Name - Chaitanya Kelkar

Roll no. - 53

Experiment 1

Aim

To analyze the student performance dataset by performing data handling, statistical analysis, and visualization using Python's core Machine Learning libraries: NumPy, Pandas, Matplotlib, and Seaborn.

1. Dataset Source

- **Source:**student_performance.csv
- **Filename:** `student_performance.csv`

2. Dataset Description

The dataset contains academic performance metrics for students, focusing on the relationship between study habits, attendance, and various scores.

- **Type:** Multivariate, Numerical.
- **Size:** 20 instances (rows) 5 attributes (columns).
- **Target Variable (for analysis):** `Final_Score`.
- **Features:**
    1. **Hours_Studied:** Number of hours spent studying (Integer).
    2. **Attendance:** Percentage of classes attended (Integer).
    3. **Assignment_Score:** Score obtained in assignments (Integer).
    4. **Midterm_Score:** Score obtained in midterms (Integer).
    5. **Final_Score:** Final examination score (Integer).

3. Mathematical Formulation of the Algorithm

Since this experiment focuses on Statistical Analysis and Preprocessing rather than a predictive model, the mathematical foundations rely on **Descriptive Statistics** and **Normalization techniques**.

A. Arithmetic Mean (): Used to find the central tendency of the scores.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

B. Standard Deviation (): Used to measure the amount of variation or dispersion in the dataset.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

C. Min-Max Normalization: Used in Exercise 1 to scale the `Final_Score` between 0 and 1.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

D. Pearson Correlation Coefficient (): Used in the Heatmap (Exercise 4) to measure the linear correlation between variables.

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

4. Algorithm Limitations

While the statistical methods used (Mean, Normalization, Correlation) are fundamental, they have specific limitations:

1. **Sensitivity to Outliers:** The Mean and Min-Max Normalization are highly sensitive to outliers. A single extremely high or low score can skew the average and compress the normalized range.
2. **Correlation vs. Causation:** The correlation analysis (Heatmap) shows that `Hours_Studied` and `Final_Score` are related, but it does not mathematically prove that studying causes higher scores (though intuitively true).
3. **Linear Assumption:** Pearson correlation only detects *linear* relationships. If the relationship between Attendance and Score was non-linear (e.g., exponential), these metrics might underestimate the connection.

5. Methodology / Workflow

The experiment followed a standard Exploratory Data Analysis (EDA) pipeline:

1. **Data Ingestion:** Loaded `student_performance.csv` into a Pandas DataFrame.
2. **Data Inspection:** Checked the shape and verified there were no missing values (Null check).
3. **Statistical Processing (NumPy):**
    - Converted columns to NumPy arrays.
    - Calculated Mean, Median, and Standard Deviation.
    - Applied Min-Max Normalization to scale data.

4. **Feature Engineering:** Created a new categorical column `Performance` based on `Final_Score` logic (Good/Average/Poor).
5. **Visualization:**
    - **Matplotlib:** Generated Line plots and Histograms to understand trends and distributions.
    - **Seaborn:** Generated Scatter plots, Heatmaps, and Boxplots for multivariate analysis.

6. Performance Analysis

In the context of Exploratory Data Analysis, "Performance" is measured by the insights derived from the data:

- **Distribution Analysis:** The Histogram of `Final_Score` showed a relatively normal distribution centered around the mean of **68.95**.
- **Trend Analysis:** The Line Plot and Seaborn Scatter Plot confirmed a **strong positive linear relationship** between `Hours_Studied` and `Final_Score`. As hours increase, scores increase consistently.
- **Correlation Analysis:** The Heatmap revealed high correlation coefficients between `Attendance`, `Assignment_Score`, and `Final_Score`, suggesting that consistent engagement throughout the semester predicts final results accurately.
- **Categorical Insight:** The Boxplot showed that students labeled as "Good" performers had significantly higher median attendance than those labeled "Poor."

7. Hyperparameter Tuning

- **Status:** Not Applicable.
- **Reason:** Hyperparameter tuning (e.g., Grid Search, Random Search) is specific to **Predictive Modeling algorithms** (like Decision Trees, SVM, or Neural Networks) where model parameters must be optimized to minimize error.
- **Context:** Since this specific experiment focused on **Data Preprocessing and Visualization** (calculating fixed statistical properties like Mean/Std Dev), there were no learning parameters to tune. The results are deterministic based on the dataset provided.

Theory

This experiment focuses on the four foundational pillars of data science and machine learning in Python.

1. NumPy (Numerical Python)

Theory: NumPy is the fundamental package for scientific computing in Python. It provides the nd array object, which is an n-dimensional array that is much faster and more efficient than standard Python lists.

2. Pandas (Python Data Analysis Library)

Theory: Pandas is built on top of NumPy and provides high-level data structures like Series (1D) and DataFrames (2D) designed to make working with structured ("tabular") data easy and intuitive. It mimics the functionality of Excel or SQL within Python.

3. Matplotlib (Plotting Library)

Theory: Matplotlib is a low-level library for creating static, animated, and interactive visualizations in Python. It provides fine-grained control over every element of a plot (axes, lines, labels).

4. Seaborn (Statistical Data Visualization)

Theory: Seaborn is a high-level interface built on top of Matplotlib. It provides a more attractive default aesthetic and simplifies the creation of complex statistical plots.

Code-

1. Uploading and reading the csv file.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('student_performance.csv')
print("Dataset loaded successfully!")
print(df.head())
```

2. Using numpy functions to calculate mean, median, mode and normalized the score.

```
print("--- Exercise 1: NumPy Basics ---")

final_scores_array = df['Final_Score'].to_numpy()

mean_val = np.mean(final_scores_array)
median_val = np.median(final_scores_array)
std_val = np.std(final_scores_array)

print(f"Mean Final Score: {mean_val:.2f}")
print(f"Median Final Score: {median_val:.2f}")
print(f"Standard Deviation: {std_val:.2f}")

min_val = np.min(final_scores_array)
max_val = np.max(final_scores_array)
normalized_scores = (final_scores_array - min_val) / (max_val - min_val)

print("First 5 normalized scores:", normalized_scores[:5])
```

3. Checking shape, columns, and missing values and creating Performance label based on Final_Score.

```
print("--- Exercise 2: Pandas Data Handling ---")

print(f"Shape of dataset: {df.shape}")
print(f"Columns: {list(df.columns)}")
print("Missing values:\n", df.isnull().sum())

def classify_performance(score):
    if score >= 85:
        return 'Good'
    elif score >= 50:
        return 'Average'
    else:
        return 'Poor'

df['Performance'] = df['Final_Score'].apply(classify_performance)

print("\nFirst 5 rows with Performance column:")
print(df.head())
```

4. Plotting Line plot: Hours_Studied vs Final_Score and Histogram of Final_Score.

```
print("--- Exercise 3: Matplotlib Visualization ---")

df_sorted = df.sort_values(by='Hours_Studied')

plt.figure(figsize=(8, 5))
plt.plot(df_sorted['Hours_Studied'], df_sorted['Final_Score'])
plt.title('Hours Studied vs Final Score (Line Plot)')
plt.xlabel('Hours Studied')
plt.ylabel('Final Score')
plt.grid(True)
plt.show()

plt.figure(figsize=(8, 5))
plt.hist(df['Final_Score'], bins=10, color='skyblue', edgecolor='black')
plt.title('Distribution of Final Scores (Histogram)')
plt.xlabel('Final Score')
plt.ylabel('Frequency')
plt.show()
```

5. Plotting scatter plot using seaborn,heatmap for correlation analysis and boxplot for categorical analysis.

```
print("--- Exercise 4: Seaborn Visualization ---")

sns.relplot(data=df, x='Hours_Studied', y='Final_Score', hue='Performance', style='Performance', s=100, height=5, aspect=1.5)

plt.figure(figsize=(8, 6))
sns.heatmap(df.select_dtypes('number').corr(), annot=True, cmap='coolwarm', fmt=".2f")

sns.catplot(data=df, x='Performance', y='Attendance', kind='box', hue='Performance', palette='Set2', order=['Good', 'Average', 'Poor'], height=5, aspect=1.5)

plt.show()
```

Output-

1.

```
Dataset loaded successfully!
   Hours_Studied  Attendance  Assignment_Score  Midterm_Score  Final_Score
0              1          60                55             50           52
1              2          65                58             55           57
2              3          70                60             58           60
3              4          75                65             62           64
4              5          80                68             65           68
```

2.

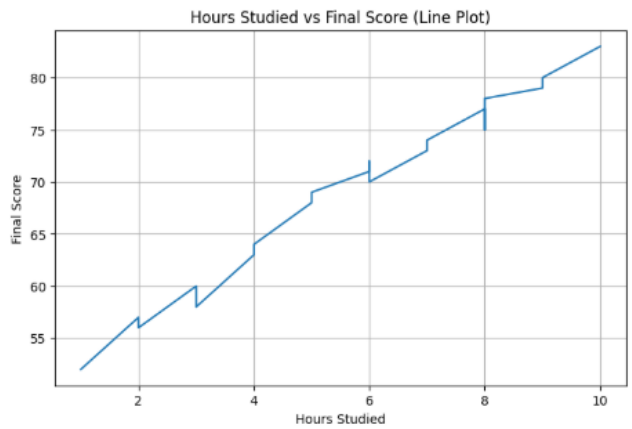
```
--- Exercise 1: NumPy Basics ---
Mean Final Score: 68.95
Median Final Score: 70.50
Standard Deviation: 8.71
First 5 normalized scores: [0.         0.16129032 0.25806452 0.38709677 0.51612903]
```

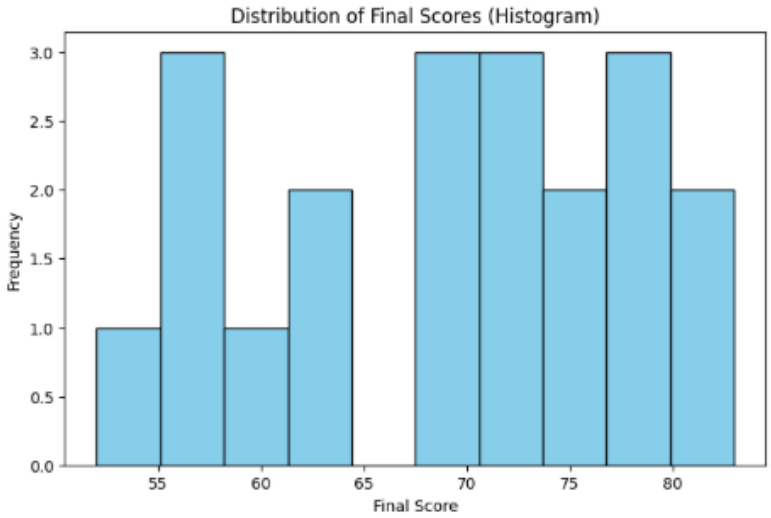
3.

```
--- Exercise 2: Pandas Data Handling ---
Shape of dataset: (20, 6)
Columns: ['Hours_Studied', 'Attendance', 'Assignment_Score', 'Midterm_Score', 'Final_Score', 'Performance']
Missing values:
 Hours_Studied       0
Attendance          0
Assignment_Score    0
Midterm_Score       0
Final_Score         0
Performance         0
dtype: int64

First 5 rows with Performance column:
   Hours_Studied  Attendance  Assignment_Score  Midterm_Score  Final_Score  \
0              1          60                55             50           52
1              2          65                58             55           57
2              3          70                60             58           60
3              4          75                65             62           64
4              5          80                68             65           68

  Performance
0     Average
1     Average
2     Average
3     Average
4     Average
```

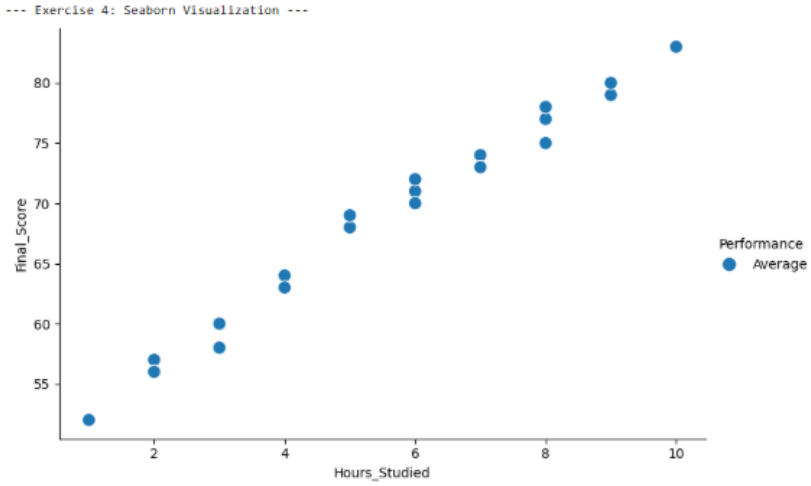
4.

```
--- Exercise 3: Matplotlib Visualization ---
```

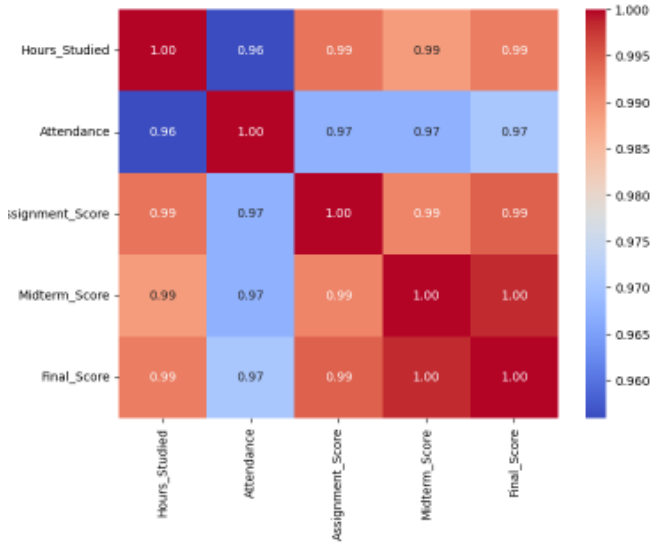


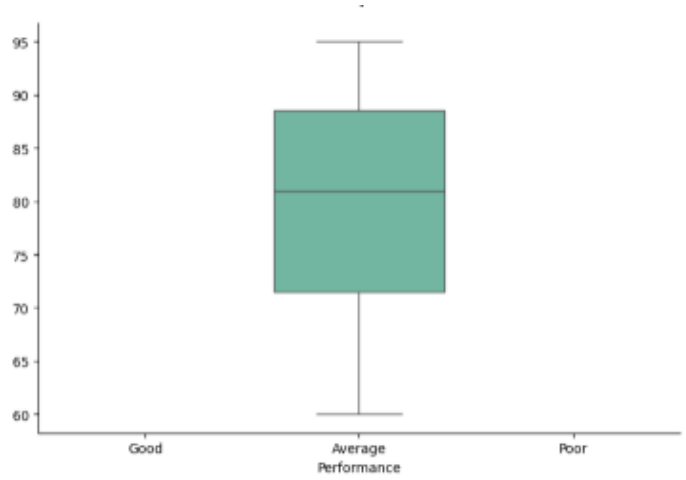

Distribution of Final Scores (Histogram)
Frequency
Final Score
0.0
0.5
1.0
1.5
2.0
2.5
3.0
55
60
65
70
75
80

5.


--- Exercise 4: Seaborn Visualization ---
Final_Score
Hours_Studied
Performance
Average
55
60
65
70
75
80
2
4
6
8
10

.


Hours_Studied
Attendance
isignment_Score
Midterm_Score
Final_Score
1.00 0.96 0.99 0.99 0.99
0.96 1.00 0.97 0.97 0.97
0.99 0.97 1.00 0.99 0.99
0.99 0.97 0.99 1.00 1.00
0.99 0.97 0.99 1.00 1.00
Hours_Studied
Attendance
Assignment_Score
Midterm_Score
Final_Score
1.000
0.995
0.990
0.985
0.980
0.975
0.970
0.965
0.960



Conclusion-

By integrating these four libraries, we transformed raw data into meaningful insights. This experiment established a solid foundation for Exploratory Data Analysis (EDA), proving that Python provides a robust and flexible ecosystem for the data preprocessing and visualization stages essential to any Machine Learning pipeline.