EXPERIMENT 0

D15A-57

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

Dataset: student_performance.csv

Columns: Hours_Studied, Attendance, Assignment_Score, Midterm_Score, Final_Score

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 numpy as np
import pandas as pd

df = pd.read_csv("student_performance.csv")
final_score_array = df['Final_Score'].to_numpy()

print("1. Final_Score array:", final_score_array)

mean_score = np.mean(final_score_array)
median_score = np.median(final_score_array)
std_dev_score = np.std(final_score_array)

print("\n2. Computed Statistics:")
print(f"   Mean: {mean_score:.2f}")
print(f"   Median: {median_score:.2f}")
print(f"   Standard Deviation: {std_dev_score:.2f}")

min_score = np.min(final_score_array)
max_score = np.max(final_score_array)


if (max_score - min_score) == 0:
    normalized_scores = np.zeros_like(final_score_array)
else:
    normalized_scores = (final_score_array - min_score) / (max_score - min_score)

print("\n3. Min-Max Normalized Final_Score:", normalized_scores)
```

Output:

```
1. Final_Score array: [52 57 60 64 68 71 74 77 79 83 63 70 75 56 69 73 80 58 72 78]

2. Computed Statistics:
   Mean: 68.95
   Median: 70.50
   Standard Deviation: 8.71

3. Min-Max Normalized Final_Score: [0.         0.16129032 0.25806452 0.38709677 0.51612903 0.61290323
 0.70967742 0.80645161 0.87096774 1.         0.35483871 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.

Code:

```
import numpy as np
import pandas as pd

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

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


def get_performance_label(score):
    if score >= 75:
        return 'Excellent'
    elif score >= 60:
        return 'Good'
    else:
        return 'Bad'

df['Performance_Label'] = df['Final_Score'].apply(get_performance_label)
print("\n3. DataFrame with 'Performance_Label':")
print(df[['Final_Score', 'Performance_Label']].head())
```

Output:

```
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

3. DataFrame with 'Performance_Label':
   Final_Score Performance_Label
0           52               Bad
1           57               Bad
2           60              Good
3           64              Good
4           68              Good
```

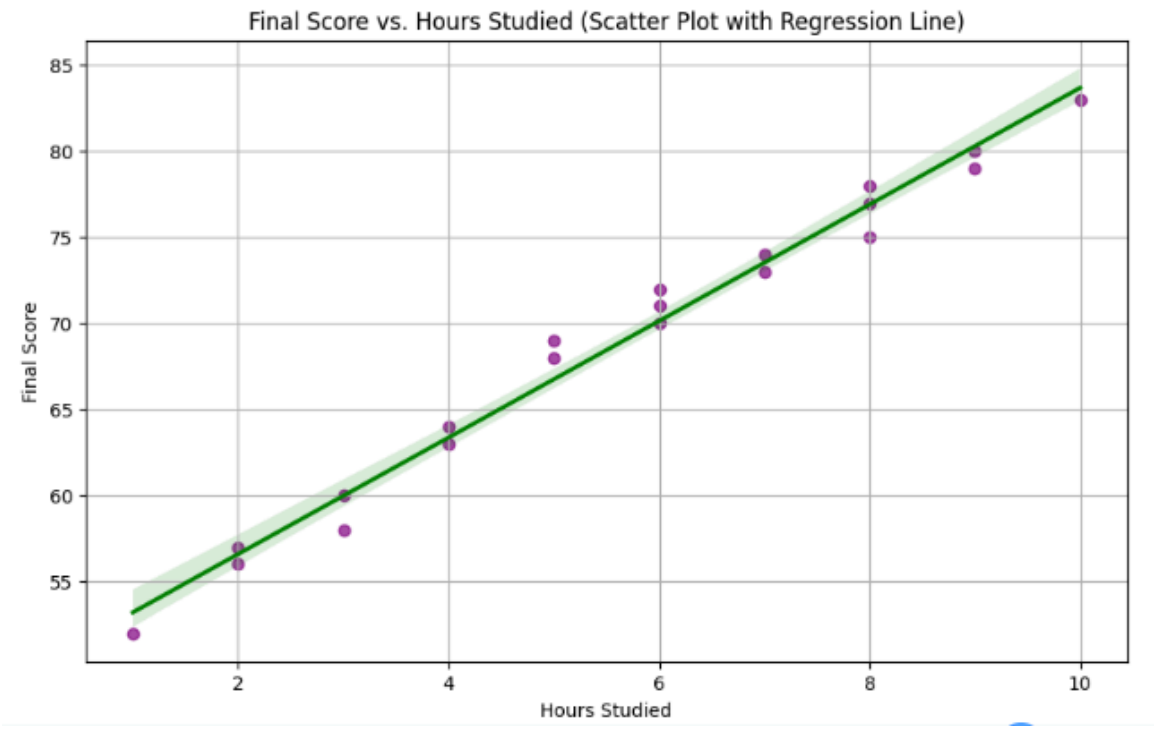
Exercise 3: Matplotlib Visualization

1. Line plot: Hours_Studied vs Final_Score.

Code:

```
import matplotlib.pyplot as plt
import numpy as np
df = pd.read_csv("student_performance.csv")

x = np.linspace(0, 10, 100)
y = np.sin(x)


plt.figure(figsize=(8, 4))
plt.plot(x, y, label='sin(x)', color='green', linestyle='--')
plt.title('Simple Sine Wave Plot')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.grid(True)
plt.legend()
plt.show()
```

Output:

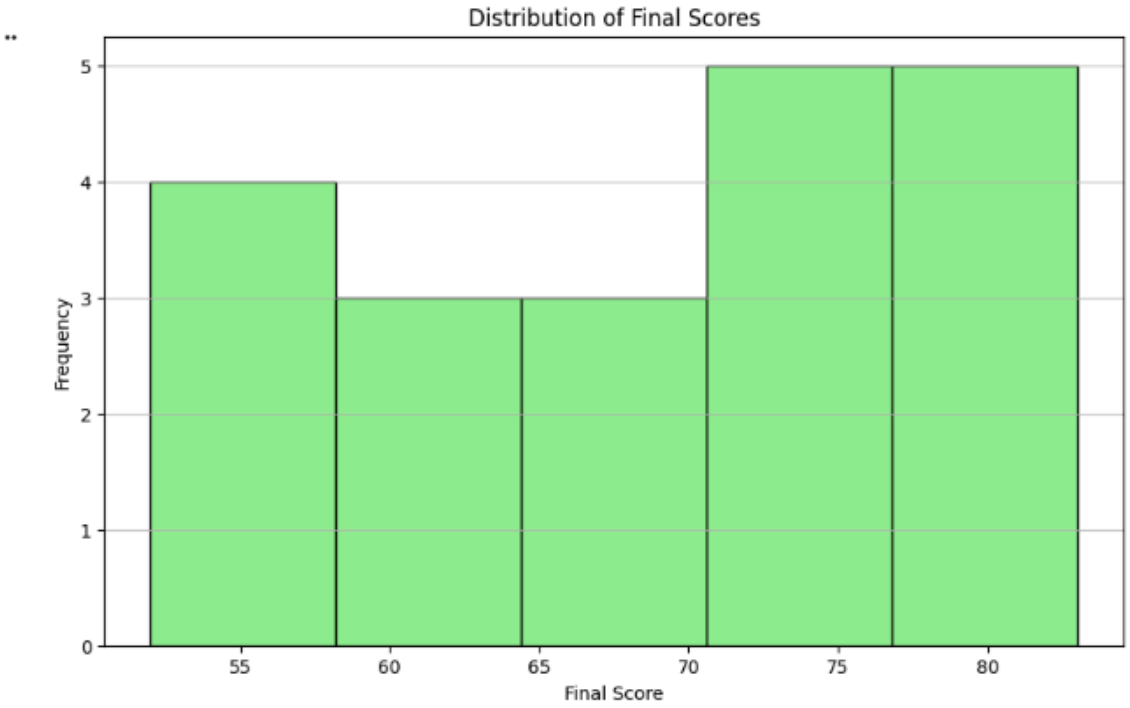


2. Histogram of Final_Score

Code:

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

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

plt.figure(figsize=(10, 6))
sns.regplot(x='Hours_Studied', y='Final_Score', data=df, ci=95, scatter_kws={'color': 'purple', 'alpha': 0.7}, line_kws={'color': 'green'})
plt.title('Final Score vs. Hours Studied (Scatter Plot with Regression Line)')
plt.xlabel('Hours Studied')
plt.ylabel('Final Score')
plt.grid(True)
plt.show()
```

Output:



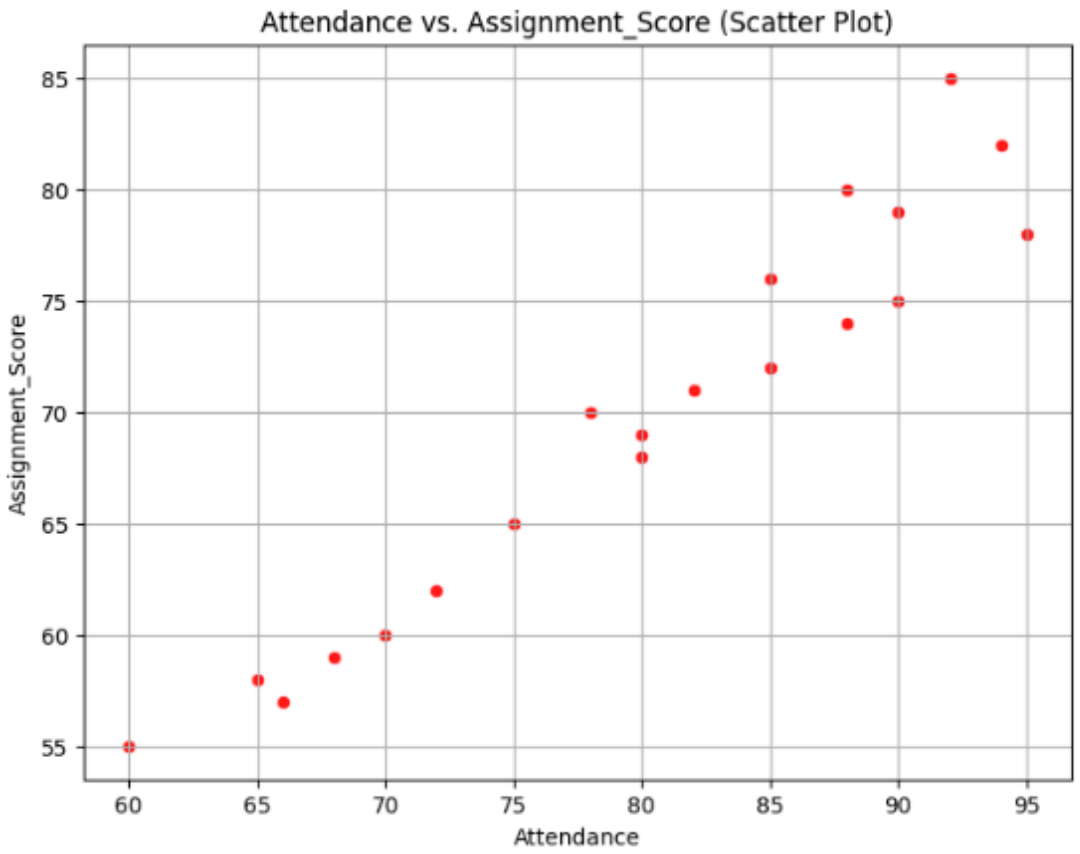
Exercise 4: Seaborn Visualization

1. Scatter plot using seaborn

Code:

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

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

plt.figure(figsize=(8, 6))
sns.scatterplot(x='Attendance', y='Assignment_Score', data=df, color='red', alpha=0.9)
plt.title('Attendance vs. Assignment_Score (Scatter Plot)')
plt.xlabel('Attendance')
plt.ylabel('Assignment_Score')

plt.grid(True)
plt.show()
```

Output:

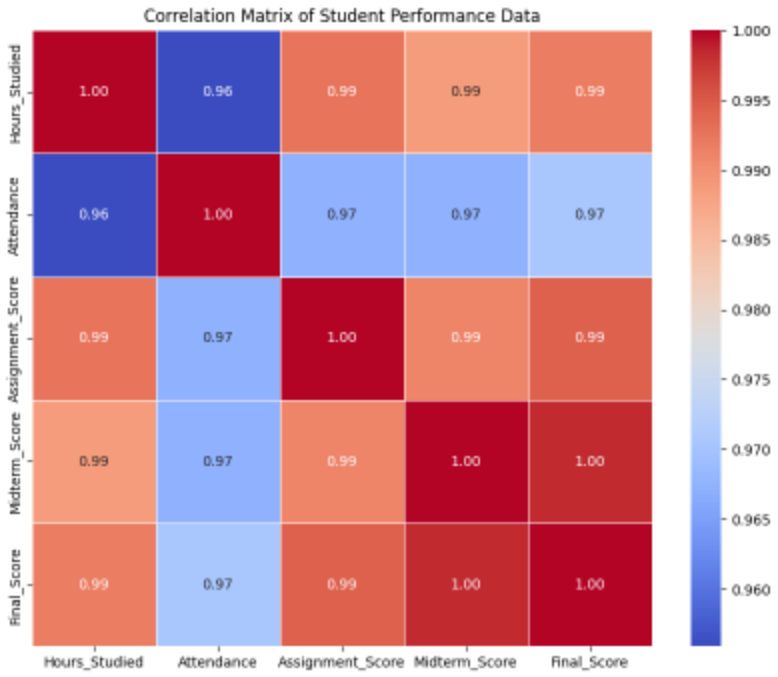


2. Heatmap for correlation analysis

Code:

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

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

correlation_matrix = df.corr(numeric_only=True)

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=.5)
plt.title('Correlation Matrix of Student Performance Data')
plt.show()
```

Output:

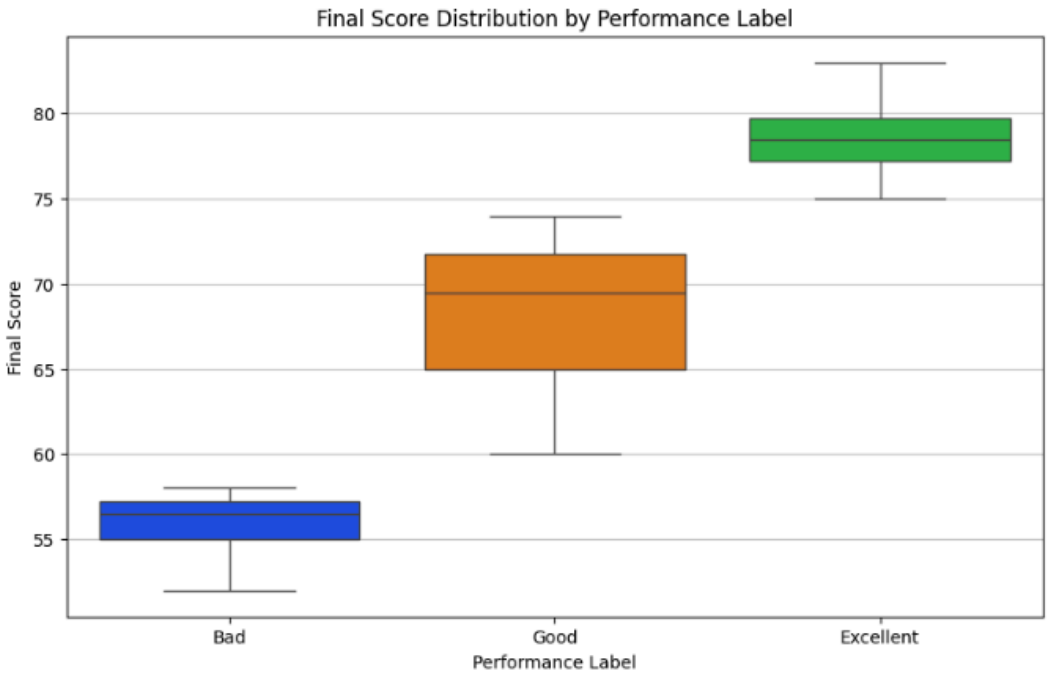


3. Boxplot for categorical analysis

Code:

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

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

def get_performance_label(score):
    if score >= 75:
        return 'Excellent'
    elif score >= 60:
        return 'Good'
    else:
        return 'Bad'

df['Performance_Label'] = df['Final_Score'].apply(get_performance_label)

plt.figure(figsize=(10, 6))
sns.boxplot(x='Performance_Label', y='Final_Score', data=df, hue='Performance_Label', palette='bright', legend=False)
plt.title('Final Score Distribution by Performance Label')
plt.xlabel('Performance Label')
plt.ylabel('Final Score')
plt.grid(axis='y', alpha=0.75)
plt.show()
```

Output:


Final Score Distribution by Performance Label
Final Score
80
75
70
65
60
55
Bad
Good
Excellent
Performance Label