THE UNIVERSITY OF DANANG
UNIVERSITY OF SCIENCE AND TECHNOLOGY
Faculty of Advanced Science and Technology

Artificial Intelligence & Big Data

Final Report

Instructor **:** Prof. Pham Van Tuan

Group Members **:** Dang Minh Quan - 21ECE
Ngo Thanh Thao - 21ECE
Tran Hoai Nha Thi - 21ECE

Da Nang,

June 2025

Applying AI in personalizing learning Python programming language content for technical students

I. Problem Statement/Introduction

1.1. Background and Motivation

With the rapid growth of online learning platforms such as Coursera, Udemy, and edX, students now have access to a vast range of courses covering programming, data science, and artificial intelligence. Among these, Python remains one of the most in-demand and foundational programming languages. However, selecting the most appropriate course for each learner—considering their background, interests, and current performance level—remains a challenging task.

There is a growing need for intelligent systems that can personalize learning pathways and recommend suitable courses to maximize learning outcomes. Leveraging AI to analyze student performance and match them with the most relevant courses can significantly improve learning efficiency and engagement.

1.2 Current Issues

- Students often choose courses without understanding if the content matches their skill level (e.g., beginner enrolling in intermediate-level content).
- There is no centralized system that cross-analyzes course metadata with student performance data to give adaptive course recommendations.
- Existing course platforms lack personalized feedback or explanation as to why a course is suitable for a particular learner.
- Students with similar goals but different learning behavior patterns may receive identical course recommendations, leading to ineffective outcomes.

1.3 Objectives

This project aims to develop an AI-powered system that provides personalized course recommendations and feedback for students interested in learning Python. The main objectives are:

- To classify student performance levels (e.g., beginner, intermediate, advanced) based on input data such as user behavior and academic background.

- To analyze similarities between students and available courses using multiple datasets from platforms like Coursera.
- To integrate Generative AI (GenAI) for generating feedback and explanations based on the matched course and the student profile.
- To offer learners suitable advice and the most appropriate learning path, improving both course completion rates and user satisfaction.

Target users include:

- College and university students in the following fields: Information Technology, Data Science, Engineering, Applied Mathematics...
- High school students with an IT orientation who want to learn programming early
- Non-CS students in the following fields: Economics, Business,...

II. Proposed Solution

To address the challenges in guiding students toward suitable learning paths based on their academic performance, we propose an integrated system that combines Traditional AI techniques with a modern LLM-based Retrieval-Augmented Generation (RAG) architecture. Our solution evaluates student performance using machine learning models trained on academic data, then recommends appropriate courses based on content similarity and relevance. These recommended courses are stored in a retrieval system, which an LLM leverages to generate personalized course suggestions and justifications in natural language. This approach ensures not only accurate evaluation but also meaningful guidance tailored to each student's learning needs and goals.

System Overview:

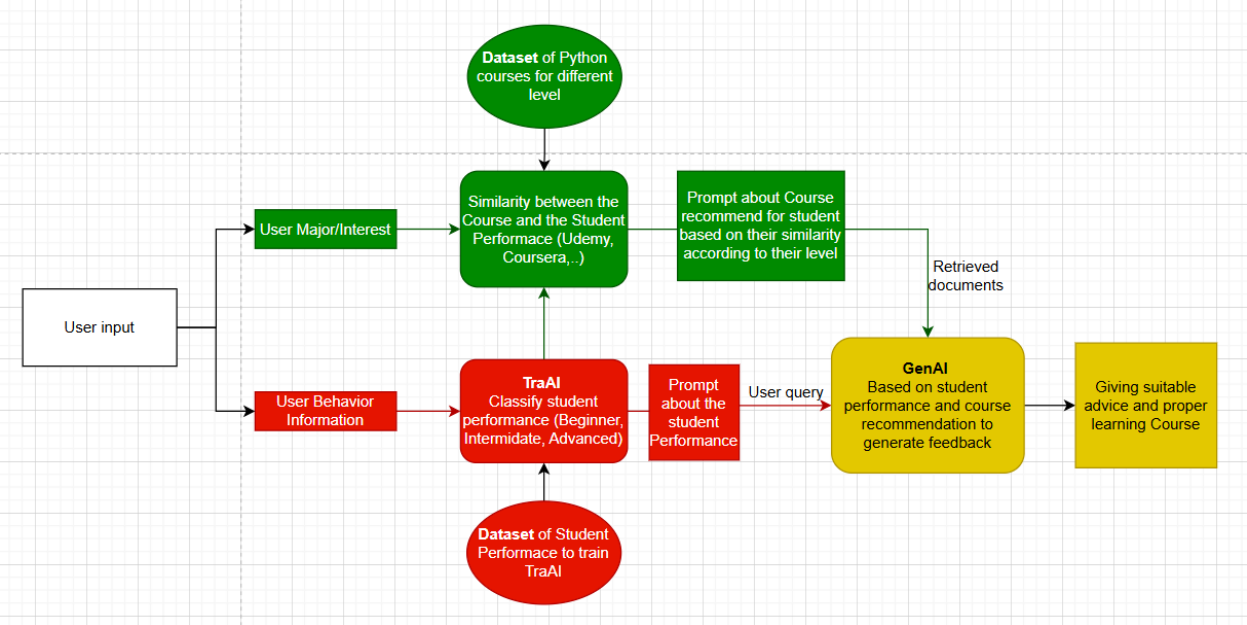


Figure 1.1 System Overview Diagram

Key Components:

- In the **Red stage**, students provide their information, which is then processed by TraAI, an AI model using algorithms like SVM and Decision Tree—to classify their current level of knowledge.
- In the **Green stage**, based on the predicted student level, the system searches a curated dataset of online courses to identify those that best match the student's profile.
- In the **Yellow stage**, a Retrieval-Augmented Generation (RAG) system is used. It retrieves relevant course content from the Green stage, while the Red stage provides a query (based on the student's evaluation) to the LLM, enabling it to generate personalized course recommendations and guidance in natural language.

Simple understanding of the system:

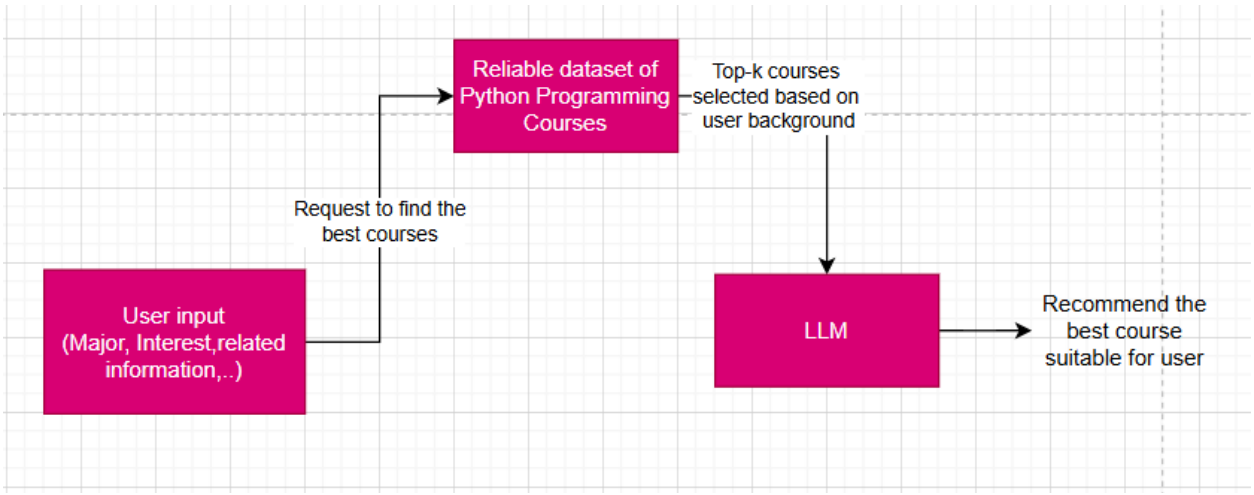


Figure 1.2 Recommendation system simple flowchart

This system is a course recommendation framework that utilizes a Large Language Model (LLM) to suggest personalized learning paths. It operates as follows:

- **User Input**: The process begins with the user providing information such as their major, interests, and related details.
- **Request to Find Best Courses**: Based on the input, a request is made to identify the most suitable courses.
- **Reliable Dataset**: A comprehensive dataset of Python Programming courses is accessed, from which the top-k courses are selected based on the user's background.
- **LLM Processing**: The LLM analyzes the selected courses and user data to determine the best course recommendation.
- **Recommendation**: The system outputs a tailored course recommendation suitable for the user.

III. Methodology:

The system integrates three components: classification of students using ML, course matching via similarity metrics, and a RAG pipeline that combines retrieved course content with student context for final suggestions.

3.1 Traditional AI for Student Performance Evaluation

3.1.1. Dataset Preparation:

To build an AI-powered system for evaluating student performance, it is essential to identify and prepare relevant datasets that reflect academic behaviors and outcomes. We have selected several publicly available student performance datasets to support this objective.

Dataset 1: [Link]
Collected from students of the Faculty of Engineering and the Faculty of Educational Sciences in 2019.

Dataset 2: [Link]
Contains data from university students in Cyprus, focusing on academic and behavioral factors.

Dataset 3: [Link]
An extended survey that includes detailed information on students' backgrounds and lifestyles.

To select the most appropriate dataset for our system, we will conduct a comparative analysis using our predefined criteria through a decision matrix.

Decision Matrix for choosing TraAI dataset:

Why	Criteria	Weight	Data 1	Data 2	Data 3	Reason
Data from reputable, academic sources that can be cited in the report	Data Source Reliability	25%	10	6	6	UCI is academic; Kaggle sources are unclear
The data has many descriptive attributes that are clearly defined for analyzing student performance.	Feature Completeness	20%	9	7	7	UCI has 33 features; others are trimmed
Knowing the year of collection helps assess the timeliness or relevance	Collection Year Clarity	10%	10	5	5	Only UCI provides collection year
Data is consistent, not missing, not significantly edited	Data Integrity	15%	9	7	6	UCI is original; Kaggle versions may be modified
Ready to use for analysis, little need for cleaning	Readiness for Analysis	10%	9	6	7	UCI is pre-cleaned; others may need work
Have clear reference documents and articles	Documentation Availability	10%	10	3	3	Only UCI includes a reference paper
Easy to download, clear license	Accessibility & License Clarity	10%	10	10	10	All are downloadable and free to use
		Total score	**9.55**	**6.35**	**6.3**	**Choose data 1**

Figure 2. Decision matrix of Higher Education Students Performance Evaluation dataset

Final chosen Dataset

- Higher Education Students Performance Evaluation data set:

https://archive.ics.uci.edu/dataset/856/higher+education+students+performance+evaluation

This dataset is **real data of 145 records** collected from the Faculty of Engineering and Faculty of Educational Sciences students in 2019 **[1]**. The purpose is to predict students' end-of-term performances using ML techniques.

Overview of the dataset

	STUDENT ID	Student Age	Sex	Graduated high-school type	Scholarship type	Additional work	artistic or sports activity	have a partner	Total salary	Transportation	...	Preparation to midterm exams 1	Preparation to midterm exams 2	Taking notes in classes	Listening in classes	Discussion improves my interest and success in the course	Flip-classroom	Cumulative grade point	Expected Cumulative grade	COURSE ID	GRADE
0	STUDENT1	2	2	3	3	1	2	2	1	1	...	1	1	3	2	1	2	1	1	1	1
1	STUDENT2	2	2	3	3	1	2	2	1	1	...	1	1	3	2	3	2	2	3	1	1
2	STUDENT3	2	2	2	3	2	2	2	2	4	...	1	1	2	2	1	1	2	2	1	1
3	STUDENT4	1	1	1	3	1	2	1	2	1	...	1	2	3	2	2	1	3	2	1	1
4	STUDENT5	2	2	1	3	2	2	1	3	1	...	2	1	2	2	2	1	2	2	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
140	STUDENT141	2	1	2	3	1	1	2	1	1	...	1	1	2	1	2	1	3	3	9	5
141	STUDENT142	1	1	2	4	2	2	2	1	4	...	1	1	3	2	2	1	5	3	9	5
142	STUDENT143	1	1	1	4	2	2	2	1	1	...	2	3	2	2	2	1	2	2	9	1
143	STUDENT144	2	1	2	4	1	1	1	5	2	...	2	1	2	1	2	1	5	3	9	4
144	STUDENT145	1	1	1	5	2	2	2	3	1	...	2	1	3	2	3	1	5	4	9	3

145 rows × 33 columns

Figure 3. Raw Student dataset

Main features as input for the TraAI model

1- Weekly study hours: (1: None, 2: <5 hours, 3: 6-10 hours, 4: 11-20 hours, 5: more than 20 hours)

2- Reading frequency (non-scientific books/journals): (1: None, 2: Sometimes, 3: Often)

3- Reading frequency (scientific books/journals): (1: None, 2: Sometimes, 3: Often)

4- Attendance to the seminars/conferences related to the department: (1: Yes, 2: No)

5- Attendance to classes (1: always, 2: sometimes, 3: never)
6- Preparation to midterm exams 1: (1: alone, 2: with friends, 3: not applicable)
7- Preparation to midterm exams 2: (1: closest date to the exam, 2: regularly during the semester, 3: never)
8- Taking notes in classes: (1: never, 2: sometimes, 3: always)
9- Listening in classes: (1: never, 2: sometimes, 3: always)
10- Cumulative grade point average in the last semester (/4.00): (1: <2.00, 2: 2.00-2.49, 3: 2.50-2.99, 4: 3.00-3.49, 5: above 3.49)
11- Expected Cumulative grade point average in the graduation (/4.00): (1: <2.00, 2: 2.00-2.49, 3: 2.50-2.99, 4: 3.00-3.49, 5: above 3.49)

And output target

12- OUTPUT Grade (0: Fail, 1: DD, 2: DC, 3: CC, 4: CB, 5: BB, 6: BA, 7: AA)

Description of the Student Performance Dataset:

```
Data columns (total 33 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   STUDENT ID                                              145 non-null    object
 1   Student Age                                             145 non-null    int64
 2   Sex                                                     145 non-null    int64
 3   Graduated high-school type                              145 non-null    int64
 4   Scholarship type                                        145 non-null    int64
 5   Additional work                                         145 non-null    int64
 6   artistic or sports activity                             145 non-null    int64
 7   have a partner                                          145 non-null    int64
 8   Total salary                                            145 non-null    int64
 9   Transportation                                          145 non-null    int64
 10  Accommodation                                           145 non-null    int64
 11  Mother Education                                        145 non-null    int64
 12  Father Education                                        145 non-null    int64
 13  Number of sisters/brothers                              145 non-null    int64
 14  Parental status                                         145 non-null    int64
 15  Mother Occupation                                       145 non-null    int64
 16  Father Occupation                                       145 non-null    int64
 17  Weekly study hours                                      145 non-null    int64
 18  Reading frequency (non-scientific books/journals)       145 non-null    int64
 19  Reading frequency (scientific books/journals)           145 non-null    int64
 20  Attendance to the seminars/conferences related to the department  145 non-null  int64
 21  Impact of your projects/activities on your success      145 non-null    int64
 22  Attendance to classes                                   145 non-null    int64
 23  Preparation to midterm exams 1                          145 non-null    int64
 24  Preparation to midterm exams 2                          145 non-null    int64
 25  Taking notes in classes                                 145 non-null    int64
 26  Listening in classes                                    145 non-null    int64
 27  Discussion improves my interest and success in the course  145 non-null  int64
 28  Flip-classroom                                          145 non-null    int64
 29  Cumulative grade point                                  145 non-null    int64
 30  Expected Cumulative grade                               145 non-null    int64
 31  COURSE ID                                               145 non-null    int64
 32  GRADE                                                   145 non-null    int64
dtypes: int64(32), object(1)
```

Figure 4. Schema of the Student Performance Dataset

Statistics of the dataset:

	Weekly study hours	Reading frequency (non-scientific books/journals)	Reading frequency (scientific books/journals)	Attendance to the seminars/conferences related to the department	Attendance to classes	Preparation to midterm exams 1	Preparation to midterm exams 2	Taking notes in classes	Listening in classes	Cumulative grade point	Expected Cumulative grade	GRADE
count	145.000000	145.000000	145.000000	145.000000	145.000000	145.000000	145.000000	145.000000	145.000000	145.000000	145.000000	145.000000
mean	3.027586	2.103448	2.082759	1.462069	1.903448	1.841379	1.772414	2.075862	2.103448	2.682759	2.758621	1.744828
std	1.148337	0.761250	0.759234	0.500287	0.729575	0.693981	0.597969	0.755377	0.742781	1.239979	1.203443	0.779773
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	2.000000	2.000000	2.000000	1.000000	1.000000	1.000000	1.000000	2.000000	2.000000	2.000000	2.000000	1.000000
50%	3.000000	2.000000	2.000000	1.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	3.000000	2.000000
75%	4.000000	3.000000	3.000000	2.000000	2.000000	2.000000	2.000000	3.000000	3.000000	4.000000	4.000000	2.000000
max	5.000000	3.000000	3.000000	2.000000	3.000000	3.000000	3.000000	3.000000	3.000000	5.000000	5.000000	3.000000

Figure 5. Statistic of the dataset

Correlation between Each features

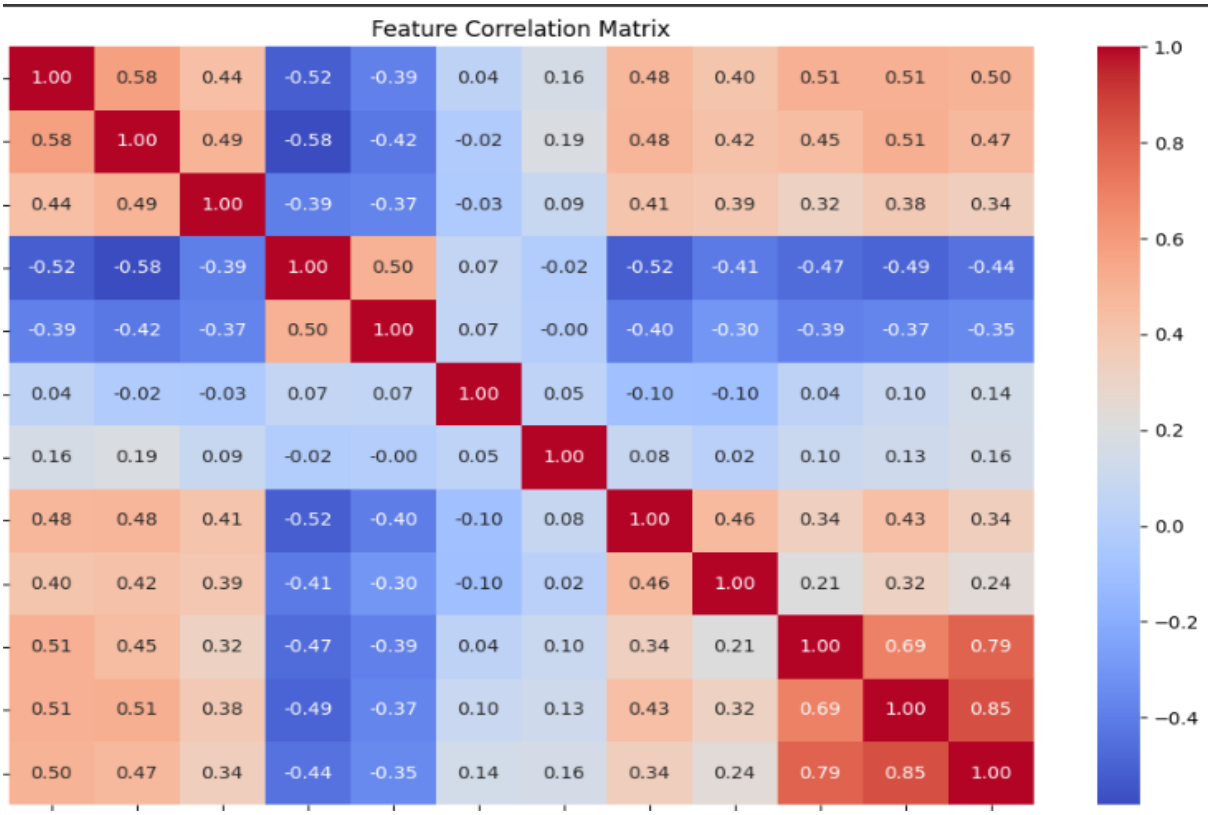


Figure 6. Correlation between each features in Student Performance dataset

Compare between 11 input features:

- **Strong Positive Correlations**: The highest correlation (1.00) appears on the diagonal, as expected, indicating perfect self-correlation for each feature. Notable strong positive correlations include Cumulative grade point average (last semester) with Expected Cumulative grade point average (0.97), suggesting a strong predictive relationship between past and expected performance.

- **Moderate Positive Correlations:** Features like Listening in classes (0.58) and Taking notes in classes (0.49) with Cumulative grade point average show moderate positive relationships, indicating that active engagement in class contributes to better grades. Similarly, Preparation to midterm exams 1 and 2 (0.44 and 0.39) correlate moderately with academic outcomes.
- **Weak or Negative Correlations:** Weekly study hours (0.16 to 0.39) and Reading frequency (non-scientific: 0.04 to 0.37, scientific: 0.07 to 0.42) show weaker or mixed correlations with other features, suggesting limited direct impact on grades. Negative correlations, such as between Attendance to seminars/conferences and Expected Cumulative grade point average (-0.35), hint at possible distractions or irrelevance for some students.
- **Attendance Impact:** Attendance to classes (0.48 with Listening, 0.40 with Taking notes) and seminars/conferences (0.34 with Attendance to classes) shows moderate positive ties, reinforcing the importance of presence in academic success.

- **Output class (Target):**

Output class is an integer

1. Grade (0: Fail, 1: DD, 2: DC, 3: CC, 4: CB, 5: BB, 6: BA, 7: AA)
    - The distribution of each grade is illustrated in Figure 5.

GRADE	
1	35
2	24
3	21
5	17
7	17
6	13
4	10
0	8

Figure 7. Distribution of each grade in Student Performance dataset

The grade distribution peaks at Grade 1 (DD) with 35 students, followed by Grade 2 (DC) with 24 and Grade 3 (CC) with 21. Higher grades include Grade 7 (AA) and Grade

5 (BB) with 17 each, while Grade 6 (BA) has 13, Grade 4 (CB) has 10, and Grade 0 (Fail) has 8, showing a skew toward lower to middle grades with some high achievers.

3.1.2 Dataset Processing:

Before applying any machine learning models, it is essential to examine and prepare the dataset to ensure it is suitable for analysis. A clear overview of the dataset allows us to make informed decisions during preprocessing and model selection. The following steps outline the data processing pipeline used in this project:

- Handle Missing data
- Remove irrelevant features that do not significantly impact student performance
- Processing Numerical/Categories Features:
    - Since all the features are in integers we standardized the data using Standard Scacler.
    - For the categories Grade we change target to Beginner -> 1, Intermediate -> 2, Advanced -> 3 (Label 0-2 change to Beginner, label 3-5 change to Intermediate, label 6-7 change to Advanced)

Input data after Processing overview:

	Weekly study hours	Reading frequency (non-scientific books/journals)	Reading frequency (scientific books/journals)	Attendance to the seminars/conferences related to the department	Attendance to classes	Preparation to midterm exams 1	Preparation to midterm exams 2	Taking notes in classes	Listening in classes	Cumulative grade point	Expected Cumulative grade	GRADE
0	3	2	2	1	1	1	1	3	2	1	1	1
1	2	2	2	1	1	1	1	3	2	2	3	1
2	2	1	2	1	1	1	1	2	2	2	2	1
3	3	1	2	1	1	1	2	3	2	3	2	1
4	2	1	1	1	1	2	1	2	2	2	2	1

Figure 8.Student Performance dataset after cleaning data

Remain only important features:

1- Weekly study hours: (1: None, 2: <5 hours, 3: 6-10 hours, 4: 11-20 hours, 5: more than 20 hours)

2- Reading frequency (non-scientific books/journals): (1: None, 2: Sometimes, 3: Often)

3- Reading frequency (scientific books/journals): (1: None, 2: Sometimes, 3: Often)

4- Attendance to the seminars/conferences related to the department: (1: Yes, 2: No)

5- Attendance to classes (1: always, 2: sometimes, 3: never)

6- Preparation to midterm exams 1: (1: always, 2: sometimes, 3: never)

7- Preparation to midterm exams 2: (1: always, 2: sometimes, 3: never)

8- Taking notes in classes: (1: never, 2: sometimes, 3: always)

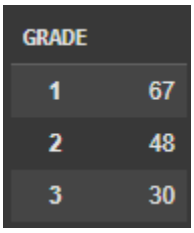
9- Listening in classes: (1: never, 2: sometimes, 3: always)

10- Cumulative grade point average in the last semester (/4.00): (1: <2.00, 2: 2.00-2.49, 3: 2.50-2.99, 4: 3.00-3.49, 5: above 3.49)

11- Expected Cumulative grade point average in the graduation (/4.00): (1: <2.00, 2: 2.00-2.49, 3: 2.50-2.99, 4: 3.00-3.49, 5: above 3.49)

Total of each Grade after changing the target value:

- Beginner -> 1, Intermediate -> 2, Advanced -> 3



GRADE	
1	67
2	48
3	30

Figure 9. Distribution of each grade in Student Performance dataset after merging

The grade distribution shows 67 students at Grade 1 (Beginner), 48 at Grade 2 (Intermediate), and 30 at Grade 3 (Advanced), indicating a larger beginner base that decreases with advancing levels, likely reflecting typical educational progression.

Split the Data:

- Divide the dataset into training, validation, and test sets (80% train, 20% test).
- Training set: Used to train the model.
- Test set: Used to evaluate final performance.
- Consider imbalance data, insufficient data by using SMOTE (Synthetic Minority Over-sampling)

3.1.3 Model Selection and Training:

This is a classification task with multiple input features (all integers), the following models are suitable:

- **Random Forest**: A robust ensemble method that handles mixed data types, captures non-linear relationships, and provides feature importance.

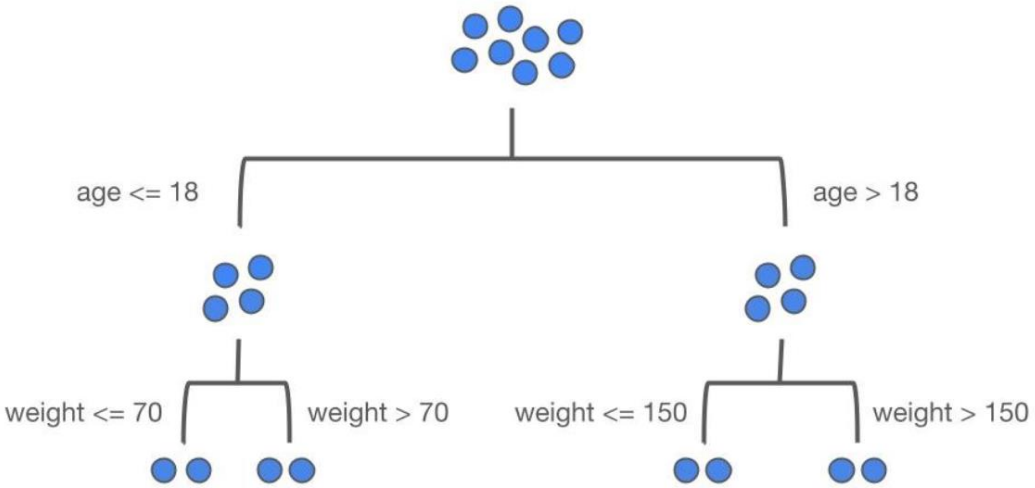


Figure 10. Random Forest Model

- **SVM**: A simpler model for classification, interpretable, but assumes linear relationships between features and the target.

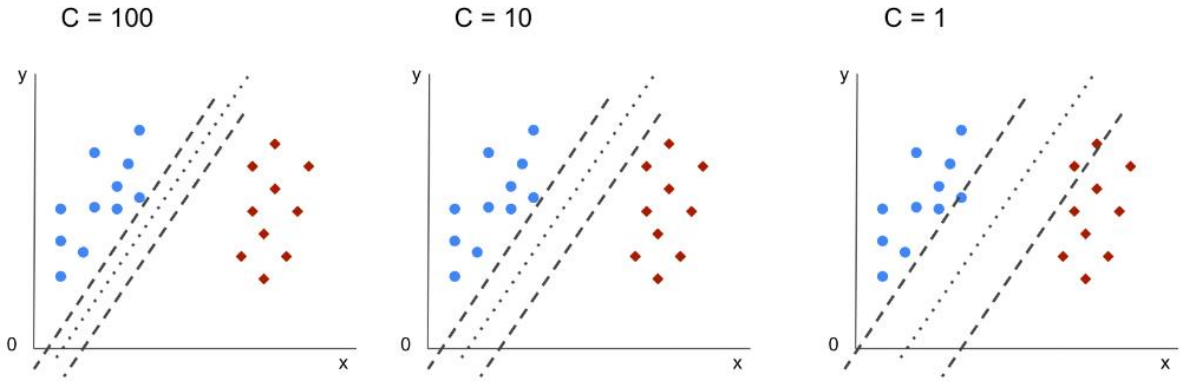


Figure 11. SVM Model

- **Gradient Boosting**: Suitable for capturing complex patterns, but may require more data and tuning.

3.2 Course Recommendation Using Similarity Matching

3.2.1 Course Data Preparation:

To develop an AI-powered application for personalized learning content, selecting and preparing a relevant course dataset is essential. We have identified several publicly available datasets that can support this objective:

Dataset 1: [Link]
A curated dataset containing detailed metadata about Python courses on Coursera, including ratings, enrollment numbers, university name, and course descriptions.

Dataset 2: [Link]
A dataset obtained via web scraping from Coursera, covering a similar set of attributes with slightly expanded scope.

Dataset 3: [Link]
A broader dataset aggregating online courses across multiple platforms, not limited to Python or Coursera alone.

To select the most appropriate dataset for our system, we will conduct a comparative analysis using our predefined criteria through a decision matrix.

Decision Matrix for choosing Programming course dataset:

Why	Criteria	Weight	Data 1	Data 2	Data 3	Reason
The official, trusted source from Coursera	Data Source Reliability	25%	9	8	7	1st got it from Coursera, 2nd also scrape, 3rd synthesized many sources
There are many attributes to analyze: title, rating, enrollments, skills...	Feature Completeness	20%	8	9	7	1st is informative; 2nd is basic enough; 3rd is both lacking and general
Knowing the year of collection helps assess the timeliness or relevance	Collection Year Clarity	10%	7	6	5	All 3 are unknown
Data is consistent, not missing, not significantly edited	Data Integrity	15%	8	7	6	1st is official, good integrity; datasets still need to be checked
Ready to use for analysis, little need for cleaning	Readiness for Analysis	10%	8	8	6	1st, 2nd are available; 3rd needs more data entry
Have clear reference documents and articles	Documentation Availability	10%	7	6	5	1st and 2nd have descriptions; 3rd simple general
Easy to download, clear license	Accessibility & License Clarity	10%	10	9	8	All are downloadable and free to use
		Total score	**8.2**	**7.6**	**6.5**	**Choose data 1**

Figure 12. Decision matrix of Course Content Analysis System dataset

Final chosen Dataset:

Course Content Analysis System dataset

https://www.kaggle.com/datasets/aayushsameershah/python-courses-on-coursera

This dataset contains information about Udemy learning python courses in various category (with a files of csv data contain information of 948 Python programming language courses in 2024)

Overview of the dataset

	Unnamed: 0	university	course	type	review	votes	students	difficulty
0	0	University of Michigan	Python for Everybody	SPECIALIZATION	4.8	225236.0	2400000.0	Beginner
1	1	Google	Python Foundations: Start Coding Today	PROFESSIONAL CERTIFICATE	4.7	21768.0	410000.0	Beginner
2	2	University of Michigan	Python Basics Bootcamp	SPECIALIZATION	4.7	15904.0	290000.0	Beginner
3	3	University of Michigan	Applied Data Science with Python	SPECIALIZATION	4.5	29358.0	730000.0	Intermediate
4	4	Universit√† di Napoli Federico II	Data Science con Python e R	SPECIALIZATION	NaN	NaN	NaN	Intermediate
...	...	...	...	...	...	...	...	...
944	944	University of Michigan	Master of Applied Data Science	DEGREE	NaN	NaN	NaN	NaN
945	945	University of Pennsylvania	Master of Computer and Information Technology	DEGREE	NaN	NaN	NaN	NaN
946	946	University of Illinois	Master of Business Administration (iMBA)	DEGREE	NaN	NaN	NaN	NaN
947	947	University of Illinois at Urbana-Champaign	Master of Science in Accounting	DEGREE	NaN	NaN	NaN	NaN
948	948	Coursera Project Network	Tensorflow Neural Networks using Deep Q-Learni...	GUIDED PROJECT	2.7	17.0	NaN	Advanced

949 rows × 8 columns

Figure 13 . Raw Python Programming Course dataset

Description of course dataset:

```
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  949 non-null    int64
 1   university  949 non-null    object
 2   course      949 non-null    object
 3   type        949 non-null    object
 4   review      713 non-null    float64
 5   votes       713 non-null    float64
 6   students    579 non-null    float64
 7   difficulty  933 non-null    object
dtypes: float64(3), int64(1), object(4)
```

Figure 14. Schema of Python programming Course dataset

Detail description:

1. Id (int): A unique identifier for each course entry.
2. course (string): The name or title of the course.
3. type (string): The category or subject area the course belongs to (e.g., Programming, Business).

4. review (float): The average rating or review score given by students, typically on a scale (e.g., 0–5).
5. votes (float): The number of individual ratings or votes the course has received.
6. students (float): The total number of students enrolled in the course.
7. difficulty (string): The difficulty level of the course, often labeled as Beginner, Intermediate, or Advanced.

3.2.2 Dataset Processing:

- Handle Missing data
- Remove irrelevant features that do not significantly impact on the course
- Create natural-language descriptions for each course

Data after Processing overview:

```
The course 'Python for Everybody' has a difficulty level of Beginner and a review score of 4.8 out of 5.
The course 'Python Foundations: Start Coding Today' has a difficulty level of Beginner and a review score of 4.7 out of 5.
The course 'Python Basics Bootcamp' has a difficulty level of Beginner and a review score of 4.7 out of 5.
The course 'Applied Data Science with Python' has a difficulty level of Intermediate and a review score of 4.5 out of 5.
The course 'Crash Course on Python' has a difficulty level of Beginner and a review score of 4.8 out of 5.
```

Figure 15. Python programming Course dataset after processing

948 course datasets are being transformed into course descriptions for future utilization.

3.2.3 Matching via Cosine Similarity:

Transforming both student profiles and course data into vector space. Embeddings are generated using a language model SentenceTransformers.

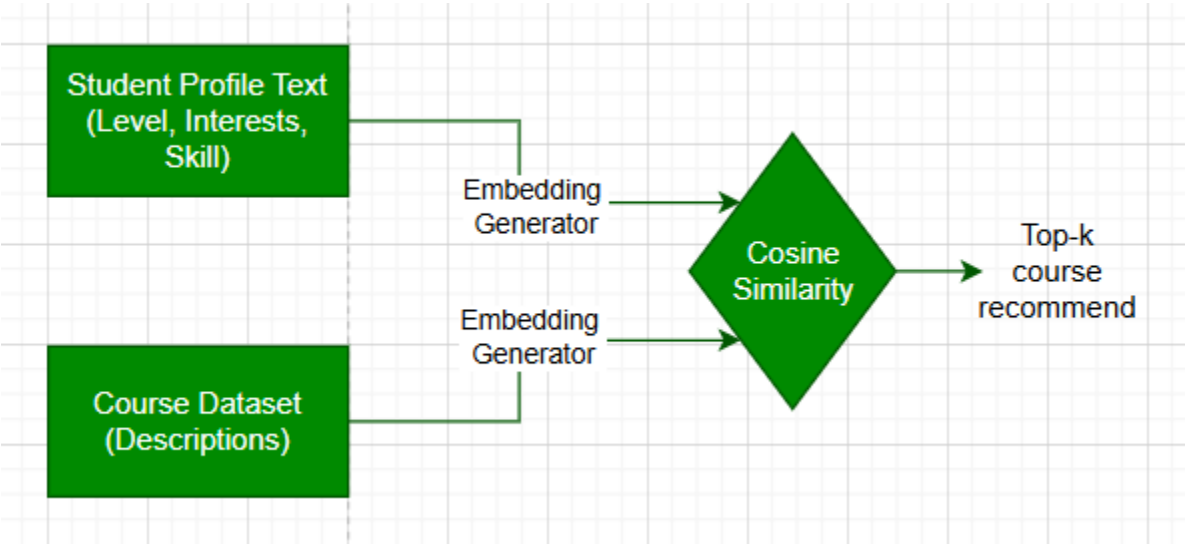


Figure 16 . Raw Python Programming Course dataset

Figure 16 illustrates the architecture of a course recommendation system designed to match student profiles with relevant courses based on semantic similarity. The system processes student and course data through an embedding-based approach, culminating in personalized course recommendations.

Embedded Student profile and Course Description

- Student Profile:

```
Based on the student's major Business, predicted to be at level Beginner
With description of themself as 'Interesting in Business Application want to learn more about Business'
```

Figure 17 . Student profile text for embedding

- Course Description:

```
The course 'Python for Everybody' has a difficulty level of Beginner and a review score of 4.8 out of 5.
The course 'Python Foundations: Start Coding Today' has a difficulty level of Beginner and a review score of 4.7 out of 5.
The course 'Python Basics Bootcamp' has a difficulty level of Beginner and a review score of 4.7 out of 5.
The course 'Applied Data Science with Python' has a difficulty level of Intermediate and a review score of 4.5 out of 5.
The course 'Crash Course on Python' has a difficulty level of Beginner and a review score of 4.8 out of 5.
```

Figure 18. Course Description text for embedding

- Courses recommendation based on similarity between Student Profile and Course Description:

```
["The course 'Advanced Classification Strategies' has a difficulty level of Advanced and a review score of 4.8 out of 5.",
 "The course 'Intro to Analytic Thinking, Data Science, and Data Mining' has a difficulty level of Intermediate and a review score of 4.1 out of 5.",
 "The course 'Data Analysis and Interpretation Capstone' has a difficulty level of Mixed and a review score of 4.7 out of 5."]
```

Figure 19 . Top-k courses recommended

3.3 RAG System Integration for Final Recommendation

This component improves the course suggestion system by combining retrieval-based relevance and generative reasoning using an LLM.

3.3.1 Overview of RAG (Retrieval-Augmented Generation)

The Retrieval-Augmented Generation (RAG) approach combines two key components to deliver personalized recommendations:

- The **Retriever** locates relevant course descriptions based on semantic similarity.
- The **Generator**, a large language model (LLM), uses this retrieved content to produce a natural-language recommendation tailored to the student.

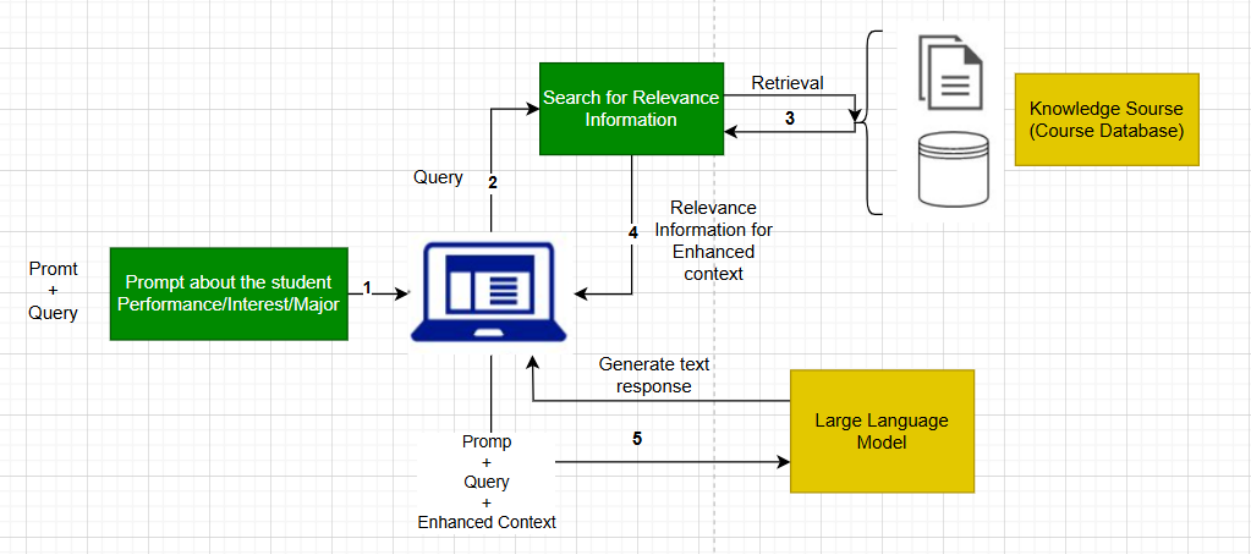


Figure 20 . Overview of RAG system

Once the student evaluation is completed, the resulting profile is transformed into a natural language query. This query is then converted into an embedding and matched against a vector index of course content to identify the most relevant documents. The top-n course entries are retrieved based on their similarity to the query. These retrieved documents are then incorporated into a prompt and passed to a large language model (LLM), which generates a tailored course recommendation.

Prompt

```
Based on the student's major Business, predicted to be at level Beginner
With description of themself as 'Interesting in Business Application want to learn more about Data Analyze in Business'
select the most suitable course from the list and explain why it is the best fit.
```

Figure 21 . Prompt text as input for LLM

Query

```
Python courses for beginner business students
interested in business applications
```

Figure 22. Query text to search for relevance information

Enhanced Content

```
'Python for Business' is for Beginner
'Code to Cash: Python for Business Growth' is for Beginner
'Python in Business' is for Intermediate
```

Figure 23. Enhanced text content retrieval from database

Prompt + Query + Enhanced Content as input for LLM

```
Business major, Beginner level.
Student is interest in: Business Application
Based on student major, level, interest of student, we choose some courses for them, pick one and explain why, these are the courses:
'Python for Business' is for Beginner
'Code to Cash: Python for Business Growth' is for Beginner
'Python in Business' is for Intermediate
```

Figure 24. Input prompt for the LLM

IV. Results and Evaluation

Objective Evaluation

4.1 Evaluation of Traditional AI Model

We evaluate the performance of our classification models—Support Vector Machine (SVM), Gradient Boosting, and Random Forest (RF)—using standard metrics, including **Accuracy**, **Precision**, **Recall**, **F1-score**, and the **Confusion Matrix**. These metrics provide a comprehensive assessment of how well each model performs in classifying the data.

- **SVM model:**

To determine the best parameters for the SVM model, a grid search was performed over a range of values for C, gamma, and the RBF kernel.

- **C** with values 0.1,1,10,100,10000.1, 1, 10, 100, 10000.1,1,10,100,1000
- **gamma** with values 1,0.1,0.01,0.001,0.00011, 0.1, 0.01, 0.001, 0.00011,0.1,0.01,0.001,0.0001
- **kernel** type set to 'rbf' (Radial Basis Function)

=> **Best parameters are**: C set to 10 and gamma set to 0.01, using the RBF kernel

Precision, Recall, and F1-Score Analysis for SVM Model:

	precision	recall	f1-score
1	0.71	0.91	0.80
2	0.88	0.58	0.70
3	0.86	1.00	0.92
accuracy			0.79
macro avg	0.82	0.83	0.81
weighted avg	0.81	0.79	0.78

Figure 25. Precision, Recall and F1-Score of SVM model

Random Forest:

To determine the best parameters for the Random Forest model, a grid search was performed over a range of values for the number of estimators, maximum depth, and minimum samples split.

- Number of estimators with values 50 and 100
- Maximum depth with values none and 5
- Minimum samples split with values 2 and

=> **Best parameters are:** maximum depth set to none, minimum samples split set to 2, and number of estimators set to 50.

Precision, Recall, and F1-Score Analysis for Random Forest:

	precision	recall	f1-score	support
1	0.91	0.91	0.91	11
2	0.92	0.92	0.92	12
3	1.00	1.00	1.00	6
accuracy			0.93	29
macro avg	0.94	0.94	0.94	29
weighted avg	0.93	0.93	0.93	29

Figure 26. Precision, Recall and F1-Score of Random Forest

Gradient Boosting

To determine the best parameters for the Gradient Boosting model, a grid search was performed over a range of values for the number of estimators, learning rate, maximum depth, and minimum samples split.

- Number of estimators with values 100 and 200
- Learning rate with values 0.1 and 0.05
- Maximum depth with values 3 and 5
- Minimum samples split with values 2 and 5

=> **Best parameters are:** learning rate set to 0.1, maximum depth set to 5, minimum samples split set to 2, and number of estimators set to 100.

Precision, Recall, and F1-Score Analysis for Gradient Boosting:

Gradient Boosting Classification Report:

	precision	recall	f1-score	support
1	0.73	1.00	0.85	11
2	0.88	0.58	0.70	12
3	0.83	0.83	0.83	6
accuracy			0.79	29
macro avg	0.81	0.81	0.79	29
weighted avg	0.81	0.79	0.78	29

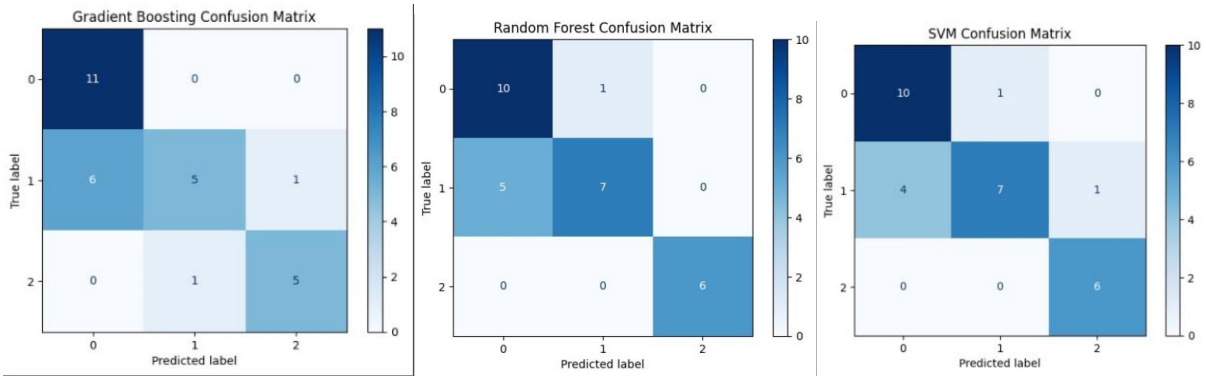
Figure 27. Precision, Recall and F1-Score of Gradient Boosting

Accuracy of each model

	Original Data	Synthesis Training data
SVM	69%	79%
RF	83%	93%
GB	72%	79%

Table 1: Accuracy of each model

Confusion Matrix of the all three model



a) **b)** **c)**

Figure 28. Confusion matrices for the Gradient Boosting, Random Forest, and SVM models, from left to right respectively.

Explanation/Comments on the results

Accuracy Improvement with Synthetic Data:

- All models show improved accuracy when trained on synthetic training data compared to original data: SVM (69% to 79%), RF (83% to 93%), and GB (72% to 79%). This suggests that synthetic data may better represent the underlying patterns, possibly due to balanced or augmented samples, enhancing model generalization on the 29-sample test set.

Confusion Matrix Insights:

- All three models (SVM, Random Forest, Gradient Boosting) perform exceptionally well for Label 0 (Beginner), with 10–11 correct predictions out of 29 samples, indicating robustness in identifying this class.
- Random Forest stands out with the highest number of correct predictions for Beginners (11) and maintains good accuracy for Advanced students (6), with no misclassifications into other labels, suggesting superior class separation.
- SVM shows a balanced performance across all levels but has the highest Intermediate misclassification (4 to Beginner), indicating potential overfitting or feature overlap.
- Gradient Boosting matches Random Forest for Beginners but struggles with Intermediate students (only 5 correct), possibly due to overemphasizing the dominant Beginner class during boosting iterations**.**

Why Random Forest Performs Best:

- Random Forest's ability to evaluate feature importance (e.g., GPA, study hours) across multiple trees likely helps it better distinguish between levels, especially for Advanced students, where it achieves perfect separation.

- Random Forest uses an ensemble of decision trees, which reduces variance and improves robustness by averaging predictions, leading to better generalization on the small dataset compared to SVM's single decision boundary or Gradient Boosting sequential error correction.
- Random Forest effectively handles the imbalanced nature of the dataset, minimizing overfit to the dominant Beginner class while maintaining accuracy for Intermediate and Advanced.

Limitations and Considerations:

- With only 145 samples, the models may overfit, especially with synthetic data, which could introduce bias if not representative of real student behavior.
- The confusion matrices show a tendency to misclassify label 2 as label 0, suggesting the models may struggle with intermediate performance levels, possibly due to insufficient training examples or feature overlap.

Test case for TraAI (testing with real user input)

These are 11 input features for classify Student Performance
1- Weekly study hours: (1: None, 2: <5 hours, 3: 6-10 hours, 4: 11-20 hours, 5: more than 20 hours)
2- Reading frequency (non-scientific books/journals): (1: None, 2: Sometimes, 3: Often)
3- Reading frequency (scientific books/journals): (1: None, 2: Sometimes, 3: Often)
4- Attendance to the seminars/conferences related to the department: (1: Yes, 2: No)
5- Attendance to classes (1: always, 2: sometimes, 3: never)
6- Preparation to midterm exams 1: (1: always, 2: sometimes, 3: never)
7- Preparation to midterm exams 2: (1: always, 2: sometimes, 3: never)
8- Taking notes in classes: (1: never, 2: sometimes, 3: always)
9- Listening in classes: (1: never, 2: sometimes, 3: always)
10- Cumulative grade point average in the last semester (/4.00): (1: <2.00, 2: 2.00-2.49, 3: 2.50-2.99, 4: 3.00-3.49, 5: above 3.49)
11- Expected Cumulative grade point average in the graduation (/4.00): (1: <2.00, 2: 2.00-2.49, 3: 2.50-2.99, 4: 3.00-3.49, 5: above 3.49)

With the following data input, the student should be classify accordingly (This is the ground true for classification)

	Beginner	Intermediate	Advanced
1	1-3	2-4	3-5

2	1	1-2	2-3
3	1	1-2	2-3
4	2	1-2	1-2
5	3	2-3	1-2
6	3	2-3	1-2
7	3	2-3	1-2
8	1	2-2	3
9	1	2-3	3
10	1-2	3-4	4-5
11	1-2	3-4	4-5

Table 2: Ground true table for evaluating Student Performance

Table 3 presents the accuracy results of the student level classification system, tested over 20 iterations across three levels: Beginner, Intermediate, and Advanced. The evaluation assesses the system's ability to correctly classify students based on their performance metrics, providing insights into its effectiveness for each level.

	Beginner	Intermediate	Advanced
Accuracy	100%	100%	80%

Table 3: Accuracy of student level classification

Comments:

- **Beginner Level:** The system achieved an accuracy of 100%, indicating perfect classification of students at the Beginner level across all 20 tests.
- **Intermediate Level**: The system also recorded an accuracy of 100%, demonstrating flawless identification of Intermediate students throughout the testing period.
- **Advanced Level**: The system attained an accuracy of 80%, reflecting a strong but less consistent performance in classifying Advanced students.

The student level classification system demonstrates outstanding accuracy for Beginner and Intermediate levels, achieving 100% in both cases. For the Advanced level, an

accuracy of 80% indicates a need for further investigation into potential factors affecting classification, such as outlier characteristics or insufficient representation in the dataset. Future improvements may involve enhancing feature sets or increasing the sample size for Advanced students to achieve consistency across all levels.

4.2 Course Recommendation Quality

Testing Cases

We evaluate the recommendation system using two methods:

- **Precision@10** measures how well the top 10 recommended courses match the student's topic of interest.
- **Alignment Check** assesses how many of those courses are appropriate for the student's current skill level.

Scenario Description: Each student level is tested across multiple areas of interest using a variety of prompts. These **prompts differ in both keyword selection and sentence structure** to reflect diverse ways students might express their learning goals. **Each test case is evaluated five times, and the average result is used for analysis.**

Beginner Level:

Test Case	Student Interest	Expected Keywords	Precision@10	Alignment Check
1	Machine Learning	ML, machine learning, deep learning	1	0
2	Data Structures	algorithms, sorting, trees, graphs	1	0.34
3	BlockChain	blockchain, smart contracts, web3	1	0.36
4	Web Development	html, css, javascript, react, web	1	0.34
5	Business	Business application, analytics, dashboard	1	1

6	SQL&noSQL	sql, nosql, mongodb, mysql	1	0.42
7	Just start to learn Python	intro, basics, beginner, getting started	1	0.94
8	Data Science	data science, pandas, numpy, data analysis, python, visualization	1	0.84

Table 4: Test cases for recommending courses to a Beginner Student.

- **Precision@10:** The scores are consistently high (mostly 1.0), which indicates that the recommendation system is doing a great job in matching the student's topic of interest.
- **Alignment Check:** There's a noticeable drop in alignment when hard or advanced topics are selected by students with likely beginner-level profiles:
  - Data Structures, BlockChain, Web Development are all 3 scores only 0.3 and SQL topic is 0.4 showing the system includes too many advanced courses for these interests.
  - Just start to learn Python and Data Science perform much better 0.9 and 0.8 respectively, indicating some moderate filtering by level, but not fully accurate. This proves that our system performs best when the student's prompt is aligned with their actual level.

Intermediate Level

Test Case	Student Interest	Expected Keywords	Precision@10	Alignment Check
1	Machine Learning	ML, machine learning, deep learning	1	0.78
2	Data Structures	algorithms, sorting, trees, graphs	1	0.7
3	BlockChain	blockchain, smart contracts, web3	1	0.74

4	Web Development	html, css, javascript, react, web	1	0.62
5	Business	Business application, analytics, dashboard	1	0.84
6	SQL&noSQL	sql, nosql, mongodb, mysql	1	0.56
7	Just start to learn Python	intro, basics, beginner, getting started	1	0.42
8	Data Science	data science, pandas, numpy, data analysis, python, visualization	1	0.5

Table 5: Test cases for recommending courses to an Intermediate Student.

- **Precision@10**: The scores remain consistently high across all test cases, with a perfect 1.0 for each student interest, indicating that the recommendation system excels at matching the student's topic of interest with the expected keywords.
- **Alignment Check**: The system's alignment with the student's level varies noticeably across different topics.
  - Areas such as Machine Learning (0.8), Data Structures (0.7), Blockchain (0.7), Web Development (0.6), Business (0.8) demonstrate moderate to good filtering, though some recommended courses may still exceed the student’s current skill level.
  - SQL/NoSQL and Data Science, both with scores of 0.5, indicate challenges in separating beginner-friendly content from more advanced material.
  - The lowest alignment is observed in the "Just start to learn Python" case, with a score of 0.4, highlighting that the system has difficulty consistently delivering truly beginner-level content.

Advanced Level:

Test Case	Student Interest	Expected Keywords	Precision@10	Alignment Check

1	Machine Learning	ML, machine learning, deep learning	1	0.46
2	Data Structures	algorithms, sorting, trees, graphs	1	0.2
3	BlockChain	blockchain, smart contracts, web3	1	0.2
4	Web Development	html, css, javascript, react, web	1	0.44
5	Business	Business application, analytics, dashboard	1	0.2
6	SQL&noSQL	sql, nosql, mongodb, mysql	1	0.1
7	Just start to learn Python	intro, basics, beginner, getting started	1	0.2
8	Data Science	data science, pandas, numpy, data analysis, python, visualization	1	0.3

Table 6: Test case for recommending courses to an Advanced Student.

- **Precision@10**: Precision@10: The scores are uniformly high at 1.0 across all test cases, demonstrating that the recommendation system effectively matches the student's topic of interest with the expected keywords
- **Alignment Check**: The alignment with the student's level is significantly lower and shows considerable inconsistency
  - Topics like Machine Learning (0.46), Web Development (0.44), and Data Science (0.3) have moderate alignment, suggesting some level filtering, but still include a notable number of mismatched courses.
  - The poorest alignment is observed with SQL/NoSQL (0.1) and "Just start to learn Python" (0.2), indicating a strong tendency to recommend advanced or inappropriate courses for beginners.

- Data Structures (0.2), Blockchain (0.2), and Business (0.2) exhibit poor alignment, while Machine Learning (0.46), Web Development (0.44), and Data Science (0.3) show slightly better but still suboptimal results.
- This pattern indicates that the system struggles with level-appropriate filtering, which could be attributed to an insufficient variety of courses at each level rather than a complete failure of the algorithm.

4.3 RAG System Output Quality

To assess the performance of the Retrieval-Augmented Generation (RAG) system, we evaluate both the **retrieval quality** and the **generation quality**.

Retrieval Evaluation:
As mentioned above in **section 4.2**
Generation Evaluation:
To evaluate the Generation of the GenAI, I will use **semantic similarity** to measure how closely the generated responses align with the reference answers in terms of meaning. This approach allows for a more flexible and accurate assessment than traditional word-overlap metrics, as it captures the underlying intent and semantics of the responses, even when different wording is used. **Sentence embeddings and cosine similarity** will be applied to quantify this alignment **for the generate and the reference response.**

Test case	Context	Semantic Similarity Score
1	**Profile text:** Business major, want to learn Data Science Intermediate level.	**0.68**
	Top courses: Top 10 Business courses with Data Science, recommended by the system	
2	**Profile text:** Programming major, want to learn Machine Learning Intermediate level.	**0.63**
	Top courses: Top 10 Programming courses with Machine Learning, recommended by the system	
3	**Profile text:** Programming major, want to learn Data Structures, Intermediate level.	**0.68**

	Top courses: Top 10 Programming courses with Data Structures, recommended by the system	
4	**Profile text:** Programming major, Just start to learn Python, Beginner level.	**0.57**
	Top courses: Top 10 Programming courses with Python Introduction, recommended by the system	
5	**Profile text:** Programming major, want to learn Data Science, Beginner level.	**0.64**
	Top courses: Top 10 Programming courses with Data Science, recommended by the system	
6	**Profile text:** Programming major, want to learn Business, Beginner level.	**0.61**
	Top courses: Top 10 Programming courses with Business, recommended by the system	
7	**Profile text:** Programming major, want to learn Machine learning, Advanced level.	**0.69**
	Top courses: Top 10 Programming courses with Business, recommended by the system	
8	**Profile text:** Programming major, want to learn Web Development, Advanced level.	**0.7**
	Top courses: Top 10 Programming courses with Web Development, recommended by the system	
9	**Profile text:** Business major, want to learn Data Science, Advanced level.	**0.67**
	Top courses: Top 10 Programming courses with Data Science, recommended by the system	

Table 7: Evaluation of Generated Responses by Student Level

The semantic similarity scores generally range from moderate to high (0.57–0.7), with the best performance observed for Advanced-level profiles (e.g., Test Cases 7 and 8) and the weakest for Beginner-level profiles (e.g., Test Case 4). This suggests the system excels when the level and topic are well-defined and advanced, but it may struggle with Beginner-level recommendations, possibly due to fewer tailored courses or less precise level

filtering. Improving the dataset with more Beginner-specific courses or refining the embedding process could enhance consistency across all levels.

Subjective Evaluation

The system performs reliably when applied to real-world student data, especially in cases where the student's profile does not include outlier characteristics — such as doing everything correctly, studying diligently, actively participating in class activities, yet achieving low academic scores. In typical scenarios, the system is able to recommend suitable courses that align with the student's current level and learning needs.

For intermediate-level students, the system tends to suggest more advanced and appropriate courses. Conversely, for advanced students, it avoids recommending beginner-level material. For beginners, while the recommendations may be limited due to a smaller pool of suitable high-quality courses, the system still attempts to guide learners toward foundational topics.

The generative AI component generally provides accurate course suggestions, including relevant topics and well-justified reasons for taking those courses. This ensures that learners not only receive appropriate recommendations but also understand the value and purpose of the suggested learning paths.

V. Future Work

Enhancing User Background Analysis

Moving forward, our project aims to expand the system into a more comprehensive Smart Recommendation System. This evolution will involve collecting a wider range of user background information, including educational history, professional experience, learning preferences, and long-term career goals. By integrating these diverse data points, the system will provide a more nuanced understanding of each user, enabling personalized and effective learning pathways.

Advanced Recommendation Algorithm

The current system utilizes a Large Language Model (LLM) to recommend Python programming courses based on a limited dataset. Future work will focus on developing a smarter algorithm that not only selects top-k courses but also dynamically adapts to

individual user profiles. This will involve leveraging machine learning techniques to analyze user interactions and feedback, ensuring continuous improvement in recommendation accuracy.

Personalized Learning Routes

To cater to the varied needs of users, the system will recommend tailored learning routes instead of recommended courses. We do that by leveraging an expanded dataset and advanced personalization techniques. This will involve analyzing a wide range of user-specific factors, including educational background, professional experience, skill level, learning pace, preferred learning styles (e.g., visual, auditory, hands-on), and career aspirations. Based on this analysis, the system will design customized learning paths that align with each user's unique profile and goals.

References

[1] *Yilmaz, N. & Şekeroğlu, B. (2019). Higher Education Students Performance Evaluation [Dataset]. UCI Machine Learning Repository.* https://doi.org/10.24432/C51G82.
[2] *Chen, Sun, Wang, Zhao, Song & Zhai (2025). Machine Learning-Driven Student Performance Prediction for Enhancing Tiered Instruction. arXiv, Feb 2025.*
[3] *Mustafa Yağcı (2022). Educational data mining: prediction of students' academic performance using machine learning algorithms. Smart Learning Environments, 9:11.*