AN INTEGRATED MCDM APPROACH FOR CLOUD SERVICE SELECTION BASED ON TOPSIS AND BWM

A PROJECT REPORT

Submitted

in the partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

DHEERAJ KALAKOTA (19B81A0518)
MUTHYALA LOKESH REDDY (19B81A0520)
HARSHITH MAKKAPATI (19B81A0523)

Under the guidance of
Mrs. V.N.V.L.S SWATHI
Senior Assistant Professor, CSE Department

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CVR COLLEGE OF ENGINEERING

(An Autonomous institution, NBA, NAAC Accredited and Affiliated to JNTUH, Hyderabad)

Vastunagar, Mangalpalli (V), Ibrahimpatnam
(M), Rangareddy (D), Telangana- 501 510
October 2022

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance has been a source of inspiration throughout the course of the project.

It is a great pleasure to convey our profound sense of gratitude to our principal **Dr. K. Ramamohan Reddy**, Vice-Principal Prof. **L. C. Siva Reddy, Dr. A. Vani Vathsala**, Head of CSE Department, CVR College Of Engineering, for havingbeen kind enough to arrange necessary facilities for executing the project in the college.

We deem it a pleasure to acknowledge our sense of gratitude to our project guide **Mrs. V.N.V.L.S. Swathi** under whom we have carried out the project work. His incisive and objective guidance and timely advice encouraged us with constant flow of energy to continue the work.

We wish a deep sense of gratitude and heartful thanks to the management for providing excellent lab facilities and tools. Finally, we thank all those whose guidance helped us in this regard.

DHEERAJ KALAKOTA (19B81A0518)
MUTHYALA LOKESH REDDY (19B81A0520)
HARSHITH MAKKAPATI (19B81A0523)

DECLARATION

We the undersigned solemnly declare that the project report titled '**AN INTEGRATED MCDM APPROACH FOR SERVICE SELECTION BASED ON TOPSIS AND BWM**' is based on our own work carried out during the course of our study under the supervision of **Mrs. V.N.V.L.S. SWATHI,** CVR College of Engineering.

We assert the statements made and conclusions are drawn are an outcome of our research work. We further certify that

1. The work contained in the report is original and has been done by us under the general supervision of our supervisor.

2. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or in the any other University of India or abroad.

3. We have followed the guidelines provided by the university in writing the report.

CERTIFICATE

This is to certify that the project titled **‘AN INTEGRATED MCDM APPROACH FOR SERVICE SELECTION BASED ON TOPSIS AND BWM’** is being submitted by **DHEERAJ KALAKOTA (19B81A0518), MUTHYALA LOKESH REDDY (19B81A0520)**, **HARSHITH MAKKAPATI (19B81A0523)** in partial fulfillment for the award ofthe degree of Bachelor of Technology in Computer Science and Engineering to the CVRCollege of Engineering, is a record bonafide work carried out by them under my guidance and supervision during the year 2020-2021. The results embodied in this project work have not been submitted to any other University or Institution for the award of any degree or diploma.

Signature of the project guide,
Mrs. V.N.V.L.S. Swathi
Senior Assistant Professor
CSE Department

Signature of the HOD,
Dr. A. Vani Vathsala
Head of Department (CSE)
CVR College of Engineering

ABSTRACT

Cloud computing has become increasingly popular and growing rapidly since it provides variety of computational services. Given the vast diversity of these services, the choice of the most appropriate Cloud Service Provider (CSP) become a great dilemma. Due to this we have to precisely evaluate services by several CSPs. The selection of the best CSP is thus a complex Multi Criteria Decision Making (MCDM) needs to be adjusted efficiently. In this project we propose a MCDM approach that is feasible, efficient and consistent using relative preferences of criteria and alternatives. The proposed approach incorporates Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) and the Best Worst Method (BWM).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

CC	Cloud Computing
CSP	Cloud Service Provider
MCDM	Multi Criteria Decision Making
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution
BWM	Best Worst Method
AHP	Analytical hierarchical process.
SLA	Service Level Agreement
QoS	Quality of Service.

1. INTRODUCTION

1.1 INTRODUCTION

One of the most recent developments within Computer science is Cloud Computing which provides services related to computer, allowing users/customers the power of computing, data storage, deployment models, infrastructures, platforms and resources on pay – per – use basis. Its focus is to provide secure, scalable, reliable, sustainable, fault tolerant and sharing of data through web base applications. The agent is a type of software that works on the behalf of its owners. Intelligent agent allows people to delegate their work to them, and they do repetitive tasks, learning, compute complex data and make decisions.

Cloud Computing (CC) has become a promising choice for businesses to replace the on premise IT infrastructure. It has changed our understanding of how to procure computing resources with high versatility, availability and minimum management effort. As a result, companies can now concentrate on their core functions leaving Cloud Service Providers (CSPs) to handle their computing assets. CSPs are vendors who lease to their customers different types of services (e.g., IaaS, PaaS, SaaS) that are dynamically provisioned based on customer's demand in a pay-as-you-go basis. The relationship between the customers and CSPs is organized according to a certain contract called Service Level Agreement (SLA).

1.2 MOTIVATION

With the rapid growth of cloud technology, more and more IT services providers compete to offer high-quality and cost-effective cloud services that best fulfil their customers' needs. Given the vast diversity of these offers, the choice of the most appropriate Cloud Service Provider (CSP) became a dilemma that confuses most cloud customers. Many diverged criteria have to be considered to precisely evaluate services offered by several CSPs, some of these criteria cannot be quantified easily such as usability and security. The selection of the best CSP is thus a complex Multi Criteria Decision Making (MCDM) problem that needs to be addressed efficiently. Previous studies of this problem employed MCDM methods that are either unfeasible when it is difficult or meaningless to quantify alternatives over criteria or computationally expensive and inconsistent when relative preferences of alternatives and criteria are used instead. In this paper, we propose a novel MCDM approach that is feasible, efficient and consistent using relative preferences of criteria and alternatives. The proposed approach incorporates Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) and the Best Worst Method (BWM) to rank CSPs using evaluation criteria characterizing their services. The integrated approach has been tested and validated through a use-case scenario which demonstrates its effectiveness and correctness. We have also compared the proposed approach to the most commonly used MCDM approach.

1.3 PROBLEM STATEMENT

- The project is 'An integrated MCDM approach for service selection based on TOPSIS and BWM'. This is a python project software that provides users the facility to make a multi criteria decision.

- This project will take the QoS factors of the criteria of different alternatives as the input for which we have to make a decision.

- The output will be the best and worst rankings for different alternatives.

1.4 PROJECT OBJECTIVES

The objective of the project is composed of TOPSIS method, which consists of three steps. They are as follows:

1. Identify the criteria to be used in the model
2. weigh the criteria by using BW method
3. Evaluation of alternatives with TOPSIS and determination of the final rank

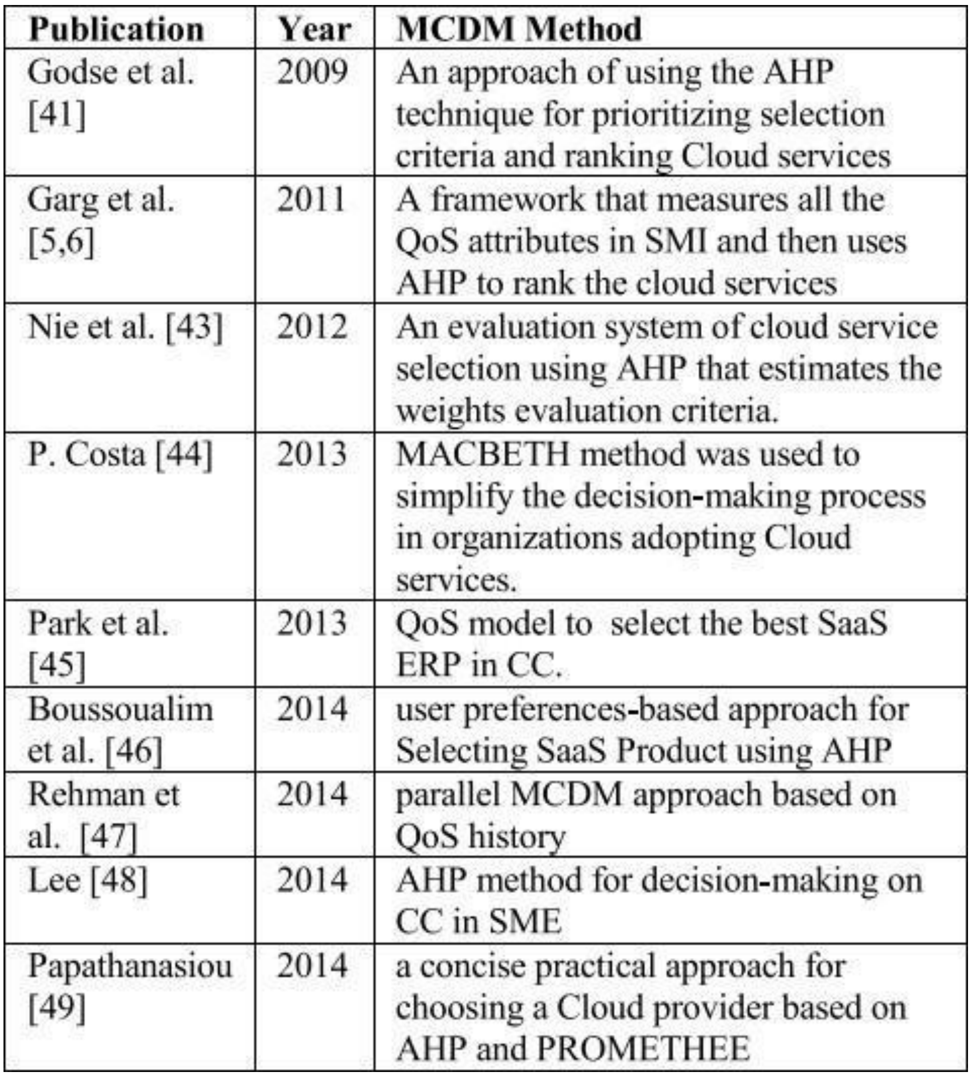
1.5 PROJECT REPORT ORGANISATION

1. This report is divided into 6 chapters after this introductory chapter.
2. Chapter 3 summarizes functional, non-functional requirements and system requirements along with software and hardware specifications.
3. Chapter 4 deals with analysis and design of the proposed model which includes use case diagram etc.
4. Chapter 5 encloses Implementation and testing of the proposed model and testing with different scenarios.
5. Chapter 6 includes conclusion and future work.
6. Chapter 7 includes reference.

2. LITERATURE SURVEY

1.2 LITERATURE SURVEY

In recent years, many research efforts have been conducted to solve the problem of cloud service selection, some of them use the MAUT techniques and others use the pairwise comparison methods. Nevertheless, more hybrid approaches are being introduced using many simple MCDM techniques. Hybrid approaches improve consumer trust and help make more accurate final decisions.



Publication	Year	MCDM Method
Godse et al. [41]	2009	An approach of using the AHP technique for prioritizing selection criteria and ranking Cloud services
Garg et al. [5,6]	2011	A framework that measures all the QoS attributes in SMI and then uses AHP to rank the cloud services
Nie et al. [43]	2012	An evaluation system of cloud service selection using AHP that estimates the weights evaluation criteria.
P. Costa [44]	2013	MACBETH method was used to simplify the decision-making process in organizations adopting Cloud services.
Park et al. [45]	2013	QoS model to select the best SaaS ERP in CC.
Boussoualim et al. [46]	2014	user preferences-based approach for Selecting SaaS Product using AHP
Rehman et al. [47]	2014	parallel MCDM approach based on QoS history
Lee [48]	2014	AHP method for decision-making on CC in SME
Papathanasiou [49]	2014	a concise practical approach for choosing a Cloud provider based on AHP and PROMETHEE

Figure 1: Literature Survey

3. SOFTWARE AND HARDWARE SPECIFICATION

3.1 Functional/Non-Functional Requirements

3.1.1 Functional Requirements

1. This software shall accept the particular csv dataset from Service Measurement Index (SMI).
2. This software shall apply TOPSIS approach to get similar and alternative solutions for the cloud service.
3. We apply BWM to rank these similar and alternative solutions which we got from TOPSIS approach.
4. We will also verify our approach by using Analytical Hierarchical Process (AHP).

3.1.2 Non-Functional Requirements

1. Usability: It defines the user interface of the software in terms of simplicity of understanding the user interface of stock prediction software, for any kind of stock trader and other stakeholders in stock market.
2. Efficiency: Maintaining the possible highest accuracy in the closing stock prices in shortest time with available data.
3. Performance: It is a quality attribute of the stock prediction software that describes the responsiveness to various user interactions with it.

3.2 SOFTWARE REQUIREMENT

The following few tools have been used to perform this analysis

1. Python IDE.
2. AWS
3. Jupyter
4. Python Libraries:
   i. Numpy
   ii. Pandas
   iii. Matplotlib
   iv. Sklearn

3.3 HARDWARE REQUIREMENTS

1. Machine with Windows or Linux platform
2. RAM 4 GB or above
3. GPU for training model
4. CPU: 2 GHz or faster
5. Architecture: 32-bit or 64-bit

4. DESIGN

4.1 USE CASE DIAGRAM

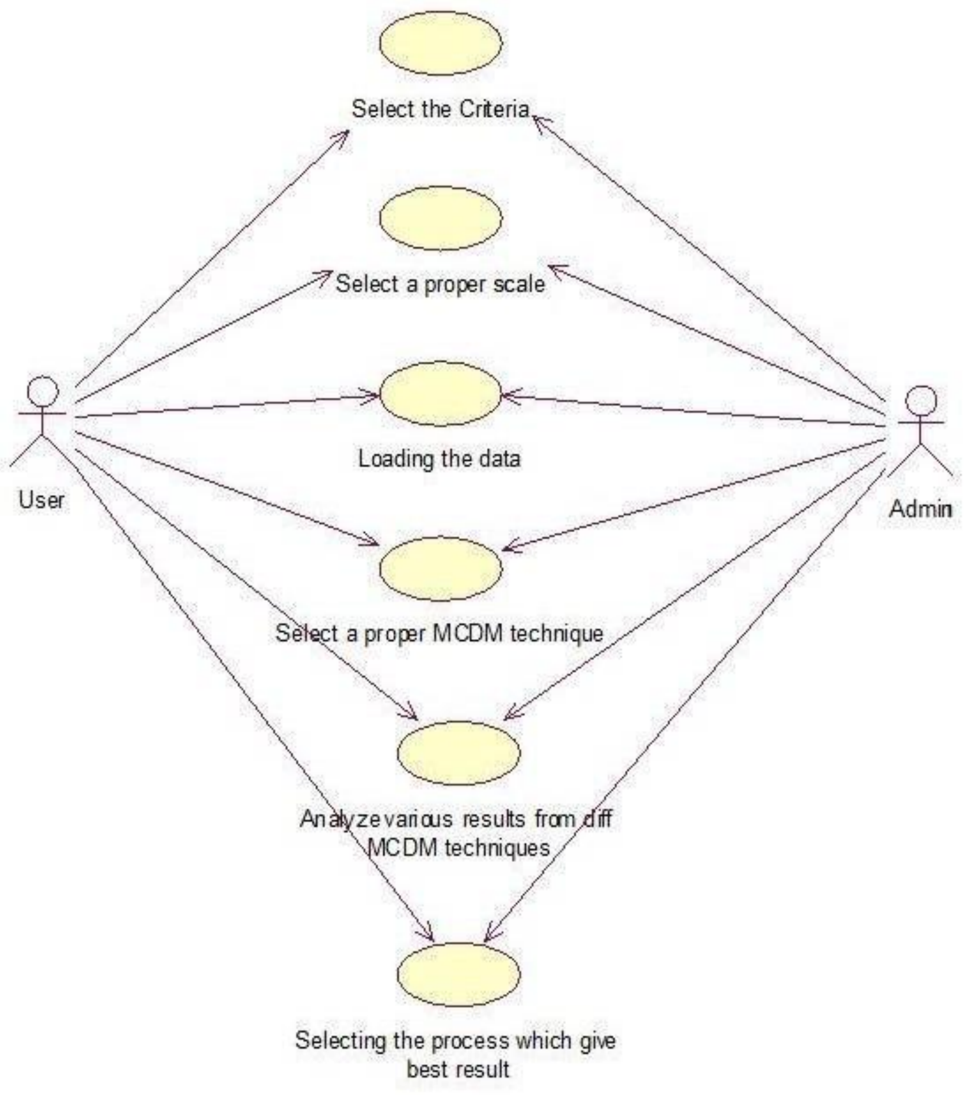

Select the Criteria
Select a proper scale
Loading the data
User
Admin
Select a proper MCDM technique
Analyze various results from diff MCDM techniques
Selecting the process which give best result

Figure 2: Use Case Diagram

In our Use Case Diagram, we have two actors namely user or client and admin. The use case diagram shows the set of use case actor and their relationships. The actor perform the respective use cases mentioned above. The admin plays a role of giving access to data and algorithms to apply.

4.2 CLASS DIAGRAM

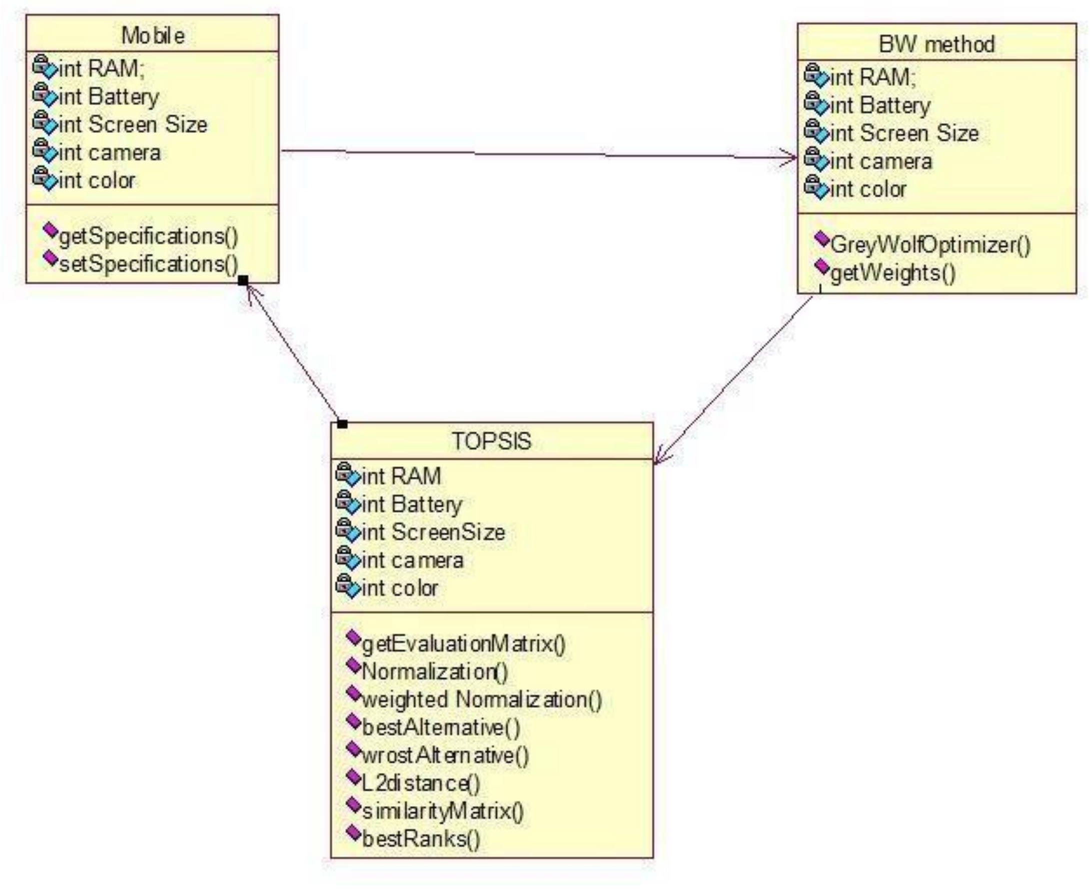


Figure 3: Class Diagram

The above class diagram depicts all the class, objects and functions of the system deployed on the TOPSIS and BW method.

4.3 ACTIVITY DIAGRAM

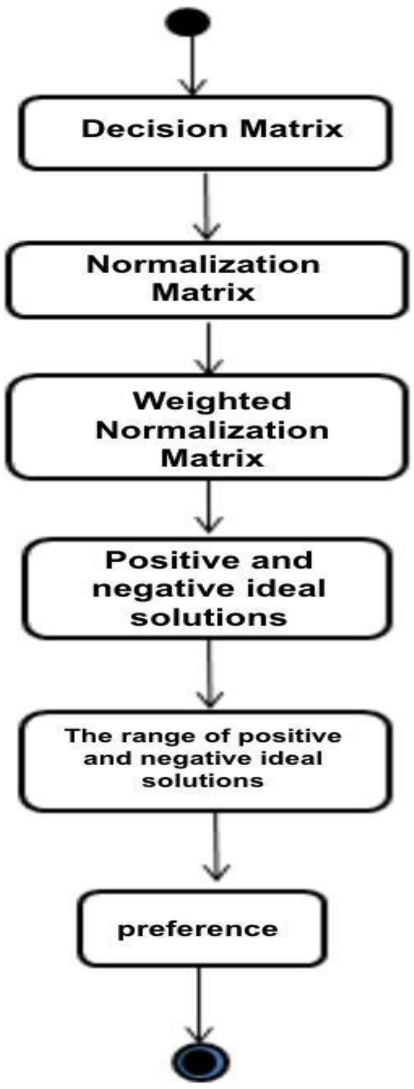
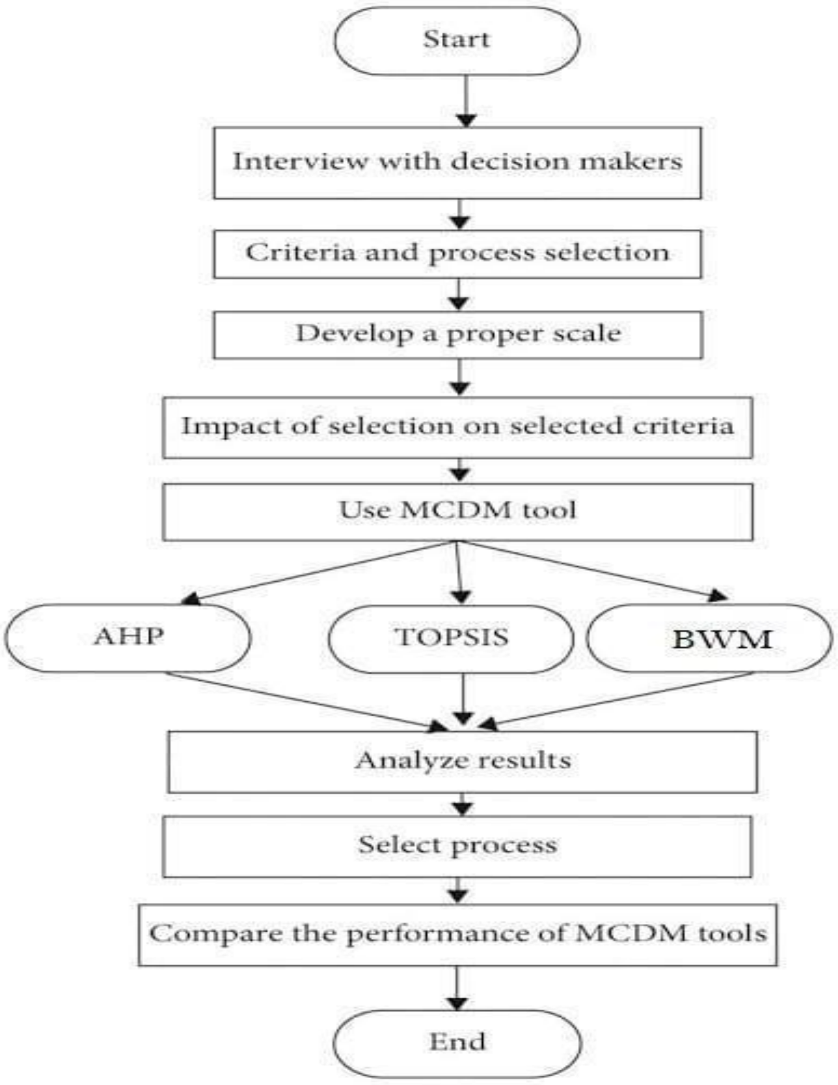

Decision Matrix
Normalization Matrix
Weighted Normalization Matrix
Positive and negative ideal solutions
The range of positive and negative ideal solutions
preference

Figure 4: Activity Diagram

At first we will take the input of evaluation matrix i.e. decision matrix, then we get weights from BW method. Then we will normalize the matrix and we will multiply with the weights obtained from the BW method which gives us the weighted normalized matrix. Now we find the positive and negative ideal solutions and we will find Euclidian distances which will give us the range of positive and negative ideal solutions. Next we will rank the different alternatives and select them according to the preferences.

4.4 ARCHITECTURE DIAGRAM



Start
Interview with decision makers
Criteria and process selection
Develop a proper scale
Impact of selection on selected criteria
Use MCDM tool
AHP
TOPSIS
BWM
Analyze results
Select process
Compare the performance of MCDM tools
End

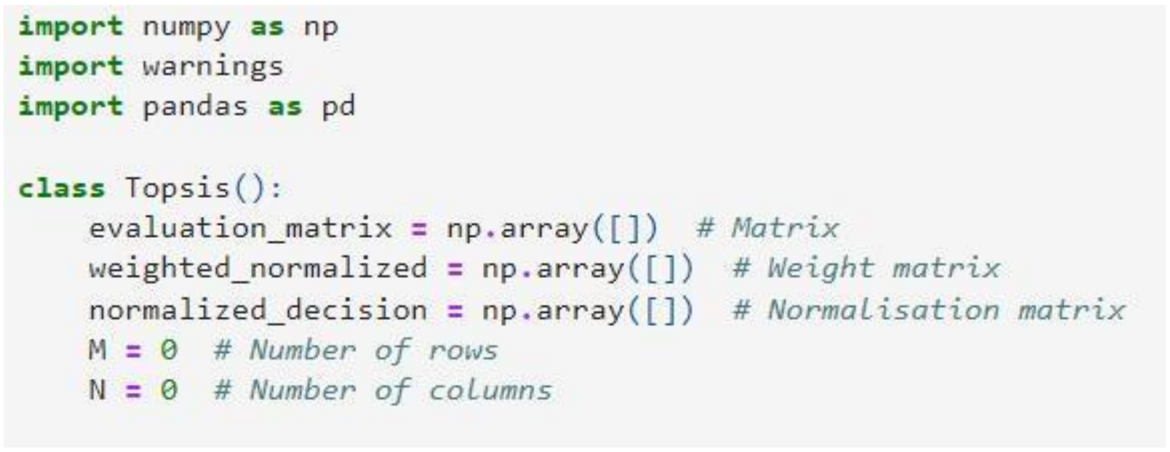
Figure 5: Architecture Diagram

First we have to Identify the objective/goal. Then we should Select Criteria/ Parameters/ Factors/ Decider. Now we have to look for different alternatives. Now we have to choose a proper MCDM technique. Now we have aggregate and analyze the result. Decision making based on the Aggregation results.

5. IMPLEMENTATION AND TESTING

5.1 IMPLEMENTATION

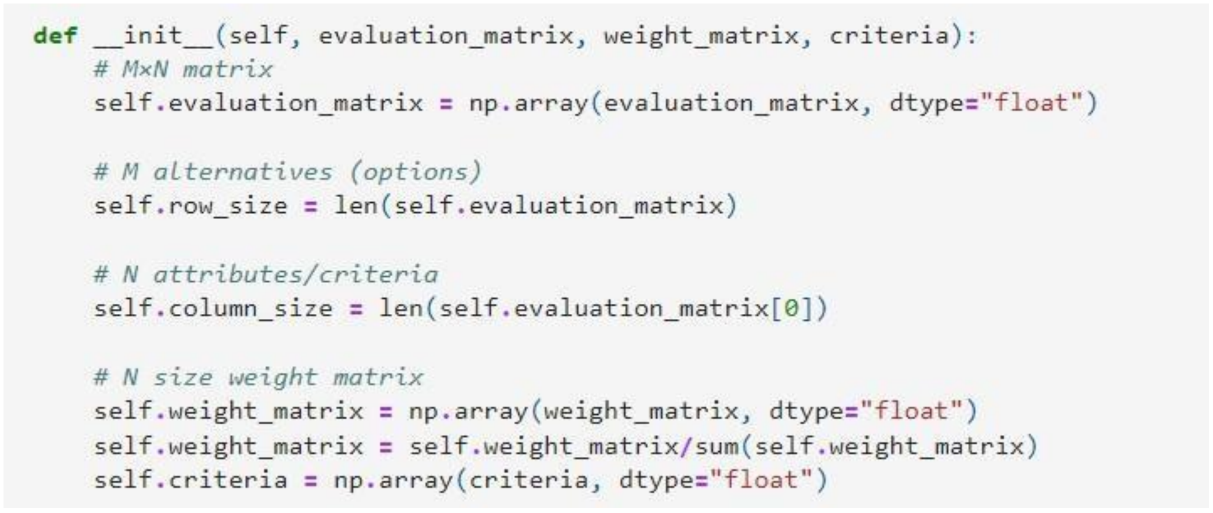
TOPSIS:



```
import numpy as np
import warnings
import pandas as pd

class Topsis():
    evaluation_matrix = np.array([])  # Matrix
    weighted_normalized = np.array([])  # Weight matrix
    normalized_decision = np.array([])  # Normalisation matrix
    M = 0  # Number of rows
    N = 0  # Number of columns
```

Figure 6: TOPSIS - Initialization

Step 1: Create a evaluation matrix consisting of m alternatives and n criteria with the intersection of each alternative and criteria

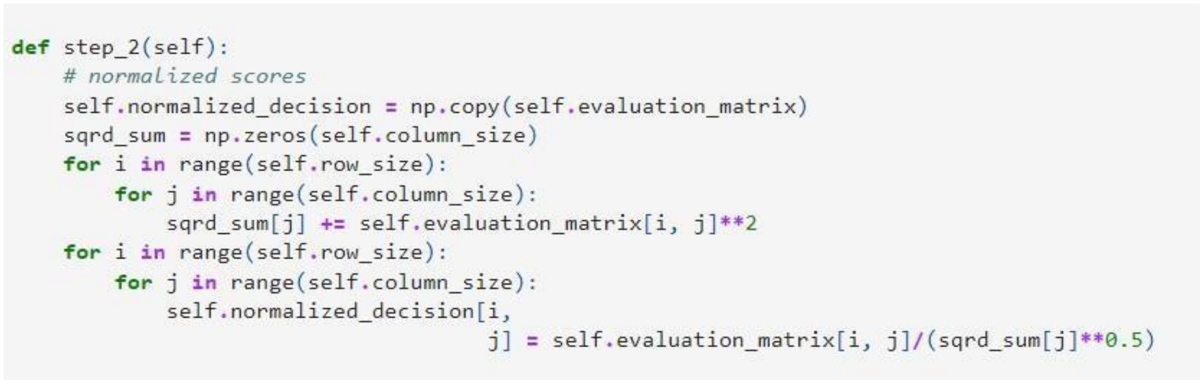


```
def __init__(self, evaluation_matrix, weight_matrix, criteria):
    # M×N matrix
    self.evaluation_matrix = np.array(evaluation_matrix, dtype="float")

    # M alternatives (options)
    self.row_size = len(self.evaluation_matrix)

    # N attributes/criteria
    self.column_size = len(self.evaluation_matrix[0])

    # N size weight matrix
    self.weight_matrix = np.array(weight_matrix, dtype="float")
    self.weight_matrix = self.weight_matrix/sum(self.weight_matrix)
    self.criteria = np.array(criteria, dtype="float")
```

Figure 7: TOPSIS – Step 1

Step 2: Calculating Normalized Matrix: We normalize each value by making it: where m is the number of rows in the dataset and n is the number of columns. I vary along rows and j varies along the column.

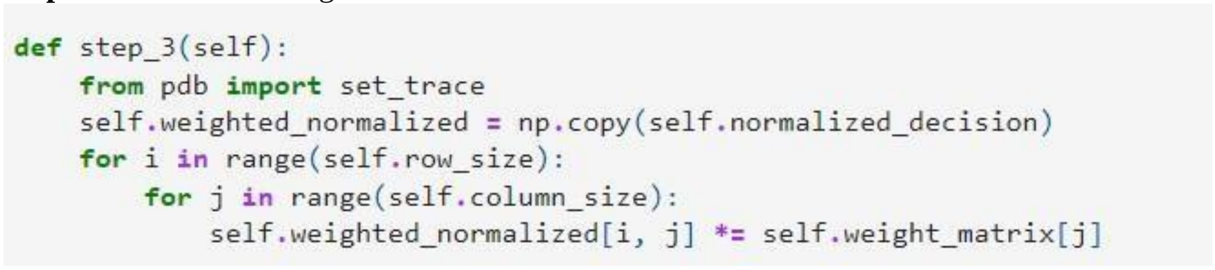
$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{k=1}^{m} x_{kj}^2}},$$
$$i = 1, 2, \ldots, m,$$
$$j = 1, 2, \ldots, n$$



```python
def step_2(self):
    # normalized scores
    self.normalized_decision = np.copy(self.evaluation_matrix)
    sqrd_sum = np.zeros(self.column_size)
    for i in range(self.row_size):
        for j in range(self.column_size):
            sqrd_sum[j] += self.evaluation_matrix[i, j]**2
    for i in range(self.row_size):
        for j in range(self.column_size):
            self.normalized_decision[i,
                                     j] = self.evaluation_matrix[i, j]/(sqrd_sum[j]**0.5)
```

Figure 8: TOPSIS – Step 2

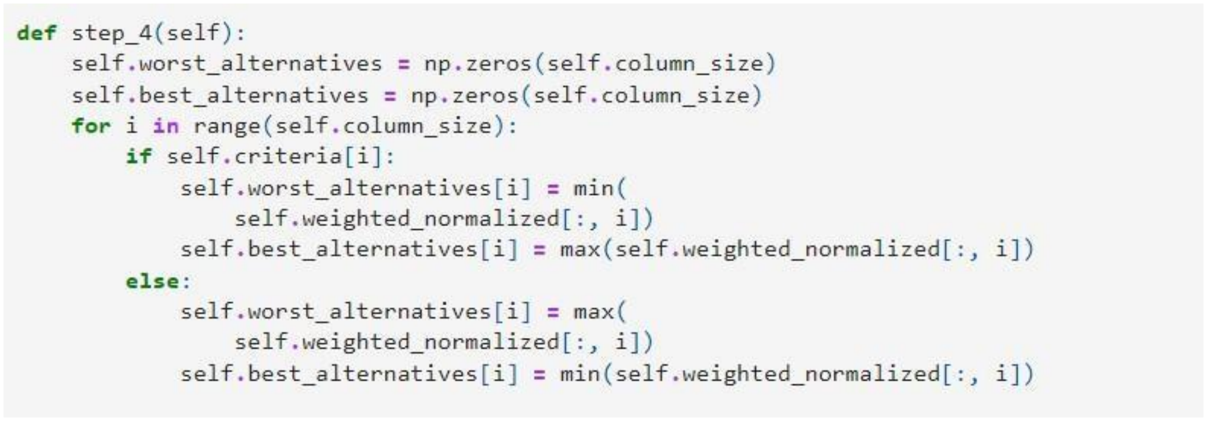
Step3: Calculate the weighted normalised decision matrix



```python
def step_3(self):
    from pdb import set_trace
    self.weighted_normalized = np.copy(self.normalized_decision)
    for i in range(self.row_size):
        for j in range(self.column_size):
            self.weighted_normalized[i, j] *= self.weight_matrix[j]
```

Figure 9: TOPSIS – Step 3

Step 4: Calculating Ideal Best and Ideal worst and Euclidean distance for each row from ideal worst and ideal best value. First, we will find out the ideal best and ideal worst value: Now here we need to see the impact, i.e. is it ‘+’ or ‘-‘ impact. If ‘+’ impact Ideal best for a column is the maximum value in that column and the ideal worst is the minimum value in that column, and vice versa for the ‘-‘ impact.

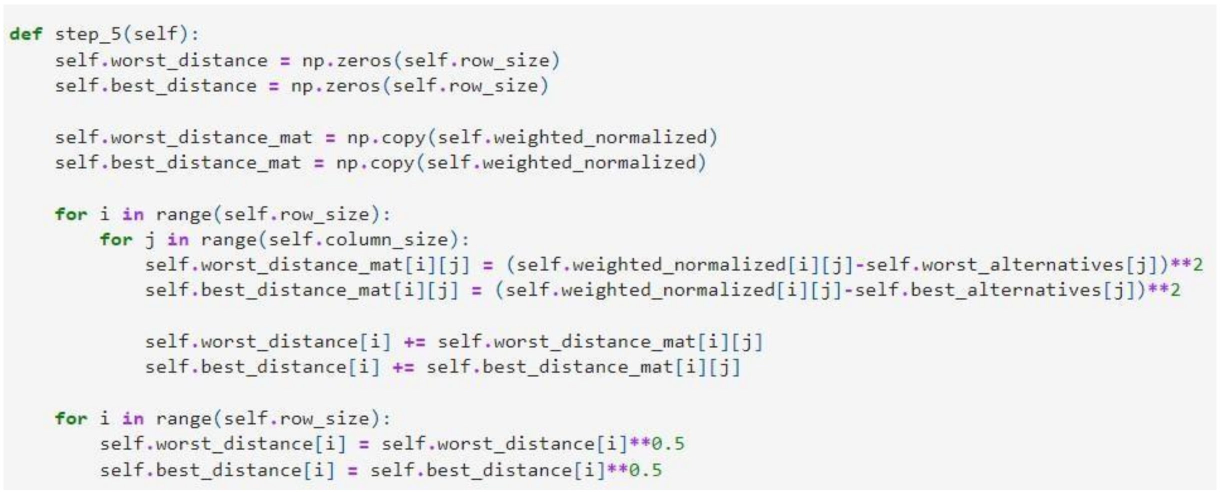


```
def step_4(self):
    self.worst_alternatives = np.zeros(self.column_size)
    self.best_alternatives = np.zeros(self.column_size)
    for i in range(self.column_size):
        if self.criteria[i]:
            self.worst_alternatives[i] = min(
                self.weighted_normalized[:, i])
            self.best_alternatives[i] = max(self.weighted_normalized[:, i])
        else:
            self.worst_alternatives[i] = max(
                self.weighted_normalized[:, i])
            self.best_alternatives[i] = min(self.weighted_normalized[:, i])
```

Figure 10: TOPSIS – Step 4

Step 5: Now we need to calculate Euclidean distance for elements in all rows from the ideal best and ideal worst, Here diw is the worst distance calculated of an ith row, where ti,j is element value and tw,j is the ideal worst for that column. similarly, we can find dib, i.e. best distance calculated on an ith row.

$$d_{iw} = \sqrt{\sum_{j=1}^{n} (t_{ij} - t_{wj})^2}, \quad i = 1, 2, \ldots, m,$$



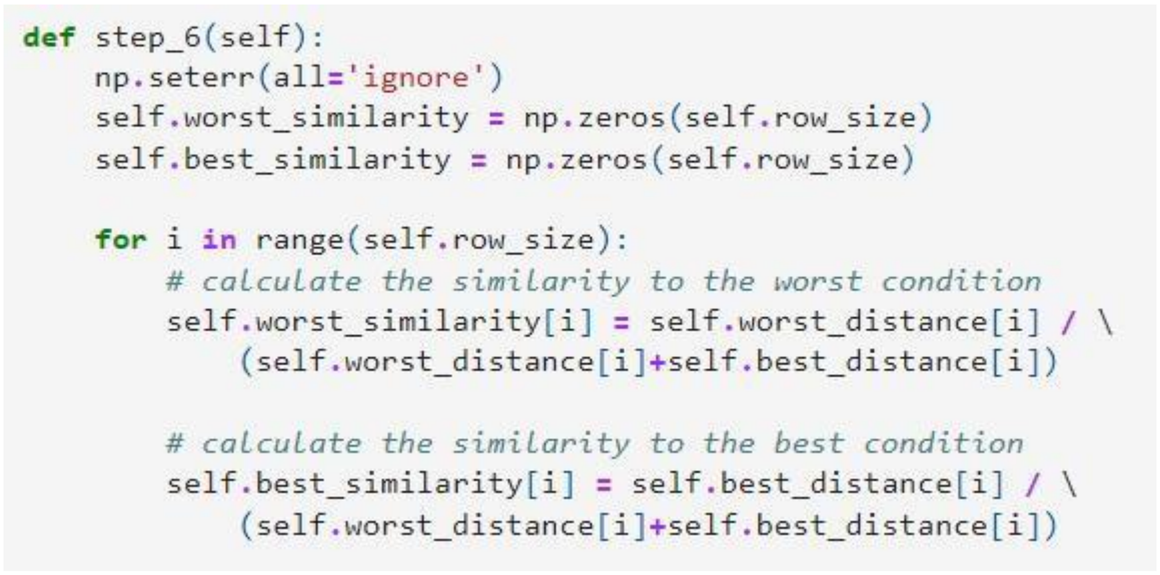
```
def step_5(self):
    self.worst_distance = np.zeros(self.row_size)
    self.best_distance = np.zeros(self.row_size)

    self.worst_distance_mat = np.copy(self.weighted_normalized)
    self.best_distance_mat = np.copy(self.weighted_normalized)

    for i in range(self.row_size):
        for j in range(self.column_size):
            self.worst_distance_mat[i][j] = (self.weighted_normalized[i][j]-self.worst_alternatives[j])**2
            self.best_distance_mat[i][j] = (self.weighted_normalized[i][j]-self.best_alternatives[j])**2

            self.worst_distance[i] += self.worst_distance_mat[i][j]
            self.best_distance[i] += self.best_distance_mat[i][j]

    for i in range(self.row_size):
        self.worst_distance[i] = self.worst_distance[i]**0.5
        self.best_distance[i] = self.best_distance[i]**0.5
```

Figure 11: TOPSIS – Step 5

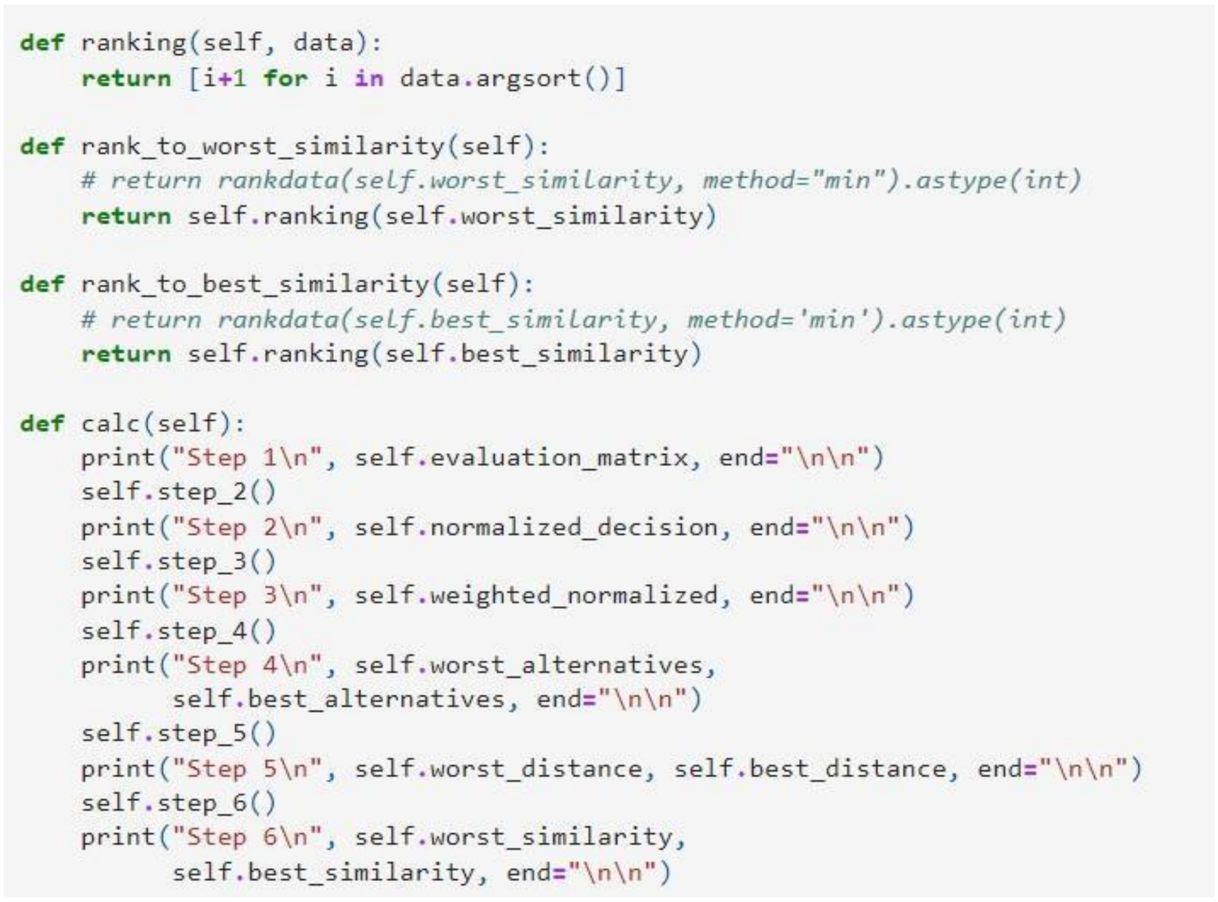
Step 6: Calculating Topsis Score and Ranking. Now we have Distance positive and distance negative with us, let's calculate the Topsis score for each row on basis of them.

TOPSIS Score = diw / (dib + diw) for each row

Now rank according to the TOPSIS score, i.e. higher the score, better the rank



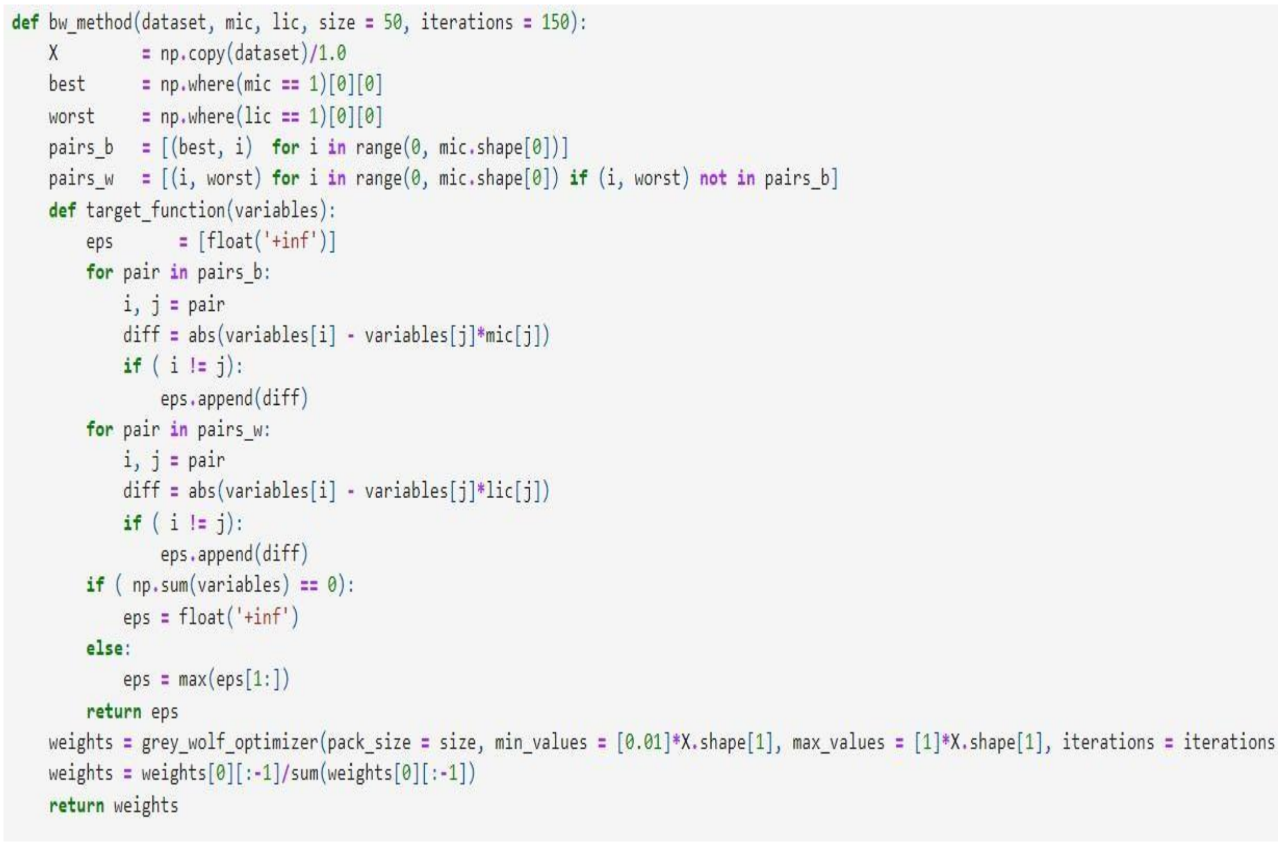
```
def step_6(self):
    np.seterr(all='ignore')
    self.worst_similarity = np.zeros(self.row_size)
    self.best_similarity = np.zeros(self.row_size)

    for i in range(self.row_size):
        # calculate the similarity to the worst condition
        self.worst_similarity[i] = self.worst_distance[i] / \
            (self.worst_distance[i]+self.best_distance[i])

        # calculate the similarity to the best condition
        self.best_similarity[i] = self.best_distance[i] / \
            (self.worst_distance[i]+self.best_distance[i])
```



```
def ranking(self, data):
    return [i+1 for i in data.argsort()]

def rank_to_worst_similarity(self):
    # return rankdata(self.worst_similarity, method="min").astype(int)
    return self.ranking(self.worst_similarity)

def rank_to_best_similarity(self):
    # return rankdata(self.best_similarity, method='min').astype(int)
    return self.ranking(self.best_similarity)

def calc(self):
    print("Step 1\n", self.evaluation_matrix, end="\n\n")
    self.step_2()
    print("Step 2\n", self.normalized_decision, end="\n\n")
    self.step_3()
    print("Step 3\n", self.weighted_normalized, end="\n\n")
    self.step_4()
    print("Step 4\n", self.worst_alternatives,
          self.best_alternatives, end="\n\n")
    self.step_5()
    print("Step 5\n", self.worst_distance, self.best_distance, end="\n\n")
    self.step_6()
    print("Step 6\n", self.worst_similarity,
          self.best_similarity, end="\n\n")
```

Figure 12: TOPSIS – Step 6

BWM:

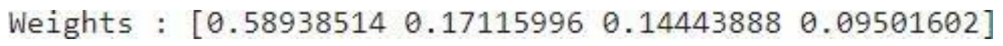


```python
def bw_method(dataset, mic, lic, size = 50, iterations = 150):
    X        = np.copy(dataset)/1.0
    best     = np.where(mic == 1)[0][0]
    worst    = np.where(lic == 1)[0][0]
    pairs_b  = [(best, i)  for i in range(0, mic.shape[0])]
    pairs_w  = [(i, worst) for i in range(0, mic.shape[0]) if (i, worst) not in pairs_b]
    def target_function(variables):
        eps       = [float('+inf')]
        for pair in pairs_b:
            i, j = pair
            diff = abs(variables[i] - variables[j]*mic[j])
            if ( i != j):
                eps.append(diff)
        for pair in pairs_w:
            i, j = pair
            diff = abs(variables[i] - variables[j]*lic[j])
            if ( i != j):
                eps.append(diff)
        if ( np.sum(variables) == 0):
            eps = float('+inf')
        else:
            eps = max(eps[1:])
        return eps
    weights = grey_wolf_optimizer(pack_size = size, min_values = [0.01]*X.shape[1], max_values = [1]*X.shape[1], iterations = iterations
    weights = weights[0][:-1]/sum(weights[0][:-1])
    return weights
```

Figure 13: BW method

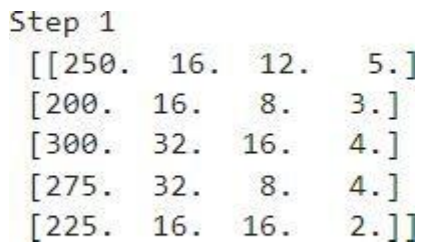
In pairwise comparison-based methods we either have methods for which we use a single vector or a full matrix. Although using one vector for the input data makes the method very data-efficient, the main weakness of methods based on only one vector is that the consistency of the provided pairwise comparisons cannot be checked. On the other hand, although using a full matrix provides the possibility of checking the consistency of the provided pairwise comparisons, methods which are based on full pairwise comparison matrix are not data-efficient. Asking too many questions from the DM, which occurs in the case of full matrix, might even contribute to the confusion and inconsistency of the DM. BWM stands in the middle. That is to say, it is the most data-efficient method which could, at the same time, provide the possibility of checking the consistency of the provided pairwise comparisons. As the two vectors are formed with considering two specific reference criteria, BWM should not be seen as a case of incomplete pairwise comparison matrix.

5.2 TESTING:



```
Weights : [0.58938514 0.17115996 0.14443888 0.09501602]
```

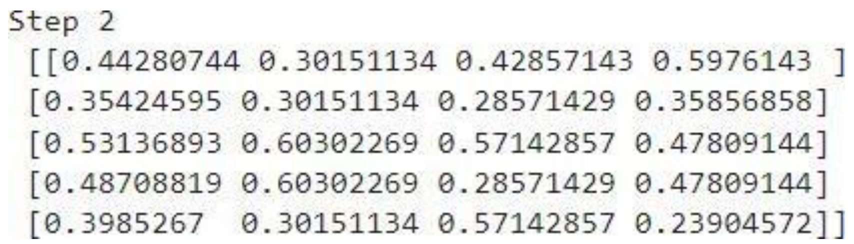
These weights are calculated using the BW(best worst) method.



```
Step 1
 [[250.  16.  12.   5.]
 [200.  16.   8.   3.]
 [300.  32.  16.   4.]
 [275.  32.   8.   4.]
 [225.  16.  16.   2.]]
```

Figure 14: Output-1

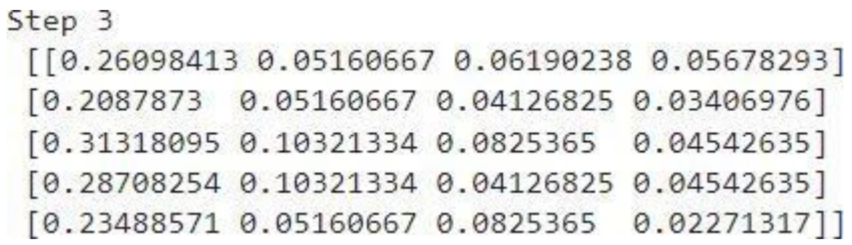
This is the evaluation matrix where columns are criteria and rows are alternatives.



```
Step 2
 [[0.44280744 0.30151134 0.42857143 0.5976143 ]
 [0.35424595 0.30151134 0.28571429 0.35856858]
 [0.53136893 0.60302269 0.57142857 0.47809144]
 [0.48708819 0.60302269 0.28571429 0.47809144]
 [0.3985267  0.30151134 0.57142857 0.23904572]]
```

Figure 15: Output-2

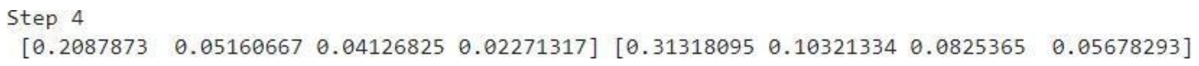
This is the normalized matrix.



```
Step 3
 [[0.26098413 0.05160667 0.06190238 0.05678293]
 [0.2087873  0.05160667 0.04126825 0.03406976]
 [0.31318095 0.10321334 0.0825365  0.04542635]
 [0.28708254 0.10321334 0.04126825 0.04542635]
 [0.23488571 0.05160667 0.0825365  0.02271317]]
```

Figure 16: Output-3

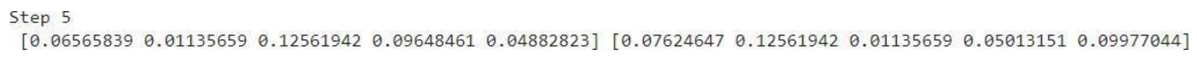
This is the weighted normalized matrix.



```
Step 4
 [0.2087873  0.05160667 0.04126825 0.02271317] [0.31318095 0.10321334 0.0825365  0.05678293]
```

Figure 17: Output-4

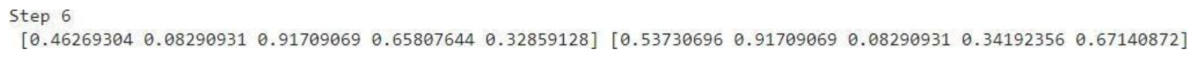
This is best and worst alternative.



```
Step 5
 [0.06565839 0.01135659 0.12561942 0.09648461 0.04882823] [0.07624647 0.12561942 0.01135659 0.05013151 0.09977044]
```

Figure 18: Output-5

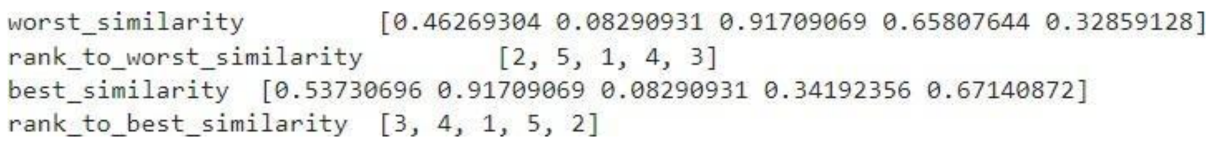
This is the Euclidean distance.



```
Step 6
 [0.46269304 0.08290931 0.91709069 0.65807644 0.32859128] [0.53730696 0.91709069 0.08290931 0.34192356 0.67140872]
```

Figure 19: Output-6

This is the TOPSIS similarity vector.



```
worst_similarity         [0.46269304 0.08290931 0.91709069 0.65807644 0.32859128]
rank_to_worst_similarity         [2, 5, 1, 4, 3]
best_similarity  [0.53730696 0.91709069 0.08290931 0.34192356 0.67140872]
rank_to_best_similarity  [3, 4, 1, 5, 2]
```

Figure 20: Output-7

This is the rankings of best and worst similarities.

6. CONCLUSION & FUTURE ENHANCEMENT

6.1 CONCLUSION:

This project tells a novel MCDM approach that is feasible, efficient and consistent using relative preferences of criteria and alternatives. The proposed approach integrates Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) and Best Worst Method (BWM) to rank CSPs using evaluation criteria characterizing their services. BWM is used for acquiring the weights of criteria and relative scores for alternatives with respect to criteria. These acquired values are utilized by TOPSIS to rank the cloud services. The proposed approach has been tested and validated through a use-case scenario which demonstrates its effectiveness and correctness. We have compared the proposed method to the most commonly used MCDM method (i.e., AHP). The results clearly showed that our proposed approach outperforms AHP in terms of computational complexity and consistency; therefore, it is more efficient and more reliable.

6.2 FUTURE ENHANCEMENT:

The future work may be expanded as the integration of BWM with different MCDM methods like AHP, ELECTRE, PROMETHEE, Gray Theory, in the cloud service selection problem and other applications.

7. REFERENCES

1. B. Varghese and R. Buyya, ‘‘Next generation cloud computing: New trends and research directions,’’ Future Gener. Comput. Syst., vol. 79, pp. 849–861, Feb. 2018.
2. E. Youssef, ‘‘Exploring cloud computing services and applications,’’ J. Emerg. Trends Comput. Inf. Sci., vol. 3, no. 6, pp. 838–847, Jul. 2012.
3. Almishal and E. A. Youssef, ‘‘Cloud service providers: A comparative study,’’ Int. J. Comput. Appl. Inf. Technol., vol. 5, no. 2, pp. 46–52, May 2014.
4. N. Upadhyay, ‘‘Managing cloud service evaluation and selection,’’ Procedia Comput. Sci., vol. 122, pp. 1061–1068, 2017.
5. S. K. Garg, S. Versteeg, and R. Buyya, ‘‘A framework for ranking of cloud computing services,’’ Future Gener. Comput. Syst., vol. 29, no. 4, pp. 1012–1023, Jun. 2013.
6. S. K. Garg, S. Versteeg, and R. Buyya, ‘‘SMICloud: A framework for comparing and ranking cloud services,’’ in Proc. 4th IEEE Int. Conf. Utility Cloud Comput., Dec. 2011, pp. 210–218.
7. F. Nawaz, M. R. Asadabadi, N. K. Janjua, O. K. Hussain, E. Chang, and M. Saberi, ‘‘An MCDM method for cloud service selection using a Markov chain and the best-worst method,’’ Knowl.-Based Syst., vol. 159, pp. 120–131, Nov. 2018.
8. Z. Rehman, F. K. Hussain, and O. K. Hussain, ‘‘Towards multi-criteria cloud service selection,’’ in Proc. 5th Int. Conf. Innov. Mobile Internet Services Ubiquitous Comput., Jun. 2011, pp. 44–48.
9. Grgurević and G. Kordić, ‘‘Multi-criteria decision-making in cloud service selection and adoption,’’ in Proc. 5th Int. Virtual Res. Conf. Tech.