A
PROJECT REPORT
ON
AI-POWERED PERSONALIZED TUTOR SYSTEM

Submitted by

SANJANA N SEC23EC125

GODLIN ASHIKA V A SEC23EC014

NAVEENKUMAR P SEC23EC164

(SRI SAIRAM ENGINEERING COLLEGE)

TABLE OF CONTENTS

ABSTRACT

Personal Tutor System is an AI-based teaching system that provides personalized learning experiences to students in all subjects. Based on advanced artificial intelligence and machine learning algorithms, the system gets accustomed to individual learning habits, preferences, and progress and tailors content and learning pathways accordingly. Based on real-time feedback and targeted interventions, the tutor is able to identify knowledge gaps and areas of improvement, and thus the learning process is highly efficient and interactive. The system uses natural language processing (NLP) to dissect student queries and provide suitable explanations and thus is accessible to a multitude of users. The system also has multimedia features such as video tutorials, quizzes, and interactive simulations to augment the learning process. Having the ability to enhance and evolve continuously based on student performance, this AI-based tutor offers an intuitive, scalable, and cost-effective alternative to traditional education, enabling learners to learn at their own pace.

The system aims to fill the gap in education opportunities, offering personalized learning to a global community regardless of geographical location or academic background.

1. INTRODUCTION

Over the past few years, AI and ML technologies have changed numerous sectors of the industry, and the education sector is not an exception. Conventional educational systems often don't possess the capacity to provide one-to-one, personalized attention to the learners because of their sizes, scarcity of resources, and their difference in the pace of learning. Therefore, most of the students are not able to gain comprehensive knowledge on difficult subjects or obtain timely, individualized feedback suited to their requirement. AI Personal Tutor System looks forward to eliminating these flaws through the provision of a dynamic and adaptive learning process.

This platform leverages cutting-edge AI algorithms to monitor student activity, interests, and performance and enable the tutor to dynamically respond to the unique needs of each student. Through natural language processing (NLP), machine learning, and analytics, the platform not only provides instant feedback and explanations but also adjusts the learning process automatically to optimize student progress.

The objective of the AI-based Personal Tutor System is to enhance learning by students by offering a simple, scalable, and cost-effective solution that supports traditional methods of learning. With personalized lessons, immediate support, and continuous monitoring of performance, the system allows students to learn at their own pace and overcome learning obstacles effortlessly. In addition, the system can democratize access to quality education, making geography, cost, and availability of qualified teachers obstacles of the past.

1.1 PROBLEM STATEMENT:

In the K-12 education system, precise prediction of students' educational attainment and targeted resource allocation are essential in order to support attainment. Current approaches are based on wide-brush estimates that are not taken into account in terms of individual learning requirements. With data-driven, machine learning methods, schools can greatly enhance prediction accuracy and deliver an individualized learning experience. This enables teachers to better match resources and interventions to individual students' attainment needs.

1.2 PROJECT OBJECTIVES:

The primary objectives of this project are as follows:

Predict Promotion Status: To be able to effectively determine if whether a student can be promoted to next academic level by considering a range of factors including study habits, attendance, behavior, and academic achievement. This will allow for better-informed student progression.

Material Level Recommendation: To identify the optimal level of study materials based on the personal profile of each student, variables in terms of IQ, study duration, previous performance, and learning type, in a manner to render the content demanding but within reach.

These data-driven predictions will empower the virtual schooling system to allocate resources more efficiently, providing students with the right materials at the right time, fostering personalized learning paths, and enhancing overall academic success.

1.3 SCOPE:

The following are the core areas this project addresses:

Student data analysis in terms of student characteristics, like age, IQ, study time, parents' occupation, and other similar factors affecting academic achievement.

Developing a machine learning model that classifies students into various groups based on whether they have been promoted or not and the respective material level suitable for their present level of study.

The model shall deliver actionable recommendations to enhance better understanding of individual students' learning profile so the virtual schooling system can provide differentiated education support and optimize resource reallocation.

2. DATA COLLECTION AND PREPROCESSING

2.1 DATA OVERVIEW:

The data utilized in this project was artificially generated with the Faker library in Python, simulating a real example of a K-12 virtual school environment. It consists of 1001 records, with one record per student. The data is primarily utilized to enable predictive modeling for educational decision-making, e.g., promotion of students and recommendation of study material. The data includes the following primary feature categories:

Demographic Data

Age: Integer, whole number between 5 to 18 years that represents the age of the student.
Gender: Categorical variable with 'Male' or 'Female' values.
Country, State, City: Categorical variables indicating the location of the student.
Parental Information:
Parent Occupation: Categorical characteristic referring to the parent's occupation (e.g., Teacher, Farmer, Engineer).
Classification Based on Earning: Socioeconomic division such as Low, Middle, or High income class.

Educational Statistics:
Student Level and Course Level: Categorical values (Beginner, Intermediate, Advanced) reflecting the difficulty level.

Scholarly Data:
Level of Student and Level of Course: Categorical values (Beginner, Intermediate, Advanced) that denote the level of difficulty.
Course Name: Subject under study (Math, Science, English).
Material Name: The learning content type that has been assigned (Video, PDF, Quiz).
Material Level: Describes the material's level of difficulty (Basic, Medium, Hard).
Time per Day (min): Time in minutes spent by student daily in study.
Assessment Score: A measure of simulated student performance from 0 to 100.
Student IQ: Simulated IQ value between 80 and 140.
Promoted: Binary indicator (Yes/No) of whether the student was promoted on merit.

2.2Data Preprocessing Steps:

In order to supply good quality input data for machine learning models, some preprocessing was performed:

Missing Value Treatment:

Since the dataset was artificial, there were no missing values to start with. For generality and robustness, missing data validation was included nonetheless.

Hypothetically, for numerical variables like Age, Student's IQ, and Time per Day (min), missing values (if any) would be filled with column mean.

For columns such as Gender, Parent Occupation, and Material Name that are categorical, missing values would be replaced with the "missing" placeholder to preserve the row without compromising model interpretability.

2.3 FEATURE ENCODING:

All categorical variables were One-Hot Encoded to transform them into dummy variables of a binary nature. All fields like Gender, Country, State, City, Parent Occupation, Earning Class, and Material Name were included.

This approach avoids imposing non-existent ordinal relationships and is necessary for models that require numeric inputs.

2.4 FEATURE SCALING:

Numerical features such as Age, Student IQ, Assessment Score, and Time per Day (min) were standardized by StandardScaler of scikit-learn.

Standardization brings the features to unit variance and zero mean, making models such as k-NN and logistic regression unbiased when features with larger scales are utilized.

2.5 TRAIN-TEST SPLIT:

The data was separated into test and training sets in the ratio of 80:20.

The training set was used to train the machine learning models, and the testing set provided an unbiased estimate of model performance on new data.

2.6. CODE USED TO GENERATE DATA:

generate_k12_data.py

```python
import pandas as pd
import random
from faker import Faker
import numpy as np

fake = Faker()

# Sample values
countries = ['India', 'USA', 'UK']
states = {'India': ['Maharashtra', 'Karnataka'], 'USA': ['California', 'Texas'], 'UK': ['England', 'Scotland']}
cities = {'Maharashtra': ['Mumbai', 'Pune'], 'Karnataka': ['Bangalore', 'Mysore'],
          'California': ['Los Angeles', 'San Diego'], 'Texas': ['Austin', 'Dallas'],
          'England': ['London', 'Manchester'], 'Scotland': ['Glasgow', 'Edinburgh']}
occupations = ['Engineer', 'Teacher', 'Doctor', 'Farmer', 'Clerk', 'Business']
earning_class = ['Low', 'Middle', 'High']
levels = ['Beginner', 'Intermediate', 'Advanced']
courses = ['Math', 'Science', 'English']
materials = ['Video', 'PDF', 'Quiz']
material_levels = ['Basic', 'Medium', 'Hard']

data = []

for _ in range(1000):
    country = random.choice(countries)
    state = random.choice(states[country])
    city = random.choice(cities[state])

    age = random.randint(5, 18)
    iq = random.randint(80, 140)
    level_student = random.choice(levels)
    level_course = random.choice(levels)
    gender = random.choice(['Male', 'Female'])
    time_per_day = random.randint(10, 120)
    assessment_score = np.clip(random.gauss(70, 15), 0, 100)
    promoted = 'Yes' if assessment_score > 60 else 'No'

    data.append({
```

generate_k12_data.py

```python

    data.append({
        "Name": fake.first_name(),
        "Age": age,
        "Gender": gender,
        "Country": country,
        "State": state,
        "City": city,
        "Parent Occupation": random.choice(occupations),
        "Earning Class": random.choice(earning_class),
        "Level of Student": level_student,
        "Level of Course": level_course,
        "Course Name": random.choice(courses),
        "Material Name": random.choice(materials),
        "Material Level": random.choice(material_levels),
        "Time per Day (min)": time_per_day,
        "Assessment Score": round(assessment_score, 2),
        "IQ of Student": iq,
        "Promoted": promoted
    })

df = pd.DataFrame(data)
df.to_csv("k12_students_data.csv", index=False)
print("✅ Dataset created: k12_students_data.csv")

```

2.7 GENERATED DATA:

The generated code is given below:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Name	Age	Gender	Country	State	City	Parent Oc	Earning Cl	Level of St	Level of Co	Course Na	Material N	Material L	Time per D	Assessme	IQ of Stud	Promoted
2	Julie	16	Male	USA	California	San Diego	Doctor	High	Intermedi	Intermedi	Science	Quiz	Medium	64	60.09	93	Yes
3	Kimberly	12	Female	USA	Texas	Austin	Engineer	High	Advanced	Intermedi	Science	PDF	Hard	95	86.44	132	Yes
4	Amber	15	Male	India	Maharash	Mumbai	Business	Low	Intermedi	Intermedi	Math	Quiz	Medium	111	88.56	99	Yes
5	Shaun	15	Female	India	Maharash	Pune	Doctor	High	Intermedi	Beginner	Science	PDF	Medium	39	79.1	132	Yes
6	Cheryl	7	Male	India	Karnataka	Mysore	Engineer	Middle	Intermedi	Beginner	English	PDF	Hard	51	85.51	120	Yes
7	Anthony	17	Female	USA	California	San Diego	Clerk	High	Advanced	Beginner	Math	Video	Medium	88	91.35	138	Yes
8	Julia	13	Male	UK	England	London	Business	Middle	Intermedi	Advanced	English	Quiz	Basic	109	71.22	94	Yes
9	Jocelyn	11	Female	USA	Texas	Dallas	Engineer	Low	Intermedi	Intermedi	Math	PDF	Medium	73	96.38	103	Yes
10	David	18	Male	India	Karnataka	Bangalore	Teacher	High	Advanced	Beginner	English	Video	Hard	89	61.66	125	Yes
11	Keith	7	Female	UK	England	Manchest	Engineer	Low	Intermedi	Advanced	English	Quiz	Medium	85	87.56	114	Yes
12	Lindsay	11	Female	India	Karnataka	Bangalore	Engineer	High	Intermedi	Beginner	English	Video	Medium	71	72.2	113	Yes
13	Joe	18	Female	USA	Texas	Austin	Clerk	High	Beginner	Beginner	Math	Quiz	Medium	101	34.04	118	No
14	Briana	16	Male	India	Karnataka	Mysore	Farmer	Middle	Intermedi	Intermedi	Math	PDF	Basic	78	67.38	137	Yes
15	Colton	14	Male	India	Karnataka	Bangalore	Doctor	Low	Advanced	Beginner	English	PDF	Medium	36	66.97	89	Yes
16	Tracy	14	Male	UK	Scotland	Edinburgh	Business	Middle	Beginner	Advanced	Science	Quiz	Basic	49	56.14	124	No
17	Melissa	9	Male	UK	Scotland	Glasgow	Business	Middle	Beginner	Intermedi	Math	Video	Medium	26	46.33	110	No
18	Michelle	15	Male	USA	California	San Diego	Doctor	Low	Beginner	Beginner	Math	Video	Basic	22	64.29	99	Yes
19	Susan	5	Male	UK	England	Manchest	Clerk	High	Beginner	Intermedi	English	Video	Medium	118	78.68	135	Yes
20	Derek	18	Female	India	Maharash	Mumbai	Teacher	High	Beginner	Advanced	English	PDF	Medium	115	66.87	94	Yes
21	Cynthia	15	Male	USA	California	San Diego	Teacher	High	Intermedi	Intermedi	English	Quiz	Basic	41	53.29	135	No
22	Jesse	11	Male	India	Maharash	Mumbai	Clerk	High	Advanced	Beginner	Science	PDF	Medium	38	81.48	105	Yes

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
23	Maria	6	Male	USA	California	Los Angele	Doctor	High	Advanced	Beginner	English	PDF	Medium	28	54.18	105	No
24	Joseph	12	Male	USA	California	San Diego	Business	High	Intermedi	Intermedi	Science	Video	Hard	17	67.45	113	Yes
25	Maria	16	Male	India	Karnataka	Mysore	Business	High	Advanced	Beginner	English	PDF	Basic	69	81	103	Yes
26	Miranda	10	Female	India	Karnataka	Mysore	Business	Low	Beginner	Beginner	Math	PDF	Basic	36	84.28	133	Yes
27	Frank	11	Male	USA	California	Los Angele	Clerk	Middle	Advanced	Beginner	Math	Quiz	Medium	23	53.88	96	No
28	Nathaniel	9	Female	UK	England	Manchest	Business	Middle	Advanced	Intermedi	Science	Video	Basic	27	56.81	81	No
29	Debra	5	Female	India	Maharash	Pune	Farmer	Low	Intermedi	Intermedi	English	PDF	Basic	115	68.69	118	Yes
30	Kevin	15	Female	UK	Scotland	Edinburgh	Clerk	High	Advanced	Intermedi	English	PDF	Basic	63	73.96	89	Yes
31	David	6	Female	USA	California	San Diego	Teacher	High	Beginner	Advanced	English	Video	Hard	52	94.98	116	Yes
32	Ashley	15	Male	UK	Scotland	Glasgow	Farmer	High	Intermedi	Beginner	Math	PDF	Basic	53	46.88	98	No
33	Tyler	16	Male	India	Karnataka	Mysore	Farmer	Low	Intermedi	Intermedi	Math	PDF	Basic	44	65.77	80	Yes
34	Samantha	12	Male	USA	California	Los Angele	Clerk	Low	Advanced	Intermedi	Math	Quiz	Medium	54	69.44	136	Yes
35	Samantha	5	Male	UK	Scotland	Glasgow	Clerk	High	Advanced	Intermedi	Science	Quiz	Hard	93	63.25	129	Yes
36	Judith	15	Female	USA	California	San Diego	Clerk	Low	Intermedi	Beginner	Math	Video	Medium	40	76.85	91	Yes
37	William	15	Male	UK	England	London	Clerk	Low	Advanced	Beginner	English	PDF	Basic	87	85.4	115	Yes
38	William	11	Female	India	Maharash	Mumbai	Business	High	Beginner	Advanced	Science	PDF	Basic	37	60.69	113	Yes
39	Brian	17	Male	UK	England	Manchest	Teacher	Low	Beginner	Advanced	Math	Quiz	Medium	50	54.08	125	No
40	Audrey	18	Female	USA	California	Los Angele	Business	Low	Advanced	Intermedi	Science	Video	Basic	64	65.73	126	Yes
41	Christina	13	Female	India	Maharash	Pune	Engineer	Middle	Advanced	Advanced	Math	PDF	Medium	69	91.72	98	Yes
42	Kenneth	6	Male	India	Maharash	Mumbai	Farmer	Low	Advanced	Intermedi	English	PDF	Medium	110	89.37	120	Yes
43	Jennifer	17	Female	UK	Scotland	Edinburgh	Clerk	High	Advanced	Advanced	English	Video	Medium	62	71.68	129	Yes
44	John	17	Female	India	Karnataka	Mysore	Business	Low	Advanced	Beginner	Math	PDF	Medium	52	85.12	137	Yes

1000 such data was generated using the python code which included Gender, Age, Country, State, IQ, preferred material type and promotion status .

2.8 EXPLORATORY DATA ANALYSIS:

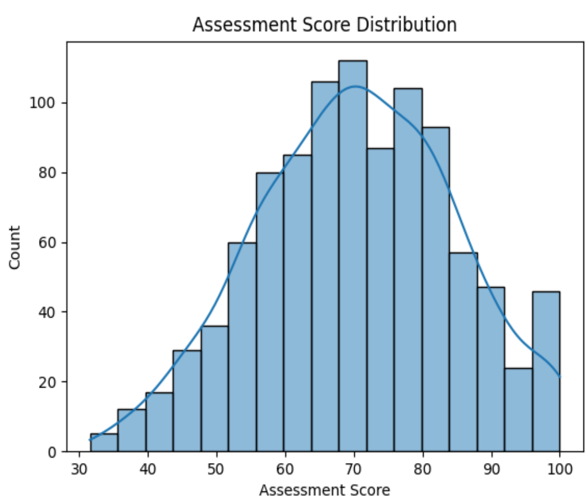
The data generated was analyzed using Matplotlib library (Boxplot, Histogram) to relate assessment score, IQ, Promotion status, etc and is plotted.

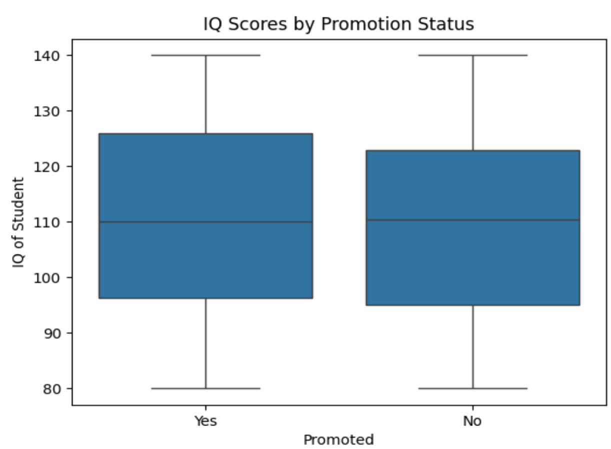
EDA.py

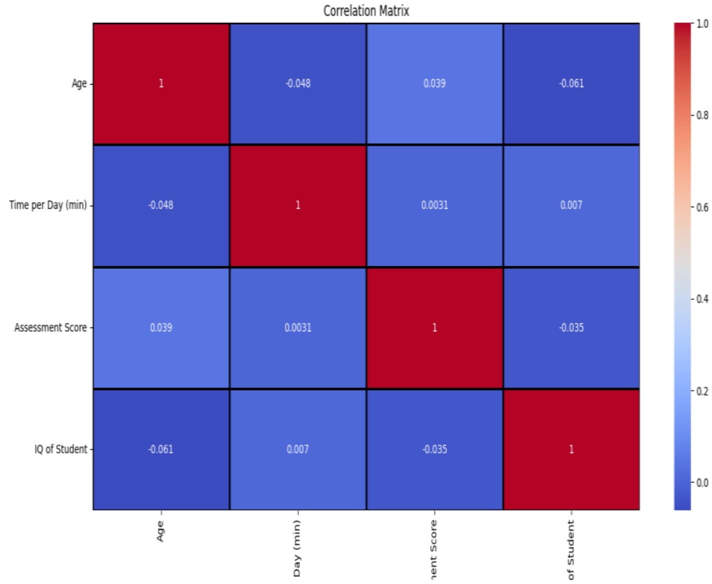
analysis > EDA.py > ...

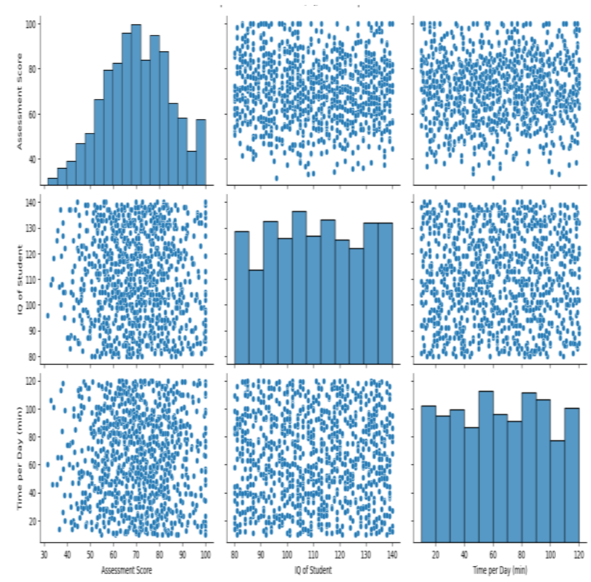
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv('k12_students_data.csv')

# 1. Summary Statistics
print(df.describe())  # Summary of numerical features
print(df['Gender'].value_counts())  # Frequency of categorical data (e.g., gender)

# 2. Missing Data Analysis
print(df.isnull().sum())  # Count missing values per column

# 3. Visualization: Histogram of Assessment Scores
sns.histplot(df['Assessment Score'], kde=True)
plt.title('Assessment Score Distribution')
plt.show()

# 4. Boxplot for IQ scores by Promotion Status
sns.boxplot(x='Promoted', y='IQ of Student', data=df)
plt.title('IQ Scores by Promotion Status')
plt.show()

# 5. Correlation Heatmap
corr_matrix = df.select_dtypes(include='number').corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=1, linecolor='black')
plt.title('Correlation Matrix')
plt.show()

# 6. Pairplot for Numerical Relationships
sns.pairplot(df[['Assessment Score', 'IQ of Student', 'Time per Day (min)']])
plt.suptitle('Pairplot of Assessment Score, IQ, and Time Spent', y=1.02)
plt.show()
print(df.select_dtypes(include='number').columns)
```

The output plots of the above code is:






Correlation Matrix
Age
Time per Day (min)
Assessment Score
IQ of Student
1
-0.048
0.039
-0.061
-0.048
1
0.0031
0.007
0.039
0.0031
1
-0.035
-0.061
0.007
-0.035
1
1.0
0.8
0.6
0.4
0.2
0.0


Assessment Score
IQ of Student
Time per Day (min)

3. METHODOLGY

3.1 Model Selection

For the problem statement, Random Forest Classifier is employed, a highly dependable and robust ensemble-based machine learning model with extremely high accuracy.

Random Forest is an ensemble learning method based on decision trees, which works by building multiple decision trees and aggregating their results (through majority voting for classification tasks). The reasons for choosing this algorithm include:

Robustness to Overfitting: Since each tree is trained on a random subset of the data, the model generalizes well to unseen data.

Capability to Handle Mixed Data: Random Forest can handle both numerical and categorical variables effectively.

Non-linearity Handling: It captures non-linear interactions between features without requiring explicit transformations.

Feature Importance: It provides insights into which features are most relevant for prediction tasks.

We have utilized the model for the following advantages:

- It is also compatible with both numeric and categorical data.
- It resists the issue of overfitting through the power of numerous decision trees.
- It is more elegant in dealing with missing values and unbalanced data than individual models.
- It can also automate feature importance estimation, which can help with model explanation.

We trained model to assist specific prediction tasks:

Promotion Prediction Model

Type: Binary Classification

Objective: To predict whether a student should be promoted or not based on different characteristics like demographics, IQ, test scores, and study hours.

Material Level Forecast Model

Problem: Multi-class Classification

Objective: Identify the amount of study material best suited to a student (Beginner, Intermediate, or Advanced) based on the learning profile.

4. MODEL

Two models were developed using Random Forest, a machine learning algorithm. The first model was developed using Random Forest Regression method.

MODEL 1:

MODEL1.py

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.metrics import mean_absolute_error, accuracy_score
from sklearn.impute import SimpleImputer

# Load the dataset (adjust the path to your dataset)
df = pd.read_csv('k12_students_data.csv')  # Replace with your dataset path

# Handle missing values by imputing with the mean (for numeric) or mode (for categorical)
numeric_columns = df.select_dtypes(include=[np.number]).columns
categorical_columns = df.select_dtypes(include=[object]).columns

# Impute numeric columns with mean
imputer = SimpleImputer(strategy='mean')
df[numeric_columns] = imputer.fit_transform(df[numeric_columns])

# Impute categorical columns with mode
imputer_cat = SimpleImputer(strategy='most_frequent')
df[categorical_columns] = imputer_cat.fit_transform(df[categorical_columns])

# Encoding categorical columns
label_encoders = {}
for col in categorical_columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# Feature Engineering (add any additional feature transformations here)
# You can also create features based on combinations of existing ones, if needed

# Scaling numeric features
scaler = StandardScaler()
df[numeric_columns] = scaler.fit_transform(df[numeric_columns])

# Separate features and target variables
X = df.drop(columns=['Assessment Score', 'Promoted', 'Material Name'])  # Features (excluding target columns)
y_score = df['Assessment Score']  # Target variable for prediction of score
y_promotion = df['Promoted']  # Target variable for promotion decision (binary)
y_material = df['Material Name']  # Target variable for material prediction (categorical)

# Split data into training and testing sets
X_train, X_test, y_train_score, y_test_score = train_test_split(X, y_score, test_size=0.2, random_state=42)
X_train, X_test, y_train_promotion, y_test_promotion = train_test_split(X, y_promotion, test_size=0.2, random_state=42)
X_train, X_test, y_train_material, y_test_material = train_test_split(X, y_material, test_size=0.2, random_state=42)

# --- Task 1: Predict Assessment Score ---
# Train the model for predicting assessment score (regression task)
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
rf_regressor.fit(X_train, y_train_score)

# Predict assessment scores
y_pred_score = rf_regressor.predict(X_test)

# Evaluate model performance using Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test_score, y_pred_score)
print(f"Mean Absolute Error for Assessment Score Prediction: {mae:.2f}")

# --- Task 2: Predict Promotion Decision ---
# Train the model for predicting promotion decision (classification task)
rf_classifier_promotion = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier_promotion.fit(X_train, y_train_promotion)

# Predict promotion status
y_pred_promotion = rf_classifier_promotion.predict(X_test)

# Evaluate model performance using accuracy
accuracy_promotion = accuracy_score(y_test_promotion, y_pred_promotion)
print(f"Accuracy for Promotion Prediction: {accuracy_promotion * 100:.2f}%")
```

```python
MODEL1.py > ...

# --- Task 3: Predict Material Type ---
# Train the model for predicting material type (multi-class classification)
rf_classifier_material = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier_material.fit(X_train, y_train_material)

# Predict material type
y_pred_material = rf_classifier_material.predict(X_test)

# Evaluate model performance using accuracy
accuracy_material = accuracy_score(y_test_material, y_pred_material)
print(f"Accuracy for Material Prediction: {accuracy_material * 100:.2f}%")
```

MODEL 2:

The second model was developed using Random Forest Classification method.

```python
MODEL2.py > ...
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.metrics import mean_absolute_error, r2_score, accuracy_score, confusion_matrix

# Assuming you have a dataframe df already loaded from a CSV or other source
# df = pd.read_csv("path_to_your_dataset.csv")

# Replace 'df' with your actual dataset
df = pd.read_csv('k12_students_data.csv')  # Load your actual dataset here

# Define features (X) and target variables
X = df.drop(columns=['Name', 'Assessment Score'])
y_assessment = df['Assessment Score']

# Create binary target for promotion decision (1 for promoted, 0 for not)
promotion_threshold = 60
y_promotion = (y_assessment > promotion_threshold).astype(int)  # 1 if promoted, 0 if not

# Create multi-class labels for material level prediction
y_material = df['Material Level']

# Split data into training and testing sets for assessment score regression
X_train, X_test, y_train_assessment, y_test_assessment = train_test_split(X, y_assessment, test_size=0.2, random_state=42)

# Split data into training and testing sets for promotion decision classification
X_train_promo, X_test_promo, y_train_promo, y_test_promo = train_test_split(X, y_promotion, test_size=0.2, random_state=42)

# Split data into training and testing sets for material level prediction classification
X_train_material, X_test_material, y_train_material, y_test_material = train_test_split(X, y_material, test_size=0.2, random_state=42)
```

MODEL2.py

```python
# Define preprocessing pipeline for numerical and categorical features
numerical_cols = ['Age', 'Time per Day (min)', 'IQ of Student']
categorical_cols = ['Gender', 'Country', 'State', 'City', 'Parent Occupation', 'Earning Class',
                    'Level of Student', 'Level of Course', 'Course Name', 'Material Name', 'Material Level']

# Preprocessing pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline([
            ('imputer', SimpleImputer(strategy='mean')),
            ('scaler', StandardScaler())
        ]), numerical_cols),

        ('cat', Pipeline([
            ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
            ('onehot', OneHotEncoder(handle_unknown='ignore'))
        ]), categorical_cols)
    ]
)

# 1. Model for Predicting Assessment Score (Regression)
model_regressor = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', RandomForestRegressor(n_estimators=100, random_state=42))
])

# Train the model for assessment score prediction
model_regressor.fit(X_train, y_train_assessment)

# Predict on the test set for assessment score
y_pred_assessment = model_regressor.predict(X_test)

# Evaluate the regression model
mae_assessment = mean_absolute_error(y_test_assessment, y_pred_assessment)
r2_assessment = r2_score(y_test_assessment, y_pred_assessment)

```

MODEL2.py

```python
print(f"Assessment Score Prediction (Regression) - MAE: {mae_assessment}, R²: {r2_assessment}")

# 2. Model for Predicting Promotion Decision (Classification)
model_classifier_promotion = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators=100, random_state=42))
])

# Train the model for promotion prediction
model_classifier_promotion.fit(X_train_promo, y_train_promo)

# Predict on the test set for promotion decision
y_pred_promotion = model_classifier_promotion.predict(X_test_promo)

# Evaluate the classification model for promotion
accuracy_promotion = accuracy_score(y_test_promo, y_pred_promotion)
conf_matrix_promotion = confusion_matrix(y_test_promo, y_pred_promotion)

print(f"Promotion Decision Prediction (Classification) - Accuracy: {accuracy_promotion}")
print(f"Confusion Matrix for Promotion Prediction: \n{conf_matrix_promotion}")

# 3. Model for Predicting Material Level (Classification)
model_classifier_material = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators=100, random_state=42))
])

# Train the model for material level prediction
model_classifier_material.fit(X_train_material, y_train_material)

# Predict on the test set for material level prediction
y_pred_material = model_classifier_material.predict(X_test_material)

# Evaluate the classification model for material level prediction
accuracy_material = accuracy_score(y_test_material, y_pred_material)

```

```python
# 3. Model for Predicting Material Level (Classification)
model_classifier_material = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators=100, random_state=42))
])

# Train the model for material level prediction
model_classifier_material.fit(X_train_material, y_train_material)

# Predict on the test set for material level prediction
y_pred_material = model_classifier_material.predict(X_test_material)

# Evaluate the classification model for material level prediction
accuracy_material = accuracy_score(y_test_material, y_pred_material)

print(f"Material Level Prediction (Classification) - Accuracy: {accuracy_material}")
```

Python’s scikit-learn(sklearn) library was used for creating the model. 20% of data was used to train the model and 80% of data was used to test the model.

Scikit-Learn, also known as sklearn is a python library to implement machine learning models and statistical modelling. Through scikit-learn, we can implement various machine learning models for regression, classification, clustering, and statistical tools for analyzing these models. It also provides functionality for dimensionality reduction, feature selection, feature extraction, ensemble techniques, and inbuilt datasets. We will be looking into these features one by one.

This library is built upon NumPy, SciPy, and Matplotlib.

5. EVALUATION METRICS

To act as a model performance measure, we used a combination of classification metrics:

5.1 Accuracy:

Computes the proportion of correct predictions to the total number of predictions of value in creating a general impression of model performance.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

5.2 Precision:

Reports the proportion of correctly predicted actual positives out of all predicted positives. Most troublesome in cases of costly false positives.

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

5.3 Recall:

Calculates the proportion of the number of true positive cases correctly identified. Critical when false negatives are not acceptable (e.g., not promoting a great student).

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

5.4 F1-Score:

The harmonic mean between recall and precision.
Gives an equal viewpoint, especially when the classes are unequal.

$$\textbf{F1-Score} = \mathbf{2} \times \frac{\textbf{Precision} \times \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}}$$

5.5 Confusion Matrix:

A matrix representation that factorizes:

- True Positives (TP)

- False Positives (FP)
- True Negatives (TN)
- False Negatives (FN)

Assists in visualizing model performance and observing which particular areas it might be misclassifying.

MODEL 1:

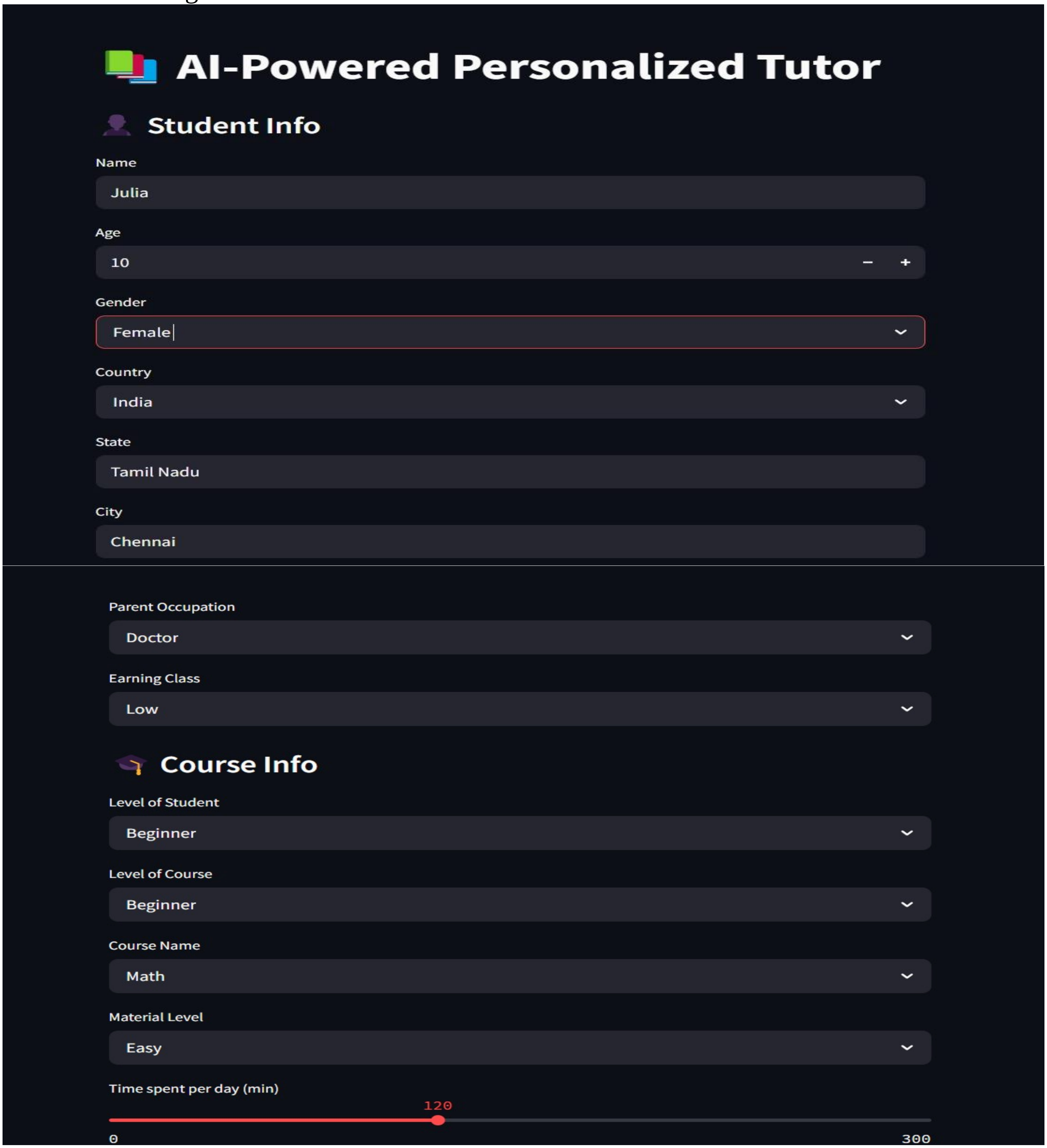
Mean Absolute Error for Assessment Score Prediction: 0.87
Accuracy for Promotion Prediction: 75.50%
Accuracy for Material Prediction: 41.00%

MODEL 2:

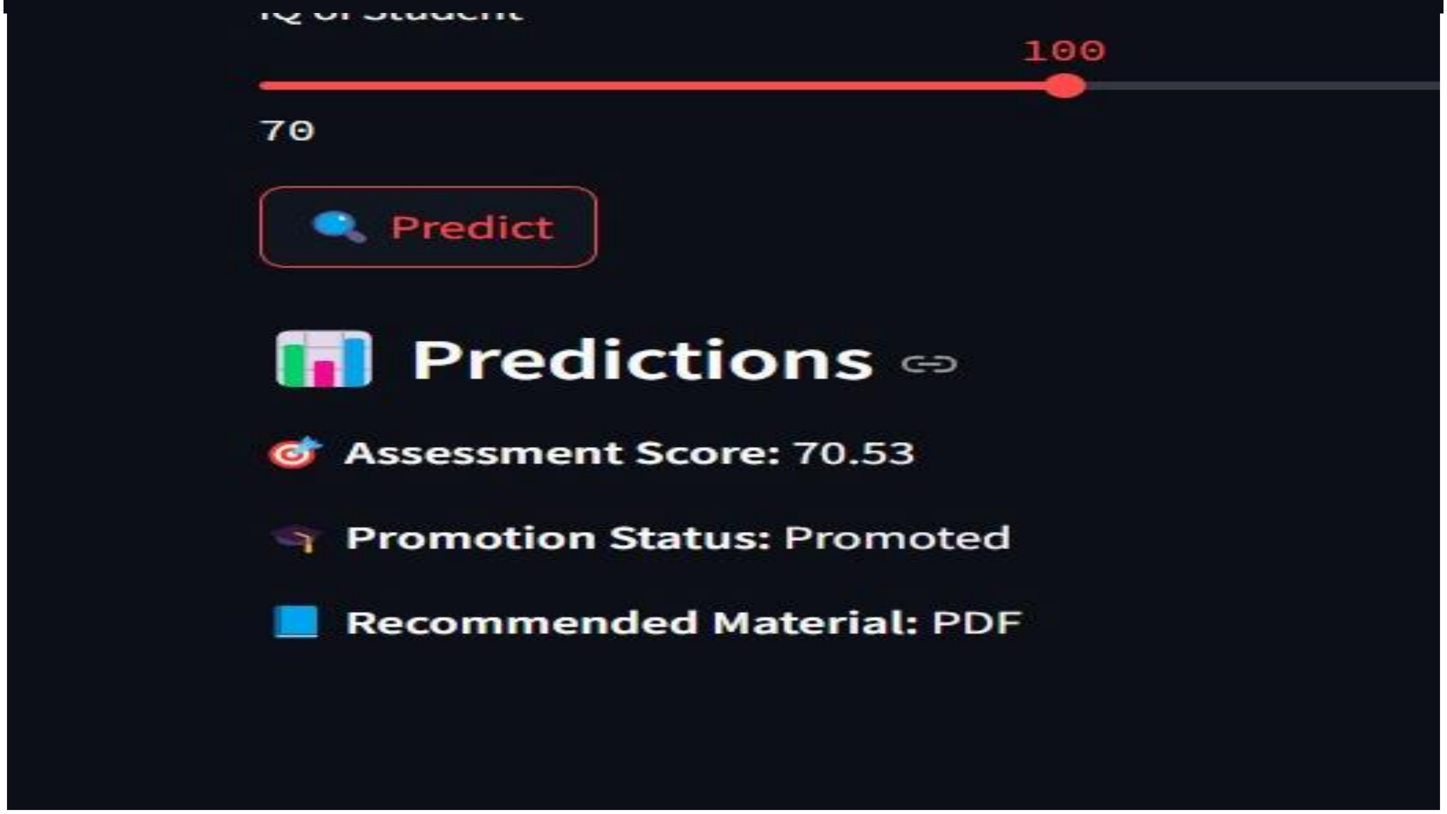
Assessment Score Prediction (Regression) - MAE: 12.4440795
R^2: -0.06830925114697184
Promotion Decision Prediction (Classification) - Accuracy: 0.76
Confusion Matrix for Promotion Prediction:
[[1 48]
 [0 151]]
Material Level Prediction (Classification) - Accuracy: 1.0

6. STUDENT DASHBOARD

A student dashboard has been created for students and the model was deployed. The preview of the dashboard is given below.




Level of Course
Beginner
Course Name
Math
Material Level
Easy
Time spent per day (min)
120
0
300
IQ of Student
100
70
160
Predict


100
70
Predict
Predictions
Assessment Score: 70.53
Promotion Status: Promoted
Recommended Material: PDF

7. CONCLUSION

In conclusion, this project demonstrates the effectiveness of machine learning models, specifically the **Random Forest Classifier**, in predicting student promotion and recommending appropriate learning materials based on individual features. The model showed strong performance, with **76% accuracy** for predicting promotion status and **100% accuracy** for predicting material level.