**UNIVERSITY OF SCIENCE**

**FACULTY OF INFORMATION TECHNOLOGY**

**A blue and white logo

Description automatically generated**

**MULTIVARIATE**

**STATISTICAL ANALYSIS**

**Report Practice 01  
NumPy**

**Lecturers**  Lý Quốc Ngọc

Nguyễn Mạnh Hùng

Phạm Thanh Tùng

**Student** Võ Nguyễn Hoàng Kim

21127090

**PHỤ LỤC**

[I. SELF-ASSESSMENT FORM 3](#_Toc161249083)

[II. LIST OF THE FUNCTIONS 3](#_Toc161249084)

[III. IMPLEMENTATION 4](#_Toc161249085)

[1. Mean and Median 4](#_Toc161249086)

[2. Order statistics 5](#_Toc161249087)

[3. Variance and Standard Deviation 7](#_Toc161249088)

[4. Correlation 8](#_Toc161249089)

[IV. SUMARIZE 9](#_Toc161249090)

[V. REFFERENCES 10](#_Toc161249091)

# **SELF-ASSESSMENT FORM**

|  |  |  |
| --- | --- | --- |
| **Features** | | **Level of completion** |
| **Mean, Median** | Mean | 100% |
| Median | 100% |
| **Order Statistics** | Max | 100% |
| Min | 100% |
| Range | 100% |
| **Variance & Stand Deviation** | Variance | 100% |
| Standard Deviation | 100% |
| **Correlation** | Correlation | 100% |

# **FEATURES**

***Note:*** *Assume X is an* ***input******array*** *initialized by the* ***np.array()*** *function with values initialized by default in the source code:*

## **Data**

* Download: <https://www.kaggle.com/datasets/zeesolver/apple-quality>
* This dataset was collected by an American agricultural company. It is based on various factors such as weight, size, and taste of an apple to assess its quality. The analysis is based on 7 attributes for evaluation and 1 factor used for tracking to classify apples into 2 types, specifically:
  + A\_id: Apple ID for tracking.
  + Size: Apple dimensions.
  + Weight: Apple mass.
  + Sweetness: Apple taste.
  + Crunchiness: Apple texture.
  + Juiciness: Apple moisture.
  + Ripeness: Apple maturity.
  + Acidity: Apple tartness.
  + Quality: Overall apple grade.
* The last attribute “Quality” is the class identifier (Good or Bad).

## **r statistics**

1. **Max**

* Implement:
  + With **np.max(),** we can get the Maximum of X (in array) or Maximum along an axis.
    - In this program, *axis* takes values 0 or 1, along which to operate [3]:
      * If this is a tuple of ints, the maximum is selected over multiple axes, instead of a single axis or all the axes as before.
      * If we set *axis = 0*, the maximum is selected along the Y-axis (vertical direction).
      * On the contrary, it will be selected along the X-axis (horizontal).
      * By default, flattened input is used.
  + In cases where there are NaN (Not a Number) values in the dataset, we can use **np.nanmax()** to handle them.
* Result:

Below are the results obtained from the experiment using the **np.max()** and **np.nanmax()** functions with the issues mentioned above

* + In normal case:

A black screen with white text

Description automatically generated

* + Exist NaN in dataset:

*A black screen with white text

Description automatically generated*

1. **Min**

* Implement:
  + With **np.min(),** we can get the Minimum of X (in array) or Minimum along an axis
    - In this program, *Axis* takes values 0 or 1, along which to operate [4]:
      * If this is a tuple of ints, the minimum is selected over multiple axes, instead of a single axis or all the axes as before.
      * If we set *axis = 0*, the minimum is selected along the Y-axis (vertical direction).
      * On the contrary, it will be selected along the X-axis (horizontal).
      * By default, flattened input is used.
  + In cases where there are NaN (Not a Number) values in the dataset, we can use **np.nanmin()** to handle them.
* Result:

Below are the results obtained from the experiment using the **np.min()** and **np.nanmin()** functions with the issues mentioned above

* + In normal case:

*A black screen with white text

Description automatically generated*

* + Exist NaN in dataset:

*A screenshot of a computer

Description automatically generated*

1. **Range**

* Implement:
  + Range in this term means the difference between max value and min value along the axis in the dataset. With **np.ptp(),** we can get the Range of value of X.
    - In this program, *axis* takes values 0 or 1, along which to find the peek [5]:
      * If this is a tuple of ints, a reduction is performed on multiple axes, instead of a single axis or all the axes as before.
      * If *axis = 0*, the range is computed along the Y-axis (vertical direction).
      * On the contrary, it will be computed along the X-axis (horizontal).
      * By default, flattened input is used.
  + NaN cases are not supported by NumPy in computing the range of values of X. It will return NaN as the result.
* Result:

Below are the results obtained from the experiment using the **np.ptp()** function with the issues mentioned above

* + In normal case:

A black screen with white text

Description automatically generated

* + Exist NaN in dataset:

A black screen with white text

Description automatically generated

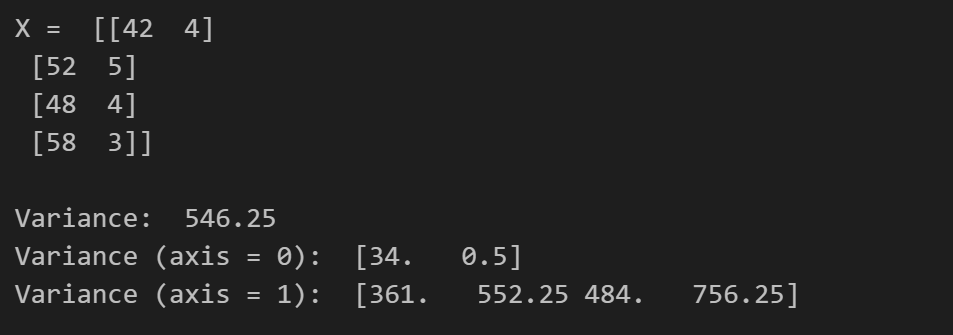
## **Variance and Standard Deviation**

1. **Variance**

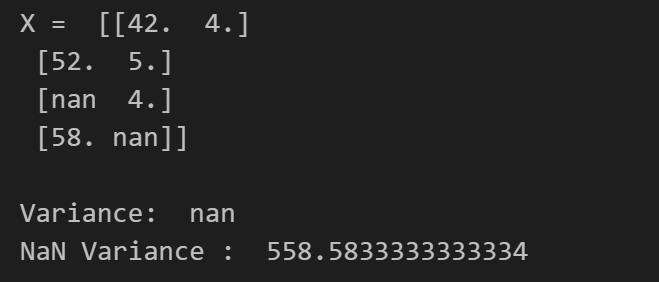
* Implement:
  + With **np.var(),** we can compute the Variance value of X along the specified axis:
    - In this program, *axis* takes values 0 or 1, along which the variance is computed [6]:
      * If this is a tuple of ints, a variance is performed over multiple axes, instead of a single axis or all the axes as before.
      * If *axis = 0*, the variance is computed along the Y-axis (vertical direction).
      * On the contrary, it will be computed along the X-axis (horizontal).
      * By default, flattened input is used.
  + In cases where there are NaN (Not a Number) values in the dataset, we can use **np.nanvar()** to handle them.
* Result:

Below are the results obtained from the experiment using the **np.var()** and **np.nanvar()** functions with the issues mentioned above

* + In normal case:



* + Exist NaN in dataset:

**

1. **Standard Deviation**

* Implement:
  + With **np.std(),** we can compute the Standard Deviation value of X along the specified *axis*:
    - In this program, *axis* takes values 0 or 1, along which the variance is computed [7]:
      * If this is a tuple of ints, a standard deviation is performed over multiple axes, instead of a single axis or all the axes as before.
      * If we set *axis = 0*, the variance is computed along the Y-axis (vertical direction).
      * On the contrary, it will be computed along the X-axis (horizontal).
      * By default, flattened input is used.
  + In cases where there are NaN (Not a Number) values in the dataset, we can use **np.nanstd()** to handle them.
* Result:

Below are the results obtained from the experiment using the **np.std()** and **np.nanstd()** functions with the issues mentioned above

* + In normal case:

A screenshot of a computer

Description automatically generated

* + Exist NaN in dataset:

A computer screen with white text

Description automatically generated

## **Correlation**

1. **Correlation**

* Implement:
  + With **np.corrcoef(),** we can get the Correlation Coefficients of X. Elements’ values in the Correlation are between -1 and 1. In this function, we need to pay attention to 2 important parameters that are commonly used [8]:
    - *input-array*: as default, it is a 1-D or 2-D array containing multiple variables and observations. Each row of x represents a variable, and each column a single observation of all those variables.
    - *rowvar:* plays an important role, affecting how the input matrix is considered when computing the correlation coefficient matrix.
      * If *rowvar = True* (default), then each row represents a variable, with observations in the columns.
      * Otherwise, the relationship is transposed: each column represents a variable, while the rows contain observations.
  + NaN cases are not supported by NumPy in computing the range of values of X. It will return NaN as the result.
* Result:

Below are the results obtained from the experiment using the **np.** **corrcoef()** functions (with *rowvar = False)* with the issues mentioned above

* + In normal case:

A computer screen shot of a black screen

Description automatically generated

* + Exist NaN in dataset:

A screenshot of a computer

Description automatically generated

# **SUMARIZE**

* In summary, in practice 01, we used various NumPy library functions to compute basic statistical quantities with different value types such as float32, float64, NaN, etc.
* Regarding NaN values, it can be observed that NumPy provides support for computation with datasets containing this data type, and the approach is generally consistent: ignoring (or removing) NaN values and compute the needed value based on the remaining values.
  + Example: To compute Means value as the result above (with *axis = none*), **np.nanmean()** has ignored the presence of NaN values in X (considering them non-existent in X), and computed the mean value based on the remaining numbers, which are 2, 5, 7, 9.

*A black background with white text

Description automatically generated*

* However, not all quantities are supported by NumPy for computation with NaN, such as Correlation and Variance. If we still attempt to use these functions with a dataset containing NaN values, the resulting NaN outputs will not be accurate.

# **REFFERENCES**

|  |  |
| --- | --- |
| [1] | NumPy Developers, "numpy.mean," NumPy, [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.mean.html. |
| [2] | NumPy Developers, "numpy.median," NumPy, [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.median.html. |
| [3] | NumPy Developers, "numpy.max," NumPy, [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.max.html. |
| [4] | Numpy Developers, "numpy.min," NumPy, [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.min.html. |
| [5] | NumPy Developers, "numpy.ptp," NumPy, [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.ptp.html. |
| [6] | NumPy Developers, "numpy.var," NumPy, [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.var.html. |
| [7] | NumPy Developers, "numpy.std," NumPy, [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.std.html. |
| [8] | NumPy Developers, "numpy.corrcoef," NumPy, [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html. |