# Personalized Student Recommendations

## Main code

import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns from sklearn.linear_model import LinearRegression from sklearn.model_selection import train_test_split from sklearn.metrics import mean_absolute_error

**---------- 1. Load the Data ----------**

**File Paths (Replace with actual file locations)**

current_quiz_file = "C:\Users\Ramya\Downloads\current_quiz.csv" historical_quiz_file = "C:\Users\Ramya\Downloads\historical_quiz.csv"

try: df_current = pd.read_csv(current_quiz_file) df_historical = pd.read_csv(historical_quiz_file) print("Data Loaded Successfully!") except Exception as e: print(f"Error loading data: {e}") exit()

**---------- 2. Analyze Student Performance ----------**

def analyze_performance(student_id): """Analyze accuracy trends, weak topics, and difficulty levels for a student.""" student_data = df_historical[df_historical['student_id'] == student_id]

if student_data.empty:
    print(f"No data found for Student ID: {student_id}")
    return None, None

# Calculate accuracy by topic
topic_accuracy = student_data.groupby('topic')['correct'].mean().reset_index()
topic_accuracy.columns = ['Topic', 'Accuracy']

# Calculate accuracy by difficulty level
difficulty_analysis = student_data.groupby('difficulty')['correct'].mean().reset_index()
difficulty_analysis.columns = ['Difficulty Level', 'Accuracy']

```
return topic_accuracy, difficulty_analysis
```

## ---------- 3. Generate Personalized Recommendations ----------

```
def generate_recommendations(student_id): """Suggest topics and difficulty levels to
focus on.""" student_data = df_historical[df_historical['student_id'] == student_id]

if student_data.empty:
    return "No data available for recommendations."

# Identify weak topics (lowest accuracy)
weak_topics = student_data[student_data['correct'] ==
0]['topic'].value_counts().head(3).index.tolist()

# Identify easy questions student should start with
easy_questions = student_data[student_data['difficulty'] ==
'Easy']['topic'].value_counts().head(3).index.tolist()

recommendations = {
    "Focus More On": weak_topics if weak_topics else "No weak topics detected",
    "Start With Easy Topics": easy_questions if easy_questions else "No easy topics
detected",
    "Practice More MCQs": "Increase practice in weak areas"
}

return recommendations
```

## ---------- 4. Define Student Persona ----------

```
def define_student_persona(student_id): """Categorize students based on performance
patterns.""" student_data = df_historical[df_historical['student_id'] == student_id]

if student_data.empty:
    return "No data available for persona classification."

avg_score = student_data['score'].mean()
```

```
if avg_score > 75:
    persona = "Top Performer - Maintain consistency!"
elif avg_score > 50:
    persona = "Average Performer - Improve weak areas!"
else:
    persona = "Needs Improvement - Focus on basics!"

return persona
```

## ---------- 5. Predict NEET Rank ----------

def predict_neet_rank(): """Predict student's NEET rank based on quiz performance using Linear Regression.""" if 'score' not in df_historical.columns: print("Error: 'score' column missing in dataset.") return None

```
df_historical['total_score'] = df_historical.groupby('student_id')['score'].transform('sum')

# Simulated NEET Scores (Replace with actual data)
df_historical['neet_rank'] = 1000 - df_historical['total_score']  # Hypothetical formula

X = df_historical[['total_score']]
y = df_historical['neet_rank']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

error = mean_absolute_error(y_test, y_pred)
print(f"NEET Rank Prediction Error: {error:.2f}")

return model
```

## ---------- 6. Main Execution ----------

if **name** == "**main**": student_id = "Student_1" # Change this to any student ID from the dataset

```
# Analyze Performance
topic_accuracy, difficulty_analysis = analyze_performance(student_id)

if topic_accuracy is not None:
    print("Topic-wise Accuracy:\n", topic_accuracy)
    print("Difficulty Level Analysis:\n", difficulty_analysis)

# Generate Recommendations
recommendations = generate_recommendations(student_id)
print("Personalized Recommendations:", recommendations)

# Define Student Persona
persona = define_student_persona(student_id)
print(f"Student Persona for {student_id}: {persona}")

# Predict NEET Rank
model = predict_neet_rank()

# Visualize Insights if data is available
if topic_accuracy is not None:
    plt.figure(figsize=(10, 5))
    sns.barplot(x="Topic", y="Accuracy", data=topic_accuracy, palette="coolwarm")
    plt.title("Topic-wise Accuracy")
    plt.xticks(rotation=45)
    plt.show()
```

READEME :

# Personalized-Student-Recommendations

Personalized Student Recommendations for NEET Preparation

**Data Overview**

This project utilizes two key datasets to analyze and provide personalized recommendations for NEET preparation based on quiz performance:

Current Quiz Data:

Contains details of a student's most recent quiz submission. Includes data such as questions, topics, responses, and the overall score for the latest quiz taken. This dataset serves as the real-time data source for the student's performance.

**Historical Quiz Data:**

Contains performance data from the last 5 quizzes taken by each student. Includes scores, response maps (which are stored as Key-Value pairs representing the question ID and the selected option ID), and other relevant metrics. This dataset serves to analyze trends and performance patterns over time.

**Task Overview**

The goal of this project is to develop a Python-based solution that analyzes the quiz performance and generates personalized recommendations for students to improve their NEET preparation.

1. Analyze the Data: Explore the dataset schema and identify patterns in student performance based on: Topics: Which topics the student excels in and which they struggle with. Difficulty Levels: How the student performs across different difficulty levels. Response Accuracy: How accurately the student responds to quiz questions.
2. Generate Insights: Weak Areas: Identify which topics or areas the student needs to focus on based on low accuracy. Improvement Trends: Track performance over time and highlight areas where the student has shown improvement. Performance Gaps: Identify areas where the student is underperforming and suggest improvements.
3. Create Recommendations: Actionable Steps: Based on the analysis, generate actionable steps that the student can take to improve their preparation: Focus on weak topics. Practice questions from specific difficulty levels. Review past quiz responses to identify patterns in mistakes. Bonus Points: Student Persona: Analyze and categorize students based on their quiz performance patterns. Generate insights on strengths and weaknesses using creative labels (e.g., "Top Performer", "Needs Improvement"). NEET Rank Prediction: Build a probabilistic model that predicts a student's potential NEET rank based on their quiz performance and historical NEET results.

**Project Features:**

Personalized Recommendations: Suggest areas to improve based on performance analysis. Performance Visualization: Visualize topic-wise accuracy and trends over time. Probabilistic NEET Rank Prediction: Predict a student's rank based on their quiz performance and simulate NEET scores. Student Persona Creation: Classify students as "Top Performer", "Average Performer", or "Needs Improvement".

**Example Output:**

Topic-wise Accuracy: A bar chart showing the student's accuracy in different topics. NEET Rank Prediction: The model's prediction of the student's NEET rank based on their performance. Personalized Recommendations: Suggested areas for improvement and actionable steps.