**Predict Student Performance using Machine Learning & Generate AI-Based Insights**

**Name:** Pooja S  
**Date:** 22/10/2025

## ****1. Abstract****

This project aims to predict whether a student will pass or fail based on academic and demographic data using a Machine Learning model. The Random Forest Classifier algorithm was used due to its high accuracy and ability to determine feature importance.  
Additionally, AI-based insights are generated to help students understand which factors most influence their academic outcomes and how to improve them.

## ****2. Objective****

* Predict student pass/fail outcomes using machine learning.
* Analyze and identify features that most influence student performance.
* Provide AI-based suggestions for students to improve academic outcomes.

## ****3. Dataset Description****

**Source:** [Kaggle – Students Performance in Exams](https://www.kaggle.com/datasets/spscientist/students-performance-in-exams)

**Dataset Features:**

| **Feature** | **Description** |
| --- | --- |
| Gender | Male or Female |
| race/ethnicity | Student’s ethnic group |
| parental level of education | Highest education level of parents |
| Lunch | Standard or Free/Reduced |
| test preparation course | Completed or None |
| math score | Score in Math exam |
| reading score | Score in Reading exam |
| writing score | Score in Writing exam |
| average\_score | Average of math, reading, and writing scores (calculated) |
| pass\_fail | Target variable: 1 = Pass, 0 = Fail |

## ****4. Methodology****

### ****Step 1: Load Dataset****

* Imported the dataset into Python using Pandas.
* Checked the first few rows to understand the structure of the data.

**Code:**

# Data handling

import pandas as pd

import numpy as np

# Machine Learning

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

# Visualization

import matplotlib.pyplot as plt

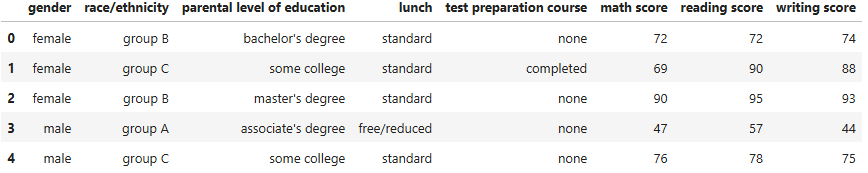
import seaborn as sns

# Load the dataset

data = pd.read\_csv("F:/ZeAI/Zyla/StudentsPerformance/StudentsPerformance.csv")

data.head()

### ****Sample Output:****



### ****Step 2: Data Preprocessing****

* Checked for missing values (none found).
* Encoded categorical variables into numbers using LabelEncoder.
* Calculated average\_score for each student.
* Created pass\_fail target column based on average score ≥50 → Pass, else Fail.

**Code:**

# Calculate average score

data['average\_score'] = (data['math score'] + data['reading score'] + data['writing score']) / 3

# create Pass/Fail column

data ['pass\_fail'] = data['average\_score'].apply(lambda x: 1 if x >= 50 else 0)

le = LabelEncoder()

categorical\_cols = ['gender', 'race/ethnicity', 'parental level of education', 'lunch', 'test preparation course']

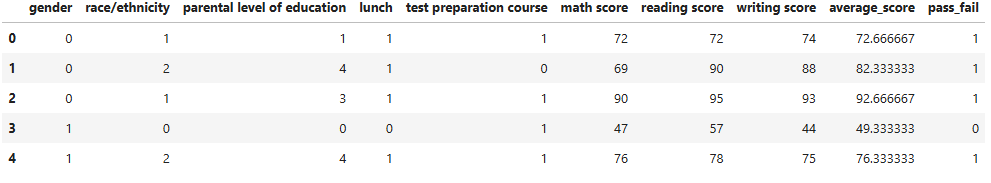
for col in categorical\_cols:

    data[col] = le.fit\_transform(data[col])

# Check the dataset after encoding

data.head()

### ****Sample Output:****



### ****Step 3: Train/Test Split****

### Split the dataset into **80% training** and **20% testing** sets for model training and evaluation.

**Code:**

# Features (X) – all columns except 'average\_score' and 'pass\_fail'

X = data[['gender', 'race/ethnicity', 'parental level of education', 'lunch',

          'test preparation course', 'math score', 'reading score', 'writing score']]

# Target (y) – the column to predict

y = data['pass\_fail']

# 80% training, 20% testing

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Check the shape

print("X\_train:", X\_train.shape)

print("X\_test:", X\_test.shape)

print("y\_train:", y\_train.shape)

print("y\_test:", y\_test.shape)

**Sample Output:**

### 

### ****Step 4: Model Selection , Training & Prediction****

* Chose **Random Forest Classifier** for its high accuracy and interpretability.
* Trained the model on the training dataset.
* Predicted pass/fail outcomes on the test set.

**Code:**

# Initialize the model

model = RandomForestClassifier(random\_state=42)

# Train the model

model.fit(X\_train, y\_train)

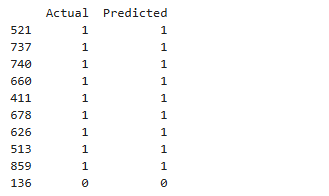
y\_pred = model.predict(X\_test)

# Compare actual vs predicted

comparison = pd.DataFrame({'Actual': y\_test, 'Predicted': y\_pred})

print(comparison.head(10))

**Sample Output:**



**Step 5: Evaluation**

* Evaluated performance using **Accuracy, Confusion Matrix, and Classification Report**.

**Code:**

# Accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

# Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")

plt.xlabel("Predicted")

plt.ylabel("Actual")

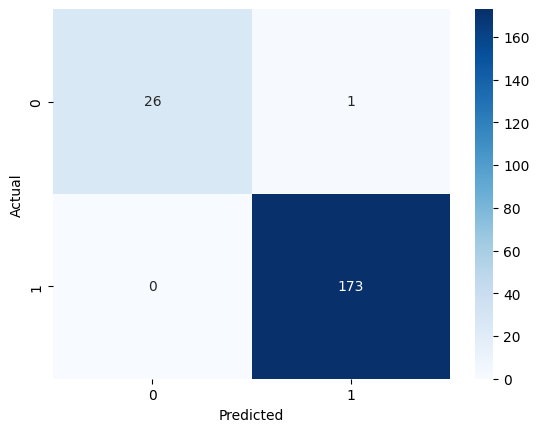
plt.show()

# Classification Report

print(classification\_report(y\_test, y\_pred))

**Sample Output:**

Accuracy: 0.995



### 

### ****Step 6: Feature Importance & Insights****

* Identified which features most impact student performance.
* Visualized feature importance to guide improvement suggestions.

**Code:**

# Get feature importance

importance = model.feature\_importances\_

# Combine feature names and importance

feature\_importance = pd.DataFrame({

    'Feature': X.columns,

    'Importance': importance

}).sort\_values(by='Importance', ascending=False)

# Display feature importance

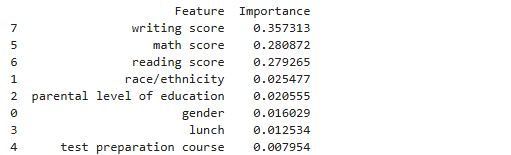
print(feature\_importance)

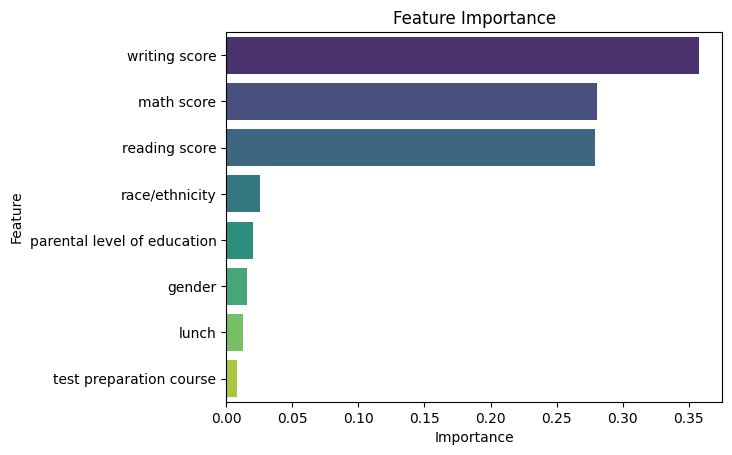
sns.barplot(x='Importance', y='Feature', data=feature\_importance, palette='viridis')

plt.title("Feature Importance")

plt.show()

**Sample Output:**

****



## ****5. AI-Based Insights & Suggestions****

* **Writing Score (0.36 importance)** → Focus on improving writing skills to increase pass probability.
* **Math Score (0.28 importance)** → Practice math regularly or seek extra help.
* **Reading Score (0.28 importance)** → Improve reading comprehension and study habits.
* Minor features (Race/Ethnicity, Parental Education, Gender, Lunch, Test Preparation Course) → Can be used to guide additional support.

**Code:**

# Create suggestions based on feature importance

print("\nAI-Based Insights & Suggestions:\n")

for index, row in feature\_importance.iterrows():

    feature = row['Feature']

    importance = row ['Importance']

    if importance > 0.2:

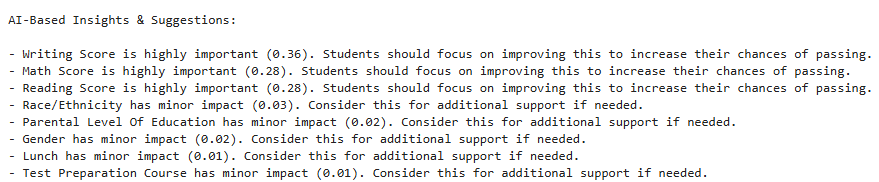
        print(f"- {feature.title()} is highly important ({importance:.2f}). Students should focus on improving this to increase their chances of passing.")

    elif importance > 0.05:

        print(f"- {feature.title()} has moderate impact ({importance:.2f}). Improving this can help but is less critical.")

    else:

        print(f"- {feature.title()} has minor impact ({importance:.2f}). Consider this for additional support if needed.")

**Sample Output:**

**6. Conclusion**

The project successfully predicts student performance with **high accuracy (99.5%)**. It identifies **key features impacting performance** and provides actionable AI-based suggestions. This can guide students and educators to enhance learning outcomes.