AI Personal Tutor

1. Introduction

1.1 Problem Statement

Traditional classroom environments or static e-learning platforms often use a "one-size-fits-all" approach, failing to cater to the individual needs of students. These systems do not consider differences in learning paces, preferred learning styles, or gaps in knowledge, which can lead to a lack of engagement and missed learning opportunities. Without adaptive learning paths or personalized feedback, students may struggle to stay motivated, and their progress may stagnate.

This problem can be mitigated by implementing a system that can:

- Tailor learning materials based on a student's past performance and current needs.
- Dynamically adapt learning content to challenge students without overwhelming them.
- Offer real-time feedback on their progress, enabling continuous improvement.

1.2 Objective

The primary objective of the AI Personal Tutor project is to develop a machine learning-driven tutoring system that can provide adaptive learning experiences. Key goals include:

- Predicting whether a student will successfully answer a question based on past behavior and performance.
- Recommending personalized learning materials or questions to address weaknesses and reinforce strengths.
- Continuously tracking a student's progress and adjusting recommendations to optimize learning outcomes.

2. Literature Review

The integration of Artificial Intelligence in education has revolutionized the learning process. Techniques such as Knowledge Tracing, Collaborative Filtering, and Performance Prediction are central to adaptive learning systems.

Knowledge Tracing is a method for predicting future student performance based on past behavior. It has been implemented using Deep Learning models like LSTM, allowing for the modeling of sequential data.

Collaborative Filtering is commonly used for recommending content based on user behavior. Matrix factorization techniques such as Singular Value Decomposition (SVD) can uncover hidden patterns in student interactions, making it effective for personalizing recommendations.

Performance Prediction models, such as XGBoost, analyze data like response times and correctness to predict a student's future success. These models help in identifying at-risk students early, allowing educators to intervene.

By combining these techniques, this project aims to create a holistic system capable of making personalized learning recommendations and predicting academic performance.

3. Technologies Used

Backend:

- **FastAPI**: A high-performance web framework used to build the backend APIs. FastAPI is ideal for building machine learning-powered web services, allowing asynchronous calls and fast model inference.
- **TensorFlow**: An open-source framework that powers the LSTM model for knowledge tracing. TensorFlow's flexibility allows for effective training and deployment of deep learning models.
- **XGBoost**: A popular model for structured data, used to predict student performance by analyzing past behaviors and features like response times and tutor modes.
- **Pickle**: Used for serializing the trained models and saving them for later inference without retraining, ensuring smooth operation in production environments.

Frontend:

- **Streamlit**: This Python-based framework allows for quick and easy creation of interactive dashboards. It is used for building the real-time interface for users (students, educators) to interact with the AI tutor.

Other Libraries:

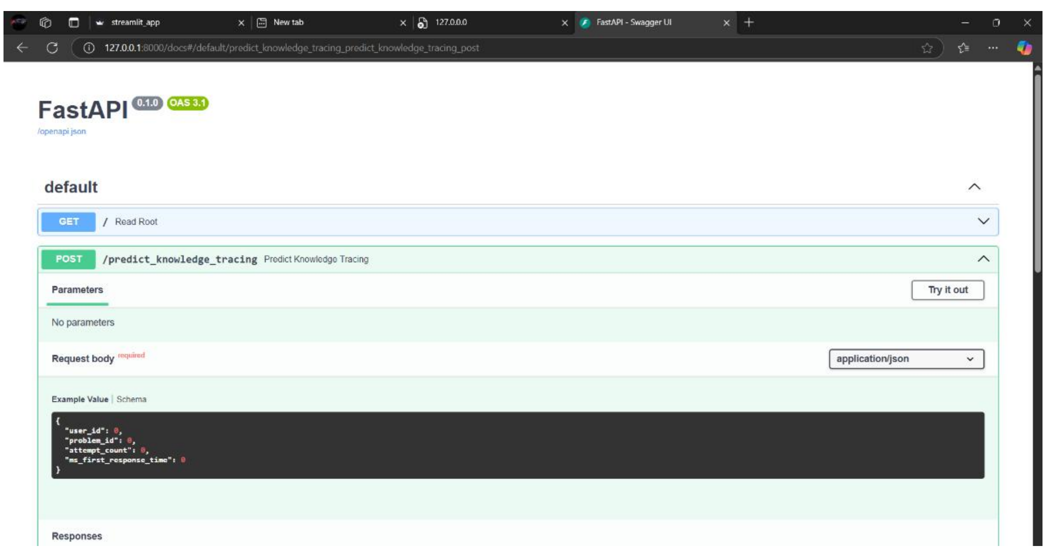
- **NumPy, Pandas**: Essential libraries for data manipulation, including cleaning, normalization, and processing.
- **Matplotlib, Seaborn**: Used for creating visualizations that help analyze data distributions, model performance, and predictions.
- **Scikit-learn**: For preprocessing tasks, splitting datasets into training and testing sets, and evaluating model performance.

4. System Architecture

The architecture of the AI Personal Tutor system consists of multiple components:

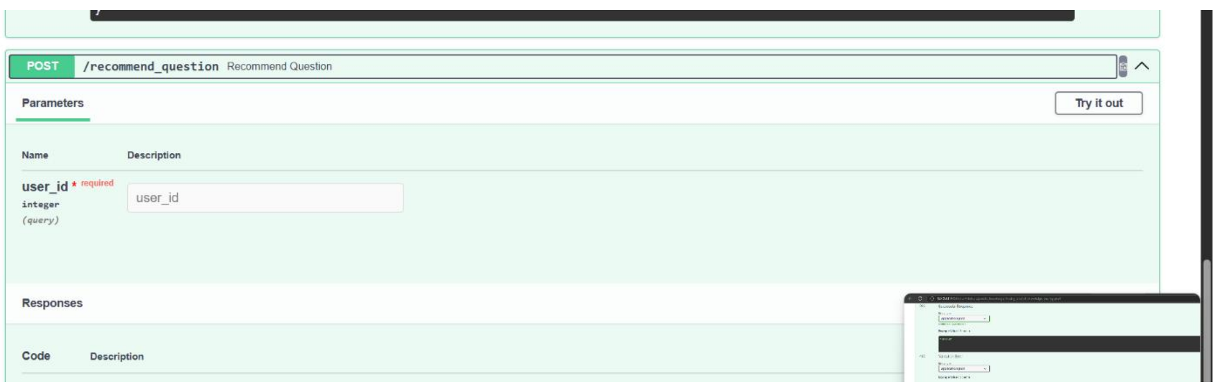
- ***Frontend (Streamlit*)**: This interface collects user inputs such as user ID, question ID, previous answers, and time taken to answer questions. Streamlit provides a smooth user experience for students and educators to interact with the AI tutor.
- ***Backend (FastAPI):*** FastAPI serves as the intermediary between the frontend and the machine learning models. It routes the input data to the appropriate model (LSTM, SVD, XGBoost) and returns real-time predictions and recommendations.
- ***Machine Learning Models***:
    - **LSTM**: This model is used for knowledge tracing and predicting whether a student will answer a question correctly.
    - **SVD**: Collaborative filtering-based model that recommends questions to students based on their previous performance.
    - **XGBoost**: Predicts future performance based on student data.

Screenshot of FastAPI Documentation:

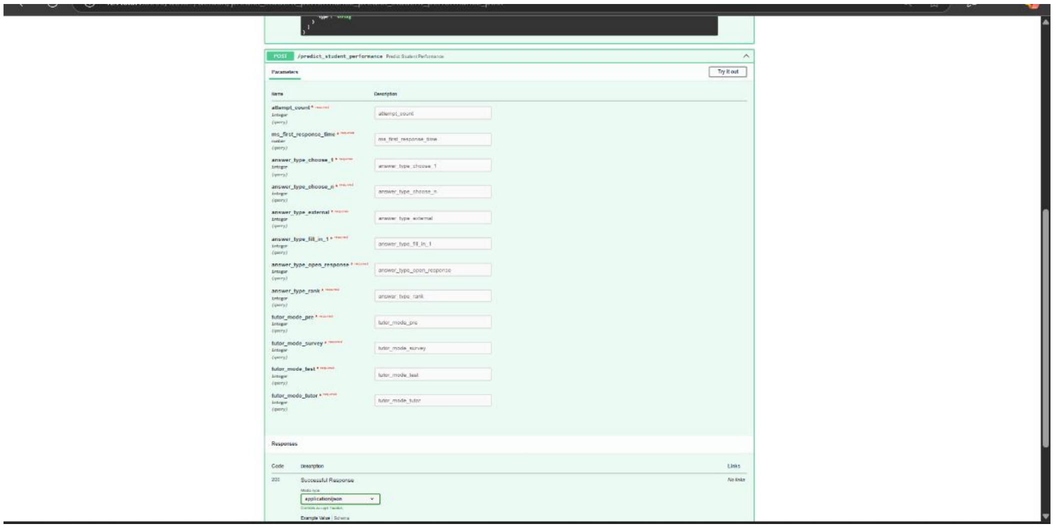


Screenshot of FastAPI Response:

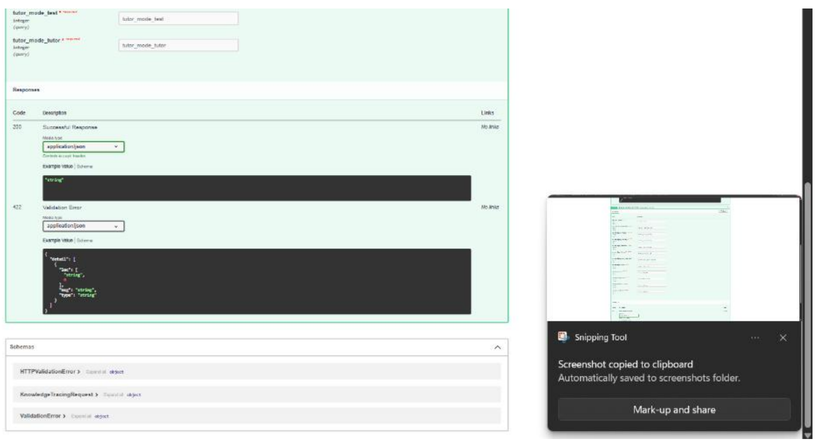
Screenshot of FastAPI Parameters:



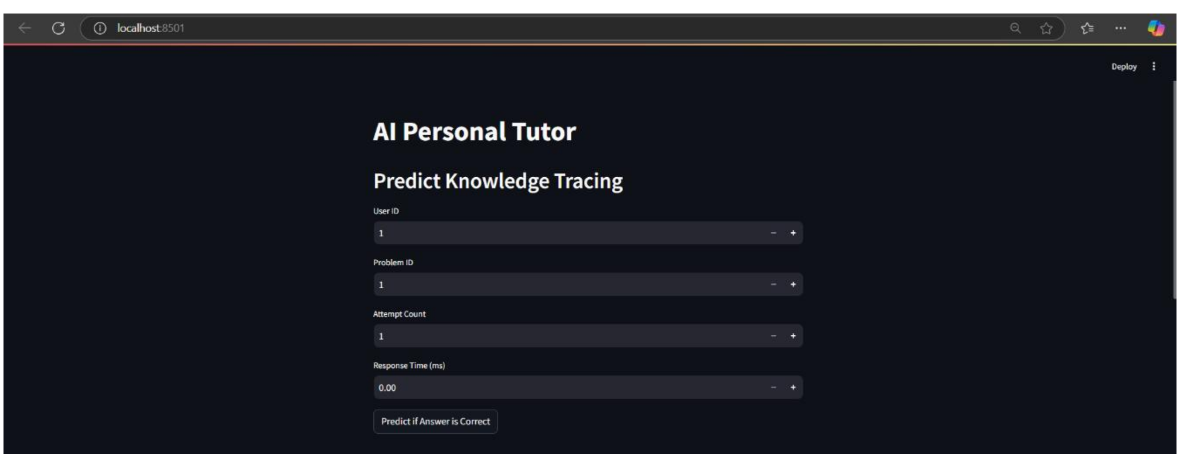
Screenshot of FastAPI Request Body:



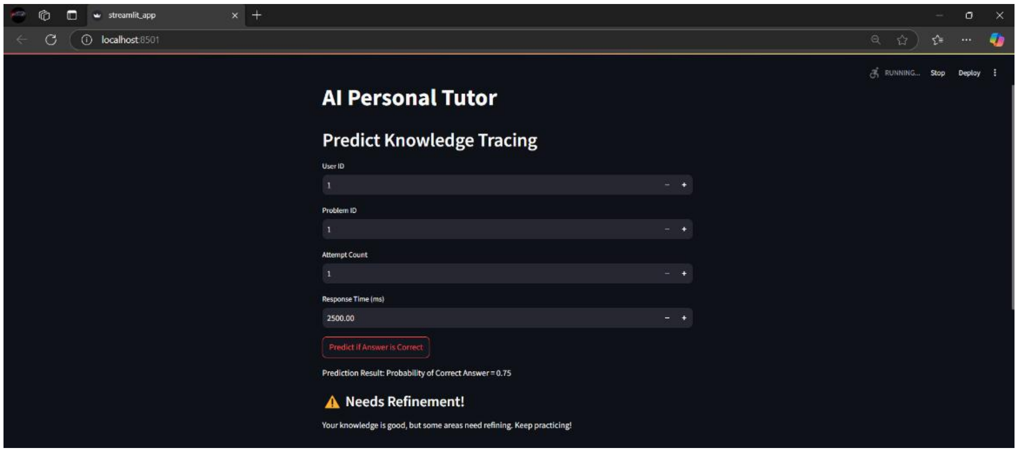
Screenshot of FastAPI Error Response:



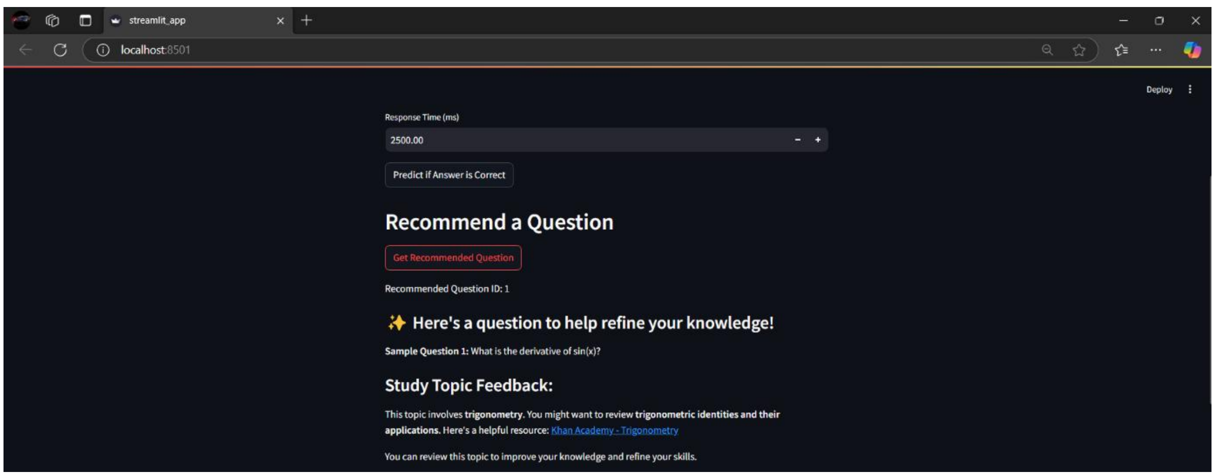
Screenshot of Streamlit Interface:



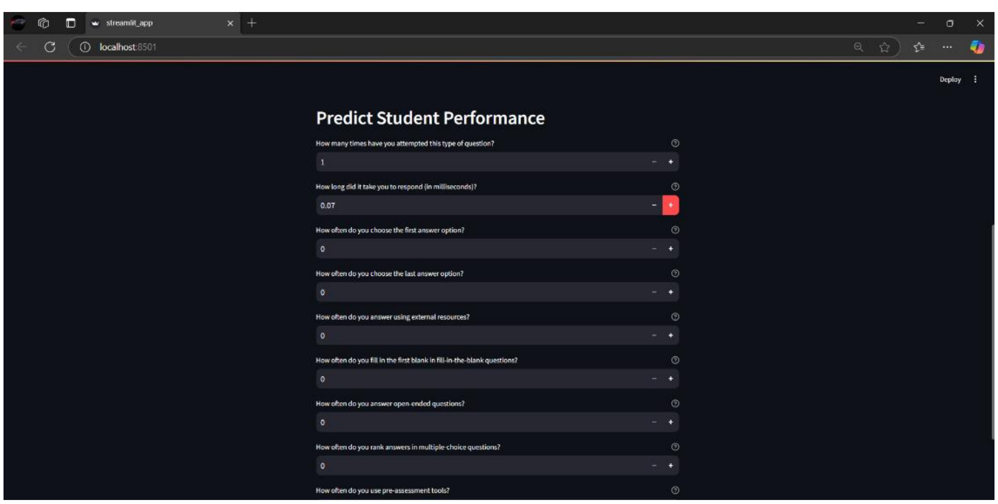
Screenshot of Streamlit Interface 2:



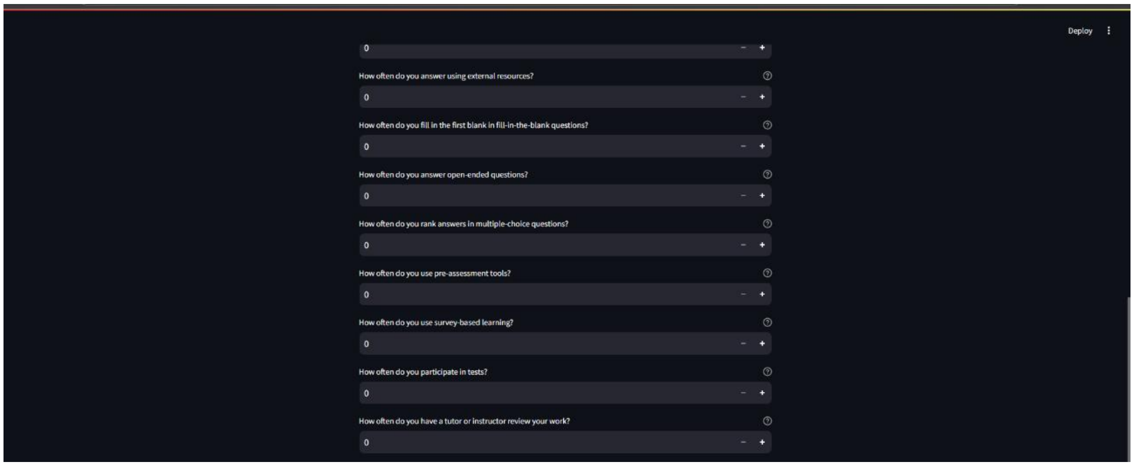
Screenshot of Streamlit Interface 3:



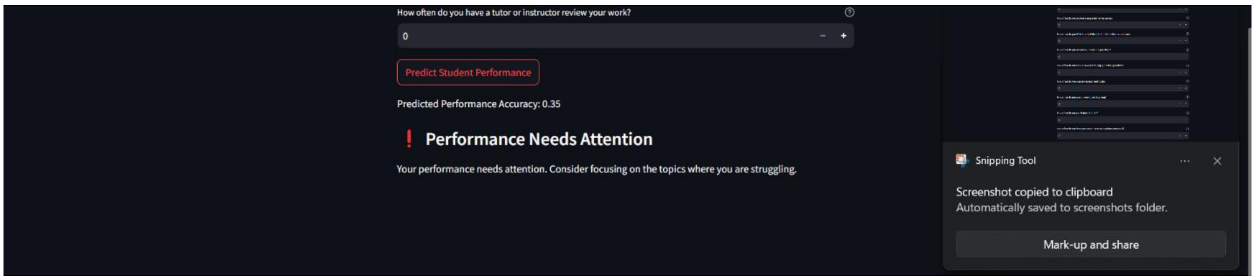
Screenshot of Streamlit Interface 4:



Screenshot of Streamlit Interface 5:



Screenshot of Streamlit Interface 6:



5. Dataset

1. **Name**: Assistments 2009–2010 Dataset.
2. **Size**: Approximately 1 million rows.
3. **Key Features**:
   - **user_id**: Identifies each student.
   - **problem_id**: The ID of the question or problem.
   - **correct**: The student's response (1 for correct, 0 for incorrect).
   - **attempt_count**: Number of attempts made by the student.
   - **ms_first_response_time**: The time taken to respond to the question.

- **tutor_mode**: The mode used by the tutor (e.g., hints or feedback).
- **answer_type**: Type of answer provided (e.g., multiple choice, written).

The target variable for the models is the **correct** field (1 or 0).

Data Preprocessing:

- **Handling Missing Values**: Missing data is imputed using the median or mode of the respective columns.
- **Normalization**: Numerical features such as response time are normalized to ensure consistency across the dataset.
- **One-Hot Encoding**: Categorical variables such as answer type are converted into numerical representations.

2. Name: ETD4 Dataset (EdNet KT4)

The dataset used in this step is the **ETD4 Dataset**, processed from the **EdNet KT4 ZIP file**. It is a large dataset containing student interaction logs and engagement data. The dataset includes several features that are relevant for predicting student behaviors and performance:

- **Columns**:
    - **timestamp**: Timestamp of the student's action.
    - **action_type**: Type of action the student took (e.g., 'enter', 'submit', 'respond').
    - **item_id**: Identifier for the item the student interacted with.
    - **cursor_time**: Duration of the action.
    - **source**: The origin of the action (e.g., 'diagnosis', 'review').
    - **user_answer**: The answer provided by the student.
    - **platform**: The platform on which the student interacted (e.g., 'web').

Preprocessing Steps:

1. **Handling Missing Values**:
    - Forward filling (fillna(method='ffill')) is used to handle missing values, especially in cursor_time, source, and platform columns.
2. **Feature Engineering**:

- **Time-related Features**: Extracted hour, day, and month from the timestamp to better understand student activity trends.
- **One-Hot Encoding**: Categorical variables like action_type, source, and platform are one-hot encoded to create numerical representations.

3. **Scaling**:
   - A MinMaxScaler is used to normalize continuous features like cursor_time for better model performance.
4. **Train-Test Split**:
   - The dataset is split into training and testing sets, with 80% used for training and 20% reserved for testing.

6. Model 1: LSTM for Knowledge Tracing

Purpose:

LSTM (Long Short-Term Memory) is used to predict whether a student will answer a question correctly, based on their historical performance.

Architecture:

- **Input**: Sequence of previous answers, response time, and question difficulty.
- **Layer Structure**: LSTM layers followed by dense layers.
- **Output**: A binary prediction (1 for correct, 0 for incorrect).

Training:

- **Epochs**: 10
- **Batch Size**: 64
- **Accuracy**: ~80.6%
- **Loss**: 0.42

Key Takeaways:

LSTM performs well in modeling sequential learning data, as it is designed to capture long-term dependencies in time-series data.

7. Model 2: SVD for Personalized Question Recommendations

Purpose:

SVD (Singular Value Decomposition) is used for collaborative filtering to recommend personalized questions based on a student's past responses.

Technique:

- **Collaborative Filtering**: SVD decomposes the user-item interaction matrix to generate recommendations.
- **Evaluation**: RMSE (Root Mean Square Error) of 0.409 on the test set indicates that the model provides good recommendations based on past performance.

8. Model 3: XGBoost for Student Performance Prediction

Purpose:

XGBoost is used to predict a student's future performance based on their previous behaviors.

Features:

- **Response Time**
- **Tutor Mode**
- **Answer Type**
- **Attempt Count**

Results:

- **RMSE**: 0.358, indicating strong predictive accuracy.

9. LSTM for Knowledge Tracing (Expanded with ETD4 Data)

Purpose:

The **LSTM model** is designed for **knowledge tracing**, predicting whether a student will answer a question correctly based on their historical interactions. The **ETD4 dataset**'s features, such as action_type, cursor_time, and user_answer, are used to feed the LSTM model.

Architecture:

- **Input**: Student interactions (timestamp, action type, platform, etc.), with features like cursor_time normalized.
- **Output**: Binary prediction (0 for incorrect, 1 for correct).

The data from the ETD4 dataset is reshaped to fit the LSTM model (samples, time steps, features).

10. Q-Learning for Difficulty Adjustment

This layer, utilizing **Reinforcement Learning (RL)**, dynamically adjusts the difficulty of the questions presented to the student. It uses rewards and penalties based on student performance, which can be tracked from the **ETD4 dataset's** interaction logs (e.g., time spent, actions taken).

- **States**: Student-question pairs.
- **Rewards**: +1 for a correct answer, -1 for an incorrect answer.
- **Actions**: Select the next question to present.

This helps personalize the learning experience by dynamically adjusting the challenge based on a student's evolving abilities.

11. Results and Evaluation

- **LSTM**: Achieves high accuracy for sequential prediction tasks.
- **SVD**: Improves question recommendations, making the learning process more personalized.
- **XGBoost**: Accurately predicts student success based on past interactions.

Strengths:

- Modular and easy to scale.
- Personalized learning paths.
- Offers real-time feedback and adapts to student progress.

Model Performance:

- **Accuracy**: The accuracy of the LSTM model on the ETD4 dataset is reported at ~80.6%, indicating strong sequential learning prediction capabilities.

- **Loss**: The loss for the model is 0.42, showing room for further optimization.

12. Conclusion

The AI Personal Tutor platform combines advanced machine learning models to offer an adaptive, personalized learning experience. By integrating models for knowledge tracing, performance prediction, and question recommendations, it addresses key challenges in traditional education, providing real-time, data-driven decision-making.

13. Future Work

- **NLP Support**: Integrating Natural Language Processing for understanding free-text answers could enhance the tutor's feedback capabilities.
- **Audio/Video Feedback**: Incorporating audio or video explanations for feedback could improve engagement.
- **Skill-based Curriculum Mapping**: Tailoring content to specific skill sets or learning objectives would further personalize the experience.