Experiment 1

Aim

To understand data handling, visualization, and exploratory data analysis using NumPy, Pandas, Matplotlib, and Seaborn for Machine Learning applications.

1. Dataset Source

The dataset used in this experiment is a Student Performance Dataset, containing academic and behavioral attributes of students.

- **Source Link:** https://github.com/Atharvasp333/ML-Lab/blob/main/datasets/student_performance.csv

2. Dataset Description

The dataset consists of student academic performance records suitable for exploratory data analysis (EDA) and feature correlation analysis.

- **Dataset Characteristics:**
  - **Type:** Tabular, numerical
  - **Size:** Medium-sized dataset (suitable for EDA)
  - **Target Variable:** Final_Score
  - **Data Quality:** No missing values detected
- **Feature Description:**

Feature Name	Description
Hours_Studied	Number of hours a student studied
Attendance	Attendance percentage
Assignment_Score	Assignment marks
Midterm_Score	Midterm examination marks

Final_Score	Final examination marks (Target variable)

3. Mathematical Formulation of the Algorithm

Although no predictive model is trained in this experiment, the following statistical and mathematical operations form the foundation of the preprocessing:

- **Mean:** Represents the central value of the data.
  $$\mu = \frac{1}{N} \sum x_i$$
  Where $i=1$ to N.
- **Standard Deviation:** Measures the amount of variation or dispersion.
  $$\sigma = \sqrt{\frac{1}{N} \sum (x_i - \mu)^{2}}$$
  Where $i=1$ to N.
- **Min-Max Normalization:** Rescales features to a range of $[0, 1]$.
  $$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$
- **Correlation Coefficient:** Measures the linear relationship between two variables.
  $$r = \frac{Cov(X,Y)}{\sigma_X \cdot \sigma_Y}$$

These mathematical concepts are critical for feature scaling, variance analysis, and pattern discovery.

4. Algorithm Limitations

Since this experiment focuses on EDA rather than predictive modeling, it has the following limitations:

- **No Prediction:** The workflow provides descriptive insights but has no classification or prediction capability.
- **No Metrics:** It is not possible to measure model accuracy or loss.
- **Sensitivity:** Techniques like Min-Max normalization are sensitive to outliers.
- **Subjectivity:** Visualization interpretation may be subjective.
- **Incompleteness:** EDA must always be followed by model-based learning for a complete ML pipeline.

5. Methodology / Workflow

The experiment follows a step-by-step workflow to process and analyze the data:

1. **Data Loading:** Load the dataset using Pandas.
2. **Conversion:** Convert numerical columns to NumPy arrays for efficient calculation.
3. **Statistical Analysis:** Compute mean, median, and standard deviation.
4. **Normalization:** Apply Min-Max scaling to normalize the data.

5. **Labeling:** Create a categorical "Performance" label based on score thresholds.
6. **Visualization:** Generate plots using Matplotlib (Line plots, Histograms) and Seaborn (Scatter plots, Boxplots).
7. **Correlation Analysis:** Analyze feature relationships using correlation heatmaps.

6. Performance Analysis

Since no ML model is trained, performance is analyzed using statistical insights and visual interpretation:

- **Study Time Impact:** Higher study hours generally correspond to higher final scores.
- **Correlations:** Strong positive correlations were observed between Assignment Score, Midterm Score, and Final Score.
- **Categorization:** Performance categories clearly separate student outcomes.
- **Distribution:** Boxplots effectively show the score spread and outliers across different categories.

These insights help in feature selection for future machine learning models.

7. Hyperparameter Tuning

Hyperparameter tuning is **not applicable** in this experiment because:

- No machine learning algorithm (e.g., Regression, Random Forest) is being trained.
- The focus of EDA is on data understanding rather than optimization of model parameters.

Exercise 1: Numpy Basics

Code:

```
import numpy as np
import pandas as pd

df = pd.read_csv('student_performance.csv')

# 1. Load Final_Score as NumPy array
final_scores = df['Final_Score'].to_numpy()

# 2. Compute mean, median, and standard deviation
mean_score = np.mean(final_scores)
median_score = np.median(final_scores)
std_dev = np.std(final_scores)

print(f"Mean Final Score: {mean_score}")
print(f"Median Final Score: {median_score}")
print(f"Standard Deviation: {std_dev}")
```

```
# 3. Perform Min-Max normalization
# Formula: (x - min) / (max - min)
min_val = np.min(final_scores)
max_val = np.max(final_scores)
normalized_scores = (final_scores - min_val) / (max_val - min_val)
print("\nFirst 5 Normalized Scores:")
print(normalized_scores[:5])
```

Output:

```
Mean Final Score: 68.95
Median Final Score: 70.5
Standard Deviation: 8.714786285388 76

First 5 Normalized Scores:
[0.         0.16129032 0.25806452 0.38709677 0.51612903]
```

Exercise 2: Pandas Data Handling

Code:

```
# 1. Check shape, columns, and missing values
print("Dataset Shape:", df.shape)
print("Columns:", df.columns.tolist())
print("\nMissing Values:\n", df.isnull().sum())

# 2. Create Performance label based on Final_Score
# Logic: >= 80 Excellent, >= 70 Good, >= 60 Average, < 60 Low
def get_performance(score):
    if score >= 80:
        return 'Excellent'
    elif score >= 70:
        return 'Good'
    elif score >= 60:
        return 'Average'
    else:
        return 'Low'

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

# Display the first few rows to verify the new column
print("\nUpdated DataFrame head:")
print(df.head())
```

Output:

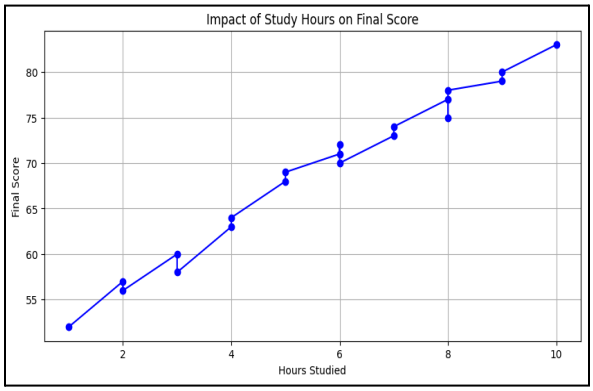
```
Dataset Shape: (20, 5)
Columns: ['Hours_Studied', 'Attendance', 'Assignment_Score', 'Midterm_Score', 'Final_Score']

Missing Values:
 Hours_Studied       0
Attendance           0
Assignment_Score     0
Midterm_Score        0
Final_Score          0
dtype: int64

Updated DataFrame head:
   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         Low
1         Low
2     Average
3     Average
4     Average
```

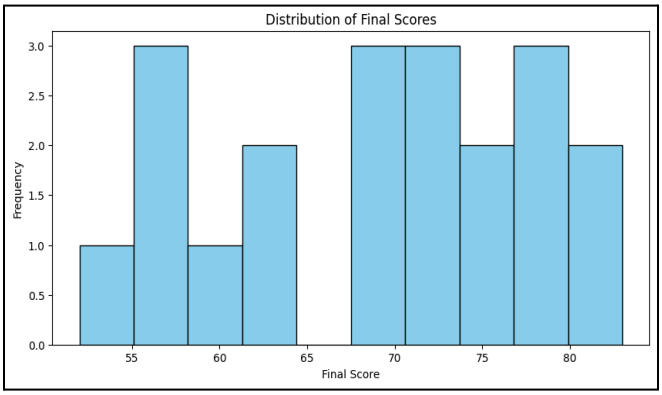
Exercise 3: Matplotlib Visualization

Code:

```
import matplotlib.pyplot as plt

# 1. Line plot: Hours_Studied vs Final_Score
# Sort data by Hours_Studied for a clean line plot
df_sorted = df.sort_values('Hours_Studied')
plt.figure(figsize=(10, 5))
plt.plot(df_sorted['Hours_Studied'], df_sorted['Final_Score'], marker='o', linestyle='-', color='b')
plt.title('Impact of Study Hours on Final Score')
plt.xlabel('Hours Studied')
plt.ylabel('Final Score')
plt.grid(True)
plt.show()

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

Output:



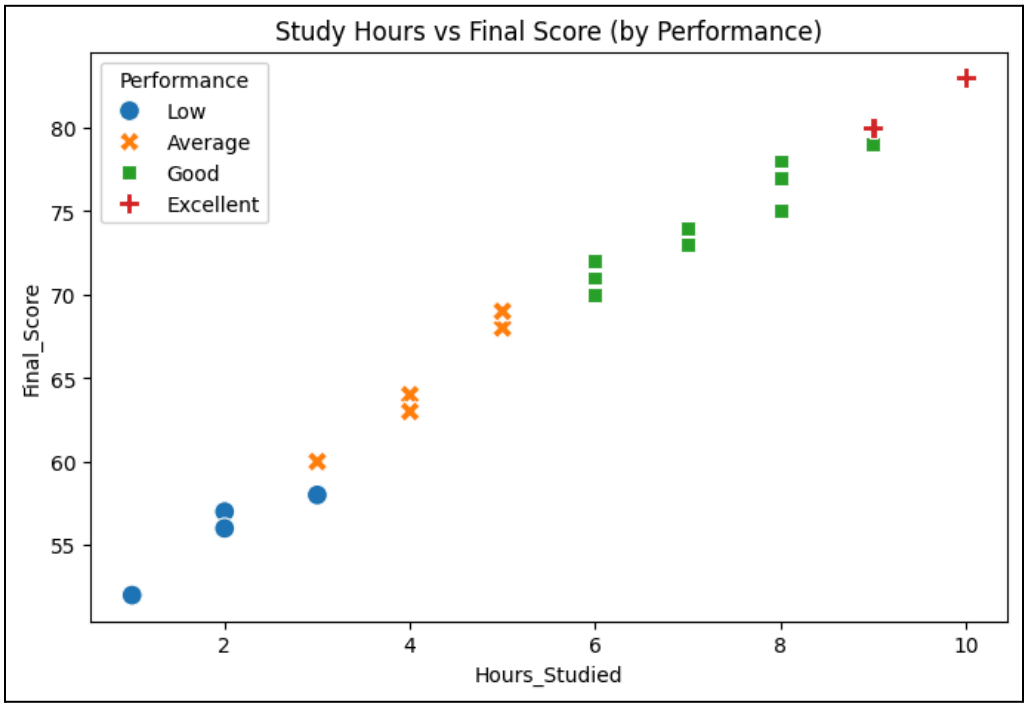


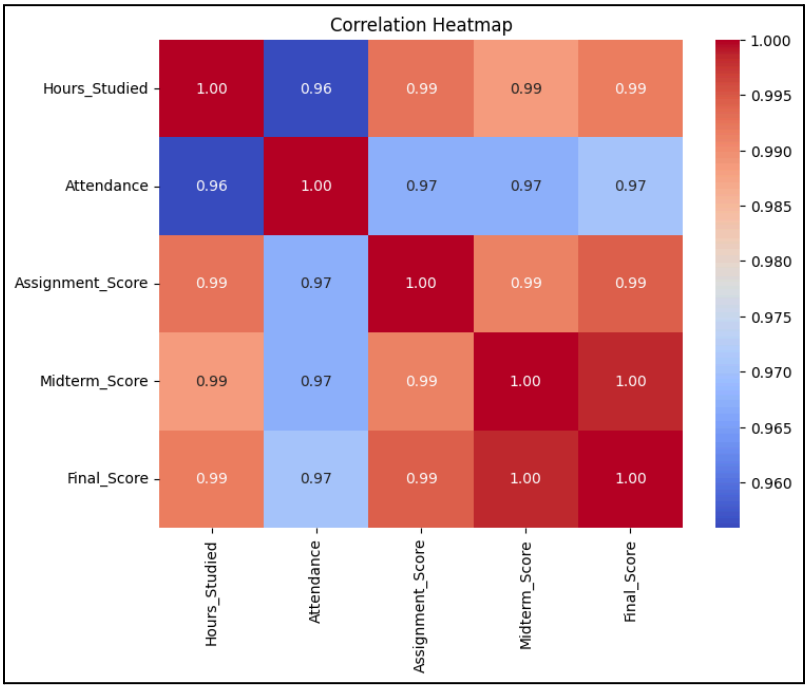
Exercise 4: Seaborn Visualization

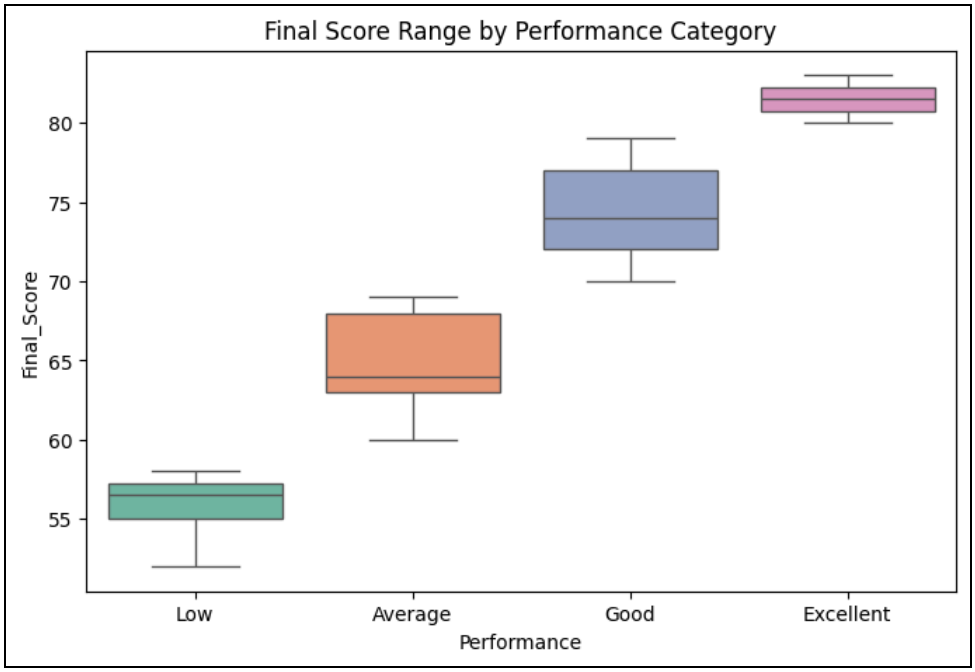
Code:

```
import seaborn as sns

# 1. Scatter plot using seaborn
plt.figure(figsize=(8, 5))
sns.scatterplot(data=df, x='Hours_Studied', y='Final_Score', hue='Performance',
style='Performance', s=100)
plt.title('Study Hours vs Final Score (by Performance)')
plt.show()

# 2. Heatmap for correlation analysis
plt.figure(figsize=(8, 6))
# Select only numeric columns for correlation
numeric_df = df.select_dtypes(include=[np.number])
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()

# 3. Boxplot for categorical analysis
plt.figure(figsize=(8, 5))
# Define a logical order for the categories
order_list = ['Low', 'Average', 'Good', 'Excellent']
sns.boxplot(data=df, x='Performance', y='Final_Score', order=order_list, palette='Set2')
plt.title('Final Score Range by Performance Category')
plt.show()
```

Output:


Study Hours vs Final Score (by Performance)
Performance
Low
Average
Good
Excellent
Final_Score
Hours_Studied
80
75
70
65
60
55
2
4
6
8
10


Correlation Heatmap
Hours_Studied
Attendance
Assignment_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
1.000
0.995
0.990
0.985
0.980
0.975
0.970
0.965
0.960



Conclusion

This lab demonstrated essential Python libraries for machine learning data analysis. NumPy enabled efficient numerical operations and normalization, while Pandas facilitated data exploration. Matplotlib and Seaborn provided powerful visualization capabilities to identify patterns, such as the strong correlation between study hours and final scores.