# Machine Learning Lab

Experiment 1: NumPy Statistical Computations

For CSE Department, Semester 06

Course Code: U23CM6L2

Compiled by
Mohammed Ufraan

March 24, 2026

# Experiment 1

**Aim:** Using Python, write a NumPy program to compute the following:

a) **Expected Value**

**Definition:** Expected Value is a fundamental concept in probability and statistics. It is defined as the weighted average of all possible values of a random variable, where each value is weighted by its probability or relative frequency.

**Input Format:**

| **Value ($x$)** | **Frequency ($f$)** |
|---|---|
| 10 | 2 |
| 20 | 3 |
| 30 | 5 |

- A list of numerical values $x$
- A corresponding list of frequencies $f$

**Formula:**

$$E(X) = \frac{\sum(x \times f)}{\sum f}$$

**Algorithm:**

1. Start the program.
2. Initialize a list of values.
3. Initialize a list of corresponding frequencies.
4. Calculate the total frequency by summing all frequency values.
5. Initialize a variable to store the sum of $x \times f$.
6. For each value and its frequency:
    - Multiply the value by its frequency.
    - Add the result to the sum.
7. Divide the total of $x \times f$ by the total frequency.
8. Display the expected value.
9. Stop the program.

**Program:**

```python
# a) Expected Value
import numpy as np
x = [10, 20, 30]
f = [2, 3, 5]
expected_value = sum(xi*fi for xi, fi in zip(x, f)) / sum(f)
print("Expected Value:", expected_value)
```

Listing 1: Expected Value using NumPy

**Expected Output:**

```
Expected Value: 23.0
```

b) **Mean**

**Definition:** The mean is the average of all numbers in the dataset.

**Input Format:**

- A numeric array of values. Example: `[1, 2, 3, 4, 5]`

**Algorithm:**

1. Start the program.
2. Input the array.
3. Convert to NumPy array.
4. Compute mean using `np.mean(array)`.
5. Print the result.
6. End the program.

**Program:**

```
# b) Mean
import numpy as np
arr = [1, 2, 3, 4, 5]
print("Mean:", np.mean(arr))
```

Listing 2: Mean using NumPy

**Expected Output:**

```
Mean: 3.0
```

c) **Standard Deviation**

**Definition:** Standard deviation measures how spread out the values in a dataset are from the mean.

**Input Format:**

- A numeric array. Example: `[1, 2, 3, 4, 5]`

**Algorithm:**

1. Start the program.
2. Input the array.
3. Convert to NumPy array.
4. Compute standard deviation using `np.std(array)`.
5. Print the result.
6. End the program.

**Program:**

```
# c) Standard Deviation
import numpy as np
arr = [1, 2, 3, 4, 5]
print("Standard Deviation:", np.std(arr))
```

Listing 3: Standard Deviation using NumPy

**Expected Output:**

```
Standard Deviation: 1.4142135623730951
```

d) **Variance**

**Definition:** Variance is the square of standard deviation and measures the spread of data.

**Input Format:**

- A numeric array. Example: `[1, 2, 3, 4, 5]`

**Algorithm:**

1. Start the program.
2. Input the array.
3. Convert to NumPy array.
4. Compute variance using `np.var(array)`.
5. Print the result.
6. End the program.

**Program:**

```python
# d) Variance
import numpy as np
arr = [1, 2, 3, 4, 5]
print("Variance:", np.var(arr))
```

Listing 4: Variance using NumPy

**Expected Output:**

```
Variance: 2.0
```

### e) Covariance

**Definition:** Covariance measures how two variables change together. Positive covariance indicates they increase together; negative covariance indicates opposite trends.

**Input Format:**

- Two numeric arrays of equal length. Example: `[1, 2, 3, 4, 5]` and `[5, 4, 3, 2, 1]`

**Algorithm:**

1. Start the program.
2. Input two arrays of equal length.
3. Convert to NumPy arrays.
4. Compute covariance using `np.cov(arr1, arr2)[0,1]`.
5. Print the result.
6. End the program.

**Program:**

```python
# e) Covariance
import numpy as np
arr1 = [1, 2, 3, 4, 5]
arr2 = [5, 4, 3, 2, 1]
print("Covariance:", np.cov(arr1, arr2)[0,1])
```

Listing 5: Covariance using NumPy

**Expected Output:**

```
Covariance: -2.5
```

### f) Covariance Matrix

**Definition:** A covariance matrix is a square matrix that shows the variance of each variable along the diagonal and the covariance between pairs of variables in the off-diagonal elements.

**Input Format:** Two numeric arrays of equal length representing marks in two subjects.

| Student | Psychology (X) | History (Y) |
|---|---|---|
| Anna | 80 | 70 |
| Caroline | 63 | 20 |
| Laura | 100 | 50 |

**Algorithm:**

1. Start the program.
2. Import the NumPy library.
3. Define an array $X$ containing marks in Psychology.
4. Define an array $Y$ containing marks in History.
5. Use the NumPy function `np.cov(X, Y)` to compute the covariance matrix.
6. Display the covariance matrix.
7. Stop the program.

**Program:**

```
# f) Covariance Matrix
import numpy as np
X = [80, 63, 100]
Y = [70, 20, 50]
print("Covariance Matrix:\n", np.cov(X, Y))
```

Listing 6: Covariance Matrix using NumPy

**Expected Output:**

```
Covariance Matrix:
 [[343.         260.        ]
  [260.         633.33333333]]
```