Week 1: Basic Data Analytics Using NumPy (8 Programs)

Aim

To learn basic data analytics using NumPy for numerical computations and array manipulations.

Programs

Program 1: Basic Array Operations

Procedure: Create a NumPy array and perform basic arithmetic operations.

Code:

```
import numpy as np

# Create a NumPy array
arr = np.array([1, 2, 3, 4, 5])
print("Array:", arr)

# Perform arithmetic operations
print("Add 5:", arr + 5)
print("Multiply by 2:", arr * 2)
```

Expected Result:

```
Array: [1 2 3 4 5]
Add 5: [ 6  7  8  9 10]
Multiply by 2: [ 2  4  6  8 10]
```

Program 2: Array Statistics

Procedure: Calculate mean, median, and standard deviation.

Code:

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
print("Mean:", np.mean(arr))
print("Median:", np.median(arr))
print("Standard Deviation:", np.std(arr))
```

Expected Result:

```
Mean: 5.5
Median: 5.5
Standard Deviation: 2.8722813232690143
```

Program 3: Reshaping Arrays

Procedure: Reshape a 1D array into a 2D array.

Code:

```
arr = np.arange(1, 13)
reshaped_arr = arr.reshape(3, 4)
print("Reshaped Array:\n", reshaped_arr)
```

Expected Result:

```
Reshaped Array:
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

Program 4: Array Indexing

Procedure: Demonstrate indexing and slicing in NumPy arrays.

Code:

```
arr = np.array([10, 20, 30, 40, 50])
print("First Element:", arr[0])
print("Last Element:", arr[-1])
print("Slice (1 to 3):", arr[1:4])
```

Expected Result:

```
First Element: 10
Last Element: 50
Slice (1 to 3): [20 30 40]
```

Program 5: Array Concatenation

Procedure: Concatenate two arrays.

Code:

```
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
concatenated = np.concatenate((arr1, arr2))
print("Concatenated Array:", concatenated)
```

Expected Result:

```
Concatenated Array: [1 2 3 4 5 6]
```

Program 6: Boolean Indexing

Procedure: Use boolean conditions to filter arrays.

Code:

```
arr = np.array([1, 2, 3, 4, 5])
filtered_arr = arr[arr > 2]
print("Filtered Array (greater than 2):", filtered_arr)
```

Expected Result:

```
Filtered Array (greater than 2): [3 4 5]
```

Program 7: Dot Product

Procedure: Calculate the dot product of two arrays.

Code:

```
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
dot_product = np.dot(arr1, arr2)
print("Dot Product:", dot_product)
```

Expected Result:

```
Dot Product: 32
```

Program 8: Linear Algebra Operations

Procedure: Perform matrix operations like inverse and determinant.

Code:

```
matrix = np.array([[1, 2], [3, 4]])
determinant = np.linalg.det(matrix)
inverse = np.linalg.inv(matrix)
print("Determinant:", determinant)
print("Inverse:\n", inverse)
```

Expected Result:

```
Determinant: -2.0
```

```
Inverse:
[[-2.   1. ]
 [ 1.5 -0.5]]
```

Week 2: Basics of Data Wrangling Features (10 Programs)

Aim

To explore basic data wrangling features using Pandas for data manipulation.

Programs

Program 1: Creating a DataFrame

Procedure: Create a DataFrame from a dictionary.

Code:

```
import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [24, 27, 22],
        'City': ['New York', 'Los Angeles', 'Chicago']}
df = pd.DataFrame(data)
print(df)
```

Expected Result:

```
      Name  Age         City
0    Alice   24     New York
1      Bob   27  Los Angeles
2  Charlie   22      Chicago
```

Program 2: Reading CSV Files

Procedure: Read a CSV file into a DataFrame.

Code:

```
df = pd.read_csv('sample.csv')  # Assuming 'sample.csv' exists
print(df.head())
```

Expected Result: (Displays first 5 rows of the dataset)

Program 3: Data Cleaning

Procedure: Clean data by removing missing values.

Code:

```
df = pd.DataFrame({'A': [1, 2, None, 4], 'B': [None, 2, 3, 4]})
df_cleaned = df.dropna()
print(df_cleaned)
```

Expected Result:

```
     A    B
1  2.0  2.0
3  4.0  4.0
```

Program 4: Filtering Data

Procedure: Filter rows based on a condition.

Code:

```
df = pd.DataFrame({'A': [1, 2, 3, 4], 'B': [5, 6, 7, 8]})
filtered_df = df[df['A'] > 2]
print(filtered_df)
```

Expected Result:

```
   A  B
2  3  7
3  4  8
```

Program 5: Grouping Data

Procedure: Group data and calculate aggregates.

Code:

```
df = pd.DataFrame({'A': ['foo', 'bar', 'foo', 'bar'],
                   'B': [1, 2, 3, 4]})
grouped = df.groupby('A').sum()
print(grouped)
```

Expected Result:

```
     B
A
bar  6
```

```
foo  4
```

Program 6: Merging DataFrames

Procedure: Merge two DataFrames.

Code:

```
df1 = pd.DataFrame({'A': ['foo', 'bar'], 'B': [1, 2]})
df2 = pd.DataFrame({'A': ['foo', 'bar'], 'C': [3, 4]})
merged_df = pd.merge(df1, df2, on='A')
print(merged_df)
```

Expected Result:

```
   A  B  C
0 foo  1  3
1 bar  2  4
```

Program 7: Pivot Tables

Procedure: Create a pivot table from data.

Code:

```
df = pd.DataFrame({'A': ['foo', 'bar', 'foo', 'bar'],
                   'B': [1, 2, 3, 4],
                   'C': [5, 6, 7, 8]})
pivot_table = df.pivot_table(values='C', index='A', columns='B', aggfunc='sum')
print(pivot_table)
```

Expected Result:

```
    1    2    3    4
A
bar NaN  6.0 NaN  8.0
foo  5.0 NaN  7.0 NaN
```

Program 8: DataFrame Transformation

Procedure: Apply transformations to a DataFrame.

Code:

```
df = pd.DataFrame({'A': [1, 2, 3, 4], 'B': [5, 6, 7, 8]})
transformed_df = df.transform(lambda x: x ** 2)
```

```
print(transformed_df)
```

Expected Result:

```
   A   B
0  1  25
1  4  36
2  9  49
3 16  64
```

Program 9: Handling Duplicates

Procedure: Identify and remove duplicates from a DataFrame.

Code:

```
df = pd.DataFrame({'A': [1, 1, 2, 3], 'B': [4, 4, 5, 6]})
df_no_duplicates = df.drop_duplicates()
print(df_no_duplicates)
```

Expected Result:

```
   A  B
0  1  4
2  2  5
3  3  6
```

Program 10: Saving DataFrames to CSV

Procedure: Save a DataFrame to a CSV file.

Code:

```
df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
df.to_csv('output.csv', index=False)
print("DataFrame saved to output.csv")
```

Expected Result:

```
DataFrame saved to output.csv
```

Week 3: Pandas Time Series Analysis (5 Programs)

Aim

To perform time series analysis using Pandas for analyzing time-based data.

Programs

Program 1: Creating Time Series Data

Procedure: Create a time series DataFrame.

Code:

```
dates = pd.date_range(start='2023-01-01', periods=5, freq='D')
data = [1, 2, 3, 4, 5]
ts = pd.Series(data, index=dates)
print(ts)
```

Expected Result:

```
2023-01-01    1
2023-01-02    2
2023-01-03    3
2023-01-04    4
2023-01-05    5
Freq: D, dtype: int64
```

Program 2: Time Series Indexing

Procedure: Index a time series using specific dates.

Code:

```
ts = pd.Series([1, 2, 3, 4, 5], index=pd.date_range('2023-01-01', periods=5))
print("Data on 2023-01-03:", ts['2023-01-03'])
```

Expected Result:

```
Data on 2023-01-03: 3
```

Program 3: Resampling Time Series Data

Procedure: Resample time series data to a different frequency.

Code:

```
ts = pd.Series([1, 2, 3, 4, 5], index=pd.date_range('2023-01-01', periods=5, freq='D'))
resampled_ts = ts.resample('2D').sum()
print(resampled_ts)
```

Expected Result:

```
2023-01-01    3
2023-01-03    7
2023-01-05    5
Freq: 2D, dtype: int64
```

Program 4: Shifting Time Series Data

Procedure: Shift time series data forward or backward.

Code:

```
ts = pd.Series([1, 2, 3, 4, 5], index=pd.date_range('2023-01-01', periods=5))
shifted_ts = ts.shift(1)
print(shifted_ts)
```

Expected Result:

```
2023-01-01    NaN
2023-01-02    1.0
2023-01-03    2.0
2023-01-04    3.0
2023-01-05    4.0
Freq: D, dtype: float64
```

Program 5: Rolling Window Calculation

Procedure: Calculate rolling mean on time series data.

Code:

```
ts = pd.Series([1, 2, 3, 4, 5], index=pd.date_range('2023-01-01', periods=5))
rolling_mean = ts.rolling(window=3).mean()
print(rolling_mean)
```

Expected Result:

```
2023-01-01    NaN
2023-01-02    NaN
2023-01-03    2.0
2023-01-04    3.0
2023-01-05    4.0
Freq: D, dtype: float64
```

Week 4: Data Visualization (Different Types of Plotting)

Aim

To explore various plotting techniques using Matplotlib and Seaborn for data visualization.

Programs

Program 1: Line Plot

Procedure: Create a simple line plot.

Code:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]
plt.plot(x, y)
plt.title("Line Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

Expected Result: A line plot showing the relationship between x and y.

Program 2: Bar Plot

Procedure: Create a bar plot.

Code:

```
categories = ['A', 'B', 'C', 'D']
values = [4, 7, 1, 8]
plt.bar(categories, values)
plt.title("Bar Plot")
plt.xlabel("Categories")
plt.ylabel("Values")
plt.show()
```

Expected Result: A bar plot with categories on the x-axis and their corresponding values on the y-axis.

Program 3: Histogram

Procedure: Create a histogram to display frequency distribution.

Code:

```
data = [1, 2, 2, 3, 3, 3, 4, 5, 5, 5, 5]
plt.hist(data, bins=5, alpha=0.7)
plt.title("Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```

Expected Result: A histogram representing the frequency of data values.

Program 4: Scatter Plot

Procedure: Create a scatter plot.

Code:

```
x = [5, 7, 8, 5, 6]
y = [7, 8, 9, 6, 5]
plt.scatter(x, y)
plt.title("Scatter Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

Expected Result: A scatter plot showing the distribution of points.

Program 5: Box Plot

Procedure: Create a box plot to show data distribution.

Code:

```
import numpy as np
import matplotlib.pyplot as plt

data = [np.random.normal(0, std, 100) for std in range(1, 4)]
plt.boxplot(data, vert=True, patch_artist=True)
plt.title("Box Plot")
plt.xlabel("Distribution")
plt.ylabel("Values")
plt.xticks([1, 2, 3], ['Std 1', 'Std 2', 'Std 3'])
plt.show()
```

Expected Result: A box plot showing the distribution of three datasets with different standard deviations.

Program 6: Area Plot

Procedure: Create an area plot.

Code:

```
x = np.arange(1, 6)
y1 = [1, 2, 3, 4, 5]
y2 = [2, 3, 4, 5, 6]
plt.fill_between(x, y1, y2, color='skyblue', alpha=0.4)
plt.plot(x, y1, color='Slateblue', alpha=0.6, linewidth=2)
plt.plot(x, y2, color='Slateblue', alpha=0.6, linewidth=2)
plt.title("Area Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

Expected Result: An area plot filling the space between two lines.

Program 7: Heatmap

Procedure: Create a heatmap to represent data values.

Code:

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

data = np.random.rand(10, 12)
sns.heatmap(data, annot=True)
plt.title("Heatmap")
plt.show()
```

Expected Result: A heatmap showing a 10x12 grid of random values with annotations.

Program 8: Pair Plot

Procedure: Create a pair plot for pairwise relationships in a dataset.

Code:

```
import seaborn as sns
import pandas as pd

df = pd.DataFrame({
    'A': np.random.rand(50),
    'B': np.random.rand(50),
    'C': np.random.rand(50)
})
sns.pairplot(df)
plt.title("Pair Plot")
plt.show()
```

Expected Result: A pair plot showing scatter plots and histograms for the combinations of three features.

Program 9: Violin Plot

Procedure: Create a violin plot to visualize the distribution of data across different categories.

Code:

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

data = sns.load_dataset("tips")
sns.violinplot(x='day', y='total_bill', data=data)
plt.title("Violin Plot")
plt.show()
```

Expected Result: A violin plot representing the distribution of total bills for different days.

Program 10: Subplots

Procedure: Create multiple plots in a single figure.

Code:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)

fig, axs = plt.subplots(2, 1)
axs[0].plot(x, y1, 'r')
axs[0].set_title('Sine Wave')
axs[1].plot(x, y2, 'b')
axs[1].set_title('Cosine Wave')
```

```
plt.tight_layout()
plt.show()
```

Expected Result: A figure with two subplots showing the sine and cosine waves.

Week 5: Data Visualization Using Numpy (3 Programs)

Aim

To visualize data using Numpy arrays with Matplotlib.

Programs

Program 1: Numpy Histogram

Procedure: Create a histogram from Numpy data.

Code:

```
import numpy as np
import matplotlib.pyplot as plt

data = np.random.randn(1000)
plt.hist(data, bins=30, alpha=0.5, color='blue')
plt.title("Histogram from Numpy Data")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```

Expected Result: A histogram showing the frequency distribution of random data.

Program 2: Numpy Scatter Plot

Procedure: Create a scatter plot using Numpy arrays.

Code:

```
import numpy as np
import matplotlib.pyplot as plt

x = np.random.rand(50)
y = np.random.rand(50)
plt.scatter(x, y)
plt.title("Scatter Plot from Numpy Data")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

Expected Result: A scatter plot showing random points.

Program 3: Numpy Line Plot

Procedure: Create a line plot using Numpy data.

Code:

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 100)
y = np.sin(x)
plt.plot(x, y)
plt.title("Line Plot from Numpy Data")
plt.xlabel("X-axis")
plt.ylabel("Sine Values")
plt.show()
```

Expected Result: A line plot showing the sine function.

Week 6: Data Visualization Using Pandas (3 Programs)

Aim

To perform data visualization using Pandas DataFrames.

Programs

Program 1: Bar Plot with Pandas

Procedure: Create a bar plot directly from a DataFrame.

Code:

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

df = pd.DataFrame({
    'Category': ['A', 'B', 'C', 'D'],
    'Values': [4, 7, 1, 8]
})
df.plot(kind='bar', x='Category', y='Values', color='purple')
plt.title("Bar Plot from Pandas DataFrame")
plt.show()
```

Expected Result: A bar plot showing the values associated with each category.

Program 2: Line Plot with Pandas

Procedure: Create a line plot directly from a DataFrame.

Code:

```
df = pd.DataFrame({
    'X': [1, 2, 3, 4],
    'Y': [10, 20, 25, 30]
})
df.plot(kind='line', x='X', y='Y', marker='o')
plt.title("Line Plot from Pandas DataFrame")
plt.show()
```

Expected Result: A line plot displaying the relationship between X and Y.

Program 3: Pie Chart with Pandas

Procedure: Create a pie chart from a DataFrame.

Code:

```
df = pd.DataFrame({
    'Categories': ['A', 'B', 'C', 'D'],
    'Values': [20, 30, 25, 25]
})
df.plot.pie(y='Values', labels=df['Categories'], autopct='%1.1f%%', startangle=90)
plt.title("Pie Chart from Pandas DataFrame")
plt.ylabel('')
plt.show()
```

Expected Result: A pie chart showing the percentage distribution of categories.

Week 7: How to Create CSV File and Save and Load

Aim

To learn how to create, save, and load CSV files using Pandas.

Programs

Program 1: Creating and Saving a CSV File

Procedure: Create a DataFrame and save it as a CSV file.

Code:

```
df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['New York', 'Los Angeles', 'Chicago']
})
df.to_csv('people.csv', index=False)
print("CSV file 'people.csv' created successfully.")
```

Expected Result:

```
CSV file 'people.csv' created successfully.
```

Program 2: Loading a CSV File

Procedure: Load the CSV file created in the previous step.
Code:

```
loaded_df = pd.read_csv('people.csv')
print(loaded_df)
```

Expected Result:

```
     Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   35      Chicago
```

Program 3: Appending to a CSV File

Procedure: Append new data to an existing CSV file.

Code:

```
new_data = pd.DataFrame({
    'Name': ['David', 'Eva'],
    'Age': [40, 22],
    'City': ['San Francisco', 'Seattle']
})
```

```
new_data.to_csv('people.csv', mode='a', header=False, index=False)
print("New data appended to 'people.csv'.")
```

Expected Result:

```
New data appended to 'people.csv'.
```

Program 4: Reading a CSV File with Different Delimiter

Procedure: Load a CSV file with a different delimiter (semicolon).

Code:

```
df = pd.read_csv('people.csv', delimiter=';')
print(df)
```

Expected Result:

```
     Name  Age           City
0   Alice   25       New York
1     Bob   30    Los Angeles
2 Charlie   35        Chicago
3   David   40  San Francisco
4     Eva   22        Seattle
```

Program 5: Loading CSV File with Custom Column Names

Procedure: Load a CSV file and specify custom column names.

Code:

```
df = pd.read_csv('people.csv', names=['Full Name', 'Age', 'Location'], header=0)
print(df)
```

Expected Result:

```
  Full Name  Age       Location
0     Alice   25       New York
1       Bob   30    Los Angeles
2   Charlie   35        Chicago
3     David   40  San Francisco
4       Eva   22        Seattle
```