**Project Structure**

bash

Copy code

/student-performance-prediction

│

├── app.py # Streamlit app for predictions and visualization

├── training.py # Script to train the model

├── student\_performance\_model.pkl # Trained model

├── feature\_names.pkl # List of features used for training

├── label\_encoder.pkl # LabelEncoder for decoding predictions

├── requirements.txt # List of dependencies

├── README.txt # Project documentation

**Setup and Installation**

**1. Clone the repository (if applicable):**

If you're using GitHub, you can clone the repository by running:

bash

Copy code

git clone https://github.com/your-username/student-performance-prediction.git

cd student-performance-prediction

**2. Create a Virtual Environment (optional but recommended):**

If you are using a virtual environment (which is recommended), create and activate it:

**For venv:**

bash

Copy code

python3 -m venv venv

source venv/bin/activate # On Linux/MacOS

venv\Scripts\activate # On Windows

**For conda (if you use Anaconda):**

bash

Copy code

conda create --name student\_performance\_env python=3.8

conda activate student\_performance\_env

**3. Install Dependencies:**

Run the following command to install the necessary libraries:

bash

Copy code

pip install -r requirements.txt

Make sure you have the required libraries listed in the requirements.txt file. The requirements.txt file should contain:

Copy code

pandas

numpy

scikit-learn

matplotlib

joblib

streamlit

**Training the Model**

**1. Run the Training Script:**

To train the model, use the provided training.py script. This will load the data, process it, and train a Random Forest Classifier.

bash

Copy code

python training.py

**2. Outputs:**

After running the training script, the following files will be saved:

* **student\_performance\_model.pkl**: The trained Random Forest model.
* **feature\_names.pkl**: The list of feature names used for training.
* **label\_encoder.pkl**: The LabelEncoder used to convert class labels (e.g., "Pass," "Fail") into numeric values.

**Running the Streamlit Application**

**1. Start the Streamlit App:**

Once the model is trained, you can run the Streamlit app to upload student data and get predictions.

bash

Copy code

streamlit run app.py

This command will start a local web server and open the app in your default web browser at http://localhost:8501.

**2. App Interface:**

* **Upload a CSV File**: Upload a CSV file containing student data. The dataset should include the following columns:
  + id\_student: Unique student identifier
  + score: The score the student obtained in their assessments
  + sum\_click: Total number of clicks on learning materials
  + total\_clicks: Aggregated click data for each student (calculated during preprocessing)
* **Predictions**: After uploading the file, the app will display predictions for each student, indicating whether they will pass, fail, or earn distinction.
* **Download Predictions**: You can download the predictions in a CSV file that includes both id\_student and the prediction.

**File Format for Input Data**

**CSV Format Example:**

Ensure your dataset follows the format below:

python

Copy code

id\_student,score,sum\_click,total\_clicks

12345,85,10,120

67890,65,8,100

...

**How the Model Works**

1. **Data Preprocessing**:
   * The dataset is loaded and preprocessed to handle missing values and encode categorical features.
   * Relevant features, such as score, sum\_click, and total\_clicks, are selected.
   * LabelEncoder is used to encode the target variable final\_result (Pass, Fail, etc.) into numeric values.
2. **Model Training**:
   * A Random Forest Classifier is trained on the features to predict student performance.
3. **Prediction**:
   * The trained model is used to predict the final\_result based on input features.
   * The predictions are decoded back into their original labels (e.g., "Pass", "Fail") using the saved LabelEncoder.

**Notes**

* **Model Training Time**: Depending on your system's performance and dataset size, model training may take several minutes.
* **Feature Engineering**: The feature total\_clicks is derived from the sum\_click column, which aggregates the number of clicks made by the student.
* **Output**: The output predictions are categorized into labels such as "Pass", "Fail", or "Distinction", but these labels are numerically encoded, so they need to be decoded back for human-readable results.

**Troubleshooting**

* **Missing Columns in Uploaded Data**: If your uploaded dataset does not contain the required columns (id\_student, score, sum\_click, etc.), the app will display an error message. Ensure that your data includes all necessary features.
* **Model Not Found**: If the app cannot find the trained model (student\_performance\_model.pkl), make sure you have run the training script and that the model file is in the same directory.