AAYUSH PATIL - 56

MLDL EXPERIMENT 0

AIM -

To understand data handling, visualization, and EDA using Python libraries essential for Machine Learning.

THEORY -

In this experiment, Python data analysis libraries such as NumPy, Pandas, Matplotlib, and Seaborn are used to analyze and visualize student data. NumPy is used for numerical operations like calculating mean, median, standard deviation, and normalization. Pandas helps in loading the CSV file, handling data, checking missing values, and creating new columns based on conditions. Matplotlib is used for basic data visualization such as line plots and histograms, while Seaborn provides advanced and attractive statistical visualizations like scatter plots, heatmaps, and boxplots. These libraries together make data analysis easy, efficient, and understandable.

OBSERVATIONS -

Exercise 1: NumPy Basics

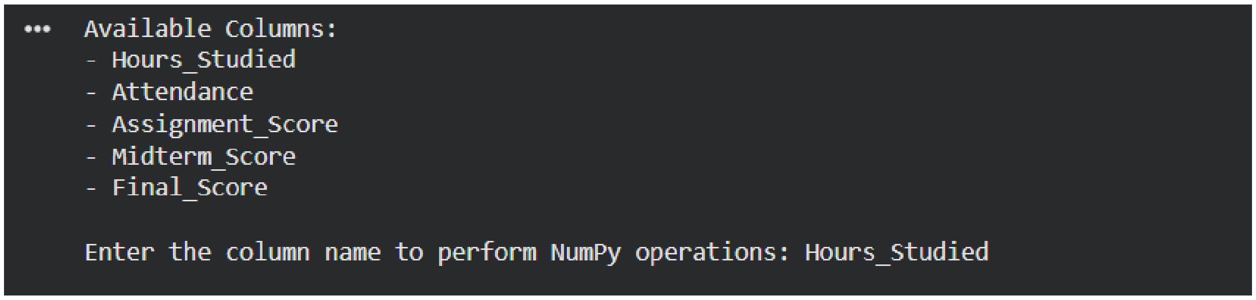
1. Load Final_Score as NumPy array.
2. Compute mean, median, and standard deviation.
3. Perform Min-Max normalization.

CODE :

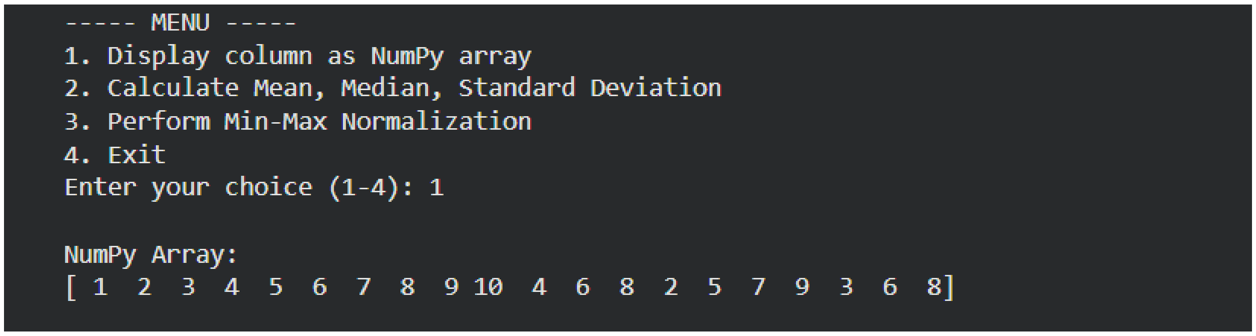
```
import pandas as pd
import numpy as np

# Load dataset
df = pd.read_csv("/content/student_performance.csv")

# Display available numeric columns
print("Available Columns:")
for col in df.columns:
    print("-", col)

# User selects column
column = input("\nEnter the column name to perform NumPy operations: ")

# Convert selected column to NumPy array
data = df[column].values

while True:
    print("\n----- MENU -----")
    print("1. Display column as NumPy array")
    print("2. Calculate Mean, Median, Standard Deviation")
    print("3. Perform Min-Max Normalization")
    print("4. Exit")

    choice = int(input("Enter your choice (1-4): "))
```

```python
if choice == 1:
    print("\nNumPy Array:")
    print(data)

elif choice == 2:
    print("\nStatistical Measures:")
    print("Mean:", np.mean(data))
    print("Median:", np.median(data))
    print("Standard Deviation:", np.std(data))

elif choice == 3:
    normalized_data = (data - np.min(data)) / (np.max(data) - np.min(data))
    print("\nMin-Max Normalized Data:")
    print(normalized_data)

elif choice == 4:
    print("\nProgram terminated.")
    break

else:
    print("\nInvalid choice. Please enter a valid option.")
```

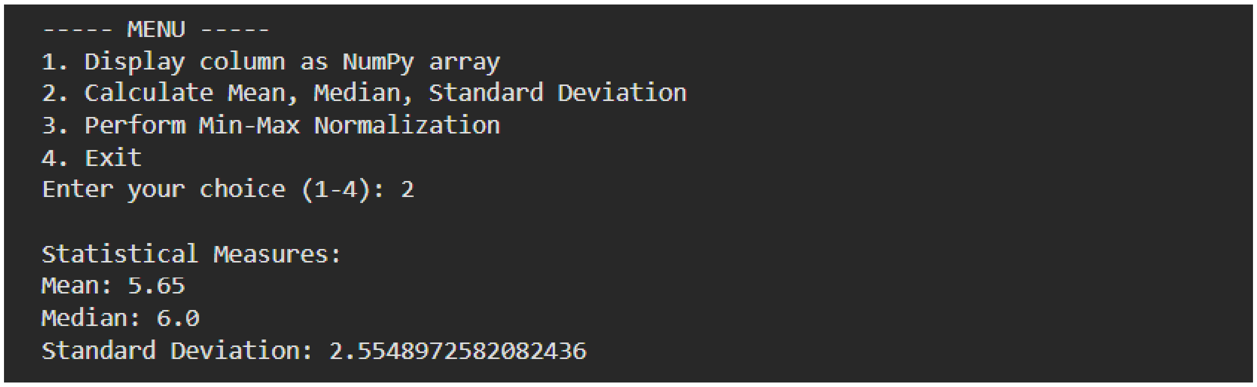
OUTPUT :

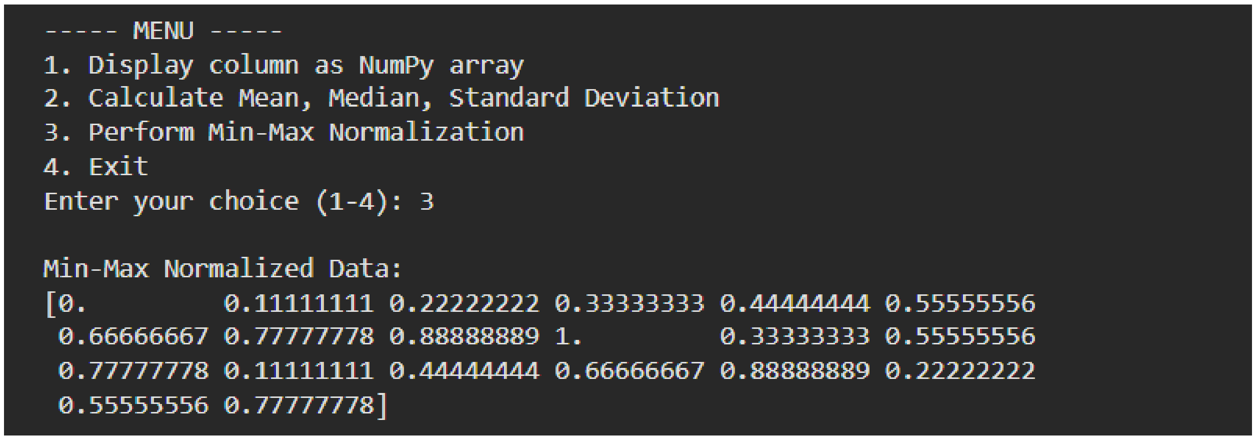


```
Available Columns:
- Hours_Studied
- Attendance
- Assignment_Score
- Midterm_Score
- Final_Score

Enter the column name to perform NumPy operations: Hours_Studied
```



```
----- MENU -----
1. Display column as NumPy array
2. Calculate Mean, Median, Standard Deviation
3. Perform Min-Max Normalization
4. Exit
Enter your choice (1-4): 1

NumPy Array:
[ 1  2  3  4  5  6  7  8  9 10  4  6  8  2  5  7  9  3  6  8]
```


```
----- MENU -----
1. Display column as NumPy array
2. Calculate Mean, Median, Standard Deviation
3. Perform Min-Max Normalization
4. Exit
Enter your choice (1-4): 2

Statistical Measures:
Mean: 5.65
Median: 6.0
Standard Deviation: 2.5548972582082436
```


```
----- MENU -----
1. Display column as NumPy array
2. Calculate Mean, Median, Standard Deviation
3. Perform Min-Max Normalization
4. Exit
Enter your choice (1-4): 3

Min-Max Normalized Data:
[0.         0.11111111 0.22222222 0.33333333 0.44444444 0.55555556
 0.66666667 0.77777778 0.88888889 1.         0.33333333 0.55555556
 0.77777778 0.11111111 0.44444444 0.66666667 0.88888889 0.22222222
 0.55555556 0.77777778]
```

Exercise 2: Pandas Data Handling

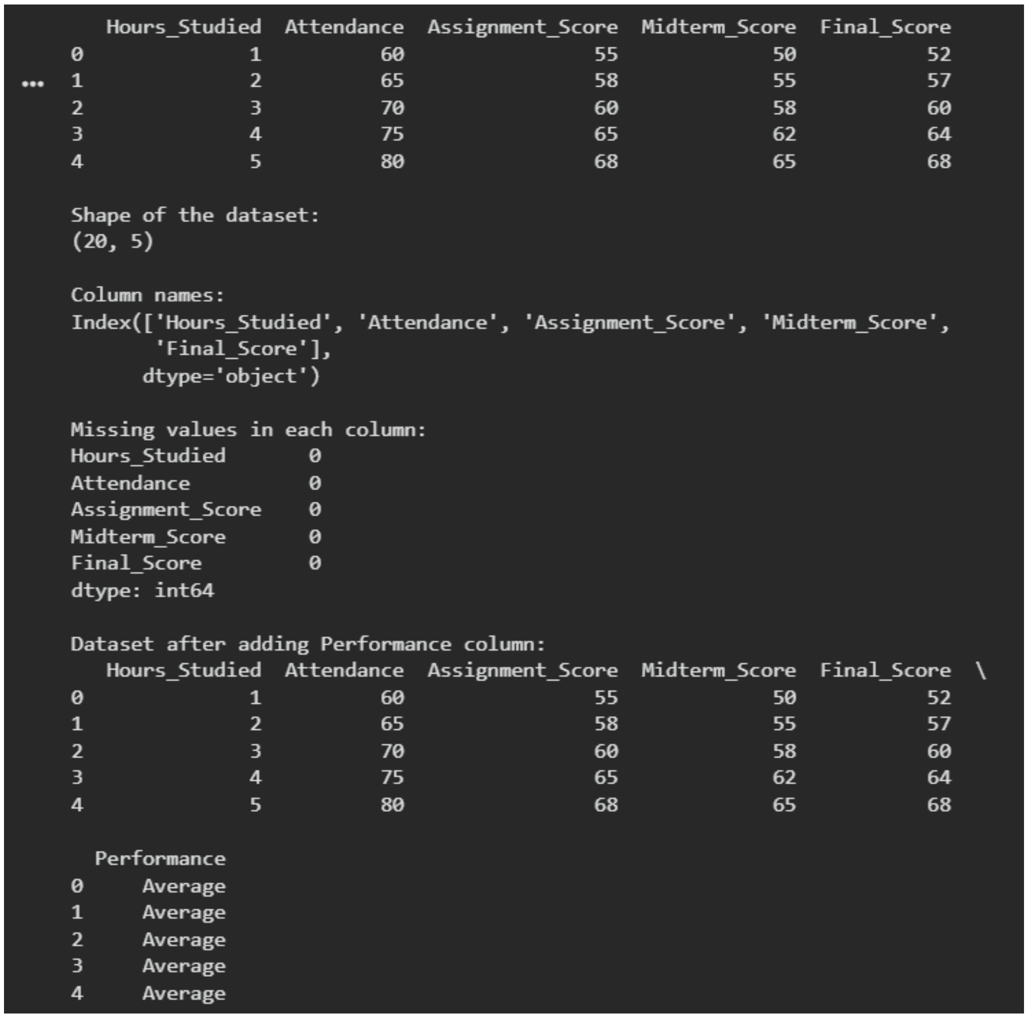
1. Load CSV file using Pandas.
2. Check shape, columns, and missing values.
3. Create a Performance label based on Final_Score.

CODE :

```
import pandas as pd

# 1. Load CSV file
df = pd.read_csv("/content/student_performance.csv")

# Display first 5 rows
print("First 5 rows of the dataset:")
print(df.head())

# 2. Check shape, columns, and missing values
print("\nShape of the dataset:")
print(df.shape)

print("\nColumn names:")
print(df.columns)

print("\nMissing values in each column:")
print(df.isnull().sum())
```

```
# 3. Create Performance label based on Final_Score
def performance_label(score):
    if score >= 75:
        return "Excellent"
    elif score >= 50:
        return "Average"
    else:
        return "Poor"

# Add new column
df["Performance"] = df["Final_Score"].apply(performance_label)

# Display updated dataset
print("\nDataset after adding Performance column:")
print(df.head())
```

OUTPUT :



```
   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

Shape of the dataset:
(20, 5)

Column names:
Index(['Hours_Studied', 'Attendance', 'Assignment_Score', 'Midterm_Score',
       'Final_Score'],
      dtype='object')

Missing values in each column:
Hours_Studied       0
Attendance          0
Assignment_Score    0
Midterm_Score       0
Final_Score         0
dtype: int64

Dataset after adding 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
```

Exercise 3: Matplotlib Visualization

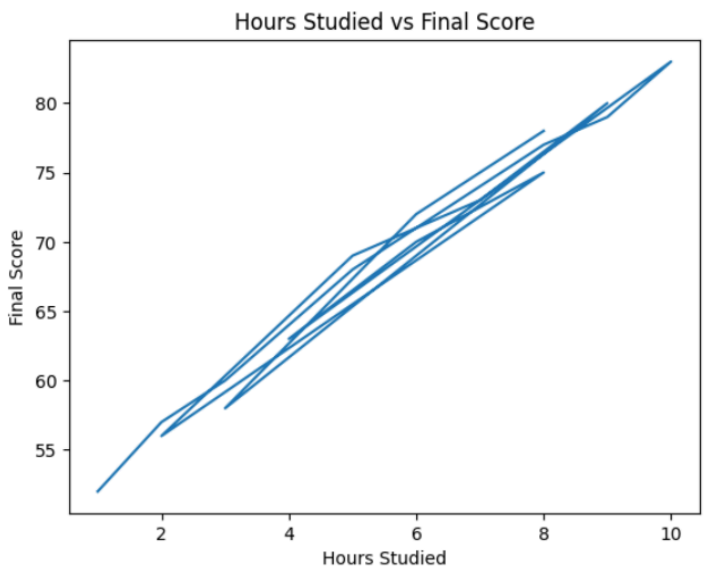
1. Line plot: Hours_Studied vs Final_Score.
2. Histogram of Final_Score.

CODE :

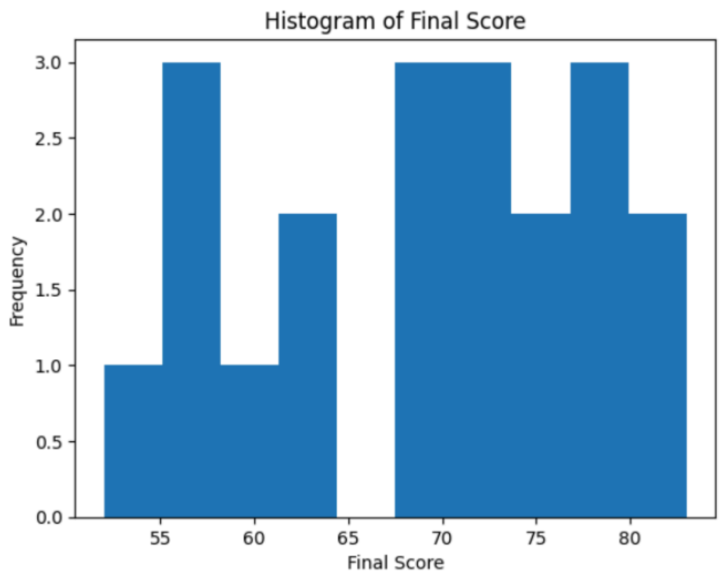
```python
import pandas as pd
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv("/content/student_performance.csv")

# 1. Line Plot: Hours_Studied vs Final_Score
plt.figure()
plt.plot(df["Hours_Studied"], df["Final_Score"])
plt.xlabel("Hours Studied")
plt.ylabel("Final Score")
plt.title("Hours Studied vs Final Score")
plt.show()

# 2. Histogram of Final_Score
plt.figure()
plt.hist(df["Final_Score"])
plt.xlabel("Final Score")
plt.ylabel("Frequency")
plt.title("Histogram of Final Score")
plt.show()
```

OUTPUT :

LINE PLOT



HISTOGRAM



Exercise 4: Seaborn Visualization

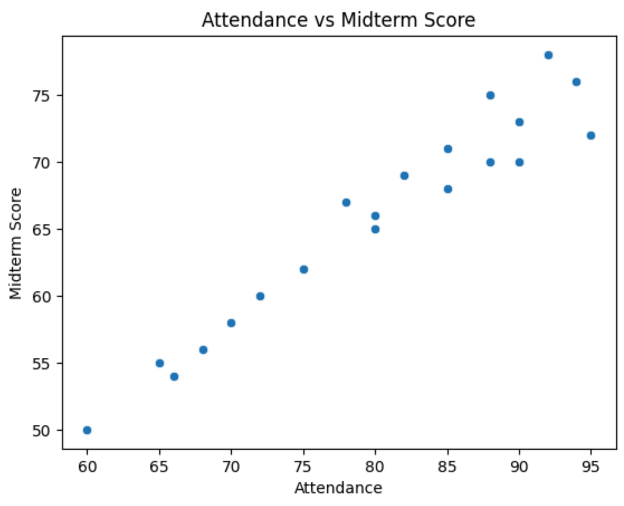
1. Scatter plot using seaborn.
2. Heatmap for correlation analysis.
3. Boxplot for categorical analysis.

CODE :

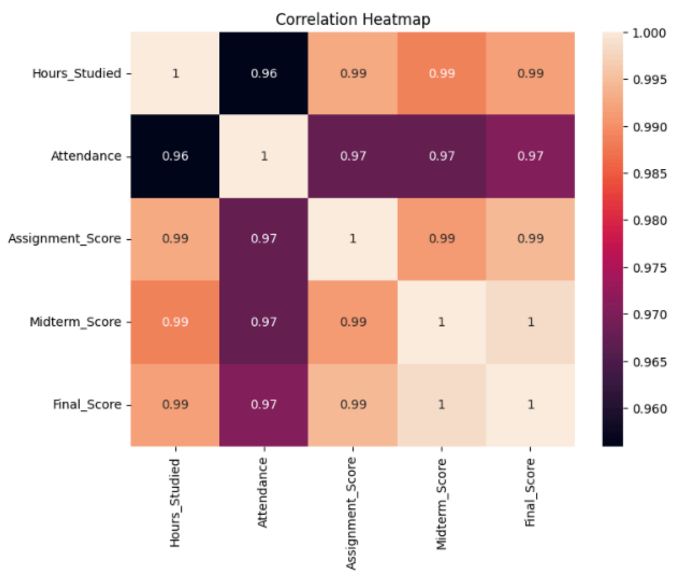
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load dataset using your exact path
df = pd.read_csv("/content/student_performance.csv")

# Create categorical column for boxplot (inside this code to avoid errors)
def performance_label(score):
    if score >= 75:
        return "Excellent"
    elif score >= 50:
        return "Average"
    else:
        return "Poor"

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

# ---------------- 1. Scatter Plot ----------------
plt.figure()
sns.scatterplot(x="Attendance", y="Midterm_Score", data=df)
plt.title("Attendance vs Midterm Score")
plt.xlabel("Attendance")
plt.ylabel("Midterm Score")
plt.show()

# ---------------- 2. Correlation Heatmap ----------------
plt.figure(figsize=(8,6))
sns.heatmap(df.select_dtypes(include="number").corr(), annot=True)
plt.title("Correlation Heatmap")
plt.show()

# ---------------- 3. Boxplot (Categorical Analysis) ----------------
plt.figure()
sns.boxplot(x="Performance", y="Assignment_Score", data=df)
plt.title("Performance vs Assignment Score")
plt.xlabel("Performance Category")
plt.ylabel("Assignment Score")
plt.show()
```

OUTPUT :

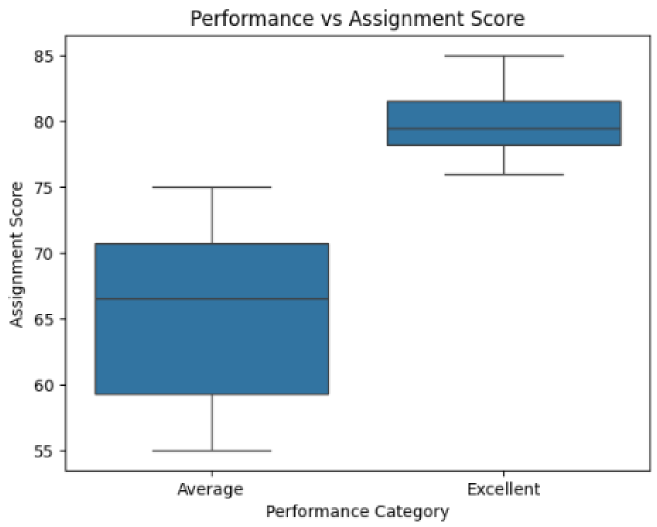
SCATTERPLOT



HEATMAP



BOXPLOT



CONCLUSION -

In this experiment, Seaborn was used to visualize student performance data effectively. The scatter plot helped in understanding the relationship between attendance and midterm scores, the heatmap showed correlations among different numerical attributes, and the boxplot compared assignment scores across performance categories. Overall, the experiment demonstrated how data visualization techniques can simplify data analysis and help in identifying patterns and relationships in student performance data.