DEA\_Analysis

Pardeepan Subramaniam

2023-11-05

Statement :

Information technology should be mindful of the need for cost reduction sustainability. Data centers can consume energy on a scale comparable to that of industrial facilities and even small towns. Hence, it is essential to implement forward-thinking and transparent energy policies to enhance energy efficiency and overall operational performance. In this context, the use of Data Envelopment Analysis (DEA) to assess energy consumption and performance metrics is recommended. This analysis should encompass a diverse sample of data centers, including those with high and low demands, as well as medium-sized and large facilities. The findings from the DEA analysis will serve as a basis for recommending effective energy policies and will empower data-center managers to identify operational inefficiencies and take corrective actions.

Summary : Based on my examination and assessment, I have appraised the energy consumption and performance metrics of data centers with the aim of advancing sustainability and reducing costs.

During this procedure, it is imperative to gather data related to energy usage, performance metrics, and other necessary information for a comprehensive dataset that encompasses various demands and sizes. Defining input and output variables, such as the number of machines, shutdown operations, and energy consumption, becomes a crucial step. My choice of utilizing the "Benchmarking" package for Data Envelopment Analysis (DEA) for both natural DEA and constant returns to scale allows me to determine the efficiency score of data centers. When operations are optimized, the efficiency score reaches 1, while any score lower than 1 indicates inadequacy in the data. To address potential inefficiencies stemming from insufficient values, I have implemented specific techniques to pinpoint the precise requirements for achieving efficiency.

Loading the csv file

x=read.csv("C:\\Users\\spard\\OneDrive\\Desktop\\QMM\\Assignment\_3\\energy.csv")  
x

## X Energy.Policy Scheduling.Model Work.Load D.C..Size Shut.Downs  
## 1 1 Always Monolithic High 1000 37166  
## 2 2 Margin Mesos High 1000 13361  
## 3 3 Gamma Omega High 1000 14252  
## 4 4 Always Mono. Low 1000 36404  
## 5 5 Exponential Mesos Low 1000 19671  
## 6 6 Load Omega Low 1000 32407  
## 7 7 Margin Mono. High 5000 6981  
## 8 8 Gamma Mono. High 5000 9877  
## 9 9 Random Mesos High 5000 33589  
## 10 10 Margin Omega High 5000 8578  
## 11 11 Exponential Omega High 5000 11863  
## 12 12 Margin Omega Low 5000 15452  
## 13 13 Margin Mono. High 10000 9680  
## 14 14 Gamma Mono. High 10000 11388  
## 15 15 Margin Omega High 10000 18150  
## 16 16 Gamma Omega High 10000 18409  
## 17 17 Gamma Mesos Low 10000 29707  
## 18 18 Random Omega Low 10000 40772  
## Computing.Time..h. MWh.Consumed Queue.Time..ms.  
## 1 104.42 49.01 90.1  
## 2 104.26 49.65 1093.0  
## 3 104.17 49.60 0.1  
## 4 49.25 23.92 78.3  
## 5 49.63 24.65 1188.7  
## 6 49.34 24.19 1.1  
## 7 99.96 237.09 126.2  
## 8 99.96 235.92 129.8  
## 9 100.03 234.90 1122.6  
## 10 100.26 239.13 0.7  
## 11 100.26 236.95 1.0  
## 12 46.70 115.82 0.5  
## 13 101.56 481.36 325.2  
## 14 101.56 479.36 327.9  
## 15 101.63 486.11 2.6  
## 16 101.63 484.69 2.5  
## 17 45.83 228.31 1107.6  
## 18 46.09 233.50 3.8

library(Benchmarking)

## Loading required package: lpSolveAPI

## Loading required package: ucminf

## Loading required package: quadprog

library(lpSolveAPI)  
library(ucminf)  
library(quadprog)

Define the input and output variables for the DEA analysis

ip=x[,c("D.C..Size","Shut.Downs")]  
op =x[,c("Computing.Time..h.", "MWh.Consumed","MWh.Consumed")]

The examination is conducted using designated inputs and outputs. The CRS model posits that data centers are functioning at an ideal scale, or any deviations from this ideal scale can be attributed to inefficiencies. In this assessment, our attention can be directed towards the underperforming data centers, those with scores below 1) to identify areas for improvement.

d=dea(ip,op,RTS = "crs")  
d

## [1] 1.0000 1.0000 0.9991 0.4818 0.4965 0.4872 1.0000 0.9826 0.9578 1.0000  
## [11] 0.9806 0.4754 1.0000 0.9939 1.0000 0.9970 0.4687 0.4783

The ‘peers’ function is commonly employed to determine the peer units associated with each data center. These peer units serve as benchmarks or reference points for less efficient units to draw insights from. The ‘lambda’ function computes the relative weights assigned to these peers, indicating the extent to which the performance of benchmarking or reference units should be emulated in order to achieve efficiency.

print(d)

## [1] 1.0000 1.0000 0.9991 0.4818 0.4965 0.4872 1.0000 0.9826 0.9578 1.0000  
## [11] 0.9806 0.4754 1.0000 0.9939 1.0000 0.9970 0.4687 0.4783

peers(d) # It determines the peers For facilities 5,6, the peer units are 1,2,4.

## peer1 peer2 peer3  
## [1,] 1 NA NA  
## [2,] 2 NA NA  
## [3,] 1 2 NA  
## [4,] 2 NA NA  
## [5,] 2 NA NA  
## [6,] 2 NA NA  
## [7,] 7 NA NA  
## [8,] 2 10 13  
## [9,] 2 15 NA  
## [10,] 10 NA NA  
## [11,] 2 13 15  
## [12,] 2 15 NA  
## [13,] 13 NA NA  
## [14,] 2 13 15  
## [15,] 15 NA NA  
## [16,] 2 15 NA  
## [17,] 2 15 NA  
## [18,] 2 15 NA

d\_Weights <- lambda(d)   
#Determine the relative weights assigned to the peers. For facility 4, the weights are 0.20, 0.08, and 0.54. The facility 6 weights are 0.34, 0.39, and 0.13.  
  
d\_Weights

## L1 L2 L7 L10 L13 L15  
## [1,] 1.000000000 0.000000000 0 0.0000000 0.0000000 0.00000000  
## [2,] 0.000000000 1.000000000 0 0.0000000 0.0000000 0.00000000  
## [3,] 0.009970484 0.989150989 0 0.0000000 0.0000000 0.00000000  
## [4,] 0.000000000 0.481772407 0 0.0000000 0.0000000 0.00000000  
## [5,] 0.000000000 0.496475327 0 0.0000000 0.0000000 0.00000000  
## [6,] 0.000000000 0.487210473 0 0.0000000 0.0000000 0.00000000  
## [7,] 0.000000000 0.000000000 1 0.0000000 0.0000000 0.00000000  
## [8,] 0.000000000 0.220982865 0 0.5914729 0.1734861 0.00000000  
## [9,] 0.000000000 2.033467411 0 0.0000000 0.0000000 0.27553094  
## [10,] 0.000000000 0.000000000 0 1.0000000 0.0000000 0.00000000  
## [11,] 0.000000000 0.536265781 0 0.0000000 0.4082527 0.02840485  
## [12,] 0.000000000 0.262566735 0 0.0000000 0.0000000 0.21144095  
## [13,] 0.000000000 0.000000000 0 0.0000000 1.0000000 0.00000000  
## [14,] 0.000000000 0.006398513 0 0.0000000 0.8022310 0.19106871  
## [15,] 0.000000000 0.000000000 0 0.0000000 0.0000000 1.00000000  
## [16,] 0.000000000 0.022365412 0 0.0000000 0.0000000 0.99479451  
## [17,] 0.000000000 0.469111194 0 0.0000000 0.0000000 0.42175357  
## [18,] 0.000000000 0.937209877 0 0.0000000 0.0000000 0.38461980