Diamantini *et al. Process Science* (2025) 2:17
https://doi.org/10.1007/s44311-025-00023-7

Process Science

RESEARCH Open Access

Predicting students' academic performance based on early career behaviors: a process-aware approach

Claudia Diamantini[1], Laura Genga[2*], Alessandro Mele[1], Alex Mircoli[3] and Domenico Potena[1]

*Correspondence:
Laura Genga
l.genga@tue.nl
[1]Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche, Ancona, Italy
[2]Department of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology, Eindhoven, The Netherlands
[3]Dipartimento di Scienze Economiche e Sociali, Università Politecnica delle Marche, Ancona, Italy

Abstract
Educational Process Mining aims at supporting educational processes by leveraging historical data of students' behaviors. In this work, we show how to leverage behaviors characterizing students' first year of university to predict whether they will graduate on time. We transform the sequence of exams a student has taken into an Instance Graph modeling both sequential and concurrent behaviors. We then compare two prediction approaches, i.e., i) providing the graphs as input to a graph neural network, and ii) extracting relevant subprocesses used as features for traditional machine learning approaches. The results show that graph neural network performs best, achieving a 93.56% accuracy against the 82.43% achieved by machine learning approaches using subprocesses. Nevertheless, we advocate that these subprocesses have value both to provide descriptive insights on students' early careers and to explain the results of the graph neural network.

Keywords Educational process mining, Curriculum mining, Student performance analysis, Subprocesses

Introduction

Today's universities are increasingly committed to continuous efforts to enhance their educational programs and improve their quality while reducing delays and dropouts. This phenomenon is particularly critical for Italian universities: on average, only 61.3% of students enrolled in a three-year bachelor's degree program complete their studies within the prescribed duration, while a further 29% graduate one to two years beyond the standard period[1].

The increasingly widespread use of digital platforms to support university teaching processes has boosted the development of evidence-based techniques to support this effort in recent years. Among these, a discipline that is gaining increasing attention is *Educational Process Mining* (EPM). The goal of EPM is to uncover patterns and trends within educational data to understand how educational processes are carried out and to

[1] https://www.almalaurea.it/en/our-data/almalaurea-surveys/graduates-profile

identify improvement opportunities (Bogarín et al. 2018). In this study, we focus on a specific branch of EPM, the so-called "curriculum mining", whose goal consists of analyzing data related to students' *careers*, i.e., the sequence of registrations of credits-bearing activities, to gain valuable insights on the curricula chosen by them. Previous studies show how these techniques can facilitate understanding behaviors that characterize students' academic performance, e.g., graduation times (Cameranesi et al. 2017; Janssenswillen 2021).

Our study showcases the application of these techniques to answer the following research question: "How does a student's exam-taking behavior in the first year impact their graduation performance?" The "exam-taking behavior" consists of the sequences of activities modeling each exam's successful or unsuccessful attempt. Namely, each activity in this process corresponds to the outcome obtained by a student who enrolled for an exam attempt. Such outcome can be positive (i.e., the students *passed* the exam), or negative (i.e., the student *failed* the exam). We chose to focus on this behavior instead of, e.g., aggregated indicators such as course enrollment, to investigate the characteristics of students' academic pathways. This task is inherently challenging due to the heterogeneity of the pathways followed by students, which stems from the lack of constraints in the exam-taking process. Indeed, while the scope and the order of courses to attend within a program are determined by the curriculum, there is usually quite some flexibility for the students in deciding when to take each exam, for which multiple opportunities are offered during the academic year, resulting in many possible paths to graduation. Note that there is no limit to how many times a student may take an exam. For every exam, a minimum of seven opportunities per year are mandatory. Additional opportunities might be offered, depending on the teachers' availability and preferences. For every exam, a minimum of seven opportunities per year are mandatory. Additional opportunities might be offered, depending on the teachers' availability and preferences. Such flexibility, while beneficial in principle, can cause some struggle for the students when determining the next exam to take, and at the same time, it makes it challenging for educators to identify students who might be in need of additional support early on.

To tackle this challenge, our goal is to investigate whether there exist any regularities or patterns in the exam-taking behavior among different students which may provide a reliable estimate of whether a given student is likely to graduate on time or not. Identifying these behaviors serves multiple purposes, such as i) developing improvements in the overall study curricula, ii) providing the students with "successful past examples", and iii) to spot which students are expected to struggle at an early stage of their career, in order to support students and mitigate the risk of graduation delays or dropping out.

This question can be modeled in terms of a classification problem. Namely, we propose an approach leveraging predictive models that can learn the relation between a set of input features modeling different characteristics of students' exam-taking behaviors in the first year and a target variable corresponding to students' graduation performance. Note that we are interested in studying the impact of both students' overall performance (e.g., the average grades of passed exams) and *process-related* features, modeling the order in which exams have been taken by students. The latter enables the development of evidence-based recommendations on *when* students should strive to attempt one or more exams, as well as to identify which (combinations of) exam(s) has(\ have) a critical impact on students' careers. To achieve this goal, first, we convert the sequence of exams

taken by each student during their first year of university (whether passed or not) into an *Instance Graph*, that is a directed, acyclic graph that models a single process execution explicitly representing possible concurrency. We focus on the behavior of students during their first year because we want to intervene to correct any negative trends before it is too late. We then propose and compare two approaches to assess the impact of students' early career on their performance in terms of graduation times. In the first one, we use the Instance Graphs as input to a Graph Convolutional Neural Network (GCNN); in the second one, we extract relevant subprocesses and use them as features in traditional machine learning approaches. The results show that the GCNN performs best, achieving a 93.56% accuracy against the 82.43% achieved by machine learning approaches using subprocesses. Finally, we leverage explainable AI techniques to assess the impact of the mined students' behavior on the graduation times.

This study is an extension of our previous work (Potena et al. 2025), where we proposed a methodology to extract exam-taking patterns modelling students' behaviors and leverage them to predict students' performance. Here, we extend that work along two directions. First, we introduce a novel approach to mine students' behaviors, leveraging process tree structures with the goal of obtaining patterns that are meaningful from a process perspective. Second, we include a whole new set of experiments aimed at comparing the classification performance obtained by i) classifiers leveraging a tabular representation, where process-related features are encoded in terms of occurrence of process patterns, and ii) classifiers leveraging graph neural networks. While the benefits of leveraging a process perspective in analyzing students' learning trajectories have already been acknowledged by previous studies in the literature (see "Related work" section), previous studies mostly leveraged simplified and often aggregated representations of process-related characteristics, by, e.g., modeling the occurrence of a single activity or considering activity sub-sequences. To the best of our knowledge, this is the first study to explore and test different and novel alternatives to model process-related features of students' exam-taking behaviors able to model both sequential and concurrent behaviors, and to assess their impact on the process outcome.

The rest of this manuscript is organized as follows. "Case study: bachelor program of an Italian university" section describes the case study. "Methodology" section illustrates the proposed methodology. "Experiments" section illustrates the obtained results. "Related work" section provides an overview of relevant related work, while "Conclusion and future works" section draws some conclusions and highlights future research directions.

Case study: bachelor program of an Italian university

Our study focuses on a three-year Bachelor's Degree program from an Italian university. The dataset comprises over 700 graduated students spanning 11 academic years (from 2010/2011 to 2020/2021).

Figure 1 reports the percentage and the number of *timely* students, i.e., students who took their degree within the standard duration of the degree program, and *late* students who experienced one or more years of delays in their graduation. As for the students enrolled in 2020, since we have information up to 2023, we only have those who took less than three and a half years, and therefore are considered timely. While Fig. 1 shows

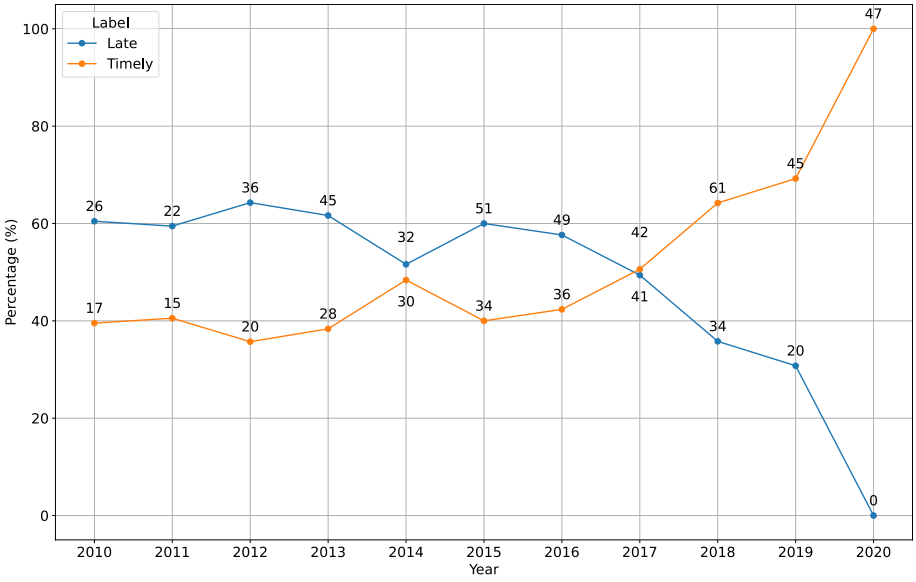


Fig. 1 Percentage of students graduating on time or late, by enrollment academic year. Labels indicate the absolute number of students in the two categories

a consistent improvement in students' performance over time, we observe that at least 30% of students graduated later than expected.

Dataset

We merged the information from two datasets that record the students' academic pathways to create the event log. First, we consider data on passed exams. In the Italian university system, exams are graded on a 30-point scale. An exam is considered passed, and the corresponding credits awarded, if the student attains a grade of 18 or above. Otherwise, the exam is failed and no credits are awarded. In the first dataset, each row corresponds to an exam passed by a student, and the following information are provided: the student's identification code, the name of the exam, the date when the exam was passed, the grade obtained, the exam's academic credits (i.e., weight) of the exam, and whether the exam is mandatory. The second dataset extends the information of the first one by including also failed exams and reasons of the failure, namely student absence (students must register in advance; otherwise, they cannot attempt it), insufficient grade, and withdrawal.

The results of merging the datasets is an event log, where the career of the student is represented as an ordered sequence of exam attempts, i.e., events (van der Aalst 2016). Each event is described by the student identification code, the name of the exam (i.e., the activity label), the credits assigned to the exam, the date of the attempt, and the grade received (set to 0 if failed). Table 1 illustrates an excerpt of the event log. The student identification code and names of the courses have been anonymized and replaced with letters for privacy reasons.

The event log consists of 708 traces, 356 for timely students and 352 for late students. To gain a better understanding of when activities are carried out w.r.t. the academic year, we have inserted two artificial events in the log as a time reference to indicate the end of first semester (*efs*), and the end of first year (*efy*). Finally, artificial *start* was added to each trace.

Table 1 Excerpt of the event log, each event corresponds to an attempt for a specific exam

Case	Event id	Properties				
		Date	Status	Exam	Credits	Grade
C_0	E_0	2025-01-20	Passed	A	9	28
	E_1	2025-02-17	Failed	B	9	0
	E_2	2025-02-18	Failed	C	6	0
	E_3	2025-03-20	Passed	B	9	23
	E_4	2025-03-25	Failed	C	6	0
C_1	E_5	2025-01-20	Failed	A	9	0
	E_6	2025-02-17	Passed	B	9	30
	E_7	2025-02-22	Passed	A	9	27
	E_8	2025-03-25	Failed	C	6	0
	E_9	2025-05-12	Passed	C	6	29
...	...	...	...	...	...	...

Methodology

Our study aims to assess the impact of students' first-year exam-taking behavior on their likelihood of graduating on time. We model this question as a classification problem, meaning that we intend to leverage predictive models to learn the relationship between features modeling the exam-taking behaviors and the graduation performance. This goal is split into two sub-goals. First, we need to determine *which features to select to represent the exam-taking behavior (G1).* Then, we need to determine *how to quantify the impact of each feature on students' graduation performance (G2).*

Concerning G1, we consider both aggregated performance features modeling the overall students' progression in the first year, and process-related features characterizing the academic pathways. As performance features, we selected a set of features in collaboration with domain experts ("Feature extraction" section). To model process-related features, we leverage a *graph* representation, as it is a natural way to model process relations. We first convert each trace in the log into an *Instance Graph (IG)* ("Instance graph building" section), i.e., a directed acyclic graph where each node corresponds to an event in the trace and edges model dependency relations among the corresponding process activities. These graphs explicitly model both sequential and concurrent behaviors.

Then, we need to decide how to *encode* these graphs to apply predictive models on them. We explored two strategies. The first one leverages a *tabular* representation, where we characterize traces on the basis of the presence of patterns according to two methodologies for deriving relevant substructures from IGs ("Subgraphs extraction" section). This strategy allows us to leverage standard classifiers using tabular data. The second strategy leverages the whole instance graphs preserving their graph structure, by giving them as input to graph neural networks. Note that in both cases, each trace is enriched using the overall performance features mentioned above. Based on the control-flow (i.e., instance graphs and extracted patterns) and the defined features, we build classifiers to predict student performance ("Classifiers discovery" section).

Finally, to address G2, we leverage explainable AI techniques to assess the impact of the exam-taking process on students' performance ("Quantifying feature contribution" section). Each step is detailed in the following sections.

Instance graph building

For the rest of this manuscript, for the sake of simplicity, we refer to an event using the corresponding activity, i.e., the name of the exam, optionally followed by the label *NP* to specify when the exam is failed. To exemplify our procedure for constructing an IG from a trace in the event log, we use the trace $\sigma = \langle start, A, B, efs, C\text{-}NP, C, D, E\text{-}NP, efy \rangle$. Events *A*, *B*, *C* and *D* correspond to different passed exams, while events *C-NP* and *E-NP* are failed exams. Events *C-NP* and *C* correspond to two attempts of the exam *C*, indicating that the exam is failed on the first attempt. We added an artificial *start* in the trace. Furthermore, *efs* and *efy* are the end of the first semester and the end of the first year, respectively. We apply an ad-hoc procedure to build the IGs to reflect the concept of concurrency in our case study. The reason is that state-of-the-art approaches for deriving a set of IGs from an event log (Diamantini et al. 2016) infer concurrency relations among process activities from the ordering relations observed in the event log. However, this notion of concurrency is not well-suited to analyze students' exam-taking behaviors for two reasons. First, the specific order in which exams are taken depends not only on the students' choices but also on the exam schedule, which may vary in different years for exams in the same semester, and may potentially introduce incorrect dependencies among the exams. Furthermore, this notion does not take into account the *time* in between consecutive exam attempts. From discussions with stakeholders from the university, we derived the assumption that two different exams for which an attempt is made within one month can be considered *concurrent*, intended as exams that the student is preparing simultaneously. This notion of temporal concurrency is not supported by traditional IG building approaches.

Figure 2 shows the temporal span of events in σ, where t_i is the date when the exam *i* is attempted. Each exam *i* is associated with the one-month interval ending at t_i, representing the study period for the exam in accordance with the assumption previously introduced. Filled areas denote temporal overlaps identifying concurrent exams. Note that events corresponding to artificial activities have instantaneous study periods. It can be observed from the figure that exams *A* and *B* occur concurrently, as do exams *C-NP*, *C*, and *D*. The concurrence of *C-NP* and *C* is a direct result of the fact that the exam dates are scheduled less than a month apart. Since a one-month overlap is an average value, we consider exams to be concurrent if their periods overlap by at least an epsilon. In this work, we set $\epsilon = 1$ day.

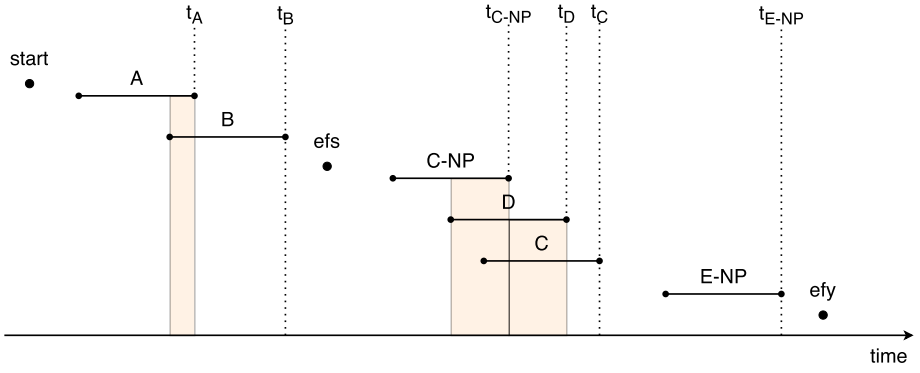


Fig. 2 Temporal span of events in σ. t_i represents the attempt date of the exam *i*. Horizontal lines represent the one-month study period ending in t_i. Filled areas are temporal overlaps

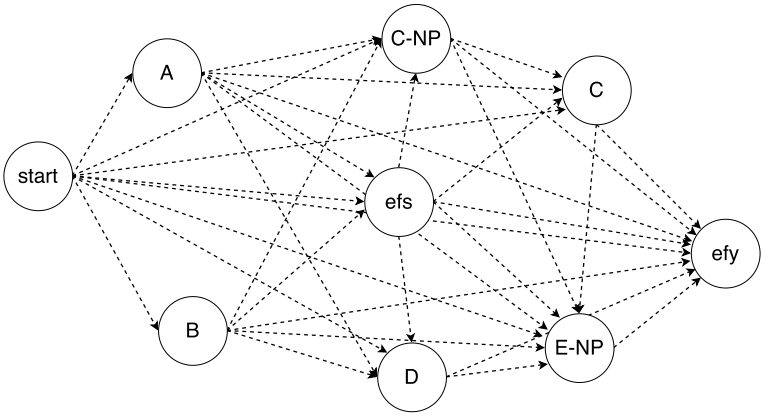


Fig. 3 Edges generated by the algorithm for building the instance graph

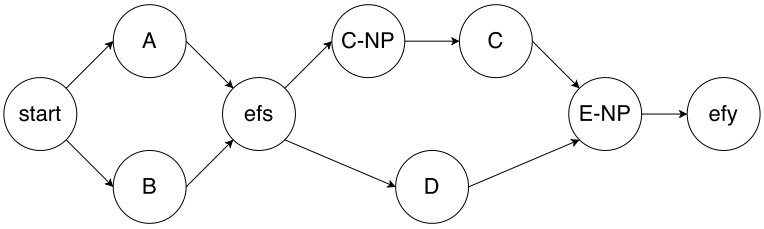


Fig. 4 The resulting instance graph

We first created one node for each event. To connect the nodes, we iterated through each pair of nodes and created an edge connecting them as follows:

a) If the pair contains at least one artificial activity, the pair is connected based on chronological order. In Fig. 3, *start*, *efs*, and *efy* are connected to every node;
b) If the pair does not contain artificial activities but the nodes are concurrent, no edges are added unless the pair represents two attempts for the same exam. In this case, the start time of the second attempt is set to the end time of the first attempt plus one day. For example, in Fig. 3, even though *C-NP* and *C* were concurrent, they are not considered as such;
c) If the pair does not contain artificial activities and the nodes are not concurrent, edges are created based on chronological order.

Once the graph is created, we applied a transitive reduction to remove superfluous edges while maintaining reachability. For example, since an edge connecting *start* to *A*, *A* to *C-NP* and *start* to *C-NP* exist, the last one is removed. The resulting IG is shown in Fig. 4.

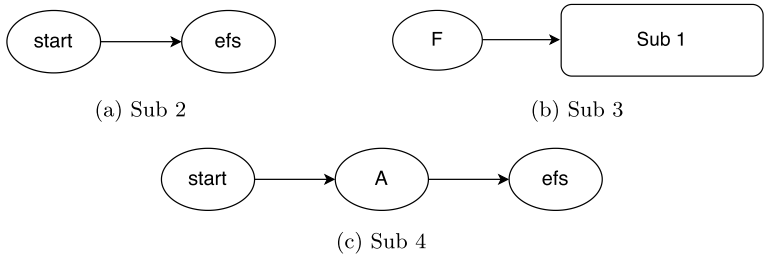
Subgraphs extraction

We apply two mining techniques to extract the most relevant substructures from the IG set based on node labels, which, in this context, correspond to the most relevant process patterns. The former technique leverages the Subdue (Jonyer et al. 2002) algorithm ("Subdue" section), while the latter ("Process pattern mining using process trees" section) is a pattern mining algorithm based on the concept of process tree.

Subdue

The first approach uses Subdue, a frequent subgraphs mining algorithm (Jonyer et al. 2002) to identify the most relevant substructures within a set of input graphs. The relevance of a substructure is measured by its ability to compress the input dataset using the substructure itself, in accordance with the Minimum Description Length (MDL) principle. At each iteration, the algorithm selects the most relevant subgraph, replaces all of its occurrences in the input graphs with a single node, and uses the resulting compressed graphs as input for the next iteration. The process continues until a given number of iterations is reached, or no further compression gain can be achieved. In this way, in early iterations the algorithm identifies substructures composed only by original nodes and edges, whereas substructures identified in later iterations may also contain as nodes substructures identified in previous iterations. The model resulting from Subdue takes the form of a lattice of the identified substructures, linked by inclusion (i.e., part-of) relationships. Substructures that are not contained in others constitute the top level of the hierarchy. As one descends the hierarchy, the subgraphs shift from highly frequent patterns across multiple input graphs (i.e., with high support) to more specific patterns occurring in only a few input graphs (i.e., with low support). Subdue generated 654 subgraphs from the set of IGs. For our analysis, to reduce dimensionality, we selected the top 100 subgraphs that capture the exam-taking patterns of interest. Each trace in the log is, therefore, characterized by a vector of 100 features, each corresponding to one of the extracted subgraphs. For each feature, a value of 1 is assigned if the corresponding subgraph is present, and 0 otherwise.

Figure 5 displays some of the first top patterns mined by Subdue for our dataset. It is straight to detect one clearly undesirable behavior, i.e., students who did not take any exam in the first semester (Sub 2, Fig. 5a). Sub 3 (Fig. 5b) shows that a common behavior is taking *F* and then performing the subprocess identified by Sub 1, i.e., taking G and then ending the first year. Finally, Sub 4 (Fig. 5c) shows that another common behaviors for these students was to pass *A* before the end of first semester. While these subs show the capability of the method to recognize characteristic students' behaviors, they also highlight its limits. Indeed, although Subdue has already been successfully used for pattern extraction from event logs (Diamantini et al. 2016), the algorithm is designed to work with generic graphs and is not specifically tailored to recognize subprocesses. As a result, the extracted structures may turn out to be less meaningful in the given domain. To address this limitation, we introduce an alternative pattern mining approach leveraging common process operators, which is explained in the following section.



(a) Sub 2 (b) Sub 3

(c) Sub 4

Fig. 5 The most relevant patterns mined by Subdue for our dataset

Process pattern mining using process trees

In this section, we introduce a process pattern mining approach based on the concept of *process tree* (Leemans et al. 2013). The idea is to extract, for each IG, its corresponding process tree and then use the subtrees as characteristic patterns of the traces. To this end, our approach leverages the function convert_to_process_tree of the Pm4Py library[2], which requires a Petri net as input. Hence, each IG is first converted into a Petri net, and then the process tree is derived. Finally, relevant patterns are derived as frequent subtrees.

To generate a Petri net from an IG, we first create a transition for each node of the IG. Then, for each edge in the graph connecting nodes n_1 and n_2, we create a place p and flows from n_1 to p and from p to n_2.

Furthermore, if two sets of transitions exist such that every transition in the first set is connected to every transition in the second set, an invisible transition is inserted to connect the two sets. This simplifies the Petri net, avoiding potential issues during its conversion to a process tree. Finally, starting and ending places are added, representing the initial and final marking of the net. The starting place is connected to the transition labeled *start* and a flow is created from the transition labeled *efy* to the ending place. Figure 6 shows the Petri net derived from the IG shown in Fig. 4.

The next step aims at deriving the set of process trees from the set of Petri nets. Figure 7 shows the process tree derived from the Petri net in Fig. 6. It should be noted that a Petri net can be converted into a process tree only if the net is block-structured. In the example depicted in Fig. 6, the logical blocks constituting the net are clearly evident. The Petri net comprises six sequential blocks: four of these blocks consist of a single transition (i.e., *start*, *efs*, *E–NP*, and *efy*), while one represents the parallelism between *A* and *B*. The final block is more complex, as it represents the parallelism between *D* and the block corresponding to the sequence of *C–NP* and *C*. In Fig. 7 the precedence relationships within a sequence are graphically represented from top to bottom to simplify the notation.

In our dataset, there are 42 cases in which the IG cannot be converted into a block-structured Petri net, and consequently, it is not possible to derive the process tree. This limitation primarily arises in scenarios where exam dates are scheduled in close temporal proximity, potentially resulting in sets of concurrent exams with non-empty intersections. Figure 8 shows a fragment of a non-block-structured Petri net. These 42 cases will be excluded from the experiments ("Classifiers discovery" section).

After converting the IGs into process trees, we extract all subtrees while excluding the leaves, as they consist of a single transition. These subtrees are used to represent each trace in a new feature space whose dimensionality equals the total number of subtrees extracted from all process trees, with each feature encoding the presence of a given subtree in the IG under consideration. It should be noted, however, that the process tree built on a specific IG could lead to the identification of subtrees that are included in

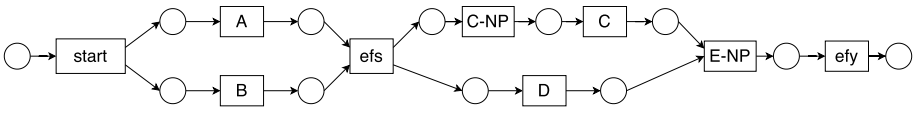


Fig. 6 The Petri net derived from the IG in Fig. 4

[2] https://processintelligence.solutions/pm4py

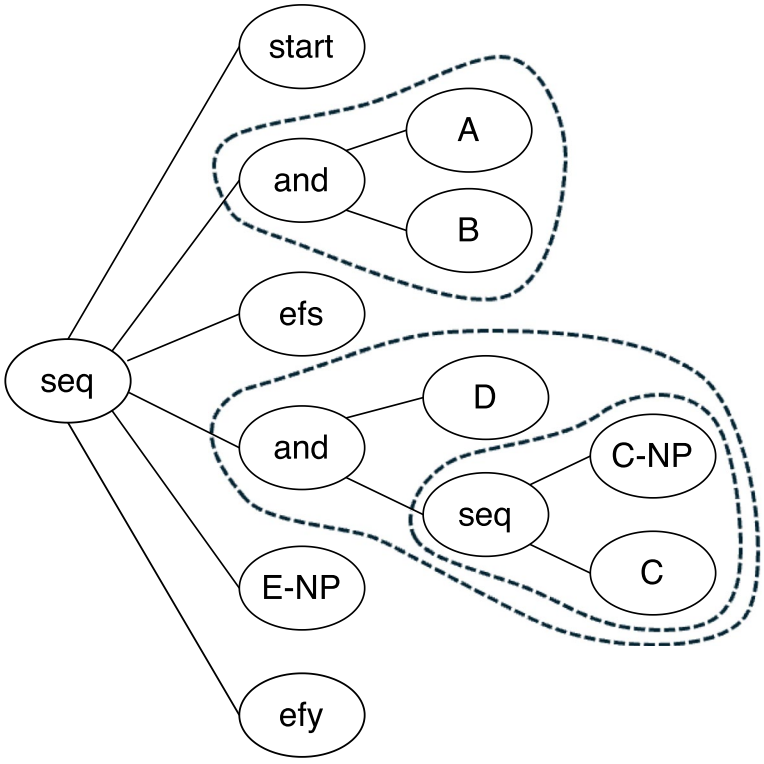


Fig. 7 The process tree derived from the Petri net in Fig. 6. The precedence relationships are graphically represented from top to bottom to simplify the notation

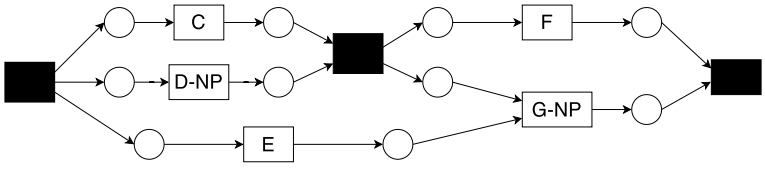


Fig. 8 A fragment of Petri net that cannot be converted into a process tree

those extracted from another IG, and this needs to be properly handled to avoid incorrect representation of traces in the new space. In fact, if subtree S_1 (e.g., <and, (A, B)>) is included within subtree S_2 (e.g., <and, (A, B, C)>), S_1 has to be present in every IG where S_2 occurs, even if S_1 is not explicitly represented in the specific IG's process tree. Consequently, each trace in the log is encoded as a feature vector of length 295, corresponding to the total number of identified subtrees. Each feature is assigned a value of 1 if the corresponding subtree is present in the trace, and 0 otherwise. Figure 9 shows the frequency with which a subtree occurs in the set of IGs (for reasons of space, only the 80 most frequent subtrees are shown). In order to reduce dimensionality, we filtered out subtrees that occurred in less than 2% of the original dataset, resulting in the selection of 62 subtrees.

Feature extraction

To incorporate additional characteristics beyond the control-flow captured by the extracted subgraphs, we also introduced a set of 12 features to characterize each case as a whole. These features were defined in collaboration with domain experts (degree

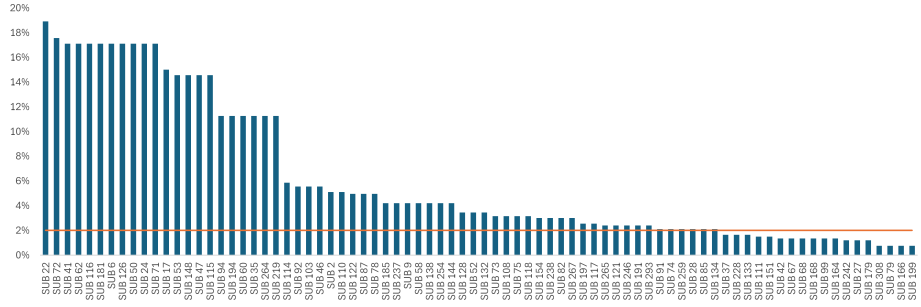


Fig. 9 Frequency distribution of the 80 most common subtrees

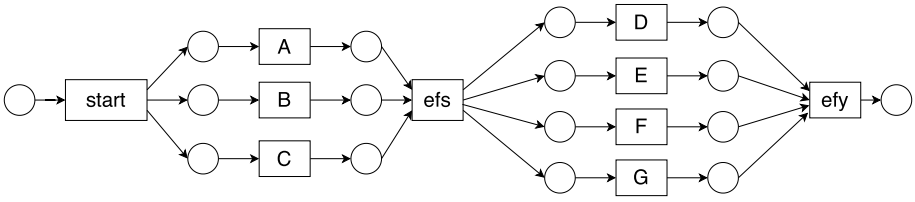


Fig. 10 Study manifesto

program coordinators and quality managers) and are: High school grade, Average and Standard deviation of the grades obtained in the first year, Credits earned in the first year (overall and separately in the first and second semesters), Number of exams passed in the first and second semesters, Number of exams failed in the first year, Average and Standard deviation of time intervals between the dates of two consecutive exams. Furthermore, we added a feature representing the Replayed fitness score, calculated for each trace (considering only passed exams) on the process model that represents the standard path defined by the study manifesto, with the addition of artificial activities indicating the *start*, the end of the first semester (*efs*) and the end of first year (*efy*) (Fig. 10). The fitness is introduced to assess how much the student's behavior deviates from the standard path defined at design-time, i.e., the manifesto.

All features were normalized such that they assume positive values not exceeding 1.

Classifiers discovery

On the basis of patterns and features extracted, we defined several datasets, each representing a different encoding of the original event log. In order to ensure a fair comparison among the classification models to be induced, we excluded from the log all traces whose IG is not convertible into a process tree. Consequently, experiments have been conducted on an event log comprising 666 traces, with 327 belonging to the class of timely students and 339 to the late class. In detail, we generated 9 datasets, as follows:

- SD: Each trace is represented by the presence of 100 patterns extracted using Subdue.
- PT: The 62 patterns extracted using the process tree are used to characterize the event log.
- F: The event log is represented in terms of the 12 features defined in "Feature extraction" section.
- $\mathbf{SD_F}$: The dataset is obtained by merging the *SD* and *F* datasets.
- $\mathbf{PT_F}$: Both the patterns extracted from the process tree and the 12 generated features are used to describe each trace.

- IG: The dataset is composed solely of the Instance Graphs, without any additional processing.
- $\mathbf{IG}^{+}$: Each node of the Instance Graph is further characterized with the credits assigned to the exam, the achieved grade (if passed, 0 otherwise), and two temporal features. The former is the interval between the exam date and the date of the previous exam in the trace. The latter is the interval between the start of the student's academic path and the exam date.
- $\mathbf{IG_F}$: Each Instance Graph is additionally characterized by the 12 features.
- $\mathbf{IG_F^+}$: All features are integrated into the IG dataset.

Note that SD, PT, and IG are three alternative approaches for representing only the control flow. The first five datasets, which have a classical tabular structure, were used to build a predictive model employing the following classification algorithms: Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), and eXtreme Gradient Boosting (XGB). For all datasets that directly contain instance graphs, we used a variant of the Deep Graph Convolutional Neural Network (DGCNN) proposed by Zhang et al. (2018), which is capable to directly process graph data. The DGCNN falls into the category of spatial-based Graph Convolutional Networks (Wu et al. 2021), in which convolution operation is interpreted as message-exchanging between nodes and their neighbors. The network is built by stacking several GraphSAGE (Hamilton et al. 2017) convolutional layers to extract high-level node features.

Let us suppose that $G = (V, E, R)$ is a graph. V represents the set of nodes, where each node is described by a feature vector of size c, i.e., the one-hot encoding of the activities. The feature vector may describe additional perspectives, e.g., the IG^{+} dataset further describes, for each node, the credits, the achieved grade, and the two temporal perspectives. E represents the set of edges, while R the set of *global attributes*, i.e., the features modeled in the F dataset.

To perform node convolutions, the graph G is described by its adjacency matrix $A \in \mathbb{R}^{|V|\times|V|}$, which provides information about the graph topology, and the feature matrix $X \in \mathbb{R}^{|V|\times c}$, which contains attributes specific to the graph domain. Each row $x_i \in \mathbb{R}^{1\times c}$ of X contains the properties of the $i-th$ node of G. The graph convolution equation adopted by GraphSAGE assumes the form of:

$$h_u^{k+1} = \sigma(W^k \cdot AVG(h_u^k, \{h_v^k \mid v \in N^*(u)\}) \tag{1}$$

where h_u^k represents the embedding of node u at the $k-th$ step of convolution, σ is the Rectified Linear Unit (ReLU) activation function ($\sigma(x) = 0, if\ x < 0, else\ x$), W^k is a matrix with learnable parameters, and $N^*(u)$ represents a random sample of node u neighbors. GraphSAGE assumes that for every node the initial embedding is equal to the input feature vector ($h_u^0 = x_u, \forall u \in V$). Equation 1 represents graph convolution in terms of node, and can be rewritten in terms of graph, which becomes:

$$Z^{k+1} = \sigma(\tilde{D}^{-1}\tilde{A}Z^kW^k) \tag{2}$$

where $Z^0 = X$, Z^k is the output of the $k-th$ convolution layer, $\tilde{A} = A + I$ the adjacency matrix with self-loops, and $\tilde{D}$ is its diagonal degree matrix with $\tilde{D}_{ii} = \sum_j A_{ij}$.

After the convolution stage, the outputs Z^k are then concatenated ($Z = [Z^1, ..., Z^k]$) and fed to SortPooling layer. The SortPooling selects, according to a hyper-parameter,

the most representative nodes, obtaining a graph with a fixed number of nodes. Then, a 1D-convolution layer is adopted to flatten the graph before feeding it to a Dense Layer. Since global attributes are not processed during the convolution stage, the Dense layer takes as input both the 1D-convolution output and the set R. This is done to ensure that graph attributes can contribute to the prediction. Finally, a Softmax activation function is applied to perform classification. Figure 11 illustrates the proposed network architecture.

Quantifying feature contribution

The final step is aimed at assessing the impact of each feature on the prediction outcome. We consider both classic feature importance techniques, which are model-dependent and assign a score to input features to a predictive model indicating the relative importance of each feature when making a prediction, and SHapley Additive exPlanations (SHAP) (Lundberg and Lee 2017), a widely used model-agnostic technique to explain the contribution of each feature to a model's prediction. For classifiers leveraging tabular data and incorporating the extracted subprocesses as features, these techniques can be directly applied to the models. However, this is not possible for graph neural networks, as whole Instance Graphs are used as input. Therefore, to interpret the impact of relevant control-flow subprocesses in the DGCNN results, we adopt the approach of inducing global surrogate models. Specifically, we construct an intrinsically interpretable model (in our experiment, we employ a decision tree) to emulate the behavior of the model trained with the DGCNN. To this end, the surrogate model is trained on the same traces used for the DGCNN, with the original labels replaced by the predictions produced by the DGCNN model. Furthermore, to analyze the impact of subprocesses, we leverage a trace encoding based on subprocess-level information, i.e., SD,PT, $\mathrm{SD_F}$, or $\mathrm{PT_F}$. Finally, we assess the feature importance and the SHAP values for the surrogate models. Note that in our experiments we only carried out this analysis for the best-performing classifier.

Experiments

In this section, we present the results obtained by applying the methodology described in the previous section to the case study. The experiments aim to answer two main research questions: RQ1) which configuration of the encoding strategy and classifier leads to the best classification performance, and RQ2) How do the features derived from RQ1 impact students' likelihood of graduating on time? To answer RQ1, we carried out a thorough comparison leveraging state-of-the art classifiers and considering

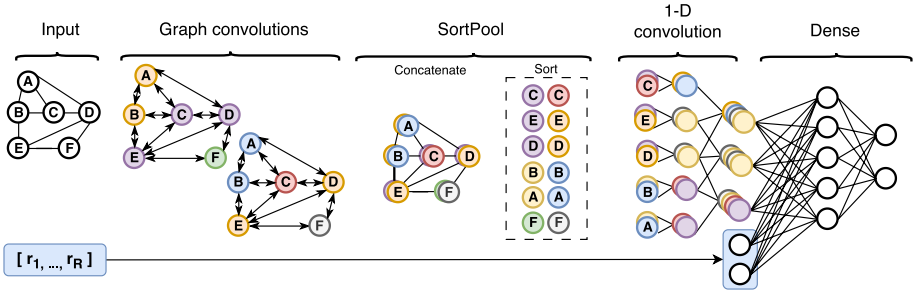


Fig. 11 DGCNN architecture

both a tabular and a graph-based representation of process-related features, with and without aggregated features related to the overall academic performance. For RQ2, we leverage classic feature importance techniques and SHAP values on the best-performing classifiers.

The following subsections are organized as follows. First, we provide details on the adopted experimental setup ("Experimental settings" section), and then we compare the results obtained using the different encodings of students' first-year academic careers ("Results: tabular datasets", "Results: graph-based datasets" sections). Finally, we discuss the contribution of the features we defined in predicting student performance ("Feature contribution" section).

Experimental settings

As stated in the previous section, we defined two types of datasets: tabular and graph-based. For the former type, i.e., SD, PT, F, $\mathrm{SD_F}$, $\mathrm{PT_F}$, we use Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), and eXtreme Gradient Boosting (XGB) algorithms to induce classification models. For graph-based datasets (i.e., IG, $\mathrm{IG^+}$, $\mathrm{IG_F}$ and $\mathrm{IG_F^+}$) we trained the DGCNN.

The parameters combined for each algorithm through grid search are shown in Table 2. For each experiment, we performed a 5-fold cross-validation and report below the best average results obtained for each algorithm and dataset.

Results: tabular datasets

Table 3 shows the best average accuracy achieved by SVM, DT, RF, and XGB on SD, PT, F, $\mathrm{SD_F}$ and $\mathrm{PT_F}$ datasets. We observe that no algorithm clearly outperforms the others on any of the datasets, while it can be seen that the Decision Tree is consistently the worst. On the other hand, we can note differences in the best performance achieved through datasets, showing the impact of the different encodings proposed. In particular, comparing the results obtained on datasets generated with the two methods for the extraction of relevant subprocesses, it is evident that using Subdue consistently yields higher accuracies than those obtained on datasets generated via process tree. Indeed, the best result for SD is approximately 6% higher than that obtained for PT, and for $\mathrm{SD_F}$, an average accuracy about 2% greater is achieved than that reached on $\mathrm{PT_F}$. This phenomenon might be explained by the fact that when using process trees, we search for meaningful structures for the process domain (i.e., subprocesses), which inherently limits the range of identifiable structures. Conversely, when using Subdue, the search aims

Table 2 Algorithms and their corresponding parameters

Model	Parameters
SVM	C:{0.1, 1, 10, 100}, gamma:{scale, auto}, kernel:{linear, rbf, poly}
DT	criterion:{gini, entropy}, splitter:{best, random}, max depth:{None, 10, 20, 30, 40, 50}, min samples split:{2, 5, 10}, min samples leaf:{1, 2, 5, 10}
RF	number of trees:{50, 100, 200}, criterion:{gini, entropy}, max depth:{None, 10, 20, 30}, min samples split:{2, 5, 10}, min samples leaf:{1, 2, 5, 10}, bootstrap:{True, False}
XGB	colsample bytree:{0.6, 0.8, 1}, learning rate:{10^{-3}, 10^{-2}, $2{\cdot}10^{-2}$}, max depth:{3, 6, 10}, number of estimators:{50, 100, 200}, subsample:{0.6, 0.8, 1}
DGCNN	k:{5, 7, 30}, learning rate:{10^{-3}, 10^{-4}, 10^{-5}}, dropout:{0.1, 0.2}, number of neurons:{32, 64}, number of graph convolution layer:{1, 2, 3, 5, 7}, batch size:{16, 32}, epochs:{200}

Table 3 Best average accuracy (with standard deviations in brackets) achieved for each algorithm and tabular dataset. Parameter configurations are also reported

Dataset	Algorithm	Parameters	Average Accuracy
PT	XGB	estimators: 50, max depth: 3, learning rate: $2 \cdot 10^{-2}$, subsample: 0.8, col sample by tree: 1.0	**70.56% (±3.13%)**
	SVM	C: 10, kernel: rbf, gamma: auto	70.26% (±3.44%)
	DT	criterion: gini, split: best, max depth: 10, min samples split: 10, min samples leaf: 2	69.51% (± 2.30%)
	RF	criterion: entropy, estimators: 50, bootstrap: False, max depth: 10, min samples split: 10, min samples leaf: 2	70.41% (± 2.31%)
SD	XGB	estimators: 50, max depth: 3, learning rate: $2 \cdot 10^{-2}$, subsample: 0.8, col sample by tree: 1.0	76.72% (± 0.92%)
	SVM	C: 100, kernel: rbf, gamma: auto	**76.87% (± 1.31%)**
	DT	criterion: gini, split: best, max depth: 10, min samples split: 10, min samples leaf: 2	73.87% (± 1.49%)
	RF	criterion: gini, estimators: 50, bootstrap: False, max depth: 10, min samples split: 10, min samples leaf: 5	76.27% (± 1.68%)
F	XGB	estimators: 200, max depth: 3, learning rate: 10^{-3}, subsample: 0.8, col sample by tree: 0.8	**80.03% (± 0.96%)**
	SVM	C: 10, kernel: linear, gamma: auto	79.43% (± 2.45%)
	DT	criterion: gini, split: best, max depth: 10, min samples split: 10, min samples leaf: 10	77.32% (± 2.98%)
	RF	criterion: gini, estimators: 50, bootstrap: False, max depth: 10, min samples split: 10, min samples leaf: 10	79.87% (± 2.21%)
PT_F	XGB	estimators: 50, max depth: 6, learning rate: 10^{-1}, subsample: 0.8, col sample by tree: 1.0	80.33% (± 2.58%)
	SVM	C: 100, kernel: linear, gamma: auto	**80.48% (± 1.76%)**
	DT	criterion: gini, split: best, max depth: 10, min samples split: 10, min samples leaf: 10	78.07% (± 3.49%)
	RF	criterion: entropy, estimators: 50, bootstrap: False, max depth: 10, min samples split: 5, min samples leaf: 1	80.02% (± 2.49%)
SD_F	XGB	estimators: 200, max depth: 3, learning rate: 10^{-2}, subsample: 0.8, col sample by tree: 0.8	81.52% (± 2.76%)
	SVM	C: 1, kernel: linear, gamma: auto	80.48% (± 2.39%)
	DT	criterion: gini, split: best, max depth: 10, min samples split: 10, min samples leaf: 10	77.77% (± 3.54%)
	RF	criterion: entropy, estimators: 50, bootstrap: False, max depth: 10, min samples split: 5, min samples leaf: 2	**82.43% (± 1.33%)***

to identify any relevant subgraph (with high support[3]) within the set of instance graphs, without requiring that the subgraphs correspond to meaningful subprocesses. This clearly helps in identifying features that better discriminate between the two classes.

Results obtained on the F dataset consistently outperform those on SD and PT, achieving an average accuracy of 80.03%, which is 3.16% higher than that obtained on SD and 9.47% higher than that on PT. This suggests that the control flow information contained in the identified substructures alone is not sufficient to achieve good accuracy. In fact, the F dataset includes not only features derived from the control flow (i.e., number of passed exams and first year failures) but also features extracted from exam attributes (e.g., time, grades, credits), student attributes (e.g., high school grade), and the alignment of the student's academic progress with the study manifesto.

[3] More precisely, the subgraphs identified by Subdue are those that best compress the input graph based on the MDL criterion.

Moving to $\mathrm{SD_F}$ and $\mathrm{PT_F}$ datasets, which are designed using both the features related to the identified substructures and the 12 features in F, we observe that performances increase with respect to $\mathrm{SD_F}$, $\mathrm{PT_F}$ and F. For $\mathrm{SD_F}$ the best classifier is obtained by using RF (82.43%), while for $\mathrm{PT_F}$ SVM returns the best accuracy (80.48%).

It is interesting to note that adding patterns to the 12 features does not provide a significant net contribution, as expected. The gap between the best performance on $\mathrm{PT_F}$ and F is only 0.3%, while the difference between results achieved on $\mathrm{SD_F}$ and F is 2.4%. Furthermore, the gap between the best performance obtained on $\mathrm{PT_F}$ and $\mathrm{SD_F}$ datasets (1.95%) is considerably smaller compared to PT and SD, where the difference is 6.31%.

This suggests, as previously observed, that the contribution of features related to the presence of control-flow substructures is marginal compared to the broader range of features contained in the F dataset. Moreover, it also suggests a possible issue related to the curse of dimensionality ($\mathrm{SD_F}$ has 112 features and $\mathrm{PT_F}$ 73), indicating that a larger number of traces would be required to achieve good performance with these feature-rich datasets.

Results: graph-based datasets

Table 4 shows the best average results achieved by DGCNN on IG, $\mathrm{IG^+}$, $\mathrm{IG_F}$ and $\mathrm{IG_F^+}$ datasets.

A good level of accuracy is achieved on the IG dataset, exceeding the best results obtained on SD and PT by more than 10%. This suggests, as expected, that using the entire graph to describe the first year of student' careers provides a greater informational contribution than representing the career through relevant subprocesses. Clearly, the result obtained may also depend on the adopted algorithm, which is based on a logic very different from that of the algorithms used in the previous subsection. However, it is worth noting that deep learning algorithms typically perform better on datasets with high cardinality, whereas in our study we have only 666 traces.

Enriching the control flow with exam-related features ($\mathrm{IG^+}$ dataset) results in an improvement, albeit limited, of 1.65%. A better result is obtained when, in addition to control flow, we also consider the 12 synthetic features described in "Feature extraction" section. In fact, the best result achieved on $\mathrm{IG_F}$ is 3.46% higher than that on $\mathrm{IG^+}$ and 5.11% higher than that on IG.

Finally, by combining all features and the control flow (i.e., $\mathrm{IG_F^+}$ dataset), we obtain the model with the highest performance, achieving an average accuracy of 93.56% on the

Table 4 Best average accuracy (with standard deviations in brackets) achieved by DGCNN on graph-based datasets. Parameter configurations are also reported

Dataset	Parameters	Average Accuracy
IG	learning rate: 10^{-5}, batch size: 32, dropout: 0.1, num layers: 1, num neurons: 64, k: 7	87.25% (± 7.36%)
$\mathrm{IG^+}$	learning rate: 10^{-4}, batch size: 32, dropout: 0.1, num layers: 3, num neurons: 64, k: 30	88.90% (± 5.97%)
$\mathrm{IG^+_{F-}}$	learning rate: 10^{-4}, batch size: 32, dropout: 0.1, num layers: 2, num neurons: 32, k: 30	90.86% (± 7.63%)
$\mathrm{IG_F}$	learning rate: 10^{-4}, batch size: 32, dropout: 0.1, num layers: 5, num neurons: 32, k: 30	92.36% (± 6.64%)
$\mathrm{IG_F^+}$	learning rate: 10^{-4}, batch size: 32, dropout: 0.1, num layers: 2, num neurons: 32, k: 7	**93.56% (± 6.17%)**

test set. Also in this case, incorporating exam-related features only marginally improves the result achieved by DGCNN, with a gap of only 1.2% compared to the result achieved on $\mathrm{IG_F}$. This could be due to the increased complexity of the network; indeed, encoding the exam-related features requires adding 4 neurons to the DGCNN's input layer for each node in the graph.

From the experiments, it can be concluded that control flow alone provides a very significant contribution to discriminate between the two classes of students (ranging from 70.56% achieved on PT to 87.25% obtained on IG), but additional features are necessary to enhance classification. Moreover, process knowledge of the entire first year appears to be even more important than understanding the subprocesses that characterize a student's career. In fact, even using only control flow (i.e., IG) results in an improvement of approximately 5% over the best model presented in Table 3.

As a drawback, the models obtained by DGCNN are not directly interpretable. Therefore, in the next subsection, we will leverage subprocesses to understand which student behaviors influence the predictions and how.

Feature contribution

In order to evaluate how the relevant control-flow subprocesses influence the prediction made by the best model obtained (i.e., that on $\mathrm{IG_F^+}$ by DGCNN), we encoded the traces as subprocesses instead of IGs and generated surrogate decision-tree models that are more interpretable. Specifically, we use the same encoding as in $\mathrm{PT_F}$. We chose patterns obtained via process trees rather than those obtained via Subdue because the representation of meaningful structures in the process domain provided by process trees is more beneficial for the interpretation task.

In details, we proceeded as follows: (i) For each trace, we used its corresponding IG to obtain the class predicted by the model obtained by DGCNN on $\mathrm{IG_F^+}$, and then replaced the true class with that prediction. Next, for each IG, we replaced it with the encoding of the related trace as used in $\mathrm{PT_F}$. In this way, we obtain a dataset that mimics the model's behavior in terms of inputs and predicted class. Then, we trained two surrogate decision trees, the first using both patterns and 12 features, the second using only subprocesses. The parameter combinations were selected performing a hyper-parameter tuning (the same of Table 2) over the complete datasets, maximizing accuracy. For both classifiers, we report and discuss feature importance in terms of both how much each feature contributes to reducing impurity at each split of the induced tree, and SHAP values.

Figure 12 presents the top 30 features with the highest impact on predictions for the first surrogate classifier, when both subprocesses and extracted features are considered. The most important feature overall (with a score of approximately 0.4) is the credits earned in the first year (CREDS_Y1). The following six features have slightly lower scores, around 0.1. These features are the IG compliance with respect to the study manifesto (REPLAYED_FITNESS), the average (MEAN_GRADES) and the standard deviation (CONSISTENCY_STD_GRADES) of the grades obtained in the first year, the high school grade, the average (MEAN_INTERVAL) and standard deviation (CONSISTENCY_STD) of the time intervals between the dates of two consecutive exams. It is worth noting that among the most important features is the replayed fitness, which indicates how viewing a student's career as a process is useful for predicting student performance. The most important subprocess is only in 9th position, namely

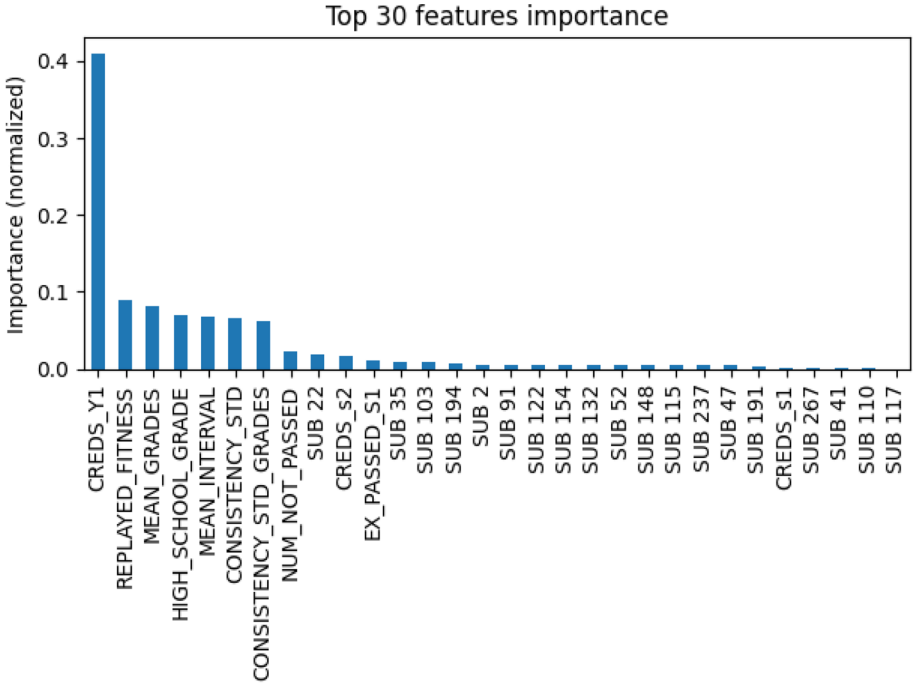


Fig. 12 Feature importance for surrogate model. Subprocesses and extracted features

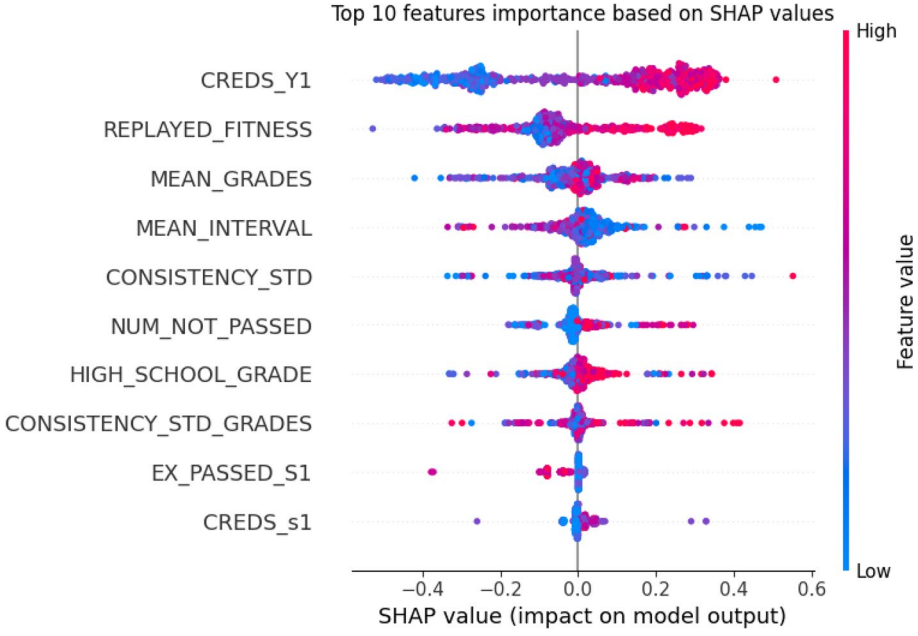


Fig. 13 SHAP values for timely class on surrogate model. Subprocesses and extracted features

Sub 22 (Fig. 16a). This poor contribution highlights the limited impact of patterns on the prediction. Figure 13 reports the contribution of the top 10 features in recognizing the timely class, as determined using SHAP. The figure shows how high values of (i) credits earned in the first year (CREDS_Y1), (ii) the IG compliance with respect to the study manifesto (REPLAYED_FITNESS), and (iii) the standard deviation (CONSISTENCY_STD_GRADES) of the grades obtained in the first year contribute positively to

predict the timely class. Another interesting aspect regards the average (MEAN_INTERVAL) and standard deviation (CONSISTENCY_STD) of the time intervals between the dates of two consecutive exams. Low values of these features contribute positively for timely class, suggesting that students who (i) take exams within the same session and (ii) achieve relatively consistent grades are more likely to graduate on time.

To assess the contribution of subprocesses alone, we examine the feature importance (Fig. 14) and SHAP values of the timely class (Fig. 15) for the second surrogate model. The top ten most important subprocesses are shown in Fig. 16.

The SHAP values indicate that, except for Sub 22, the absence of patterns in a student's career (shown in blue in Fig. 15) provides a limited contribution to predict the late class. Conversely, the presence of these subprocesses significantly contributes to predict the timely class.

The fact that Sub 22 contributes negatively to the prediction of the timely class is easily explained, as this pattern describes a student who neither passed nor even attempted any exam in the first year. According to the manifesto (Fig. 10), the patterns that most influence the prediction (Subs 62, 72, 110, and 132) refer to exams observed during the first semester of the first year. Although the presence of such patterns pushes the prediction toward the timely class, we note that no pattern represents the case in which all exams are successfully passed simultaneously. A common characteristic among these patterns is exam A. Merely attempting exam A, which presumably indicates that the student prepared for it, seems to be a boost to graduating on time. Indeed, Sub 72 suggests that A is a challenging exam. It is noted that Sub 110 is included in Sub 132, indicating that passing C in addition to passing A and studying B within the same month has an even stronger influence on graduating on time.

Subs 94, 134, and 267 model behaviors related to the second semester. It is worth noting that the exam F, which is not included in any subs of Fig. 16, has no apparent

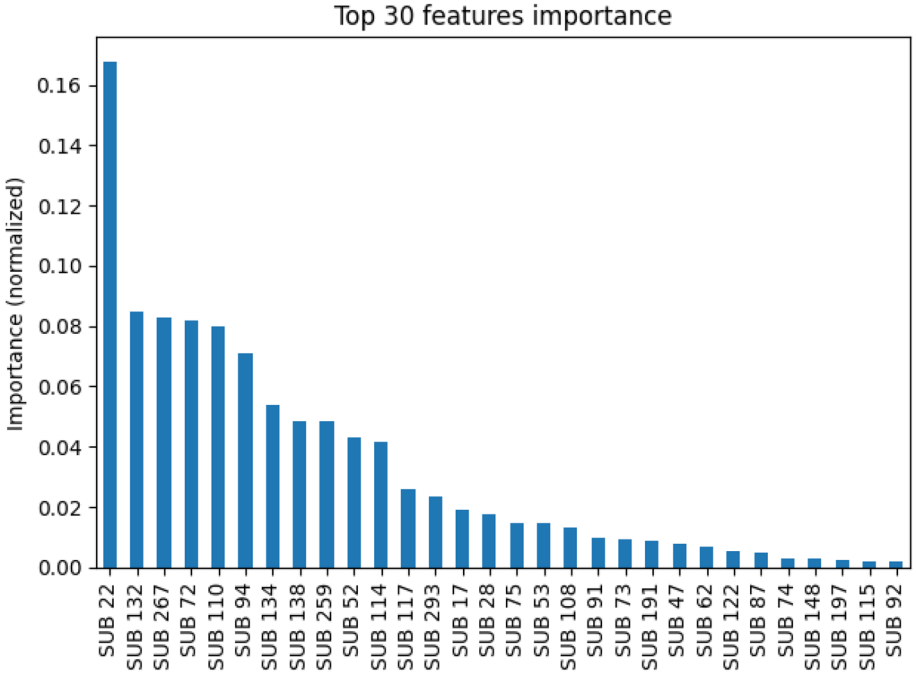


Fig. 14 Feature importance for surrogate model. Subprocesses only

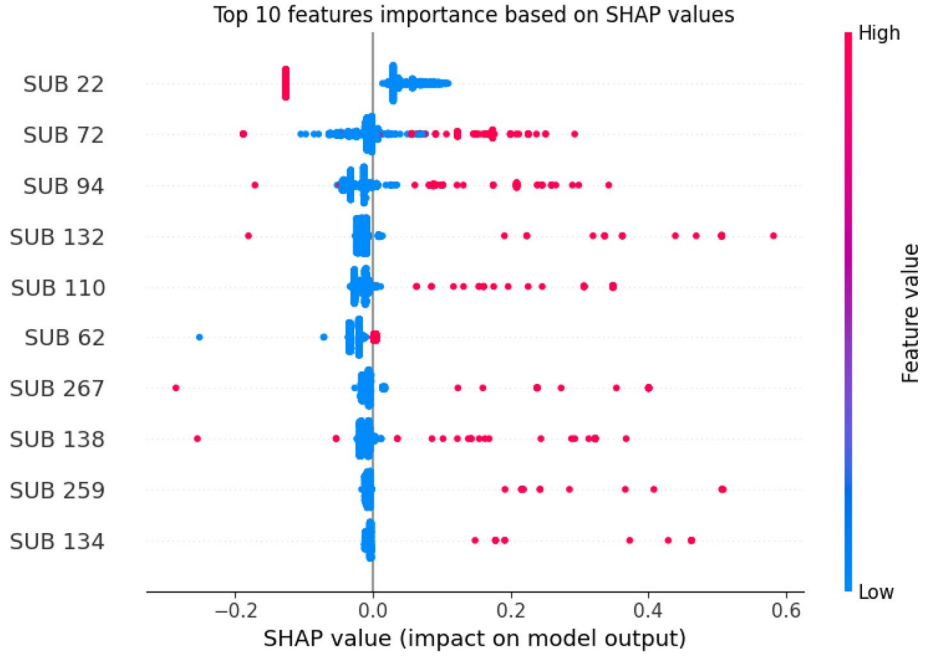


Fig. 15 SHAP values for timely class on surrogate model. Subprocesses only

influence on the model's prediction. Sub 134 suggests that passing D within one month of a previous failed attempt positively impacts graduating within the expected timeframe. Furthermore, passing the exam G studying it together with other exams, appears to positively influence the prediction of the timely class (Subs 259 and 267). Another inclusion relationship exists between Sub 94 and Sub 267. In this case, studying E and passing D within one month shifts the prediction toward the timely class, while also passing G provides an additional advantage. Finally, the exam B is never passes in any of top ten important subs, indicating that passing B within the first year does not appear to influence the prediction.

Discussion

The proposed approach shows promising practical relevance for monitoring and optimizing academic performance. By enabling early identification of potentially at-risk students, the predictive model allows academic institutions to effectively allocate support resources (e.g., tutors, additional lecture hours, additional in-class tutorials) toward the students who need them most. Beyond prediction, our approach provides insights into the reasons behind late graduation, expressed in terms of the behavioral paths students follow. This process-oriented explanation allows academic decision makers to make improvements in order to discourage patterns associated with suboptimal academic outcomes, for instance adding prerequisites, or modifying the scheduling of course offerings across academic years and semesters. Moreover, the identification of subprocesses witch positively impacts on timely graduation offers a basis for communicating effective evidence-based strategies to students, potentially guiding them toward more successful academic paths.

Interestingly, the robustness of our model across cohorts remains stable, regardless of variations in instructors and thus in the manner and level of detail with which the program is covered, and in the examination methods. This suggests that the structure of the

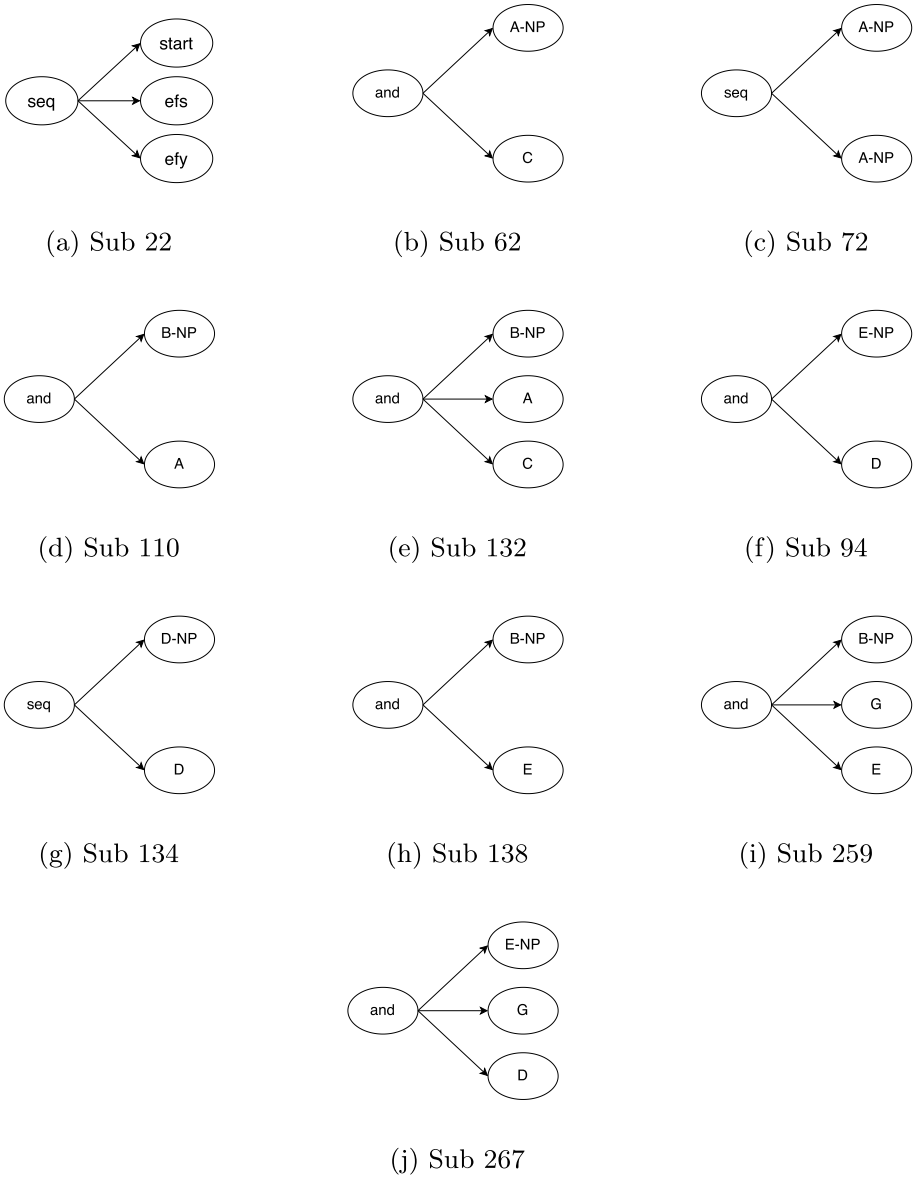


Fig. 16 Patterns with greatest impact on prediction

learning paths and the objective of the courses play a decisive role in students' exam-taking behaviors, reinforcing the generalizability of the observed patterns.

Overall, these results show the potential benefits of explicitly considering the impact of the order in which exams are taken when investigating students' performance, an aspect still under investigated in the literature (see "Related work" section).

Nevertheless, the approach presents some limitations. For instance, assuming a fixed one-month preparation window before exams may not reflect actual student behavior; in future work we could evaluate different approaches, such as defining preparation periods proportional to the number of ECTS credits. Moreover, the deployment of the proposed method may be constrained by current privacy regulations; for example, it may be necessary to obtain students' informed consent at enrollment before applying the predictive model. It is worth noting that the proposed methodology becomes increasingly useful the earlier the prediction is performed in the student's academic path. In the

considered case study, we do not have enough data to make a reliable prediction in the first semester. Indeed, the number of attempts to pass exams during the first semester alone is very low. Hence we must postpone the analysis to the end of the first year.

Another technical limitation concerns the inability to convert a small subset of the IGs into a block-structured Petri net, which leads to a small loss of analyzable students (i.e., 42 out of over 708 students). However, this affects only the interpretability layer and not the prediction, so this limitation does not undermine the practical value of the approach, given that decision makers are typically more interested in global behavioral patterns than individual-level interpretability.

Related work

Educational Data Mining (EDM) is an emerging discipline that aims to understand and improve students' learning process (Peña-Ayala 2014; Romero and Ventura 2007). Our work is mainly related to EDM approaches that analyze students' academic performance and their failure. A popular trend in this respect consists in modeling students according to predefined features and applying machine learning to predict student's performance (Dekker et al. 2009; Pardos et al. 2012; Guruler et al. 2010; Herzog 2005; Lassibille and Gómez 2008; Romero et al. 2008). Many of these studies provide a perspective complementary to the one provided by our analysis, taking into account factors external to the graduation process itself. Even studies centered on the graduation process usually perform a data-oriented analysis, in which students' behaviors are encoded in terms of features without considering the study program's underlying structure. In this respect, our work is similar to the one in Campagni et al. (2015), which proposes to model and analyze students' careers. They introduce the notion of *ideal career* that corresponds to the career of a graduated student who took each exam just after the end of the corresponding course. Different metrics are used to measure the distance between each student's career and the ideal one (e.g., the Bubblesort distance). Compared to our approach, the work in Campagni et al. (2015) does not exploit the potentialities of process-based analysis in modeling students' behaviors. In particular, they do not infer any model representing the overall students' behaviors. In contrast, we exploit process formalisms and domain knowledge to explicitly account for parallelisms, thus allowing us to obtain a more accurate evaluation of the difference between single careers and the ideal path.

The application of process mining techniques to educational data, referred to as *Educational Process Mining* (EPM) (Bogarín et al. 2018), is a subject that has been recently gaining increasing interest. EPM has been applied to deal with different educational problems. One of the most studied applications of EPM is the analysis and improvement of students' learning process in on-line learning environments where a number of approaches have investigated the use of process discovery techniques to understand students' learning behaviors (Bogarín et al. 2014; Vidal et al. 2016; Real and Pimentel 2025; Sit and Kong 2024), possibly correlating them with students' performance (Mukala et al. 2015; Nai et al. 2024). Other studies showcased the use of conformance checking techniques to compare students' behaviors against an expected standard, e.g., to evaluate the effectiveness of recommended learning paths (Martínez-Carrascal et al. 2023; Davis et al. 2016). Other frequently mentioned applications of EPM are computer-supported collaborative learning tools (Bergenthum et al. 2012; Reimann et al. 2009), and professional training (Bergenthum et al. 2012; Cairns et al. 2015). However, only a few

works investigated the applications of EPM to curriculum mining. Trcka and Pechenizkiy (2009) propose a set of patterns modeling typical constraints of academic curricula and use these patterns to analyze the graduation process. However, unlike the present work, they do not infer the process model representing students' careers and do not focus on delay analysis. Studies in Cameranesi et al. (2017); Potena et al. (2024) consider the entire student's career with a specific focus on distinguishing behaviors related to better or worse performing students. The former applies process discovery techniques to curriculum event logs with the aim of comparing behaviors of students that performed well or poorly in terms of years required to complete the graduation process and final grade. However, no predictive analysis is carried out. The latter shifts the focus to classes of timely and late students, similar to our study However, the only features considered there for the prediction are the number of exams taken in the first semester and in the first year, together with the average grade. Furthermore, only a simple logistic regression model is tested. Janssenswillen (2021) carry out several analysis on student's trajectories, aimed at discovering i) deviations between prescribed and actual trajectories, and their impact on students' performance, ii) the impact of failing one or more courses on the remaining part of the students' trajectory, and iii) analyzing the decisions for course selection. Hobeck et al. (2023) analyze the exam-taking process and program study compliance. In their analysis, these studies do not consider the potential impact of exam-taking process patterns on students' performance. In this regard, the approach proposed in Wang and Zaïane (2018) is closer to the one adopted in this paper, as they evaluate the use of process mining, dependency graphs, and sequence pattern mining to develop a course recommender system. These recommendations, however, only come from simple statistics on differences between previous behaviors of similar students and the corresponding outcome. In this work, we perform an in-depth analysis of how to leverage machine learning and deep learning approaches to learn robust relations among students' behaviors and their outcomes. Furthermore, they do not leverage our notion of process patterns, nor of instance control-flow.

Conclusion and future works

This study focused on the analysis of the students' behaviors during their first year. The goal was to determine if and how these behaviors influenced graduation times. The experimental results showed the feasibility of leveraging these properties to estimate students' performance early during their careers. In particular, the experiments show that adopting a process-aware perspective when modeling students' careers has a significant impact on the classification performance. These results suggest further research on the topic. In future work, we want to extend our analysis to the second year of students' careers. In the university under investigation, this year is characterized by even higher flexibility, with the students having the opportunity to freely choose several elective courses. This makes it interesting, but also challenging, to determine whether there are regularities in their behaviors and whether they could be leveraged to identify students in need of help or to provide the students with concrete recommendations. Another potential improvement involves altering the concept of concurrency employed in this study. In the current work, exams taken within a month or less were considered as occurring in parallel. Adopting a different concurrency criterion would lead to the generation of different instance graphs and, consequently, different subgraphs. Finally,

we aim to explore how to transform the insights gained from this analysis into practical recommendations. These recommendations will help degree program coordinators suggest optimal paths to students at the time of enrollment or the best career paths at various stages, considering their current progress.

Acknowledgements
Omitted for peer review.

Authors' contributions
All authors contributed equally to the conceptualization, the methodology, the experiments and their interpretation. All authors contributed equally to the original draft preparation.

Funding
Not applicable.

Data availability
No datasets were generated or analysed during the current study.

Declarations

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 31 March 2025 / Accepted: 18 August 2025
Published online: 10 September 2025

References

Bergenthum J Robiand Desel, Harrer A, Mauser S (2012) Modeling and mining of learnflows. In: Transactions on Petri Nets and Other Models of Concurrency V. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 22–50. https://doi.org/10.1007/978-3-642-29072-5_2

Bogarín A, Cerezo R, Romero C (2018) A survey on educational process mining. WIREs Data Min Knowl Discov 8:e1230. https://doi.org/10.1002/widm.1230

Bogarín A, Romero C, Cerezo R, Sánchez-Santillán M (2014) Clustering for improving educational process mining. In: Proceedings of the Fourth International Conference on Learning Analytics And Knowledge. pp 11–15. https://doi.org/10.1145/2567574.2567604

Cairns AH, Gueni B, Assu J, Joubert C, Khelifa N (2015) Analyzing and improving educational process models using process mining techniques. In: Proceedings of 5th International Conference on Advances in Information Mining and Management. International Academy, Research, and Industry Association (IARIA), pp 17–22

Cameranesi M, Diamantini C, Genga L, Potena D (2017) Students' careers analysis: a process mining approach. In: Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics. New York, pp 1–7. https://doi.org/10.1145/3102254.310227

Campagni R, Merlini D, Sprugnoli R, Verri MC (2015) Data mining models for student careers. Expert Syst Appl 42:5508–5521. https://doi.org/10.1016/j.eswa.2015.02.052

Davis D, Chen G, Hauff C, Houben GJ (2016) Gauging mooc learners' adherence to the designed learning path. In: Proceedings of the 9th International Conference on Educational Data Mining. pp 54–61

Dekker G, Pechenizkiy M, Vleeshouwers J (2009) Predicting students drop out: a case study. In: Proceedings of the 2nd International Conference on Educational Data Mining. pp 41–50

Diamantini C, Genga L, Potena D, van der Aalst W (2016) Building instance graphs for highly variable processes. Expert Syst Appl 59:101–118

Diamantini C, Genga L, Potena D (2016) Behavioral process mining for unstructured processes. J Intell Inf Syst 47:5–32. https://doi.org/10.1007/s10844-016-0394-7

Guruler H, Istanbullu A, Karahasan M (2010) A new student performance analysing system using knowledge discovery in higher educational databases. Comput Educ 55:247–254. https://doi.org/10.1016/j.compedu.2010.01.010

Hamilton WL, Ying R, Leskovec J (2017) Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, pp 1025–1035

Herzog S (2005) Measuring determinants of student return vs. dropout/stopout vs. transfer: a first-to-second year analysis of new freshmen. Res High Educ 46:883–928. https://doi.org/10.1007/s11162-005-6933-7

Hobeck R, Pufahl L, Weber I (2023) Process mining on curriculum-based study data: A case study at a german university. In: Process Mining Workshops. Springer Nature Switzerland, Cham, pp 577–589. https://doi.org/10.1007/978-3-031-27815-0_42

Janssenswillen G (2021) Student trajectories in higher education. In: Unearthing the Real Process Behind the Event Data: The Case for Increased Process Realism. Springer International Publishing, Cham, vol 412. pp 177–193. https://doi.org/10.1007/978-3-030-70733-0_8

Jonyer I, Cook DJ, Holder LB (2002) Graph-based hierarchical conceptual clustering. J Mach Learn Res 2:19–43. https://doi.org/10.1162/153244302760185234

Lassibille G, Gómez LN (2008) Why do higher education students drop out? Evidence from Spain. Educ Econ 16:89–105. https://doi.org/10.1080/09645290701523267

Leemans SJJ, Fahland D, van der Aalst WMP (2013) Discovering block-structured process models from event logs - a constructive approach. In: Application and Theory of Petri Nets and Concurrency. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 311–329. https://doi.org/10.1007/978-3-642-38697-8_17
Lundberg SM, Lee SI (2017) A unified approach to interpreting model predictions. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, pp 4768–4777
Martínez-Carrascal JA, Munoz-Gama J, Sancho-Vinuesa T (2023) Evaluation of recommended learning paths using process mining and log skeletons: Conceptualization and insight into an online mathematics course. IEEE Trans Learn Technol 17:555–568. https://doi.org/10.1109/TLT.2023.3298035
Mukala P, Buijs JC, Leemans M, van der Aalst WM (2015) Learning analytics on coursera event data: A process mining approach. In: Proceedings of the 5th International Symposium on Data-Driven Process Discovery and Analysis. CEUR-WS.org, pp 18–32
Nai R, Sulis E, Genga L (2024) Enhancing e-learning effectiveness: a process mining approach for short-term tutorials. J Intell Inf Syst 62:1773–1794. https://doi.org/10.1007/s10844-024-00874-9
Pardos ZA, Gowda SM, Baker RS, Heffernan NT (2012) The sum is greater than the parts: ensembling models of student knowledge in educational software. ACM SIGKDD Explor Newsl 13:37–44. https://doi.org/10.1145/2207243.2207249
Peña-Ayala A (2014) Educational data mining: a survey and a data mining-based analysis of recent works. Expert Syst Appl 41(4):1432–1462. https://doi.org/10.1016/j.eswa.2013.08.042
Potena D, Genga L, Basta A, Mercati C, Diamantini C (2024) Evidence-based student career and performance analysis with process mining: A case study. Process Mining Workshops. Springer Nature Switzerland, Cham, pp 349–360. https://doi.org/10.1007/978-3-031-56107-8_27
Potena D, Genga L, Galeazzi L, Vigano G, Diamantini C (2025) Assessing the impact of exam preparation process on students' careers. Process Mining Workshops. Springer Nature Switzerland, Cham, pp 142–153. https://doi.org/10.1007/978-3-031-82225-4_11
Real E, Pimentel E (2025) An educational process mining model on students' paths data from virtual learning environments. Technol Knowl Learn 1–26. https://doi.org/10.1007/s10758-025-09824-y
Reimann P, Frerejean J, Thompson K (2009) Using process mining to identify models of group decision making in chat data. In: Proceedings of the 9th International Conference on Computer Supported Collaborative Learning. International Society of the Learning Sciences, pp 98–107
Romero C, Ventura S (2007) Educational data mining: a survey from 1995 to 2005. Expert Syst Appl 33:135–146. https://doi.org/10.1016/j.eswa.2006.04.005
Romero C, Ventura S, Espejo PG, Hervás C (2008) Data mining algorithms to classify students. In: Proceedings of the 1st International Conference on Educational Data Mining. pp 8–17
Sit CYE, Kong SC (2024) A deep learning framework with visualisation for uncovering students' learning progression and learning bottlenecks. J Educ Comput Res 62:3–29. https://doi.org/10.1177/07356331231200
Trcka N, Pechenizkiy M (2009) From local patterns to global models: towards domain driven educational process mining. In: Proceedings of the 9th International Conference on Intelligent Systems Design and Applications. IEEE Computer Society, USA, pp 1114–1119. https://doi.org/10.1109/ISDA.2009.159
van der Aalst W (2016) Getting the data. In: Process Mining: Data Science in Action. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 125–162. https://doi.org/10.1007/978-3-662-49851-4_5
Vidal JC, Vázquez-Barreiros B, Lama M, Mucientes M (2016) Recompiling learning processes from event logs. Knowl-Based Syst 100:160–174. https://doi.org/10.1016/j.knosys.2016.03.003
Wang R, Zaïane OR (2018) Sequence-based approaches to course recommender systems. In: Database and Expert Systems Applications: 29th International Conference. Springer International Publishing, Cham, pp 35–50. https://doi.org/10.1007/978-3-319-98809-2_3
Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS (2021) A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst 32:4–24. https://doi.org/10.1109/TNNLS.2020.2978386
Zhang M, Cui Z, Neumann M, Chen Y, Association for the Advancement of Artificial Intelligence (2018) An end-to-end deep learning architecture for graph classification. Proceedings of the AAAI Conference on Artificial Intelligence 32:4438–4445. https://doi.org/10.1609/aaai.v32i1.11782

Publisher's Note