

Implementasi Metode TOPSIS
Untuk Sistem Prediksi Penyakit Diabetes



Dosen Pengampu :

Wahyu Sri Utami ,S.Si.,M.Sc.

Disusun Oleh

- | | |
|-----------------------------|------------|
| 1. Ayu Suci Khadijah | 5210411105 |
| 2. Nendy Nailul Autor | 5210411117 |
| 3. Dhea Desliana putri | 5210411138 |
| 4. Nicholas Bagus Pamungkas | 5210411183 |

PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS & TEKNOLOGI
UNIVERSITAS TEKNOLOGI YOGYAKARTA

2024/2025

STUDI KASUS

A. Diabetes

Diabetes adalah sekelompok penyakit metabolik yang ditandai dengan peningkatan kadar gula darah (hiperglikemia) yang disebabkan oleh kekurangan produksi insulin, resistensi insulin, atau keduanya. Insulin adalah hormon yang dihasilkan oleh pankreas dan berfungsi untuk mengatur kadar gula darah.

B. Topsis

Topsis (Techbique For Order Preference By Similarity To Ideal Solution) adalah metode pengambilan keputusan yang dapat di gunakan untuk menyelesaikan masalah dalam sistem pendukung keputusan yang di dasarkan pada konsepnya dimana alternatif terpilih bukan hanya yang memiliki jarak terpendek dari solusi ideal positif tetapi juga memiliki jarak terpanjang dari solusi ideal negatif .

C. Langkah Topsis

1. Normalisasi Data
2. Menentukan bobot untuk setiap kriteria
3. Menghitung nilai terbobot
4. Menghitung matriks ideal positif dan negatif.
5. Menghitung jarak Euclidean dari setiap alternatif terhadap kedua matriks ideal
6. Menghitung nilai kedekatan relatif (proximity) dari setiap alternatif terhadap matriks ideal positif dan negatif.
7. Menghitung nilai keseluruhan (total closeness) untuk setiap alternatif.
8. Mengurutkan alternatif berdasarkan nilai keseluruhan dari yang tertinggi ke terendah.

D. Data

index	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1
10	4	110	92	0	0	37.6	0.191	30	0

E. Perhitungan Manual

a. Normalisasi data

Untuk menormalisasi data membutuhkan formula:

Nilai normalisasi :

$$\frac{\text{Nilai Asli}}{\sqrt{\sum \text{Nilai Asli}^2}}$$

Berikut adalah hasil normalisasi untuk setiap kriteria :

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DPF	Age	Outcome
0.2778	0.4292	0.2821	0.1268	0.0653	0.2626	0.4704	0.4991	0.3904
0.0556	0.2417	0.2714	0.1134	0.0653	0.1996	0.3042	0.3491	0
0.4444	0.6875	0.2571	0	0.0653	0.1869	0.5036	0.3563	0.3904
0.0556	0.2583	0.2714	0.1019	0.1117	0.2189	0.1595	0.2328	0
0	0.5375	0.1714	0.1268	0.3661	0.2957	0.8417	0.3699	0.3904
0.2222	0.3625	0.2914	0	0.0653	0.2081	0.2581	0.3428	0
0.1667	0.1958	0.1429	0.1204	0.1077	0.0222	0.2074	0.2095	0.3904
0.5556	0.3583	0	0	0.0653	0.3207	0.1343	0.2556	0
0.1111	0.7917	0.2786	0.1586	0.5201	0.2898	0.1488	0.6428	0.3904
0.4444	0.5208	0.6	0	0.0653	0	0.2563	0.6734	0.3904

b. Menentukan bobot

Misalnya, kita memberikan bobot $w_i=0.1$ untuk setiap kriteria.

Menghitung Nilai Terbobot

Dengan bobot yang telah ditentukan, kita akan mengalikan nilai normalisasi dengan bobot.

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DPF	Age	Outcome
0.02778	0.04292	0.02821	0.01268	0.00653	0.02626	0.04704	0.04991	0.03904
0.00556	0.02417	0.02714	0.01134	0.00653	0.01996	0.03042	0.03491	0
0.04444	0.06875	0.02571	0	0.00653	0.01869	0.05036	0.03563	0.03904
0.00556	0.02583	0.02714	0.01019	0.01117	0.02189	0.01595	0.02328	0
0	0.05375	0.01714	0.01268	0.03661	0.02957	0.08417	0.03699	0.03904
0.02222	0.03625	0.02914	0	0.00653	0.02081	0.02581	0.03428	0
0.01667	0.01958	0.01429	0.01204	0.01077	0.00222	0.02074	0.02095	0.03904
0.05556	0.03583	0	0	0.00653	0.03207	0.01343	0.02556	0
0.01111	0.07917	0.02786	0.01586	0.05201	0.02898	0.01488	0.06428	0.03904
0.04444	0.05208	0.06	0	0.00653	0	0.02563	0.06734	0.03904

c. Menghitung Matriks Idel Positif dan Negatif

Untuk matriks ideal positif, kita mengambil nilai maksimum dari setiap kriteria, sedangkan untuk matriks ideal negatif, kita mengambil nilai minimum dari setiap kriteria.

	Positif	Negatif
Pregnancies	0.5556	0
Glucose	0.7917	0.0196
BloodPressure	0.6	0
SkinThickness	0.1586	0
Insulin	0.5201	0.0065
BMI	0.3207	0.0022
DiabetesPedigreeFunction	0.8417	0.0134
Age	0.6734	0.0209
Outcome	0.3904	0

d. Menghitung Jarak Euclidean

Berikut adalah jarak Euclidean untuk setiap alternatif:

Alternatif	(D^+)	(D^-)
A1	0.9361	0.5012
A2	0.9363	0.5024
A3	0.9346	0.5032
A4	0.9343	0.5039
A5	0.9367	0.5004
A6	0.9360	0.5025
A7	0.9364	0.5017
A8	0.9367	0.5031
A9	0.9356	0.5011
A10	0.9362	0.5021

e. Menghitung Nilai Kedekatan Relatif

Berikut adalah nilai kedekatan relatif terhadap matriks ideal positif ($C^+(A)C^+(A)$) dan negatif ($C^-(A)C^-(A)$) untuk setiap alternatif:

A	C+	C-
A1	0.3488	0.6512
A2	0.3491	0.6509
A3	0.3497	0.6503
A4	0.3502	0.6498
A5	0.3481	0.6519
A6	0.3491	0.6509
A7	0.3485	0.6515
A8	0.3497	0.6503
A9	0.3488	0.6512
A10	0.3494	0.6506

f. Menghitung Nilai Keseluruhan

Berikut adalah nilai keseluruhan ($C(A)C(A)$) untuk setiap alternatif:

A	$C+ + C- = \text{Total}$
A1	$0.3488 + 0.6512 = 1.0000$
A2	$0.3491 + 0.6509 = 1.0000$
A3	$0.3497 + 0.6503 = 1.0000$
A4	$0.3502 + 0.6498 = 1.0000$
A5	$0.3481 + 0.6519 = 1.0000$
A6	$0.3491 + 0.6509 = 1.0000$
A7	$0.3485 + 0.6515 = 1.0000$
A8	$0.3497 + 0.6503 = 1.0000$
A9	$0.3488 + 0.6512 = 1.0000$
A10	$0.3494 + 0.6506 = 1.0000$

g. Mengurutkan Alternatif

Dengan nilai keseluruhan ($C(A)C(A)$) yang sama untuk setiap alternatif, maka urutan tidak berubah. Jadi, semua alternatif memiliki nilai keseluruhan yang sama, yaitu 1.00001.0000, sehingga urutan relatif mereka tetap tidak berubah.

F. Source Code

Input:

```
import pandas as pd
import numpy as np

# Fungsi untuk normalisasi
def normalize(data):
    normalized_data = data.apply(lambda x: x / np.sqrt(np.sum(x**2)), axis=0)
```

```
return normalized_data
```

Fungsi untuk menghitung matriks terbobot

```
def weighted_matrix(normalized_data, weights):
```

```
    weighted_matrix = normalized_data * weights
```

```
    return weighted_matrix
```

Fungsi untuk menghitung matriks solusi ideal positif (PSI+) atau solusi ideal negatif (NSI-)

```
def ideal_solution(weighted_matrix, impacts):
```

```
    if impacts == '+':
```

```
        ideal_sol = weighted_matrix.max()
```

```
    else:
```

```
        ideal_sol = weighted_matrix.min()
```

```
    return ideal_sol
```

Fungsi untuk menghitung jarak relatif dari setiap alternatif terhadap solusi positif ideal dan negatif ideal

```
def relative_closeness(weighted_matrix, psi_plus, nsi_minus):
```

```
    d_plus = np.sqrt(np.sum((weighted_matrix - psi_plus)**2, axis=1))
```

```
    d_minus = np.sqrt(np.sum((weighted_matrix - nsi_minus)**2, axis=1))
```

```
    relative_closeness = d_minus / (d_minus + d_plus)
```

```
    return relative_closeness
```

Fungsi untuk melakukan perankingan

```
def topsis_ranking(data, weights, impacts):
```

```
    normalized_data = normalize(data)
```

```
    weighted_matrix_data = weighted_matrix(normalized_data, weights)
```

```

psi_plus = ideal_solution(weighted_matrix_data, '+')

nsi_minus = ideal_solution(weighted_matrix_data, '-')

rc_values = relative_closeness(weighted_matrix_data, psi_plus, nsi_minus)

ranked_indices = np.argsort(rc_values)[::-1] + 1

return ranked_indices, psi_plus, nsi_minus

# Baca file CSV

file_path = r'/content/diabetes.csv'

data = pd.read_csv(file_path)

# Pilih kolom yang akan digunakan

selected_columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI',
'DiabetesPedigreeFunction', 'Age']

data = data[selected_columns]

# Bobot untuk setiap kriteria

weights = np.array([1, 1, 1, 1, 1, 1, 1, 1])

# Dampak (+ atau -) untuk setiap kriteria

impacts = '+'

# Lakukan perankingan dan ambil solusi ideal

ranked_indices, psi_plus, nsi_minus = topsis_ranking(data, weights, impacts)

# Pengambilan keputusan

if not ranked_indices.empty:

```



```

print("Perankingan Alternatif:")

for i, idx in enumerate(ranked_indices):

    print(f"Rank {i+1}: Alternatif ke-{idx}")


best_alternative = ranked_indices.iloc[0]

decision = f"Alternatif terbaik adalah yang pertama (Alternatif {best_alternative})"

else:

    decision = 'Tidak ada alternatif yang dinilai'

```

Output:

```

↔ Alternatif Perankingan:
Peringkat 1 : Alternatif ke-229
Peringkat 2: Alternatif ke-371
Peringkat 3 : Alternatif ke-14
Peringkat 4: Alternatif ke-585
Peringkat 5: Alternatif ke-248
Peringkat 6: Alternatif ke-410
Peringkat 7: Alternatif ke-287
Peringkat 8: Alternatif ke-187
Peringkat 9: Alternatif ke-112
Peringkat 10 : Alternatif ke-9
Peringkat 11: Alternatif ke-376
Peringkat 12: Alternatif ke-154
Peringkat 13: Alternatif ke-446
Peringkat 14: Alternatif ke-221
Peringkat 15: Alternatif ke-754
Peringkat 16: Alternatif ke-459
Peringkat 17: Alternatif ke-716
Peringkat 18: Alternatif ke-216
Peringkat 19: Alternatif ke-487

```

M Kode M Teks

```
▶ Peringkat 745: Alternatif ke-270
↳ Peringkat 746: Alternatif ke-586
Peringkat 747: Alternatif ke-439
Peringkat 748: Alternatif ke-523
Peringkat 749: Alternatif ke-618
Peringkat 750: Alternatif ke-730
Peringkat 751: Alternatif ke-408
Peringkat 752: Alternatif ke-467
Peringkat 753: Alternatif ke-462
Peringkat 754: Alternatif ke-197
Peringkat 755: Alternatif ke-695
Peringkat 756: Alternatif ke-620
Peringkat 757: Alternatif ke-348
Peringkat 758: Alternatif ke-269
Peringkat 759: Alternatif ke-76
Peringkat 760: Alternatif ke-56
Peringkat 761: Alternatif ke-91
Peringkat 762: Alternatif ke-698
Peringkat 763: Alternatif ke-431
Peringkat 764: Alternatif ke-590
Peringkat 765: Alternatif ke-495
Peringkat 766: Alternatif ke-61
Peringkat 767: Alternatif ke-427
Peringkat 768: Alternatif ke-82
```

```
▶ print("\nKeputusan:", decision)
```

```
↳ Keputusan: Alternatif terbaik adalah yang pertama (Alternatif 229)
```

Input

```
import matplotlib.pyplot as plt

# Plotting diagram scatter

plt.figure(figsize=(8, 6))

plt.scatter(range(len(ranked_indices)), ranked_indices, color='skyblue', marker='o')

plt.xlabel('Alternatif')

plt.ylabel('Peringkat')

plt.title('Perankingan Alternatif dengan Metode Fuzzy TOPSIS')

plt.xticks(range(len(ranked_indices)), ranked_indices)

plt.gca().invert_yaxis() # Dibalikkan agar peringkat tertinggi ada di atas

plt.grid(axis='y', linestyle='--', alpha=0.7)
```

```
plt.show()
```

Output:



