Extracting Rules from Event Data for Study Planning

Majid Rafiei[1], Duygu Bayrak[1], Mahsa Pourbafrani[1], Gyunam Park[1], Hayyan Helal[2], Gerhard Lakemeyer[2], and Wil M.P. van der Aalst[1]

[1] Chair of Process and Data Science, RWTH Aachen University, Aachen, Germany
[2] Knowledge-Based Systems Group, RWTH Aachen University, Aachen, Germany

Abstract. In this study, we examine how event data from *campus management systems* can be used to analyze the study paths of higher education students. The main goal is to offer valuable guidance for their study planning. We employ process and data mining techniques to explore the impact of sequences of taken courses on academic success. Through the use of decision tree models, we generate data-driven recommendations in the form of rules for study planning and compare them to the recommended study plan. The evaluation focuses on RWTH Aachen University computer science bachelor program students and demonstrates that the proposed course sequence features effectively explain academic performance measures. Furthermore, the findings suggest avenues for developing more adaptable study plans.

Keywords: Event Data, Decision Trees, Campus Management Systems, Machine Learning

1 Introduction

In higher education, study programs have specific examination regulations covering various aspects, such as degree requirements, grading criteria, thesis guidelines, and module grade cancellation procedures. These regulations provide guidelines for students to follow throughout their studies. For example, the examination regulation for the RWTH Bachelor computer science program of 2018 mandates that students must have accumulated a minimum of 120 credit points (CP) before registering for the thesis. With such requirements in mind, students can choose different paths and courses based on their interests. The examination regulations often include a recommended study plan to guide students in completing their compulsory courses in the most suitable order.

Following the recommended study plan can be beneficial for timely graduation, assuming all required courses are successfully completed. However, students have different capacities to handle the workload suggested by the study plan each semester, leading some to deviate from it and lose its guidance. Therefore, assistance and guidance are necessary to help students effectively plan their studies. One way to provide this assistance is by analyzing historical study path data, which can reveal characteristics of successful paths resulting in good overall GPAs. By extracting insights from such data, we can generate rules or recommendations to support students in making informed decisions about their course selection and study plans.

This paper focuses on utilizing event data from Campus Management Systems (CMS) to understand how students progress in their study programs. Various feature extraction methods capture the course sequence. Decision tree models trained with these features and Key Performance Indicators (KPIs) like GPA provide data-driven study planning recommendations to students. We evaluate the proposed features and models using data from computer science bachelor program students at RWTH Aachen University. The results show that the features effectively explain academic performance measures, like overall GPA and course grades. Comparing different features and models reveals their similar predictive effectiveness. We extract study planning rules from the models and discuss characteristics of study paths leading to positive or negative academic outcomes. Adhering to the recommended course sequence correlates with academic success, but deviations can also lead to positive outcomes, providing opportunities for more flexible study plans.

The remainder of this paper is structured as follows. In Section 2, we provide preliminaries. Section 3 outlines the related work. We introduce the used dataset in Section 4. Our main contributions are highlighted in Section 5. In Section 6, we present the evaluation, and Section 7 concludes the paper.

2 Preliminaries

Event Data (Log) Process mining utilizes event data which is a collection of events, where each event has the following essential attributes: *case-id*, *activity*, and *timestamp* [1]. The case-id identifies the instance, the activity represents the action, and the timestamp records the activity's time. Additional attributes like resources, costs, and people may provide contextual information. See Table 1 for a sample event log. Events represent specific activities in the process, and multiple events form a case. Using timestamps, we create traces, representing each case's activity sequence. For example, the trace $\sigma = \langle op, pp, pa, sh \rangle$ corresponds to the activities for order (case) with id 1 in Table 1. Event logs can be represented as multisets of traces, such as $L = [\langle op, pp, pa, sh \rangle^2]$ for Table 1.

Directly Follows Graph (DFG) DFGs are forms of process models, represented as directed graphs, where nodes are activities and edges signify the *directly-follows* relationship between activities. Fake start and end activities connect all first and last activities. DFG discovery involves counting how often one activity follows another. Figure 1 displays the DFG discovered from the event log $L = [\langle a,b,d \rangle^{10}, \langle a,b,c,d \rangle^{20}, \langle a,c,d \rangle^{5}]$.

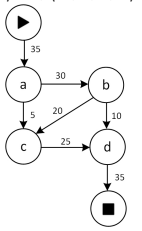


Fig. 1: The DFG discovered from event log $L = [\langle a,b,d \rangle^{10}, \langle a,b,c,d \rangle^{20}, \langle a,c,d \rangle^{5}]$.

Order ID	Customer ID	Activity	Timestamp
1	1	Order placement (op)	01-04-2023 16:02:00
1	1	Picking products (pp)	01-04-2023 19:46:00
2	2	Order placement (op)	01-04-2023 20:13:00
1	1	Packing (pa)	02-04.2023 08:07:00
2	2	Picking products (pp)	02-04.2023 08:35:00
2	2	Packing (pa)	02-04.2023 09:21:00
1	1	Shipping (sh)	02-04.2023 10:05:00
2	2	Shipping (sh)	02-04.2023 10:05:00

Table 1: A fragment of an event log.

Decision Tree A Decision Tree is a tree-based structure used for classification tasks. It has internal nodes representing decision points and leaf nodes representing decision outcomes. In classification, the model predicts a class label for input data by considering descriptive attributes [8]. Various techniques exist for selecting the best attribute for a split in a decision tree, e.g., *Information Gain* [9] or *Gini Index* [4].

3 Related Work

A comprehensive survey of process mining techniques in the educational context can be found in [3]. In the first categories of studies, students are grouped based on academic performance indicators, like course grades, and specific process models (e.g., DFG) are discovered for each group. The relation between students' usage of the online educational system and their grades is explored in [7] and [6], with consistent findings that actively engaged students to achieve better grades. These studies focus on applying process mining to analyze student groups individually.

Besides analyzing different student groups, study paths derived from CMSs and curriculum mining techniques are employed. The main focus in the second category of studies is on the recommendation and improvement of the study path. In [2], the effectiveness of a curriculum is evaluated by comparing predefined study paths with actual study paths taken by students, focusing on first-time course enrollments and excluding retakes. Results indicate the need for potential curriculum modifications, as some courses were not taken in the predefined semester. In [11], the authors discuss using process mining, particularly process discovery, on curriculum data to uncover study paths followed by students. They propose the development of a course recommender system by comparing processes followed by successful and less successful students. Also, in [5], the authors utilize process mining techniques to discover process models and gain insights regarding typical patterns followed by students.

In [12], the authors explored three different sequence-based course recommendation systems, including one based on process mining. Despite [11], in this approach, a process model was not discovered, but a causal footprint approach [1] for conformance checking was utilized to compute the similarity between a student and successful students based on their study paths. The recommendations were based on the taken courses by successful students with similar paths. Data mining techniques predict academic performance using diverse factors. They are also used to analyze demographics, digital footprints, and academic indicators, e.g., grades [13,10].

Our research utilizes process and data mining techniques to explore connections between courses using event data from a CMS. Unlike previous studies, we focus on students' study paths across multiple courses, including the impact of retaking courses. Our aim is to support students starting their studies by suggesting appropriate courses. Additionally, we analyze study path characteristics related to academic performance, distinguishing our approach from comparing actual study paths to recommended plans.

4 Dataset Description

Descriptive analysis is performed to gain insights into different aspects of the data. We analyze event data extracted from RWTH Aachen University's CMS, compris-

student-id	course-id	credit	time-start	time-end	semester	grade	final -status	gender	nationality	study-time
331322	course-107	8	05.02.19	08.02.19	1	3.3	PASSED	gender-2	country 1	3.0027
331324	course-107	8	05.02.19	08.02.19	1	3	PASSED	gender-2	country 1	3.0027
331354	course-107	8	05.02.19	08.02.19	1	1.7	PASSED	gender-2	country 2	3.0027
...	...	...	...	...	...	...	...	...	...	...

Fig. 2: Fragment of a larger event log. Each row corresponds to an event.

ing exam attempts and grade entries of computer science bachelor's students. The data covers the period from the winter semester 2018/19 to the summer semester 2021, focusing on students following the examination regulations of 2018. Filtering ensures the inclusion of only exams aligned with mandatory courses for the Computer Science Bachelor's program. The cleaned event data consists of 10751 events, 1411 students, and 18 courses.

Figure 2 shows a fragment of the event data, where each row represents an event indicating an exam taken by a student. This includes both passed and failed exam attempts. Each event has the presented attributes in Table 2. We consider the student-id attribute as the case identifier. The exam attempts for specific courses, identifiable by the course-id attribute, are considered as activities. Timestamp options include the exam date or a coarser-grained timestamp, such as the semester.

By assuming the exam date, i.e., the time-start attribute, as the timestamp attribute, we get the dotted chart shown in Figure 3. In this chart, each dot refers to an event (exam attempt). The dots are aligned horizontally according to the timestamp and vertically according to the case, i.e., dots in a single horizontal line represent a single student and define a trace.

From left to right, we can spot six groups, each representing an exam period. The first exam period will take place between January and April 2019. The second period is from July to October 2019, and so on. In addition, we can see three student cohorts. The first began its studies in the winter semester of 2018/19 and has study IDs ranging from 328954 to 342392. The second cohort began a year later, in winter semester 2019/20, and covered IDs 343430-365485. The third and final cohort, which covers IDs 369089-386368, began in the winter semester 2020/21.

5 Approach

Figure 4 provides an overview of our approach to discovering study planning rules from event data. We utilize decision trees for their interpretability and feature

Table 2: Event attributes of the event data.

Attributes	Explanation
Student-id	the anonymized unique ID of the student that took the exam
Course-id	the anonymized name of the course
Credit	the number of ECTS-points assigned to the course
Time-start	the date when the exam was written
Time-end	the date of exam result published to CMS
Semester	semester counter value when the student took the exam
Grade	the grade of the exam attempt (can be missing)
Final-status	the status of the exam result (PASSED or FAILED)
Gender	the anonymized gender of the student taking the exam
Nationality	the anonymized nationality of the student taking the exam (country-1 or other)
Study-time	student's study duration (years) at the time of data extraction from the CMS

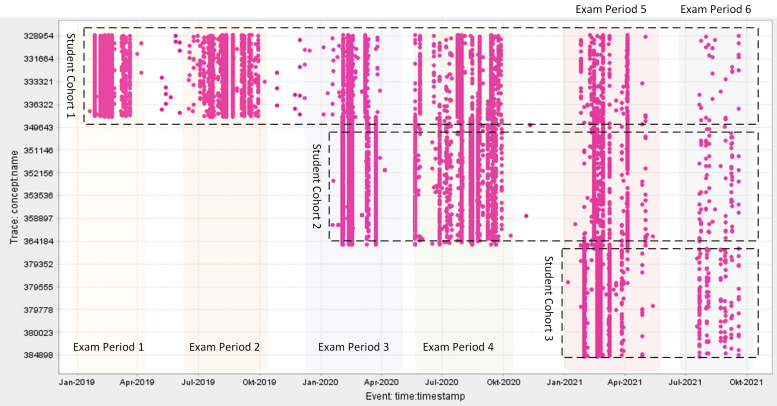


Fig. 3: Dotted chart showing exam attempt events of computer science bachelor students. The dotted chart was created using ProM Lite 1.3.

interaction capturing. Each path from the root to a leaf node in the trained decision tree can be translated into a rule. The discovered rules depend on the chosen descriptive features and label attributes. For example, using course semesters as descriptive features and overall GPA as the label attribute, a rule can suggest an excellent GPA if specific courses are taken in certain semesters. Our focus is mainly on order-related descriptive features, considering that recommended study plans often rely on course order.

5.1 Running Example

Table 3: Study path of an example student.

Semester	1	2	3	4	5
Courses	course-1 course-2	course-3	course-1	-	course-4 course-5

We illustrate different feature types using the student example from Table 3. In this case, the student's course sequence includes taking courses 1 and 2 in semester 1, course 3 in semester 2, retaking course 1 in semester 3 (after previously failing it in semester 1), not taking any courses in semester 4, and finally taking courses 4 and 5 in semester 5.

5.2 Feature Extraction

We introduce the following set of features that can be classified into atomic and non-atomic: *course semester*, *course order*, *course distance*, *path length*, *directly follows*, and *eventually follows*. With atomic features, we treat each exam attempt as an atomic event, while non-atomic features consider course life cycles and summarize

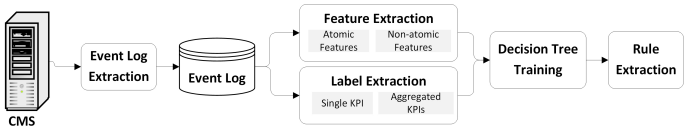


Fig. 4: Overview of the approach.

multiple exam attempts within a single course. The course life cycle starts with the first attempt and ends with the last attempt. We mainly focus on explaining the atomic features because the non-atomic features are incremental extensions of the atomic features.

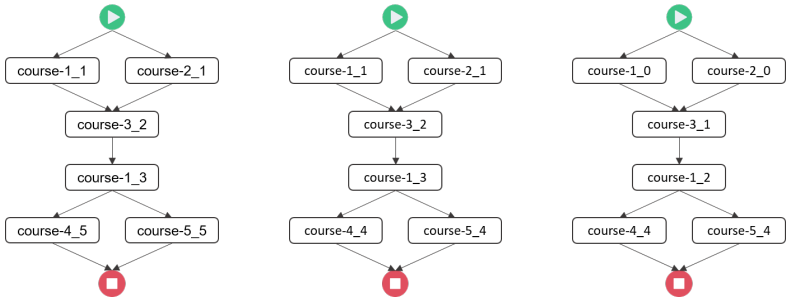
Atomic Course Semester (a-cs-): A course semester feature describes the semester in which a particular course was taken. It includes the course ID and an index value indicating the corresponding semester. This binary feature is set to true if the specified course was taken in the specified semester. In our example, the extracted course semester features are as follows: *a-cs-course-1-1 = true, a-cs-course-2-1 = true, a-cs-course-3-2 = true, a-cs-course-1-3 = true, a-cs-course-4-5 = true,* and *a-cs-course-5-5 = true.* All other features with different index values for those mentioned courses have the value false.

Atomic Course Order (a-co-): This feature captures the order of taken courses. It consists of the course ID and an index value specifying the order value of that course. The order value starts with one for the first course taken and increases for subsequent courses in the order they were taken. For our student example, the atomic course order features extracted are as follows: *a-co-course-1-1 = true, a-co-course-2-1 = true, a-co-course-3-2 = true, a-co-course-1-3 = true, a-co-course-4-4 = true,* and *a-co-course-5-4 = true.* Features with different index values for these courses have the value false. Note that course-4 and course-5 have the order value 4, even though they were taken in semester 5. The course order feature only captures course order and ignores any breaks between semesters.

Atomic Course Distance (a-cd-): This feature captures the number of semesters between each course and the first course taken. The feature includes the course ID and an index value indicating the semester distance. The first course taken has a distance of 0, courses taken in the subsequent semester have a distance of 1, and so on. In our example, *a-cd-course-1-0 = true, a-cd-course-2-0 = true, a-cd-course-3-1 = true, a-cd-course-1-2 = true, a-cd-course-4-4 = true,* and *a-cd-course-5-4 = true.* Every feature with other index values for those courses has the value false.

Atomic Path Length (a-pl-): The Path length feature focuses on measuring the number of edges that need to be traversed to go from one node to another in the DFG. Nodes representing courses taken in the same semester are arranged in parallel and share the same predecessor and successor nodes. The path length between these parallel nodes is set to zero, indicating that they are taken in the same semester. Therefore, the Path length feature captures the number of semesters that elapse between taking one course and taking another course, while disregarding any semester breaks when calculating the semester distance. If no path exists between two courses, the path length feature gets -1.

Note that to avoid loops in DFGs due to retaking courses after a failure, we include the previously introduced indices, i.e., semester, order, and distance, in activity names. Moreover, to be able to model concurrency, we convert DFGs to partial orders. Figure 5 depicts partial orders for our student example using different indices. Consequently, we define three different types of this feature: atomic path length with semester-index (*a-pl-s*), atomic path length with order-index (*a-pl-o*), and atomic path length with distance-index (*a-pl-d*).



(a) semester index **(b)** order index **(c)** distance index

Fig. 5: Partial orders of our student example obtained using different indices in the node names.

Atomic Directly Follows (a-df-): This feature indicates whether one course is directly followed by another course or not. The directly follows features are binary, representing the edges in the partial order. If there is a directed edge between two courses in the partial order, the corresponding directly follows feature will have a value of true. Note that similar to the path length feature, since we consider three different types of partial orders, we also consider three different types of this feature including *a-df-s* (for semester-index), *a-df-o* (for order-index), and *a-df-d* (for distance-index). In our running example, for instance, *a-df-s-course-1-1* $\rightarrow$ *course-3-2* = *true*. The courses not directly following each other get the value *false*.

Atomic Eventually Follows (a-ef-): This feature captures whether a course was eventually taken after another course. This feature tries to capture long-distance relations between courses. Again similar to the path length and directly follows features, we consider three different types of this feature including *a-ef-s* (for semester-index), *a-ef-o* (for order-index), and *a-ef-d* (for distance-index). The values of this feature are also binary. For instance, in our running example, considering semester as the index, *a-ef-s-course-1-1* $\rightarrow$ *course-3-2* = *true*. The courses not eventually following each other get the value *false*.

We extend the atomic features to incorporate the non-atomic definitions by introducing two features for each course, representing the start and end. The index values in these features indicate the semester, order, or distance values of the start and end, respectively. For our student example, the extracted non-atomic course semester features are: *na-cs-s-course-1-1* = *true*, *na-cs-e-course-1-1* = *false*, and *na-cs-e-course-1-3* = *true*.

To extend the features that are based on parial orders to non-atomic features, we need to extend the partial orders of individual students. This extension involves introducing two activities for each course: one representing the start and another representing the end of the course. In the extended partial order, there will be a directed edge between two activities if they happen directly after one another based on the semester value. Figure 6 illustrates the extended partial order for the student example. In this partial order, we can observe that the start and end activities of course-1 are arranged at different levels, indicating a longer life cycle caused by a retake. On the other hand, for the other courses, the start and end

activities are parallel aligned, indicating that they happen at the same time. Note that for non-atomic partial-order-based features, since we already have the start and end indicators, we do not face the loop issues due to retaken courses. Thus, we do not need indices for the courses.

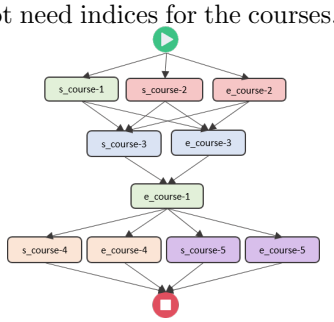


Fig. 6: Partial order of student example including the concept of course life cycles.

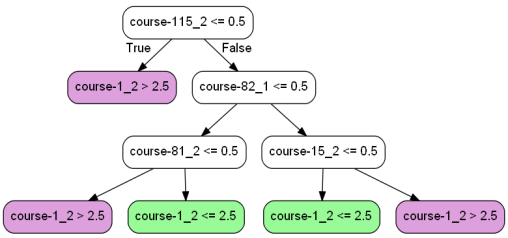


Fig. 7: Decision tree trained using atomic course semester features and 2-level course-1-2 grade label.

5.3 Label Extraction

To extract labels that capture academic performance, we need to establish metrics for measurement. One can consider two different groups: aggregate KPIs and single KPIs. The former assesses performance at a broader study level, while the latter focus on individual courses. In each group, we can define different KPIs. For instance, in the aggregated group, we can define three different KPIs: time-to-degree, overall GPA, and dropout. Also, in the single group, we can consider the course grade and pass/fail status as two different KPIs.

Some of the above-mentioned metrics are not applicable to our data due to our data limitations. For instance, the available data do not contain information about a student's graduation. The data only consists of exam attempts and grade entries of mandatory courses, excluding electives, courses in the applied subject area, and the thesis. Therefore, it is not possible to determine whether a specific student has already graduated or calculate their time-to-degree using the available data.

At the same time, since we follow the same approach for extracting rules from different label features, we only focus on the overall GPA and course-grade metrics. The grades and consequently overall GPAs in German universities are based on a 1-5 scale, where the lower grades are considered better, and 4.0 is considered as the passing threshold. We can group the overall GPA into different levels. For instance, a four-level classification is as follows: excellent ($GPA \leq 1.5$), good ($1.5 < GPA \leq 2.5$), satisfactory ($2.5 < GPA \leq 3.5$), and sufficient ($3.5 < GPA \leq 4.0$), and a two-level classification is as follows: good ($GPA \leq 2.5$) and satisfactory ($2.5 < GPA \leq 4.0$).

5.4 Rule Extraction

With features and labels extracted, the next step is rule extraction. We train a decision tree using the descriptive features and labels, then convert each root-to-leaf path into a rule. For instance, using the atomic course semester feature and a 2-level course grade label for course 1 taken in the second semester, we can obtain

Table 4: The five study planning rules.

1. **IF** course-115-2 ≤ 0.5 **THEN** course-1-2 > 2.5
2. **IF** course-115-2 > 0.5 **AND** course-82-1 ≤ 0.5 **AND** course-81-2 ≤ 0.5 **THEN** course-1-2 > 2.5
3. **IF** course-115-2 > 0.5 **AND** course-82-1 ≤ 0.5 **AND** course-81-2 > 0.5 **THEN** course-1-2 ≤ 2.5
4. **IF** course-115-2 > 0.5 **AND** course-82-1 > 0.5 **AND** course-15-2 ≤ 0.5 **THEN** course-1-2 ≤ 2.5
5. **IF** course-115-2 > 0.5 **AND** course-82-1 > 0.5 **AND** course-15-2 > 0.5 **THEN** course-1-2 > 2.5

the tree shown in Figure 7. To prevent overfitting in decision trees, we determine the maximum depth through experimentation and accuracy assessment. Note that in the decision nodes, if a course is taken in the specified semester, then *course semester ¡= 0.5* is evaluated as False. This decision tree can be translated into five study planning rules, which correspond to the paths from the root to any leaf node, Table 4. Note that not all rules may be equally relevant, as some may cover only a few instances or have low accuracy. Thus, we use a relevancy measure combining both the accuracy of the rules and the number of instances covered by the rule.

The extracted rules can be interpreted as follows: (1) If a student takes course-1 in the second semester without concurrently enrolling in course-115, it is anticipated that their grade in course-1 will be greater (worse) than 2.5, (2) If a student takes course-1 in semester 2 and enrolls in course-115 in the same semester and does not enroll in course-81 concurrently and course-82 in the preceding semester, it is foreseen that their grade in course-1 will exceed 2.5, signifying a below-average performance, (3) In contrast to rule 2, if the student concurrently enrolls in course-81 while having taken course-1 and course-115, their grade in course-1 is expected to be 2.5 or less, suggesting a better performance, (4) If a student concurrently enrolls in course-1 and course-115, has previously taken course-82, and doesn't take course-15 concurrently, they are projected to receive a grade in course-1 of 2.5 or less, and (5) If the student also takes course-15 concurrently with the rest except course-82, and course-82 is taken in semester 1, it is expected that their grade in course-1 will be greater than 2.5. Next, we evaluate the performance of each decision tree model trained on different feature and label combinations.

6 Evaluation

We developed a tool to train decision tree models on various features and discover rules from these models to predict good/bad grades in different courses. The source code is available on Github (https://github.com/m4jidRafiei/AIStudyBuddy-RuleExtractor). Note that here we only show the results for course grade metrics.

6.1 Course Grade Prediction

In this subsection, we predict course grades based on a two-level class (grade $\leq$ 2.5 and grade $>$ 2.5) using the dataset from section 4. Features are extracted following the approach in subsection 5.2. We focus on four key courses (IDs 45, 115, 71, and 131) frequently taken in different study stages. To train and evaluate decision tree models using the different datasets, we use a 4-fold cross-validation. In each iteration, we specify one fold as the test set and train a decision tree model

Table 5: Mean accuracy, precision, and recall values (%) of predicting the grade of different courses.

	Course-45	Course-115	Course-71	Course-131
Accuracy	67±5	68±1	60±3	69±2
Precision (Grade $\leq$ 2.5)	67±6	62±1	57±3	67±2
Precision (Grade $>$ 2.5)	76±8	76±2	64±2	73±4
Recall (Grade $\leq$ 2.5)	86±8	78±1	60±4	75±6
Recall (Grade $>$ 2.5)	45±17	60±2	61±5	64±5

Table 6: The three most relevant study planning rules.

1. **IF** course-140-4 $>$ 0.5 **AND** course-104-2 $>$ 0.5 **AND** course-106-4 $>$ 0.5 **AND** course-82-1 $>$ 0.5 **AND** course-115-4 $\leq$ 0.5 **THEN** class: course-131-4 $\leq$ 2.5
2. **IF** course-140-4 $\leq$ 0.5 **AND** course-78-3 $\leq$ 0.5 **AND** course-15-3 $\leq$ 0.5 **AND** course-1-4 $\leq$ 0.5 **THEN** class: course-131-4 $>$ 2.5
3. **IF** course-140-4 $>$ 0.5 **AND** course-104-2 $\leq$ 0.5 **AND** course-15-4 $>$ 0.5 **THEN** class: course-131-4 $>$ 2.5

on the remaining 3 folds. We evaluate the trained model on the test set, retain the evaluation score, and discard the model. Finally, we compute the average performance of the model using the evaluation scores of each iteration.

We use consistent hyperparameters for all decision tree models, including the Gini Index for splitting. The stopping criteria are a maximum tree depth of 5, a minimum of 1 sample per leaf node, and a minimum of 2 samples to split an internal node. We evaluate performance using accuracy, recall, and precision. Table 5 summarizes the performance metrics. Generally, average accuracy values are observed to be above 65%, except for models predicting grades for course-71, which began with an accuracy of 55%. Notably, the features were less successful in accounting for the grades in course-71 compared to the other three courses, as revealed by the mean precision and recall values.

For all four courses, the models struggled to accurately identify students with a course grade higher than 2.5, as reflected by mean recall values of 60% or lower. Nonetheless, with a mean precision of at least 64% across all four courses, the models tended to be correct when they did predict a grade above 2.5. Regarding the prediction of grades lower or equal to 2.5, models forecasting grades for course-45 and course-131 showed good performance, with average precision and recall values exceeding 67%. For course-115, the models performed well in detecting students with a grade lower or equal to 2.5, indicated by a mean recall of 78%, but with a reduced precision of 62%.

6.2 Extracting Rules

We analyze study planning rules for distinguishing good and bad course-131 grades. Specifically, we focus on rules extracted from the model trained on atomic course semester features to predict the fourth-semester course grade for course-131. See Figure 8 for a visual representation of the trained model. The three most relevant study planning rules based on a combined measure of sample count and accuracy are presented in Table 6.

Rule 1 indicates that to achieve a grade below 2.5 in course-131 during the fourth semester, the student should take courses 106 and 140 concurrently. They

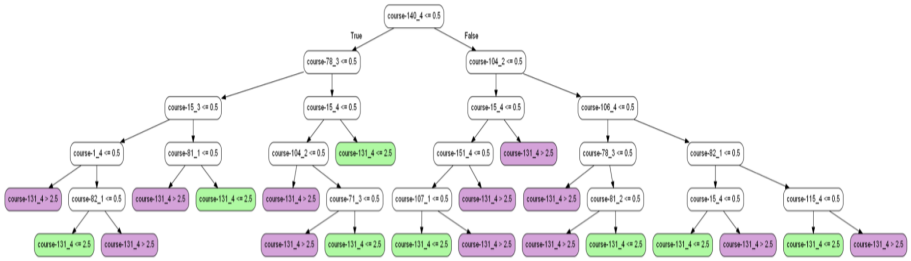


Fig. 8: Decision Tree to predict the grade for course-131 taken in 4th semester. For readability, we removed "cs-" in the feature prefix.

sem	1	2	3	4	>4
c-82	x				
c-104		x			
c-115	x	x	x		x
c-106				x	
c-140				x	

(a) Rule 1 (grade <= 2.5)

sem	1	2	3	4	>4
c-1	x	x	x		x
c-15	x	x		x	x
c-78	x	x		x	x
c-140	x	x	x		x

(b) Rule 2 (grade > 2.5)

sem	1	2	3	4	>4
c-104	x		x	x	x
c-15				x	
c-140				x	

(c) Rule 3 (grade > 2.5)

Fig. 9: Comparison of three most relevant atomic course semester rules (for predicting the grade of course-131) with suggested semesters by the university. The suggested semester is highlighted in orange. Rule conditions are indicated by a "x".

should also enroll in courses 82 and 104 during the first and second semesters. However, they should avoid taking course-115 concurrently with course-131 and enroll in it in any semester other than the fourth.

Figure 9(a) shows the recommended study plan compared to rule 1 conditions. The alignment between the rule's conditions and the advised study plan is evident, except for course-115. The model indicates that students can be flexible with course-115 scheduling without affecting course-131 performance. This suggests a positive outcome if course-115 is not taken alongside course-131.

Rules 2 and 3 prescribe study paths anticipated to have poor grades (above 2.5) in course-131. Rule 2 predicts a bad grade if courses 1 and 140 are not undertaken in parallel with course-131 in the fourth semester, and if courses 15 and 78 are not taken directly prior in the third semester. Figure 9(b) depicts these conditions and demonstrates their alignment with the deviation from the study plan for courses 15, 78, and 140. However, the condition concerning course-1 does not conflict with the proposed study plan, as its enrollment is suggested for the second semester.

On the other hand, Rule 3 forecasts a bad grade if courses 15 and 140 are undertaken concurrently with course-131, and if course-104 is not enrolled in the advised second semester. As depicted in Figure 9(c), course-140 is suggested for concurrent enrollment with course-131 in the study plan, but course-15 is advised to be taken prior to course-131.

7 Conclusion

Our study used process and data mining techniques to investigate the impact of course sequences on academic success by generating data-driven study planning recommendations for computer science bachelor program students at RWTH Aachen

University. These findings point to the possibility of developing more adaptable study plans. One limitation is that we focused on students who studied for the standard three-year period at RWTH Aachen University, which may not fully capture the long-term impact of study planning rules on academic performance. We lack information about holiday semesters, which can affect the accuracy of exam result assignments. Additionally, our analysis focused solely on mandatory courses in the computer science bachelor program, excluding elective courses, required courses from outside computer science, and the thesis which could affect the GPA. Our future steps include investigating the impact of the time between grade publication and the next exam on student performance, investigating elective course combinations that lead to better academic performance, and considering alternative classification models to uncover additional study planning rules.

Acknowledgement

The authors gratefully acknowledge the financial support by the Federal Ministry of Education and Research (BMBF) for the joint project AIStudyBuddy (grant no. 16DHBKI016).

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016)
2. Bendatu, Y., Yahya, B.N.: Sequence matching analysis for curriculum development. Jurnal Teknik Industri **17** (2015)
3. Bogarín, A., Cerezo, R., Romero, C.: A survey on educational process mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **8** (2017)
4. Breiman, L.: Classification and regression trees. Routledge (2017)
5. Cameranesi, M., Diamantini, C., Genga, L., Potena, D.: Students' careers analysis: a process mining approach. In: Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics. ACM (2017)
6. Cenka, B.A.N., Santoso, H.B., Junus, K.: Analysing student behaviour in a learning management system using a process mining approach. Knowledge Management and E-Learning **14**(1), 62 – 80 (2022)
7. Etinger, D.: Discovering and mapping LMS course usage patterns to learning outcomes. In: Proceedings of the 3rd International Conference on Intelligent Human Systems Integration (IHSI 2020). vol. 1131, pp. 486–491. Springer (2020)
8. Maimon, O.Z., Rokach, L.: Data mining with decision trees: theory and applications, vol. 81. World scientific (2014)
9. Quinlan, J.R.: Induction of decision trees. Machine learning **1**, 81–106 (1986)
10. Romero, C., Ventura, S.: Educational data mining: A review of the state of the art. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on **40**, 601 – 618 (2010)
11. Wang, R., Zaiane, O.R.: Discovering process in curriculum data to provide recommendation. In: Educational Data Mining (2015)
12. Wang, R., Zaïane, O.: Sequence-Based Approaches to Course Recommender Systems, 2018, Proceedings, Part I, pp. 35–50 (2018)
13. Yagci, M.: Educational data mining: prediction of students' academic performance using machine learning algorithms. Smart Learning Environments **9** (2022)