{2, 4, 6, 8, 10\}$. A value of $q = 2$ produced the most promising results. Additionally, we set the hidden vector h'_i $\forall i \in [1, q]$ size equal to 100. To prevent overfitting during the learning, we used a dropout factor $\rho \in [0.05, 0.25]$ with a step of 0.05. We reached the best trade-off performance-overfitting when $\rho = 0.15$.

- *Stacked raw GRU* - Similarly to the previous hyperparameter setting, we obtained the best performances of the model when the stacking level of this component $p = 2$. We set the hidden vector $h''_i \in [1, p]$ size equal to 100. Finally, as in the stacked embedded GRU, the dropout factor for the connections in the GRU is 0.15.

- *Fully connected component* - We choose the number of layers according to a grid search on the set $[1, 10]$ where the best performances are achieved with a number of layers equal to 5.

6.1.2 Evaluation Settings

Evaluation metric Since, as shown in Chapter 4, all datasets exhibit a class imbalance, accuracy is not an informative metric since the model could achieve good performances by predicting only the majority class. Moreover, Saito et al. [115] show that the visual interpretation of Receiver Operating Characteristics (ROC) curves, when dealing with imbalanced datasets, is biased by a large number of true negatives. The same study shows that Precision-Recall (PR) curves are a better metric to evaluate the performances of binary classifiers in an unbalanced setting because they evaluate the ratio between the predicted positive and the true positive instances. A related popular evaluation metric is the area under the precision-recall curve (AUCPR) that we adopt in our study. However, since the majority class in our datasets might change from one course (or e-degree) to another, we weigh Precision (P) and Recall (R) by the support (i.e., the number of true instances). Then, we use these weighted metrics to build the PR curve and calculate the AUCPR. Finally, for completeness, we show the average F1, precision, and recall scores to express the goodness of classification for each strategy. Notice that deep strategies output real numbers in the range of [0,1] according to the sigmoid function. Hence, we use thresholds $\theta \in \{0.1, 0.2, ..., 0.9\}$ to binarise the labels and calculate the average R (F1) on all θ. In other words, all the scores over θ are set to 1, and those under θ are set to 0.

Selection of the time window The real-time detection of dropout students is of paramount importance. It is critical to minimise the time window during which students are observed to predict their risk of dropout early. In our experiments, we analysed the performances of each method with different time windows of different fixed interval lengths (i.e. ℓ_1,ℓ_2,...,ℓ_n). When training a model, we extract the first ℓ_i $\forall i \in [1, n]$ rows (days) of $\mathcal{M}_s^{\xi,|\mathcal{E}|}$ producing a matrix $\mathcal{M}_s^{\ell_i,|\mathcal{E}|}$ for each student s. Notice that $\ell_i << \xi$ $\forall i$. In case of simple machine learning strategies, we flatten the time-matrix $\mathcal{M}_s^{\ell_i,|\mathcal{E}|}$ into a vector $\mathcal{M}_s^{\ell_i \times |\mathcal{E}|}$. We expect that, in the scenario of e-degrees, with the increase of $i \in [1, n]$, the performances of GRU-AE will have a non-decreasing monotonic trend. In other words, the performances corresponding to ℓ_{i+1} will be greater than or equal to those of ℓ_i where $i \in [1, n)$. Hence, the longer the history of a particular student s is, the more certain the dropout status is. This phenomenon is justified because of the long-term dependencies captured from the built-in component in GRU-AE.

For the CoxProp[2] model to produce meaningful results, we need to modify the input data accordingly. In this scenario, for each student s and ℓ_i, we squash the rows of $\mathcal{M}_s^{\ell_i,|\mathcal{E}|}$ into a vector $\mathcal{M}_s^{|\mathcal{E}|} = \sum_{i=1}^{\ell_i} \mathcal{M}_s^{\ell_i,|\mathcal{E}|}[i]$. Moreover, we have to add an additional feature to the matrix that depicts the time in which the dropout event has

[2]Implementation guidelines can be found at `https://lifelines.readthedocs.io/en/latest/fitters/regression/CoxPHFitter.html`

occurred. Notice that we do not possess information about the exact time in which a specific student has abandoned their studies. Rather, we know that a student s has abandoned/persisted at the end of course c (i.e $f'(c)$). Hence at each ℓ_i, the added feature will be ℓ_i because the dropout status of s is immutable since we are taking into consideration the status at $f'(c)$. Differently from the other methods that have a global ground truth label, for CoxProp we need to repeat n times the dropout label and associate it to ℓ_i $\forall i \in [1, n]$. Finally, if the survivability probability exceeds the threshold 0.75, we indicate that s is a persister in ℓ_i; otherwise, s is a dropout student.

Training characteristics The loss function for deep strategies is binary cross-entropy (BCE). We use the ADAM optimiser with a learning rate of 10^{-3} $\beta_1 = 9 \times 10^{-1}$, $\beta_2 = 9.99 \times 10^{-1}$, and a weight decay of 5×10^{-5} for every deep model. We use 50 epochs for the three- and five-layered networks, GRU-AE, LSTM, and the simple CNN. In contrast, we use 10 epochs for ConRec, 15 epochs for CFIN and 100 for the WINNOW base component of KENS. We specify the batch size for each dataset to be 16. Finally, we divide all datasets into 70:10:20 splits for training, validation, and test, respectively.

6.2 Results and Discussions

This Section demonstrates that GRU-AE is statistically comparable to the compared methods in short-term MOOCs while outperforming them in long-term and complex e-degrees. Furthermore, we provide the reader with analyses that illustrate the effectiveness of our proposed method for both MOOCs and online degrees.

6.2.1 Loss Function Ablation Study

Prior to comparing state-of-the-art strategies with GRU-AE, we perform an ablation study of the combined loss function, specifically the Huber loss - i.e. reconstruction component - according to the hyperparameter Ω. Table 6.1 illustrates the performances in terms of average AUCPR and F1 on 30 runs for each dataset. Because the binary cross-entropy (BCE) loss function of the predictive component does not include any hyperparameters, we vary Ω only for the Huber loss and assess the differences in performances. We perform an ANOVA test on the 30 runs and verify that the mean of GRU-AE with the different loss function is not similar. Afterwards, we exploit a post-hoc Tukey HSD with a p-value of 0.01 to extract those couples of compared methods (signed with * or §) whose mean performances are statistically and significantly similar. The last row of the table depicts the best-performing GRU-AE variation that we use to compare our performances with the methodologies in the literature (see Section 6.2.2). For completeness purposes, we also include MSE and MAE as the loss functions for the reconstruction component of GRU-AE to empirically assess the interval of Ω values for which the Huber loss translates into MAE/MSE. Notice that the most contribution on the overall performances comes from the MAE component of the Huber loss (i.e. $|y_i - \widetilde{y}_i| > \Omega$) because the performances tend to decrease with the increase of the magnitude order of Ω.

Table 6.1. Ablation study of the performances of GRU-AE according to the different loss functions. Notice that the binary cross-entropy remains the same since it represents the loss of the prediction component. We alter the loss function for the reconstruction component to asses the impact of the hyperparameters on the overall architecture. A value is in bold if it is the best value on average (30 runs); it is labelled (* or §) when it is not significantly diverse from any other model with the same label according to a one-way ANOVA test with post-hoc Tukey HSD with p-value of 0.01.

	XuetangX		KDDCup15		Unitelma	
	AUCPR	F1	AUCPR	F1	AUCPR	F1
$\text{Huber}_{\Omega=10^{1}}$ + BCE	0.6555*	0.6199*	0.6814*	0.6498*	0.6700*	0.6399*
$\text{Huber}_{\Omega=10^{0}}$ + BCE	0.6589*	0.6101*	0.6807*	0.6478*	0.6750*	0.6378*
$\text{Huber}_{\Omega=10^{-1}}$ + BCE	0.7544	0.7290	0.8676	0.8021	0.7331	0.7009
$\text{Huber}_{\Omega=10^{-3}}$ + BCE	0.7112§	0.7027§	0.8312§	0.7845§	0.7282§	0.6820§
MSE + BCE	0.6672*	0.6182*	0.6890*	0.6525*	0.6789*	0.6443*
MAE + BCE	0.7189§	0.7091§	0.8357§	0.7789§	0.7200§	0.6898§
$\text{Huber}_{\Omega=10^{-2}}$ + BCE	**0.7900**	**0.7520**	**0.9346**	**0.8430**	**0.7715**	**0.7346**

Therefore, when $\Omega \to 10^1$, the Huber loss is represented as a simple MSE (see Equation 5.1), and MSE induces GRU-AE to have statistically significantly similar performances as the GRU-AE variant with the Huber loss with $10^0 \leq \Omega \leq 10^1$. Contrarily, when $\Omega \geq 10^{-3}$, the Huber function translates into MAE (notice the § annotation in the Table). The variant of the Huber loss with $\Omega = 10^{-1}$ reaches satisfactory performances, but the one with $\Omega = 10^{-2}$ reaches the maximum of average AUCPR and F1 scores.

6.2.2 Deep vs Simple Learning Approaches

In order to perform a thorough analysis of the performance of the state-of-the-art methods with our proposed framework, we consider the following aspects and compare the results to the performances of GRU-AE.

AUCPR performance of deep vs. simple machine learning techniques. Considering the irregular patterns that the temporal series in our datasets have, simple machine learning strategies underperform w.r.t. more complex and deep methods. Figures 6.2 and 6.3 illustrate the average AUCPR scores on 30 runs for all time windows ℓ_i on XuetangX and KDDCup15, respectively. Notice that, with the increase of ℓ_i, deep strategies perform significantly better, whereas predictions based on observing shorter time windows are comparable to simpler models. Notice that all methods perform better than the *MC* baseline, which indicates that they have successfully learned to distinguish between the binary classes Additionally, our method (red curve) in XuetangX is the best performing strategy among all the compared ones. Contrarily, Figure 6.3 shows that DNN-5 is the best performing method. Concerning Unitelma (Figure 6.4), we observe a variegated behaviour as the dimension of the temporal windows increases, with deep strategies (and especially GRU-AE) outperforming simple methods only as the length of the time

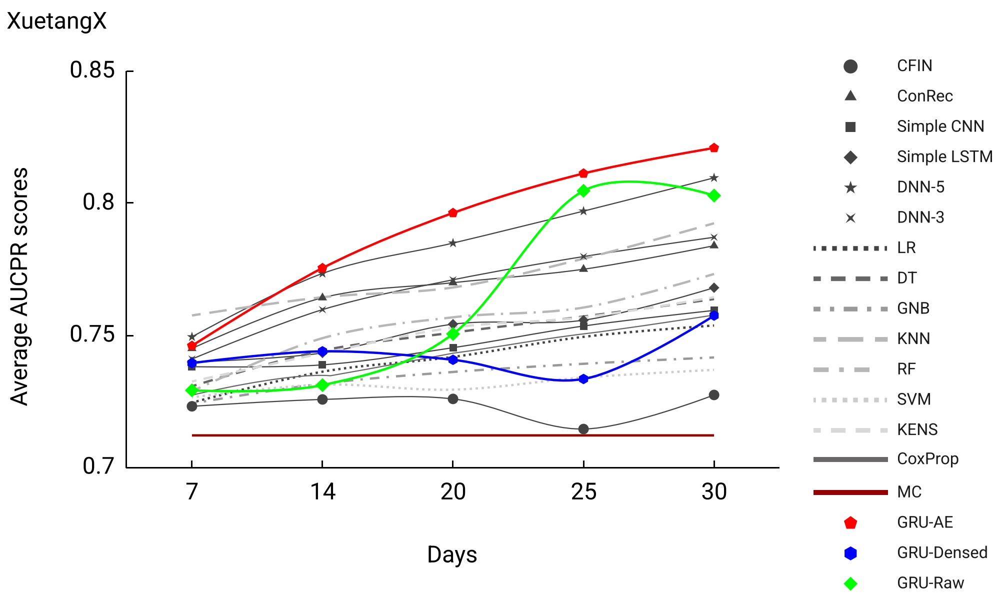


Figure 6.2. Average AUCPR scores on 30 runs for XuetangX. The curves without symbols represent off-the-shelf machine learning strategies. Meanwhile, the curves with symbols are deep learning approaches. Notice how the majority class (MC) approach is constant with the change of ℓ_i since the majority class does not change if the history per student trajectory is enlarged/shrunk.

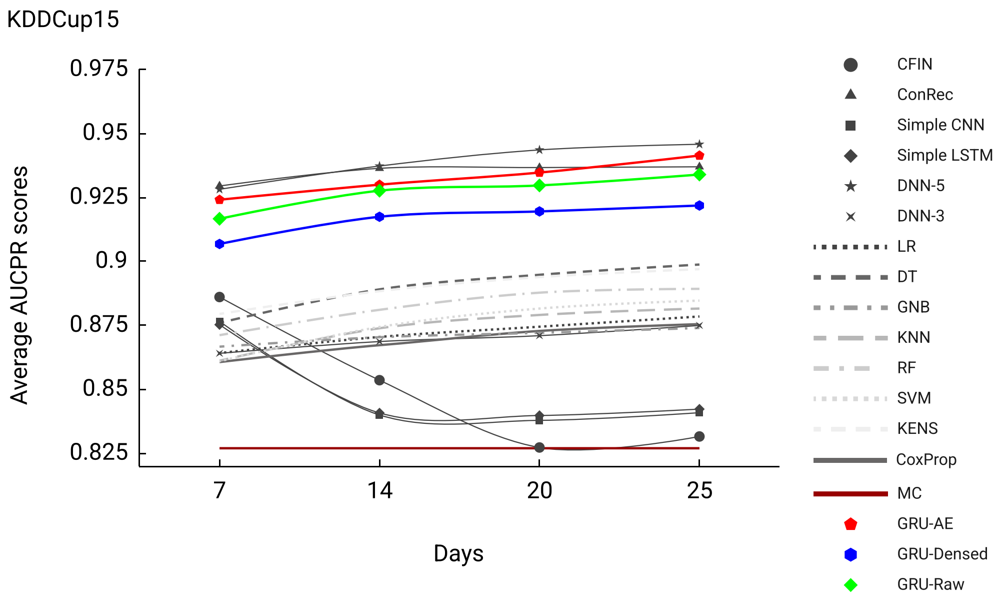


Figure 6.3. Average AUCPR scores on 30 runs for KDDCup15. The curves without symbols represent off-the-shelf machine learning strategies. Meanwhile, the curves with symbols are deep learning approaches. Notice how the majority class (MC) approach is constant with the change of ℓ_i since the majority class does not change if the history per student trajectory is enlarged/shrunk.

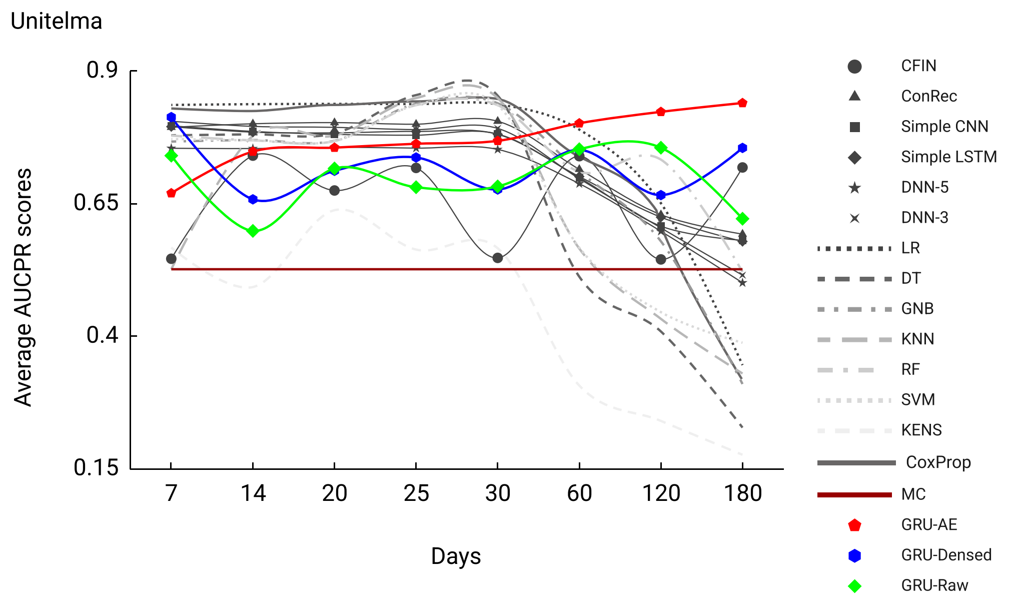


Figure 6.4. Average AUCPR scores on 30 runs for Unitelma. The curves without symbols represent off-the-shelf machine learning strategies. Meanwhile, the curves with symbols are deep learning approaches. Notice how the majority class (MC) approach is constant with the change of ℓ_i since the majority class does not change if the history per student trajectory is enlarged/shrunk.

windows increases over 30 days. As expected, our proposed method shows a higher capability of capturing long-term time dependencies. Figure 6.4 shows that all the state-of-the-art strategies have a non-decreasing trend with a history of 30 days for the temporal series. Successively, the performances of GRU-AE increase with the increment of ℓ_i; whereas the AUCPR curve of the compared methods has a steep decrease reaching worse performances than the naive MC baseline, especially that of simple machine learning approaches. This phenomenon in trend shifting of performances is due to the dependencies of e-tivities in longer periods. While simple machine learning techniques reach their peak performances at $\ell_5 = 30$, their decrease for ℓ_6, ℓ_7 and ℓ_8 suggests that these methods overfit the data, having to cope with an average per-day sparsity of $\sim 97\%$.

Furthermore, as mentioned in Section 6.1.2, machine learning methods take the flattened version of time-matrix $\mathcal{M}^{\ell_i,|\mathcal{E}|}$ in input, thus losing the time-information between the e-tivities. The performances decrease in the case of simpler methods and with deep and more complex ones. DNN-5 performs badly at the end of the time window (i.e. $\ell_8 = 180$) since it does not learn the temporal information coming from an embedded representation of the temporal series. Similarly, ConRec, whose performances are not significantly different from that of GRU-AE, according to the ANOVA test in KDDCup15, has an analogous performance curve as DNN-5. Finally, GRU-AE successfully extrapolates hidden information when the time window expands by merging dense and raw trajectories.

To summarise, Figures 6.2, 6.3 and 6.4 show that GRU-AE improves over the methods presented in Section 6.1.1, mostly in the scenario of long-term e-degrees.

	XuetangX	KDDCup15	Unitelma
CFIN	0.7233	0.8497	0.6539
ConRec	0.7676	<u>0.9349</u>	0.7427
Simple CNN	0.7470	0.8489	0.7267
Simple LSTM	0.7522	0.8486	0.7296
DNN-5	0.7829	**0.9357**	0.6950
DNN-3	0.7677	0.8697	0.7244
LR	0.7411	0.8719	0.7469
DT	0.7493	0.8897	0.6501
GNB	0.7346	0.8708	0.6981
KNN	0.7723	0.8740	0.6388
RF	0.7536	0.8824	0.7443
SVM	0.7316	0.8754	0.6729
KENS	0.7500	0.8897	0.4440
CoxProp	0.7562	0.8672	0.7418
MC	0.7121	0.8270	0.5265
GRU-AE	**0.7900**	<u>0.9346</u>	**0.7715**
GRU-Densed	0.7430	<u>0.9164</u>	0.7214
GRU-Raw	0.7637	0.9270	0.6938

Table 6.2. Average AUCPR values (over 30 runs) for all methods. A value is in bold if it is the best value on average; it is underlined when it is not significantly diverse from the best-performing model according to a one-way ANOVA test with post-hoc Tukey HSD with p-value of 0.01.

Nevertheless, to show no statistically significant difference between the performances of GRU-AE and the other top performing state-of-the-art methods, we performed a statistical significance evaluation relying on a one-way ANOVA test with a post-hoc Tukey HSD test with a p-value threshold equal to 0.01. The group observations that we give in input to the ANOVA test are the scores on 30 runs of each model. The hypothesis that we want to prove wrong with the one-way ANOVA is that *all models have the same average performances (i.e., $\mathcal{H}_0 : \mu_1, = ... =, \mu_n$).* Once the one-way ANOVA establishes that the means of the performances are different (i.e. $\mathcal{H}_0$ is rejected), we apply a Tukey HSD test to determine the pairs of models whose mean AUCPRs are statistically significantly unequal. Those pairs of models that have a difference between the average performances greater than the Tukey-criterion according to a p-value of 0.01 are considered statistically different; otherwise, the models are statistically indifferent. Then, we inspect the statistically different pairs that involve the best performing model on average. For completeness purposes, Table 6.2 reports the average AUCPR scores on all-time windows for each method in the datasets. For each dataset, the technique with the best performance score on average is bold-faced. The scores that are statistically and significantly indifferent to the best-performing model are underlined. The Table demonstrates that, on average, GRU-AE is the best system. Otherwise, GRU-AE does not show a statistically significant difference compared to the other best methods.

Finally, in Figure 6.5, we illustrate a strict comparison between GRU-AE and the second best-performing methods or those that have statistically significantly similar performances. Here, we depict each time-window ℓ_i for all datasets. We demonstrate

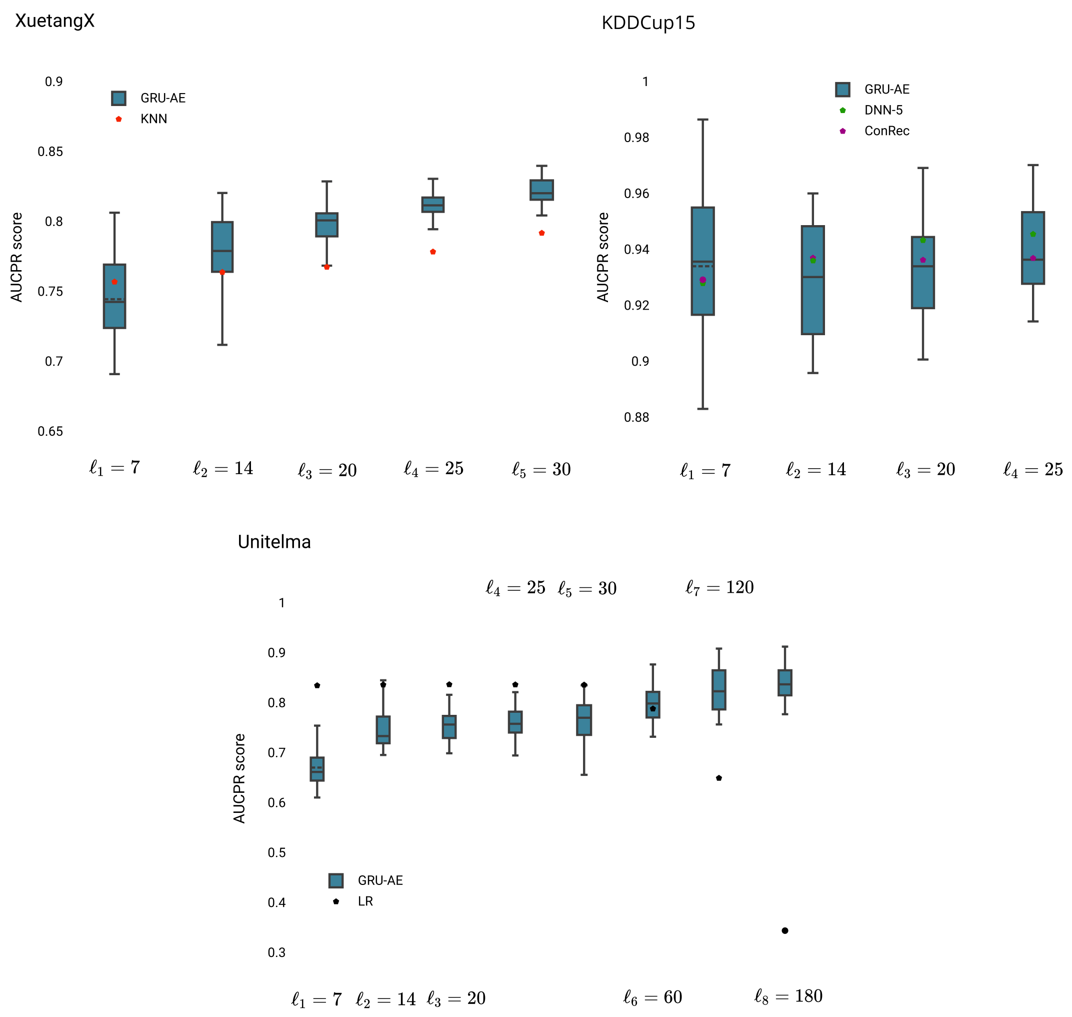


Figure 6.5. Distribution of performances of GRU-AE in all three datasets. To demonstrate the superiority (or statistically significant similarity) of GRU-AE for each dataset, we choose the second best-performing strategy on average or all similar strategies according to the ANOVA test. Here, we illustrate the AUCPR scores for every time window ℓ on 30 different runs. Notice that we do not represent the distribution of performances for the other methods. Rather, we show the mean AUCPR score for a specific ℓ_i.

how, when ℓ increases, the performances of the other methods either remain comparable or degrade abruptly (see Unitelma). Notice that, for visualisation purposes, we only provide the mean AUCPR score for the other methods for each ℓ_i. In the case of KDDCup15, DNN-5 and ConRec remain within the performance distribution of GRU-AE during the entire history of the student trajectories because, according to the one-way ANOVA test with the post-hoc Tukey test, they have statistically significantly similar performances (see Table 6.2). Contrarily, in XuetangX and Unitelma, k-nearest neighbours (KNN) and the linear regression (LR) reside under the mean performances and, successively, outside the box plots, clearly indicating a difference from the proposed strategy. Notice that the average performances of GRU-AE and the other compared methods follow the curve trend present in Figures

	XuetangX	KDDCup15	Unitelma
CFIN	0.6718	0.7719	0.4920
ConRec	0.7522	0.8422	0.5194
Simple CNN	0.7366	0.8340	0.6284
Simple LSTM	0.7379	0.8236	0.5167
DNN-5	0.7330	0.8194	0.5978
DNN-3	0.6972	0.8203	0.5554
LR	0.4948	0.7873	0.6535
DT	0.4487	0.4378	0.4299
GNB	0.7436	0.7849	0.6640
KNN	**0.7531**	0.7873	0.6423
RF	0.7523	0.4378	0.6987
SVM	0.7052	0.7849	0.4640
KENS	0.7513	0.7893	0.4640
CoxProp	0.5012	0.7779	0.6430
MC	0.6218	0.7849	0.4449
GRU-AE	0.7520	**0.8430**	**0.7346**
GRU-Densed	0.7473	0.7792	0.7283
GRU-Raw	0.7511	0.8164	0.7264

Table 6.3. Average F1 scores for 30 runs on all time-windows ℓ_i. A value is in bold if it is the best value on average; it is underlined when it is not significantly diverse from the best-performing model according to a one-way ANOVA test with post-hoc Tukey HSD with p-value of 0.01.

6.2, 6.3, and 6.4.

Precision, Recall, and F-measure of deep vs simple machine learning techniques. To assess the classifiers' general performances and resilience to false negatives, we also show F1 and Recall average scores. Table 6.3 shows the average F1 scores on 30 runs on all time-windows ℓ_i. As with the AUCPR scores, we performed a one-way ANOVA test to determine whether the performances are truly different or show negligible variance. The MOOC courses present interesting cases, with XuetangX having the highest number of methods whose performances do not have any statistical difference. Here, simple machine learning (i.e., KNN) and ensemble models (i.e., RF and KENS) have higher F1 scores.

Nevertheless, according to the ANOVA test, our method is statistically significantly similar to the previous methods on average. GRU-AE is the most performing method in KDDCup15, with ConRec having similar scores. Finally, as expected, our method excels in Unitelma being the sole system with the best performing strategy.

Furthermore, to show the absence of the statistical difference in F1 scores in XuetangX and KDDCup15 between GRU-AE and the other compared methods, we calculate the average recall score on 30 different runs on all-time windows ℓ_i. We remark that Recall is more important than Precision because, in critical scenarios such as recuperating a student who intends to withdraw from their studies, we can tolerate false positives but not false negatives. Figures 6.6, 6.7, and 6.8 clearly show that GRU-AE and its variants outperform the state-of-the-art strategies. As expected and demonstrated by Table 6.3, the average recall scores of GRU-AE in Unitelma lead by a margin of 25% over the best performing method (i.e., CFIN).

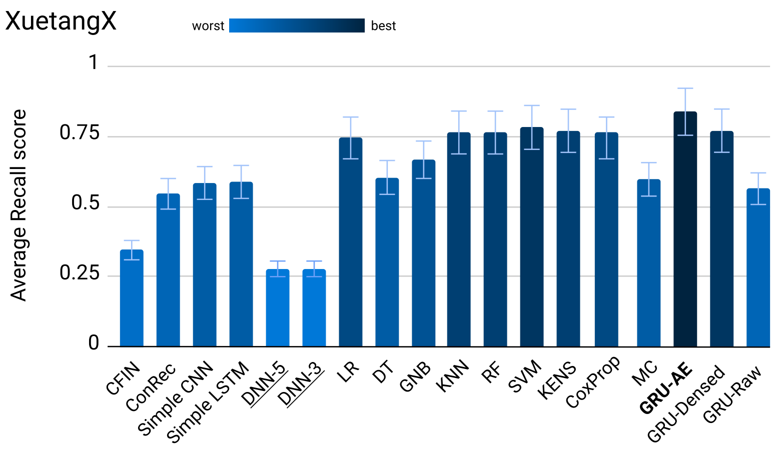


Figure 6.6. Average recall scores on 30 runs for XuetangX. The darkest bin corresponds to the highest recall score, meanwhile the lightest ones to the lowest. When two or more bins share the same shade, then their performances are statistically significantly similar. For illustration purposes, we present in bold the best-performing method, whereas in underscore the worst-performing one.

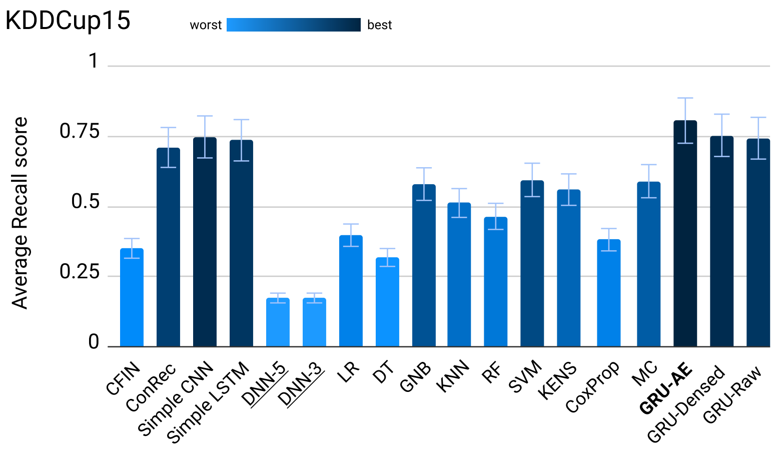


Figure 6.7. Average recall scores on 30 runs for KDDCup15. The darkest bin corresponds to the highest recall score, meanwhile the lightest ones to the lowest. When two or more bins share the same shade, then their performances are statistically significantly similar. For illustration purposes, we present in bold the best-performing method, whereas in underscore the worst-performing one. Notice that, differently from XuetangX, deep strategies have similarly distributed performances w.r.t. GRU-AE, the best-performing model. This aspect suggests that KDDCup15 has more regular student interactions in time such that there is an inter-dependency between the engendered e-tivities, hence the higher AUCPR scores in Table 6.2.

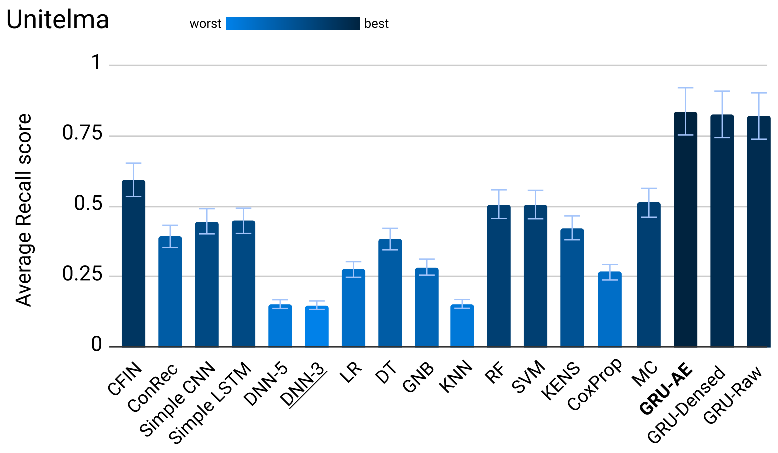


Figure 6.8. Average recall scores on 30 runs for Unitelma. The darkest bin corresponds to the highest recall score, meanwhile the lightest ones to the lowest. When two or more bins share the same shade, then their performances are statistically significantly similar. For illustration purposes, we present in bold the best-performing method, whereas in underscore the worst-performing one. Notice that the components - i.e. GRU-raw and GRU-densed - of the overall architecture and GRU-AE itself have visually detached performances from the rest of the strategies because of the long-term relationship between e-tivities in Unitelma

Sensitivity Analysis. We perform a sensitivity analysis to describe the benefits of the components in our framework. Here, we detach from the overall architecture one of the two components and measure the AUCPR of the two variants w.r.t. the entire model.Hence, we denote the model with *GRU-Densed* (see Figure 6.9) without the stacked raw GRU component in the architecture. Similarly, we denote the model with *GRU-Raw* (see Figure 6.10) without the stacked embedded GRU in the architecture.

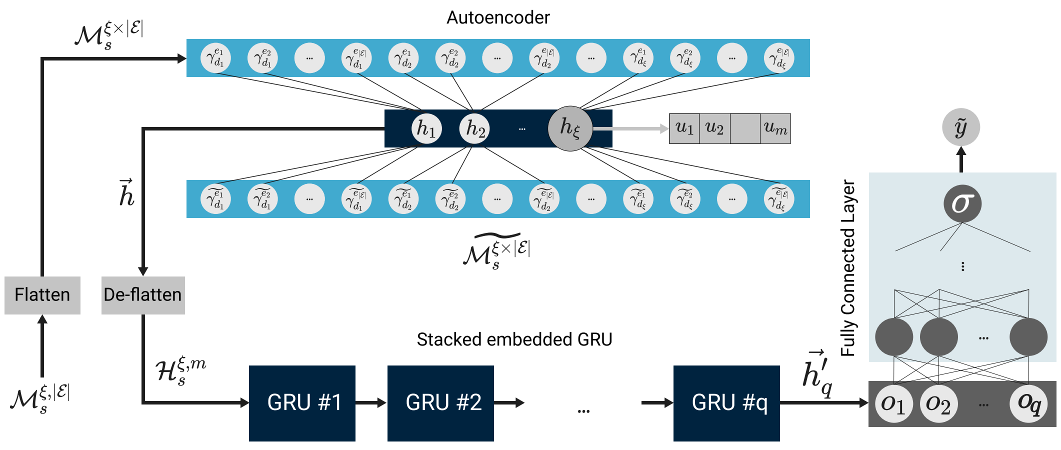


Figure 6.9. GRU-Densed component architecture.

Although the difference in AUCPRs and F1 is not that substantial (see the rightmost columns of Tables 6.2 and 6.3), the average daily sparsity and inter-e-tivity

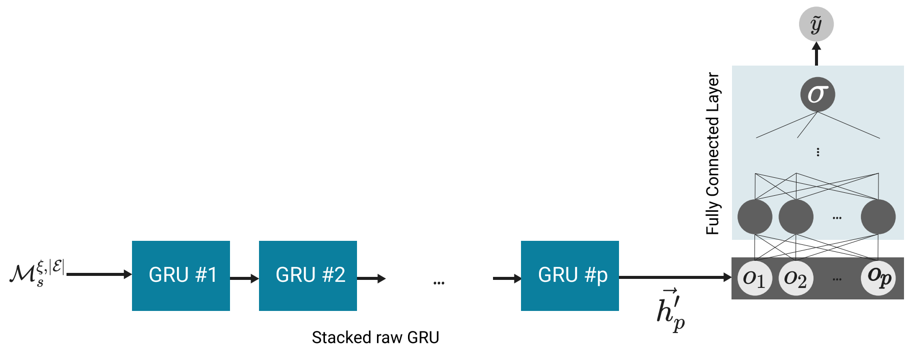


Figure 6.10. GRU-Raw component architecture.

daily gap induces GRU-Raw to underperform (on average) in Unitelma (see Table 6.2) while it performs better than GRU-Densed in both XuetangX and KDDCup15. We conclude that GRU-AE benefits from both GRU-Raw and GRU-Densed, noticing that combining these components into an overall architecture makes the model more complex and, thus, capable of extracting latent information and combining them in an end-to-end approach.

6.2.3 Reconstruction Error Analysis on Short and Long-Term Trajectories

While we have demonstrated that GRU-AE has better/comparable performances w.r.t. state-of-the-art methods, it is interesting to analyse the relationship between the reconstruction error of the reconstruction component (see Figure 5.2) and the overall performances of the architecture. Figure 6.11 illustrates the curves corresponding to the AUCPR scores and the reconstruction error for each time window $\ell_i \; \forall i \in [1, n]$. We report all values as averages of 30 different runs. Notice that the AUCPR scores are the same as those in Figures 6.2, 6.3, and 6.4. Whereas the reconstruction errors are reported as the MSE for every student trajectory averaged on 30 runs. In other words, the reconstruction error for a particular ℓ_i for a certain student s is

$$RMSE^s_{\ell_i} = \sqrt{\sum_{j=1}^{\ell_i} \sum_{k=1}^{|\mathcal{E}|} (\mathcal{M}_s^{j,k} - \widetilde{\mathcal{M}_s^{j,k}})^2}$$

where $\mathcal{M}_s^{j,k}$ represent the original value for the k-th e-tivity in day j for student s, and $\widetilde{\mathcal{M}_s^{j,k}}$ is the reconstructed counterpart. For visualisation purposes, we normalise the reconstruction error in the range [0,1] by applying a min-max scaling operator as follows:

$$NRMSE^s_{\ell_i} = \frac{RMSE^s_{\ell_i} - \min_{s \in \mathcal{S}} RMSE^s_{\ell_i}}{\max_{s \in \mathcal{S}} RMSE^s_{\ell_i}}$$

Finally, the normalised reconstruction error for all students $s \in \mathcal{S}$ is the mean

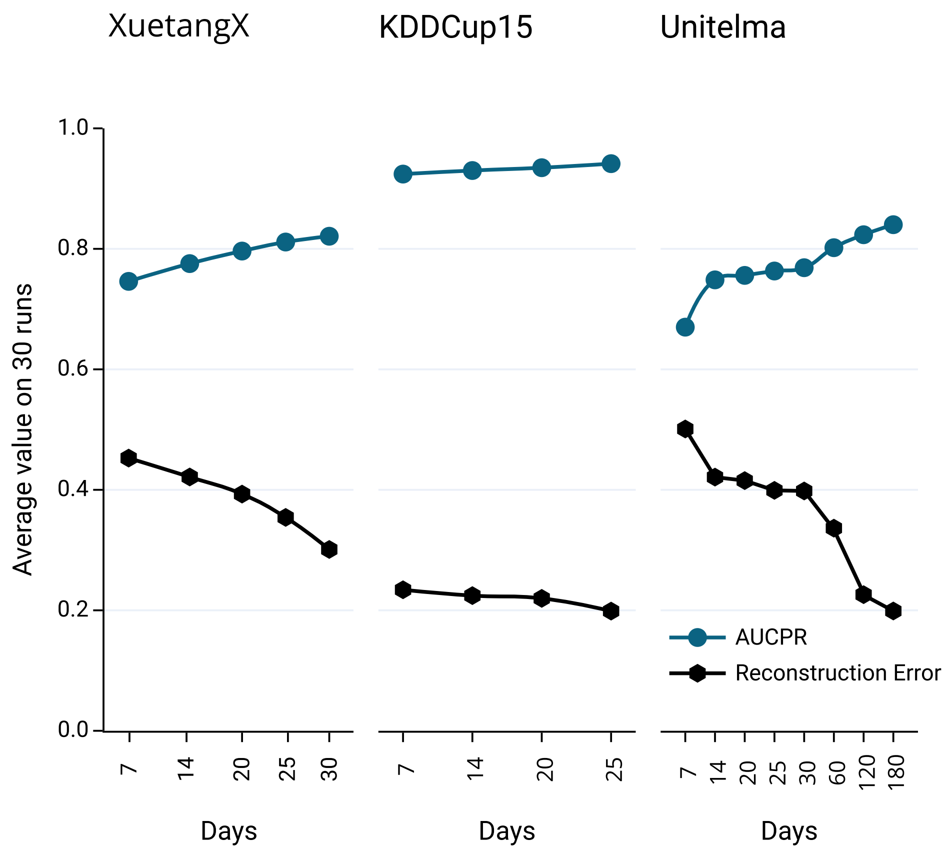


Figure 6.11. The relationship between the reconstruction error of the original trajectories and the average AUCPR score over 30 runs for GRU-AE. Notice that the reconstruction errors have been normalised in the range [0,1] for illustration purposes. The average AUCPR score is the same to the ones reported in the AUCPR trends in Figures 6.2, 6.3, 6.4. Meanwhile, the reconstruction error, for each $\ell_1, ..., \ell_n$, is the RMSE for all student trajectory averaged over 30 different runs.

reconstruction error of $NRMSE^s_{\ell_i}$ $\forall s \in \mathcal{S}$:

$$NRMSE_{\ell_i} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} NRMSE^s_{\ell_i}$$

As expected, while the AUCPR curve has a non-decreasing trend for all datasets, the reconstruction error has a non-increasing trend. The reconstruction component of GRU-AE plays an important role in boosting the performances of the overall architecture - hence the upwards trend of the AUCPR curve - while minimising the dissimilarity between the original student trajectory and the reconstructed one. Furthermore, notice how the reconstruction error curve has the steepest decreasing trend in Unitelma w.r.t. the other MOOC datasets. The reconstruction error decreases abruptly after considering time windows larger than 30 days, a similar tipping point for the AUCPR curve of GRU-AE in Figure 6.4. Additionally, notice

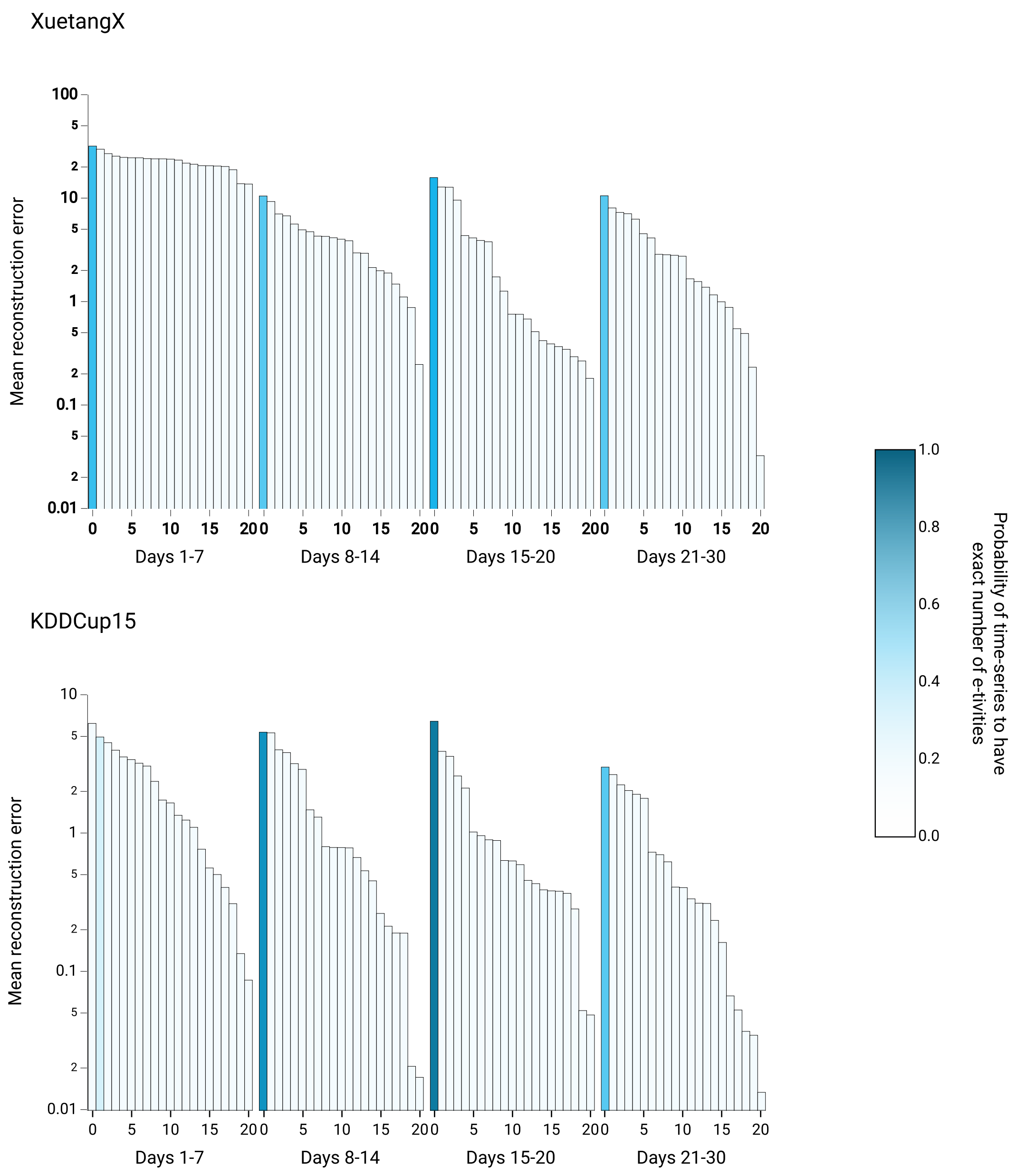


Figure 6.12. Average reconstruction error for time-series that contain exactly i e-tivities for specific time-intervals on XuetangX and KDDCup15. The x-axis in each subplot represents those trajectories with exactly i e-tivities engendered within that particular time-frame (e.g. Days 1-7 or Days 15-20). The y-axis illustrates the mean reconstruction error for all temporal series that contain exactly i e-tivities. The shades of each bin in the plot represent the probability of having exactly i e-tivities in all time-series of the datasets as in Figures 4.2 and 4.3. For visualisation purposes we illustrate this probability as a heatmap. The y-axis is log scaled.

the difference between the reconstruction errors in XuetangX and KDDCup15. The former has a clear downward trend, and the performances reach newer highs with each ℓ_i. Contrarily, in KDDCup15, the reconstruction component does not

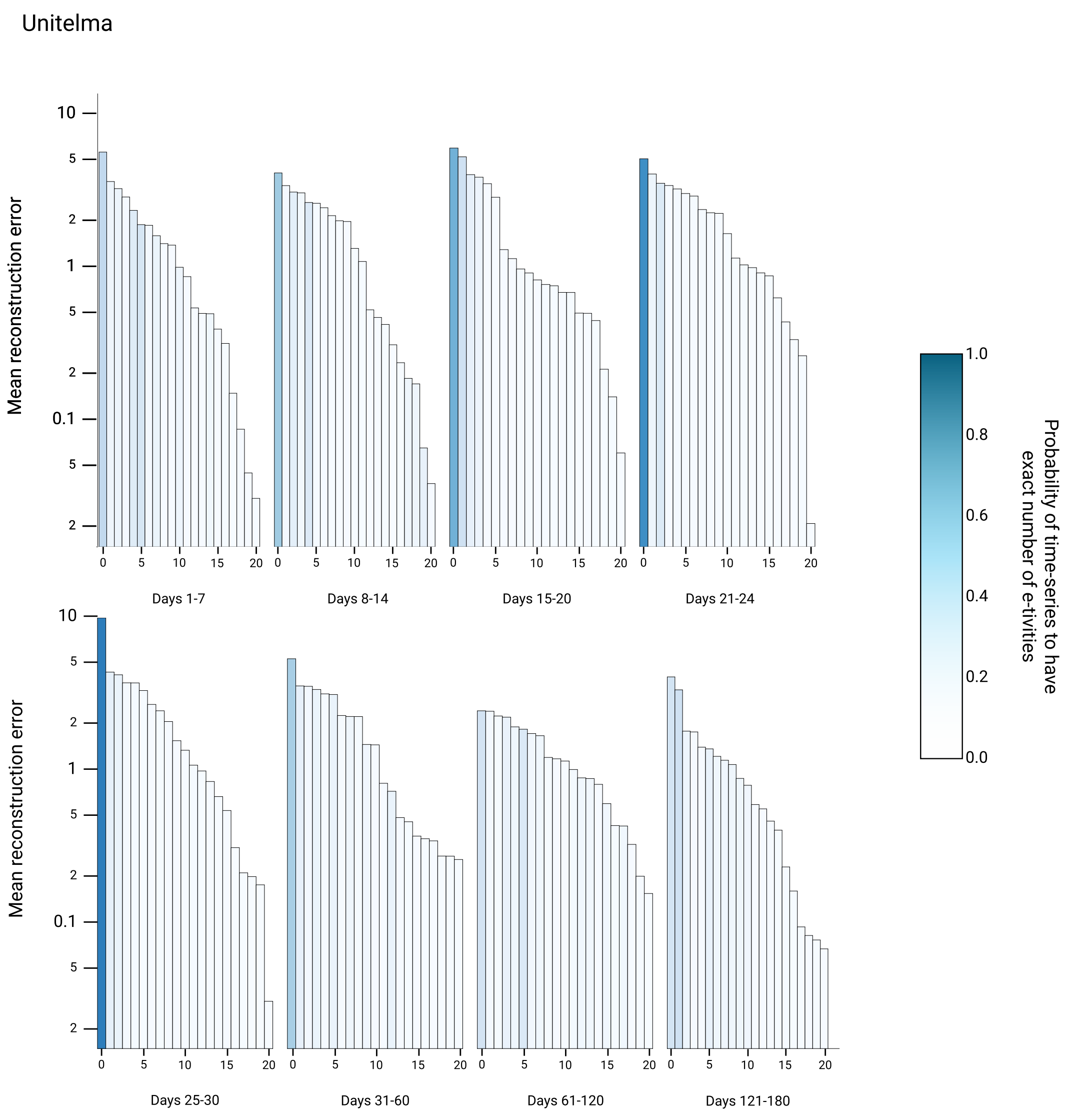


Figure 6.13. Average reconstruction error for time-series that contain exactly i e-tivities for specific time-intervals on Unitelma. The x-axis in each subplot represents those trajectories with exactly i e-tivities engendered within that particular time frame (e.g. Days 1-7 or Days 15-20). The y-axis illustrates the mean reconstruction error for all temporal series that contain exactly i e-tivities. The shades of each bin in the plot represent the probability of having exactly i e-tivities in all time-series of the datasets as in Figure 4.4. For visualisation purposes, we illustrate this probability as a heatmap. The y-axis is log scaled.

substantially impact the performances, and rather, it maintains a constant trend throughout the entire period of the history of the student time series. Without loss of generalisation, we can conclude that there is an inverse relationship between the overall performances of GRU-AE in terms of AUCPR scores and the average normalised reconstruction error of the input data.

To motivate the efficacy of our strategy on discriminating between dropouts and persisting students, in Figures 6.12 and 6.13, we depict the average reconstruction error for the student time series in specific time-intervals. For each time-interval $[d_b, d_f]$, we measure the average reconstruction error for every temporal-matrix $\mathcal{M}_s^{\xi,|\mathcal{E}|}$ of each student s in $[b, f]$. In other words, we use GRU-AE to reconstruct $\mathcal{M}_s^{[d_b,d_f],|\mathcal{E}|}$, the temporal matrix of student s projected in the time-interval with beginning day d_b and end day d_f. Notice how the average reconstruction error for those trajectories that do not contain any e-tivities is constantly high throughout all time intervals. Having a high reconstruction error for the time series that do not contain any e-tivities means that the reconstruction component of GRU-AE does not include them in the normal distribution of the other students and labels them as dropouts. Contrarily, the time-series with ten or more e-tivities generated throughout the interaction of a particular student s with the OLEs have a lower reconstruction error because they contain meaningful information that GRU-AE can exploit to determine the status of s.

Furthermore, in the scenario of MOOCs, the reconstruction error follows a non-increasing trend when the time-interval comes closer towards the maximum window length ℓ (see Table 4.1). Based on the performances in Table 6.2, it is clear that also the average reconstruction error entails a more difficult classification task for XuetangX than KDDCup15 (i.e. the errors for XuetangX are an order of magnitude bigger than KDDCup15). Meanwhile, the reconstruction error in the scenario of Unitelma follows a bell shape. While students are more interested in interacting with the course content at the beginning and ending of their trajectories, the middle suffers from the most sparsity, inducing our method's reconstruction error to become higher. However, even in the middle of the designated time intervals - i.e. days 21-24 and 25-30 - the trajectories containing more e-tivities present lower reconstruction errors than the others. In conclusion, we can determine that the reconstruction error in specific time intervals is directly proportionate to the probability of the time series containing a specific number of e-tivities (i.e. sparsity of the time series).

Chapter 7

Extending to the General Context of Symbolic Trajectory Prediction Applications

Besides the student dropout prediction problem, several other tasks are included in the more general category of symbolic trajectory prediction applications. We here consider the problem of survival prediction in critical care telehealth systems. Like SDP, predicting a particular patient's survival probability according to specific laboratory test results is a critical task. While students interact directly with OLEs to generate e-tivities, patients do not have any interaction patterns in this scenario. Nevertheless, patients are involved in several events such as laboratory examinations, periodical medical checks, and hospital recoveries in certain critical situations.

These personal files can be translated into time-series (i.e. every event generated by a particular patient is associated with a specific timestamp) relying on sequence labelling modelisation techniques described in Section 3.3.2. Thus, we have a similar problem as SDP, where dropouts become deceased patients, and persisters recover and are released from the hospital, suggesting how our input modelisation approaches are generalisable and tunable to any other trajectory prediction problem besides SDP.

7.1 eICU: A Fatality-based Telehealth Dataset

The medical dataset eICU describes fatalities of patients admitted to multi-centre critical care in US hospitals between 2014 and 2015. Patient-related events include laboratory tests, medications, admissions, physical examinations, and visits. We prune all patient trajectories, including zero events, which results in a final set of 65k trajectories. Similar to the datasets in the student dropout scenario, as suggested in [46], we perform a daily grouping of events concerning each patient. Therefore, we represent each individual's trajectory u with a time-matrix $\mathcal{M}_u^{\xi,|\mathcal{E}|}$ where $\xi << \ell$ is the length of the adopted time-window in days and $|\mathcal{E}|$ is the number of different events types (i.e., features) available in the dataset. In detail, a cell $(i, j) \in \mathcal{M}_u$ contains the number of events of type j found in day i of a trajectory u.

Num. of event types m	Max. time window length ℓ in days	Num. of time-series (trajectories)	Avg. per-day sparsity	Avg. day gap between events	Class distribution (0:1)
9	30	~65k	73.4%	8.11 ± 7.74	91.45% : 8.55%

Table 7.1. eICU dataset characteristics

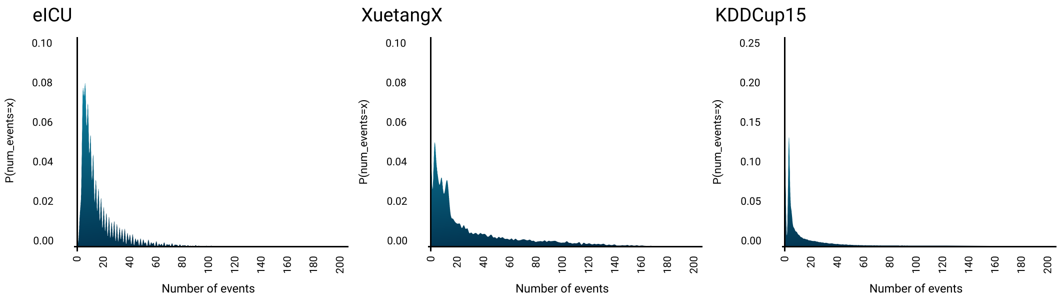


Figure 7.1. Probability mass function for the number of per-day events for each dataset on max-length sequences.

Table 7.1 summarises the characteristics of the eICU dataset similarly to those illustrated in Table 4.1. The Table shows that eICU exceeds the class imbalance of all time-related student dropout datasets with only 8.55% of fatalities (class 1). This severe imbalancement induces the patient survival prediction - henceforth PSP - problem to treat fatalities as anomalous instances in the dataset, thus making their identification more arduous in terms of evaluation. Contrarily, the average day gap between events within the same patient trajectory is more contained than those of SDP because eICU contains patients in critical health conditions posed to regular and periodical check-ups and blood tests. Additionally, notice that the average gap between events in eICU has the highest variance. These gaps are a common phenomenon in medical trajectories since events are not recorded in real-time, as instead happens in the e-learning domains. By not recording the events in real-time, eICU is a far more challenging domain than e-learning. In e-learning, an absence of events corresponds, being a relevant predictor of dropout, to a sporadic usage of the platform by the students. In the medical domain at hand, a gap with an absence of events is ambiguous: i.e. it may describe a healthy patient (class 0) that needed only some test or a critical patient that died - class 1 - during their hospitalisation.

Finally, Figure 7.1 shows the probability distribution of finding i daily events, no matter their type, in a randomly selected trajectory u for XuetangX, KDDCup15, and eICU. We do not consider Unitelma for this comparison because trajectories in eICU are short-term or at most have a one-year duration until either the death or the complete recovery from hospitalisation of a particular patient. Although all distributions are Zipfian, and KDDCup15 exhibits the steepest curve, the probability of at least 1-5 events is much higher (up to over 20%) w.r.t. the other datasets. Moreover, the probability mass function of eICU resembles that of XuetangX, with the difference that all patients have at least one event in their medical files. Therefore, we expect the performances of eICU to follow the same trend as those in XuetangX. All in all, the data analysis performed suggests that eICU represents a challenging PSP problem.

7.1.1 The Class Imbalance Effect on Hospital Fatalities

To analyse the effect on the performance of class imbalance - observed especially in the eICU dataset - we train our systems both with eICU "as is" and with an oversampling method called ADASYN [52], which is currently reported amongst the most effective. We oversample the minority class using five nearest neighbours for sampling with a balanced level of the synthetic samples $\beta = 1$. Thus, we balance the dataset such that the majority and minority classes have the same number of examples. Apart from this, we do not perform any other feature normalisation or pre-processing of any sort before training. The rest of the training hyperparameters are set as described in Section 6.1.2. Finally, we perform experiments similar to those described in Section 6.2 for the eICU dataset and compare the results.

Notice that we did not perform any class balancing in the educational datasets because the class imbalance in the SDP scenario is not as severe as in the e-health domain (compare the class distributions in Tables 4.1 and 7.1) . Additionally, ADASYN generates synthetic data based on k-nearest neighbours w.r.t. a specific instance belonging to the minority class. Thus, if x_u is randomly chosen from the set of neighbours of an instance x_i in the minority class, then ADASYN generates a synthetic instance $s_i = x_i + (x_u - x_i) \times \lambda$ where $\lambda \in [0, 1]$. We empirically verified that this generation of synthetic samples does not provide any information/performance gain for the methods presented in Section 6.1.1 in the educational datasets because the daily sparsity of the student sequences exceeds 90% and the average inter-activity day gap can span for over ~ 46 days (e.g. Unitelma's case). Moreover, the student sequences in the minority class contain at most 3 e-tivities throughout all their monitoring time. The issue is more pronounced when taking into consideration long-term e-degrees such as Unitelma. Hence, the neighbours of each of the instances belonging to the minority class do not contain enough variation for ADASYN to be able to generate meaningful synthetic samples. This particular challenge induces ADASYN to produce replicas of x_i scaled by λ and it impedes the prediction strategies to retrieve benefits from the balanced dataset. In other cases where the trajectories of persisting and dropout students do not have any significant difference[1], we witnessed an increase in the false positive rate, which, in turn, makes the AUCPR and F1 decrease substantially after applying ADASYN.

7.1.2 Experiments and Discussions

Figure 7.2 illustrates the average AUCPR scores on 30 runs for different time windows in the set $\{7, 14, 20, 25, 30\}$ for both eICU and eICU balanced with ADASYN. Similar to short-term MOOCs, all AUCPR curves in eICU have a non-decreasing trend when the number of days in the temporal matrix increases. The separation of GRU-AE from the rest of state-of-the-art methods is clearer when the portion of patient history becomes larger. Contrarily, in a perfectly balanced scenario where patient recoveries and fatalities have the exact number of instances, there is no clear

[1]In Unitelma, students collaborate and share study materials among each other outside the authorised OLEs. In other words, if a student s completes course c and downloads its materials (e.g. projects, video transcripts, minute files), s can disseminate them on social media without any repercussion. Additionally, this material spread among the peers of s can lead them to complete/fail c without having to interact with the e-platform, creating so-called *ghost* students.

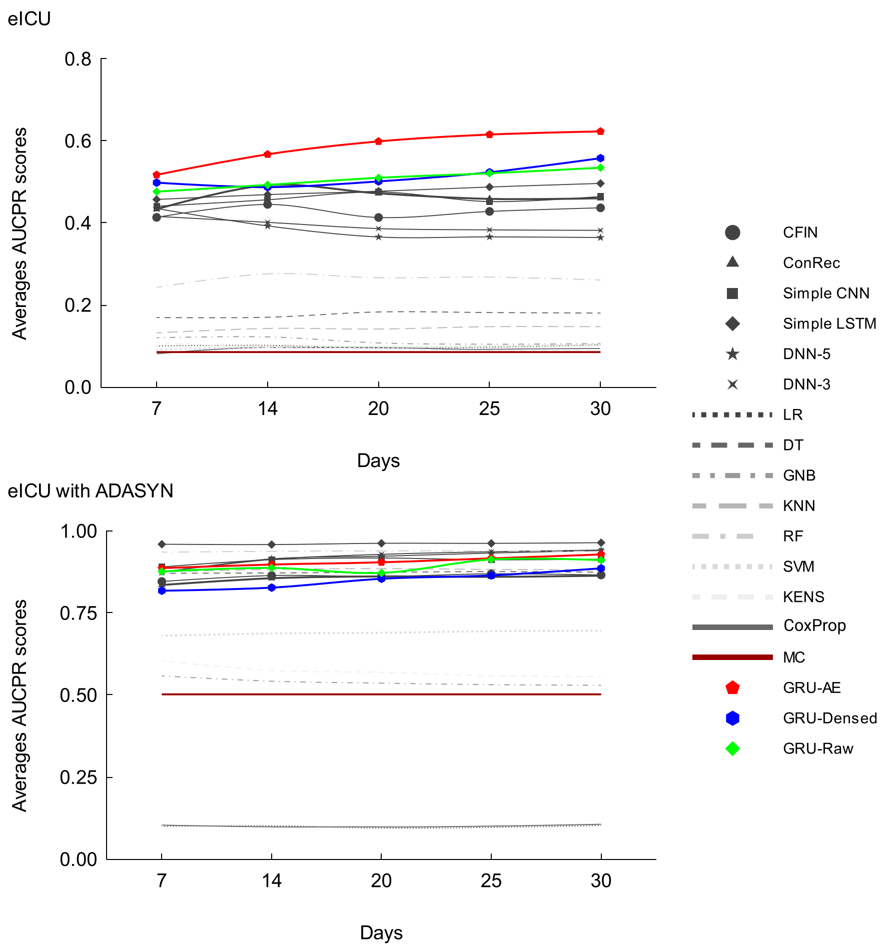


Figure 7.2. Average AUCPR scores on 30 runs for eICU (up) and eICU balanced with ADASYN (down).

distinction between simple machine and deep learning approaches. Additionally, although there is some upward trend going on all AUCPR curves, the patient history does not impact the performances as much as in the normal scenario of eICU.

Furthermore, because the average per-day event gap is not pronounced as in the e-learning domain, GRU-AE does not capture important latent information to have a great increase in performances (i.e. the performances remain approximately similar throughout the entire history of patient recoveries in hospital). Finally, the balanced eICU is an easier task also for simple learning techniques that, differently from the imbalanced version of the dataset, have comparable results. To prove this, in Table 7.2, we present the average AUCPR values for all time windows for eICU and eICU with ADASYN. While the one-way ANOVA test with post-hoc Tukey HSD does not entail any of the strategies to be significantly similar to the best-performing one in either eICU or eICU with ADASYN, notice how GRU-AE, being the best-performing method in eICU, on average does not compare with the other simpler methods (i.e. Simple CNN, Simple LSTM, DNN-5, and DNN-3). Additionally, in eICU with

	eICU	eICU with ADASYN
CFIN	0.4273	0.8613
ConRec	0.4633	0.8553
Simple CNN	0.4578	0.9093
Simple LSTM	0.4774	**0.9614**
DNN-5	0.3848	0.9167
DNN-3	0.3937	0.9190
LR	0.0993	0.0993
DT	0.1769	0.8735
GNB	0.1120	0.5397
KNN	0.1421	0.8815
RF	0.2628	0.9379
SVM	0.0977	0.6897
KENS	0.0950	0.5726
CoxProp	0.1019	0.0999
MC	0.0852	0.5022
GRU-AE	**0.5843**	0.9067
GRU-Densed	0.5136	0.8500
GRU-Raw	0.5069	0.8921

Table 7.2. Average AUCPR values (over 30 runs) for all methods for eICU and eICU with ADASYN. A value is in bold if it is the best value on average.

ADASYN, Random Forests is the second best-performing method meaning that ensemble strategies are useful to reduce the variance of the trajectories of deceased patients, thus incrementing the overall performances of the strategy (i.e. KENS has a ~ 6.1 times increment from eICU to eICU with ADASYN).

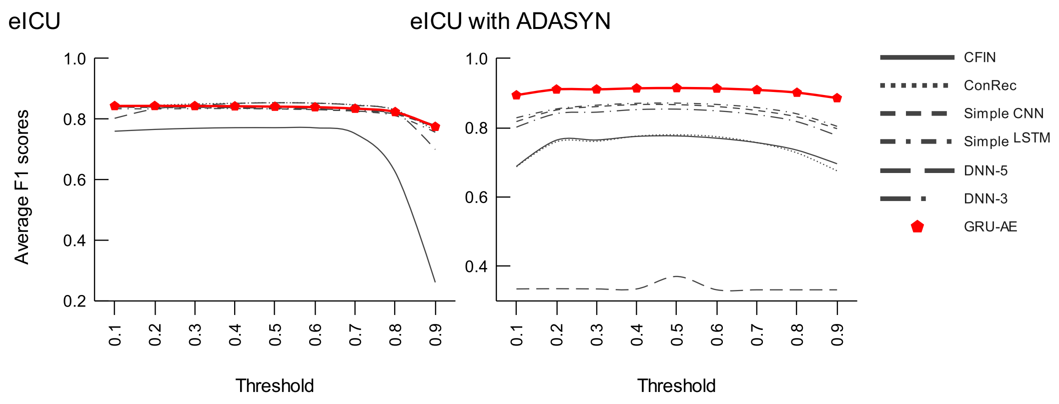


Figure 7.3. F1 scores averaged on all time-windows varying the threshold θ on eICU and eICU with ADASYN.

In order to assess the general performances and resilience of the classifiers to false negatives, we also show F1 and Recall - Figures 7.3 and 7.4 when varying the threshold θ introduced in Section 6.1.2. Notice that the trend of the curves[2] is

[2]We study only the scores according to the variation of θ for the deep strategies because surface

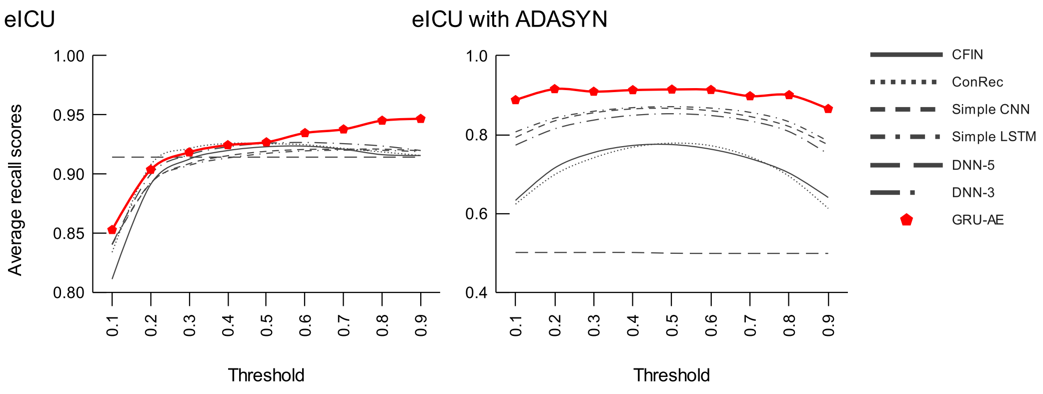


Figure 7.4. Recall score averaged on all time-windows varying the threshold θ on eICU and eICU with ADASYN.

non-decreasing when $\theta \to 1$ because the majority class of eICU is 0. The bell-shaped curve for eICU with ADASYN indicates that the classifiers' posterior probability distribution follows a uniform distribution with a mean approximately equal to 0.5. Therefore, one can think that the number of the posterior probabilities less than 0.5 is equal to those greater than or equal to 0.5. In fact, $P(x < 0.5) = P(x \geq 0.5)$ under a uniform distribution, justifying that the F1 and R scores reach their peak at $\theta = 0.5$ and have mirrored performances for the other thresholds.

As observed in Section 7.1, the trajectories in the medical domain eICU present several challenges, like the variegated number of events per trajectory (Figure 7.1), the variance of day-gaps and the high-class unbalance (see Table 7.1). Because eICU is the most challenging dataset, one expects that, for all methods, the average AUCPR scores would be lower than those in the other datasets. Figure 7.2 confirms this expectation. Now, consider Figures 7.3 and 7.4. Although we do not show precision curves, we can deduce that, since F1 and R are high, but AUCPRs are lower in eICU, the precision score must be small. On the other side, in risk prediction scenarios, such as a critical care system, we can tolerate false positives but not false negatives. Next, to study the effect of high-class imbalance on performance, we augment eICU with ADASYN. In Figure 7.2, the class balancing strategy entails a substantial increase (almost 40%) in terms of AUCPRs between the unbalanced and the balanced versions of eICU for all methods. In conclusion, when analysing the curves in Figures 7.3 and 7.4, GRU-AE, although not the best-performing strategy in terms of mean AUCPR scores, has the steadiest trend of them all in terms of F1 and Recall, respectively. Additionally, considering that false alarms - i.e. predicting that a particular patient might die instead of missing it - are important in a critical hospitalisation scenario, the recall curve of GRU-AE on eICU with ADASYN suggests that our method is the most suitable predictor for patient fatalities. Contrarily, GRU-AE is comparable with other methods until $\theta > 0.7$, where it reaches higher highs with $\theta \to 1$ (see the plot for eICU in Figure 7.4).

machine learning methods imply a horizontal line as their output (0/1) does not change with θ.

7.2 Other Temporal Health-Related Scenarios

Here, we describe additional examples of health-related scenarios where the properties of GRU-AE are being exploited in our ongoing work.

7.2.1 Prediction of viral spread

As of 2020's diffusion of coronavirus infectious cases, researchers have contributed to the literature on predicting the dissemination trend in specific countries. Most works use daily aggregated statistics[3] on the infection data such as number of recovered patients, number of deaths, and number of patients in intensive care. Additionally, with mobility patterns being publicly available[4,5], researchers have tested isolation policies issued by the governments to assess their efficacy in containing the virus's spread. In this manner, Aragona et al. [4, 5] have exploited - where applicable - daily statistics and mobility information. They forecast the daily positive cases at t time steps further than the reference window taken into consideration. However, the authors ignore the temporal modelisation of the mobility patterns. In other words, they use condensed features of movement transitions from one public area to the other. Therefore, we can exploit the movement patterns at each timestamp to create a time matrix similar to what is illustrated in Section 3.4. In this scenario, the student set $\mathcal{S}$ corresponds to the set of different countries taken into consideration - named Υ. Hence, $\Upsilon = \{\upsilon_1, \upsilon_2, \dots, \upsilon_{|\Upsilon|}\}$. Now, instead of having a time matrix $\mathcal{M}$ for each student $s \in \mathcal{S}$, we have a time matrix for each country $\upsilon_i \in \Upsilon$. For $\mathcal{M}$ to be appropriately adapted for this scenario, we have to model the set of e-tivities $\mathcal{E}$ to represent the possible actions that people can perform. In this case, we have means of transportation and areas in which people can transit. Hence, the set $\mathcal{A} = \{\alpha_1, \alpha_2, \dots, \alpha_{|\mathcal{A}|}\}$ corresponds to $\mathcal{E}$. Finally, the time matrix $\mathcal{M}_{\upsilon}^{\xi,\mathcal{A}}$ represents the mobility patterns $\alpha_i \in \mathcal{A}$ in each day for country $\upsilon \in \Upsilon$ in a period of ξ days. Notice that the target of this particular problem is to predict the daily positive cases (i.e. regression problem). By simply scaling the daily positive cases of each country to the interval [0,1], GRU-AE can be exploited to predict the infective cases of any country with daily mobility information.

7.2.2 Anomaly detection in behavioural sequences

Another interesting scenario where GRU-AE can be used with a few tweaks is that of elderly behaviour monitoring[6]. Older people (i.e. patients in retirement homes) can be tracked throughout their daily routine by introducing some non-invasive sensors and a smartwatch to communicate with them. Therefore, these patients can perform various activities such as going to the toilet, taking showers, and dining in the canteen. When the smartwatch is in the vicinity of the sensors representing the areas in which a specific activity occurs, then the tuple patient-activity gets

[3]`https://ourworldindata.org/coronavirus`

[4]`https://www.google.com/covid19/mobility/`

[5]`https://covid19.apple.com/mobility`

[6]This is based on the study conducted on the Lazio region project eLinus (Avviso Pubblico "Emergenza Coronavirus e oltre" - Domanda prot. n. A0376-2020-070051, CUP: F84E21000000006) in support to DATAWIZARD SRL.

persistently memorised at time t. The memorised activities can be ordered in time for each patient, thus represented as discrete temporal series. By analysing the trend shifts of the patient behaviours (i.e. action engenderings), we can stabilise whether anomalies happen throughout the monitoring period. Therefore, healthcare personnel can intervene promptly by treating these patients who frequently suffer from mental diseases or severe depression. The goal here is to predict whether there is an anomaly in the behaviour of the patient trajectory in a specific period (e.g. daily, weekly, monthly). As expected, the formalisation of the time matrix in SDP can also be utilised in this case. Here, the student set $\mathcal{S}$ becomes the set of patients $\mathcal{P} = \{\pi_1, \pi_2, \ldots, \pi_{|\mathcal{P}|}\}$. The set of e-tivities $\mathcal{E}$ corresponds to the set of actions the patients perform $\widehat{\mathcal{E}}$. Therefore, the time matrix $\mathcal{M}_{\pi}^{\xi,\widehat{\mathcal{E}}}$ depicts the daily aggregated history of patient π performing activities in $\widehat{\mathcal{E}}$ for ξ days. Similar to the previous scenario, GRU-AE is a possible solution to identify anomalous days in the behavioural pattern of elderly patients.

Chapter 8

Conclusion

Nowadays, education, and especially online education, is one of the top industries in the world. OLEs are increasingly gaining popularity because of several benefits of learning anywhere, anyplace, and anytime. Public and private universities and many corporations have adopted e-learning systems for employee training and learning to create a collaborative learning environment. However, despite the benefits of distance learning courses, educational institutions have a growing concern for low retention rates and, in general, low certification/graduation rates of these kinds of degrees. Students enrolled in online degree programs have a higher chance of dropping out than those attending a conventional classroom environment. For corporate universities, the main issue is the lack of motivation to learn for employees and the inability to decipher one's preferred learning style. It may lead to poor efficacy and alignment of employee needs with strategic organisational goals. Therefore, it is of paramount importance for both public and private institutions to increase the efficiency and efficacy of learning outcomes, both for social and economic reasons.

Early prediction of students at the risk of dropout is one of the challenging tasks researchers face in the learning analytics field. The purpose of this work was to present an in-depth analysis of student dropout prediction (SDP) in e-learning environments, under the central perspective, but not exclusive, of machine learning. We organised existing literature according to a hierarchical classification that follows the workflow of design choices in SDP. Furthermore, we introduced a formal notation to uniformly describe the alternative dropout models adopted by researchers in the field and model MOOCs and e-degrees. Besides synthesising the most promising predictive strategies presented in the literature, we analysed several additional, and yet crucial, issues such as the evaluation strategy, the identification of data sets available for comparative studies and the consideration of privacy issues.

State-of-the-art approaches to predicting student withdrawal in OLEs have focused primarily on the more common scenario of single MOOCs. However, the recent pandemic has widely expanded the ubiquity of OLE platforms, extending them to entire degree courses and, thus, introducing novel challenges. We presented GRU-AE to mitigate the problems of feature sparsity, long trajectories, and long temporal gaps during which students remain inactive that arise in the context of e-degrees. GRU-AE exploits an autoencoder to mitigate feature sparsity and stacked embedded GRUs to extract latent information and temporal dependencies even in the presence of long temporal gaps. Our methodology has been compared with state-of-the-art simple and deep architectures, both in the scenario of short and long-term courses. We have shown that our solution surpasses the others in long-term e-degrees and is not statistically different in comparison with the best methods in the context of MOOCs.

We also observed that deep methods, in general, outperform simple machine learning methods and ensembles in problems where temporal sequences of events as the input data. However, we demonstrated that the superiority of each deep method over the others is often minimal. At the same time, the highest impact on performance is either due to the complexity of input data (in particular, the ratio between the number of different features and the available dimension of training data) or to the application of data engineering methods (e.g. methods to cope with unbalanced classes). Furthermore, different performance indicators may tell a different story. The ranking of systems may significantly change with different evaluation measures that may be more or less appropriate depending upon the task objectives and the distribution of data samples. We deduce that researchers should carefully investigate the contexts in which one system is superior to the others by using datasets with different characteristics, variable parameters, and evaluation measures.

While SDP has been characterised as a binary classification problem, an anomaly detection approach on the time-dependent student time series is also viable. Dropout students are, in fact, time series that do not fit the normal distribution of the data, thus being outliers; meanwhile, persisting students are the inliers. Since GRU-AE already contains an autoencoding component in the overall architecture, we can exploit a temporal autoencoder (e.g. LSTM autoencoder) to reconstruct the input trajectory. Afterwards, the reconstruction of the input can be exploited to rank order the instances. The trajectories that are harder to reconstruct are expected to be at the top of the ordered list. The higher the reconstruction error, the more anomalous the time series is. Thus, we can employ a decision function (e.g. step function) to decide the dropout status of each student. This way, one can solve the SDP problem while referring to a completely different research field yet unexplored in Learning Analytics.

Because of the black-box nature of GRU-AE to predict the dropout status in OLEs, performing interpretability is arduous. Thus, as an extension to this work, predicting why students drop out from a specific online course/degree is of paramount importance in devising intervention strategies. Integrating a global attention mechanism in the prediction component of GRU-AE gives insight on which course phase (e.g. week, semester) gives the highest contribution to the dropout decision. Meanwhile, a local attention mechanism indicates which e-tivities have the highest impact preventing students from withdrawing early from a particular course. In the end, the attention mechanism provides a score ranking of each feature in the time matrix. It provides solid ground to design tailored study programmes for each student in the OLE.

Additionally, as an add-on to the autoencoding feature of GRU-AE, we aim to explore in our future studies the prediction uncertainty (according to Monte Carlo dropouts) concerning the reconstruction error. Model uncertainty can improve the identification of more subtle dropout cases where only the generation of e-tivities throughout the student interaction is not enough. Lastly, we will perform experiments on predicting the daily positive cases on the coronavirus infectious curves and the elderly daily behaviour patterns using our proposed method. More importantly, we will generalise the sequence input modelling to include series with timely ordered events, thus, providing a contribution to the time series analysis literature.

Bibliography

[1] Qasem A. Al-Radaideh, Emad M. Al-Shawakfa, and Mustafa I. Al-Najjar. Mining student data using decision trees. In *International Arab Conference on Information Technology (ACIT'2006), Yarmouk University, Jordan*, pages 1–5, 2006.

[2] Sattar Ameri, Mahtab J. Fard, Ratna B. Chinnam, and Chandan K. Reddy. Survival analysis based framework for early prediction of student dropouts. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 903–912. ACM, 2016.

[3] Bussaba Amnueypornsakul, Suma Bhat, and Phakpoom Chinprutthiwong. Predicting attrition along the way: The uiuc model. In *Proceedings of the EMNLP 2014 Workshop on Analysis of Large Scale Social Interaction in MOOCs*, pages 55–59, Doha, Qatar, 2014. Association for Computational Linguistics.

[4] Dario Aragona, Luca Podo, Bardh Prenkaj, and Paola Velardi. Coronna: a deep sequential framework to predict epidemic spread. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 10–17, 2021.

[5] Dario Aragona, Luca Podo, Bardh Prenkaj, and Paola Velardi. Latent and sequential prediction of the novel coronavirus epidemiological spread. *ACM SIGAPP Applied Computing Review*, 21(3):5–18, 2021.

[6] Gaurav Arora, Afshin Rahimi, and Timothy Baldwin. Does an lstm forget more than a cnn? an empirical study of catastrophic forgetting in nlp. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association*, pages 77–86, 2019.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[8] Behdad Bakhshinategh, Osmar R Zaiane, Samira El Atia, and Donald Ipperciel. Educational data mining applications and tasks: A survey of the last 10 years. *Education and Information Technologies*, 23(1):537–553, 2018.

[9] V.K. Balakrishnan. *Schaum's Outline of Graph Theory: Including Hundreds of Solved Problems*. McGraw Hill Professional, New York, NY, USA, 1997.

[10] Gaston Baudat and Fatiha Anouar. Generalized discriminant analysis using a kernel approach. *Neural computation*, 12(10):2385–2404, 2000.

[11] Papia Bawa. Retention in online courses: Exploring issues and solutions—a literature review. *Sage Open*, 6(1):2158244015621777, 2016.

[12] Johannes Berens, Kerstin Schneider, Simon Görtz, Simon Oster, and Julian Burghoff. Early detection of students at risk–predicting student dropouts using administrative student data and machine learning methods. Munich, Germany, 2018. CESifo Working Paper.

[13] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[14] Carl R Boyd, Mary Ann Tolson, and Wayne S Copes. Evaluating trauma care: the triss method. trauma score and the injury severity score. *The Journal of trauma*, 27(4):370–378, 1987.

[15] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[16] Leo Breiman. *Classification and regression trees.* Routledge, Abingdon, Oxfordshire, UK, 2017.

[17] Peter J. Brockwell, Richard A. Davis, and Matthew V. Calder. *Introduction to time series and forecasting*, volume 2. Springer, New York, NY, USA, 2002.

[18] Rebecca Brown, Collin Lynch, Yuan Wang, Michael Eagle, Jennifer Albert, Tiffany Barnes, Ryan Shaun Baker, Yoav Bergner, and Danielle S. McNamara. Communities of performance & communities of preference. In *CEUR Workshop Proceedings*, volume 1446, USA, 2015. CEUR-WS.

[19] Vicki Carter. Do media influence learning? revisiting the debate in the context of distance education. *Open Learning: The Journal of Open, Distance and e-Learning*, 11(1):31–40, 1996.

[20] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. *Semi-Supervised Learning.* The MIT Press, Cambridge, MA, USA, 2010.

[21] Jing Chen, Jun Feng, Xia Sun, Nannan Wu, Zhengzheng Yang, and Sushing Chen. Mooc dropout prediction using a hybrid algorithm based on decision tree and extreme learning machine. *Mathematical Problems in Engineering*, 2019, 2019.

[22] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 785–794, New York, NY, USA, 2016. ACM, ACM.

[23] Yujing Chen, Aditya Johri, and Huzefa Rangwala. Running out of stem: a comparative study across stem majors of college students at-risk of dropping

out early. In *Proceedings of the 8th International Conference on Learning Analytics and Knowledge*, pages 270–279. ACM, 2018.

[24] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[25] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[26] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[27] David R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, 1972.

[28] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, New York, NY, USA, 2006. ACM, ACM.

[29] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

[30] Gerben W. Dekker, Mykola Pechenizkiy, and Jan M. Vleeshouwer. Predicting students drop out: A case study. *International Working Group on Educational Data Mining*, 2009.

[31] Inderjit S Dhillon and Suvrit Sra. Generalized nonnegative matrix approximations with bregman divergences. In *NIPS*, volume 18. Citeseer, 2005.

[32] David P. Diaz. *Comparison of student characteristics, and evaluation of student success, in an online health education course.* PhD thesis, Nova Southeastern University, 2000.

[33] Mucong Ding, Kai Yang, Dit-Yan Yeung, and Ting-Chuen Pong. Effective feature learning with unsupervised learning for improving the predictive models in massive open online courses, 2018.

[34] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[35] Mi Fei and Dit-Yan Yeung. Temporal models for predicting student dropout in massive open online courses. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 256–263. IEEE, 2015.

[36] Wenzheng Feng, Jie Tang, and Tracy Xiao Liu. Understanding dropouts in moocs. In *AAAI 2019*, 2019.

[37] Jonathan E Finkelstein. *Learning in real time: Synchronous teaching and learning online*, volume 5. John Wiley & Sons, 2006.

[38] Karen Frankola. Why online learners drop out. *WORKFORCE-COSTA MESA-*, 80(10):52–61, 2001.

[39] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[40] S Hari Ganesh and A Joy Christy. Applications of educational data mining: a survey. In *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pages 1–6. IEEE, 2015.

[41] Elena Gaudioso, Miguel Montero, and Felix Hernandez-Del-Olmo. Supporting teachers in adaptive educational systems through predictive models: A proof of concept. *Expert Systems with Applications*, 39(1):621–625, 2012.

[42] Niki Gitinabard, Farzaneh Khoshnevisan, Collin F. Lynch, and Elle Yuan Wang. Your actions or your associates? predicting certification and dropout in moocs with behavioral and social features, 2018.

[43] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[44] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[45] Cameron C. Gray and Dave Perkins. Utilizing early engagement and machine learning to predict student outcomes. *Computers & Education*, 131:22–32, 2019.

[46] Liu Haiyang, Zhihai Wang, Phillip Benachour, and Philip Tubman. A time series classification method for behaviour-based dropout prediction. In *2018 IEEE 18th International Conference on Advanced Learning Technologies (ICALT)*, pages 191–195. IEEE, 2018.

[47] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In José Mira and Francisco Sandoval, editors, *From Natural to Artificial Neural Computation*, pages 195–201, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[48] David J Hand and Keming Yu. Idiot's bayes—not so stupid after all? *International statistical review*, 69(3):385–398, 2001.

[49] Frank E Harrell. Cox proportional hazards regression model. In *Regression modeling strategies*, pages 475–519. Springer, 2015.

[50] HarvardX. HarvardX Person-Course Academic Year 2013 De-Identified dataset, version 3.0, 2014.

[51] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[52] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE IJCNN*, pages 1322–1328. IEEE, 2008.

[53] Jiazhen He, James Bailey, Benjamin I.P. Rubinstein, and Rui Zhang. Identifying at-risk students in massive open online courses. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[54] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, New York, NY, USA, 1992.

[55] Michael Herbert. Staying the course: A study in online student satisfaction and retention. *Online Journal of Distance Learning Administration*, 9(4):300–317, 2006.

[56] Erin Heyman. Overcoming student retention issues in higher education online programs. *Online Journal of Distance Learning Administration*, 13(4), 2010.

[57] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[58] David W Hosmer and Stanley Lemeshow. *Applied survival analysis: regression modelling of time to event data*. Wiley, 2002.

[59] Deng Houtao, Runger C. George, Tuv Eugene, and Martyanov Vladimir. A time series forest for classification and feature extraction. *Inf. Sci.*, 239:142–153, 2013.

[60] Qian Hu and Huzefa Rangwala. Academic performance estimation with attention-based graph convolutional networks. *International Educational Data Mining Society*, 2019.

[61] Ya-Han Hu, Chia-Lun Lo, and Sheng-Pao Shih. Developing early warning systems to predict students' online learning performance. *Computers in Human Behavior*, 36:469–478, 2014.

[62] Ozan İrsoy and Claire Cardie. Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 720–728, Doha, Qatar, October 2014. Association for Computational Linguistics.

[63] Cristina Isidro, Rosa M Carro, and Alvaro Ortigosa. Dropout detection in moocs: An exploratory analysis. In *2018 International Symposium on Computers in Education (SIIE)*, pages 1–6. IEEE, 2018.

[64] Christiana Kartsonaki. Survival analysis. *Diagnostic Histopathology*, 22(7):263–270, 2016.

[65] Gordon V Kass. An exploratory technique for investigating large quantities of categorical data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 29(2):119–127, 1980.

[66] Tom Kasuba. Simplified fuzzy artmap. *AI Expert*, 1993.

[67] Usha Keshavamurthy and H. S. Guruprasad. Learning analytics: A survey. *International Journal of Computer Trends and Technology (IJCTT)*, 18(6), 2014.

[68] Angelika Kimmig, Stephen Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. A short introduction to probabilistic soft logic. In *Proceedings of the NIPS Workshop on Probabilistic Programming: Foundations and Applications*, pages 1–4, 2012.

[69] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[70] David G Kleinbaum and Mitchel Klein. The cox proportional hazards model and its characteristics. In *Survival analysis*, pages 97–159. Springer, 2012.

[71] Jon M Kleinberg, Mark Newman, Albert-László Barabási, and Duncan J Watts. *Authoritative sources in a hyperlinked environment*. Princeton University Press, 2011.

[72] Marius Kloft, Felix Stiehler, Zhilin Zheng, and Niels Pinkwart. Predicting mooc dropout over weeks using machine learning methods. In *Proceedings of the EMNLP 2014 Workshop on Analysis of Large Scale Social Interaction in MOOCs*, pages 60–65, 2014.

[73] Georgios Kostopoulos, Sotiris Kotsiantis, and Panagiotis Pintelas. Estimating student dropout in distance higher education using semi-supervised techniques. In *Proceedings of the 19th Panhellenic Conference on Informatics*, pages 38–43, New York, NY, USA, 2015. ACM, ACM.

[74] S. Kotsiantis, C. Pierrakeas, and P. Pintelas. Predicting students' performance in distance learning using machine learning techniques. *Applied Artificial Intelligence*, 18(5):411–426, 2004.

[75] Sotiris Kotsiantis, Kiriakos Patriarcheas, and Michalis Xenos. A combinational incremental ensemble of classifiers as a technique for predicting students' performance in distance education. *Knowledge-Based Systems*, 23(6):529–535, 2010.

[76] Sotiris Kotsiantis, Christos Pierrakeas, and Panagiotis Pintelas. Preventing student dropout in distance learning using machine learning techniques. In *International conference on knowledge-based and intelligent information and engineering systems*, pages 267–274, New York, NY, USA, 2003. Springer.

[77] Sotiris Kotsiantis, Christos Pierrakeas, Ioannis Zaharakis, and Panagiotis Pintelas. *Efficiency of machine learning techniques in predicting students performance in distance learning systems*, pages 297–306. University of Patras Press, 2003.

[78] Zlatko J. Kovačić. Early prediction of student success: Mining student enrollment data. In *Proceedings of Informing Science & IT Education Conference*. Citeseer, 2010.

[79] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.

[80] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[81] A Latif, AI Choudhary, and AA Hammayun. Economic effects of student dropouts: A comparative study. *Journal of global economics*, 2015.

[82] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[83] R. J Lewis. An introduction to classification and regression tree (cart) analysis. In *Annual meeting of the society for academic emergency medicine*, volume 14, San Francisco, California, USA, 2000.

[84] Wentao Li, Min Gao, Hua Li, Qingyu Xiong, Junhao Wen, and Zhongfu Wu. Dropout prediction in moocs using behavior features and multi-view semi-supervised learning. In *2016 international joint conference on neural networks (IJCNN)*, pages 3130–3137. IEEE, 2016.

[85] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

[86] Chu Kiong Loo and MVC Rao. Accurate and reliable diagnosis and classification using probabilistic ensemble simplified fuzzy artmap. *IEEE Transactions on Knowledge and Data engineering*, 17(11):1589–1593, 2005.

[87] Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying relu and initialization: Theory and numerical examples. *Communications in Computational Physics*, 28(5):1671–1706, 2020.

[88] Ioanna Lykourentzou, Ioannis Giannoukos, Vassilis Nikolopoulos, George Mpardis, and Vassili Loumos. Dropout prediction in e-learning courses through the combination of machine learning techniques. *Computers & Education*, 53(3):950–965, 2009.

[89] S Madeh Piryonesi and Tamer E El-Diraby. Using machine learning to examine impact of type of performance indicator on flexible pavement deterioration modeling. *Journal of Infrastructure Systems*, 27(2):04021005, 2021.

[90] Majid Bashir Malik, Asger M. Ghazi, and Rashid Ali. Privacy preserving data mining techniques: current scenario and future prospects. In *2012 Third International Conference on Computer and Communication Technology*, pages 26–32. IEEE, 2012.

[91] Laci Mary Barbosa Manhães, Sérgio Manuel Serra da Cruz, and Geraldo Zimbrão. Wave: an architecture for predicting dropout in undergraduate courses using edm. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pages 243–247, New York, NY, USA, 2014. ACM, ACM.

[92] Mary McHugh. Interrater reliability: The kappa statistic. *Biochemia medica : časopis Hrvatskoga društva medicinskih biokemičara / HDMB*, 22:276–282, 10 2012.

[93] Othon Michail. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, 12(4):239–280, 2016.

[94] John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine learning*, 4(2):227–243, 1989.

[95] Saurabh Nagrecha, John Z. Dillon, and Nitesh V. Chawla. Mooc dropout prediction: lessons learned from making pipelines interpretable. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 351–359. International World Wide Web Conferences Steering Committee, 2017.

[96] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[97] Mark EJ Newman. A measure of betweenness centrality based on random walks. *Social networks*, 27(1):39–54, 2005.

[98] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

[99] Alvaro Ortigosa, Rosa M Carro, Javier Bravo-Agapito, David Lizcano, Juan Jesús Alcolea, and Oscar Blanco. From lab to production: Lessons learnt and real-life challenges of an early student-dropout prevention system. *IEEE transactions on learning technologies*, 12(2):264–277, 2019.

[100] Mahesh Pal. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222, 2005.

[101] Mahesh Pal and Paul M. Mather. An assessment of the effectiveness of decision tree methods for land cover classification. *Remote sensing of environment*, 86(4):554–565, 2003.

[102] Karl Pearson. X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, 1900.

[103] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[104] Alejandro Peña-Ayala. Educational data mining: A survey and a data mining-based analysis of recent works. *Expert systems with applications*, 41(4):1432–1462, 2014.

[105] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[106] Bardh Prenkaj, Damiano Distante, Stefano Faralli, and Paola Velardi. Hidden space deep sequential risk prediction on student trajectories. *Future Generation Computer Systems*, 125:532–543, 2021.

[107] Bardh Prenkaj, Giovanni Stilo, and Lorenzo Madeddu. Challenges and solutions to the student dropout prediction problem in online courses. In Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux, editors, *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pages 3513–3514. ACM, 2020.

[108] Bardh Prenkaj, Paola Velardi, Giovanni Stilo, Damiano Distante, and Stefano Faralli. A survey of machine learning approaches for student dropout prediction in online courses. *ACM Computing Surveys*, 53-3, 2020.

[109] Jiezhong Qiu, Jie Tang, Tracy Xiao Liu, Jie Gong, Chenhui Zhang, Qian Zhang, and Yufei Xue. Modeling and predicting learning behavior in moocs. In *Proceedings of the ninth ACM international conference on web search and data mining*, pages 93–102, New York, NY, USA, 2016. ACM, ACM.

[110] Lin Qiu, Yanshen Liu, Quan Hu, and Yi Liu. Student dropout prediction in massive open online courses by convolutional neural networks. *Soft Computing*, pages 1–15, 2018.

[111] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[112] Arti Ramesh, Dan Goldwasser, Bert Huang, Hal Daume III, and Lise Getoor. Learning latent engagement patterns of students in online courses. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[113] Carly Robinson, Michael Yeomans, Justin Reich, Chris Hulleman, and Hunter Gehlbach. Forecasting student achievement in moocs with natural language

processing. In *Proceedings of the sixth international conference on learning analytics & knowledge*, pages 383–387, New York, NY, USA, 2016. ACM, ACM.

[114] Lior Rokach and Oded Z. Maimon. *Data mining with decision trees: theory and applications*, volume 69. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2nd edition, 2014.

[115] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *Plos One*, 10(3):1–21, 03 2015.

[116] K. Saranya, K. Premalatha, and SS. Rajasekar. A survey on privacy preserving data mining. In *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, pages 1740–1744. IEEE, 2015.

[117] Robert E. Schapire. Explaining adaboost. In *Empirical inference*, pages 37–52. Springer, New York, NY, USA, 2013.

[118] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[119] Bernhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert. *Kernel methods in computational biology.* MIT press, 2004.

[120] Aobakwe Senosi and George Sibiya. Classification and evaluation of privacy preserving data mining: a review. In *2017 IEEE AFRICON*, pages 849–855. IEEE, 2017.

[121] Belinda G. Smith. *E-learning technologies: A comparative study of adult learners enrolled on blended and online campuses engaging in a virtual classroom.* PhD thesis, Capella University, 2010.

[122] Maria Stetter. Best practices in asynchronous online instruction. In *Society for Information Technology & Teacher Education International Conference*, pages 245–247. Association for the Advancement of Computing in Education (AACE), 2018.

[123] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[124] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[125] Steven Tang and Zachary A Pardos. Personalized behavior recommendation: A case study of applicability to 13 courses on edx. In *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 165–170, 2017.

[126] Colin Tankard. What the gdpr means for businesses. *Network Security*, 2016(6):5–8, 2016.

[127] Colin Taylor, Kalyan Veeramachaneni, and Una-May O'Reilly. Likely to stop? predicting stopout in massive open online courses, 2014.

[128] Terry M Therneau and Patricia M Grambsch. The cox model. In *Modeling survival data: extending the Cox model*, pages 39–77. Springer, 2000.

[129] Juliana Tolles and William J Meurer. Logistic regression: relating patient characteristics to outcomes. *Jama*, 316(5):533–534, 2016.

[130] Thibaut Vidal and Maximilian Schiffer. Born-again tree ensembles. In *International Conference on Machine Learning*, pages 9743–9753. PMLR, 2020.

[131] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10:3152676, 2017.

[132] Wei Wang, Han Yu, and Chuyan Miao. Deep model for dropout prediction in moocs. In *Proceedings of the 2nd International Conference on Crowd Science and Engineering*, pages 26–32, New York, NY, USA, 2017. ACM, ACM.

[133] Annika Wolff, Zdenek Zdrahal, Andriy Nikolov, and Michal Pantucek. Improving retention: predicting at-risk students by analysing clicking behaviour in a virtual learning environment. In *Proceedings of the third international conference on learning analytics and knowledge*, pages 145–149, New York, NY, USA, 2013. ACM, ACM.

[134] Michalis Xenos, Christos Pierrakeas, and Panagiotis Pintelas. A survey on student dropout rates and dropout causes concerning the students in the course of informatics of the hellenic open university. *Computers & Education*, 39(4):361–377, 2002.

[135] Wanli Xing and Dongping Du. Dropout prediction in moocs: Using deep learning for personalized intervention. *Journal of Educational Computing Research*, 57(3):547–570, 2019.

[136] Diyi Yang, Tanmay Sinha, David Adamson, and Carolyn Penstein Rosé. Turn on, tune in, drop out: Anticipating student dropouts in massive open online courses. In *Proceedings of the 2013 NIPS Data-driven education workshop*, volume 11, page 14, USA, 2013. Curran Associates, Inc.

[137] Jingyu Zhao, Feiqing Huang, Jia Lv, Yanjie Duan, Zhen Qin, Guodong Li, and Guangjian Tian. Do RNN and LSTM have long memory? In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11365–11375. PMLR, 13–18 Jul 2020.

[138] Zhi-Hua Zhou and Ming Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge & Data Engineering*, 17:11:1529–1541, 2005.

[139] Mengxiao Zhu, Yoav Bergner, Yan Zhan, Ryan Baker, Yuan Wang, and Luc Paquette. Longitudinal engagement, performance, and social connectivity: a mooc case study using exponential random graph models. In *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*, pages 223–230, New York, NY, USA, 2016. ACM, ACM.