EXPERIMENT - 0

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.
https://github.com/tanaya2005/MLDL-lab/blob/main/Experiment-0/expanded_student_performance.csv

2. Dataset Description

The dataset contains academic performance data of students.

- Dataset Characteristics

- Type: Tabular (Numerical)
- Domain: Education / Academic Performance
- Target Variable: Final_Score
- Size: Medium-sized dataset
- Data Quality: Contains missing (null) values

- Feature Description

Feature Name	Description
Hours_Studied	Number of hours studied
Attendance	Attendance percentage
Assignment_Score	Assignment marks
Midterm_Score	Midterm exam marks
Final_Score	Final exam marks (Target)

3. Mathematical Formulation of the Algorithm

Mean	$\mu = \frac{1}{N}\sum x_i$
Standard Deviation:	$\sigma = \sqrt{\frac{1}{N}\sum (x_i - \mu)^2}$
Min-Max Normalization:	$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$
Correlation Coefficient:	$r = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$

4. Algorithm Limitations

- No prediction capability
- Only descriptive analysis
- Visualization interpretation is subjective
- Correlation does not imply causation
- Min-max scaling sensitive to outliers

5. Methodology / Workflow

1. Load dataset
2. Check and handle missing values
3. Compute statistical measures
4. Normalize data
5. Create performance category

6. Generate visualizations
7. Perform correlation analysis

6. Performance Analysis

Since no model is trained:

- Study hours positively impact final score
- Assignment and midterm scores show strong correlation
- Performance categories clearly separate students
- Boxplots highlight distribution spread

CODE

Google Colab Link:
https://colab.research.google.com/drive/1j0nBG4actTpNfK-R8QXmB8Dxtqlz6bkb?usp=sharing

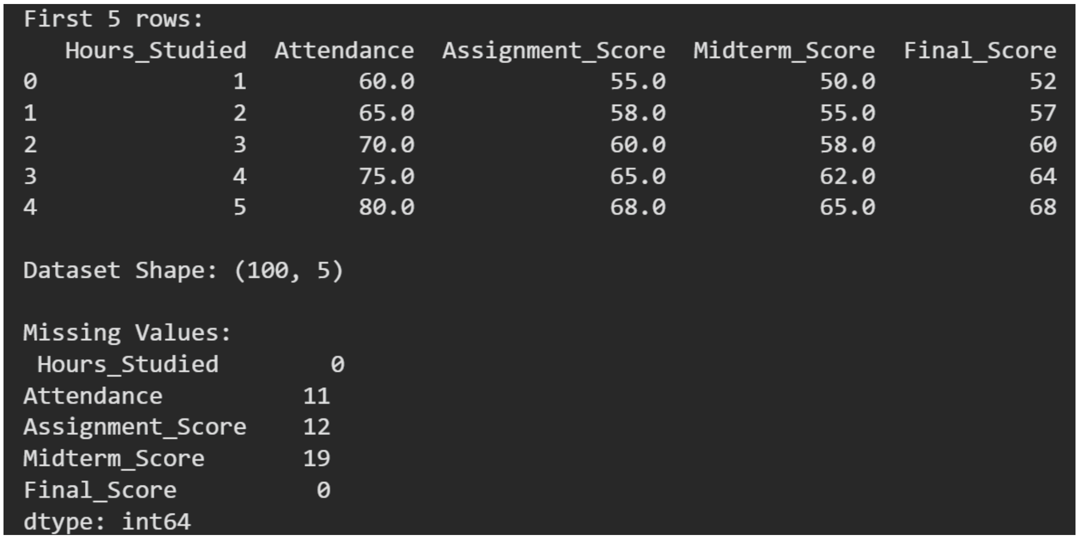
1. Import libraries

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

2. Load Dataset

```
df = pd.read_csv('/content/expanded_student_performance.csv')
print("First 5 Rows:")
print(df.head())

print("\nShape:", df.shape)
print("\nMissing Values:\n", df.isnull().sum())
```

Output:

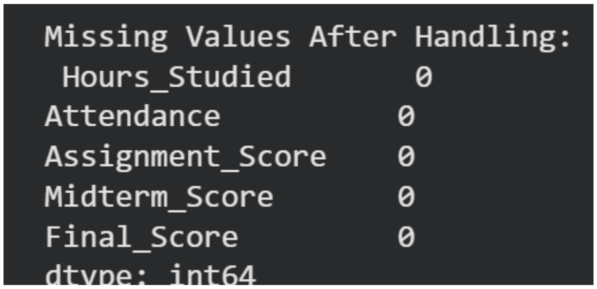


```
First 5 rows:
   Hours_Studied  Attendance  Assignment_Score  Midterm_Score  Final_Score
0              1        60.0              55.0           50.0           52
1              2        65.0              58.0           55.0           57
2              3        70.0              60.0           58.0           60
3              4        75.0              65.0           62.0           64
4              5        80.0              68.0           65.0           68

Dataset Shape: (100, 5)

Missing Values:
 Hours_Studied        0
Attendance          11
Assignment_Score    12
Midterm_Score       19
Final_Score          0
dtype: int64
```

3. Handle Missing Values

```
# Fill numerical missing values using mean
df.fillna(df.mean(), inplace=True)

print("\nMissing Values After Handling:\n", df.isnull().sum())
```

Output:

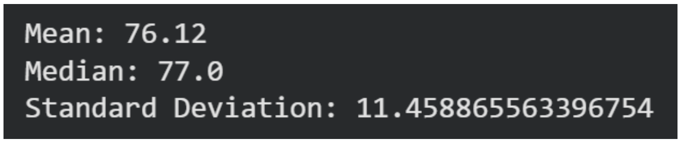


```
Missing Values After Handling:
 Hours_Studied       0
Attendance          0
Assignment_Score    0
Midterm_Score       0
Final_Score         0
dtype: int64
```

4. NumPy Statistical Analysis

```
final_scores = df['Final_Score'].to_numpy()

print("Mean:", np.mean(final_scores))
print("Median:", np.median(final_scores))
print("Standard Deviation:", np.std(final_scores))
```

Output:

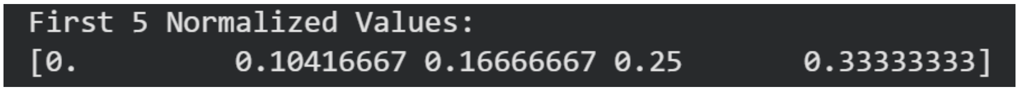


```
Mean: 76.12
Median: 77.0
Standard Deviation: 11.458865563396754
```

5. Min-Max Normalization

```
min_val = np.min(final_scores)
max_val = np.max(final_scores)

normalized = (final_scores - min_val) / (max_val - min_val)

print("First 5 Normalized Values:")
print(normalized[:5])
```

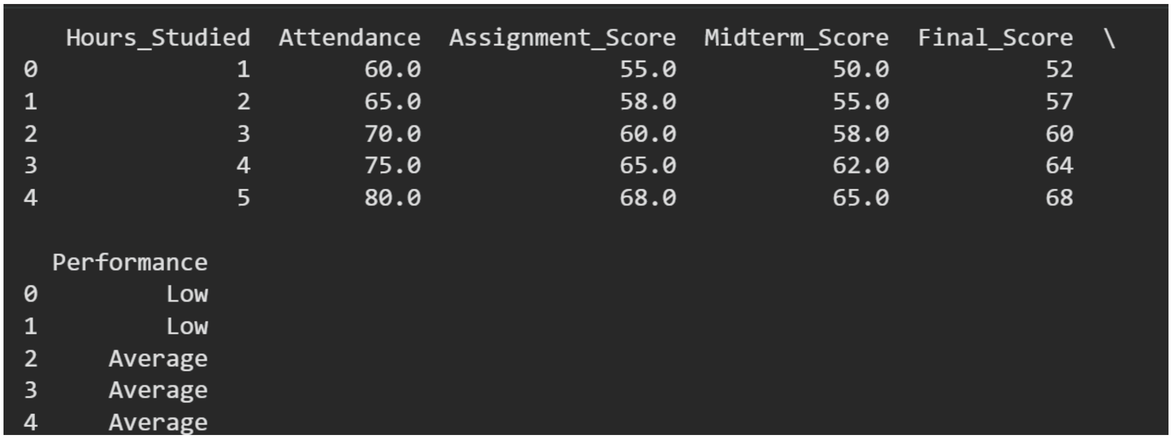
Output:



```
First 5 Normalized Values:
[0.         0.10416667 0.16666667 0.25       0.33333333]
```

6. Create Performance Label

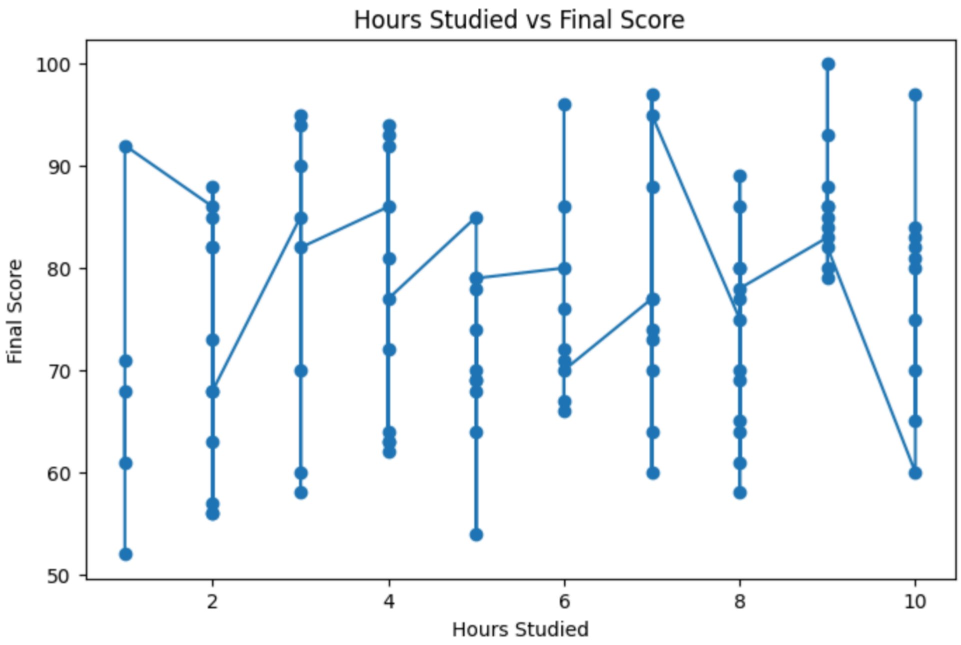
```
def get_performance(score):
    if score >= 80:
        return 'Excellent'
    elif score >= 70:
        return 'Good'
    elif score >= 60:
        return 'Average'
    else:
        return 'Low'

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

print(df.head())
```



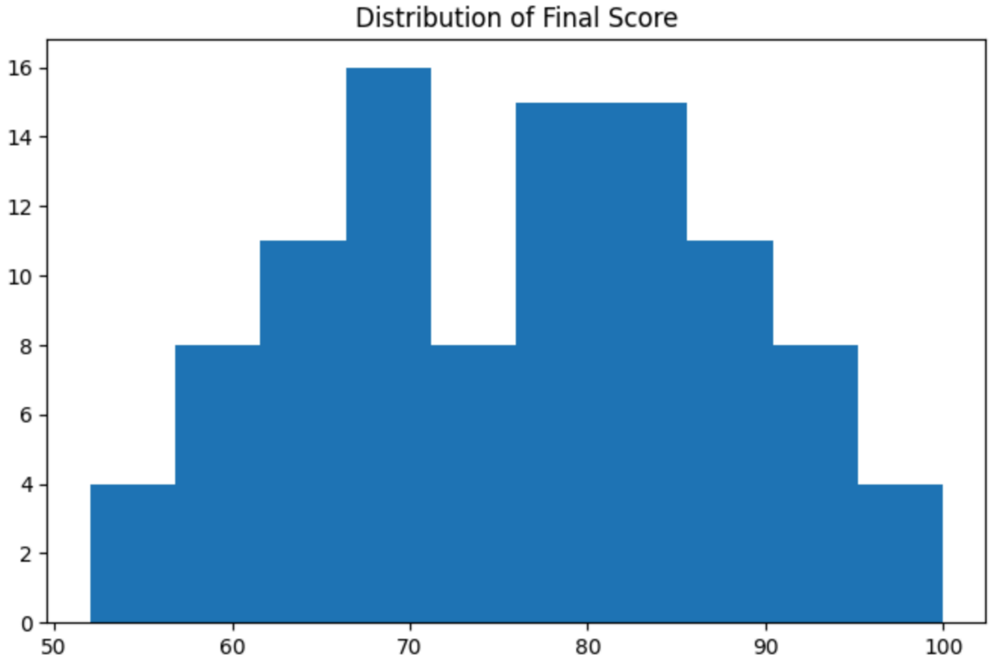
```
   Hours_Studied  Attendance  Assignment_Score  Midterm_Score  Final_Score  \
0              1        60.0              55.0           50.0           52
1              2        65.0              58.0           55.0           57
2              3        70.0              60.0           58.0           60
3              4        75.0              65.0           62.0           64
4              5        80.0              68.0           65.0           68

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

7. Matplotlib Visualizations

```
# Line Plot
df_sorted = df.sort_values('Hours_Studied')

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

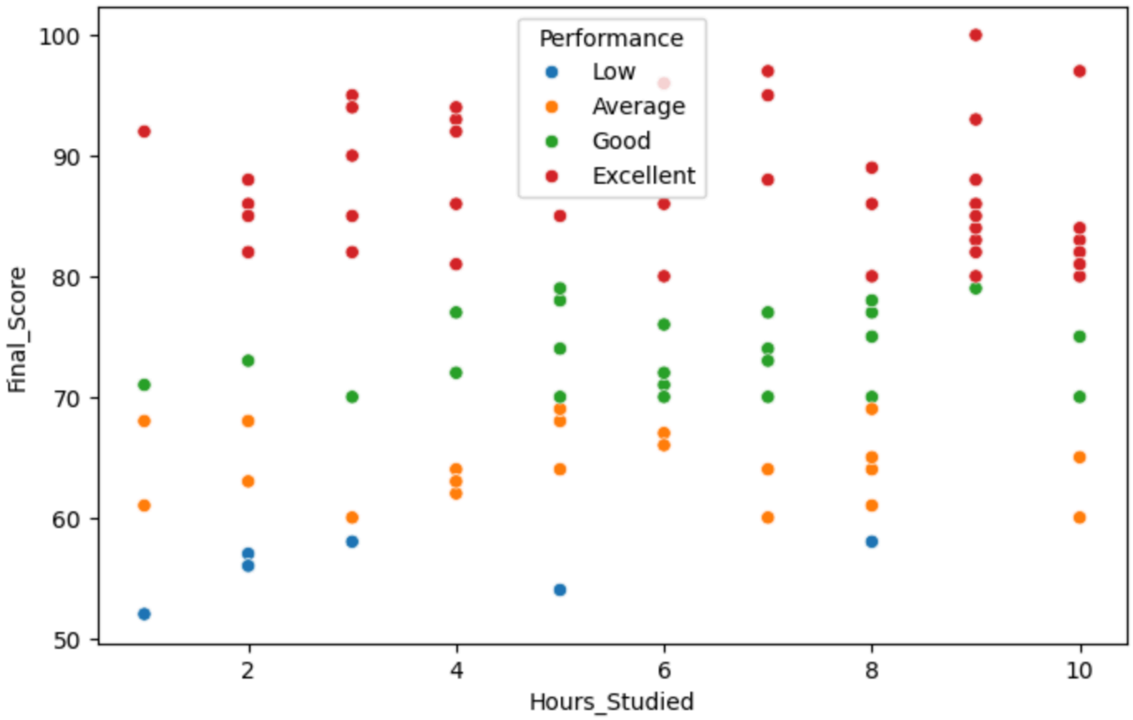


```
# Histogram
plt.figure(figsize=(8,5))
plt.hist(df['Final_Score'], bins=10)
plt.title("Distribution of Final Score")
plt.show()
```

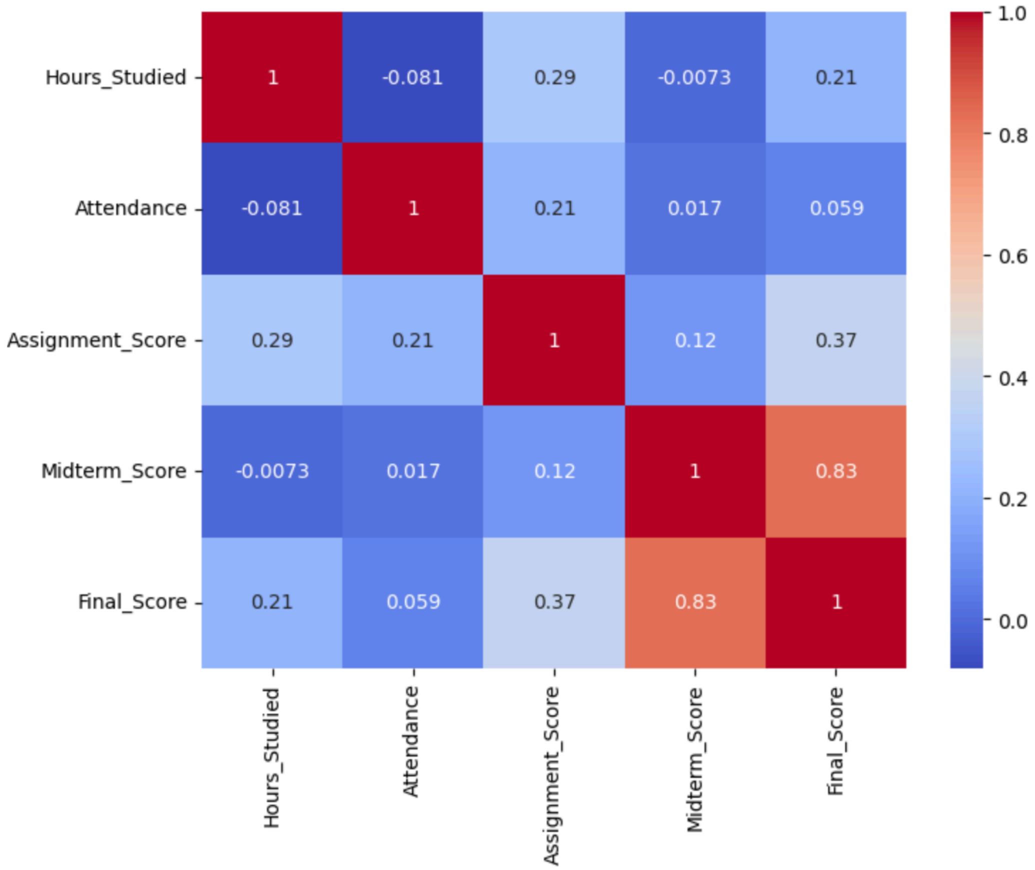


8. Seaborn Visualizations

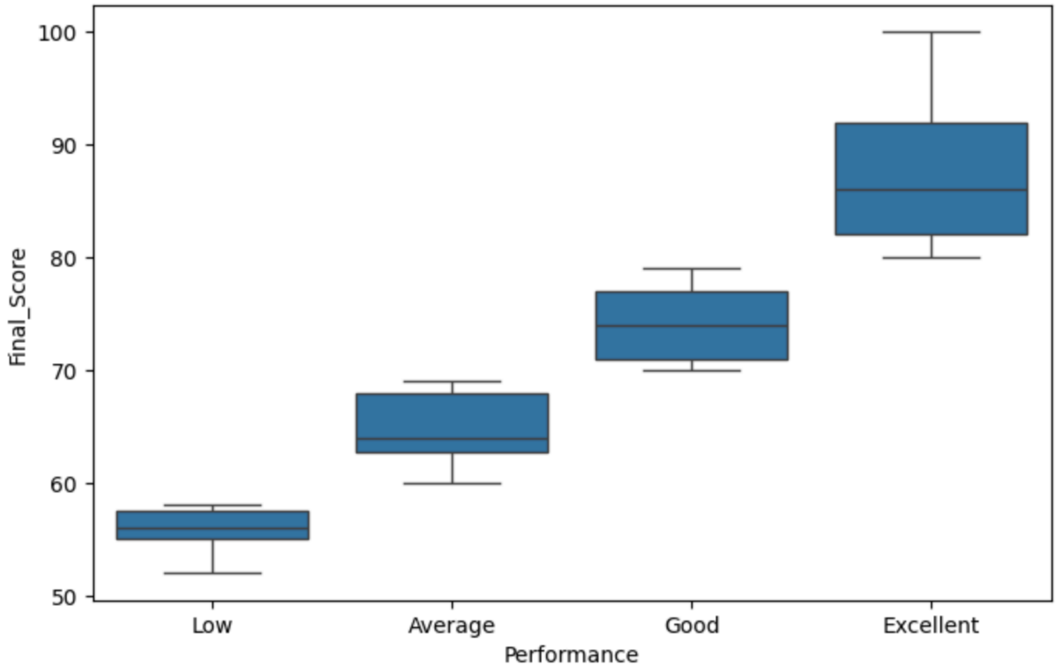
```
# Scatter Plot
plt.figure(figsize=(8,5))
sns.scatterplot(data=df, x='Hours_Studied', y='Final_Score', hue='Performance')
plt.show()
```



```
# Heatmap
plt.figure(figsize=(8,6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.show()
```



```
# Boxplot
plt.figure(figsize=(8,5))
sns.boxplot(data=df, x='Performance', y='Final_Score')
plt.show()
```



CONCLUSION

This experiment demonstrated complete ML pipeline development including data cleaning, visualization, regression modeling, classification, evaluation, and hyperparameter tuning. Missing value handling improved data quality, while visualization revealed strong correlation between study hours and academic performance. Logistic regression successfully classified pass/fail students with good accuracy.