SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V11I11P104

Volume 11 Issue 11, 28-44, November 2024

Original Article

Leveraging Machine Learning for Predictive Pathways in Higher Education: A Case Study at Al-Zaytoonah University of Jordan

Mohammad Muhairat[1], Wael Alzyadat[1], Ameen Shaheen[1*], Aysh Alhroob[1], A. Nasser Asfour[1]

[1]*Department of Software Engineering, Al-Zaytoonah University of Jordan, Amman, Jordan.*

**Corresponding Author : a.shaheen@zuj.edu.jo*

Received: 03 September 2024 Revised: 05 October 2024 Accepted: 04 November 2024 Published: 03 December 2024

Abstract *- This study investigates the application of Machine Learning (ML) for developing predictive pathways in higher education, focusing on the software engineering bachelor program at Al-Zaytoonah University of Jordan. The primary objective is to create a comprehensive mapping system that assists academic planning by exploring various scenario combinations. The study utilizes the Apriori algorithm to identify frequent item sets and generate association rules, thus providing a robust approach for predicting academic trajectories. The analysis involves extracting and restructuring data from the study plan, followed by in-depth pattern identification using advanced ML techniques. The results emphasize the importance of incorporating domain knowledge to enhance prediction accuracy and reliability. This study lays the foundation for innovative academic planning tools, offering significant potential for broader applications in educational and other domains. Future work will focus on refining the predictive models and expanding the approach to other educational programs, aiming to further improve the effectiveness of academic planning and decision-making processes.*

Keywords *- Machine Learning, Predictive pathways, Apriori algorithm, Data mining, Academic planning component.*

1. Introduction

Machine Learning (ML) holds transformative potential for education by offering customized instruction and real-time feedback, which significantly enhances learning outcomes and the overall quality of education [1]. Through personalized educational experiences, machine learning helps students grasp complex concepts more effectively and maintain higher levels of engagement. Additionally, it improves assessment and evaluation processes by reducing biases inherent in traditional methods. However, association rules in machine learning often face limitations due to predefined support score thresholds, which can restrict the accuracy of impact assessments [2, 3].

While ML techniques are widely used to predict general student outcomes, there remains a gap in applying these techniques for guiding students through personalized academic pathways within structured, requirements-intensive curricula. In programs such as software engineering, students must navigate complex course requirements, prerequisites, and credit hour constraints, making effective academic planning challenging. This study addresses this gap by developing a machine learning-based predictive pathway model specifically tailored to the software engineering bachelor's program at Al-Zaytoonah University of Jordan.

Addressing this issue requires exploring alternative methodologies to refine these thresholds and enhance accuracy. By integrating domain knowledge into association rule generation with defined restrictions, more meaningful combinations and a comprehensive understanding of datasets can be achieved [4]. The predictive pathways are an innovative solution designed to address significant challenges faced by students in the software engineering bachelor program at Al-Zaytoonah University of Jordan.

This study aims to comprehensively present viable pathways in a structured format, assisting students, academic advisors, and stakeholders in making informed decisions regarding course selections (Force). This approach ensures strict adherence to academic requirements and credit hour limitations inherent in the program. The software engineering bachelor program at Al-Zaytoonah University spans 133 credit hours over four academic years, reflecting the university's commitment to offering a robust and well-rounded educational experience.

The program operates within a structured framework of three distinct semesters each academic year: the first semester, the second semester, and the summer semester; these semesters are crucial milestones in the academic journey, guiding students toward meeting credit hour thresholds while ensuring balanced academic progression. However, navigating the course pathways within this program presents significant challenges. The intricate nature of academic planning requires a deep understanding of credit hour limitations, prerequisite sequences, and strategic course allocation across semesters. Additionally, the dynamic nature of credit hour constraints, especially after reaching 90 credit hours, adds complexity that demands careful adaptation and planning to ensure a seamless academic trajectory [4, 5]. Machine learning methods can predict various scenarios based on historical data and patterns [6].

The primary goal of this study is to leverage machine learning to develop a predictive pathway approach tailored to the needs of students in the software engineering bachelor program at Al-Zaytoonah University of Jordan. This approach aims to provide a clear and structured presentation of viable academic pathways, helping students, academic advisors, and stakeholders make informed decisions regarding course selections. The study seeks to enhance academic planning and decision-making within the program by ensuring strict adherence to academic requirements and credit hour limitations.

The significance of this study is multifaceted. Firstly, it has the potential to revolutionize educational practices by offering customized instruction and real-time feedback, leading to improved learning outcomes and greater student engagement. Secondly, it aims to create a fairer assessment process by mitigating traditional biases. Thirdly, the predictive pathway approach offers a structured framework for academic planning, helping students and advisors navigate the complexities of course selection and academic progression. Lastly, the methodologies and insights from this study can be adapted and applied to other educational institutions, contributing to the broader field of educational technology and administration.

This article is organized methodically to guide readers through our research process. It starts with Section 2, which reviews existing studies on scenario prediction using machine learning. Building on this foundation, Section 3 introduces our predictive pathway approach, detailing the application of machine learning techniques. Section 4 outlines the experimental setup and results, highlighting the process of dataset generation and the implementation of three analytical tiers using Python 3.9 to derive statistical outcomes for different scenarios. In the subsequent section, we discuss our findings, addressing the challenges encountered and future directions for research. Through this predictive approach, we aim to provide a detailed understanding and practical solutions to enhance academic planning and decision-making for software engineering students at Al-Zaytoonah University.

2. Literature Review and Scenario Prediction Using Machine Learning

In this section, there are two main subsections. Section 2.1 provides an overview of how Machine Learning (ML) predicts students' academic performance. Section 2.2 presents a detailed analysis of alternative approaches and methods employed in scenario prediction, comparing them with machine learning-based approaches. It also highlights the strengths and limitations of various methods used for scenario prediction in educational settings.

2.1. Overview

Machine Learning (ML) has emerged as a powerful tool in the education sector, offering the potential for personalized learning experiences and more effective dissemination of educational content. With the increasing volume of data and evolving needs of higher education, particularly in digital education, ML techniques have significantly grown, creating more efficient learning experiences for diverse student populations [7]. One of the primary applications of ML in education is the provision of personalized learning experiences, which can significantly improve student engagement and predict academic performance and employability [8, 9]. Researchers have explored various ML techniques to address these goals. For instance, ML algorithms have been used to predict students' Graduation On Time (GOT) and estimate student outcomes in degree programs. Techniques such as Random Forest, Support Vector Machine, K-Nearest Neighbors, and Naïve Bayes have been employed and validated through cross-validation and parameter tuning [10, 11].

ML is also utilized to assess teaching effectiveness and inform instructional strategies. Academic administrators can leverage ML to identify areas of strength and weakness in teaching faculty, enabling targeted improvements in educational quality. Studies have demonstrated the application of ML in evaluating teacher effectiveness, thereby aiding in developing more effective instruction [1]. Numerous studies have focused on the efficacy of ML in predicting students' academic performance. For example, research has investigated the use of Generative Adversarial Networks (GANs) and Artificial Neural Networks (ANNs) to predict student performance and enhance the quality of information in universities. These studies highlight the potential of ML techniques to provide valuable insights into student performance, although they do not explicitly address other researchers' inquiries within the academic domain.

2.2. Related Works

This section provides an in-depth analysis of alternative approaches and methods employed in scenario prediction, comparing them with machine learning-based approaches.

Statistical methods have long been used in educational data analysis. Techniques such as linear regression, logistic regression, and factor analysis have been employed to predict student outcomes and identify factors influencing academic performance. While effective, these methods often lack the ability to handle large datasets and complex relationships as efficiently as machine learning algorithms.

- Linear Regression: Used to predict student retention rates based on high school GPA and standardized test scores. This method is effective in educational settings but is limited in handling complex, non-linear relationships [12].
- Logistic Regression: Employed to identify at-risk students by analyzing demographic and academic data. It effectively predicts student dropout but requires extensive data preprocessing and feature selection [13].
- Factor Analysis: Identifies factors affecting student motivation and engagement. This method provides insights into key elements influencing student success but lacks predictive capabilities compared to more advanced machine learning methods [14].
- Machine Learning Algorithms: Demonstrate superior performance in predicting student outcomes due to their ability to manage large datasets and identify complex patterns. Algorithms such as Random Forest, Support Vector Machine, K-Nearest Neighbors, and Naïve Bayes have been tested to predict students' Graduation on Time (GOT) and estimate outcomes in degree programs. These algorithms have been validated using cross-validation and parameter tuning, showing promising results in various studies [15, 16].
- Generative Adversarial Networks (GANs) and Artificial Neural Networks (ANNs) are employed to enhance the quality of information and predict student performance. These advanced machine learning models are particularly effective in handling non-linear relationships and large datasets, making them suitable for complex educational data. Studies have shown these models can significantly improve the accuracy of predictions compared to traditional methods [17].
- Decision Trees and Random Forest: Among the most effective techniques for predicting student performance. These models are easy to interpret and provide insights into factors influencing academic outcomes. They have been widely used to assess teacher effectiveness, inform instructional strategies, and predict student success [18, 19]
- Support Vector Machines (SVM) and K-Nearest Neighbors (KNN): Popular machine learning algorithms used in educational prediction. SVM is effective in high-dimensional spaces and suitable for both regression and classification tasks. KNN is a simple, instance-based learning algorithm that is easy to implement and interpret. Both algorithms have shown effectiveness in predicting student outcomes and identifying at-risk students [20]

There are many traditional statistical methods, such as linear and logistic regression, with advanced machine learning techniques, including GANs and ANNs. Traditional methods are user-friendly and interpretable but struggle with complex and large datasets. In contrast, machine learning models excel in accuracy and handling complexity but require significant computational resources and technical expertise. Selecting the appropriate method depends on data complexity and analysis goals, with a combination of both approaches potentially enhancing educational outcomes.

Table 1. Comparison of scenario prediction methods in education

Methods	Strengths	Limitations
Linear Regression	Effective for simple, linear relationships; interpretable	Struggles with complex, non-linear relationships
Logistic Regression	Good for binary classification (e.g., dropout prediction); interpretable	Requires extensive preprocessing and feature selection; limited to binary outcomes
Factor Analysis	Identifies underlying factors affecting performance; interpretable	Lacks predictive capabilities; not suitable for large, complex datasets
Machine Learning Algorithms	Handles large datasets well; identifies complex patterns; high predictive accuracy	Requires significant computational resources and tuning; less interpretable
GANs and ANNs	Handles non-linear relationships; effective for complex datasets; high accuracy	Computationally intensive; requires large datasets and careful tuning
Decision Trees and Random Forest	Easy to interpret; identifies key factors; effective for classification and regression	Can be prone to overfitting; requires careful tuning and validation
SVM and KNN	Effective in high-dimensional spaces; good for both classification and regression	SVM requires parameter tuning; KNN can be computationally intensive for large datasets

Table 1 highlights the strengths and limitations of various methods used for scenario prediction in educational settings. Traditional statistical methods like linear and logistic regression are often more interpretable and straightforward. These methods provide simplicity and clarity in their predictions, making them accessible for educators and

administrators who may not have a deep technical background. However, they struggle to handle the complexity and scale of modern educational data, particularly when it involves non-linear relationships or large datasets.

In contrast, machine learning algorithms and advanced techniques like Generative Adversarial Networks (GANs) and Artificial Neural Networks (ANNs) offer superior performance and accuracy. These advanced models excel in managing large datasets and identifying complex patterns, making them highly effective for predicting student outcomes and other educational scenarios. Despite their advantages, these methods require significant computational resources and expertise to implement effectively. They are also less interpretable than traditional statistical methods, hindering their adoption in some educational settings.

Based on Table1, A critical gap in academic planning by using the Apriori algorithm to identify common course patterns within a degree program, enabling scenario predictions that cater to individual student needs. Unlike other approaches, our model integrates specific degree requirements and prerequisites, providing practical guidance aligned with academic policies. This section examines current methodologies, evaluates their effectiveness, and highlights our predictive pathway model's unique contributions to the broader field of machine learning in education.

By understanding the differences between these methodologies, researchers and educators can better choose the appropriate tools for their specific needs and objectives. While traditional statistical methods offer ease of use and interpretability, advanced machine learning models provide the accuracy and complexity needed for modern educational data analysis. Selecting the right approach depends on the specific requirements of the educational setting, including the nature of the data and the goals of the analysis. Ultimately, by leveraging the strengths of both traditional and advanced methodologies, educators and researchers can enhance educational outcomes through more informed decision-making. This informed approach allows for selecting the most suitable predictive tools, leading to better planning, intervention, and overall improvement in educational practices.

3. Predictive Pathway Approach Leveraging Machine Learning (PPALML)

Predictive Pathway Approach Leveraging Machine Learning (PPALML) is organized into three distinct tiers designed to define and preserve specific domains. These elicitation, elaboration, and reinforcement tiers work together to create a comprehensive mapping system. This system allows for the exploration of various combination scenarios, facilitating a thorough investigation of diverse possibilities, as shown in Figure 1.

These elicitation, elaboration, and reinforcement tiers work together to create a comprehensive mapping system. This system allows for exploring various combination scenarios, facilitating a thorough investigation of diverse possibilities.

Figure 1 visually represents the flow and interaction between different components of the predictive pathway approach. It demonstrates how data is systematically transformed and analyzed through each tier to produce meaningful scenarios ultimately. The stages involved in the predictive pathway approach, which leverages machine learning techniques to model study plan scenarios, are detailed in subsections 3.1, 3.2, and 3.3.

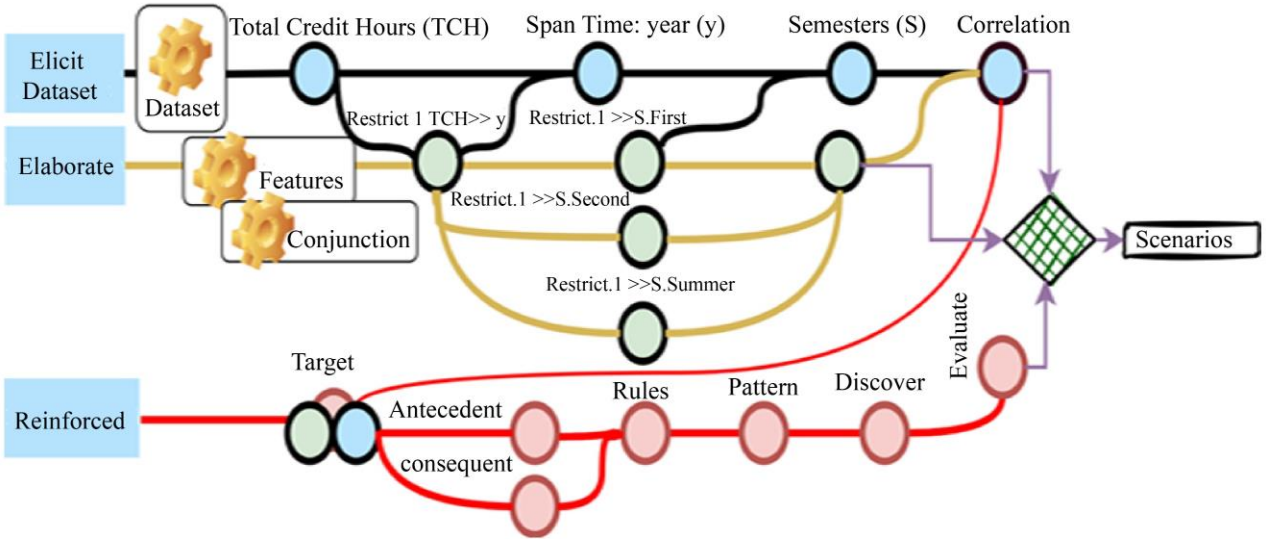

Fig. 1 Predictive Pathway Approach Leveraging Machine Learning (PPALM)

3.1. Elicit Dataset Tier

The Elicit Dataset process, as shown in Figure 1, involves extracting and restructuring data from the study plan for the software engineering bachelor's program [21]. This program requires a total of 133 credit hours, with most courses assigned 3 credit hours each, except for one specific course that carries 1 credit hour. Therefore, completing the bachelor's degree in software engineering necessitates 45 individual courses. Notably, students must complete 90 credit hours, equivalent to 30 courses, before registering for the 1-credit hour course.

The minimum time span to complete the bachelor's degree is 3 academic years. This means that students need to complete 45 credit hours each academic year, distributed as follows: 15 courses in the first academic year (45 credit hours), 15 courses in the second academic year (45 credit hours), and 15 courses in the third academic year (43 credit hours), which includes the specific 1-credit hour course. Typically, the normal span for completing the degree ranges from 4 to 6 academic years, with a maximum limit of 7 years. Students who fail to complete the 133 credit hours within this period are terminated from the program.

Each academic year is structured into three semesters: the first semester, the second semester, and the summer semester. The first and second semesters are mandatory, while the summer semester is elective. For the first and second semesters, students can choose to register for 12, 15, or 18 credit hours, or alternatively, 13, 16, or 19 credit hours if they have already completed 90 credit hours. For the summer semester, students can register for 0, 6, or 9 credit hours or 0, 7, or 10 credit hours under the same condition. Once students have completed 90 credit hours, they have a one-time choice of registering for 13, 16, or 19 credit hours in the first or second semesters and 0, 7, or 10 credit hours in the summer semester, focusing on the remaining 44 credit hours.

A correlation-based approach is used to create precise datasets. This method identifies relationships between various items within the dataset. Positive correlation values near 1 indicate a strong positive relationship, values around 0 indicate no relationship, and negative values near -1 indicate a negative relationship between the items. The items considered are the total credit hours, the academic years (1, 2, 3, 4), and the semesters (first semester, second semester, and summer semester).

To utilize the correlation method, connections are established between the indices of items to substantiate the chosen targets for the entire dataset [22]. The correlation equation is outlined as follows:

$$Correlation = \frac{n(\sum xy)-(\sum x)(\sum y)}{\sqrt{[n\sum x^2-(\sum x)^2][n\sum y^2-(y)^2]}} \quad (1)$$

Where n number of item points. ∑xy sum of the products of paired scores, ∑x and ∑y sums of the x-values and y-values.

This Equation (1) helps determine the strength and direction of the relationships between items, facilitating a comprehensive understanding of the data structure and aiding in creating an accurate and meaningful dataset.

The correlation heatmap visually expresses the significance of these relationships, with red indicating the strongest positive correlation (score +1) and blue to dark blue indicating no relationship or a negative relationship. This can be seen in Figure 2.

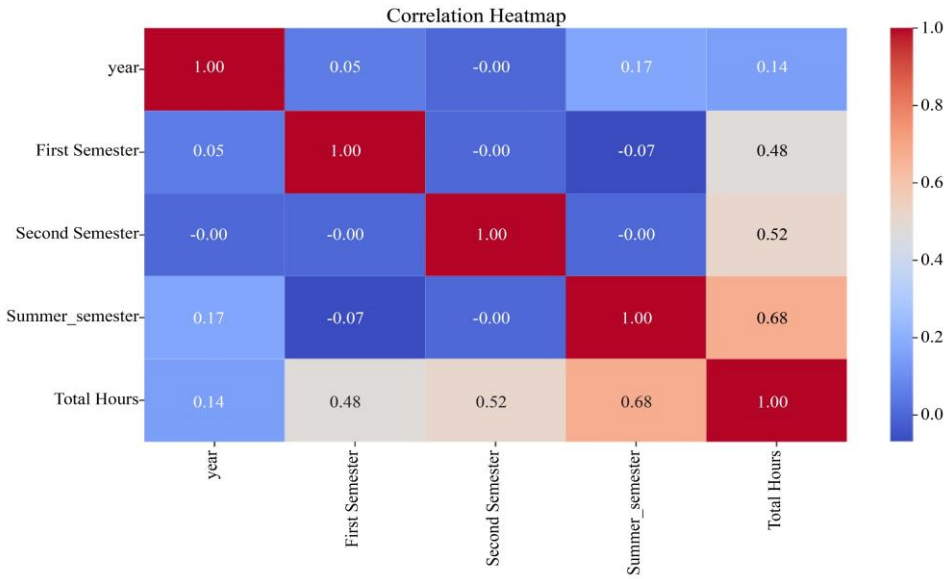

Fig. 2 Correlation between Items

The correlation method plays a crucial role in identifying the impact of specific variables within a dataset. Notable correlations are evident from the data depicted in Figure 2. There is a significant positive correlation of 0.68 between the summer semester and the total hours. Additionally, the second semester shows a correlation of 0.52 with the total hours, and the first semester has a correlation of 0.48 with the total hours. These correlations highlight the associations between various semesters and the total hours, providing valuable insights into their interrelationships within the dataset.

On the other hand, a negative correlation of -0.07 is observed between the first semester and the summer semester. The correlation between the first semester and the year is 0.05, while the correlation between the summer semester and the year is 0.17. These findings illustrate the varying relationships between different time frames within the dataset, enhancing our understanding of their interactions.

The Elicit Dataset tier is a crucial initial step in the predictive pathway approach. By meticulously extracting and restructuring data from the study plan, we have developed a comprehensive dataset that includes all the necessary elements for further analysis. This tier ensures that the data is well-organized and primed for detailed examination.

As we move into Section 3.2, the Elaborate Tier, we will engage in a deeper analysis of this dataset. The Elaborate Tier will focus on refining the data and identifying significant patterns and correlations essential for accurate scenario predictions. This step is critical for enhancing our understanding of the dataset and ensuring that our predictive models are both robust and reliable. The insights derived from this in-depth analysis will form the foundation for effective and strategic academic planning in the following stages.

3.2. Elaborate Tire

We carefully define the constraints governing the dataset, particularly its overall parameters. This includes a specific criterion where the total credit hours amount to 90, with exceptions for courses that equate to one credit hour. Additionally, the dataset spans a range of total credit hours across each academic year.

The generated dataset is divided into two distinct components. The first component shows the aggregate credit hours per year, aiming to reach the collective goal of 133 credit hours throughout the academic program. The second component details how these credit hours are distributed across individual semesters. The figure below illustrates the structural delineation of both components, providing a comprehensive understanding of this division.

The raw data structure, as illustrated in Figure 3, outlines the distribution of Total Credit Hours (TCH), totalling 133 hours. These hours are distributed across four academic years, labeled TCH_Year.1, TCH_Year.2, TCH_Year.3, and TCH_Year.4, with each year's credit hours ranging from 24 to 45. Each course typically holds a weight of 3 credits.

However, there is an exception that allows for a TCH value exceeding 90, thus expanding the range to 25-46 credit hours for that specific course and semester. This exception applies only once per semester, allowing for flexibility in academic planning. This detailed breakdown helps understand the allocation and distribution of credit hours, ensuring that the overall academic requirements are met efficiently.

The Elaborate Tier has provided a comprehensive approach for defining and analyzing the constraints within our dataset. By meticulously organizing the Total Credit Hours across four academic years and incorporating necessary exceptions, we have established a solid foundation for understanding academic planning. This groundwork is essential as we proceed to the next phase of our study.

In Section 3.3, the Reinforced Tier, we will expand on this analysis. This tier will involve applying advanced machine learning techniques to strengthen the dataset uncovering deeper patterns and correlations. Our goal is to enhance the accuracy and reliability of our predictive models, ensuring they are robust and effective for academic planning.

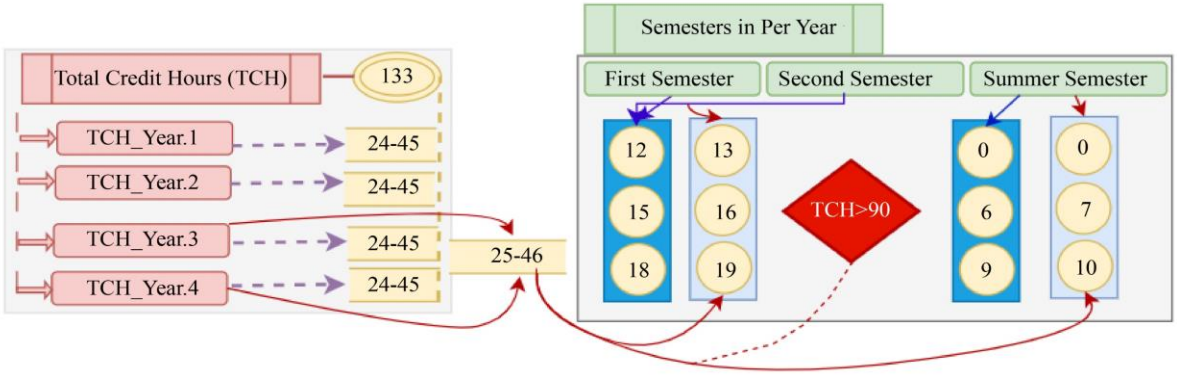

Fig. 3 Structure raw data

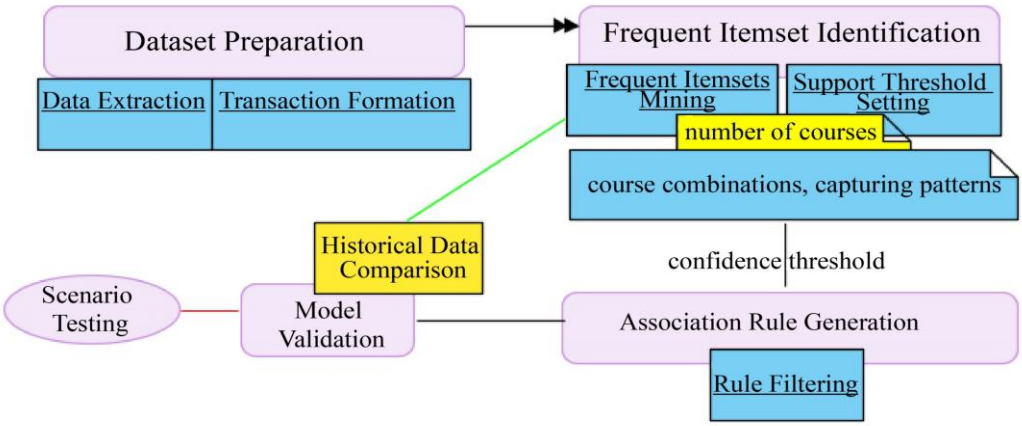

Fig. 4 The apriori algorithm

3.3. Reinforced Tire

The purpose of the Reinforced Tier is to generate patterns that encompass all scenarios within specified constraints. This involves applying the Apriori algorithm, which identifies frequent item sets using a bottom-up methodology. The Apriori algorithm was specifically applied to identify common patterns and associations among courses within the structured requirements of the program, the Apriori algorithm was specifically applied to identify common patterns and associations among courses within the structured requirements of the program. The figure below is the step-by-step implementation. Implementing the Apriori algorithm provided structured, data-driven insights, enhancing academic planning by aligning predictive pathways with the software engineering program's specific course requirements and constraints.

The process includes the following steps:

3.3.1. Initialization

Set the minimum support threshold based on the dataset, which includes 4 columns and 15 transactions. The highest frequency item set observed is 13.

$$frequent\ itemset = \frac{highest\ frequency\ itemset}{Transactions} \times 100\% \qquad (2)$$

From the frequent item set observed in Table 2, we calculate the frequency as follows: Out of 15 transactions, the highest frequent item set is approximately 86.7% (13 out of 15). To set the minimum support threshold, subtract this percentage from 100% to get the complementary percentage: 100% - 86.7% = 13.3%, converted to decimal form as 0.133. Thus, the minimum support threshold is 0.13. Thus, the minimum support threshold is 0.13. The minimum support count based on this threshold is shown in Equation (3):

$$min_{support_{count}} = total_{rows} \times minimum\ support\ threshold \qquad (3)$$

With a total of 16 rows and a minimum support threshold of 0.13, the minimum support count is 2.08.

3.3.2. Association Rules

These are established through a heuristic approach, as shown in Figure 1. The rules cover a variety of antecedent and consequent items involving combinations of 1, 2, and 3 items. Six distinct pairs are formed: (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), and (3, 1). The number of rules for these pairs is 4, 8, 4, 8, 4, and 4, respectively, indicating different levels of emphasis and occurrence across these pairs.

The predictive pathway approach using machine learning is a robust method for data reshaping, preserving meaningful correlations within the dataset. It ensures that crucial elements such as year and total credit hours are retained, which are essential for the task. When generating datasets, it is vital to follow predefined policies and constraints, especially those related to yearly segmentation and total credit hours. As demonstrated by the Elaborate Tier, this approach highlights the importance of these dataset components. Additionally, it aims to identify all feasible scenarios while eliminating duplications, a key aspect addressed within the Reinforced Tier framework. The significance of using the predictive pathway approach with machine learning is twofold. Firstly, it represents a pioneering implementation of machine learning in this domain, addressing significant challenges faced by students in the software engineering bachelor's program at Al-Zaytoonah University of Jordan. Secondly, its impact extends beyond scenario generation, finding applications in diverse fields such as Electronic Health Records (EHR), where it helps predict health statuses to prevent risks. Similarly, in meteorology, the precision of forecasting is crucial, emphasizing the need for accurate predictions rather than speculative ones, thus embracing the concept of "far-present" forecasting. The Reinforced Tier is pivotal in refining and validating the predictive models by generating comprehensive patterns and scenarios within defined constraints. By leveraging the Apriori algorithm and establishing robust association rules, this tier enhances the depth and reliability of our dataset analysis. This meticulous approach ensures that the predictive pathway model is not only accurate but also

adaptable to various applications beyond the educational domain. Moving to Section 4, we will establish the experimental setup for implementing this predictive pathway approach using machine learning. This setup will further demonstrate the practical applications and effectiveness of our model in real-world scenarios, paving the way for innovative solutions and improved academic planning.

Table 2. Total credit hours per year

Year	TCH.Min	TCH.Min+1	TCH.Min+2	TCH.Min	TCH.Min+3	TCH.Min+4	TCH.Min+5	TCH.Min+6	TCH.Min+7	TCH.Min+8	TCH.Min+9	TCH.Min+10	TCH.Min+11	TCH.Min+12	TCH.Min+13	TCH.Max
y.1	24	27	30	33	36	39	42	0	0	0	0	0	0	0	0	45
y.2	24	27	30	33	36	39	42	0	0	0	0	0	0	0	0	45
y.3	24	25	27	28	30	31	33	34	36	37	39	40	42	43	45	46
y.4	24	25	27	28	30	31	33	34	36	37	39	40	42	43	45	46

4. Experiment and Results

Data collection followed the structure outlined in the Elicit Dataset tier, utilizing instruments aligned with the correlation method. The primary target item, Total Credit Hours (TCH), shows a positive relationship with other variables, though its correlation with the year variable is relatively weak. The objective begins by focusing on the year variable, aiming to distribute a total of 133 credit hours over a span of four years. Table 2 presents the number of credit hours for each year. The first column shows the time span required to reach the target credit hours for the software engineering study plan. The subsequent 15 columns detail the properties of these credit hours.

Observations:

- For years 1 and 2 (y.1 and y.2), the values in the columns TCH.Min to TCH.Max predominantly ranges from 24 to 45 credit hours, with several zeros at the end. This indicates that each course typically weighs 3 credit hours, and both years total 90 credit hours, meeting the restriction that each course weighs 3 credit hours.
- For years 3, 4, and 5 (y.3, y.4, and y.5), the values across the time intervals vary more, starting from 24 and ending at 46, without zeros towards the end. This variation indicates the inclusion of courses weighing both 3 credit hours and 1 credit hour, achieving above 90 credit hours for these years.

Detailed data collection and analysis help understand the distribution and correlation of credit hours across the academic years, providing a solid foundation for further predictive modeling and academic planning. Table 2 presents the number of credit hours for each year. The first column shows the time span required to reach the target credit hours for the software engineering study plan. The subsequent columns detail the properties of these credit hours. Table 2 defines all possible combinations to reach the target value of 133 credit hours. When reaching 90 credit hours, the values allow one of these: [25, 28, 31, 34, 37, 40, 43, 46] in each combination. Figure 4 shows how to find all combinations of numbers from each row. The combined scenario depicted in Figure 5 outlines a systematic approach to identifying various parameters, each with its specific conditions. Table 2 provides raw data parameters sourced from two primary factors: year and credit hours. These factors exhibit distinct potentials for registration across different academic years.

I. Define the input data.
   1) Table.2 data
   2) Target value=133
   3) Normal values = [24,27,30,33,36,39,42,45]
   4) Allowed values = [25,28,31,34,37,41,43,46]
   5) Restriction value = 90

II. Restriction the based on sum up select values from rows.
   1) Less than 90 credit hours normal values [24,27,30,33,36,39,42,45]
   2) Equal or above 90 use allowed values only on time.
   3) Then return to normal values to reach target values =133

III. Find combinations.
IV. Iterate through each row in Table.1data
V. Is the sum of values in the row greater than or equal to 90.
VI. Iterate through each row's combinations.
VII. Print combinations for the row

Fig. 5 Combinations for years

Year 1 (y.1) and Year 2 (y.2): Both align with credit hour values ranging from 24 to 45. This consistency is due to the restriction that each course is valued at 3 credit hours, resulting in a total that must be less than 90 credit hours.

Year 3 (y.3) and Year 4 (y.4): These years correspond to credit hour values ranging from 24 to 46. The slight increase in the maximum value is due to the allowance for courses

valued at 1 credit hour, which can be included once the total credit hours reach or exceed 90.

This distinction between the years highlights the restrictive value parameters, where cumulative credit hours below 90 must adhere to the normal values [24, 27, 30, 33, 36, 39, 42, 45]. Once 90 credit hours are reached or exceeded, the allowable values [25, 28, 31, 34, 37, 40, 43, 46] can be used, providing flexibility in course selection.

To generate combinations for the dataset described in Table 2, adhering to the procedure outlined in Figure 5, Python can be used to cover all potential combinations. The following script demonstrates how to achieve this using Python's tkinter library for the graphical interface and itertools for generating combinations.

To derive feasible combinations of annual credit hours summing up to a total of 133, the dataset comprises several essential components crucial for analysis, as shown in Figure 6.

- Target Value: The goal of 133 credit hours.
- Normal Values: Credit hours typically range from 24 to 45.
- Restriction Values: Exceptions for credit hours exceeding 90 but limited to a single occurrence per term.
- Allowed Values: The permissible set of credit hours, which includes [25, 28, 31, 34, 37, 40, 43, 46].

These dataset attributes, as referenced in Table 1, are extensively utilized in the study to ensure comprehensive analysis and accurate results. The computational process integrates three pivotal functions:

1. Iteration Through Data: Extracting necessary information.
2. Validation of Sum Against Restriction: Ensuring compliance with criteria.
3. Employing itertools: Generating valid combinations adhering to predefined conditions.

The resulting combinations manifest six distinct attributes: the aggregate credit hours for each academic year, the cumulative credit hours across all years, and the temporal span covered by these years. In total, the script generates 156 combinations that meet the stipulated criteria. These combinations serve as comprehensive representations of feasible credit hour distributions, offering valuable insights for academic exploration and decision-making processes.

```
import tkinter as tk
from tkinter import ttk
from itertools import combinations
table_1_data = [
    [24, 27, 30, 33, 36, 39, 42, 0, 0, 0, 0, 0, 0, 0, 0, 45], [24, 27, 30, 33, 36, 39, 42, 0, 0, 0, 0, 0, 0, 0, 0, 45],
    [24, 25, 27, 28, 30, 31, 33, 34, 36, 37, 39, 40, 42, 43, 45, 46],[24, 25, 27, 28, 30, 31, 33, 34, 36, 37, 39, 40, 42, 43, 45, 46]]
target_value = 133
normal_values = [24, 27, 30, 33, 36, 39, 42, 45]
allowed_values = [25, 28, 31, 34, 37, 40, 43, 46]
restriction_value = 90
def find_combinations(data, target, normal, allowed, restriction):
    all_combinations = []
    for row in data:
        if sum(row) >= restriction:
            row_values = row[:]
            row_combinations = []
            for allowed_val in allowed:
                if allowed_val in row_values:
                    row_values.remove(allowed_val)
                    for r in range(1, len(row_values) + 1):
                        for subset in combinations(row_values, r):
                            if sum(subset) == target - allowed_val:

row_combinations.append(tuple(sorted(subset + (allowed_val,))))
            all_combinations.append(row_combinations)
        else:
            row_combinations = []
            for r in range(1, len(row) + 1):
                for subset in combinations(row, r):
                    if sum(subset) == target:
                        row_combinations.append(subset)
            all_combinations.append(row_combinations)
    return all_combinations
result_combinations = find_combinations(table_1_data, target_value, normal_values, allowed_values, restriction_value)
root = tk.Tk()
root.title("Combinations Tree")
tree = ttk.Treeview(root)
tree.pack()
for idx, combinations in enumerate(result_combinations, start=1):
    tree.insert("", tk.END, f"Row {idx}", text=f"Combinations for row y.{idx}")
    if any(combinations):
        for i, comb in enumerate(combinations, start=1):
            tree.insert(f"Row {idx}", tk.END, text=f"Combination {i}: {comb}")
    else:
        tree.insert(f"Row {idx}", tk.END, text="No combinations found for this row")
root.mainloop()
```

Fig. 6 Script of combinations for years

Table 3 categorizes the credit hour combinations based on their span time across years. The first group, spanning 3 years (Rows 1, 2, 3, and 4), comprises combinations with credit hours 42, 45, and 46 distributed across Years 1, 2, and 3. These combinations represent the shortest path to program completion. The second group, spanning 4 years, includes combinations with credit hours ranging from 24 to 46 across all four years (Rows 5 to 9). This group offers greater flexibility in scheduling but requires a longer study duration.

The output of combinations addresses several issues by considering two main factors: indexing and the location of values.

From Table 1, two dimensions are present: the horizontal dimension labeled "year" (span time) and another horizontal dimension labeled from TCH.Min to TCH.Max. These dimensions' result in a 4x16 matrix, with general credit hour values ranging from 24 to 46 spread across different locations.

For example, the credit hour values [24, 27, 30, 33, 36, 39, 42, 45] are held in four different locations, indicating four different addresses. Conversely, the credit hour values [25, 28, 31, 34, 37, 40, 43, 46] are held in two different locations, indicating two different addresses.

This distinction highlights how credit hours are distributed across the academic years, impacting the flexibility and duration of study plans. Below is an excerpt from Table 3 for illustration:

Table 3. Sample of combinations based on years and credit hours

Combination	Year.1	Year.2	Year.3	Year.4	Sum Value	Span Time	Groups
1	42	45	46	0	133	3	Group.1
2	42	45	46	0	133	3	
3	42	45	46	0	133	3	
4	42	45	46	0	133	3	
5	24	24	40	45	133	4	Group.2
6	24	27	36	45	133	4	
7	27	28	33	45	133	4	
8	27	30	33	43	133	4	
9	30	30	34	39	133	4	

This structured approach ensures that all potential combinations are considered, comprehensively analysing feasible credit hour distributions for effective academic planning.

The Index.Credit_Hour column in Table 4 illustrates the presence of each credit hour value across different years. For instance, a credit hour value 24 appears consistently across all years, denoted by 'T' values in Address.Year.1, Address.Year.2, Address.Year.3, and Address.Year.4. This uniform presence across all addresses signifies its existence every year.

Conversely, the credit hour value 25 is found only in Year.3 and Year.4, marked by 'T', while year.1 and Year.2 display 'F', indicating its absence in those years. Using the Index.Credit_Hour column and the address years, the Apriori algorithm can be applied to discover frequent item sets through a bottom-up approach.

The heuristic rule strategy starts by examining four distinct association rules involving singular antecedents and consequents:

1. Address.Year.2 -> Address.Year.1
2. Address.Year.1 -> Address.Year.2
3. Address.Year.4 -> Address.Year.3
4. Address.Year.3 -> Address.Year.4

These rules are subsequently classified into two cohesive groups: The first group encompasses relationships between Address.Year.1 and Address.Year.2. The second group pertains to associations between Address.Year.3 and Address.Year.4.

The rules' index values are segregated into two sets, as illustrated in Figure 6. For example, examining index value 27 reveals four diverse address combinations: (1,2), (2,2), (3,3), and (4,3). Conversely, index value 28 demonstrates two distinct addresses: (3,4) and (4,4), which are not linked to Year.1 and Year.2. This example highlights the permutations of indices through various address relationships.

The presence of singular antecedents and consequents holds significant implications in adjusting patterns. For instance, considering index value 27, if we focus on addresses, there is a requirement to interchange the values of four addresses.

This necessitates using a 'swap' structure employing pointer (*) and reference (&) methods to explicitly illustrate the specific relationship. Similarly, index value 28 also involves the utilization of pointers and references. However, an additional method, such as a 'switch' mechanism, becomes necessary to prevent the absence of Year.1 and Year.2 in this context.

The illustrated pattern in Figure 6 outlines the generation of thirty-two rules, categorized based on the arrangement of antecedents and consequents. These rules are derived by examining the index values associated with each rule and assessing the presence and significance of each index value within every rule. The support measure is employed to

determine the importance of an index value. The fundamental equation used for this assessment is:

$$\text{Support (antecedents-consequents)} = \frac{\textit{Transactions containing both antecedents and consequents}}{\textit{Total Transactions}} \quad (6)$$

The support value, which ranges from 0 to 1, denotes the strength of the association within frequent item sets. Higher support values, nearing one, signify a stronger and more meaningful relationship between the items. Conversely, lower support values, nearing zero, indicate weaker associations or less frequent itemset relationships.

Table 4. Possible location factor for index credit hour

Address. Year.1	Address. Year.2	Address. Year.3	Address. Year.4	Index.Credit_Hour	Total_credit_hour/ Index.Credit_Hour	Total_credit_hour/ Index.Credit_Hour-4	Total_credit_hour/ Index.Credit_Hour-4*24
T	T	T	T	24	5.541666667	1.541666667	37
F	F	T	T	25	5.32	1.32	33
T	T	T	T	27	4.925925926	0.925925926	25
T	T	F	F	28	4.75	0.75	21
T	T	T	T	30	4.433333333	0.433333333	13
F	F	T	T	31	4.290322581	0.290322581	9
T	T	T	T	33	4.03030303	0.03030303	1
F	F	T	T	34	3.911764706	-0.088235294	-3
T	T	T	T	36	3.694444444	-0.305555556	-11
F	F	T	T	37	3.594594595	-0.405405405	-15
F	F	T	T	39	3.41025641	-0.58974359	-23
F	F	T	T	40	3.325	-0.675	-27
T	T	T	T	42	3.166666667	-0.833333333	-35
F	F	T	T	44	3.022727273	-0.977272727	-43
T	T	T	T	45	2.955555556	-1.044444444	-47
F	F	T	T	46	2.891304348	-1.108695652	-51

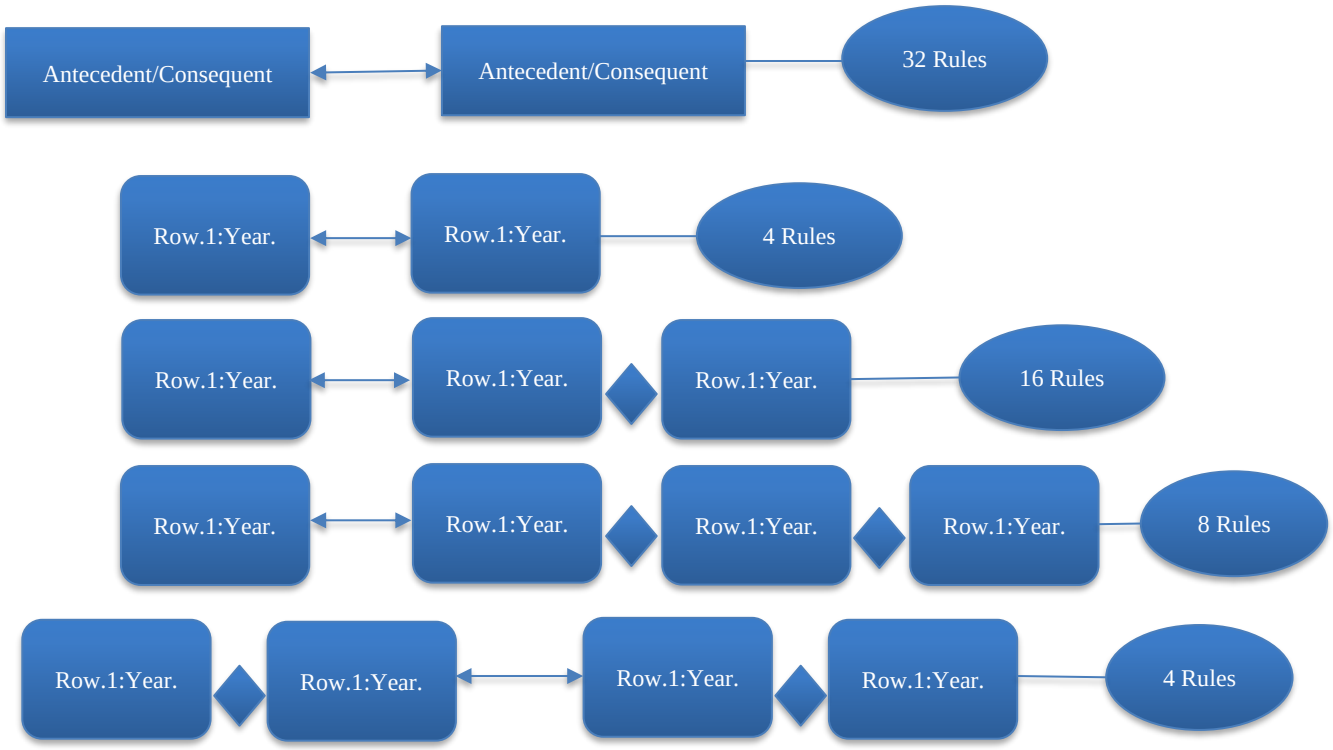

Fig. 7 Pattern to generalize rules

By analyzing these support values, we can effectively determine the significance of each rule and the strength of the associations they represent. This method provides valuable insights into the data, helping to identify the most relevant and impactful patterns.

By analyzing these support values, we can effectively determine the significance of each rule and the strength of the associations they represent. This method provides valuable insights into the data, helping to identify the most relevant and impactful patterns.

Table 5. Support for antecedents and consequents

Antecedents	Consequents	Antecedent Support	Consequent Support	Support
frozenset({'Address.Year.4'})	frozenset({'Address.Year.3'})	0.9375	0.9375	**0.9375**
frozenset({'Address.Year.3'})	frozenset({'Address.Year.4'})	0.9375	0.9375	**0.9375**
frozenset({'Address.Year.2'})	frozenset({'Address.Year.1'})	0.5	0.5	**0.5**
frozenset({'Address.Year.1'})	frozenset({'Address.Year.2'})	0.5	0.5	**0.5**
frozenset({'Address.Year.4'})	frozenset({'Address.Year.3', 'Address.Year.2', 'Address.Year.1'})	0.9375	0.4375	**0.4375**
frozenset({'Address.Year.2'})	frozenset({'Address.Year.3', 'Address.Year.4', 'Address.Year.1'})	0.5	0.4375	**0.4375**
frozenset({'Address.Year.3'})	frozenset({'Address.Year.2', 'Address.Year.4', 'Address.Year.1'})	0.9375	0.4375	**0.4375**
frozenset({'Address.Year.4', 'Address.Year.1'})	frozenset({'Address.Year.3', 'Address.Year.2'})	0.4375	0.4375	**0.4375**
frozenset({'Address.Year.3'})	frozenset({'Address.Year.2', 'Address.Year.4'})	0.9375	0.4375	**0.4375**
frozenset({'Address.Year.3', 'Address.Year.1'})	frozenset({'Address.Year.2', 'Address.Year.4'})	0.4375	0.4375	**0.4375**
frozenset({'Address.Year.3', 'Address.Year.2'})	frozenset({'Address.Year.4', 'Address.Year.1'})	0.4375	0.4375	**0.4375**
frozenset({'Address.Year.2', 'Address.Year.4', 'Address.Year.1'})	frozenset({'Address.Year.3'})	0.4375	0.9375	**0.4375**
frozenset({'Address.Year.3', 'Address.Year.4', 'Address.Year.1'})	frozenset({'Address.Year.2'})	0.4375	0.5	**0.4375**
frozenset({'Address.Year.3', 'Address.Year.2', 'Address.Year.1'})	frozenset({'Address.Year.4'})	0.4375	0.9375	**0.4375**
frozenset({'Address.Year.3', 'Address.Year.2', 'Address.Year.4'})	frozenset({'Address.Year.1'})	0.4375	0.5	**0.4375**
frozenset({'Address.Year.4'})	frozenset({'Address.Year.3', 'Address.Year.2'})	0.9375	0.4375	**0.4375**
frozenset({'Address.Year.2', 'Address.Year.4'})	frozenset({'Address.Year.3', 'Address.Year.1'})	0.4375	0.4375	**0.4375**
frozenset({'Address.Year.3', 'Address.Year.2'})	frozenset({'Address.Year.4'})	0.4375	0.9375	**0.4375**
frozenset({'Address.Year.2', 'Address.Year.4'})	frozenset({'Address.Year.3'})	0.4375	0.9375	**0.4375**
frozenset({'Address.Year.4'})	frozenset({'Address.Year.3', 'Address.Year.1'})	0.9375	0.4375	**0.4375**
frozenset({'Address.Year.3'})	frozenset({'Address.Year.4', 'Address.Year.1'})	0.9375	0.4375	**0.4375**
frozenset({'Address.Year.4', 'Address.Year.1'})	frozenset({'Address.Year.3'})	0.4375	0.9375	**0.4375**
frozenset({'Address.Year.3', 'Address.Year.1'})	frozenset({'Address.Year.4'})	0.4375	0.9375	**0.4375**

frozenset({'Address.Year.1'})	frozenset({'Address.Year.2', 'Address.Year.4'})	0.5	0.4375	**0.4375**
frozenset({'Address.Year.2'})	frozenset({'Address.Year.4', 'Address.Year.1'})	0.5	0.4375	**0.4375**
frozenset({'Address.Year.4', 'Address.Year.1'})	frozenset({'Address.Year.2'})	0.4375	0.5	**0.4375**
frozenset({'Address.Year.2', 'Address.Year.4'})	frozenset({'Address.Year.1'})	0.4375	0.5	**0.4375**
frozenset({'Address.Year.1'})	frozenset({'Address.Year.3', 'Address.Year.2'})	0.5	0.4375	**0.4375**
frozenset({'Address.Year.2'})	frozenset({'Address.Year.3', 'Address.Year.1'})	0.5	0.4375	**0.4375**
frozenset({'Address.Year.3', 'Address.Year.1'})	frozenset({'Address.Year.2'})	0.4375	0.5	**0.4375**
frozenset({'Address.Year.3', 'Address.Year.2'})	frozenset({'Address.Year.1'})	0.4375	0.5	**0.4375**
frozenset({'Address.Year.1'})	frozenset({'Address.Year.3', 'Address.Year.2', 'Address.Year.4'})	0.5	0.4375	**0.4375**

4.1. Support Analysis of Association Rules

The support analysis unveils distinctive patterns within the association rules, as shown in Table 5. Rules exhibiting high support values, such as:

{Address.Year.4}->{Address.Year.3}and {Address.Year.3} -> {Address.Year.4} (with support values of 0.9375), frequently co-occur in transactions. Conversely, rules with lower support values, such as:

{Address.Year.2}->{Address.Year.1}and {Address.Year.1} -> {Address.Year.2}, exhibit less frequent co-occurrences.

4.2. Pattern Analysis

1. Symmetric Associations: Certain rules demonstrate symmetric associations. For instance, {Address.Year.1}->{Address.Year.2} and {Address.Year.2} -> {Address.Year.1} both have equal support values of 0.5.
2. Complex Rules: More complex rules, such as {Address.Year.2,Address.Year.4}-> {Address.Year.3,Address.Year.1}and {Address.Year.3,Address.Year.1}-> {Address.Year.4}, also display substantial support at 0.4375.

4.3. Complex Relationships

The presence of rules with multiple items in both antecedents and consequents signifies intricate relationships. These rules suggest that specific combinations of address years are more likely to co-occur, unveiling diverse associations and dependencies among various combinations of address years. The support values for each rule, which include swapped antecedents and consequents, are equal. This often results in duplicates, especially when building patterns for the entire dataset. The support values for these rules are 0.4375, 0.5, and 0.9375, indicating that the frequent item sets are not covered in all rules, which should ideally have a support value equal to 1.

4.4. Pattern Analysis Sequence

- Select the index credit hour (designated as 'ict') from column 5.
- Subdivide the total value 133 based on the 'ict' values, providing the total credit hours indicated in column 6.
- Extract relevant data output from column 4, representing a span of 4 years.
- Utilize the calculated index credit hour results from the previous step in the analysis.

4.5. Association Rules and Support Score Thresholds

The utilization of association rules is constrained by predefined support score thresholds (0.4375, 0.5, and 0.9375), which highlight the need for exploring alternative methodologies to refine these thresholds and enhance accuracy in impact assessment. The fluctuations in support scores among antecedents, consequents, and their relationships underscore the necessity for a resilient mechanism to systematically address these considerations across diverse scenarios.

4.6. Scenarios

- Scenario 1: A stringent support score threshold of 0.9375 captures highly reliable patterns within the dataset. This threshold's robustness is evident in the consistently high support scores observed for both antecedents and consequents. From a domain knowledge perspective, it isolates the most salient patterns, aligning with an expert understanding of the domain. This approach results in dataset segmentation into two distinct yet highly relevant groups characterized by the presence of either Year.3 or Year.4 within both antecedents and consequents.

- Scenario 2: Moderately supported rules, identified with a threshold of 0.5, reveal patterns within the dataset that merit cautious interpretation. While not as robust as those with the 0.9375 threshold, they offer potentially valuable insights. These rules consistently involve Year.1 and Year.2 as either antecedents or consequents, suggesting a potential relationship warranting further exploration.

Using association rules guided by support score thresholds is instrumental in leveraging domain knowledge. Scenario 1 highlights the effectiveness of a stringent threshold in capturing highly reliable patterns, while Scenario 2 underscores the importance of cautious interpretation for moderately supported rules. Integrating domain knowledge into association rule generation and defined restrictions enhances the creation of meaningful combinations and contributes to a comprehensive understanding of the dataset. The experiment and results section has delivered a comprehensive dataset analysis, employing support measures and association rule mining to reveal significant patterns. By applying the Apriori algorithm and evaluating both high and moderate support thresholds, we identified critical relationships and dependencies among credit hours across different academic years. These findings affirm the predictive pathway approach's robustness while pointing to areas for further investigation. Integrating domain knowledge has been crucial in enhancing the analysis and ensuring the results' relevance and accuracy. Moving forward, these insights will provide a solid foundation for optimizing academic planning and decision-making processes, leading to more informed and effective educational strategies. To provide a comprehensive understanding, we compare our results with related works that have explored similar domains using various methodologies, as shown in Table 6. The comparison focuses on key aspects such as accuracy, methodology, computational complexity, and practical applicability.

Table 6. Comparative analysis of PPALML with related works

Aspect	PPALML (Our Study)	Bakri et al. [10]	Shete et al. [11]	Althunibat et al. [24]
Objective	Predict academic pathways for software engineering students	Predict students' graduation on time using machine learning algorithms	Track and predict student performance in degree programs using various ML algorithms	Evaluate the learning experience of students using the Learning Management System (Moodle)
Methodology	Apriori algorithm, support measure, correlation analysis	Random Forest, SVM, K-Nearest Neighbors, Naïve Bayes	Decision Trees, Random Forests, Neural Networks	Classification and clustering techniques
Accuracy	High accuracy in identifying feasible academic scenarios	Validated using cross-validation, high predictive accuracy	High accuracy in tracking and predicting student performance	Effective in evaluating and improving the learning experience
Computational Complexity	Moderate, due to the iterative nature of the Apriori algorithm	High requires extensive parameter tuning and cross-validation	High involves multiple algorithms and extensive computational resources	Moderate, depending on the complexity of classification and clustering tasks
Practical Applicability	High, tailored to specific academic requirements of software engineering students	High, applicable for predicting graduation outcomes in higher education	High, useful for academic planning and performance tracking in various degree programs	Moderate, focused on the learning experience within a specific LMS

The PPALML is specifically designed to predict academic pathways for software engineering students, leveraging the Apriori algorithm to identify frequent item sets and correlation analysis to validate the results. [10] use Random Forest, SVM, K-Nearest Neighbors, and Naïve Bayes to predict student graduation outcomes. [11] employ Decision Trees, Random Forests, and Neural Networks to track and predict student performance, while [24] use classification and clustering techniques to evaluate the learning experience within an LMS.

4.7. Objective and Methodology

The PPALML is specifically designed to predict academic pathways for software engineering students, leveraging the Apriori algorithm to identify frequent itemsets and correlation analysis to validate the results. [10] use Random Forest, SVM, K-Nearest Neighbors, and Naïve Bayes to predict student graduation outcomes. [11] employ Decision Trees, Random Forests, and Neural Networks to track and predict student performance, while [24] use

classification and clustering techniques to evaluate the learning experience within an LMS.

4.8. Accuracy

Our study demonstrates high accuracy in identifying feasible academic scenarios, similar to the high predictive accuracy reported by [10, 11, 24], and also reports effective evaluation and improvement in the learning experience, although their focus differs from ours.

4.9. Computational Complexity

The PPALML approach has moderate computational complexity due to the iterative nature of the Apriori algorithm. In contrast, [10, 11] require extensive parameter tuning and cross-validation, making their methodologies more computationally intensive. [24] present a moderate level of complexity, depending on the classification and clustering tasks.

4.10. Practical Applicability

The PPALML approach is highly applicable to academic planning, tailored to the specific requirements of software engineering students [10, 11], and offers methodologies applicable to predicting graduation outcomes and tracking student performance, respectively. [24] focus on the learning experience within an LMS, providing moderate practical applicability. The PPALML approach distinguishes itself through its tailored focus on academic planning for software engineering students, high accuracy, moderate computational complexity, and strong practical applicability. The comparative analysis highlights our methodology's strengths and unique features, showcasing its effectiveness in optimizing academic pathways. Future research should continue to refine and expand the PPALML approach to enhance its applicability across different academic disciplines and educational settings.

5. Conclusion

The results of our study highlight the significant contributions and importance of utilizing machine learning techniques, specifically the PPALML, in optimizing academic planning for the software engineering bachelor's program at Al-Zaytoonah University of Jordan. The PPALML framework, through the use of the Apriori algorithm and support measures, has proven highly effective in identifying and generating meaningful academic scenarios that align with credit hour requirements and enhance decision-making processes.

Our study lays the groundwork for a novel approach to academic planning and advising, leveraging advanced machine learning techniques to provide tailored and accurate pathway predictions. This approach ensures that students meet academic requirements and assists academic advisors in making informed decisions that can profoundly impact students' educational trajectories.

The current study was confined to the software engineering program at Al-Zaytoonah University of Jordan, focusing on specific credit hour distributions and academic requirements. Future research should aim to extend the applicability of the PPALML framework to other academic programs and institutions, refining and expanding its utility across various educational settings.

These results represent a significant first step towards integrating machine learning into academic planning, demonstrating the potential for improved accuracy and efficiency in managing academic pathways. The insights gained from this study underscore the value of data-driven decision-making and the integration of domain knowledge in enhancing the relevance and applicability of predictive models. This study forms the backbone for future research aimed at further optimizing academic planning and decision-making processes. By building on the PPALML framework, researchers and educators can continue to develop more sophisticated and adaptable models, ultimately contributing to better educational outcomes and more efficient academic administration.

Future work will involve applying the PPALML framework across different academic disciplines and institutions to validate its effectiveness in diverse contexts. Additionally, exploring the integration of more advanced machine learning algorithms, developing user-friendly interfaces, and conducting longitudinal studies will be crucial next steps. Collaboration with policymakers and other educational institutions will also be essential to ensure the broad adoption and continuous improvement of predictive models in academic planning.

In conclusion, the PPALML approach offers a promising pathway to revolutionize academic planning by providing accurate, data-driven insights and personalized recommendations. The framework's adaptability to various educational contexts makes it a versatile tool for enhancing the academic planning process, ensuring students' success, and improving institutional efficiency. As we continue to refine and expand this approach, its potential impact on the educational landscape becomes increasingly significant, paving the way for more informed and effective educational strategies.

Acknowledgments

We would like to express our deepest gratitude to everyone who contributed to the success of this project, titled "Development of an Intelligent System for Electronic Document Management Based on Machine Learning Techniques (Case Study-Student File Archiving System at Al-Zaytoonah University)", Project Number: 37 / 17 / 2023-2022. First and foremost, we thank Al-Zaytoonah University for providing us with the opportunity and the necessary resources

to undertake this project. The support from the administration and the IT department has been invaluable in every step of our research and development process. We also acknowledge the contributions of our colleagues and fellow researchers who provided their insights and assistance, helping us navigate challenges and refine our approach. Special thanks to the members of the IT department for their technical support and for facilitating access to essential data and tools. Furthermore, we extend our appreciation to the students and staff at Al-Zaytoonah University, who participated in the case study and provided valuable feedback, which was instrumental in enhancing the practical aspects of our system. This project would not have been possible without the collective efforts and support of all these individuals and organizations. Thank you for your contributions and commitment to advancing the field of intelligent document management systems.

References

[1] Ahmad Almufarreha, and Muhammad Arshad, "Promising Emerging Technologies for Teaching and Learning: Recent Developments and Future Challenges," *Sustainability*, vol. 15, no. 8, pp. 1-21, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[2] Bilal Hawashin et al., "An Efficient Hybrid Similarity Measure Based on User Interests for Recommender Systems," *Expert Systems*, vol. 37, no. 5, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[3] Nouhaila El Koufi, and Abdessamad Belangour, "Toward a Responsive System for Precision Marketing Based on RFM Model, Deep learning and Features Importance Ranking: A Case Study of Morocco," *International Journal of Advances in Soft Computing and its Applications*, vol. 16, no. 2, pp. 19-29, 2024. [Google Scholar] [Publisher Link]

[4] Wei Jin, "Research on Machine Learning and its Algorithms and Development," *Journal of Physics: Conference Series*, *5th International Conference on Intelligent Computing and Signal Processing*, Suzhou, China, vol. 1544, pp. 1-6, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[5] Iqbal H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," *SN Computer Science*, vol. 2, pp. 1-21, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Huan-Bin Wang, and Yang-Jun Gao, "Research on Parallelization of Apriori Algorithm in Association Rule Mining," *Procedia Computer Science*, vol. 183, pp. 641-647, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[7] Hussan Munir, Bahtijar Vogel, and Andreas Jacobsson, "Artificial Intelligence and Machine Learning Approaches in Digital Education: A Systematic Revision," *Information*, vol. 13, no. 4, pp. 1-26, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis, "Explainable AI: A Review of Machine Learning Interpretability Methods," *Entropy*, vol. 23, no. 1, pp. 1-45, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[9] Bingqing Yang, "An Empirical Study on the Application of Machine Learning for Higher Education and Social Service," *Journal of Global Information* Management, vol. 30, no. 7, pp. 1-16, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[10] Rizal Bakri, Niken Probondani Astuti, and Ansari Saleh Ahmar, "Machine Learning Algorithms with Parameter Tuning to Predict Students' Graduation-on-Time: A Case Study in Higher Education," *Journal of Applied Science, Engineering, Technology, and Education*, vol. 4, no. 2, pp. 259-265, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[11] Mohini Shete et al., "Tracking and Predicting Student Performance in Degree Programs Using Machine Learning," *2022 5th International Conference on Contemporary Computing and Informatics*, Uttar Pradesh, India, pp. 1949-1954, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[12] Cristiano Mauro Assis Gomes, and Enio Jelihovschi, "Presenting the Regression Tree Method and its Application in a Large-Scale Educational Dataset," *International Journal of Research & Method in Education*, vol. 43, no. 2, pp. 201-221, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[13] Harman Preet Singh, and Hilal Nafil Alhulail, "Predicting Student-Teachers Dropout Risk and Early Identification: A Four-Step Logistic Regression Approach," *IEEE Access*, vol. 10, pp. 6470-6482, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[14] Neema Mduma, "Data Balancing Techniques for Predicting Student Dropout Using Machine Learning," *Data*, vol. 8, no. 3, pp. 1-14, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] Esmael Ahmed, "Student Performance Prediction Using Machine Learning Algorithms," *Applied Computational Intelligence and Soft Computing*, vol. 2024, no. 1, pp. 1-15, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[16] John T. Avella et al., "Learning Analytics Methods, Benefits, and Challenges in Higher Education: A Systematic Literature Review," *Online Learning*, vol. 20, no. 2, pp. 13-29, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[17] Shams Forruque Ahmed et al., "Deep Learning Modelling Techniques: Current Progress, Applications, Advantages, and Challenges," *Artificial Intelligence Review*, vol. 56, pp. 13521-13617, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] David Baneres, M. Elena Rodríguez-Gonzalez, and Montse Serra, "An Early Feedback Prediction System for Learners At-Risk Within a First-Year Higher Education Course," *IEEE Transactions on Learning Technologies*, vol. 12, no. 2, pp. 249-263, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[19] Thao-Trang Huynh-Cam, Long-Sheng Chen, and Huynh Le, "Using Decision Trees and Random Forest Algorithms to Predict and Determine Factors Contributing to First-Year University Students' Learning Performance," *Algorithms*, vol. 14, no. 11, pp. 1-17, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[20] Ihsan A. Abu Amra, and Ashraf Y.A. Maghari, "Students Performance Prediction Using KNN and Naïve Bayesian," *2017 8th International Conference on Information Technology*, Amman, Jordan, pp. 909-913, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[21] "*Study Plan for Bachelor Program - Study Plan Development and Updating Procedures/ Software Engineering Department*," Al-Zaytoonah University of Jordan, Faculty of Science and IT, pp. 1-4, 2022. [Publisher Link]

[22] Mohammad Hasan Altarawneh et al., "The Relationship between Cross-Cutting Factors and Knowledge, Learning Outcomes, and Skills in Dual Degree Programs," *Journal of Theoretical and Applied Information Technology*, vol. 102, no. 8, pp. 3410-3422, 2024. [Google Scholar] [Publisher Link]

[23] Ya-Han Hu, and Yen-Liang Chen, "Mining Association Rules with Multiple Minimum Supports: A New Mining Algorithm and a Support Tuning Mechanism," *Decision Support Systems*, vol. 42, no. 1, pp. 1-24, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[24] Ahmad Althunibat et al., "Learning Experience of Students Using the Learning Management System: User's Perspective on the Use of Moodle in the University of Jordan," *Advances in Human-Computer Interaction*, vol. 2023, no. 1, pp. 1-11, 2023. [CrossRef] [Google Scholar] [Publisher Link]