EduBot: LLM-Based Academic Support Chatbot

Shady Ali[1], Adrian Gamal[2], Ahmed Sameh[3], and Nour Hany[4]

[1]College of Science and Engineering, University of Minnesota, Twin Cities
ahme0971@umn.edu
[2]College of Science and Engineering, University of Minnesota, Twin Cities,
Gamal008@umn.edu
[3]College of Science and Engineering, University of Minnesota, Twin Cities,
sameh002@umn.edu
[4]College of Science and Engineering, University of Minnesota, Twin Cities,
hany0003@umn.edu

December 2024

Abstract

EduBot is an AI-powered academic support chatbot that combines Large Language Models (LLMs) and Machine Learning Models (MLMs) to assist students in improving their study habits, predicting exam scores, and enhancing their academic performance. The system features a three-tier architecture, including a user-friendly interface, intelligent processing modules, and a robust database. The LLM identifies user intent, extracts relevant information, and provides personalized study recommendations, while the MLM predicts exam scores based on factors like study hours, motivation, and attendance. EduBot stores user data securely using Azure Flexible Database to support model training and personalized responses.

Keywords: Large Language Models, Machine Learning, Chatbot, Database, Education

Contents

List of Abbreviations

LLM	Large Language Model
MLM	Machine Learning Model
GB	Gradient Boosting Regression Model
GB	Extreme Gradient Boosting Regression Model
SVR|SVM	Support Vector Machine Regression Model
ET	Extra Trees Regression Model

Chapter 1

Introduction

1.1 Background

Over the past 3-4 years, chatbots have gained more and more popularity among people. Specifically speaking, it is mainly Language Model-based chatbots who caused this for their never-seen before ability of interpreting humans' language and responding to most queries smoothly and accurately.

The main architecture that revolutionized chatbots and introduced language models in their current form is Transformer (Vaswani [2017]) with the self-attention mechanism, which made a breakthrough in machine translation, text summarization, and language generation (Zhao et al. [2023]). Large language models generally work by training them to predict next words in sentences over large scale corpora (Floridi and Chiriatti [2020]).

Large Language Models became widely used and utilized across different industries (Kasneci et al. [2023]) whether by directly interacting with users in chatbots, using them in business software and more. Some companies customize pre-trained models on new data to increase their performance on their domain-specific queries, others could potentially use prompting techniques to help the give the models more context on the user query for better responses (Liu et al. [2023]).

Since customizing and fine-tuning models can be in most cases quite expensive, most services offering customized models are pricey, specially for students. And language models cannot actually make mathematically-based predictions like classical machine learning models. We propose EduBot, a simple chatbot that utilizes both LLMs and Machine Learning to provide academic and productivity support to students.

1.2 Problem statement

Over the past 4 years, using chat-bots and Large Language Models (ChatGPT, Gemini, Claude, etc.) with different kinds has become more popular and mainstream between people across different disciplines and fields. Students across various level academic levels tend to use them more often as such chat-bots can help with their learning process, assignments, projects, and giving them tips and support to increase their academic performance. However, most chatbots have some shortcomings:

1. They cannot give statistically based predictions for test scores as an example, as LLMs cannot reason in general and, generally speaking, are just probabilistic next word predictors.

2. Most LLMs aren't trained for a specific domain and are more general case usage, so most answers they generate have tendency to be broad and general in meaning, words, etc. And most domain specific LLMs have pricey subscriptions which would be difficult for students to afford with little to no presence of LLMs trained or fine-tuned specifically for providing academic-assistance to students.

1.3 Aims and objectives

1.3.1 Results Prediction

If a student asked for a test score prediction and provided related details such as their study per week time, teacher and course's quality, previous scores, their personal motivation level, sleep hours, etc, EduBot uses a Machine Learning Model that utilizes such features to make a statistically accurate prediction of the students' results and scores in the future.

1.3.2 Academic Performance & Productivity Improvement

EduBot utilizes a powerful LLM which is continuously fine-tuned specifically to give each student personalized recommendations based on their desired goals and targets, whether it is to help them pass through a course, ace a course, or even assistance with their thesis!

1.4 Solution approach

EduBot utilizes LLM-based module to recognize user intentions and queries, respond to their questions efficiently using few-shot learning and other techniques, and extract useful information from the user's input. EduBot utilizes MLM Module for test scores prediction based on specific information such as attendance percentage, study time, and more using Machine Learning models like Gradient Boosting and Support Vector Machines. Finally, EduBot has a dedicated Database to store users' information and chat history

1.5 Summary of contributions

- **Shady Ali:**
    1. LLM Module
    2. MLM Module
    3. Data Preprocessing
    4. System Analysis & Design
    5. Report & Presentation
- **Adrian Gamal:**
    1. Front-End
    2. Controller
    3. Modules Integration
    4. System Analysis & Design
    5. Report & Presentation

- **Ahmed Sameh:**
    1. Database & Database Access Layer
    2. Database Controller
    3. Report & Presentation
    4. System Analysis & Design
    5. Data Preprocessing
- **Nour Hany:**
    1. Database & Database Access Layer
    2. Database Controller
    3. Report & Presentation
    4. System Analysis & Design
    5. Data Preprocessing

Chapter 2

Methodology

2.1 System Overview

We designed EduBot to for the main functionality of assisting students in their learning journey, which includes things such as productivity improvement, Explanation of certain scientific topics, assistance in understanding and solving assignments, and maybe predicting future exam scores given specific information to know whether they are on the right track for success or maybe they could improvise their methods and habits.
We also plan to include more functionalities for academic advisors so that they can view their students' study habits and important information and get suggestions on how to help them out.
Hence, EduBot architecture is divided into 3 layers including two main components satisfying the use-cases mentioned above:

- LLM Module: This module is the mind of the chatbot, powered by a State-of-Art Large Language Model. It analyzes and responds to the users' queries with high accuracy using several prompting techniques implemented in-code.
- MLM Module: This module is contains a powerful machine learning model which enables EduBot to make accurate predictions for future exams and tests' scores.

We implemented a 3-Tier Architecture, having the LLM and MLM modules in the business logic layer, great and simple front-end for user interactions, and a powerful database to store and utilize user's information and chat history to find useful insights for future versions of EduBot.

2.2 System Analysis

We started with the system analysis to know what exactly we want to achieve and what functionalities we are aiming to provide. The following diagrams illustrate what we reached so far for the functional requirements of EduBot.

2.2.1 EduBot Use Cases

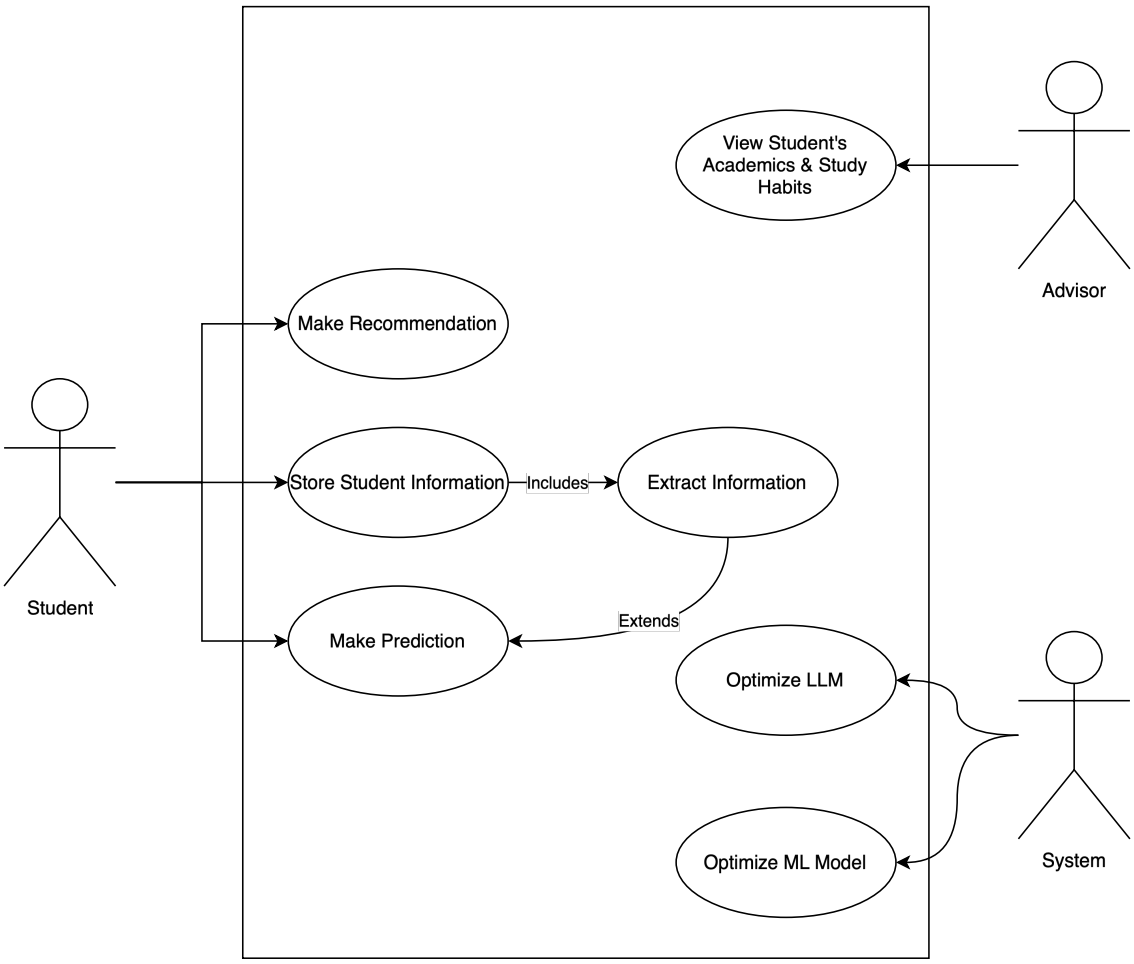


Figure 2.1: Use Case Diagram illustrating the main use cases of EduBot

From figure[2.1] we can see 7 main functionalities of EduBot with main focus on students.
EduBot can provide students with:

1. Request a recommendation: EduBot can provide recommendations and assistance for students in their studies, or for improving their study habits and productivity.
2. Store student information: EduBot can store students information to give them more personalized experience, this includes the process of extracting the information from the student's messages or by directly requesting it if needed.
3. Make prediction: EduBot can provide exam and test scores predictions for the students given specific information like their study hours, attendance percentage, etc. This might also need EduBot to extract information from the student's messages if needed.

EduBot can provide academic advisors with:

1. View student's academics & study habits: EduBot can provide the academic advisor with their student's academics and their study habits and also provide some suggestions to help their students out.

EduBot should continually improve it's performance by utilizing the available in its database to optimize the Large Language Models and Machine Learning Models used.

2.2.2 Activity Flow

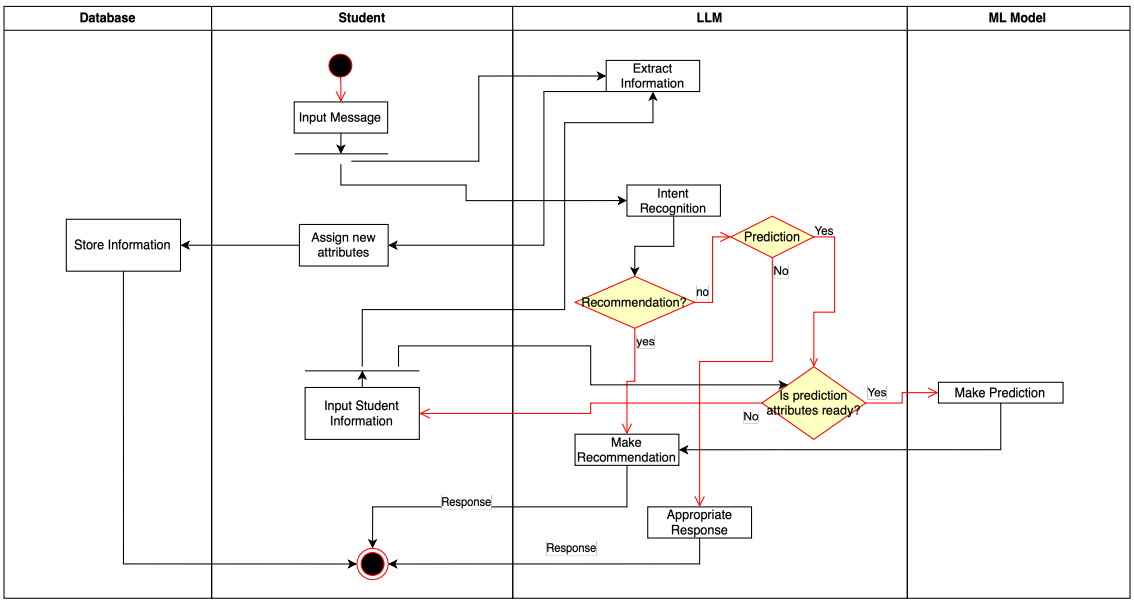


Figure 2.2: Activity Diagram illustrating main use cases for students

From figure[2.2] we illustrate the main use cases of EduBot for students in more details with the in-system flow for each use case. There are 4 swimlanes representing which action is done by whom or where.

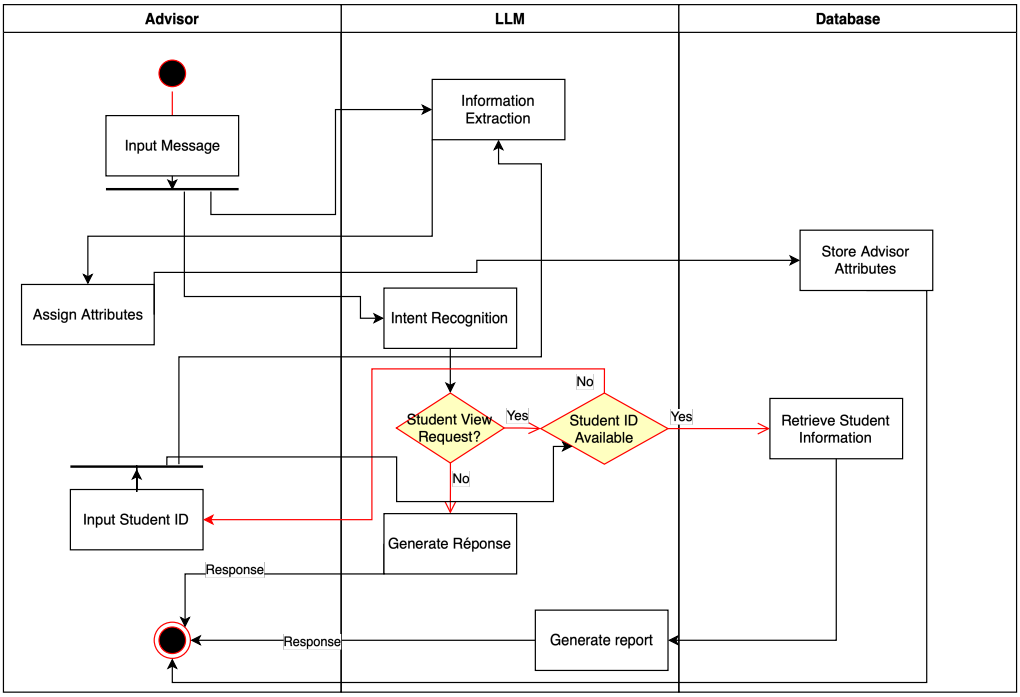


Figure 2.3: Activity Diagram illustrating main use cases for academic advisors

From figure[2.3] we illustrate the main activity flow in the case that an academic advisor is using EduBot.

2.3 System Architecture & Design

We utilized a 3-Tier Architecture as illustrated in figure[2.4]:

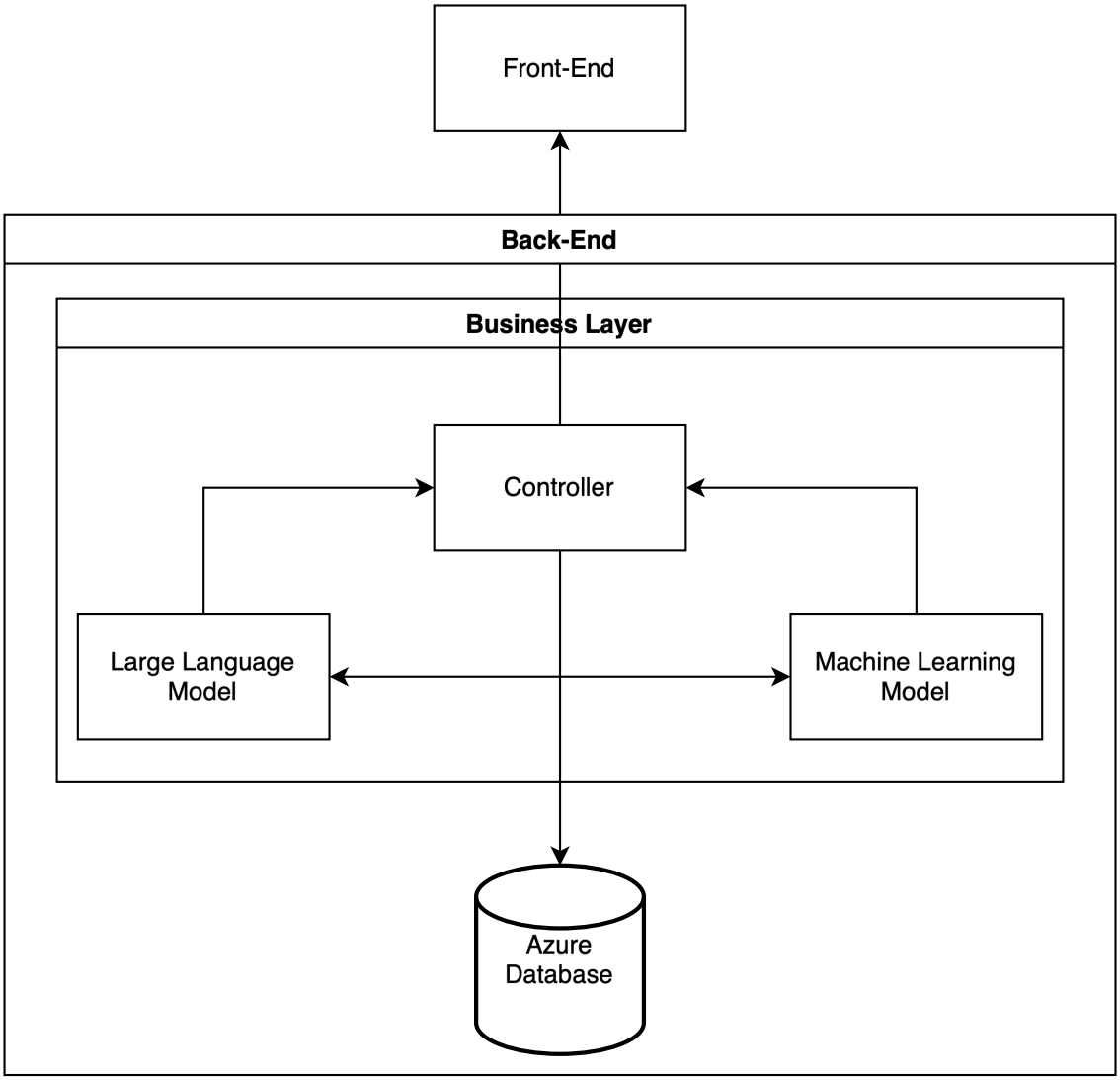


Figure 2.4: EduBot's Architecture

Database Controller (Data Access Layer)

The Database Controller serves as the interface between the application and the database, enabling seamless communication and data management. It handles all operations needed for EduBot, ensuring the integration of the EduBot system's back-end components with the underlying database.

Responsibilities:

- Establish and maintain a secure connection to the Azure database.
- Perform CRUD (Create, Read, Update, Delete) operations, if needed, for entities such as Users, Student Profiles, Course Details, and Chat Histories.

- Provide efficient methods for reading data, including chat histories and training datasets, using SQL views.

2.4 Front-End (User Interface)

The frontend is written in ReactJS as a portable component that can be integrated into any educational website that wishes to provide its students with a smart educational bot. It is lightweight and presents a minimal rendering overhead on the base website. It is separated from the backend and the core processing and communicates with the backend through HTTP requests.

2.5 Machine Learning Model

Before we talk about the Machine Learning Model, we must start by the dataset we used for training. We initially used Guyn [2021] Dataset which is a synthetically generated dataset that have several features to predict exam scores like sleeping hours, studying hours, motivation level, teacher quality, and more. However, as promising as it seemed, the quality of the data was really poor with low correlations, seemingly inconsistsent patterns across the data points. So even after several cleaning, preprocessing, and sampling efforts, most models couldn't generalize on it well.

Feature	Value
Study Hours per Week	Numerical
Sleep Hours per Night	Numerical
Previous Exam Scores	Numerical (0-100)
Motivation Level	Categorical (Low-Medium-High)
Class Attendance	Numerical (Percentage)
Teacher Quality	Categorical (Low-Medium-High)
Resource Access	Categorical (Yes-No)
Extracurricular Activities	Categorical (Yes-No)
School Type	Categorical (Public-Private)
Peer Influence	Categorical (Negative-Neutral-Positive)
Learning Disabilities	(Yes-No)
Distance from Home	(Near-Moderate-Far)
Physical Activity	Categorical (Yes-No)
Tutoring	Categorical (Yes-No)
Grade	Numerical (0-100)

Table 2.1: Dataset used for modeling

Unfortunately, we couldn't find other alternatives for this dataset easily, until we found Kharoua [2021] dataset, it's much smaller and have less features, but it was more explainable and models were able to generalize on it better.
Onto the Machine Learning Models used, we used a stacking algorithm of 4 models, 3 base models and 1 meta model. Stacking algorithm generally works by using multiple models and deciding which one works best on what pattern of the data.

Stacking algorithm uses the base models in fitting to the training data while the meta model learns how to best combine the base models predictions.

Feature	Value
Study Hours per Week	Numerical
Class Absence	Numerical (Percentage)
Extracurricular Activities	Categorical (Yes-No)
Physical Activity	Categorical (Yes-No)
Tutoring	Categorical (Yes-No)
Grade	Numerical (0-100)

Table 2.2: Dataset used for modeling

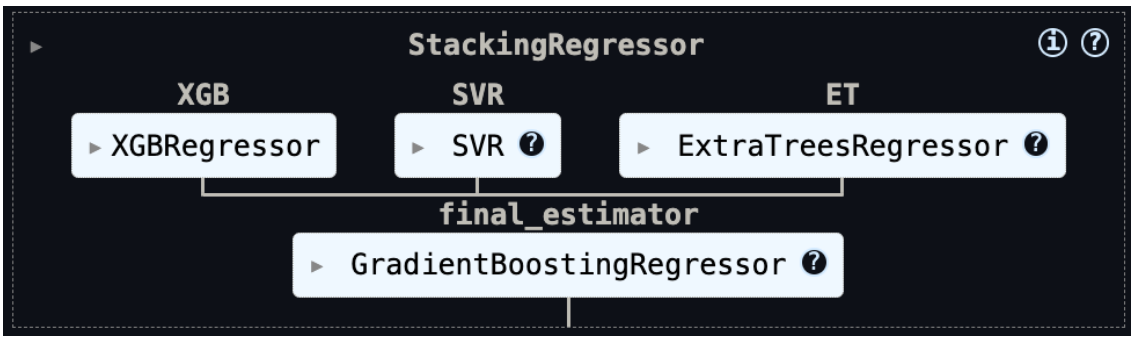


Figure 2.5: Stacking architecture used

As in figure[2.5], we used Extreme Gradient Boosting Regressor, Support Vector Machine Regressor, and an Extra Trees Regressor as base models. And a Gradient Boosting Regressor as the meta regressor. As you might noticed we did not just use stacking, but most models are also use ensemble methods like boosting (XGB & GB) and bagging (ET).

Boosting works by training several estimators and different samples of the training data with each estimator sequentially focusing on the mistakes of the previous estimator, while bagging uses multiple estimators on different samples of the training data and they all vote on the final prediction.

Finally, SVR works by using a specific kernel function to map the features to higher-dimensional space to capture non-linear relationships better. It then fits the regression hyper-plane or line on the data points.

2.6 Large Language Model

As we decided to start with using an LLM through an API and not locally, we use llama-3.3-70b-versatile using Groq's API. llama is an open source LLM architecture by Meta. And Llama 3.3 70B AI [2024] is their latest model which offers similar performance compared to Llama 3.1 405B model with a fraction of the cost and size.

The LLM module has three main, overlapping tasks:
Intention recognition: as the mind of the chatbot, we use the LLM to recognize the intention behind the user's query, we decided to divide the intentions into 3 categories:

1. Prediction: the user would be asking for a prediction in some way, this would lead us to validate that we have the information needed to make the prediction and then make a call for the MLM module to make the prediction.

2. Extraction: the user might sometimes provide a message that has useful information for

us, such as their name, age, and also academics and information that we would need later for making predictions and personalized recommendations.

3. Recommendation: instead of predictions or providing useful information for EduBot, the user may just ask for recommendations, suggestions, or maybe other unexpected queries.

We guide the LLM with system prompting to be able to output the correct intention behind the user's query, we will discuss the prompting techniques we use in each scenario in [2.6.1].

2.6.1 Few/Zero-Shot Prompting

Few-shot prompting is an In-Context Learning technique used with LLMs. In this approach, we provide demonstrations within the prompt to guide the model towards improved performance. These demonstrations serve as conditioning for subsequent examples, where we expect the model to generate responses.

- For Intention Recognition, we provide the LLM by the following prompt:

```
messege =
f"""Given the following text, say prediction if the user wants to get a prediction,
say extraction if the user is providing information or a self report about themself, say recommendation for any other context: \n
User input: \"{user_input}\"

Follow these rules:
1-For prediction, be sure that the user actually wants you to predict something, grade, score, test, etc.
2-For extraction, the user should be providing some information about themselves, like their name, gender, age, academic information about them, etc.
3-For recommendation, the user should either want a recommendation from you to help them at something, or a general message which you can handle yourself with no need for extraction or making a prediction.
4-you must only respond with 1 word of the three indicated above.

Check the following messages and the intent behind them as an example to what you should recognize as what where 'In' is the input message and 'Out' is the Intent recognized from the message:

(1)
In: Hello, my name shady ali, I'm 24
Out: extraction

(2)
In: What do you think my next score will be?
Out: prediction
```

```
21
22 (3)
23 In: I am really at loss of what to do to improve my study habits and
   ↪ grades, what can I do?
24 Out: recommendation
25
26 (4)
27 In: Hello
28 Out: recommendation
29 """
```

- For Information Extraction, we use the following prompt:

```
1  # Few-Shot Prompt
2  prompt = f"""
3  You are the best model to extract data from raw texts to desired Json
   ↪ format. you will be provided user messages that you need to
   ↪ extract specfic information from it into JSON format.
4  You are tasked with converting the given text into a JSON object with
   ↪ the specified structure.
5  Please follow these guidelines:
6
7  1. - If the provided text is empty or does not contain any relevant
   ↪ information, return the JSON structure with all values as an
   ↪ None.
8  - If the provided text contains multiple instances of the same
   ↪ information (e.g., multiple names), use the one that relates to
   ↪ the user the most and not anyone else's.
9  - If the provided text contains conflicting information (e.g.,
   ↪ different ages), use the one that relates to the user the most
   ↪ and not anyone else's.
10
11 2. Extract relevant information from the provided text and map it to
   ↪ the corresponding keys in the JSON structure.
12
13 3. If a particular key's value is not found in the given text, leave
   ↪ the value as an None.
14
15 4. Do not include any additional information or formatting beyond the
   ↪ requested JSON object.
16
17 5. Make sure to transform each of "motivation_level",
   ↪ "teacher_quality", "resource_access" values into (low, medium,
   ↪ high) categories.
18
19 6. You must follow the given JSON structure exactly, including the
   ↪ key names.
20
```

```
7. remember to specify the gender.

8. Make sure to make the values of "age", "study_hours_per_week",
↪  "sleep_hours_per_night", "previous_exam_scores",
↪  "class_attendance" as numbers only.

9. If values of "age" or "sleep_hours_per_night" are zero, leave the
↪  value as an empty string.

10. Make sure to transform the values of
↪  "extracurricular_activities", "tutoring", "physical_activity",
↪  into (0, 1) categories such that 0 means 'no' and 1 means 'yes'.

11. Make sure to transform the values of "school_type" into (private,
↪  public) categories.

12. Make sure to transform the values of "peer_influence" into
↪  (positive, negative, neutral) categories.

13. Make sure to transform the values of "distance_from_home" into
↪  (near, moderate, far) categories.

14. Make sure to stick to the given format and value categories.

15. Make sure to transform the values of "learning_disabilities" into
↪  (yes, no) Categories.



Here are some examples, I'm gonna provide you the raw_texts and json
↪  structure.
raw_texts:
1-Hey, I’m Shady, 21, male. I go to a private school and study about
↪  20 hours a week. I usually get 6 hours of sleep a night, and my
↪  last exam score was 85. My motivation level is medium, and I
↪  attend 90% of my classes. Teachers are great, and resource access
↪  is decent. I’m involved in extracurricular activities and do
↪  about 6 hours of physical activity weekly. The school is far from
↪  home, but I manage. Peer influence is neutral, and thankfully, I
↪  don’t have any learning disabilities. I take tutoring sessions
↪  yes.
2-Resources are decent-nothing fancy, but they work. I get about 6
↪  hours of sleep most nights, which isn’t ideal but manageable. My
↪  last exam score was 85%.
3-Classes are alright! I attend regularly, maybe 85-90%. My study
↪  schedule’s flexible, but I squeeze in a few hours daily. Sleep’s
↪  hit-or-miss-usually 5-6 hours.
```

```
4-Hi, I'm Shady, a 21-year-old male. I study around 25 hours weekly, get 6 hours of sleep nightly, and scored 87 in my last exam. Motivation is at 8/10, with decent teacher support and 92% attendance.
5-I've been trying to study consistently (about 3-4 hours a day), and motivation's not bad! Teachers are okay, and I sleep about 6 hours. Last exam? 83%.
6-Hey, I've been studying around 20 hours weekly-pretty manageable. Ahmed says he studies way less, but his motivation is crazy high! My last exam score was 85%.
7-Classes are alright. I think Salma mentioned she's getting better sleep-like 7 hours nightly. Me? Still around 6. Teachers are supportive, though!
json_structure:{json.dumps(self.user_attributes.model_json_schema(), indent=2)}
    """
```

The JSON structure for extraction provided for the LLM is as follows:

```python
class user_attributes(BaseModel):
    name: Optional[str]
    age: Optional[int]
    gender: Optional[str]
    study_hours_per_week: Optional[int]
    sleep_hours_per_night: Optional[int]
    previous_exam_scores: Optional[int]
    motivation_level: Optional[str]
    class_attendance: Optional[int]
    teacher_quality: Optional[str]
    resource_access: Optional[str]
    extracurricular_activities: Optional[int]
    school_type: Optional[str]
    peer_influence: Optional[str]
    learning_disabilities: Optional[str]
    distance_from_home: Optional[str]
    physical_activity: Optional[int]
    tutoring: Optional[int]
```

- For prediction:

```python
# Now we can provide a prediction
prediction = self.mlmodel.predict(self.user.prediction_attributes())

# Wrap up the prediction in a message
message = f"""
given the chat history so far and the user's following information along with this exam score prediction ({prediction}) for them: \n
user's attributes: {self.user.__dict__} \n
```

```
8
9  can you announce the prediction to the user provided along with
   ↪  specific tips and recommendations to improve themselves more
   ↪  given their history and what you know so far?
10 """
```

- Finally, for recommendations and the general messages, we actually use Zero-Shot Learning here and we implement this message as the starting system message in the chat so the LLM follows it in any response it does:

```
1  start_system_message = """
2  You are the best model to assist students in improving their
   ↪  productivity, study habits, and helping them in their studies.
3  you are the best at doing the following:
4  1- Making recommendations: you provide effective suggestions and
   ↪  assistance in their studies and productivity and your suggestions
   ↪  with the information you know about them if provided.
5  2- Sometimes you will be provided with an exam score prediction, you
   ↪  need to explain possible reasons behind such score given what you
   ↪  know about them and how they can improve or what to do next.
6
7  when you are asked for recommendation, you must be specific for the
   ↪  studying or productivity techniques you might suggest them,
   ↪  citing where you know such techniques from if possible, and maybe
   ↪  suggesting that they check the source themselves.
8
9  Here are some possible books, authors and people who provide advices
   ↪  on productivity and studying:
10
11 1- Atomic Habits Book, by James Clear.
12
13 2- The Slight Edge, by Jeff Olson.
14
15 3- The 7 Habits of Highly Effective People, by Stephen Covey.
16
17 4- Eat That Frog!: 21 Great Ways to Stop Procrastinating and Get More
   ↪  Done in Less Time, by Brian Tracy.
18
19 5- Huberman's Lab, a podcast by Andrew Huberman.
20
21 You do not have to use these examples all the time or even mention
   ↪  all of them, but get an idea on what you might suggest from them
   ↪  or similar books you know of.
22
23     """
```

These prompts help the LLM produce accurate and favorable results, whether on the user's end with the predictions and recommendations or our end with the intention recognition or

the information extraction.

2.7 Database

2.7.1 Initial Database Design

The initial design phase of EduBot's database was centered around establishing a comprehensive and robust structure to store and manage essential data efficiently. The goal was to create a system capable of supporting EduBot's functionalities, such as personalized academic recommendations, score predictions, and study habit analysis.

Entity-Relationship Diagram (ERD)

The first step in the design process was constructing an Entity-Relationship Diagram (ERD) to outline the relationships between key entities. The initial ERD featured a large table named **Academics**, which consolidated all information about students, their courses, and academic performance. This table served as the backbone for EduBot's data storage during its early development.

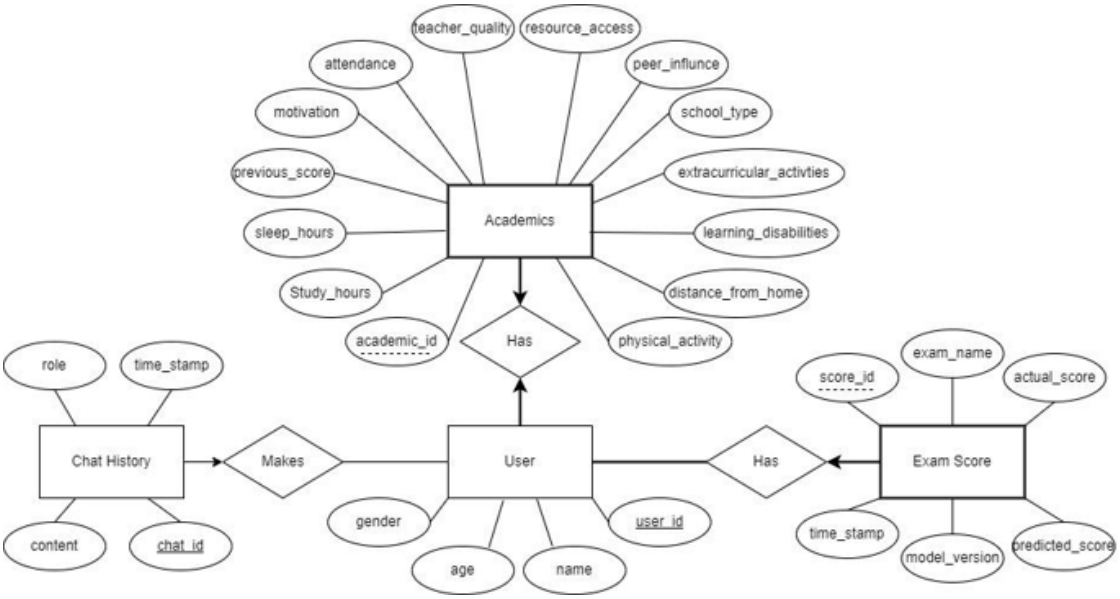


Figure 2.6: Our Initial ERD

2.7.2 How the Design Evolved (ERD, Normalization, Indices, and Views)

The decision to integrate Azure Flexible Database early in the design phase ensured that EduBot's database could evolve alongside its increasing user base and functionality.

Normalization and Updated ERD

The **Academics** table was normalized into smaller, more manageable tables to eliminate redundancy and improve query performance:

- **Student Profile:** Stores individual student details, such as personal information and study habits.

- **Course Details:** Contains information about the courses students are enrolled in and their associated academic performance.

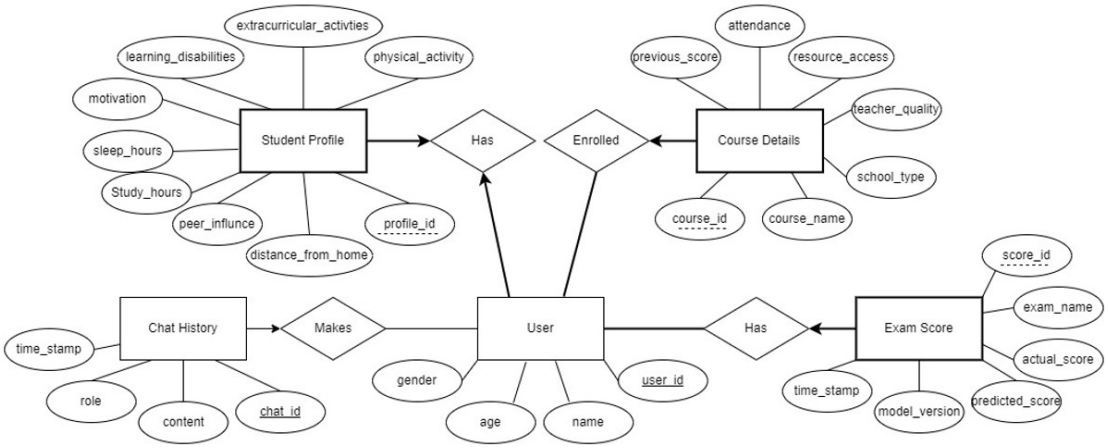


Figure 2.7: EduBot's Final ERD

Normalization minimized redundancy and created a more scalable database structure. The updated ERD reflected these changes, illustrating the refined relationships between entities and the improved data organization.

Indexing Strategy

To optimize query performance and ensure efficient data retrieval, indices were implemented on critical columns:

- **Index on user_id:** Speeds up joins between related tables and reduces latency for queries involving user-specific data.
- **Index on timestamp:** Optimizes retrieval and sorting of chat logs by time for real-time analysis.

The following SQL commands were used to implement the indices:

```
-- Indexes for performance optimization
CREATE INDEX idx_studentprofile_user_id ON StudentProfile(user_id);
CREATE INDEX idx_coursedetails_user_id ON CourseDetails(user_id);
CREATE INDEX idx_examscore_user_id ON ExamScore(user_id);
CREATE INDEX idx_chathistory_user_id ON ChatHistory(user_id);
CREATE INDEX idx_chathistory_timestamp ON ChatHistory(timestamp);
```

These indexing strategies significantly enhanced the database's responsiveness, particularly for time-sensitive operations.

Creating Views

To simplify data retrieval, views were created to provide aggregated and pre-filtered data for common queries. Views also proved instrumental for generating training datasets for machine learning models, enabling dynamic extraction of all relevant fields in a single step without manual table joins. For example:

- **Training Data View:** A view designed for machine learning model training, dynamically combining data from multiple tables such as *User*, *Student Profile*, *Course Details*, and *Exam Score*. This view includes critical fields like user demographics, study habits, course performance, and exam outcomes.

The following SQL command demonstrates the creation of the *Training Data View*:

```sql
CREATE VIEW TrainingDataView AS
SELECT
    u.user_id,
    u.name,
    u.age,
    u.gender,
    sp.study_hours_per_week,
    sp.sleep_hours_per_night,
    sp.previous_exam_scores,
    sp.motivation_level,
    sp.class_attendance,
    sp.extracurricular_activities,
    sp.physical_activity,
    sp.tutoring,
    sp.learning_disabilities,
    sp.distance_from_home,
    sp.peer_influence,
    cd.course_name,
    cd.teacher_quality,
    cd.resource_access,
    cd.school_type,
    e.exam_name,
    e.actual_score,
    e.predicted_score,
    e.model_version,
    e.timestamp
FROM User u
JOIN StudentProfile sp ON u.user_id = sp.user_id
LEFT JOIN CourseDetails cd ON u.user_id = cd.user_id
JOIN ExamScore e ON u.user_id = e.user_id;
```

The creation of views reduced the complexity of queries while ensuring data consistency, accuracy, and seamless integration with downstream processes like machine learning.

2.7.3 Platform Selection: Azure Flexible Database

Given EduBot's requirements for scalability, flexibility, and cloud-based management, we chose **Azure Flexible Database**. This platform offered key advantages, including:

- **Dynamic Resource Scaling:** Ensures the database can handle fluctuating workloads seamlessly.
- **Cloud-Based Management:** Reduces hardware overhead and simplifies maintenance.

Chapter 3

Results

3.1 Performance Evaluation

3.1.1 Machine Learning Model

Model	R^2	Mean Squared Error	Mean Absolute Error
Extra Trees Regressor	≈ 0.50	≈ 0.39	≈ 0.38
Gradient Boost Regressor	≈ 0.64	≈ 0.27	≈ 0.34
Extreme Gradient Boost Regressor	≈ 0.54	≈ 0.36	≈ 0.40
Support Vector Regressor	≈ 0.67	≈ 0.26	≈ 0.30
Stacking Model	≈ 0.69	≈ 0.25	≈ 0.30

Table 3.1: Machine Learning Models' performance over the training data

In [3.1] we illustrate the performance of the machine learning models over the training data. We can interpret R^2 score as how much of the data can be explained as how well does the models' variables explain or approximate the outputs.

GB and SVR were the best models to perform independently unlike ET and XGB while the stacking model outperformed all of them in all metrics.

We also tried several other models beginning from Linear Regression, Decision Trees and Random Forests up to Multi-Layer Perceptrons, but we decided to move forward with the ones illustrated above for now.

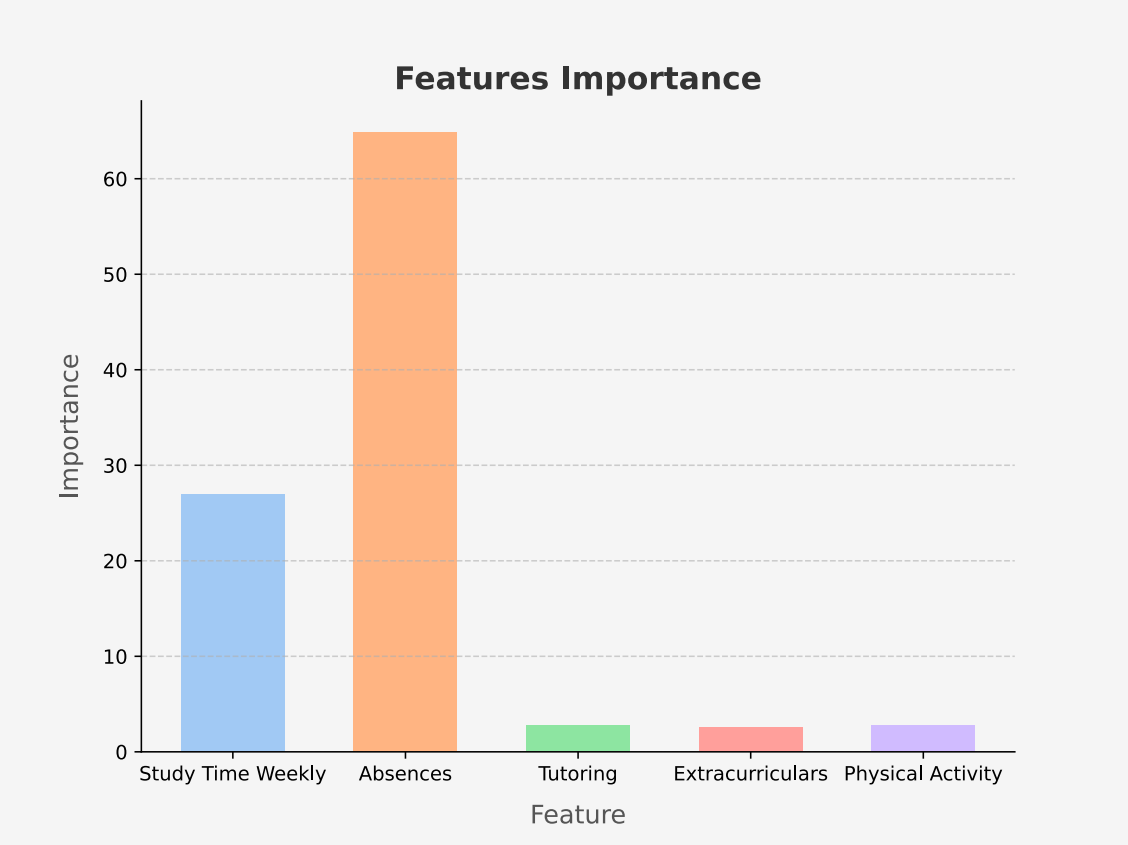


Figure 3.1: Most Important Features by Percentage

In figure [3.1], we can see the importance of each feature decided by the model. Particularly, this is from the ET Regressor, the other models did not differ much. We can see that the absence from lectures was the biggest factor for deciding the exam grade followed by the study time weekly.

We tried using L1 regularization (LASSO) to decrease the model dependency on the Absence feature solely but were unsuccessful.

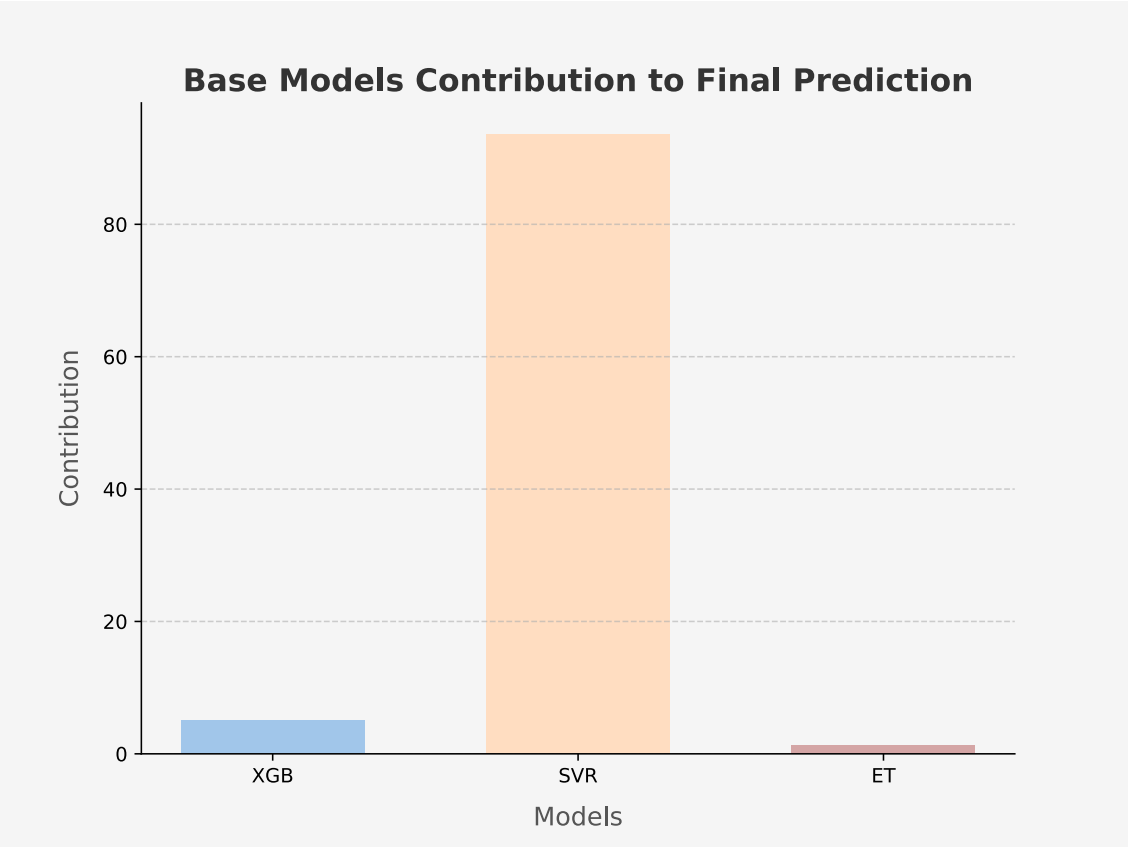


Figure 3.2: Base Models Contribution to the final prediction in the stacking model

In figure [3.2], we can see that SVR model contributed the most to the final prediction of the Stacking model, followed by XGB and finally ET. This also reflects how the SVR performed noticeably better than most other models independently too as illustrated in table [3.1].

We would like to note that these are just initial values and we do not believe the dataset we used is quite at the level needed to make a good generalized model with high accuracy. In other words, we are aware that the features used and the dataset is still lacking a lot and we believe that a Machine Learning model is as good as the data it is fed, so we look forward improving our training data and models more through interactions with EduBot or gaining access on better data sources as this was the best we could access at the time.

3.2 Database Implementation

The database implementation phase focused on translating the final ERD into an actual database schema while maintaining integrity constraints and optimizing for performance. Each table was implemented based on the normalized structure, adhering to relationships and dependencies as defined in the ERD.

3.2.1 SQL Code for Table Creation

The following SQL code demonstrates the creation of the primary tables:

```sql
-- User Table
CREATE TABLE User (
    user_id CHAR(36) PRIMARY KEY DEFAULT (UUID()),
    name VARCHAR(255),
    age FLOAT,
    gender INT
);

-- ExamScore Table
CREATE TABLE ExamScore (
    score_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id CHAR(36) NOT NULL,
    exam_name VARCHAR(255) NOT NULL,
    actual_score FLOAT,
    predicted_score FLOAT,
    model_version VARCHAR(50),
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE
);

-- ChatHistory Table
CREATE TABLE ChatHistory (
    chat_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id CHAR(36),
    message_content TEXT NOT NULL,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE SET NULL
);

-- StudentProfile Table
CREATE TABLE StudentProfile (
    profile_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id CHAR(36) NOT NULL UNIQUE,
    study_hours_per_week FLOAT,
    sleep_hours_per_night FLOAT,
    motivation_level INT,
    extracurricular_activities INT,
    peer_influence INT,
    learning_disabilities INT,
    distance_from_home INT,
    physical_activity FLOAT,
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE
);

-- CourseDetails Table
CREATE TABLE CourseDetails (
    course_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id CHAR(36) NOT NULL,
```

```
    course_name VARCHAR(255) NOT NULL,
    teacher_quality INT,
    previous_exam_scores FLOAT,
    class_attendance INT,
    resource_access INT,
    school_type VARCHAR(50),
    FOREIGN KEY (user_id) REFERENCES User(user_id)
);
```

3.2.2 Ensuring Integrity Constraints

All primary keys, foreign keys, and relationships defined in the ERD in figure [2.7] were implemented to maintain data consistency. For example:

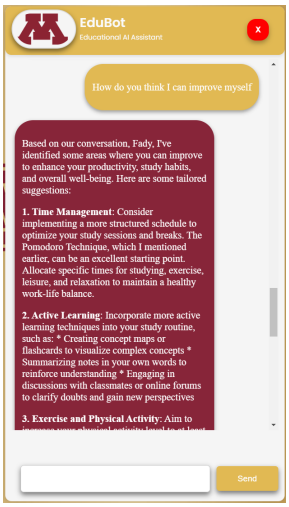
- The StudentProfile table includes a foreign key reference to User, enforcing the dependency. Also, the StudentProfile is a weak entity, where the `user_id` column is both `NOT NULL` (ensuring total participation) and `UNIQUE` (ensuring uniqueness), maintaining the relationship integrity.
- Deletion rules, such as `ON DELETE CASCADE` and `ON DELETE SET NULL`, were added to maintain referential integrity.

This schema ensures that the database remains consistent, scalable, and optimized for the various functionalities of EduBot.

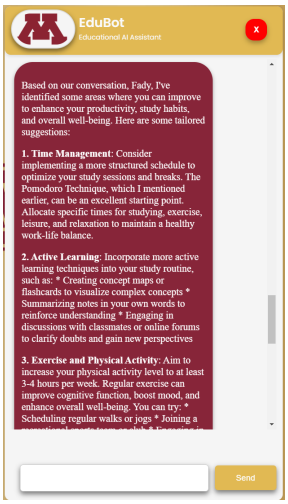
3.3 EduBot

In this section we illustrate the results we produced with EduBot, along with the GitHub Repository of the project to try EduBot yourself!

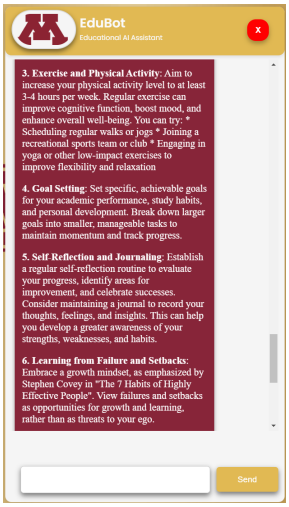
3.3.1 Recommendations for Students



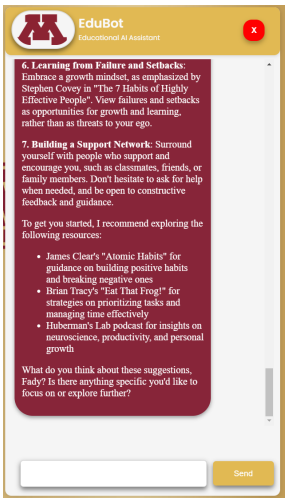
((a)) Recommendation



((b)) Recommendation 2



((c)) Recommendation 3



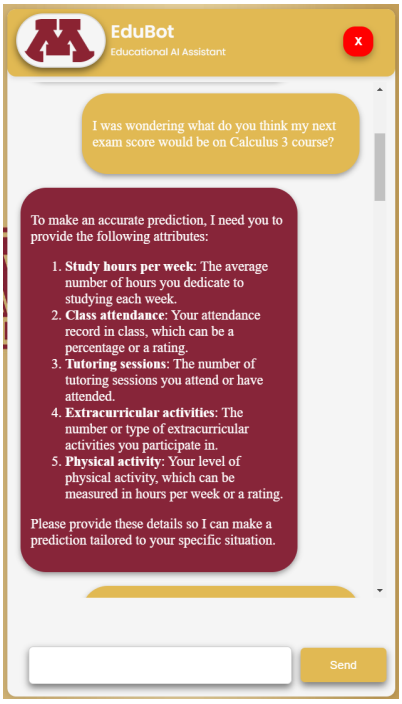
((d)) Recommendation 4

Figure 3.3: Student asking EduBot for a recommendation

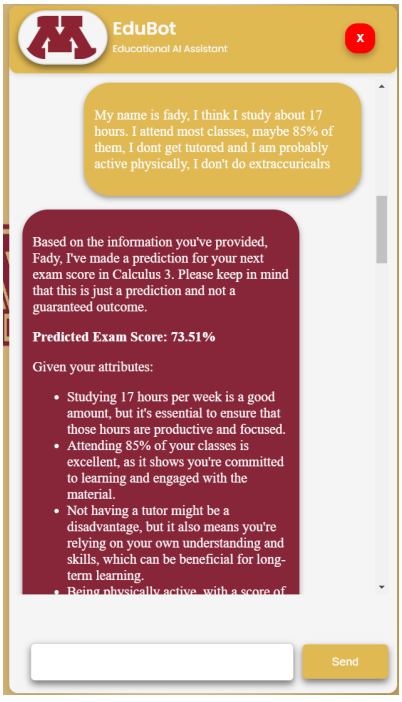
The user asked the Edubot for advice on how to improve themselves, and the bot replied with suggestions based on the chat history it had with the current user.

3.3.2 Predictions for Students

The user asked for prediction of their exam scores, and was asked to provide information about themselves. After that, the bot predicted the student's score based on the info provided by the user.



((a)) EduBot asking the student to provide more details



((b)) EduBot giving a score prediction along with some tips to improve

Figure 3.4: Caption

3.3.3 Information Extraction & Storage

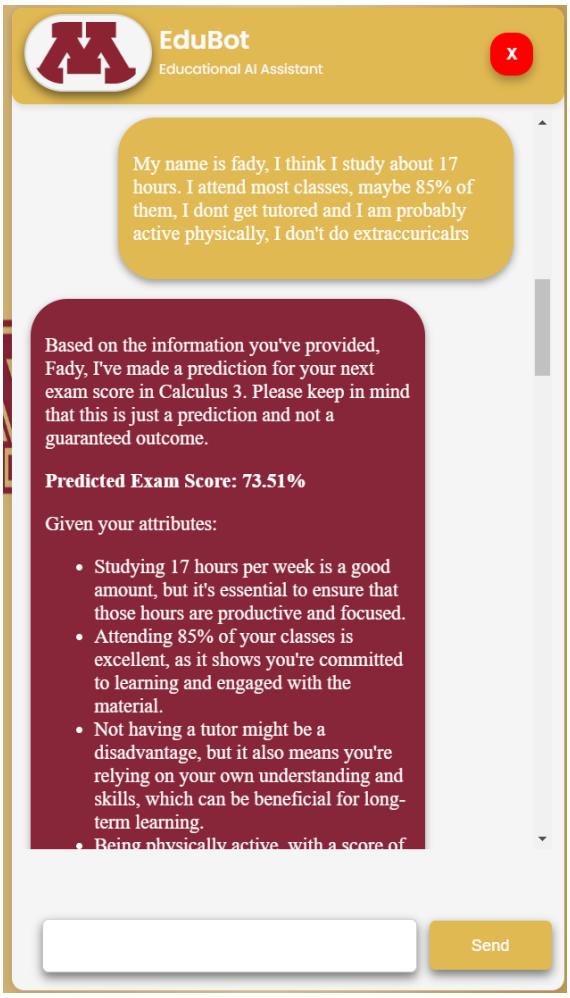


Figure 3.5: Prediction provided given the student's information

When the user asked for prediction and provided his info, the system (specifically LLM) extracted the data in a json format to be sent to the MLM,

user attributes JSON before info message:

```
{'user': {'name': None, 'age': None, 'gender': None},
'academics': {'study_hours_per_week': None, 'sleep_hours_per_night': None,
↪  'previous_exam_scores': None, 'motivation_level': None, 'class_attendance':
↪  None, 'teacher_quality': None, 'resource_access': None,
↪  'extracurricular_activities': None, 'school_type': None, 'peer_influence':
↪  None, 'learning_disabilities': None, 'distance_from_home': None,
↪  'physical_activity': None}}
```

after extraction from info message:

```
{'user': {'name': 'fady', 'age': None, 'gender': None},
```

```
2 'academics': {'study_hours_per_week': 17, 'sleep_hours_per_night': None,
↪  'previous_exam_scores': None, 'motivation_level': None, 'class_attendance':
↪  85, 'teacher_quality': None, 'resource_access': None,
↪  'extracurricular_activities': 0, 'school_type': None, 'peer_influence':
↪  None, 'learning_disabilities': None, 'distance_from_home': None,
↪  'physical_activity': 1}}
```

So the LLM can successfully extract information from text as needed, map the information values into the desired data types and format, and return it in JSON.

You can check out our project's code and try EduBot yourself here: https://github.com/muha-0/EduBot

Chapter 4

Conclusions and Future Work

4.1 Conclusions

EduBot, powered by Large Language Models (LLMs) and Machine Learning Models (MLMs), supports students academically. It offers personalized study recommendations, exam score predictions, and data-driven insights. The system's three-tier architecture ensures efficient interaction.

The LLM module understands user intents, extracts information, and provides relevant advice. The MLM module predicts exam scores based on study hours, attendance, and motivation. The Azure Flexible Database stores user data securely and efficiently.

Limitations exist, such as the need for high-quality datasets and hardware constraints for running a local LLM. However, APIs for LLMs provide scalability and efficiency for this phase.

Future enhancements include a locally fine-tuned LLM, improved data quality, and access to student performance insights for academic advisors. These improvements aim to create a more robust and effective academic assistant.

4.2 Limitations

Dataset for modeling:

- We were not able to find datasets with better quality or variables that explain the exam scores more logically than what we used until this points, some quality/realistic datasets that we found were private whether from research papers or other sources and we were not able to gain permission to access them unfortunately.

LLM implementation:

- We were initially planning to implement a local LLM to be able to personalize and fine-tune more on our domain and tasks, but that use hard and impractical to do yet for two reasons:

    1. We were still inexperienced with using LLMs so it was hard for us to figure out how to implement it locally while maintaining scalability and being able to use it on any device due to their large sizes and power requirements which was also impractical to figure out on our own devices too.
    2. Using an API to call the LLM would allow us to scale and deploy EduBot more efficiently, use larger and more powerful models like Llama 3.2 90B and 3.3 70B

which would improve EduBot performance greatly, and we can still customize the LLM on our specific tasks through in-context learning and prompting techniques like Few and Zero Shot Learning.

We still plan to use our own customized LLM in the future but we believe using one through an API is good enough at this phase.

4.3 Future work

In the future, we aim to implement several improvements and features to enhance our system's effectiveness and user experience:

1. **Local Large Language Model (LLM) Integration:** We plan to incorporate a local LLM that will be optimized and fine-tuned to handle data specific to academic assistance. By focusing on study habits and productivity enhancement, this model will aid students more effectively by providing tailored advice and support.
2. **Enhanced Data Quality for Model Training:** To improve the accuracy and reliability of our machine learning models, we intend to use more representative and high-quality data reflecting students' grades. This will ensure that our models are trained with relevant and precise data, resulting in better performance and outcomes.
3. **Advisor Access to Academic Insights:** We will enable academic advisors to have access to their students' academic performance and study habits. This feature will allow advisors to provide more personalized and timely guidance, helping students improve their learning strategies and academic results.
4. **Improved Chatbot Responsiveness:** By extracting meaningful insights from user conversations and information, we aim to significantly elevate the quality of our chatbot responses. This improvement will make interactions more helpful and relevant, ultimately enhancing the user experience.

4.4 Acknowledgment

We would love to extend our thanks and huge appreciation to Professor Jaideep and our TA Jiaxiang Tang for their huge help to us throughout this course, for the great content and lessons that enriched us and made us get more deep into topics important topics in Database systems and Data generally that a computer scientist must know and learn. This course was a great experience and helped us level up our skills and knowledge.

References

Meta AI. Llama 3.3 70b: Model card and prompt formats, 2024. URL https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_3. Accessed: 2024-12-12.

Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020.

L. Guyn. Student performance factors, 2021. URL https://www.kaggle.com/datasets/lainguyn123/student-performance-factors.

Enkelejda Kasneci, Kathrin Sessler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, Stephan Krusche, Gitta Kutyniok, Tilman Michaeli, Claudia Nerdel, Jürgen Pfeffer, Oleksandra Poquet, Michael Sailer, Albrecht Schmidt, Tina Seidel, Matthias Stadler, Jochen Weller, Jochen Kuhn, and Gjergji Kasneci. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103: 102274, 2023. ISSN 1041-6080. doi: https://doi.org/10.1016/j.lindif.2023.102274. URL https://www.sciencedirect.com/science/article/pii/S1041608023000195.

Rabie El Kharoua. Students performance dataset, 2021. URL https://www.kaggle.com/datasets/rabieelkharoua/students-performance-dataset/data.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.