Fuzzy-TOPSIS

Fuzzy-TOPSIS is a step up over simplified TOPSIS. It also encourages the same approach where we rank the alternatives based on their closeness to the right solution. However, the fact that this works better than simplified TOPSIS is that it takes in account of the human thinking as well, so in place of taking a concrete value, we take a range of value which is denoted with the help of fuzzy numbers.

**4.1 Concept algorithm of Fuzzy-TOPSIS**

Step 1- Find the weight of the criteria. Criteria is an important factor which decides by what weightage, it impacts the final outcome.

Here, the users give their own weights for all criteria.

Step 2- Now based on linguistic, we can make a dataset having criteria and their alternatives along with triangular fuzzy number. For comparison, we can use the scale below.

In this each element is represent as xij where i range from 1 to m denoting alternatives and j range from 1 to n denoting criteria.A black and white text on a white background

Description automatically generated

Step 3- Next creating the fuzzy dataset, we can proceed to build weighted normalized fuzzy decision matrix.

* Every element in weighted normalized fuzzy decision matrix is represented using Vij.
* Vi j = [vi j]mn such that i = 1, 2,..., n; j = 1, 2,..., m;

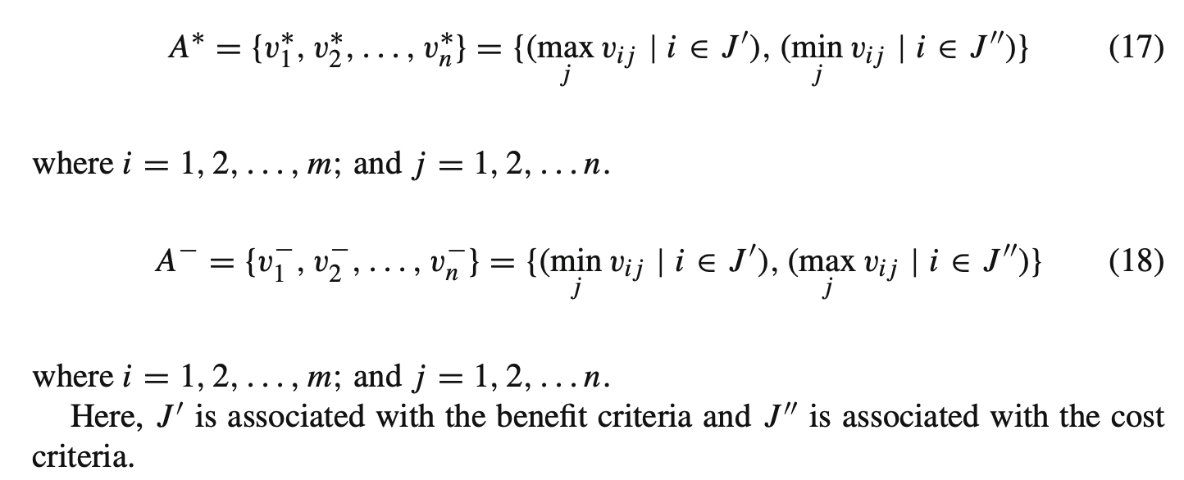
A math equation with black text

Description automatically generated

* Where wj represents the weight of every criteria.

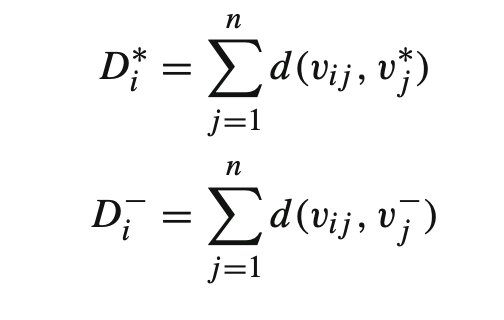
Step 4- Identifying the fuzzy positive ideal solution (FPIS) and fuzzy negative ideal solution (FNIS) The best (FPIS solution) is represented by A∗ and the worst (FNIS solution) is denoted as A− for each specified criterion and is calculated.

For calculating the FPIS and FNIS we can use several equations:

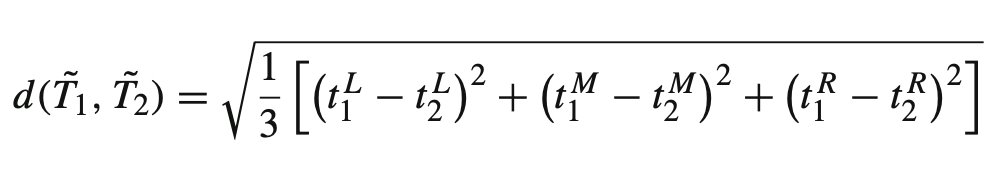


Step 5- Calculating the distance of each alternative from the FPIS and FNIS. The distance

(D∗ i , D− i ) of each alternative from A∗ and A− is computed with the help of two equations.



We can find the distance between two fuzzy points using the equation:



Step 6- Computing the closeness coefficient of each alternatives(Cci). This can be calculated for each alternative.

A mathematical equation with black text

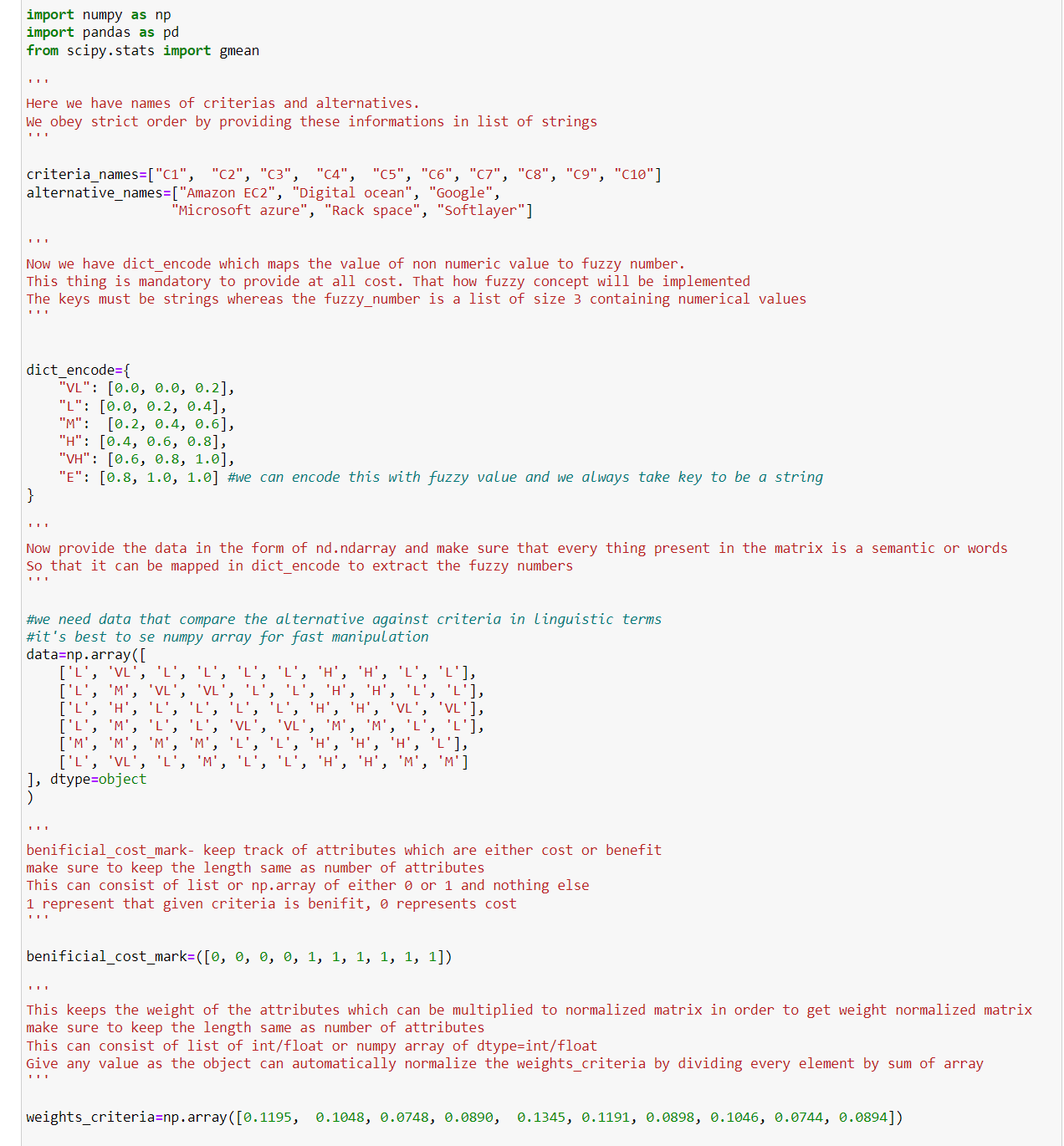
Description automatically generated

Step 7- Rank the preference order. The different alternatives are ranked based on the computed CCi values in the descending order, using which the most feasible alternatives are selected.

**4.2 Package Code**

Code for Fuzzy-TOPSIS is divided in two parts. First part is the parameter check and second part contains the logic of implementing Fuzzy-AHP.

Input Field:

****

List of input parameters provided in the example:

1. Criteria and Alternatives:

- `criteria\_names`: A list containing the names of criteria used for decision-making. In this example, there are 10 criteria named "C1" to "C10".

- `alternative\_names`: A list containing the names of alternatives to be evaluated. In this case, there are 6 alternatives named "Amazon EC2", "Digital Ocean", "Google", "Microsoft Azure", "Rack Space", and "Softlayer".

2. Fuzzy Encoding:

- `dict\_encode`: A dictionary mapping non-numeric values (linguistic terms) to fuzzy numbers. Each key represents a linguistic term (string), and the corresponding value is a list of three numerical values representing the lower, medium, and upper bounds of the fuzzy number.

3. Data Comparison Matrix:

- `data`: A numpy array representing the comparison matrices for each criterion. The matrix contains linguistic terms comparing each alternative against each criterion. The terms are mapped to fuzzy numbers using `dict\_encode`.

4. Benefit or Cost Marking:

- `benificial\_cost\_mark`: A list indicating whether each criterion is a benefit or a cost. In this example, there are 10 criteria, and the list specifies whether each criterion is a benefit (1) or a cost (0).

5. Weights of Criteria:

- `weights\_criteria`: A numpy array specifying the weights of criteria. These weights determine the relative importance of each criterion in the decision-making process. The values provided are already normalized.

These inputs collectively define the structure and content necessary to conduct Fuzzy-TOPSIS analysis.

Parameter check:

The code for parameter checks of Fuzzy-TOPSIS is provided below-

****

The `Check\_parameters\_Fuzzy\_TOPSIS` class perform checks:

1. Data Type and Structure Checks:

- The constructor (`\_\_init\_\_` method) initializes the object attributes with the provided parameters.

- It verifies that `criteria\_names`, `alternative\_names`, `weights\_criteria`, `benificial\_cost\_mark`, and `data` are of the correct types (list, ndarray).

- It ensures that `weights\_criteria` and `benificial\_cost\_mark` have the same length as the number of criteria.

- The method checks that `data` is a 2D numpy array with the correct shape based on the number of alternatives and criteria.

- If any of these conditions are not met, it raises a `ValueError` with an appropriate error message.

2. Normalization of Weights Criteria:

- The method normalizes the `weights\_criteria` array by dividing each element by the sum of all elements. This ensures that the weights sum up to 1, as required for TOPSIS analysis.

- This normalization step ensures that the weights represent the relative importance of each criterion accurately.

3. Beneficial or Cost Mark Checks:

- It confirms that elements in `benificial\_cost\_mark` are either 0 or 1, indicating whether each criterion is a benefit or a cost.

- This check ensures that each criterion is correctly classified as a benefit or a cost, which is essential for TOPSIS analysis to determine the ideal alternative.

4. Fuzzy Encoding Checks:

- The method verifies that `dict\_encode` is a dictionary where keys are strings (linguistic terms) and values are lists of three numeric values representing the l, m, u of fuzzy numbers.

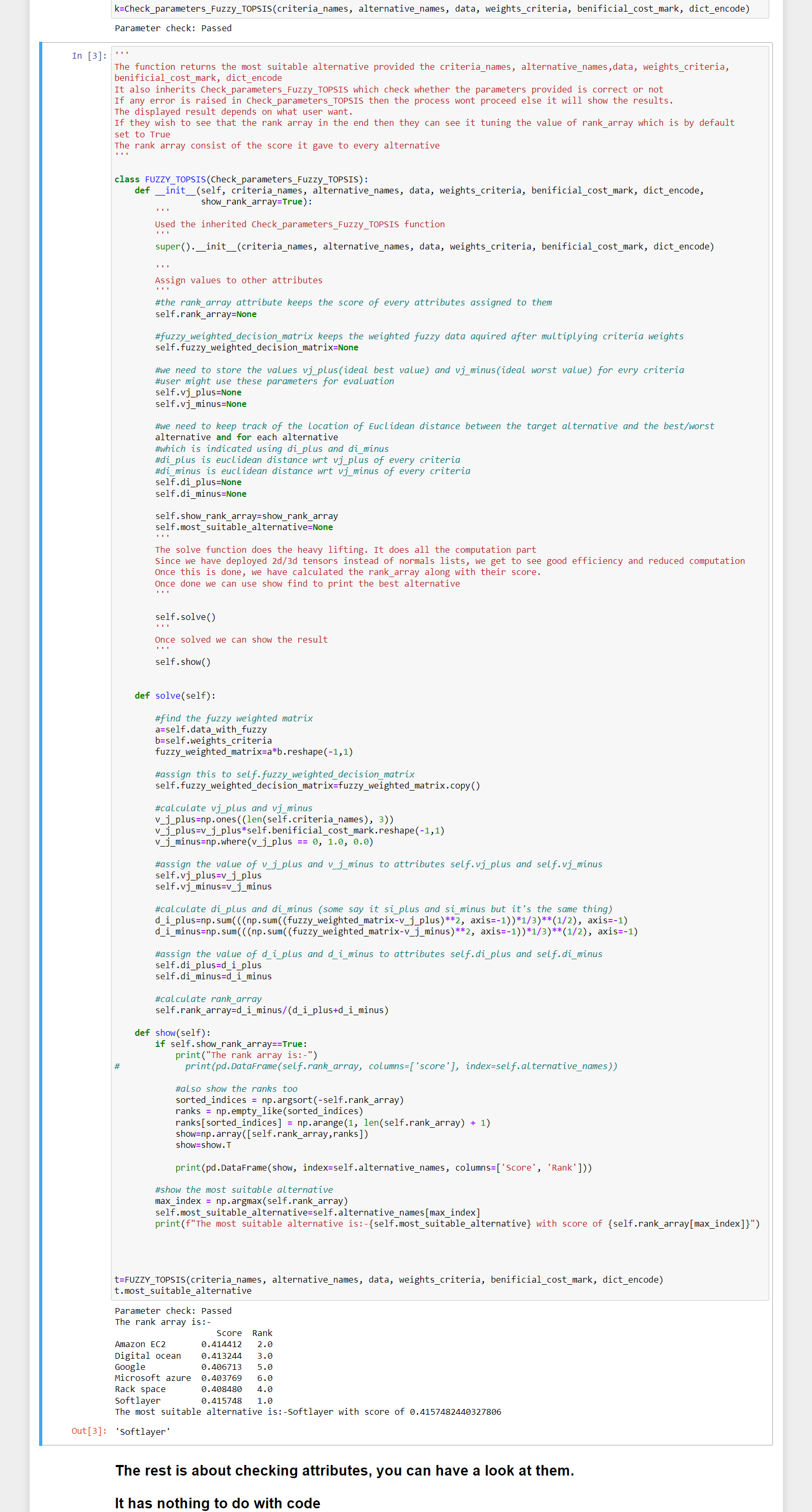
- It checks that all elements in the `data` array can be mapped to the keys in `dict\_encode`, ensuring that all linguistic terms in the data have corresponding fuzzy representations.

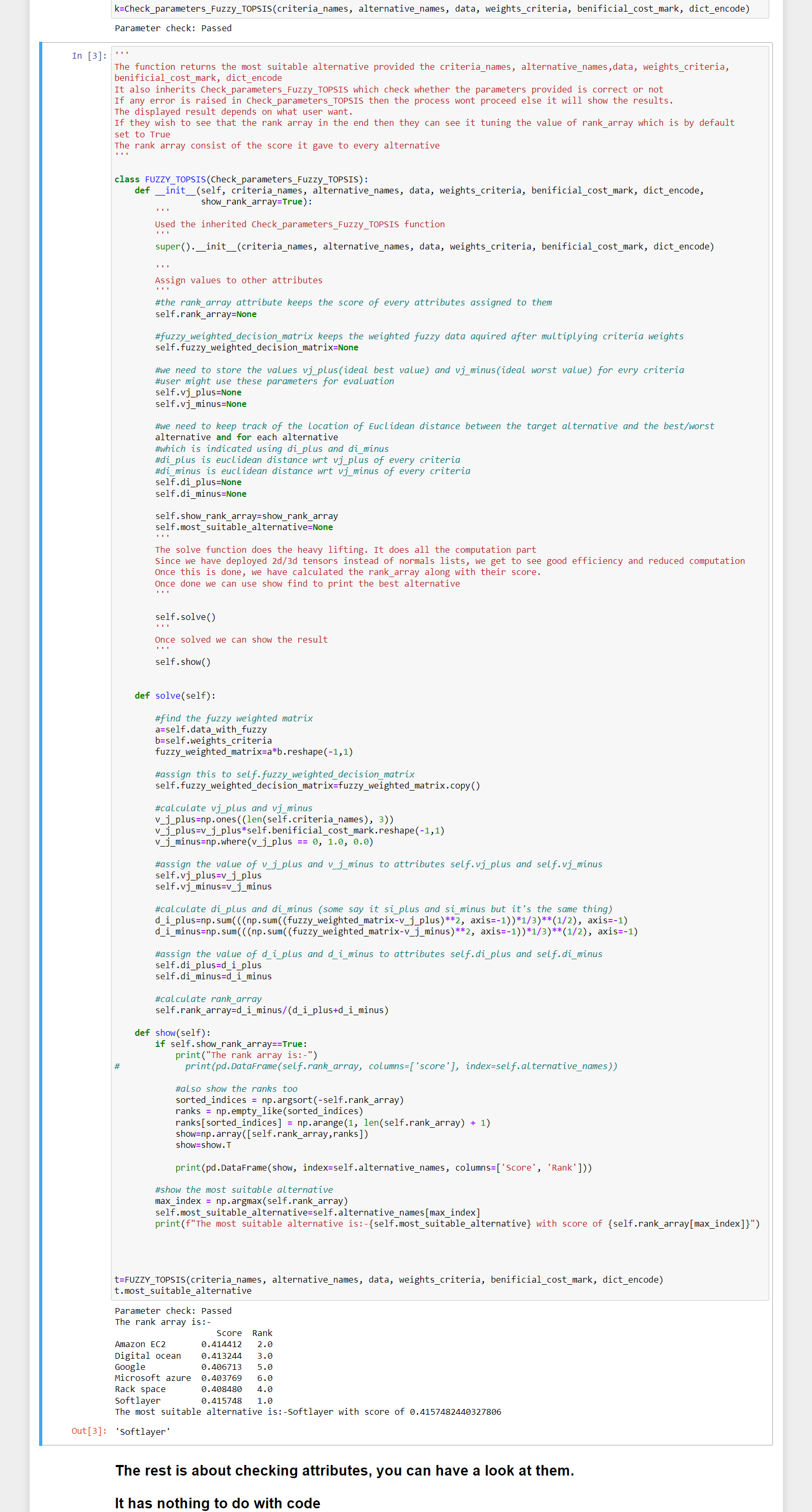
- If any term in the data is not present in the dictionary, it raises a `ValueError` to indicate that the element is not present in the dictionary.

5. \*\*Overall Check and Error Handling:\*\*

- If any check fails, a `ValueError` is raised with an appropriate error message indicating the specific issue encountered.

- If all checks pass, the method prints "Everything is fine" to indicate that the provided parameters are valid and ready for further processing.

Fuzzy-TOPSIS code:****

****

Working-

1. Initialization:

- Inheritance: Inherits from the `Check\_parameters\_Fuzzy\_TOPSIS` class to ensure that the input parameters are validated before proceeding with the TOPSIS computation. This helps in preventing errors due to incorrect input.

- Attribute Initialization: Initializes various attributes necessary for the TOPSIS computation and result presentation, such as:

- `rank\_array`: Placeholder for the TOPSIS scores of each alternative.

- `fuzzy\_weighted\_decision\_matrix`: Holds the result of element-wise multiplication between the fuzzy data and the criteria weights, representing the weighted decision matrix.

- `vj\_plus` and `vj\_minus`: Arrays representing the ideal best and ideal worst values for each criterion, respectively.

- `di\_plus` and `di\_minus`: Arrays holding the Euclidean distances between each alternative and the ideal best/worst values for each criterion.

- `show\_rank\_array`: A flag indicating whether to display the rank array during result presentation.

- `most\_suitable\_alternative`: Placeholder for the name of the most suitable alternative determined by TOPSIS.

2. Solving the Problem:

- Computing Fuzzy Weighted Decision Matrix:

Performs element-wise multiplication between the fuzzy data and the criteria weights to obtain the fuzzy weighted decision matrix. This matrix reflects the importance of each alternative with respect to each criterion.

- Calculating Ideal Values:

Computes the ideal best (`vj\_plus`) and ideal worst (`vj\_minus`) values for each criterion based on the specified beneficial/cost nature of criteria.

- Determining Euclidean Distances:

Calculates the Euclidean distances between each alternative and the ideal best/worst values for each criterion.

- Computing TOPSIS Score:

Derives the TOPSIS score (`rank\_array`) for each alternative based on the normalized distances. Alternatives with higher scores are considered more suitable.

3. Showing Results:

- Displaying Rank Array (Optional):

If `show\_rank\_array` is set to `True`, prints the rank array, which contains the TOPSIS scores of each alternative. This provides insight into the relative suitability of alternatives.

- Presenting Alternative Rankings:

Sorts the alternatives based on their TOPSIS scores in descending order, assigning ranks accordingly. This ranked list offers a clear comparison of alternatives' suitability.

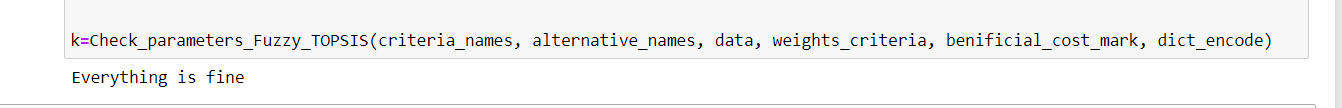
- Identifying Most Suitable Alternative:

Determines the most suitable alternative with the highest TOPSIS score and prints its name, providing a straightforward recommendation for decision-making.

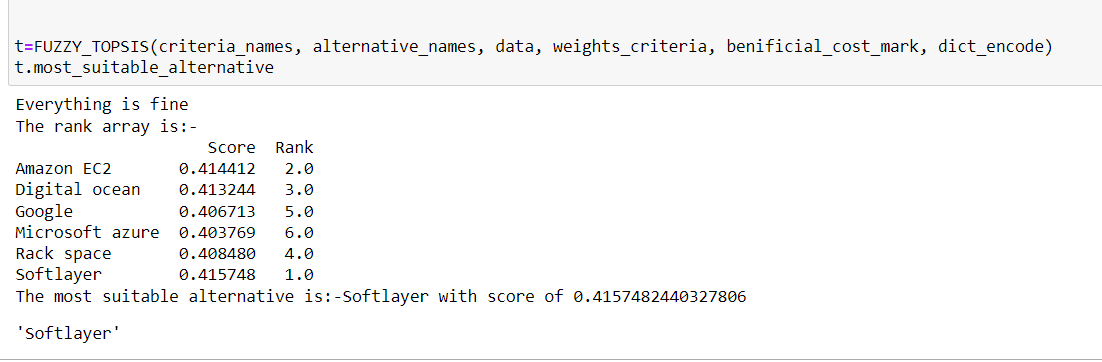
By performing these detailed actions, the `FUZZY\_TOPSIS` class ensures accurate computation of Fuzzy TOPSIS.

**4.3 Results**

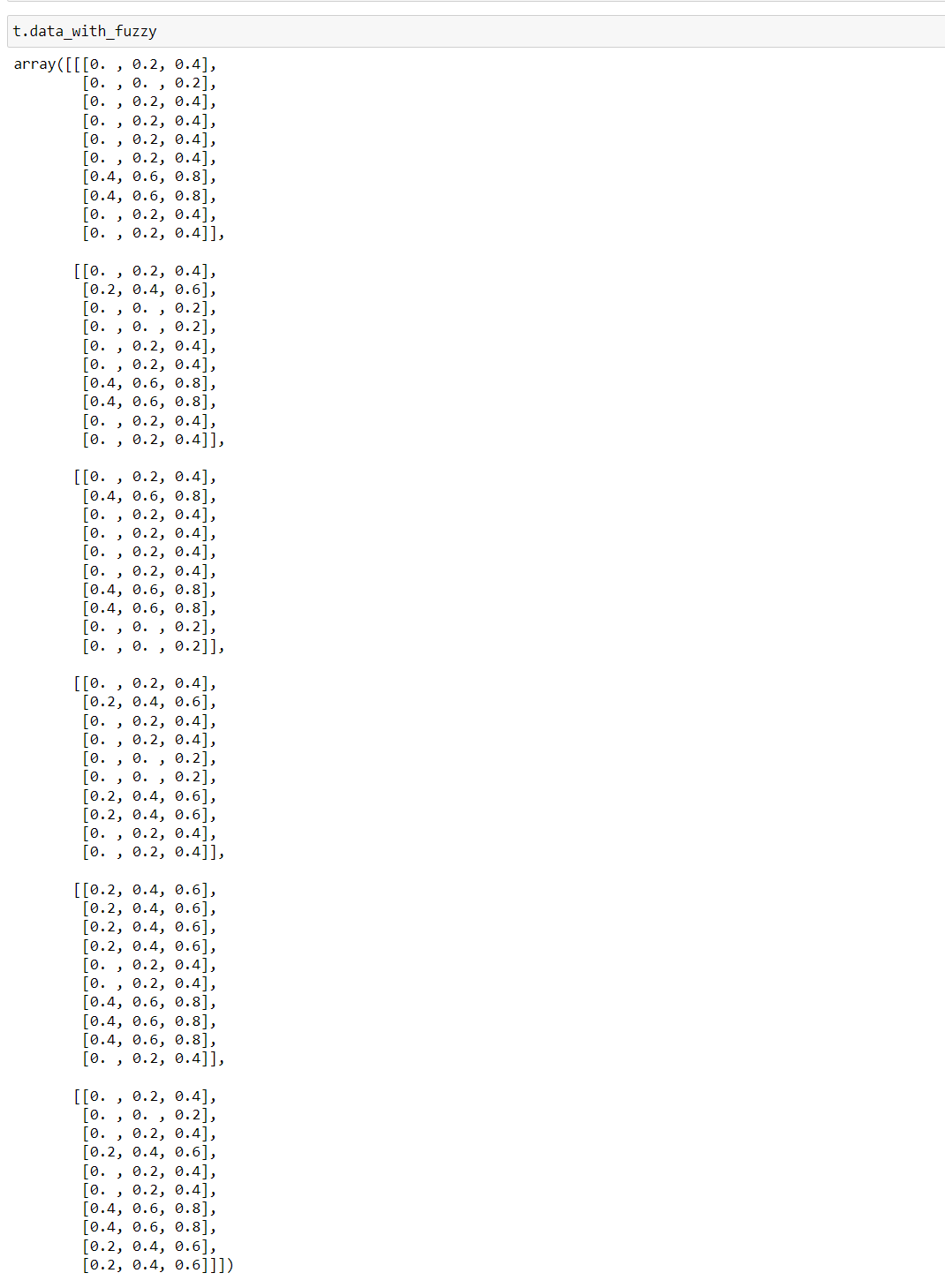
Result of Check\_parameters\_Fuzzy\_TOPSIS:

****

Fuzzy\_TOPSIS result:

****

Other Attributes:

****

