Home Assignment <13>: Comprehensive Test Metrics Analysis using NumPy

Learning Objective:

The objective of this assignment is to apply advanced NumPy operations — including slicing, arithmetic operations, power functions, copy mechanisms, and filtering — to analyze large-scale test execution data.

Expected Completion Time:

Best Case: 25 minutes
Average Case: 35 minutes

Assignment Details:

ou are analyzing execution durations (in seconds) for 50 automated test cases across 5 cycles.

a) **Generate synthetic data using NumPy:**

- Create a **5x50 matrix** (`5 cycles × 50 tests`) with random execution times between **5 and 50 seconds**.

b) **Perform the following analyses:**

1. **Statistical Analysis**
   - Calculate the **average execution time per cycle**.
   - Identify the **test case with the maximum execution time** in the entire dataset.
   - Find the **standard deviation of execution times** for each cycle to measure consistency.
2. **Slicing Operations**
   - Extract the first 10 test execution times from **Cycle 1**.
   - Extract the last 5 test execution times from **Cycle 5**.
   - Extract every alternate test from **Cycle 3**.
3. **Arithmetic Operations**
   - Perform element-wise **addition** and **subtraction** between **Cycle 1** and **Cycle 2**.
   - Perform element-wise **multiplication** and **division** between **Cycle 4** and **Cycle 5**.
4. **Power Functions**
   - Square and cube all execution times.
   - Apply a **square root transformation** on the dataset.
   - Apply **logarithmic transformation** (`np.log(array+1)`) to normalize skewed data.
5. **Copy Operations**
   - Create a **shallow copy** of the dataset and modify one cycle. Observe if the original changes.
   - Create a **deep copy** using `.copy()` and modify it. Confirm the original remains unchanged.
6. **Filtering with Conditions**
   - Extract all test cases in **Cycle 2** that take more than **30 seconds**.
   - Identify tests that consistently (in **every cycle**) take more than **25 seconds**.
   -

- Replace all execution times below **10 seconds** with `10` (minimum thresholding).

Hints :

- Use `np.random.randint(5, 51, size=(5, 50))` to generate the dataset.
- Use slicing (`array[start:end:step]`) for extracting subsets.
- Use `+`, `-`, `*`, `/` directly on NumPy arrays for arithmetic.
- Use `np.power()`, `np.sqrt()`, and `np.log()` for transformations.
- Use `.view()` for shallow copy and `.copy()` for deep copy.
- Use boolean indexing for filtering: `array[array > 30]`.

Expected Outcome:

Upon completion of this assignment, you should be able to:

- Perform statistical analysis on multidimensional datasets.
- Apply slicing to extract meaningful subsets.
- Execute element-wise arithmetic operations.
- Use power functions for mathematical transformations.
- Understand shallow vs deep copy in NumPy.
- Apply conditional filtering to extract or replace specific values.