Student Performance Prediction

Collage Of Computing Science

Department Of Software Engineering

Course Title: Fundamentals of Machine Learning

Course Code: Seng4091

Individual Assignment

Name	ID
Elias Aynekulu	1402639

Submitted to: Derbew F.

Problem Definition

In education, assessing students' academic performance is crucial to understanding and improving learning outcomes. However, accurately predicting exam performance can be challenging due to the multifaceted factors that influence it, such as study habits, motivation, and access to resources.

This project aims to predict students' exam scores using machine learning. By leveraging historical data and performance indicators, the model helps educators and parents identify key areas for improvement and make data-driven interventions to enhance academic outcomes.

Data Source and Description

Source of Data

The dataset StudentPerformanceFactors.csv was obtained from **Kaggle**, a public data-sharing platform that hosts datasets and competitions for machine learning projects.

- **Source**: Kaggle
- **Accessibility**: Publicly available, suitable for research and educational purposes.

License / Terms of Use

- The dataset is public and can be freely used for educational and research purposes.

Data Structure

The dataset is organized in a **tabular format**, with rows representing individual student records and columns representing performance-related features.

Features

The dataset contains both **numerical** and **categorical** features. These include:

Feature Name	Data Type	Description
Hours_Studied	Numerical	Number of hours studied per week.
Attendance	Numerical	Attendance percentage of the student.
Sleep_Hours	Numerical	Average hours of sleep per night.
Previous_Scores	Numerical	Average scores from previous exams.
Parental_Involvement	Categorical (1-5)	Level of parental involvement in the student's education (ordinal).

Feature Name	Data Type	Description
Motivation_Level	Categorical (1-5)	Self-reported motivation level (ordinal).
Teacher_Quality	Categorical (1-5)	Quality of teaching as perceived by the student (ordinal).
Internet_Access	Binary	Whether the student has access to the internet (0 = No, 1 = Yes).
Extracurricular_Activities	Binary	Participation in extracurricular activities (0 = No, 1 = Yes).
Family_Income	Numerical	Annual family income (in USD).
Learning_Disabilities	Binary	Whether the student has any learning disabilities (0 = No, 1 = Yes).
Physical_Activity	Numerical	Hours of physical activity per week.
Distance_from_Home	Numerical	Distance from home to school (in kilometers).
Access_to_Resources	Binary	Access to educational resources (0 = No, 1 = Yes).
Gender	Binary	Gender of the student (0 = Female, 1 = Male).
Parental_Education_Level	Categorical	Parent's highest education level (ordinal: 0 = Low, 1 = Medium, 2 = High).
Peer_Influence	Binary	Influence of peers on the student (0 = No, 1 = Yes).
School_Type	Binary	Type of school (0 = Public, 1 = Private).
Tutoring_Sessions	Numerical	Weekly hours spent in tutoring sessions.

Exploratory Data Analysis (EDA)

EDA was conducted to understand the structure, relationships, and anomalies in the data. Key findings are as follows:

Distributions

- Numerical features like Hours_Studied and Attendance were visualized using histograms.
- These visualizations helped identify skewness and concentration of data points around specific ranges.

```
# Visualize distributions of numerical features
numerical_features = ['Hours_Studied', 'Attendance', 'Previous_Scores', 'Sleep_Hours',
'Exam_Score']
df[numerical_features].hist(bins=20, figsize=(12, 8))
plt.suptitle("Distribution of Numerical Features")
plt.show()
```

Relationships

Scatterplots were used to explore relationships between features and the target variable (Exam_Score).

- Strong positive correlation observed between Hours_Studied and Exam_Score.

```
# Scatter plots for numerical features vs. Exam_Score
plt.figure(figsize=(15, 10))
for i, column in enumerate(numerical_features[:-1], 1):
    plt.subplot(3, 2, i)
    sns.scatterplot(data=df, x=column, y='Exam_Score', alpha=0.6, color='green')
    plt.title(f'{column} vs. Exam_Score')
    plt.xlabel(column)
    plt.ylabel('Exam Score')
plt.tight_layout()
plt.show()
```

Outliers

Outliers in numerical data were identified using boxplots and handled using the Interquartile Range (IQR) method.

```
# Detect outliers using boxplots for numerical features
plt.figure(figsize=(15, 10))
for i, column in enumerate(numerical_features, 1):
    plt.subplot(3, 2, i)
    sns.boxplot(data=df, y=column, color='lightblue')
    plt.title(f'Boxplot of {column}')
    plt.ylabel(column)
plt.tight_layout()
plt.show()
```

Correlation Matrix

A heatmap of feature correlations highlighted significant relationships, such as Previous_Scores correlating positively with Exam_Score.

```
# Correlation matrix for numerical features
corr_matrix = df[numerical_features].corr()

# Visualize the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix (Numerical Features Only)')
plt.show()
```

Preprocessing Steps

Missing Values

- **Categorical Features**: Missing values were replaced with the mode.
- **Numerical Features**: Missing values were replaced with the median to avoid distortion caused by extreme values.

```
# Check for missing values
print("Missing Values in the Dataset:")
print(df.isnull().sum())

# Handle missing values
categorical_missing = ["Teacher_Quality", "Parental_Education_Level",
"Distance_from_Home"]
for col in categorical_missing:
    df[col].fillna(df[col].mode()[0], inplace=True)

numerical_missing = ["Family_Income", "Physical_Activity"]
for col in numerical_missing:
    df[col].fillna(df[col].median(), inplace=True)
```

Encoding

Categorical data was encoded:

- Ordinal features (e.g., Motivation_Level) were mapped to numerical scales.

```
# Encode ordinal features
ordinal_mapping = {"low": 0, "medium": 1, "high": 2}
ordinal_cols = ["Motivation_Level", "Parental_Involvement", "Teacher_Quality"]
for col in ordinal_cols:
    df[col] = df[col].map(ordinal_mapping)
```

Scaling

Numerical features were standardized to improve model performance.

```
from sklearn.preprocessing import StandardScaler

# Standardize numerical features
scaler = StandardScaler()
numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])
```

Model Selection and Training

Model Used

Random Forest Regressor was chosen for its ability to handle nonlinear relationships and interpret feature importance.

Training

The data was split into training (80%) and testing (20%) sets, ensuring the model had sufficient data for learning and validation.

```
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X = df.drop('Exam_Score', axis=1)
y = df['Exam_Score']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Model Evaluation

Model Evaluation Metrics:

Mean Absolute Error: 1.17

Mean Squared Error: 5.19

R-squared: 0.63

Deployment

Backend

- Built using FastAPI.
- Exposes a /predict endpoint for predictions.
- Deployed on Render:https://ml-8-ztbs.onrender.com

Frontend

- A simple HTML form allows users to input data, which is sent as a POST request to the FastAPI backend.
- Hosted alongside the backend.

Future Improvements

1. Include more features (e.g., Study Environment, Parental Support).
2. Experiment with advanced models like Gradient Boosting or Neural Networks.
3. Implement real-time monitoring and logging in deployment.