

## EXPERIMENT

### AIM:

To implement NumPy, Pandas, Matplotlib & Seaborn for Machine Learning

### THEORY:

#### 1. NumPy (Numerical Python)

NumPy is a fundamental Python library used for numerical computing and handling multi-dimensional arrays. It provides efficient data structures such as `ndarray` and mathematical functions to perform statistical and algebraic operations.

In this experiment, NumPy was used to:

- Convert the `Final_Score` column into a NumPy array.
- Compute statistical measures such as **mean, median, and standard deviation**.
- Perform **Min-Max normalization**, which scales data between 0 and 1.

Normalization is important in Machine Learning because it ensures that all features contribute equally to the model and prevents bias due to large value differences.

#### 2. Pandas (Data Handling and Preprocessing)

Pandas is a powerful Python library used for data manipulation and analysis. It provides data structures such as `DataFrame` and `Series` that allow easy handling of structured datasets.

In this experiment, Pandas was used to:

- Load the dataset from a CSV/Excel file.
- Examine dataset shape, column names, and missing values.
- Handle missing values using **mean imputation**.

- Create a new categorical column called Performance based on Final\_Score.

Handling missing values is a critical preprocessing step in Machine Learning to ensure data quality and model reliability.

### 3. Matplotlib (Data Visualization)

Matplotlib is a widely used Python library for creating static, animated, and interactive visualizations.

In this experiment, Matplotlib was used to:

- Create a **Line Plot** to visualize the relationship between Hours\_Studied and Final\_Score.
- Generate a **Histogram** to analyze the distribution of final scores.

Visualization helps in identifying patterns, trends, and distributions within the dataset, which is an essential part of Exploratory Data Analysis (EDA).

### 4. Seaborn (Advanced Statistical Visualization)

Seaborn is a high-level visualization library built on top of Matplotlib. It provides attractive and informative statistical graphics.

In this experiment, Seaborn was used to:

- Create a **Scatter Plot** to examine the relationship between study hours and final score.
- Generate a **Heatmap** to analyze correlations between all numerical features.
- Create a **Boxplot** to compare performance categories.

The heatmap revealed strong positive correlations among academic performance metrics, while the boxplot clearly differentiated student performance categories.

## 5. Exploratory Data Analysis (EDA)

EDA is the process of analyzing datasets to summarize their main characteristics using statistical and visualization techniques.

In this experiment:

- Statistical measures were computed.
- Missing values were handled.
- Feature relationships were visualized.
- Correlation among variables was examined.

EDA helps in understanding the dataset before applying Machine Learning models.

### CODE & OUTPUT

#### **Dataset: student\_performance.csv**

Columns: Hours\_Studied, Attendance, Assignment\_Score, Midterm\_Score, Final\_Score

	A	B	C	D	E
1	Hours_Studied	Attendance	Assignment_Score	Midterm_Score	Final_Score
2	1	60	55	50	52
3	2	65	58	55	57
4	3	70	60	58	60
5	4	75	65	62	64
6	5	80	68	65	68
7	6	85	72	68	71
8	7	90	75	70	74
9	8	95	78	72	77
10	9	88	80	75	79
11	10	92	85	78	83
12	4	72	62	60	
13	6	78	70		70
14	8	85	76	71	75
15	2	66	57	54	56
16	5	80	69	66	69
17	7	88	74	70	73
18	9	94	82	76	80
19	3	68		56	58
20	6		71	69	72
21	8	90	79	73	78
22					

### **Exercise 1: NumPy Basics**

1. Load Final\_Score as NumPy array.
2. Compute mean, median, and standard deviation.
3. Perform Min-Max normalization.

```
▶ import numpy as np
import pandas as pd

df = pd.read_excel("student_performance_modified.csv.xlsx")

final_scores = df['Final_Score'].dropna().values

mean = np.mean(final_scores)
median = np.median(final_scores)
std_dev = np.std(final_scores)

print("Mean:", mean)
print("Median:", median)
print("Standard Deviation:", std_dev)

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

normalized = (final_scores - min_val) / (max_val - min_val)
print("Min-Max Normalized Values:\n", normalized)

...
*** Mean: 69.26315789473684
Median: 71.0
Standard Deviation: 8.830819865753408
Min-Max Normalized Values:
[0.          0.16129032 0.25806452 0.38709677 0.51612903 0.61290323
 0.70967742 0.80645161 0.87096774 1.          0.58064516 0.74193548
 0.12903226 0.5483871 0.67741935 0.90322581 0.19354839 0.64516129
 0.83870968]
```

## Exercise 2: Pandas Data Handling

1. Load CSV file using Pandas.
2. Check shape, columns, and missing values.
3. Create Performance label based on Final\_Score.

```

import pandas as pd

df = pd.read_excel("student_performance_modified.csv.xlsx")
df.head()

# Shape
print("Shape of dataset:", df.shape)

# Column names
print("Columns:", df.columns)

# Missing values
print("Missing values:\n", df.isnull().sum())

df.fillna(df.mean(numeric_only=True), inplace=True)

print("Missing values after filling:\n", df.isnull().sum())

def performance_label(score):
    if score >= 75:
        return "Excellent"
    elif score >= 60:
        return "Good"
    else:
        return "Average"

df["Performance"] = df["Final_Score"].apply(performance_label)

df.head()

```

```

Shape of dataset: (20, 5)
Columns: Index(['Hours_Studied', 'Attendance', 'Assignment_Score', 'Midterm_Score',
       'Final_Score'],
       dtype='object')
Missing values:
Hours_Studied      0
Attendance         1
Assignment_Score   1
Midterm_Score      1
Final_Score         1
dtype: int64
Missing values after filling:
Hours_Studied      0
Attendance         0
Assignment_Score   0
Midterm_Score      0
Final_Score         0
dtype: int64

```

	Hours_Studied	Attendance	Assignment_Score	Midterm_Score	Final_Score	Performance	grid icon
0	1	60.0	55.0	50.0	52.0	Average	
1	2	65.0	58.0	55.0	57.0	Average	
2	3	70.0	60.0	58.0	60.0	Good	
3	4	75.0	65.0	62.0	64.0	Good	
4	5	80.0	68.0	65.0	68.0	Good	

## Exercise 3: Matplotlib Visualization

1. Line plot: Hours\_Studied vs Final\_Score.
2. Histogram of Final\_Score.

```

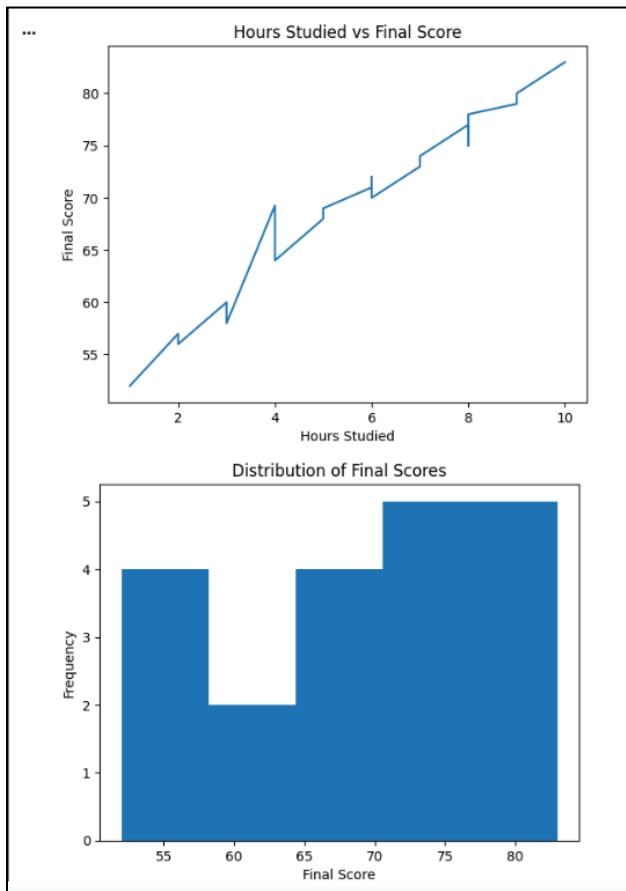
❶ import matplotlib.pyplot as plt

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

plt.figure()
plt.plot(df_sorted['Hours_Studied'], df_sorted['Final_Score'])
plt.xlabel("Hours Studied")
plt.ylabel("Final Score")
plt.title("Hours Studied vs Final Score")
plt.show()

plt.figure()
plt.hist(df['Final_Score'], bins=5)
plt.xlabel("Final Score")
plt.ylabel("Frequency")
plt.title("Distribution of Final Scores")
plt.show()

```



## Exercise 4: Seaborn Visualization

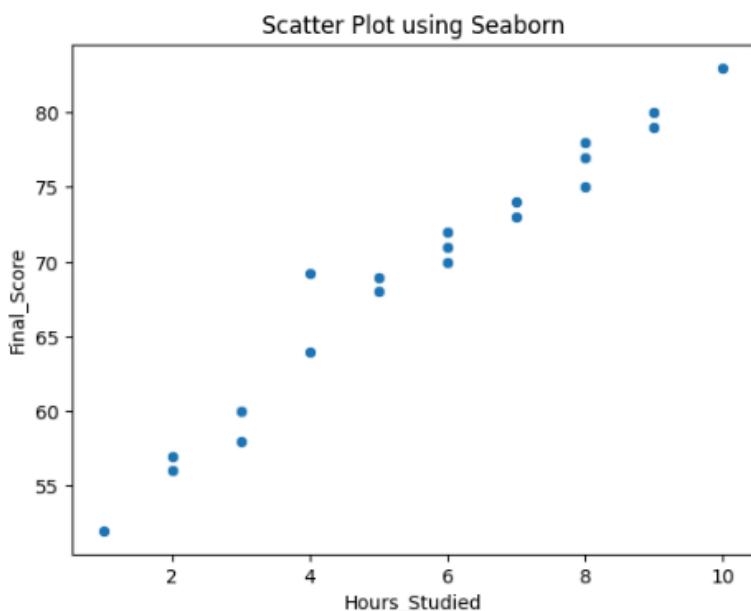
1. Scatter plot using seaborn.
2. Heatmap for correlation analysis.
3. Boxplot for categorical analysis.

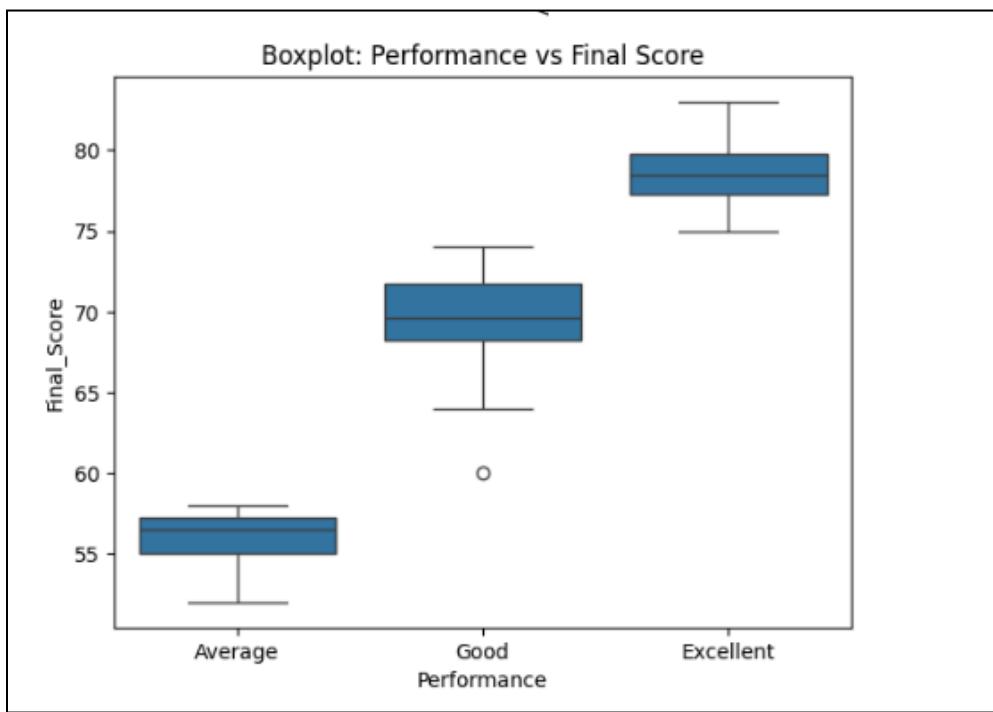
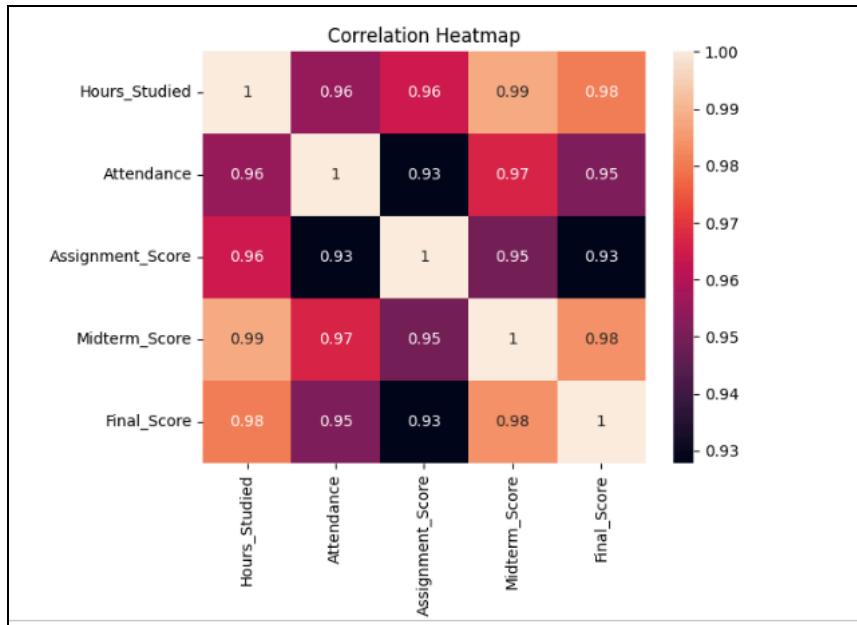
```
import seaborn as sns

plt.figure()
sns.scatterplot(x='Hours_Studied', y='Final_Score', data=df)
plt.title("Scatter Plot using Seaborn")
plt.show()

plt.figure()
correlation = df.corr(numeric_only=True)
sns.heatmap(correlation, annot=True)
plt.title("Correlation Heatmap")
plt.show()

plt.figure()
sns.boxplot(x='Performance', y='Final_Score', data=df)
plt.title("Boxplot: Performance vs Final Score")
plt.show()
```





## **CONCLUSION**

The experiment was successfully conducted using NumPy, Pandas, Matplotlib, and Seaborn libraries for data preprocessing and visualization. Missing values were handled effectively using mean imputation, and statistical measures were computed using NumPy.