**Project Summary for EDNet K3 - Content Recommendation System**

**Dataset Details:**

* **Dataset**: [EDNet K3 Dataset](https://github.com/riiid/ednet)
* **Description**:  
  The EDNet K3 dataset contains user interactions with educational content. It captures students' engagement behaviors, including answering questions, viewing lectures, and more.
* **Structure**:
  + Features include student ID, content ID, timestamp, correct/incorrect answers, and additional metadata.
  + Labels are based on student performance and engagement for recommending appropriate content (basic or advanced).
* **Usage**:
  + The dataset is preprocessed for supervised learning.
  + Features are extracted for model training to classify whether a student should receive **basic** or **advanced** learning materials.

**Google Colab NOTEBOOK’s Documentation:**

**Models Used:**

1. **K-Nearest Neighbors (KNN) Classifier**:
   * **Architecture**:
     + Instance-based model using the 5 closest neighbors (n\_neighbors = 5) to predict the class label.
   * **Working**:
     + No training phase; uses distance calculation at prediction time.
2. **Deep Learning Model (Keras Sequential Model)**:
   * **Architecture**:
     + Input Layer: Number of features from the dataset.
     + Hidden Layer 1: 64 neurons + ReLU activation + 30% Dropout.
     + Hidden Layer 2: 32 neurons + ReLU activation + 30% Dropout.
     + Output Layer: 1 neuron + Sigmoid activation (binary output).
   * **Optimizer**: Adam
   * **Loss Function**: Binary Crossentropy
   * **Callbacks**: EarlyStopping (patience = 2 epochs)
   * **Model Saving**: The trained deep learning model is saved in Keras format (recommendation\_model.keras).

**Expected Results:**

* **Goal**: Predict whether a student should be recommended **basic** or **advanced** content.
* **Evaluation Metrics**:
  + **Accuracy** (for both KNN and Deep Learning models)
  + **Classification Report** (Precision, Recall, F1-Score)
* **Results Achieved**:
  + Both models showed good performance, and accuracy is printed after evaluation on the test set.

**Google Colab Notebook Link:**

* You can access the full training and evaluation process here:  
  **[https://colab.research.google.com/drive/1Yhy8OIRFWTd2xNWGQ80e19YMOiVAaxfQ?usp=sharing]**

**Files Delivered in the Zip file was:**

* model\_training.ipynb (Google Colab notebook)
* .py version for standalone execution
* Trained Model Weights (recommendation\_model.keras)
* Dataset Preprocessing Code
* PDF Report (Quick Overview)
* Dataset Files

**Real-Time Personalized Learning Recommendation Streamlit App**

**Purpose:**

This app provides **real-time content recommendations** to students based on their interaction behavior using a **trained deep learning model** (recommendation\_model.keras).  
It helps decide whether the student should be recommended **basic practice exercises** or **advanced learning content**.

**Technology Stack:**

* **Frontend/UI**: Streamlit (Python web framework)
* **Backend Model**: Pretrained TensorFlow/Keras deep learning model
* **Visualization**: Matplotlib, Seaborn
* **Other Libraries**: NumPy, Pandas, streamlit, scikit-learn, streamlit

**How the App Works:**

1. **User Interaction Inputs (Sidebar)**:
   * **Action Type**: Select between 'enter', 'respond', or 'submit'.
   * **User Answer**: Select 'Correct', 'Incorrect', or 'No Answer'.
   * **Platform Used**: Choose 'mobile', 'web', or 'tablet'.
   * **Hour of the Day**: Pick an hour from 0 to 23.
   * **Day of the Week**: Select any day from Monday to Sunday.
2. **Input Encoding**:
   * Inputs are numerically encoded before feeding into the deep learning model.
3. **Prediction**:
   * The trained Keras model predicts whether the student needs:
     + **Basic content** (0)
     + **Advanced content** (1)
4. **Recommendations**:
   * If **Basic** -> Recommends a short, simple video.
   * If **Advanced** -> Recommends a deep learning or advanced topic video.
   * Recommendations are randomly selected from a list of pre-defined educational videos.
5. **Session Summary**:
   * All user inputs and recommendations are saved in session memory and displayed in a table.
6. **Visual Insights**:
   * **Action Type Distribution**
   * **Platform Usage**
   * **Recommendation Distribution**  
     (Auto-generated graphs are shown for better understanding of user behavior.)
7. **Model Used**:
   * Deep Learning Sequential Model
     + Input: User interaction features
     + Hidden Layers: Dense layers with ReLU activations and Dropout regularization
     + Output: Sigmoid output for binary classification (Basic or Advanced)

**Features:**

* **Simple and clean** UI with real-time predictions
* **Personalized content recommendations** based on student behavior
* **Session history tracking** (see all inputs and predictions during the app session)
* **Insightful visualizations** with graphs and charts
* **Fully responsive design** — works on any browser without setup
* **Saved Model Loading** — no need to retrain, instantly usable

**To run the app locally:**Install Streamlit: In terminal run pip install -r requirements.txt  
Run the app: streamlit run app.py