Copras

Complex Proportional Assessment (COPRAS) is very useful technique which is normally used for multi-attribute decision making. COPRAS mainly focus on deriving relative weight and utility of each alternative. This utility can be converted in terms of ranks of all alternative to select the most favourable option amongst all.

**5.1 Concept algorithm of COPRAS-**

1) Provide the dataset which contains rows as alternative and columns as criteria.

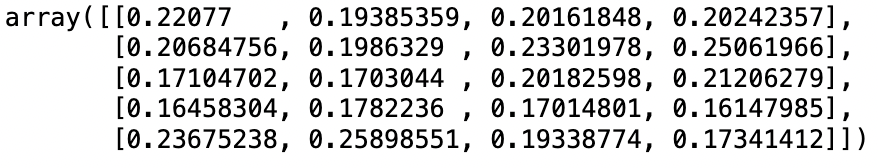


Here the green columns represents the benefit criteria and green columns represents the cost criteria which can be distinguished with the help of ‘benificial\_cost\_mark’ provided in the input as we discuss further.

Here the alternative are:- ["Maxwell", "Dhoni", "AB de Villiers", "Andre Russell", "Rinku Singh"]

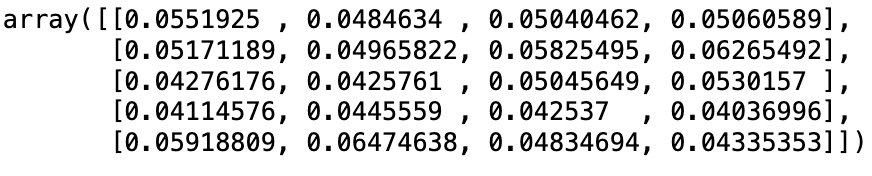
And the criteria are:- ["Batting Average", "Batting Strike Rate", "Bowling Average", "Bowling Strike Rate"]

2) Normalize the dataset such that every element is divided with the sum of the all element present in that columns



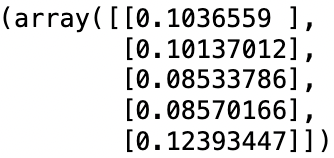
The normalized dataset

3) Multiply it with the weight provided by the user. Here the user provides weight of each column as- [0.25, 0.25, 0.25, 0.25], so just multiply the weights to get the ‘weighted\_normalized\_matrix’ which will look like this.



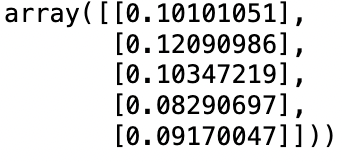
Weighted normalized matrix

4) Sum through all the columns for each respective row(alternative) for the benefit criteria through which we attain the value of bi. The dimension of bi is (number of alternatives, 1) and for this case it is:-



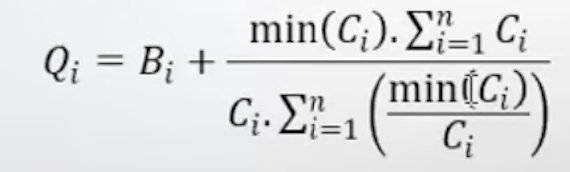
bi

5) Sum through all the columns for each respective row(alternative) for the cost criteria through which we attain the value of ci. The dimension of ci is (number of alternatives, 1) and for this case it is:-

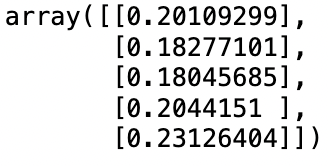


Ci

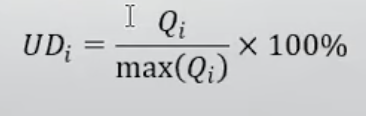
6) Now compute the value of qi which can be given as:-



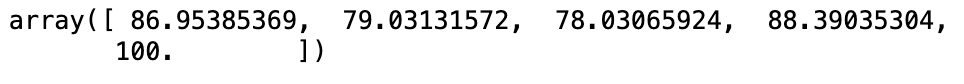
So we compute the value of qi for every alternative which turns out to be:-



7) And so, we can compute ‘rank array’ or ‘UDi’ for each alternative which can be given as: -

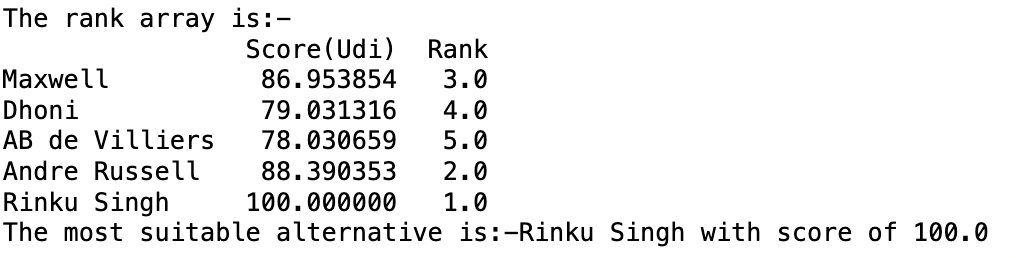


And once we compute this, it will turn out to be:-



The udi or the rank array

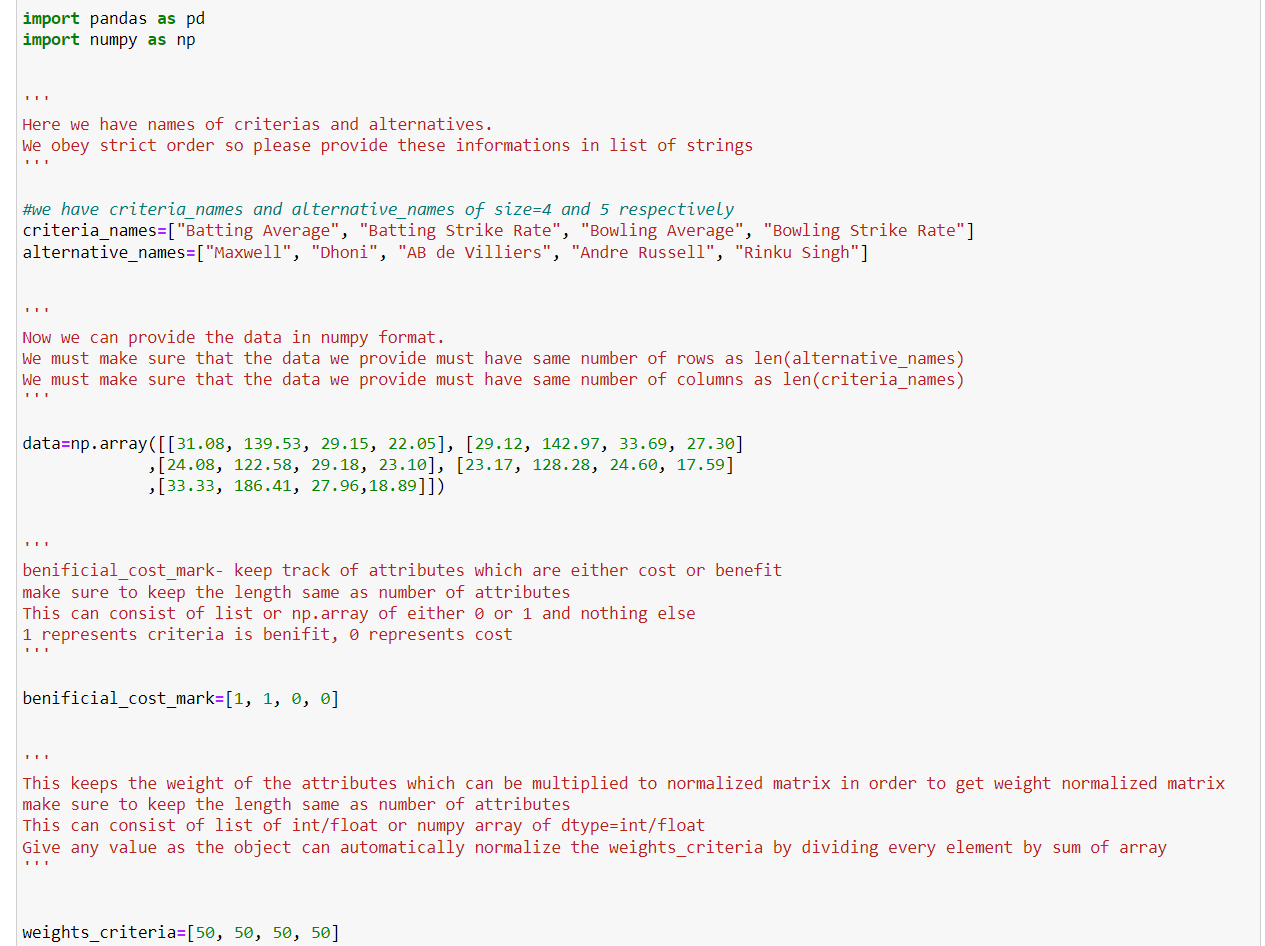
8)Now all we have to do is rank the alternative based in their scores:-

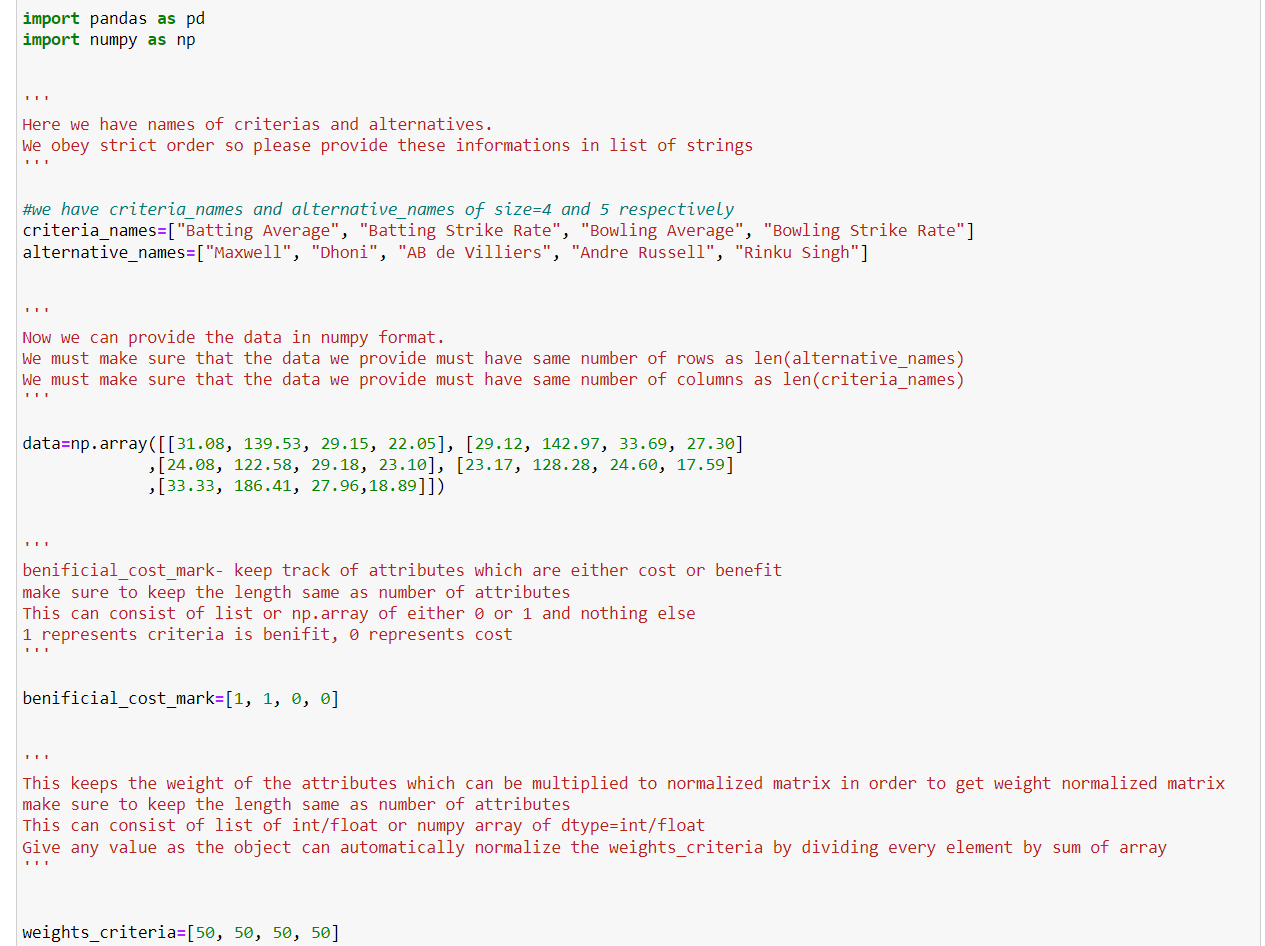


The result shows that the last alternative “Rinku Singh” fits as the best alternative for the given criteria’s.

**5.2 Package Code**

Input parameter:





The input parameters provided are:

1. Criteria and Alternatives:

- `criteria\_names`: A list containing the names of criteria used for evaluation.

- `alternative\_names`: A list containing the names of alternatives being evaluated.

2. Data:

- `data`: A NumPy array representing the performance data of each alternative on each criterion.

3. Beneficial/Cost Mark:

- `benificial\_cost\_mark`: A list indicating whether each criterion is considered beneficial (1) or a cost (0).

4. Criteria Weights:

- `weights\_criteria`: A list specifying the weights assigned to each criterion. These weights are used to calculate the weighted normalized decision matrix.

These parameters provide the necessary information to perform Copras analysis and determine the most suitable alternative based on the specified criteria and their weights.

Parameter check:

Below is the code for parameter check-





Here are the checks performed in the `Check\_parameters\_COPRAS` class:

1. Criteria and Alternative Names Validation:

- Ensure `criteria\_names` and `alternative\_names` are lists.

- Verify that every element in these lists is a string.

2. Data Validation:

- Check if `data` is a NumPy array.

- Ensure that the data type of elements in the array is either integer or float.

- Validate that the array has exactly two dimensions.

- Verify that the shape of the array matches the number of alternatives and criteria provided.

3. Weights Criteria Validation:

- Check whether `weights\_criteria` is a list or a NumPy array.

- Ensure that the array has one dimension and its length matches the number of criteria.

- Validate that every value in the array is numeric (integer or float).

- Normalize the weights criteria by dividing each element by the sum of the array.

4. Beneficial/Cost Mark Validation:

- Check whether `benificial\_cost\_mark` is a list or a NumPy array.

- Ensure that the array has one dimension and its length matches the number of criteria.

- Verify that every value in the array is either 0 or 1.

These checks ensure that the input parameters provided for COPRAS analysis are valid and consistent, preventing errors during computation. If any of the checks fail, a `ValueError` is raised with an appropriate error message. If all checks pass, a message indicating "Everything is fine" is displayed.

Copras code:



Here are the actions performed in the `COPRAS` class:

1. Initialization:

- The class inherits from `Check\_parameters\_COPRAS` to validate the input parameters.

- The constructor initializes the attributes with the provided input parameters.

2. Attribute Initialization:

- Attributes such as `rank\_array`, `normalized\_matrix`, `weighted\_matrix`, `bi`, `ci`, `qi`, and `udi` are initialized to `None`.

3. Computation (solve method):

- Normalized the input data matrix.

- Multiplied the normalized matrix with the criteria weights to obtain the weighted matrix.

- Calculated the sums of benefit (bi) and cost (ci) criteria for each alternative.

- Computed the relative significance (qi) for each alternative.

- Calculated the utility degree (udi) for each alternative.

- Converted the udi into a rank array for ranking alternatives.

4. Displaying Results (show method):

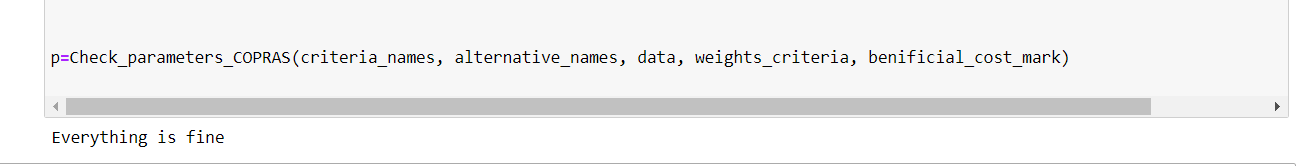
- If `show\_rank\_array` is set to `True`, prints the rank array along with ranks.

- Identifies and prints the most suitable alternative based on the highest utility degree.

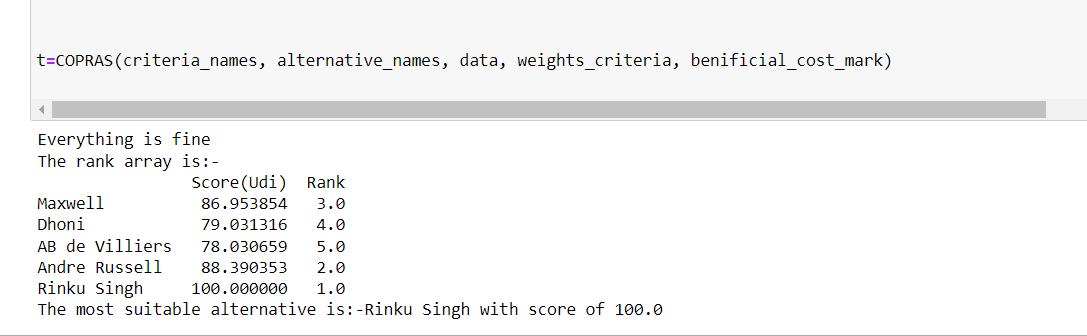
These actions collectively perform the COPRAS analysis on the provided input data and criteria, providing the user with the rank array and the most suitable alternative based on the analysis.

**5.3 Results**

Check\_parameters\_COPRAS-



COPRAS result-



Other attributes-

