Redefining Prerequisites Through Text Embeddings: Identifying Practical Course Dependencies

Şükrü Kaan Tetik, Emirhan Toprak, Senem Kumova Metin[a] and Hande Aka Uymaz[b]
İzmir University of Economics, Department of Software Engineering, İzmir, Turkey

Keywords: Software Engineering Education, Course Prerequisites, Embedding Models, Machine Learning, SHAP.

Abstract: This study proposes a framework to support undergraduate students in course selection by identifying implicit prerequisites and predicting performance in elective courses. Unlike traditional prerequisite rules that rely solely on curriculum design, our approach integrates students' academic history and course-level semantic information. We define two core tasks: (T1) identifying practical prerequisites that significantly impact success in a target course, and (T2) predicting student success in elective courses based on academic profiles. For T1, we analyze prior course performance and learning outcomes using SHAP (SHapley Additive exPlanations) to determine the most influential courses. For T2, we build student representations using course descriptions and learning outcomes, then apply embedding models (Sentence-BERT, Doc2Vec, Universal Sentence Encoder) combined with classification algorithms to predict course success. Experiments demonstrate that embedding-based models, especially those using Sentence-BERT, can effectively predict course outcomes. The results suggest that incorporating semantic representations enhances curriculum design, course advisement, and prerequisite refinement.

1 INTRODUCTION

In university education, selecting the right courses at the right time is a critical decision stage that may have several effects on the student's academic journey, which requires careful consideration. Although students are required to take certain compulsory courses within their department programs, they also have the opportunity to take elective courses that allow them to either diversify their competencies or specialize in particular areas. For instance, in the departments such as computer and software engineering, the practical skills together with the theoretical skills affect the success of the further courses. These course selection decisions can significantly influence not only students' academic performance and future course choices but also the competencies they will have acquired by the time of graduation. Typically, universities define course enrollment rules based on factors such as a student's current academic level, whether a prerequisite course has been successfully completed, or credit thresholds. However, student success is not solely determined by these explicit institutional rules; it also depends on personal knowledge, skills, competencies, and prior performance in specific courses or course groups. Making course selections based solely on general academic criteria may negatively impact students' academic performance, reduce their GPA, or misguide their long-term academic planning.

In the literature, the problem of guiding students in course selection has often been approached through the adaptation of recommendation system techniques, machine learning methods, and hybrid frameworks (Atalla et al., 2023; Zhu and Wang, 2022; Esteban et al., 2020). These systems typically rely on either students' historical course data or patterns identified from similar student profiles.

This study aims to improve course performance among undergraduate students in the field of software engineering by providing practical prerequisites for achieving success in a given course and presenting the competencies that a student should possess before enrolling in a course. In line with this primary objective, two specific tasks (T) were defined to facilitate the development of solutions through different approaches.

T1. To identify the implicit or practical prerequisites, beyond the formally defined institutional requirements, that contribute to student success in a given course.

[a] https://orcid.org/0000-0002-9606-3625
[b] https://orcid.org/0000-0002-3535-3696

Tetik, Ş. K., Toprak, E., Kumova Metin, S. and Aka Uymaz, H.
Redefining Prerequisites Through Text Embeddings: Identifying Practical Course Dependencies.
DOI: 10.5220/0013682800004000
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 17th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2025) - Volume 1: KDIR*, pages 49-59
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

For each course offered by an educational institution, a set of prerequisites, such as success in a specific course or group of courses and attendance requirements, is defined within the framework of the existing curriculum. A student who meets these prerequisites is allowed to enroll in the corresponding course. This first task aims to investigate how these prerequisites are formed and applied in practice. The outcome of the task may be used to update/extend the prerequisites, considering the practical results of the current system.

T2. To evaluate the extent to which a student's success in an elective course can be predicted based on their existing academic profile. This task focuses on predicting whether a student will succeed, or to what extent they will succeed, in a course they plan to take, based on their existing competencies. The proposed prediction system has the potential to assist students in evaluating whether they meet the course requirements and to support more informed and confident course enrollment decisions.

To address the tasks defined in this study, we proposed the following methodology. First, to identify implicit prerequisites that contribute to student success (T1), we developed two modeling approaches: one based on students' past course performances and another based on the learning outcomes of the completed courses. In both cases, we represented student profiles using vector-based representations and applied SHAP (SHapley Additive exPlanations) (Lundberg and Lee, 2017) to interpret which prior courses or learning outcomes had the greatest impact on success in a target course. Second, to evaluate and predict student success in elective courses (T2), we constructed student profiles using course descriptions and learning outcomes, combined with students' letter grades. We employed several embedding models, including Doc2Vec (Le and Mikolov, 2014), SBERT (Reimers and Gurevych, 2019), and the Universal Sentence Encoder to transform this textual data into feature vectors, which were then used to train classification models. The classification performance was evaluated using cross-validation and F1 scores across multiple algorithms.

The remainder of this paper is structured as follows: Section 2 provides a review of the relevant literature and background. Section 3 describes the dataset utilized in this study. Section 4 presents the proposed methodology for identifying practical prerequisites and constructing predictive student profiles. Section 5 details the experimental setup and discusses the results obtained. Finally, Section 6 concludes the paper.

2 LITERATURE REVIEW

In this section, we review relevant literature and background concepts related to our approach. First, we present essential academic terms that are frequently referenced in this context. Then, we examine related work in the domain of course recommendation systems and prerequisite discovery, focusing on techniques such as semantic similarity, and the application of large language models (LLMs).

2.1 Background

This subsection introduces several terms frequently used both in this paper and in related literature reviews, such as *syllabus*, *transcript*, and *grade point average*. A *syllabus* is a document prepared by instructors that outlines the goals of a course, weekly topics, required materials, learning outcomes, grading policies, and credit information. It acts like a roadmap for both instructors and students during the semester. In general, *learning outcomes* are presented in the syllabus, which explains in simple and clear terms what a student should be able to do, understand, or apply after they complete the course. A *transcript* is an official academic record that lists all courses a student has taken, with the corresponding letter grades and credit information. It is a comprehensive document summarizing a student's academic performance over semesters. The *Grade Point Average (GPA)*, which also appears on the transcript, is a numerical measure of the general academic performance of a student. It is calculated by taking the average of grade points corresponding to letter grades, weighted by course credits. GPA is widely used to assess a student's academic standing and to make decisions about graduation or honors.

2.2 Related Work

In the literature, one of the course recommendation systems was proposed by Atalla et.al, which presents a data-driven framework for guiding students in course selection (Atalla et al., 2023). The authors propose a system that combines curriculum dependency analysis with student performance modeling to assist academic advising. Their methodology involves constructing a Course Dependency Graph (CDG) to capture prerequisite relationships and curriculum flow, and then applying matrix factorization techniques to model students' performance patterns based on historical grade data. This combination allows the system to recommend courses that are both pedagogically appropriate and aligned with a stu-

dent's academic profile.

Anh et. al. proposed a course recommendation model that emphasizes the use of learning outcomes as the core representation of both student profiles and course content (Anh et al., 2021). In their approach, each course is described by a set of learning outcomes, and student profiles are built based on the learning outcomes of previously completed courses. To quantify the similarity between a student and a potential future course, the authors employ semantic similarity measures, comparing the student's acquired learning outcomes with those required by upcoming courses. This allows the system to recommend courses that align well with a student's current competencies. Their model demonstrates that learning outcome-based representations can offer a more meaningful and educationally relevant basis for course recommendation than relying solely on course names or historical grades.

Van Deventer et al present a novel course recommendation system that leverages LLMs to interpret students' natural language queries (Deventer et al., 2024). By employing a Retrieval-Augmented Generation (RAG) framework, the system generates a course description based on the user's input. Then they embedded this description into a vector space and compared it with existing course descriptions to identify the most semantically similar courses. The study demonstrates the potential of LLMs in capturing nuanced student interests and providing personalized course recommendations.

Aytekin and Saygın propose a novel approach for detecting prerequisite relations between educational concepts using fine-tuned large LLMs which are GPT-3 (Brown et al., 2020) and LLAMA2 (Touvron et al., 2023) (Aytekin and and, 2025). Their method formulates the task as a binary classification problem and trains LLMs with custom prompts and completion strings that include both the classification and an explanatory justification. According to their results, the fine-tuned models not only achieve state-of-the-art performance across several benchmark datasets but also generate human-comparable explanations.

3 DATASET

In this study, two main datasets were utilized to develop and evaluate the proposed models: one comprising course-related textual content and the other consisting of anonymized academic records of students. These datasets are essential in capturing both the structural and semantic aspects of university courses as well as students' historical academic performance. By combining these two data sources, we aimed to build a comprehensive foundation for modeling student profiles and understanding the implicit dynamics influencing course success. The subsections below describe the datasets and preprocessing steps in further detail.

3.1 Course Information Dataset

To obtain the course descriptions and learning outcomes for the transcript dataset, relevant information was collected from the official departmental web pages of İzmir University of Economics. A total of 1,654 course entries were gathered.

3.2 Transcript Dataset

The transcript dataset is constructed from the academic records of graduates of İzmir University of Economics (IUE). The raw transcript data required preprocessing, as it included records spanning the past 20 years. This meant that some courses were no longer offered and had no accessible information available. Additionally, course selection rules and restrictions have changed over time.

The dataset originally included 1,313 unique students, 1,017 distinct courses, and 10 unique grade scores.

As a first step, outlier data, such as students who had taken courses from the Food Engineering department, were removed. Then, using the course information collected from department websites, outdated or currently unavailable courses were identified and matched with their updated versions, if available. Courses that are too old or irrelevant to the current curriculum were eliminated. The cleaned and refined dataset was then used for all subsequent processes.

In the transcript dataset, students' performance in each course is represented by a letter grade. Table 1 shows these letter grades together with corresponding point intervals, coefficients, and academic status indicators. Accordingly, a profile is maintained for each student, consisting of course–letter grade pairs.

The dataset contains 1,307 different graduated students' anonymised transcript information from 2003 to 2025, obtained from software and computer engineering students of IUE. To visualize student performance, the average grade for each year is calculated. As shown in Figures 1 and 2, the overall average grade value is approximately 2.5.

The dataset includes both the elective and the mandatory courses. There are 912 distinct elective courses, which are grouped into five categories: game, software, artificial intelligence and machine

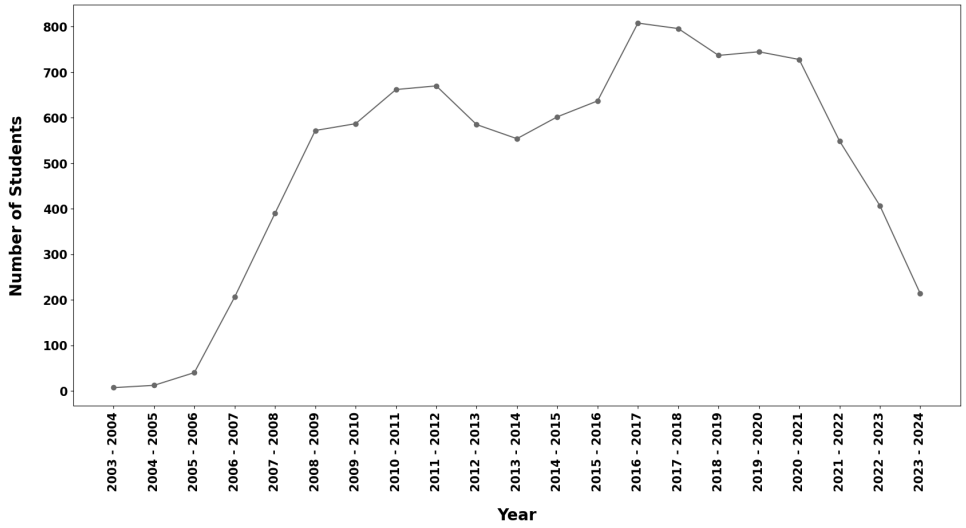

Figure 1: Number of students over the years.

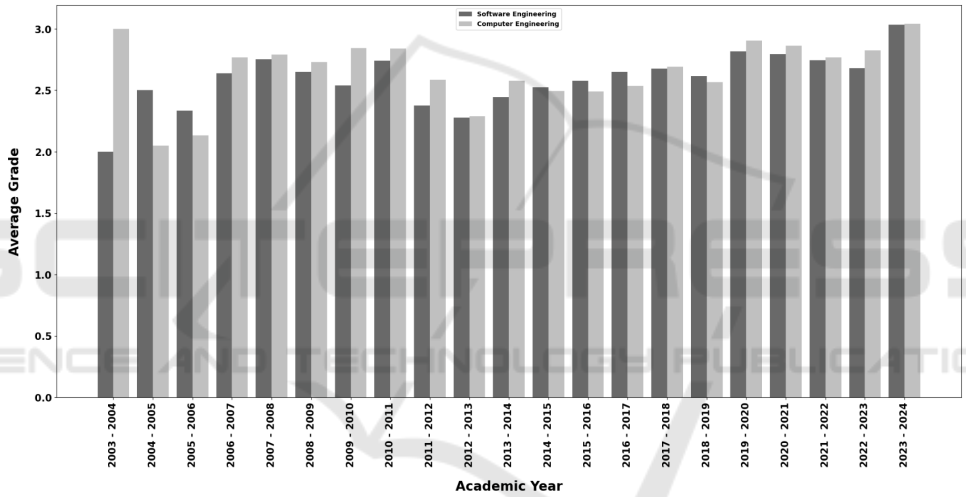

Figure 2: Average GPA over the years.

Table 1: Grading Scale with Corresponding Letter Grades, Grade Point Coefficients, and Academic Status in IUE.

Points	**Letter Grades**	**Coefficient**	**Status**
90-100	AA	4,00	Successful
85-89	BA	3,50	Successful
80-84	BB	3,00	Successful
75-79	CB	2,50	Successful
70-74	CC	2,00	Successful
65-69	DC	1,50	Successful
60-64	DD	1,00	Successful
50-59	FD	0,50	Unsuccessful
≤ 49	FF	0,00	Unsuccessful
-	EX (course transferred from external transcript)	-	Successful
-	S (Satisfactory)	-	Successful
-	P (Pass)	-	Successful

learning, web, and mobile development. The total number of students enrolled in each category was calculated. The results show that 3,347 students enrolled in software courses, 1,474 in game programming courses, 1,276 in web courses, 753 in artificial intelligence courses, and 551 in mobile development courses. Average grade scores were also computed for each group. Game programming related courses had the highest average grade at 2.98, followed by mobile development courses at 2.93. Software courses averaged 2.52, artificial intelligence courses 2.47, and web courses had the lowest average at 2.40.

Furthermore, the top ten most-enrolled elective courses are selected to examine and visualize the distribution of students across the course categories, as shown in Figure 3.

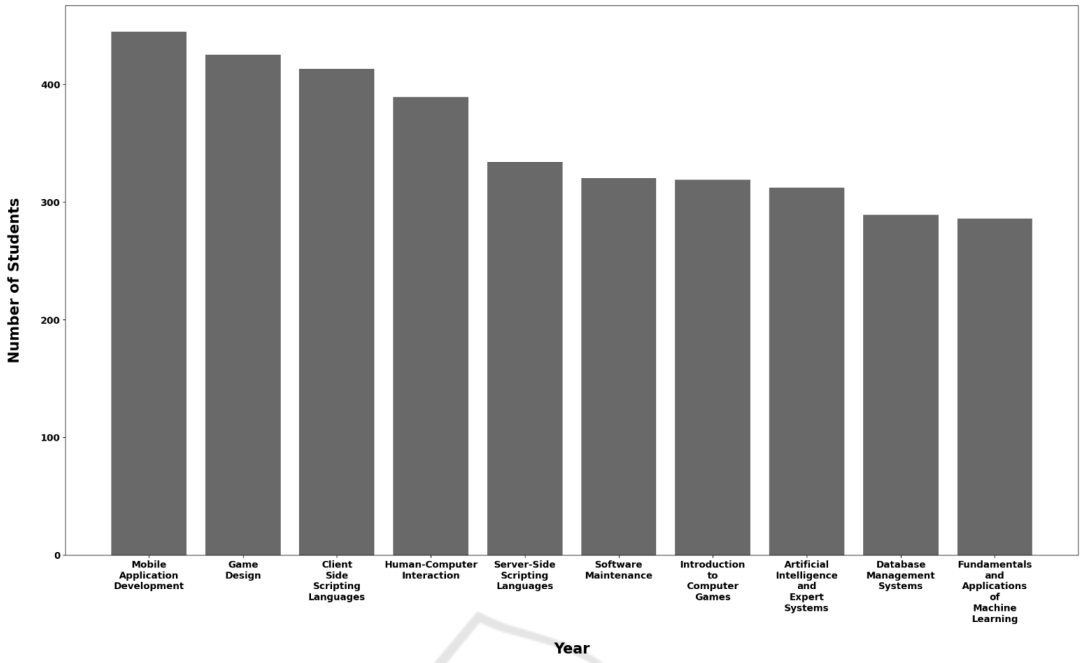

Figure 3: Distribution of the top ten most enrolled elective courses across course categories.

In addition, elective courses categorized as POOL courses are included. These are: POOL 3 (Economics), POOL 4 (Humanities), POOL 5 (Art and Communication), POOL 6 (Ethics and Public Awareness). These courses aim to broaden students' perspectives by fostering critical thinking, social awareness, and interdisciplinary connections. The average grade scores for POOL 3, POOL 4, POOL 5, and POOL 6 are 3.03, 3.00, 3.15, and 3.00, respectively.

The second part of the dataset consists of 735 mandatory courses, whose average grades are also taken into account. These courses are categorized as Software Engineering Department courses, Computer Engineering Department courses, and Mathematics and Science courses. Course grade averages are calculated for the first three years of the curriculum, as there are no mandatory courses from these departments in the senior (fourth) year. The average grade scores for Software Engineering Department courses are 2.61 in the first year, 2.71 in the second year, and 2.14 in the third year. Mandatory Computer Engineering courses are offered in the second and third years, with average scores of 2.10 and 2.50, respectively. For Mathematics and Science courses, the average scores are 2.32, 2.19, and 2.33 for the first, second, and third years, respectively.

4 METHODOLOGY

This section outlines the methodology designed for the two main tasks addressed in this study.

T1. *To identify the implicit or practical prerequisites, beyond the formally defined institutional requirements, that contribute to student success in a given course.*

Within the scope of this task, two different approaches were employed to seek a solution.

1. Identifying the courses that most significantly influence success in a specific target course.

   To achieve this, each student's previously completed courses and their corresponding letter grades were used to construct a profile, namely, a representation vector.

2. Examining the contribution of course learning outcomes (LOs).

   Here, student profiles were represented based on the learning outcomes of the courses they had completed. Given that some learning outcomes may be semantically similar, Sentence-BERT (SBERT) (Reimers and Gurevych, 2019) embeddings were utilised to represent LOs in the vector space, and cosine similarity was calculated between them. Learning outcomes with a cosine similarity greater than 0.85 were merged to reduce redundancy and ensure conceptual consistency in the representation.

To identify the top n courses that most significantly influenced the predicted performance in the target course, the SHAP method (Lundberg and Lee, 2017) was applied to the learned representation vectors. SHAP offers a consistent approach to model interpretability by assigning an importance value to each input feature based on its contribution to the model's output. This method operates by evaluating the impact of each feature on the prediction, analyzing how the model's output changes when the feature is included or excluded across various combinations.

T2. *To evaluate the extent to which a student's success in an elective course can be predicted based on their existing academic profile.*

To predict the extent to which a student will succeed in a given course, it is possible to utilise data collected from various sources that reflect the student's background and competencies. Within the scope of this task, the student's transcript, considered a more reliable source, was used to construct student representations, or in other words, profiles.

For this task, two different types of profiles were constructed for each student. The first profile (*content description profile (CDP)*) was based on the content descriptions of the courses the student had completed, while the second *(learning outcome profile (LOP))* utilised the learning outcomes associated with those courses. In both approaches, the grade the student received in each course was incorporated into the profile without disrupting the textual integrity of the content. For example, in the first type of profiles, the descriptions of courses taken by the student are first updated with an expression based on the success status of the student in the relevant course, and then appended to each other depending on the order in the SHAP results, and a single profile text is created. This text is then converted to a *profile vector* employing the embedding model. In this study, we employed Doc2Vec (Le and Mikolov, 2014), which generates vector representations for variable-length documents, enabling document-level similarity and classification tasks; Sentence-BERT (SBERT) (Reimers and Gurevych, 2019), a modification of the BERT architecture designed for efficient sentence similarity tasks; and Universal Sentence Encoder (Cer et al., 2018), which generates fixed-dimensional embeddings for sentences. A similar procedure is followed to build learning outcome profiles. In Table 3, sample profile texts are provided for a student who completed 3 courses (C1, C2 and C3) with grades AA, CC, and DD, respectively.

The profile embeddings are employed to train a number of classification (CL) models. The main aim of the CL process is to predict the level of success on the given course. The letter grades are categorized to success levels as given in Table 2 where the expression regarding the success level that is employed to build CDP and LOP text is also given in rightmost column. The performance of CL models together with alternative embeddings are measured by the average F1 measure. The F1 score is the harmonic mean of precision and recall where a high value refers to a successful classification performance. In the classification process, 5 fold cross validation is applied to avoid overfitting, and the CL models that are evaluated in this study are Decision Tree (DT), Multi-Layer Perceptron (MLP), Gaussian Naive Bayes (GNB), K-Nearest Neighbour (KNN), Support Vector Classifier (SVC), and Logistic Regression (LR), Random forest classifier (RFC) which are selected due to their distinct methodological approaches.

Table 2: Categorization of letter grades to success levels.

Letter Grades	**Coefficient**	**Description**	**Success Level**
AA	4.00	Very good	Excellent
BA	3.50	Good-Very good	Excellent
BB	3.00	Good	Excellent
CB	2.50	Average-Good	Pass
CC	2.00	Average	Pass
DC	1.50	Average-Weak	Pass
DD	1.00	Weak	Fail
FD	0.50	Very Weak-Fail	Fail
FF	0.00	Fail	Fail
EX	-	Pass (course transferred from external transcript)	Pass
S	-	Satistfactory	Pass
P	-	Pass	Pass

5 EXPERIMENTAL SETUP AND RESULTS

In this section, we describe the experimental steps undertaken to address the two main tasks of this study. The first part focuses on Task 1 (T1), where we utilize students' past course performances and learning outcomes to identify implicit prerequisites that contribute to success in target courses. This is achieved through SHAP-based interpretability applied on predictive models trained with course description-based and learning outcome-based representations. The latter part of the experiments relates to Task 2 (T2), where we construct embedding-based student profiles using course content and learning outcomes, and employ various classification models to predict student success in elective courses. The

Table 3: Sample profile texts.

Course	Grade	Course Content Description	Course Learning Outcomes	CDP Text	LOP Text
C1	AA	This course introduces the students to the fundamental concepts of programming using Java programming language.	1- will be able to define the fundamental concepts in programming 2- will be able to write, compile and debug programs in Java language 3- will be able to use control structures 4- will be able to design functions in Java codes	Excellent at: This course introduces the students to the fundamental concepts of programming using Java programming language	Excellent at: will be able to define the fundamental concepts in programming Excellent at: will be able to write, compile and debug programs in Java language Excellent at: will be able to use control structures Excellent at: will be able to design functions in Java codes
C2	CC	This course covers the fundamental concepts of object-oriented programming using Java programming language.	1- will be able to define classes in Java programming language. 2- will be able to define the features of object-oriented programming languages. 3- will be able to develop programs in Java programming language using objects. 4- will be able to use inheritance technique in class designs with Java programming language. 5- will be able to implement the polymorphism concept in Java programming language.	Pass at: This course covers the fundamental concepts of object-oriented programming using Java programming language.	Pass at: will be able to define classes in Java programming language. Pass at: will be able to define the features of object-oriented programming languages. Pass at: will be able to develop programs in Java programming language using objects. Pass at: will be able to use inheritance technique in class designs with Java programming language. Pass at: will be able to implement the polymorphism concept in Java programming language.
C3	DD	The course provides the fundamental concepts of software engineering discipline and gives concepts of abstraction, problem solving and systematic view.	1- explain engineering, software, computer and system engineering 2- define software processes 3- gather the software requirements 4- design using UML 5 - explain software verification and validation	Fail at: The course provides the fundamental concepts of software engineering discipline and gives concepts of abstraction, problem solving and systematic view.	Fail at: explain engineering, software, computer and system engineering Fail at: define software processes Fail at: gather the software requirements Fail at: design using UML Fail at: explain software verification and validation

results from these embedding-driven models are evaluated to assess their effectiveness in supporting informed course enrollment decisions. The first four steps focus on Task 1 (T1), aiming to identify practical prerequisites through SHAP analysis, while steps five and six correspond to Task 2 (T2), involving embedding-based profile construction and predictive modeling for student success. The details about the steps can be seen as follows:

1. Target Course Selection and Training Dataset Construction:

Firstly, in the experimental phase of the study, four mandatory third-year software engineering courses (coded as *SE1*, *SE2*, *SE3*, and *SE4*) were selected, along with four popular elective courses from the areas of *game development (GD), web technologies (WT), artificial intelligence (AI)*, and *mobile programming (MP)*, serving as the target courses. The selection process prioritized courses with high student enrollment to ensure both broad representativeness and practical relevance. Additionally, efforts were made to include courses that span a variety of subfields within the discipline, enabling a more in-depth exploration of curriculum design and teaching methods. A summary of the selected target courses and

Table 4: The list of target compulsory (C) and elective (E) courses and their titles and descriptions.

Course Code	Course Type	Course Descriptive Title	Description
SE1	C	Software Architecture	This course covers the principals behind the software design patterns and their application in constructing software components.
SE2	C	Concepts of Programming Languages	The following topics will be included in the course: lexical and syntax analysis, names, bindings, type checking, scopes, data types, expressions, assignment statements, subprograms, implementing subprograms, abstract data types and encapsulation constructs, support for object-oriented programming, exception handling, event handling.
SE3	C	Systems Programming	To acquaint students with basic knowledge to develop systems programs that involves multi-threading and computer networks. It provides an introduction to multi-threading, socket programming and information security.
SE4	C	Software Specification and Design	In this course, students learn the theoretical and practical aspects of specification and design stages of software engineering. More, this course enables students to realize software specification and design phases of sample projects with real clients.
GD	E	Game Development	In this course, students learn about the process of game development and use this information to develop their own games.
WT	E	Web Technologies	This course introduces the students to the fundamental concepts of web programming using HTML, CSS, JavaScript, jQuery and JSON.
AI	E	Artificial Intelligence	This course provides an introduction to Artificial Intelligence (AI). In this course we will study a number of theories, mathematical formalisms, and algorithms, that capture some of the core elements of computational intelligence. We will cover some of the following topics: search, logical representations and reasoning, automated planning, representing and reasoning with uncertainty, decision making under uncertainty, and learning.
MP	E	Mobile Programming	Mobile devices, mobile applications and their requirements, developing mobile applications, using web services and databases in mobile applications.

their brief descriptions is provided in Table 4.

Then, we created eight separate training datasets, one for each of the eight courses. For example, in the training dataset for the course *SE1*, the first column contains the student ID, and the remaining columns correspond to all courses that were taken by at least one student before *SE1*. Since each student had taken a different set of prior courses, not all columns are filled for every student. If a student did not take a particular course, the corresponding entry is marked as N/A. Otherwise, we recorded the letter grade they received in that course (e.g., AA, BA, etc.). In this way, we constructed eight training datasets, each tailored to one of the eight target courses.

In addition to this grade-based representation, we also constructed an alternative representation based on the learning outcomes (LOs) of the courses students had previously completed. SBERT embeddings were used to represent each LO in a vector space, and cosine similarity was calculated to merge semantically similar LOs (similarity 0.85), ensuring a more consistent and conceptually meaningful feature space. These LO-based representations enabled us to model students not only based on their academic performance, but also based on the underlying competencies they acquired.

2. Course Filtering:

Secondly, we applied filtering to courses and LOs presented in the training dataset in order to decrease the size of each representation. Two filters are applied to reduce the number of courses and eliminate certain compulsory courses included in the academic curriculum in accordance with the regulations established by the Council of Higher Education (YÖK). The first filter removes courses that appear in less than %25 of the samples in the dataset. The second filter excludes science courses and other unrelated courses that have high levels of student preference.

For filtering LOs, firstly, the LOs related with courses that are not presented in one of the courses of a software engineering student's curriculum were eliminated. Secondly, LOs of English courses were also removed from the dataset.

3. Training Model Descriptions:

After filtering some of the courses (features) in the eight training datasets, each of them was then used to train seven different classification models: Decision Tree, Multi-layer Perceptron (MLP), Gaussian Naive Bayes, K-Nearest Neighbors (KNN), Support Vector Classifier (SVC), Logistic Regression and Random Forest.

In total, this resulted in 112 model training runs, corresponding to 7 models trained on each of the 8 datasets using 2 different representation vectors. For each combination, we applied 5-fold cross-validation and evaluated performance using the F1-score. The model with the highest average F1-score was selected as the best-performing model for each dataset.

4. SHAP-Based Identification of Impactful Prior Courses:

In this step, we aimed to identify which prior courses had the most significant impact on students' performance in target courses. To achieve this, we employed SHAP (Lundberg and Lee, 2017) on each of the eight best-performing models (determined in the previous step), using their respective training datasets.

This analysis allowed us to quantify the influence of each input feature (i.e., prior courses) on the grade prediction for a specific target course. For each of the eight target courses, we selected the top two most influential prior courses based on their SHAP values. For example, in the case of the target course *SE1*, the most impactful prior courses were identified as *SE1_A* and *SE1_B*.

5. Constructing Embedding Datasets Based on Influential Prior Courses:

Following the SHAP analysis, we constructed two new embedding datasets for each of the eight target courses, using the selected influential prior courses. The datasets are as follows:

- *Content Description Profile (CDP) based dataset:* For each student, we retrieved the CDP texts as represented in Table 3 corresponding to the two selected prior courses and concatenated them.
- *Learning Outcome Profile (LOP) based dataset:* Similarly, we retrieved the LOP texts for the same two prior courses and concatenated them.

Each of these text representations for each students was then fed into three different embedding models (Doc2Vec, SBERT, and the Universal Sentence Encoder (USE)) to obtain vector representations of students. For SBERT, "all-mpnet-base-v2" with 768-length embeddings and for USE, "Dimitre/universal-sentence-encoder" [1] having vector size 512, models are utilized.

6. Predictive Modeling with Embeddings:

The embeddings obtained from the *CDP* and *LOP* datasets were used as input features for predictive modeling. We trained models using the same seven classifiers employed in the previous phase (e.g., Logistic Regression, Random Forest, etc.) and evaluated their performance using 5-fold cross-validation. The goal was to determine whether representations based on influential prior courses could effectively predict student performance in the target courses.

The classification F1 scores of all configurations show slight variations, ranging from 0.50 to 0.65. The best performance, with an F1 score of 0.65, was achieved using course grades as the best representation combined with course content description profiles as embeddings, where the Sentence-BERT embedding method was employed. Additionally, prediction performance was generally higher for elective target courses compared to mandatory courses.

Additionally, the outcomes of the most influential courses identified for our eight target courses can be interpreted as follows:

SE1 – Software Architecture

- An introductory-level course on programming was identified as the most influential course based on both course descriptions and learning outcomes. As a foundational programming course, it equips students with essential skills in logic, control structures, and basic problem-solving, which directly support their ability to recognize and apply software design patterns in SE1.
- Based on course descriptions, discrete mathematics emerged as a practical prerequisite. The logical reasoning and formal structures covered in this course—such as sets, relations, and graphs—are closely related to the abstraction and structure-oriented thinking required in SE1.
- According to learning outcomes, the course on database management systems was also identified as an influential course. Understanding databases and system components may enhance students' ability to make architectural software decisions, thereby indirectly supporting the design-oriented learning objectives in SE1.

SE2 – Concepts of Programming Languages

- Based on both course content and LOs two programming courses given to first year students were determined. First course lays the groundwork for understanding language syntax and semantics, which is deepened in SE2. Second introduces object-oriented programming, which is essential in understanding language paradigms, encapsulation, and inheritance that are the core topics in SE2.

[1] https://huggingface.co/Dimitre/universal-sentence-encoder

SE3 – System Programming

- Experimental results based on both course content and learning outcomes indicate that database management systems course has a significant impact on success in SE3. This may be because the course provides experience in system-level data manipulation, which complements the networked and multithreaded programming concepts covered in SE3.
- Course content-based experimental results highlight the introduction to programming as an influential course, as it establishes the problem-solving and logical reasoning skills necessary for writing programs in SE3.
- Experiments based on learning outcomes suggest that SE1 has a significant effect, since understanding system architecture and modularity helps students develop robust and concurrent systems in SE3

SE4 – Software Specification and Design

- The results based on course content and learning outcomes suggest that Calculus II given in first year is an influential course. This may be because the course enhances analytical thinking and formal modeling skills, both of which are essential for managing complex software project design and specification in SE4.
- Another influential course identified is the course on object-oriented analysis and design, as it provides fundamental methods and modeling tools, such as UML diagrams, that students directly apply in SE4 while developing real-world software projects.

GD – Game Development

- According to course content-based analysis, SE2 was identified as an influential course, as understanding functional, object-oriented, and imperative paradigms helps students implement logic and scripting more effectively within game engines.
- Secondly, both course content and learning outcome-based experiments highlight SE1 as influential, since it enables students to design scalable, maintainable, and efficient architectures—an essential skill for developing complex game systems.
- Additionally, learning outcome-based analysis suggests the course on probability and statistics as a contributing course. This may be because probability and statistics skills enhance game logic, particularly in areas such as randomness, AI behavior, and physics simulations.

WT – Web Technologies

- According to course content-based analysis, Human-Computer Interaction course was identified as an influential course, as it provides essential insights into user experience and interface design principles, which are critical for front-end web development.
- Database management systems course was also determined to be impactful based on both of the experiments, as it equips students with the necessary skills to design and implement backend systems—an essential component of full-stack web applications.
- According to learning outcome-based analysis, the course on history of civilization was found to be influential.

MP – Mobile Programming

- According to both of experiments, database management systems course was identified as an influential course, as many mobile applications rely on local or remote databases.
- The introductory-level course on programming was also found to be impactful, as it develops core programming skills such as logic, control flow, and event handling—fundamental competencies required in mobile application interfaces and frameworks.
- Additionally, SE1 was identified as influential according to LO-based experiments, as the ability to design modular and maintainable systems is essential for building scalable and robust mobile applications.

AI – Artificial Intelligence

- According to course content-based analysis, SE3 was identified as an influential course, as it provides students with essential knowledge in memory management, concurrency, and low-level optimization—all of which are critical for developing efficient artificial intelligence implementations.
- The introductory-level course on programming was also found to be impactful, as it lays the groundwork for algorithmic thinking and control structures, which are fundamental for implementing AI algorithms effectively.
- According to learning outcome-based analysis, the courses on programming were highlighted as relevant, as they emphasize object-oriented logic and class structure design, supporting the development of AI agents and rule-based systems.
- Additionally, Physics was determined to be influential.

6 CONCLUSIONS

In this study, we proposed a framework for supporting undergraduate students in the course selection process by identifying implicit prerequisites and predicting success in elective courses. By utilizing anonymized transcript data and course-level textual information, we constructed student profiles based on both academic performance in courses and learning outcomes. These profiles were transformed into embedding representations using various natural language processing models.

Two main tasks were addressed: (1) discovering the practical prerequisites that significantly contribute to course success, and (2) evaluating the extent to which a student's success in an elective course can be predicted based on their existing academic background. Through SHAP-based analysis, we identified prior courses with the highest impact on performance, while embedding-based classification models achieved promising F1 scores—particularly when Sentence-BERT was used with course content profiles.

Our results demonstrate that combining structured academic records with semantic representations of course content can lead to a more informed and personalized course selection process and especially identification and potential revision of course prerequisites based on the analysis of existing student performance data.

ACKNOWLEDGEMENTS

The anonymized transcript data used in this study were kindly provided by İzmir University of Economics. All personal data were anonymized prior to analysis and securely stored, with access restricted to the research team.

REFERENCES

Anh, N., Nguyen, H. H., Nguyen, D.-L., and Le, M.-D. (2021). A course recommendation model for students based on learning outcome. *Education and Information Technologies*, 26.

Atalla, S., Daradkeh, M., Gawanmeh, A., Khalil, H., Mansoor, W., Miniaoui, S., and Himeur, Y. (2023). An intelligent recommendation system for automating academic advising based on curriculum analysis and performance modeling. *Mathematics*, 11:1098.

Aytekin, M. C. and and, Y. S. (2025). Discovering prerequisite relations using large language models. *Interactive Learning Environments*, 33(2):1670–1688.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Cer, D., Yang, Y., yi Kong, S., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., and Kurzweil, R. (2018). Universal sentence encoder.

Deventer, H. V., Mills, M., and Evrard, A. (2024). From interests to insights: An llm approach to course recommendations using natural language queries. *ArXiv*, abs/2412.19312.

Esteban, A., Zafra, A., and Romero, C. (2020). Helping university students to choose elective courses by using a hybrid multi-criteria recommendation system with genetic optimization. *Knowledge-Based Systems*, 194:105385.

Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Inui, K., Jiang, J., Ng, V., and Wan, X., editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Touvron, H., Martin, L., Stone, K., Almahairi, A., Babaei, Y., Bashlykov, S., Batra, S., Baumgartner, T., Bhosale, S., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Zhu, L. and Wang, B. (2022). *Course Selection Recommendation Based on Hybrid Recommendation Algorithms*, pages 476–482.