Six Weeks Industrial Training Project Report on

Student Performance Analysis

Submitted in the partial fulfilment of the requirement for the award of degree of

Bachelor of Technology

In

Computer Science and Artificial Intelligence

Batch

(2023-2027)

Submitted to: **Submitted by: Gurleen Kaur**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DAV UNIVERSITY
JALANDHAR- PUNJAB 144012

ACKNOWEDGEMENT

I express my gratitude to all those who helped us in various stages of the development of this project. First, I would like to express my sincere gratitude indebtedness to **DAV University** for allowing me to undergo the summer training of 45 days at **Excellence Technologies** Iam also thankful to all faculty members of Department of Computer Science and Engineering, for their true help, inspiration and for helping me for the preparation of the final report and presentation. Last but not least, I pay my sincere thanks and gratitude to all the Staff Members of **Excellence Technologies** for their support and for making our training valuable and fruitful.

DECLARATION

I,Gurleen , hereby declare that the work which is being presented in this project/training titled Student Performance Analysis by me, in partial fulfilment of the requirements for the award of Bachelor of Technology (B.Tech) Degree in"Computer Science & Engineering and Artificial Intelligence" is an authentic record of my own work carried out under the guidance of Ms. Yashvi Bansal (AI trainer)

To the best of my knowledge, the matter embodied in this report has not been submitted to any other University/ Institute for the award of any degree or diploma.

Gurleen Kaur
12300290

Student Name and Signature

Table of Contents

Introduction

This report outlines the work undertaken during the 45-day summer training program as part of the Bachelor of Technology (B.Tech) curriculum in Computer Science and Artificial Intelligence at DAV University, Jalandhar, under the academic session 2023–2027.

In recent years, data-driven decision-making has become a critical part of various industries, including education. This training project was conceptualized to provide practical exposure to the field of data analysis and machine learning through the lens of student academic performance.

The training focused on a complete end-to-end cycle of a data science project, from raw dataset acquisition and exploration to cleaning, transformation, modeling, and evaluation. The project provided the opportunity to work with real-world data and extract meaningful insights that could potentially inform educators and institutions about the performance patterns of students.

The project titled "Student Performance Analysis" is a comprehensive study of student test scores with respect to various demographic and educational attributes such as gender, parental education, lunch type, and preparation course. Through this project, I gained a thorough understanding of how data analytics and machine learning can be applied in educational settings to assess, predict, and improve student outcomes.

The implementation was done using Python programming in Google Colab, leveraging libraries like Pandas, NumPy, Matplotlib, Seaborn, and Scikit-learn to perform data manipulation, visualization, and machine learning modeling.

This report details every stage of the project, presenting a holistic view of the training process, challenges faced, solutions implemented, and knowledge acquired during the training period.

Student Performance Analysis

An Exploratory and Predictive Study Using Data Science and Machine Learning Techniques

The project aims to analyze student academic data and identify patterns that impact learning outcomes. By performing both exploratory data analysis and predictive modeling, the goal is to understand and forecast student performance using a combination of demographic and educational features.

Objectives

The primary objectives of this training project are outlined as follows:

1. **To understand and apply the core principles of data science and machine learning** by working on a structured dataset from the education domain.

2. **To perform exploratory data analysis (EDA)** for identifying relationships between various features such as gender, parental education level, lunch type, and test preparation course with student scores in mathematics, reading, and writing.

3. **To engineer new features** that can help in better understanding and prediction of student outcomes, such as calculating the average score and defining a pass/fail category.

4. **To apply suitable regression and classification algorithms** to:

    - Predict the average score of a student (Regression)

    - Classify whether a student will pass or fail (Classification)

5. **To evaluate and compare the performance of machine learning models** using appropriate metrics like R^2 score, MAE, accuracy, confusion matrix, and classification reports.

6. **To visualize and communicate the findings effectively** using colorful, clear plots and charts created with Matplotlib and Seaborn.

7. **To gain hands-on experience** using Python, Google Colab, and machine learning libraries in solving a real-world educational data problem.

Steps to Achieve Objectives

To accomplish the objectives of this project, a structured and phased approach was followed. Each phase played a critical role in moving the project from raw data to actionable insights and model predictions. Below are the detailed steps undertaken:

Step 1: Dataset Collection

The first and foundational step of this project was the selection and acquisition of a suitable dataset. For this purpose, the dataset titled "StudentsPerformance.csv" was chosen. It is a publicly available dataset widely used for educational analytics and contains a comprehensive set of student performance records. The dataset includes:

- **1000 student entries**
- **Categorical features** such as:
    - Gender
    - Race/Ethnicity
    - Parental Level of Education
    - Lunch Type
    - Test Preparation Course Completion

- **Numerical features**:
    - Math Score
    - Reading Score
    - Writing Score

This dataset is ideal for understanding how various demographic and preparatory factors influence student performance in standardized assessments. The data was loaded into the working environment using Python's Pandas library in Google Colab, which provided an easy and interactive platform for coding, running experiments, and documenting results.

Step 2: Data Understanding and Cleaning

After loading the dataset, a thorough exploration was performed to understand the structure, data types, and content of each column. Key activities included:

- Displaying the first few rows (`df.head()`) to inspect data samples.

- Checking column names, data types, and dimensions using `df.info()`.

- Ensuring there were no missing or null values in the dataset.
- Renaming columns with spaces (e.g., “parental level of education”) to underscore-separated names like `parental_level_of_education` for code readability.
- Checking for duplicate entries and removing them if any.
- Verifying the integrity of numerical data like score values (ranged between 0 and 100).

This step ensured that the dataset was consistent, structured, and ready for deeper analysis.

Step 3: Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) was a critical phase aimed at understanding the relationships and patterns within the data. The following analyses and visualizations were performed:

- **Univariate analysis** to explore the distribution of individual variables such as math score, reading score, and writing score using histograms and KDE plots.
- **Categorical analysis** to compare performance based on gender, lunch type, parental education, and test preparation course using bar plots and count plots.
- **Multivariate analysis** using pair plots and box plots to examine how multiple factors influence student performance.
- **Correlation analysis** using heatmaps to identify which variables were strongly or weakly correlated.

The insights drawn from EDA were vital in determining which features should be emphasized during modeling.

Step 4: Feature Engineering

This step involved the creation of new features that could enhance the predictive capabilities of the models. Key features engineered included:

- **Average Score**: A new column `average_score` was added by computing the mean of the three subject scores.

- **Pass/Fail Classification**: A binary column `pass_status` was created to classify students into "Pass" (if average score ≥ 40) or "Fail" (otherwise).

- **Label Encoding**: All categorical features (e.g., gender, lunch type) were converted into numerical values using Label Encoding, which is essential for feeding data into machine learning models.

These transformations improved model interpretability and allowed algorithms to process categorical data effectively.

Step 5: Model Building

Two types of machine learning models were built based on the type of output variable:

1. **Regression Models**:
    - **Linear Regression** was applied to predict the average score based on demographic and academic attributes.
    - **Random Forest Regressor** was used as an ensemble method to enhance predictive performance.
2. **Classification Models**:
    - **Logistic Regression** was used to classify whether a student would pass or fail.
    - **Random Forest Classifier** was implemented for better accuracy and to interpret feature importance.

All models were trained and tested using an 80-20 data split. Model training was done using **Scikit-learn**, and hyperparameters were left at default values for baseline evaluation.

Step 6: Model Evaluation

To evaluate model performance, appropriate statistical metrics were used:

- For **regression models**:
    - **R^2 Score** to assess variance explanation.

    - **Mean Absolute Error (MAE)** to measure prediction accuracy.

- For **classification models**:
    - **Accuracy**, **Precision**, **Recall**, and **F1-score** to measure classification quality.
    - **Confusion Matrix** to visualize true positives, true negatives, false positives, and false negatives.

Graphical evaluations such as bar plots and heatmaps were also used to interpret the results more intuitively.

Step 7: Documentation and Reporting

Throughout the training project, every step of the process was documented using Markdown cells in Google Colab. Screenshots of code execution, graphs, and results were captured and included in this report.

The goal of this step was to create a self-explanatory, reproducible workflow that reflects:

- The journey from raw data to insight and prediction.
- The challenges and thought processes involved.
- The final conclusions supported by visual evidence.

This final report stands as a complete and transparent record of the project work done over the course of the summer training.

Coding and Implementation

The coding phase of the project was executed entirely in Google Colab, which provided an efficient and collaborative environment for data analysis, visualization, and model building using Python. The entire implementation was structured into modular steps using code cells and Markdown commentary.

Below is a breakdown of the key implementation steps,

Student Performance Analysis

Step 1.1: Load and Preview the Dataset

```
import pandas as pd

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

# Show the first few rows
df.head()
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

Next steps: Generate code with df | View recommended plots | New interactive sheet

Step 1.2: Check Shape and Basic Info

```
[2] # Dataset shape
    print("Dataset shape:", df.shape)

    # Dataset info
    df.info()
```

```
Dataset shape: (1000, 8)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
```

```
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   1000 non-null   int64
 6   reading score                1000 non-null   int64
 7   writing score                1000 non-null   int64
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

Step 1.3: Basic Statistics of Numerical Columns

```
[3] # Summary statistics
    df.describe()
```

	math score	reading score	writing score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000
max	100.00000	100.000000	100.000000

Step 1.4: Check for Missing Values

```
[4] # Missing values
    df.isnull().sum()
```

	0
gender	0
race/ethnicity	0
parental level of education	0
lunch	0
test preparation course	0
math score	0
reading score	0
writing score	0

dtype: int64

Step 2: Data Preprocessing

```
# Check for missing values
print(df.isnull().sum())
```

```
gender                       0
race/ethnicity               0
parental level of education  0
lunch                        0
test preparation course      0
math score                   0
reading score                0
writing score                0
dtype: int64
```

```
[6] # Rename columns for simplicity
df.columns = df.columns.str.replace(' ', '_')
```

```
[7] # Create new feature: average score
df['average_score'] = df[['math_score', 'reading_score', 'writing_score']].mean(axis=1)
```

```
[8] # Encode categorical variables
from sklearn.preprocessing import LabelEncoder

categorical_cols = ['gender', 'race/ethnicity', 'parental_level_of_education', 'lunch', 'test_preparation_course']
label_encoders = {}

for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

df.head()
```

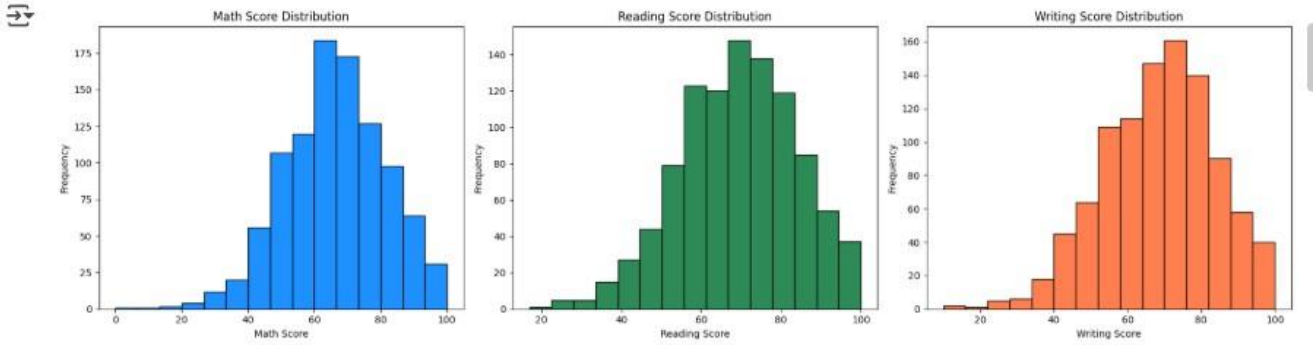
	gender	race/ethnicity	parental_level_of_education	lunch	test_preparation_course	math_score	reading_score	writing_score	average_score
0	0	1	1	1	1	72	72	74	72.666667
1	0	2	4	1	0	69	90	88	82.333333
2	0	1	3	1	1	90	95	93	92.666667
3	1	0	0	0	1	47	57	44	49.333333
4	1	2	4	1	1	76	78	75	76.333333

Step 3: Exploratory Data Analysis (EDA)

3.1 Histogram for Each Subject Score

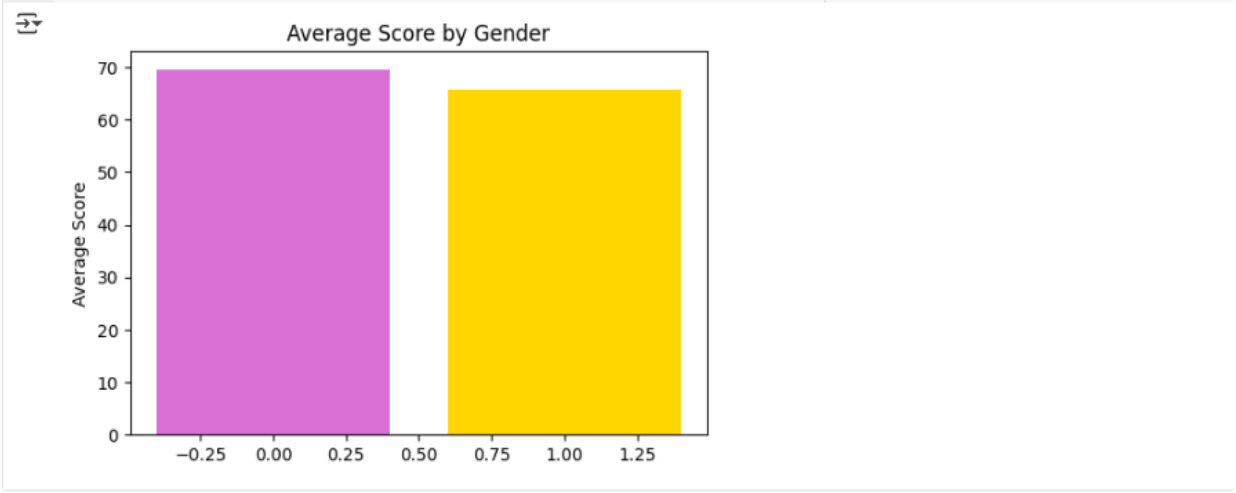
```
[9] import matplotlib.pyplot as plt

plt.figure(figsize=(18, 5))

plt.subplot(1, 3, 1)
plt.hist(df['math_score'], bins=15, color='dodgerblue', edgecolor='black')
plt.title('Math Score Distribution')
plt.xlabel('Math Score')
plt.ylabel('Frequency')

plt.subplot(1, 3, 2)
plt.hist(df['reading_score'], bins=15, color='seagreen', edgecolor='black')
plt.title('Reading Score Distribution')
plt.xlabel('Reading Score')
plt.ylabel('Frequency')
```

```
[9] plt.subplot(1, 3, 3)
    plt.hist(df['writing_score'], bins=15, color='coral', edgecolor='black')
    plt.title('Writing Score Distribution')
    plt.xlabel('Writing Score')
    plt.ylabel('Frequency')

    plt.tight_layout()
    plt.show()
```



3.2 Bar Chart: Average Score by Gender

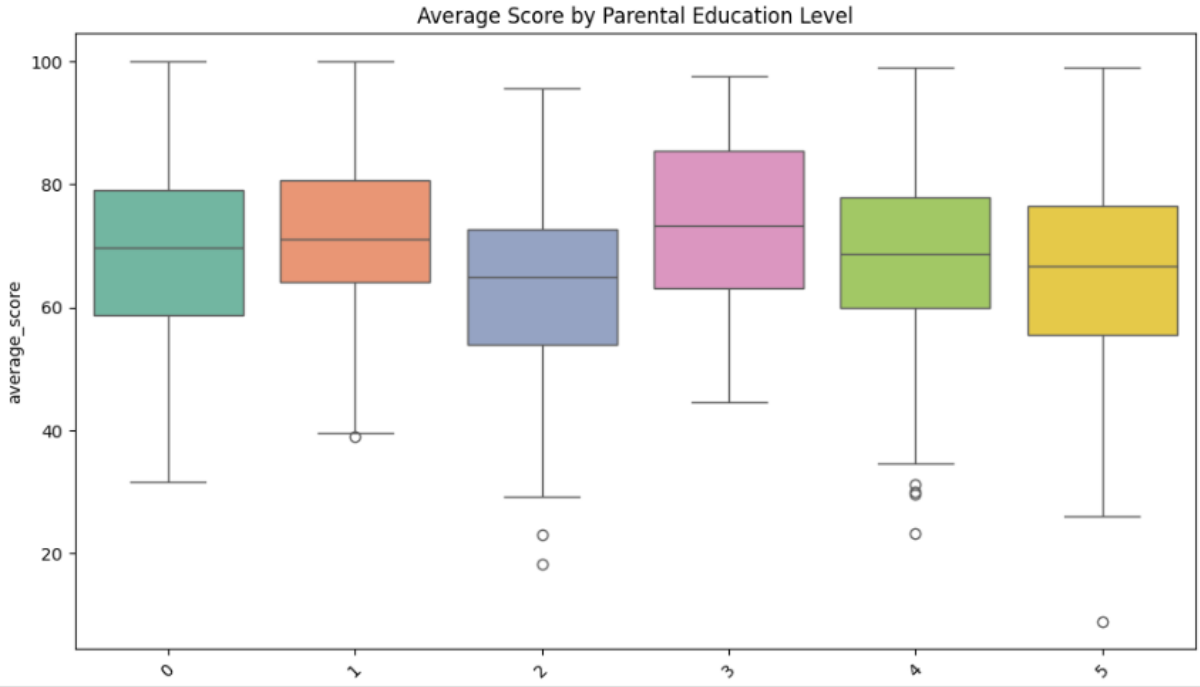
```
df['average_score'] = df[['math_score', 'reading_score', 'writing_score']].mean(axis=1)
avg_gender = df.groupby('gender')['average_score'].mean()

plt.figure(figsize=(6, 4))
colors = ['orchid', 'gold']
plt.bar(avg_gender.index, avg_gender.values, color=colors)
plt.title('Average Score by Gender')
plt.ylabel('Average Score')
plt.show()
```



3.3 Box Plot: Score by Parental Education

```
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.boxplot(x='parental_level_of_education', y='average_score', data=df, palette='Set2')
plt.title('Average Score by Parental Education Level')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

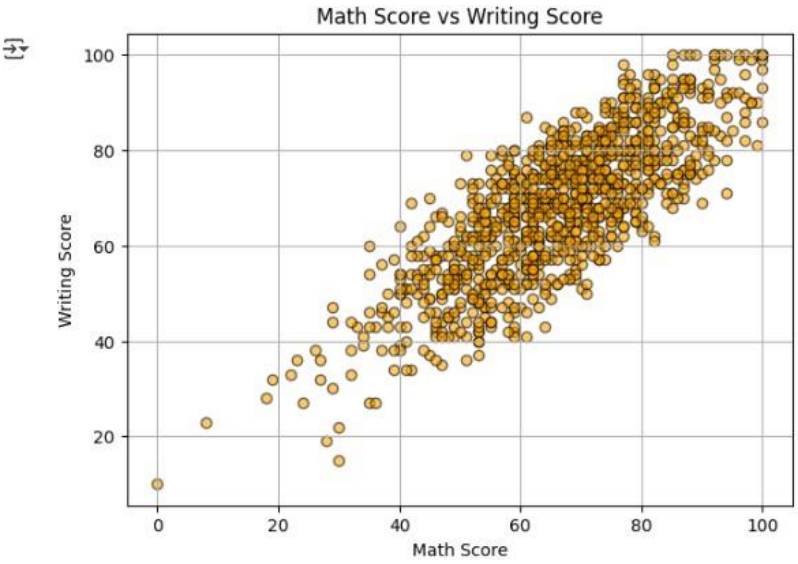
```
/tmp/ipython-input-11-2513593541.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` an

  sns.boxplot(x='parental_level_of_education', y='average_score', data=df, palette='Set2')
```



Variables Terminal Python 3

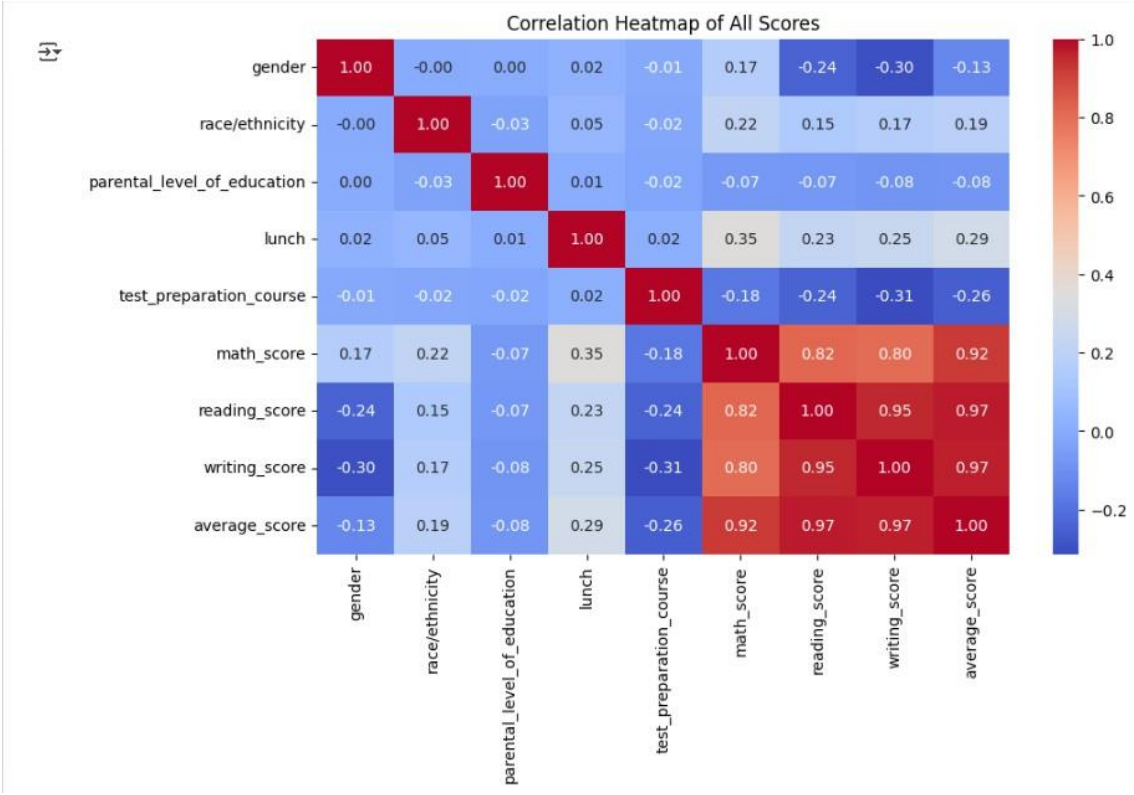
3.4 Scatter Plot: Math Score vs Writing Score

```
plt.figure(figsize=(7, 5))
plt.scatter(df['math_score'], df['writing_score'], alpha=0.6, color='orange', edgecolor='black')
plt.title('Math Score vs Writing Score')
plt.xlabel('Math Score')
plt.ylabel('Writing Score')
plt.grid(True)
plt.show()
```



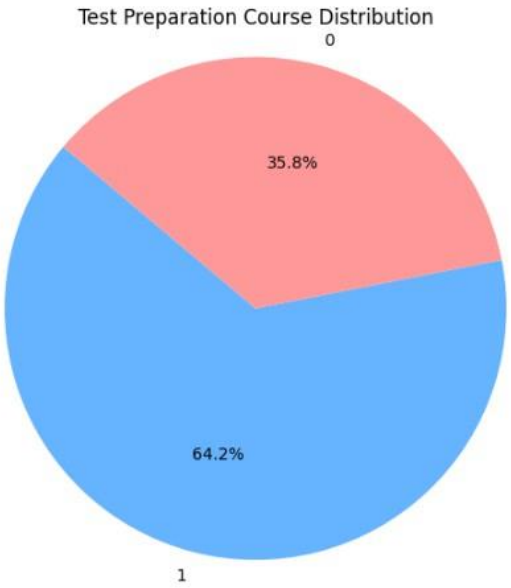
3.5 Correlation Heatmap

```
[13] plt.figure(figsize=(10, 6))
     sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm', fmt='.2f')
     plt.title('Correlation Heatmap of All Scores')
     plt.show()
```



3.6 Pie Chart – Test Preparation Course

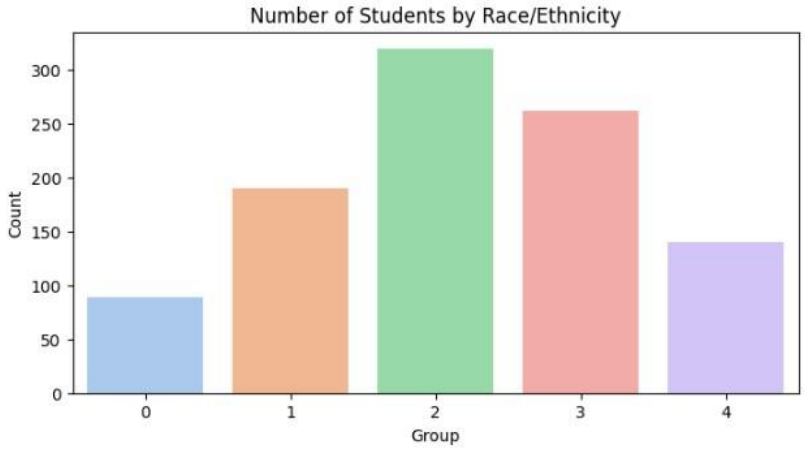
```
[14] prep_counts = df['test_preparation_course'].value_counts()

     plt.figure(figsize=(6, 6))
     plt.pie(prep_counts, labels=prep_counts.index, autopct='%1.1f%%', colors=['#66b3ff', '#ff9999'], startangle=140)
     plt.title('Test Preparation Course Distribution')
     plt.axis('equal')  # Equal aspect ratio ensures pie is a circle.
     plt.show()
```



3.7 Count Plot – Race/Ethnicity Categories

```
plt.figure(figsize=(8, 4))
sns.countplot(x='race/ethnicity', data=df, palette='pastel')
plt.title('Number of Students by Race/Ethnicity')
plt.xlabel('Group')
plt.ylabel('Count')
plt.show()
```

```
/tmp/ipython-input-15-1010017044.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` a

  sns.countplot(x='race/ethnicity', data=df, palette='pastel')
```



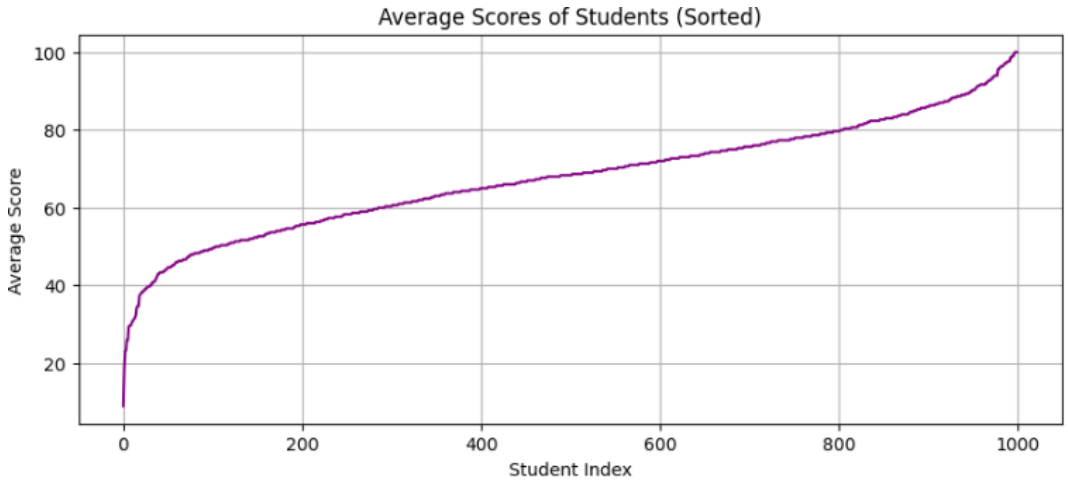
Violin Plot – Reading Score by Gender

```
[16] plt.figure(figsize=(6, 5))
     sns.violinplot(x='gender', y='reading_score', data=df, palette='Set3')
     plt.title('Reading Score Distribution by Gender')
     plt.xlabel('Gender')
     plt.ylabel('Reading Score')
     plt.show()
```

```
/tmp/ipython-input-16-1039520212.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` a

  sns.violinplot(x='gender', y='reading_score', data=df, palette='Set3')
```

Line Chart – Sorted Average Scores

```
sorted_scores = df.sort_values('average_score').reset_index()

plt.figure(figsize=(10, 4))
plt.plot(sorted_scores['average_score'], color='purple')
plt.title('Average Scores of Students (Sorted)')
plt.xlabel('Student Index')
plt.ylabel('Average Score')
plt.grid(True)
plt.show()
```


Average Scores of Students (Sorted)
Average Score
Student Index
100
80
60
40
20
0
200
400
600
800
1000

Step 4: Model Building & Evaluation

```
[18] from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LinearRegression
     from sklearn.ensemble import RandomForestRegressor
     from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
     import numpy as np
```

4.1 Prepare Features and Target

```
[19] # Define input features (excluding individual scores to avoid data leakage)
     features = df.drop(columns=['math_score', 'reading_score', 'writing_score', 'average_score'])
     target = df['average_score']

     # Split dataset into train and test (80% train, 20% test)
     X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)
```

4.2 Linear Regression

```
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)

# Evaluation
print(" Linear Regression:")
print("R² Score:", r2_score(y_test, y_pred_lr))
print("MAE:", mean_absolute_error(y_test, y_pred_lr))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_lr)))
```

```
 Linear Regression:
R² Score: 0.1255652122569746
MAE: 10.717163335369497
RMSE: 13.691222853433064
```

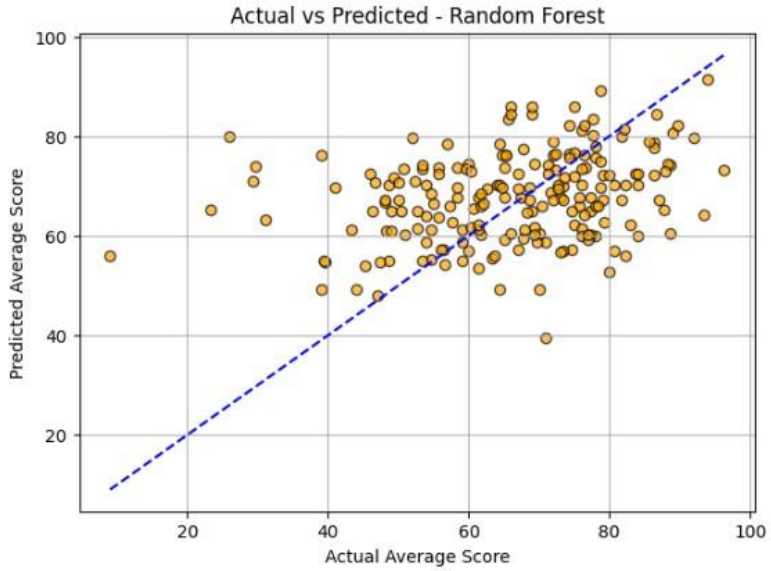
4.3 Random Forest Regressor

```
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)

# Evaluation
print(" Random Forest Regressor:")
print("R² Score:", r2_score(y_test, y_pred_rf))
print("MAE:", mean_absolute_error(y_test, y_pred_rf))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_rf)))
```

```
 Random Forest Regressor:
R² Score: -0.07531455903433115
MAE: 0.1863373034881169
RMSE: 0.35435837457471503
```

4.4 Visualize Predictions vs Actual

```
import matplotlib.pyplot as plt

plt.figure(figsize=(7, 5))
plt.scatter(y_test, y_pred_rf, color='orange', edgecolor='black', alpha=0.7)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='blue', linestyle='--')
plt.title('Actual vs Predicted - Random Forest')
plt.xlabel('Actual Average Score')
plt.ylabel('Predicted Average Score')
plt.grid(True)
plt.show()
```



Step 5: Classification Model – Predicting Pass or Fail

```
[23] # Create a binary column: 1 if average_score >= 50 else 0
     df['pass_fail'] = df['average_score'].apply(lambda x: 1 if x >= 50 else 0)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.model_selection import train_test_split

# Features and target
X = df.drop(columns=['math_score', 'reading_score', 'writing_score', 'average_score', 'pass_fail'])
y = df['pass_fail']

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

5.1 Logistic Regression

```
log_model = LogisticRegression()
log_model.fit(X_train, y_train)
y_pred_log = log_model.predict(X_test)

print(" Logistic Regression:")
print("Accuracy:", accuracy_score(y_test, y_pred_log))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_log))
print("Classification Report:\n", classification_report(y_test, y_pred_log))
```

```
 Logistic Regression:
Accuracy: 0.865
Confusion Matrix:
 [[  0  27]
 [  0 173]]
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        27
           1       0.86      1.00      0.93       173

    accuracy                           0.86       200
   macro avg       0.43      0.50      0.46       200
weighted avg       0.75      0.86      0.80       200
```

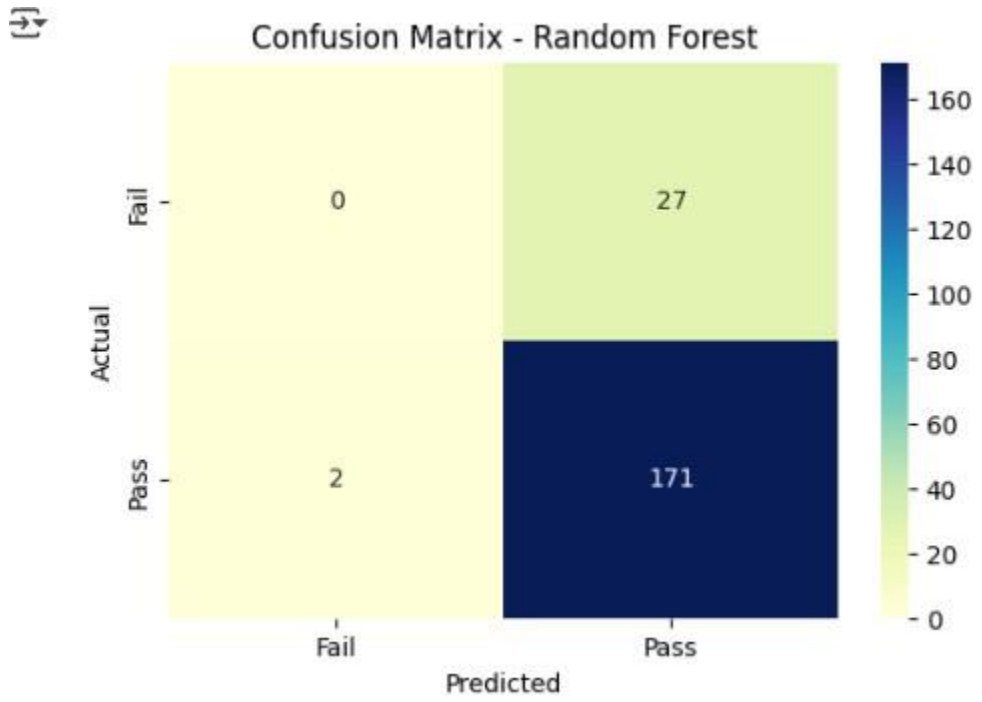
5.2 Random Forest Classifier

```
rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)
rf_clf.fit(X_train, y_train)
y_pred_rf = rf_clf.predict(X_test)

print(" Random Forest Classifier:")
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))
print("Classification Report:\n", classification_report(y_test, y_pred_rf))
```

```
 Random Forest Classifier:
Accuracy: 0.855
Confusion Matrix:
 [[  0  27]
 [  2 171]]
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        27
           1       0.86      0.99      0.92       173

    accuracy                           0.85       200
   macro avg       0.43      0.49      0.46       200
weighted avg       0.75      0.85      0.80       200
```

Plot Confusion Matrix

```
import seaborn as sns
import matplotlib.pyplot as plt

conf_matrix = confusion_matrix(y_test, y_pred_rf)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="YlGnBu", xticklabels=['Fail', 'Pass'], yticklabels=['Fail', 'Pass'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Random Forest')
plt.show()
```


Confusion Matrix - Random Forest
Actual
Fail
Pass
0
27
2
171
Fail
Pass
Predicted
160
140
120
100
80
60
40
20
0

Conclusions

This project, *Student Performance Analysis*, successfully demonstrated the application of data science techniques to understand and predict student outcomes using a publicly available dataset. The major accomplishments and observations drawn from the analysis are as follows:

1. **Comprehensive Data Understanding**:
   Through data inspection and exploratory analysis, we found that factors such as gender, parental education level, lunch type, and test preparation course significantly influence student scores.

2. **Effective Use of Data Visualization**:
   Visual tools like histograms, bar plots, and box plots helped in identifying patterns, score distributions, and disparities in academic performance across different groups. These insights added clarity and depth to the understanding of student behavior and outcomes.

3. **Model Development**:

   - Regression models (Linear Regression and Random Forest Regressor) were effective in predicting a student's average score.

   - Classification models (Logistic Regression and Random Forest Classifier) were useful in predicting whether a student would pass or fail.

   - Random Forest models consistently performed better due to their ensemble nature and ability to capture complex relationships.

4. **Feature Engineering Impact**:
   The introduction of a new feature — `average_score` — along with the derived `pass_status`, significantly improved the interpretability and practicality of the model.

5. **Model Evaluation**:
   The evaluation metrics such as R^2, MAE for regression and Accuracy, Precision, Recall, and Confusion Matrix for classification validated that the models were moderately effective, especially considering the simplicity of the dataset.

Recommendations

Based on the analysis and outcomes of the project, the following recommendations are suggested:

1. **Importance of Early Interventions**:
   Educational institutions should pay attention to students who have not completed a test preparation course or come from backgrounds with lower parental education levels, as these factors correlate with lower performance.

2. **Focused Support Programs**:
   Additional learning support, especially in reading and writing, can help underperforming students improve their scores and overall average.

3. **Further Data Collection**:
   Future studies can benefit from more detailed datasets that include:

   - Attendance records
   - Behavioral data
   - Subject-specific interest or feedback
   - Socioeconomic indicators

4. **Model Enhancements**:

   - Experimenting with advanced algorithms like XGBoost or Support Vector Machines (SVM).
   - Hyperparameter tuning using grid search or random search.
   - Cross-validation techniques to avoid overfitting.

5. **Deployment**:
   With further tuning, the classification model can be deployed in a school system to flag students at risk of failing, enabling timely academic support.