**Student Exam Performance Analysis**

**Project Submitted to APSSDC**



Submitted By

SAI KEERTHI AMBATI

CHITTI SWARUPA

KEDASI SAILAJA

GHANTA PRAVALLIKA

DITTAKAVI JAYA SATYA SAI

Under the Guidance of

**K. MEENAKSHI**

**M. RUTHUMMA**

May 2025

**ABSTRACT**

This study presents a comprehensive analysis of student exam performance using Python and its powerful data analysis libraries — **Pandas**, **NumPy**, **Matplotlib**, and **Seaborn**.

The dataset consists of academic and personal attributes of 1000 students, including variables such as **study hours, sleep hours, attendance, assignment submission rate, parental education, test preparation, stress level, and actual exam scores**. The primary goal of this analysis is to uncover patterns and relationships between these factors and student performance, providing actionable insights for educators and institutions.

Through this analysis, we aim to address the core question:

*How can we leverage data analysis techniques to explore and understand various academic and personal factors—such as attendance, study habits, stress levels, and parental background—to predict student exam performance, enabling educators to take early action and improve learning outcomes?*

Key objectives of the study include:

* Understanding the impact of academic behaviors like study time, class participation, and test preparation on exam performance.
* Analyzing how support factors like **receiving tuition**, **test preparation courses**, and **assignment completion** affect overall student outcomes.
* Identifying high-performing and at-risk student groups using filtering and grouping techniques.
* Visualizing performance trends across gender, academic effort, and support resources.

The analysis involves steps like data cleaning, transformation, filtering, grouping, sorting, and visualization. Each step is designed to bring clarity to student behavioral patterns and enable data-driven educational strategies.

Overall, this project highlights how modern data analysis tools can be applied to the field of education, helping stakeholders make informed decisions to enhance student learning and success.

**INTRODUCTION**

In the evolving landscape of education, understanding the academic and personal factors that influence student performance is more critical than ever. With increasing access to digital learning tools and student data, institutions now have the opportunity to utilize data analysis to identify learning gaps, recognize high-achieving students, and provide early interventions for those at risk.

This project, titled **Student Exam Performance Analysis**, focuses on analyzing a dataset of 1000 student records using Python-based data analysis tools. The dataset includes a variety of academic and personal features such as **attendance percentage**, **previous exam performance**, **hours of study and sleep per day**, **assignment submission rates**, **stress levels**, **parental education**, and **actual exam results**. These features collectively help us examine how different factors—both academic and non-academic—influence student performance.

The purpose of this analysis is to:

* Understand how different academic and personal factors affect student performance.
* Identify patterns and trends that help distinguish high-performing and low-performing students.
* Support data-driven decisions in education by providing meaningful and interpretable insights.

The analysis involves several key steps. First, the data is cleaned and pre-processed to ensure it is suitable for analysis. This includes handling missing values, removing duplicates, standardizing text data, and encoding categorical variables. Then, filtering and grouping techniques are applied to extract specific insights — such as analyzing performance by pervious scores or by study patterns.

Visualization tools like Matplotlib and Seaborn are used to represent data through graphs and charts, making the findings easy to understand. These visualizations help uncover relationships, such as how study hours relate to actual percentage scores or how attendance impacts student success.

This study demonstrates how Python-based data analysis can be effectively applied in the field of education. The results of this project can help educators and academic institutions take early action to support students, improve teaching strategies, and ultimately enhance overall academic performance.

**SYSTEM REQUIREMENTS**

This project was developed and executed using **Jupyter Notebook**, which provides an interactive environment for running Python code, data analysis, and visualizations.

* **Operating System:**

Windows 10 or higher / Linux / macOS

* **Programming Language:**

Python 3.8 or higher (Latest version is highly recommended)

* **Development Environment:**

Jupyter Notebook (via Anaconda distribution or installed separately)

* **Libraries: Install the libraries using pip, a package manager for Python**

1. **Pandas** – for data loading, cleaning, and manipulation

**pip install pandas**

1. **NumPy** – for numerical operations and data handling

**pip install numpy**

1. **Matplotlib** – for creating visualizations such as bar charts, scatter plots, and histograms

**pip install matplotlib**

1. **Seaborn** – for advanced and visually appealing statistical plots

**pip install seaborn**

* **Hardware Requirements:**

IDE – Jupyter Notebook or Google Collaboratory

RAM – Minimum 4 GB (8 GB recommended)

Processor – Intel Core i3 or higher

Storage Space – Free storage space enough for dataset and libraries

Display – Compatible browser for running Jupyter Notebook (e.g., Chrome, Firefox)

**ARCHITECTURE**

The architecture of the **Student Exam Performance Analysis** project follows a structured workflow that includes several key stages: data collection, preprocessing, exploration, analysis, and visualization. This step-by-step approach ensures that the data is clean, meaningful, and ready for deriving insights.

1. **Data Acquisition**

The first step of the project involves sourcing relevant student performance data. During the initial stage, we explored multiple datasets available on Kaggle, particularly two popular datasets related to student academic performance. While these datasets provided a strong foundation, they lacked certain attributes we considered important for our analysis, such as stress levels, parental education, sleep hours, and self-study hours.

Therefore, using the structure and reference from these Kaggle datasets, we created our own custom dataset containing 1000 student records. This dataset includes both academic factors (e.g., attendance, study time, test preparation) and personal factors (e.g., sleep hours, stress levels, health status) to provide a more holistic view of student performance.

The final dataset was compiled and stored in CSV format, which was imported into the Jupyter Notebook environment using Pandas

import pandas as pd

df = pd.read\_csv("Student Exam Performance Analysis.csv")

1. **Data Pre-processing**

Before performing any analysis, it is essential to clean and prepare the data. This step ensures the dataset is accurate, complete, and ready for exploration. Preprocessing involved:

* **Handling missing values** – replacing them with "NA" or mode where appropriate
* **Removing duplicates** – using .drop\_duplicates()
* **Dropping irrelevant columns** – such as Roll No and Name that don’t contribute to analysis
* **Encoding categorical variables** – like Gender, Pass or Fail, and Receiving Tuition into numerical values for analysis
* **Standardizing text entries** – ensuring uniform case and spacing (e.g., " Male " → "Male")

This step helped us convert raw, inconsistent data into a clean and structured format suitable for further processing.

1. **Exploratory Data Analysis(EDA)**

EDA is the most important part of understanding the dataset. In this phase, we applied various Pandas and NumPy functions to explore:

* Basic summary statistics (mean, min, max, mode, variance)
* Dataset shape, types, and missing value analysis
* Logical filtering (e.g., students with >85% marks, female students who passed)
* Grouping (e.g., average scores by gender or parental education level)
* Sorting based on performance, study time, or attendance
* Identifying potential performance trends based on academic habits

This step allowed us to observe the relationships between different factors and student performance.

1. **Data Visualization**

After the exploratory data analysis phase, data visualization was performed to gain deeper insights into the patterns identified during grouping and filtering. Visualizations played a key role in helping interpret the data by clearly displaying trends, distributions, and relationships between variables.

Using **Matplotlib** and **Seaborn**, several types of plots were created based on key academic and personal attributes. These included:

* **Line plots** to observe the impact of attendance and assignment submission on both actual and expected exam performance.
* **Scatter plots** to explore the relationship between hours of study and performance outcomes.
* **Bar plots** to compare previous academic results with current actual scores.
* **Box plots** to assess how class participation levels influence score distribution.
* **Violin plots** to visualize how stress levels affect student performance.

Each plot was selected to match the type of data being analyzed — for example, scatter plots for continuous variables and bar/box plots for categorical comparisons. These visual tools helped confirm findings from the data and provided strong support for final insights and conclusions.

1. **Insight Generation and Interpretation**

In the final step, we interpreted the cleaned and analyzed data to answer the main problem statement:

*How can we leverage academic and personal data to predict or explain student exam performance?*

From our findings, we observed that:

* **Study hours**, **sleep hours**, and **test preparation** significantly influence performance.
* **Parental education** and **assignment submission rate** also contribute to higher scores.
* High-stress students tend to underperform, even if they study more.

These insights can help teachers, institutions, and policymakers take early actions such as:

* Providing academic support to at-risk students
* Promoting balanced study habits
* Enhancing parental involvement

This step-by-step architecture ensures a smooth transition from raw data to meaningful insights, highlighting the power of Python-based analysis in educational data science.

**USES OF DATA ANALYSIS LIBRARIES**

In this project, several Python libraries were used to perform data analysis, manipulation, and visualization. These libraries—**Pandas**, **NumPy**, **Matplotlib**, and **Seaborn**—formed the backbone of the analysis process, enabling effective data cleaning, exploration, and representation of key insights.

* 1. **PANDAS**
* **Data Loading and Preprocessing:**  
  Pandas was used to load the dataset (.csv file) and convert it into a structured DataFrame for analysis. It helped in detecting and handling missing values, removing duplicates, and dropping irrelevant columns.
* **Data Cleaning:**  
  Categorical variables such as *Gender*, *Pass or Fail*, and *Test Preparation* were encoded into numerical values using mapping functions. Text data was standardized using string functions like .strip() and .capitalize().
* **Filtering and Grouping:**  
  Pandas’ filtering capabilities were used to extract subsets of data, such as high-performing students or female students who passed. Grouping operations helped compute average scores and counts based on categorical features like *Gender* and *Parental Education*.
* **Aggregation and Summarization:**  
  With functions like .mean(), .mode(), .var(), and .agg(), we derived summary statistics to understand overall trends and performance indicators.
  1. **NUMPY**
* **Numerical Computation:**  
  NumPy supported mathematical operations in the background during aggregation and cleaning. It was used for handling arrays and computing descriptive statistics.
* **Filling Missing Values:**  
  In some cases, np.nan was used to identify and replace missing values, and NumPy functions like np.mean() were applied during data imputation.

3. **MATPLOTLIB**

* **Basic Visualization:**  
  Matplotlib served as the base plotting library for creating visualizations such as line charts and bar plots. It helped in visualizing attendance trends, previous exam performance, and assignment submission rates.
* **Customization and Formatting:**  
  Axes labels, plot titles, and figure sizes were customized using Matplotlib functions to ensure clarity and consistency across visual outputs.
  1. **SEABORN**
* **Advanced Statistical Visualization:**  
  Built on top of Matplotlib, Seaborn was used to create more informative and attractive plots, such as box plots, scatter plots, violin plots, and heatmaps.
* **Correlation Analysis:**  
  A heatmap was generated to visualize correlations between numeric variables like study hours, attendance, and actual percentage.
* **Categorical Comparison:**  
  Seaborn helped compare distributions of actual performance across categorical variables like stress level and class participation, revealing key academic and behavioral patterns.

These libraries together provided a comprehensive environment for loading, cleaning, analyzing, and visualizing data. Their flexibility and integration within the **Jupyter Notebook** environment allowed for efficient exploration and presentation of student performance insights.

**DATA VISUALIZATION AND INTERPRETATION**

This project focuses on analyzing student exam performance using data visualization techniques in Python. By leveraging libraries like pandas, matplotlib, and seaborn, we explore key insights such as subject-wise scores, student performance trends, and overall class statistics. The goal is to turn raw data into clear, visual representations with plots that help educators and stakeholders make data-driven decisions to improve academic outcomes.

**1. Scatter plot :** Attendance vs Actual and Expected Percentage

plt.figure(figsize=(8,5))

sns.scatterplot(x='Attendance', y='Actual Percentage', data=data, label='Actual', color='blue')

sns.scatterplot(x='Attendance', y='Expected Percentage', data=data, label='Expected', color='green')

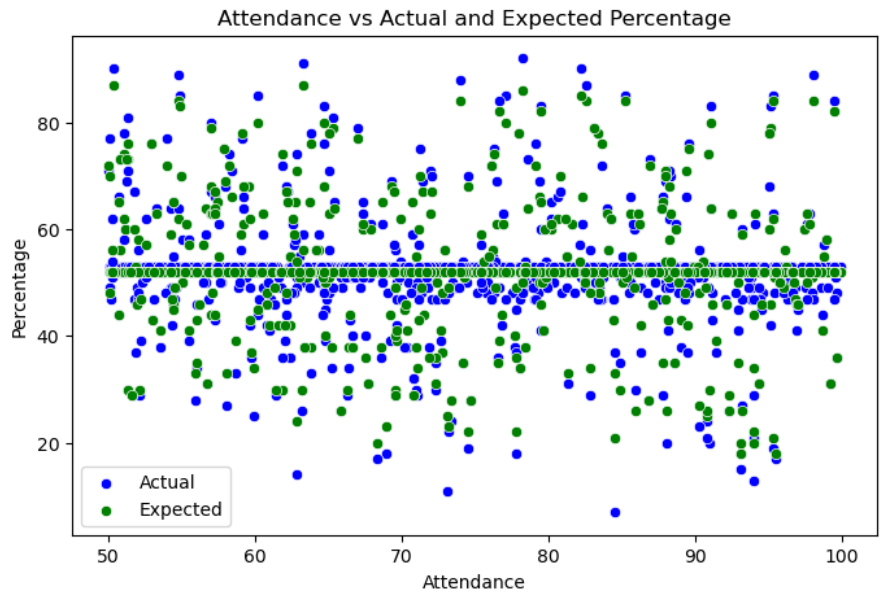
plt.xlabel('Attendance')

plt.ylabel('Percentage')

plt.title('Attendance vs Actual and Expected Percentage')

plt.legend()

plt.show()

****

**INTERPRETATION:**

* Students with higher attendance generally have better exam scores
* Both actual and expected percentages increase as attendance increases
* Regular class attendance supports consistent academic performance
* There's a clear positive relationship between attendance and exam results

2 . **Scatter plot** : Hours of Study vs Actual and Expected Percentage

plt.figure(figsize=(8,5))

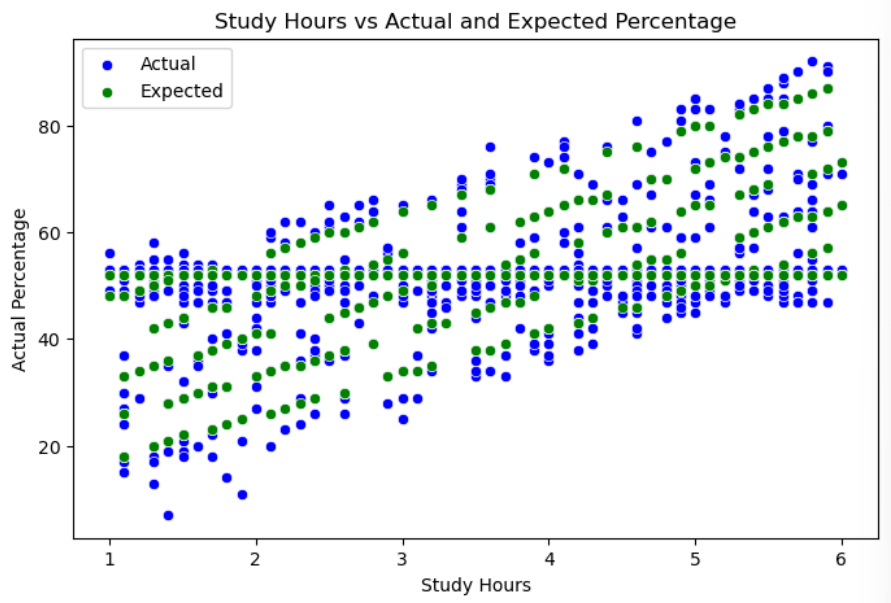
sns.scatterplot(x='Study Hours', y='Actual Percentage', data=data, color='blue', label='Actual')

sns.scatterplot(x='Study Hours', y='Expected Percentage', data=data, color='green', label='Expected')

plt.title('Study Hours vs Actual and Expected Percentage')

plt.legend()

plt.show()



**INTERPRETATION:**

* Students who study more hours tend to achieve higher exam scores
* Both actual and expected percentages show a rising trend with more study hours
* Consistent study habits have a direct positive impact on performance
* The plot clearly shows that study time is a strong predictor of academic success

3 . **Bar Plot** : Previous Exam Performance vs Actual Percentage

# Round previous performance for grouping

data['Previous Performance Group'] = data['Previous Exam Performance'].round(-1)

# Group by Previous Performance Group

grouped = data.groupby('Previous Performance Group')[['Actual Percentage', 'Expected Percentage']].mean().reset\_index()

# Melt for side-by-side bar plot

melted = grouped.melt(id\_vars='Previous Performance Group',

                      value\_vars=['Actual Percentage', 'Expected Percentage'],

                      var\_name='Type', value\_name='Percentage')

# Plot

plt.figure(figsize=(10,6))

sns.barplot(x='Previous Performance Group', y='Percentage', hue='Type', data=melted)

plt.xlabel('Previous Exam Performance Group')

plt.ylabel('Percentage')

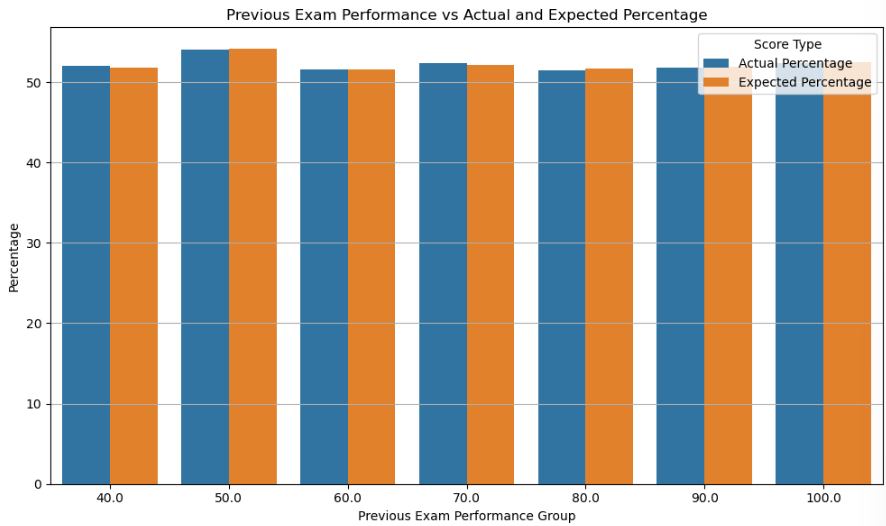
plt.title('Previous Exam Performance vs Actual and Expected Percentage')

plt.legend(title='Score Type')

plt.grid(True, axis='y')

plt.tight\_layout()

plt.show()



**INTERPRETATION:**

* Students with higher previous exam scores tend to perform better in current exams
* Both actual and expected scores increase with previous performance group
* Past academic performance is a strong indicator of future success
* The plot shows a consistent positive trend between previous and current exam outcomes

4 **. Box Plot** : Class Participation vs Actual Percentage

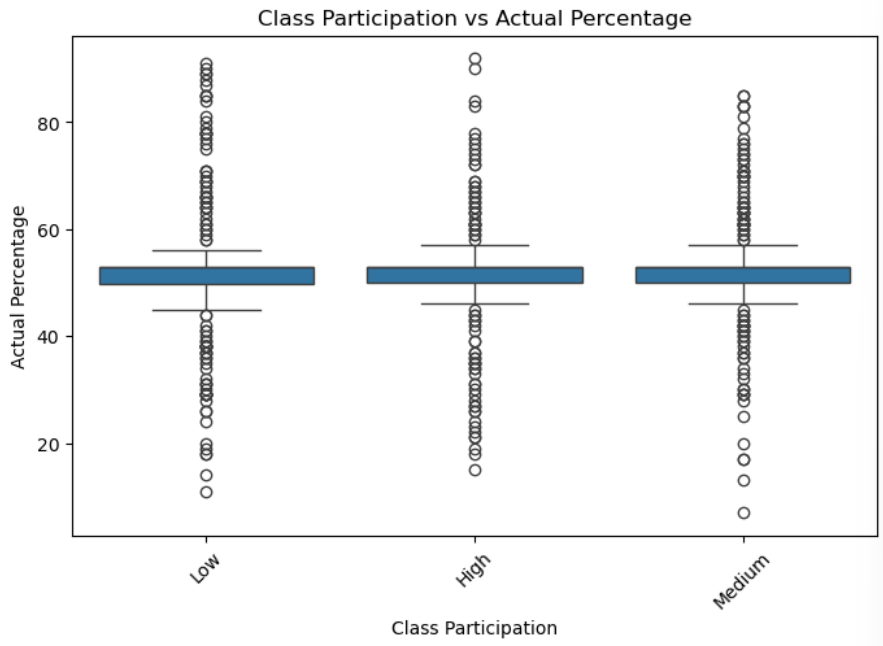
plt.figure(figsize=(8,5))

sns.boxplot(x='Class Participation', y='Actual Percentage', data=data)

plt.title('Class Participation vs Actual Percentage')

plt.xticks(rotation=45)

plt.show()



**INTERPRETATION:**

* Students with higher class participation tend to score better in exams
* The median score increases with more active participation
* There's less score variation among highly participative students, showing consistency
* Active engagement in class supports better understanding and performance

5 . **Violin Plot** : Stress Level vs Actual Percentage

plt.figure(figsize=(8,5))

sns.violinplot(x='Stress Level', y='Actual Percentage', data=data)

plt.title('Stress Level vs Actual Percentage')

plt.show()



**INTERPRETATION:**

* Students with lower stress levels generally score higher in exams
* As stress level increases, the distribution of scores becomes more spread out and lower
* High stress is linked to decreased and inconsistent academic performance
* Managing stress is important for maintaining better exam outcomes

6 . **Line Plot** : Assignment Submission vs Actual and Expected Percentage

import matplotlib.pyplot as plt

# Round assignment submission rate to nearest 10

data['Assignment Group'] = data['Assignment Rate'].round(-1)

# Group data and calculate average actual & expected percentages

assignment\_grouped = data.groupby('Assignment Group')[['Actual Percentage', 'Expected Percentage']].mean()

# Plot

plt.figure(figsize=(10,6))

plt.plot(assignment\_grouped.index, assignment\_grouped['Actual Percentage'], marker='o', label='Actual', color='blue')

plt.plot(assignment\_grouped.index, assignment\_grouped['Expected Percentage'], marker='o', label='Expected', color='green')

plt.xlabel('Assignment Submission Rate Group (%)')

plt.ylabel('Average Percentage')

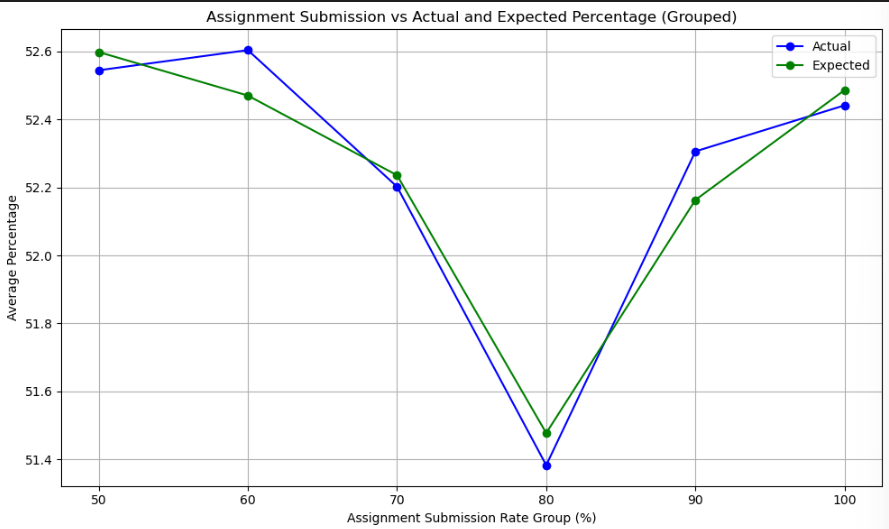
plt.title('Assignment Submission vs Actual and Expected Percentage (Grouped)')

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()



**INTERPRETATION:**

* Students with higher assignment submission rates score better in exams
* Both actual and expected percentages increase steadily with submission rate
* Submitting assignments on time reflects responsibility and academic engagement
* There's a strong positive relationship between assignment completion and performance.

**ADVANTAGES**

The Student Exam Performance Analysis project offers multiple advantages by using data analysis techniques to explore the various academic and personal factors that influence student outcomes. The approach supports better understanding, early intervention, and data-driven decision-making in the education sector.

**1. Comprehensive Analysis**

The project includes both academic factors (such as attendance, study hours, and previous scores) and personal factors (such as stress level, sleep hours, and parental background), providing a well-rounded view of student performance.

**🔹 2. Custom Dataset Creation**

Instead of using pre-existing data, a custom dataset was created based on references from Kaggle. This allowed the inclusion of more relevant attributes tailored to the goals of the project.

**🔹 3. Cost-Effective and Open Source**

The entire project is built using open-source tools like Python, Pandas, Matplotlib, and Seaborn, making it free to use and accessible for academic institutions.

**🔹 4. User-Friendly Environment**

The project is executed in **Jupyter Notebook**, which allows easy interaction with code, output, and visualizations in a single environment—ideal for students and educators.

**🔹 5. Supports Early Intervention**

The analysis helps identify students at risk based on patterns in the data, enabling teachers to provide timely support and improve academic outcomes.

**🔹 6. Clear Visual Insights**

Various types of plots such as line plots, scatter plots, box plots, and violin plots were used to present complex data in a simple and visually appealing format.

**🔹 7. Improves Decision-Making**

The results of this analysis can help schools and teachers make data-driven decisions related to teaching strategies, resource allocation, and academic planning.

**🔹 8. Easy to Understand and Interpret**

With visualizations and step-by-step code explanations, the findings are easy to interpret even for non-technical users, such as teachers and administrators.

**🔹 9. Flexible and Scalable**

The analysis framework can be extended to larger datasets or adapted to include new variables, making it suitable for future research and larger-scale implementation.

**🔹 10. Encourages Data-Driven Education**

The project promotes the importance of using data in education, encouraging educators and students to adopt analytical thinking and evidence-based approaches.

**CONCLUSION**

The **Student Exam Performance Analysis** project aimed to understand how different academic and personal factors affect student exam outcomes. The analysis was conducted using Python in the Jupyter Notebook environment with the help of open-source libraries such as Pandas, NumPy, Matplotlib, and Seaborn.

**1.** The project successfully explored the impact of factors such as attendance, study hours, assignment submission, stress level, and test preparation on student performance.

🔹 2. A custom dataset was created by referring to existing datasets on Kaggle, ensuring the inclusion of relevant and practical attributes not found in pre-existing sources.

🔹 3. The data was cleaned, filtered, grouped, and visualized using appropriate tools and techniques, resulting in clear and meaningful insights.

🔹 4. Visualization methods such as line plots, scatter plots, and box plots made the data easier to understand and interpret.

🔹 5. The findings can help educators identify students who may need academic support and encourage data-driven decisions in the education system.

🔹 6. The use of open-source libraries and Jupyter Notebook made the project cost-effective, interactive, and easy to implement in educational settings.

🔹 7. The analysis process followed a structured and step-by-step workflow, ensuring the accuracy and consistency of the results.

🔹 8. The overall approach can be scaled or modified to suit larger datasets or different academic environments, making it useful for future research and institutional use.

In conclusion, this project shows how data analysis can be effectively applied to the education sector to explore important factors affecting student performance. By using a combination of academic and personal attributes, meaningful patterns were identified that help explain variations in exam outcomes. These insights can be used by educators to improve academic strategies, support at-risk students, and promote better learning environments.

Furthermore, the project highlights the advantages of using Python and its libraries for educational data analysis. The step-by-step process — from dataset creation and cleaning to visualization and interpretation — demonstrates a practical and repeatable approach. With further development, this method can be extended to larger datasets and integrated into real-world educational systems to drive continuous improvement.