# Lab 19

# Student Name: Navneet P

# Student ID: AF0411619

## *Topic – Numpy Statistical Functions*

Functions used in this assignment: -

1. **numpy.mean()**: Calculates the average (arithmetic mean) of the elements in an array along a specified axis

2. **numpy.median()**: Computes the median (middle value) of the elements in an array, which separates the higher half from the lower half.

3. **numpy.std()**: Computes the standard deviation, a measure of the amount of variation or dispersion of a set of values in an array.

4. **astype()**: Converts the data type of a NumPy array to a specified type (e.g., from float64 to float32).

5. **numpy.genfromtxt()**: Loads data from a text file, with support for handling missing values and specifying data types for each column.

6. **numpy.isnan()**: Checks whether each element in an array is NaN (Not a Number) and returns a boolean array indicating the presence of NaN values.

7. **numpy.savetxt()**: Saves an array to a text file in a specified format, allowing for control over delimiters and headers.

Q1. How to find the mean of every NumPy array in the given list?
Input: list = [ np.array([3, 2, 8, 9]), np.array([4, 12, 34, 25, 78]), np.array([23, 12, 67]) ]

Solution:

Question 1 Python

```python
import numpy as np

# Define a list containing several numpy arrays
mlist = [np.array([3, 2, 8, 9]), np.array([4, 12, 34, 25, 78]), np.array([23, 12, 67])]

# Print the list of arrays
print("Given List of arrays..\n", mlist)
print()

# Initialize an empty list to store means of each array
all_means = []

# Loop over each array in mlist to calculate the mean
for i, arr in enumerate(mlist):
    # Calculate the mean of the current array
    result = np.mean(arr)

    # Print the current array and its calculated mean
    print(f"Array {i+1}:", arr)
    print(f"Mean of Array {i+1} =", result)
    print()

    # Append the calculated mean to the all_means list
    all_means.append(result)

# After the loop, print the list of all means
print("Mean of Each Array: ")
print(all_means)
```

Output:

```
Given List of arrays..
 [array([3, 2, 8, 9]), array([ 4, 12, 34, 25, 78]), array([23, 12, 67])]

Array 1: [3 2 8 9]
Mean of Array 1 = 5.5

Array 2: [ 4 12 34 25 78]
Mean of Array 2 = 30.6

Array 3: [23 12 67]
Mean of Array 3 = 34.0

Mean of Each Array:
[5.5, 30.6, 34.0]
```

Q2. Compute the median of the flattened NumPy array Input: x_odd = np.array([1, 2, 3, 4, 5, 6, 7])

Solution:

Question 2 — Python

```python
import numpy as np

# Define a numpy array with an odd number of elements
x_odd = np.array([1, 2, 3, 4, 5, 6, 7])

# Define a numpy array with an even number of elements
x_even = np.array([1, 2, 3, 4, 5, 6])

# Print the original array with odd number of elements
print("Printing the Original Array of odd number of elements:")
print(x_odd)

# Calculate the median of the odd-numbered array using np.median()
x_odd_median = np.median(x_odd)

# Print the median value of the odd-numbered array
print("Median of the array that contains odd number of elements:", x_odd_median)
print()

# Print the original array with an even number of elements
print("Printing the Original Array of even number of elements:")
print(x_even)

# Calculate the median of the even-numbered array using np.median()
x_even_median = np.median(x_even)

# Print the median value of the even-numbered array
print("Median of the array that contains even number of elements:", x_even_median)
```

Output:

```
Printing the Original Array of odd no of elements:
[1 2 3 4 5 6 7]
Median of the array that contains odd no of elements 4.0

Printing the Original Array of even no of elements:
[1 2 3 4 5 6]
Median of the array that contains even no of elements 3.5
```

Q3. Compute the standard deviation of the NumPy array Input: arr = [20, 2, 7, 1, 34]

Solution:

Question 3 Python

```python
import numpy as np  # Importing the numpy library for numerical operations

# Defining a Python list with integer elements
arr = [20, 2, 7, 1, 34]

# Converting the list to a numpy array for efficient numerical computation
num = np.array(arr)

# Print the given numpy array
print("Given array: ", num)

# Calculate the standard deviation of the array (default type: float64)
sd = np.std(num)

# Convert the standard deviation result to a lower precision (float32)
f32 = sd.astype("float32")

# Print the standard deviation with higher precision (default float64)
print("\nStandard Deviation with more precision (float64):")
print(sd, type(sd))  # Printing the value and type of 'sd' (float64)

# Print the standard deviation with lower precision (float32)
print("\nStandard Deviation with less precision (float32):")
print(f32, type(f32))  # Printing the value and type of 'f32' (float32)
```

Output:

```
Given array:  [20  2  7  1 34]

Standard Deviation with more precision (float64):
12.576167937809991 <class 'numpy.float64'>

Standard Deviation with less precision (float32):
12.576168 <class 'numpy.float32'>
```

Q4. Suppose you have a CSV file named 'house_prices.csv' with price information, and you want to perform the following
operations:

1. Read the data from the CSV file into a NumPy array.
2. Calculate the average of house prices.
3. Identify house price above the average.
4. Save the list of high prices to a new CSV file. Note: Download 'house_prices.csv' file from LMS.

Solution:

Question 4 — Python

```python
import numpy as np

try:
    # Attempting to load the CSV file with two columns (Index, Price)
    # dtype defines 'Index' as an integer (i4) and 'Price' as a float (f8)
    data = np.genfromtxt(r'C:\Python Programs\numpy\house_prices.csv', dtype=[('Index',
'i4'), ('Price', 'f8')], delimiter=",", encoding=None)

    # Display the total number of records in the data
    print("Total no of records found: ", len(data))
    print("Displaying only 6 records for observational purposes..\n")

    # Iterate over the first few records (6 records for observation)
    count = 0
    for record in data:
        count += 1
        index, value = record  # Seperate the index and price from each record
        if index == -1:  # Skip rows where the index is -1 (if it's used to represent
missing or invalid data)
            continue
        print(f"Index: {index}, Price: {value}")
        if count > 6:  # Limit output to the first 6 records
            break

    # Initialize empty lists to store valid prices and their corresponding indexes
    prices = []
    indexes = []
    counter = 0

    # Loop through all the records to collect non-NaN prices and their indexes
    for record in data:
        counter += 1
        price = record["Price"]  # Extract the price from each record
        index = record["Index"]  # Extract the index from each record
```

```
        if not np.isnan(price):  # If the price is not NaN (valid), store it
            prices.append(price)
            indexes.append(index)

    # Convert the collected prices list into a numpy array for numerical calculations
    prices_array = np.array(prices)

    # Calculate the average price from the prices array and cast it to float32 for
precision control
    avg = np.mean(prices_array).astype("float32")
    print("\nAverage Price:", avg)

    # Identify and store prices above the calculated average, along with their
corresponding indexes
    high_prices = []
    for i in range(len(prices_array)):
        if prices_array[i] > avg:  # Check if the price is above average
            high_prices.append((indexes[i], prices_array[i]))  # Append the index and price
as a tuple
```

```
    # Save the high-priced houses into a new CSV file 'high_prices.csv'
    np.savetxt('high_prices.csv', high_prices, delimiter=",", fmt="%s", header="INDEX |
 PRICE")

    # Display some of the high-priced houses (limit to first 6 records for observation)
    print_counter = 0
    print("\nHigh priced Houses: ")
    print("Displaying only 6 records for observational purposes..\n")

    # Print header
    print("Index  |  Price")

    # Iterate and print high-priced houses
    for index, price in high_prices:
        print_counter += 1
        print("  ", index, "\t", price)
        if print_counter > 6:  # Limit to the first 6 records for display
            break
```

```
# Handle any exceptions that occur during file reading, processing, or writing
except Exception as e:
    print("Some error occurred: ", e)

# Ensure that this block runs regardless of whether an exception occurred or not
finally:
    print("All operations completed successfully.")
    print("New CSV file creation was a success.")
```

Output:

```
 Total no of records found:  169856
 Displaying only 6 records for observational purposes..

 Index: 0, Price: 6000.0
 Index: 1, Price: 13799.0
 Index: 2, Price: 17500.0
 Index: 4, Price: 18824.0
 Index: 5, Price: 6618.0
 Index: 6, Price: 2538.0

 Average Price: 7584.263
```

```
High priced Houses:
Displaying only 6 records for observational purposes..

Index  |  Price
   1     13799.0
   2     17500.0
   4     18824.0
   7     10435.0
   8     10000.0
   9     11150.0
   10    12174.0
All operations completed successfully.
New CSV file creation was a success.
```