Dataset Analysis

The calculation steps using the Simple Additive Weighting (SAW) method:  
1. Determining Alternative (Ai) 2. Determine the criteria to be used as a reference in decision making (Cj) 3. Determine the preference weight or level of importance (W) for each criterion 4. Determine the Match Value of each criterion 5. Make a decision matrix (x) obtained from the suitability rating for each alternative (Ai) with each criterion (Cj). 6. Perform the normalization step of the decision matrix (x) by calculating the value of the normalized performance rating (Rij) from the alternative (Ai) on the criteria (Cj) 7. The result of normalization (Rij) forms a normalized matrix (R) 8. The final result of the preference value (Vi) is obtained from the sum of the normalized matrix row elements (R) with the preference weights (W) corresponding to the matrix column elements (W). With: = rank for each alternative = weighted value of each criterion = normalized performance rating value.

# import csv  
import pandas as pd  
import numpy as np  
import pingouin as pg  
  
# read xls file  
df = pd.read\_excel('quiz\_20221.xlsx')  
df

|  | dosen | ta | nim | kdmk | nmmk | q1 | q2 | q3 | q4 | q5 | q6 | q7 | q8 | q9 | q10 | saran |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | Prof. Dr. Ir EDI NOERSASONGKO, M.Kom | 20221 | A11.2017.10120 | AF201703 | TECHNOPRENEURSHIP | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | NaN |
| 1 | Prof. Dr. Ir EDI NOERSASONGKO, M.Kom | 20221 | A11.2017.10350 | AF201703 | TECHNOPRENEURSHIP | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | NaN |
| 2 | Prof. Dr. Ir EDI NOERSASONGKO, M.Kom | 20221 | A11.2018.11309 | AF201703 | TECHNOPRENEURSHIP | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | NaN |
| 3 | Prof. Dr. Ir EDI NOERSASONGKO, M.Kom | 20221 | A11.2019.11618 | AF201703 | TECHNOPRENEURSHIP | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | ok |
| 4 | Prof. Dr. Ir EDI NOERSASONGKO, M.Kom | 20221 | A11.2019.11622 | AF201703 | TECHNOPRENEURSHIP | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | NaN |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 20092 | DIBYO ADI WIBOWO, S.Si., M.Si | 20221 | A11.2020.80018 | A11.54508 | STRATEGI ALGORITMA | 4 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | Tidak ada |
| 20093 | DIBYO ADI WIBOWO, S.Si., M.Si | 20221 | A11.2020.80018 | A11.54812 | METODE NUMERIK | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | Tidak ada |
| 20094 | DEWI PERGIWATI, S.Kom., M.Kom | 20221 | A11.2018.11461 | AF201704 | DASAR DASAR KOMPUTASI | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | NaN |
| 20095 | DEWI PERGIWATI, S.Kom., M.Kom | 20221 | A11.2021.13607 | AF201704 | DASAR DASAR KOMPUTASI | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 4 | NaN |
| 20096 | AYU ASHARI, S.S.T, M.Kes | 20221 | A11.2020.13081 | U201701 | DASAR KEWIRAUSAHAAN | 3 | 3 | 3 | 4 | 3 | 3 | 3 | 3 | 3 | 3 | NaN |

df2 = df.copy()  
  
for i in range(len(df2)):  
 df2.loc[i, 'nim'] = df2.loc[i, 'nim'] + str(" - ") + df2.loc[i, 'kdmk']  
  
# remove unnecessary columns  
df2 = df2.drop(['ta','dosen','saran','nmmk','kdmk'], axis=1)  
  
# change column name  
df2 = df2.rename(columns={'nim':'Alternative'})  
  
# insert code column in front of dataframe  
df2.insert(loc=0, column='Code', value=1)  
  
#loop through code column  
for i in range(len(df2)):  
 val = df2.loc[i, 'Code'] + i  
 df2.loc[i, 'Code'] = "A"+str(val)  
  
# remove index column  
df2 = df2.reset\_index(drop=True)  
  
df2

|  | Code | Alternative | q1 | q2 | q3 | q4 | q5 | q6 | q7 | q8 | q9 | q10 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | A1 | A11.2017.10120 - AF201703 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 1 | A2 | A11.2017.10350 - AF201703 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 2 | A3 | A11.2018.11309 - AF201703 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 3 | A4 | A11.2019.11618 - AF201703 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 4 | A5 | A11.2019.11622 - AF201703 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 20092 | A20093 | A11.2020.80018 - A11.54508 | 4 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 20093 | A20094 | A11.2020.80018 - A11.54812 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 20094 | A20095 | A11.2018.11461 - AF201704 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 20095 | A20096 | A11.2021.13607 - AF201704 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 4 |
| 20096 | A20097 | A11.2020.13081 - U201701 | 3 | 3 | 3 | 4 | 3 | 3 | 3 | 3 | 3 | 3 |

## Normalize Dataset

Because all of the criterias are benefit, so we only use one formula to normalize all values

if there are cost criteria, then we use formula below:

df3 = df2.copy()  
  
df3 = df3.drop(['Alternative'], axis=1)  
  
df3

|  | Code | q1 | q2 | q3 | q4 | q5 | q6 | q7 | q8 | q9 | q10 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | A1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 1 | A2 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 2 | A3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 3 | A4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 4 | A5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 20092 | A20093 | 4 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 20093 | A20094 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 20094 | A20095 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 20095 | A20096 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 4 |
| 20096 | A20097 | 3 | 3 | 3 | 4 | 3 | 3 | 3 | 3 | 3 | 3 |

Calculation example for A0:

R = Normalized Value  
i = Index alternative  
j = Index criteria  
x = Alternative value  
Max(x) = Maximum alternative value

# Select only the numeric columns in df3  
numeric\_cols = df3.select\_dtypes(include='number').columns  
  
# Divide all numeric columns in df3 by 4  
df3[numeric\_cols] = df3[numeric\_cols] / 4  
  
df3

|  | Code | q1 | q2 | q3 | q4 | q5 | q6 | q7 | q8 | q9 | q10 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | A1 | 0.50 | 0.50 | 0.50 | 0.5 | 0.50 | 0.50 | 0.50 | 0.50 | 0.50 | 0.50 |
| 1 | A2 | 1.00 | 1.00 | 1.00 | 1.0 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 2 | A3 | 1.00 | 1.00 | 1.00 | 1.0 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 3 | A4 | 1.00 | 1.00 | 1.00 | 1.0 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 4 | A5 | 1.00 | 1.00 | 1.00 | 1.0 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 20092 | A20093 | 1.00 | 0.75 | 0.75 | 1.0 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 20093 | A20094 | 1.00 | 1.00 | 1.00 | 1.0 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 20094 | A20095 | 1.00 | 1.00 | 1.00 | 1.0 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 20095 | A20096 | 1.00 | 1.00 | 1.00 | 1.0 | 1.00 | 1.00 | 0.75 | 0.75 | 0.75 | 1.00 |
| 20096 | A20097 | 0.75 | 0.75 | 0.75 | 1.0 | 0.75 | 0.75 | 0.75 | 0.75 | 0.75 | 0.75 |

## Calculate Final Alternative value by adding weight

V = Final value  
W = Weight value  
R = Normalized alternative value  
n = Number of data  
i = Index alternative  
j = Index criteria

weights = [1,3,2,3,4,4,1,2,5,5]  
  
df4 = df3.copy()  
  
for i in range (0,10):  
 df4[f'q{i+1}'] = df4[f'q{i+1}'] \* weights[i]  
  
df4['Total'] = df4.sum(axis=1)  
  
df4 = df4.drop(['q1','q2','q3','q4','q5','q6','q7','q8','q9','q10'], axis=1)  
  
df4

C:\Users\Administrator\AppData\Local\Temp\ipykernel\_24844\1810081016.py:8: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric\_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.  
 df4['Total'] = df4.sum(axis=1)

|  | Code | Total |
| --- | --- | --- |
| 0 | A1 | 15.00 |
| 1 | A2 | 30.00 |
| 2 | A3 | 30.00 |
| 3 | A4 | 30.00 |
| 4 | A5 | 30.00 |
| ... | ... | ... |
| 20092 | A20093 | 28.75 |
| 20093 | A20094 | 30.00 |
| 20094 | A20095 | 30.00 |
| 20095 | A20096 | 28.00 |
| 20096 | A20097 | 23.25 |