**17.Statistical analysis**

import numpy as np

import pandas as pd

# Sample NumPy array

data = np.array([50, 55, 60, 65, 70, 75, 80, 85, 90, ])

print("Original Array:\n", data)

# 1. Mean, Median, Standard Deviation

print("\nMean:", np.mean(data))

print("Median:", np.median(data))

print("Standard Deviation:", np.std(data))

# 2. Percentile Scores

percentiles = [25, 50, 75, 90]

for p in percentiles:

 print(f"{p}th Percentile:", np.percentile(data, p))

# 3. Euclidean Distance between two arrays (manual method)

array1 = np.array([1, 2, 3])

array2 = np.array([4, 5, 6])

euclidean\_distance = np.sqrt(np.sum((array1 - array2) \*\* 2))

print("\nEuclidean Distance between array1 and array2:", euclidean\_distance)

# 4. Correlation between two columns

combined = np.array([[85, 78], [90, 85], [78, 80], [92, 95], [88, 90]])

df = pd.DataFrame(combined, columns=['Math', 'English'])

print("\nCorrelation between Math and English:\n", df.corr())

# 5. Probabilistic Sampling

choices = np.array(['apple', 'banana', 'cherry'])

probabilities = [0.2, 0.5, 0.3]

sampled = np.random.choice(choices, size=5, p=probabilities)

print("\nProbabilistic Sampling:", sampled)

# 6. Moving Average

window\_size = 3

moving\_avg = np.convolve(data, np.ones(window\_size)/window\_size, mode='valid')

print("\nMoving Average (window=3):", moving\_avg)

**Output:**

Original Array:

 [50 55 60 65 70 75 80 85 90]

Mean: 70.0

Median: 70.0

Standard Deviation: 12.909944487358056

25th Percentile: 60.0

50th Percentile: 70.0

75th Percentile: 80.0

90th Percentile: 86.0

Euclidean Distance between array1 and array2: 5.196152422706632

Correlation between Math and English:

              Math   English

Math     1.000000  0.751382

English  0.751382  1.000000

Probabilistic Sampling: ['banana' 'cherry' 'banana' 'banana' 'banana']

Moving Average (window=3): [55. 60. 65. 70. 75. 80. 85.]

**18. Data cleaning**

import numpy as np

# Sample NumPy array with missing values represented as np.nan

data = np.array([[1, 2, np.nan], [4, np.nan, 6], [7, 8, 9]])

# 1. Find the positions of missing values

missing\_positions = np.where(np.isnan(data))

print("Positions of missing values:", list(zip(missing\_positions[0], missing\_positions[1])))

# 2. Drop rows that contain missing values

cleaned\_data\_rows = data[~np.isnan(data).any(axis=1)]

print("Array after dropping rows with missing values:\n", cleaned\_data\_rows)

# 3. Replace all missing values with a specified value (e.g., 0)

filled\_data = np.where(np.isnan(data), 0, data)

print("Array after replacing missing values:\n", filled\_data)

# 4. Drop all missing values from the array

flattened\_cleaned\_data = data[~np.isnan(data)]

print("Array after removing all missing values:\n", flattened\_cleaned\_data)

**Output:**

Positions of missing values: [(np.int64(0), np.int64(2)), (np.int64(1), np.int64(1))]

Array after dropping rows with missing values:

 [[7. 8. 9.]]

Array after replacing missing values:

 [[1. 2. 0.]

 [4. 0. 6.]

 [7. 8. 9.]]

Array after removing all missing values:

 [1. 2. 4. 6. 7. 8. 9.]