

MULTI-ATTRIBUTE UTILITY THEORY ANALYSIS

- **Fransiskus Jremiegi S (205314062)**
- **Ignatius Marshel Peter Chan
(205314114)**
- **Angel Katarina Sinurat (205314134)**

MULTI-ATTRIBUTE UTILITY THEORY (MAUT)

Suatu pendekatan dalam pengambilan keputusan yang kompleks, terutama ketika terlibat banyak atribut atau kriteria yang perlu dipertimbangkan. MAUT memungkinkan pengambil keputusan untuk menggabungkan informasi dari berbagai aspek yang relevan, termasuk preferensi, nilai, dan bobot relatif dari setiap atribut, untuk membuat keputusan yang lebih informatif dan terstruktur.

MENGAPA METODE INI PENTING

- | | | | |
|-----------|------------------------------------|-----------|--|
| 01 | Pertimbangan Multikriteria | 05 | Fleksibilitas dalam Pemilihan Alternatif |
| 02 | Struktur Keputusan yang Jelas | 06 | Penghematan Waktu dan Sumber Daya |
| 03 | Menggabungkan Preferensi Subjektif | 07 | Manajemen Risiko yang Lebih Baik |
| 04 | Peningkatan Transparansi | | |

Gagasan Pokok MAUT

★ Atribut dan Kriteria



MAUT berfokus pada pemilihan di antara beberapa alternatif berdasarkan beberapa atribut atau kriteria. Atribut ini dapat mencakup berbagai faktor seperti biaya, kualitas, kecepatan, keamanan, dan lainnya yang relevan dengan konteks pengambilan keputusan.

★ Preferensi Subjektif



MAUT mengakui bahwa preferensi terhadap atribut tersebut bersifat subjektif dan bervariasi antara individu atau kelompok yang terlibat dalam pengambilan keputusan.

★ **Bobot Atribut & Fungsi Utilitas**



Setiap atribut diberi bobot yang mencerminkan tingkat kepentingan relatif. Fungsi utilitas digunakan untuk mengukur tingkat kepuasan atau nilai yang diberikan oleh setiap alternatif pada setiap atribut.

★ **Penilaian & Perbanding Alternatif**



Alternatif dievaluasi dengan mengalikan nilai utilitas setiap atribut dengan bobotnya masing-masing dan menjumlahkan hasilnya dan maut memungkinkan perbandingan antara alternatif berdasarkan nilai total yang dihasilkan.

★ **Transparansi dan Pengambilan Keputusan yang Diketahui dengan Jelas**



Maut memungkinkan proses pengambilan keputusan yang dapat dijelaskan dengan baik dan transparan.

Langkah-langkah

- Normalisasi data
- Menghitung nilai normalisasi
- Perkalian antara nilai normalisasi dengan bobot referensi
- Menentukan tingkat kelayakan
- Menentukan perangsangan

Kasus

Menentukan kelayakan lahan pembibitan kelapa sawit dengan menggunakan metode Multi Attribute Utility Theory (MAUT) agar lebih efektif dan efisien bagi pihak pimpinan dalam mengambil suatu keputusan dengan demikian perusahaan tidak mengalami kerugian.

Referensi:

J. Hutagalung, A. H. Nasyuha and T. Pradita, "Sistem Pendukung Keputusan Menentukan Kelayakan Lahan Pembibitan Menggunakan Metode Multi Attribute Utility Theory," Journal of Computer System and Informatics (JoSYC), vol. 4, p. 79–87, 2022.

Data

No	Blok	Luas Area (Ha)	Curah Hujan	Jenis Tanah	Ketinggian (MDPL)	Kemiringan (°)
1	Blok 1	36,48	2380	Alluvial	150	2
2	Blok 13	13,92	1780	Regosol	200	4
3	Blok 14	53,7	1998	Regosol	200	2
4	Blok 15	17,99	1822	Regosol	120	5
5	Blok 16	50,21	2459	Regosol	100	8
6	Blok 17	69,09	1945	Regosol	130	2
7	Blok 30	20,94	1540	Alluvial	100	3
8	Blok 31	28,02	2180	Alluvial	200	2
9	Blok 3A	13,97	1600	Regosol	80	4
10	Blok 4A	30,45	1614	Regosol	110	6
11	Blok 7A	22,22	2170	Alluvial	90	4

IMPLEMENTASI PYTHON

Baca File

```
url = "Data/DataKelayakanLahanPembibitan.xlsx"  
df = pd.read_excel(url)  
df.head()
```

	No	Block	Luas Area	Curah Hujan	Jenis Tanah	Ketinggian	Kemiringan
0	1	Blok 1	36.48	2380	Alluvial	150	2
1	2	Blok 13	13.92	1780	Regosol	200	4
2	3	Blok 14	53.70	1998	Regosol	200	2
3	4	Blok 15	17.99	1822	Regosol	120	5
4	5	Blok 16	50.21	2459	Regosol	100	8

Preprocessing

```
df = df.drop(['No'], axis=1)
df.head()
```

	Block	Luas Area	Curah Hujan	Jenis Tanah	Ketinggian	Kemiringan
0	Blok 1	36.48	2380	Alluvial	150	2
1	Blok 13	13.92	1780	Regosol	200	4
2	Blok 14	53.70	1998	Regosol	200	2
3	Blok 15	17.99	1822	Regosol	120	5
4	Blok 16	50.21	2459	Regosol	100	8

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 6 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Block           11 non-null    object
1   Luas Area       11 non-null    float64
2   Curah Hujan     11 non-null    int64
3   Jenis Tanah     11 non-null    object
4   Ketinggian     11 non-null    int64
5   Kemiringan      11 non-null    int64
dtypes: float64(1), int64(3), object(2)
memory usage: 656.0+ bytes
```

Transformasi Nilai Pada Tiap Atribut

```
def transformLuasArea(valuasi):  
    if valuasi <= 15:  
        return 1  
    elif valuasi > 15 and valuasi <= 25:  
        return 2  
    elif valuasi > 25 and valuasi <= 35:  
        return 3  
    elif valuasi > 35 and valuasi <= 50:  
        return 4  
    elif valuasi > 50:  
        return 5  
    else:  
        return 0
```

```
df['Luas Area'] = df['Luas Area'].apply(transformLuasArea) # Melakukan transformasi data pada luas area  
df
```

	Block	Luas Area	Curah Hujan	Jenis Tanah	Ketinggian	Kemiringan
0	Blok 1	4	2380	Alluvial	150	2
1	Blok 13	1	1780	Regosol	200	4
2	Blok 14	5	1998	Regosol	200	2
3	Blok 15	2	1822	Regosol	120	5
4	Blok 16	5	2459	Regosol	100	8
5	Blok 17	5	1945	Regosol	130	2
6	Blok 30	2	1540	Alluvial	100	3
7	Blok 31	3	2180	Alluvial	200	2
8	Blok 3A	1	1600	Regosol	80	4
9	Blok 4A	3	1614	Regosol	110	6
10	Blok 7A	2	2170	Alluvial	90	4

```
def transformCurahHujan(valuasi):
    if valuasi <= 1700:
        return 1
    elif valuasi > 1700 and valuasi <= 1800:
        return 2
    elif valuasi > 1800 and valuasi <= 1900:
        return 3
    elif valuasi > 1900 and valuasi <= 2000:
        return 4
    elif valuasi > 2000:
        return 5
    else:
        return 0
```

```
df['Curah Hujan'] = df['Curah Hujan'].apply(transformCurahHujan) # Melakukan transformasi data pada curah hujan
df
```

	Block	Luas Area	Curah Hujan	Jenis Tanah	Ketinggian	Kemiringan
0	Blok 1	4	5	Alluvial	150	2
1	Blok 13	1	2	Regosol	200	4
2	Blok 14	5	4	Regosol	200	2
3	Blok 15	2	3	Regosol	120	5
4	Blok 16	5	5	Regosol	100	8
5	Blok 17	5	4	Regosol	130	2
6	Blok 30	2	1	Alluvial	100	3
7	Blok 31	3	5	Alluvial	200	2
8	Blok 3A	1	1	Regosol	80	4
9	Blok 4A	3	1	Regosol	110	6
10	Blok 7A	2	5	Alluvial	90	4

```
def transformJenisTanah(valuasi):
    if valuasi == 'Regosol':
        return 1
    elif valuasi == 'Alluvial':
        return 2
    else:
        return 0
```

```
df['Jenis Tanah'] = df['Jenis Tanah'].apply(transformJenisTanah) # Melakukan transformasi data pada jenis tanah
df
```

	Block	Luas Area	Curah Hujan	Jenis Tanah	Ketinggian	Kemiringan
0	Blok 1	4	5	2	150	2
1	Blok 13	1	2	1	200	4
2	Blok 14	5	4	1	200	2
3	Blok 15	2	3	1	120	5
4	Blok 16	5	5	1	100	8
5	Blok 17	5	4	1	130	2
6	Blok 30	2	1	2	100	3
7	Blok 31	3	5	2	200	2
8	Blok 3A	1	1	1	80	4
9	Blok 4A	3	1	1	110	6
10	Blok 7A	2	5	2	90	4

```
def transformKetinggian(valuasi):
    if valuasi <= 89:
        return 4
    elif valuasi > 89 and valuasi <= 129:
        return 3
    elif valuasi > 129 and valuasi <= 169:
        return 2
    elif valuasi > 169:
        return 1
    else:
        return 0
```

```
df['Ketinggian'] = df['Ketinggian'].apply(transformKetinggian) # Melakukan transformasi data pada ketinggian
df
```

	Block	Luas Area	Curah Hujan	Jenis Tanah	Ketinggian	Kemiringan
0	Blok 1	4	5	2	2	2
1	Blok 13	1	2	1	1	4
2	Blok 14	5	4	1	1	2
3	Blok 15	2	3	1	3	5
4	Blok 16	5	5	1	3	8
5	Blok 17	5	4	1	2	2
6	Blok 30	2	1	2	3	3
7	Blok 31	3	5	2	1	2
8	Blok 3A	1	1	1	4	4
9	Blok 4A	3	1	1	3	6
10	Blok 7A	2	5	2	3	4

```
def transformKemiringan(valuasi):
    if valuasi <= 2:
        return 3
    elif valuasi > 2 and valuasi <= 5:
        return 2
    elif valuasi > 5 and valuasi <= 8:
        return 1
    else:
        return 0
```

```
df['Kemiringan'] = df['Kemiringan'].apply(transformKemiringan) # Melakukan transformasi data pada kemiringan
df
```

	Block	Luas Area	Curah Hujan	Jenis Tanah	Ketinggian	Kemiringan
0	Blok 1	4	5	2	2	3
1	Blok 13	1	2	1	1	2
2	Blok 14	5	4	1	1	3
3	Blok 15	2	3	1	3	2
4	Blok 16	5	5	1	3	1
5	Blok 17	5	4	1	2	3
6	Blok 30	2	1	2	3	2
7	Blok 31	3	5	2	1	3
8	Blok 3A	1	1	1	4	2
9	Blok 4A	3	1	1	3	1
10	Blok 7A	2	5	2	3	2

Mencari Nilai Min & Max Pada Tiap Atribut

```
5... MinimalLuasArea = df['Luas Area'].min()
MaksimalLuasArea = df['Luas Area'].max()

print('Nilai Minimal Luas Area = ', MinimalLuasArea)
print('Nilai Maksimal Luas Area = ', MaksimalLuasArea)

Nilai Minimal Luas Area = 1
Nilai Maksimal Luas Area = 5

5... MinimalCurahHujan = df['Curah Hujan'].min()
MaksimalCurahHujan = df['Curah Hujan'].max()

print('Nilai Minimal Curah Hujan = ', MinimalCurahHujan)
print('Nilai Maksimal Curah Hujan = ', MaksimalCurahHujan)

Nilai Minimal Curah Hujan = 1
Nilai Maksimal Curah Hujan = 5

5... MinimalJenisTanah = df['Jenis Tanah'].min()
MaksimalJenisTanah = df['Jenis Tanah'].max()

print('Nilai Minimal Jenis Tanah = ', MinimalJenisTanah)
print('Nilai Maksimal Jenis Tanah = ', MaksimalJenisTanah)

Nilai Minimal Jenis Tanah = 1
Nilai Maksimal Jenis Tanah = 2

5... MinimalKetinggian = df['Ketinggian'].min()
MaksimalKetinggian = df['Ketinggian'].max()

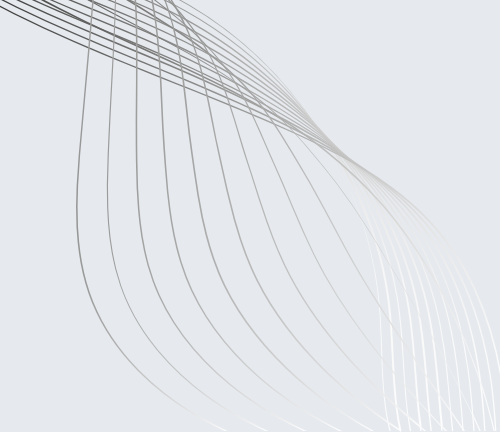
print('Nilai Minimal Ketinggian = ', MinimalKetinggian)
print('Nilai Maksimal Ketinggian = ', MaksimalKetinggian)

Nilai Minimal Ketinggian = 1
Nilai Maksimal Ketinggian = 4


5... MinimalKemiringan = df['Kemiringan'].min()
MaksimalKemiringan = df['Kemiringan'].max()

print('Nilai Minimal Kemiringan = ', MinimalKemiringan)
print('Nilai Maksimal Kemiringan = ', MaksimalKemiringan)

Nilai Minimal Kemiringan = 1
Nilai Maksimal Kemiringan = 3
```

Normalisasi Nilai Pada Tiap Atribut



```
df_LuasArea = pd.DataFrame(columns=['Perhitungan Luas Area'])

for i in df['Luas Area'] :
    Hitung = (i - MinimalLuasArea) / (MaksimalLuasArea - MinimalLuasArea)
    # print(Hitung)
    df_LuasArea = pd.concat([df_LuasArea, pd.DataFrame({'Perhitungan Luas Area': [Hitung]})], ignore_index=True)
```

df_LuasArea

Perhitungan Luas Area	
0	0.75
1	0.00
2	1.00
3	0.25
4	1.00
5	1.00
6	0.25
7	0.50
8	0.00
9	0.50
10	0.25

```
df_CurahHujan = pd.DataFrame(columns=['Perhitungan Curah Hujan'])
```

```
for i in df['Curah Hujan'] :  
    Hitung = (i - MinimalCurahHujan) / (MaksimalCurahHujan - MinimalCurahHujan)  
    # print(Hitung)  
    # df_hasil = df_hasil.assign(PerhitunganCurahHujan=Hitung)  
    df_CurahHujan = pd.concat([df_CurahHujan, pd.DataFrame({'Perhitungan Curah Hujan': [Hitung]})], ignore_index=True)  
    # df_hasil['Perhitungan Curah Hujan'] = Hitung
```

```
df_CurahHujan
```

Perhitungan Curah Hujan

0	1.00
1	0.25
2	0.75
3	0.50
4	1.00
5	0.75
6	0.00
7	1.00
8	0.00
9	0.00
10	1.00

```
df_JenisTanah = pd.DataFrame(columns=['Perhitungan Jenis Tanah'])
```

```
for i in df['Jenis Tanah'] :  
    Hitung = (i - MinimalJenisTanah) / (MaksimalJenisTanah - MinimalJenisTanah)  
    # print(Hitung)  
    df_JenisTanah = pd.concat([df_JenisTanah, pd.DataFrame({'Perhitungan Jenis Tanah': [Hitung]})], ignore_index=True)
```

```
df_JenisTanah
```

Perhitungan Jenis Tanah	
0	1.0
1	0.0
2	0.0
3	0.0
4	0.0
5	0.0
6	1.0
7	1.0
8	0.0
9	0.0
10	1.0

```
df_Ketinggian = pd.DataFrame(columns=['Perhitungan Ketinggian'])
```

```
for i in df['Ketinggian'] :  
    Hitung = (i - MinimalKetinggian) / (MaksimalKetinggian - MinimalKetinggian)  
    # print(Hitung)  
    df_Ketinggian = pd.concat([df_Ketinggian, pd.DataFrame({'Perhitungan Ketinggian': [Hitung]})], ignore_index=True)
```

```
df_Ketinggian
```

Perhitungan Ketinggian	
------------------------	--

0	0.333333
1	0.000000
2	0.000000
3	0.666667
4	0.666667
5	0.333333
6	0.666667
7	0.000000
8	1.000000
9	0.666667
10	0.666667



```
df_Kemiringan = pd.DataFrame(columns=['Perhitungan Kemiringan'])
```

```
for i in df['Kemiringan'] :  
    Hitung = (i - MinimalKemiringan) / (MaksimalKemiringan - MinimalKemiringan)  
    # print(Hitung)  
    df_Kemiringan = pd.concat([df_Kemiringan, pd.DataFrame({'Perhitungan Kemiringan': [Hitung]})], ignore_index=True)
```

df_Kemiringan

Perhitungan Kemiringan	
0	1.0
1	0.5
2	1.0
3	0.5
4	0.0
5	1.0
6	0.5
7	1.0
8	0.5
9	0.0
10	0.5

Mengabungkan Kolom Pada Dataframe Untuk Setipa Perhitungan Normalisasi Pada Atribut

```
df_hasil = pd.concat([df['Block'], df_LuasArea, df_CurahHujan, df_JenisTanah, df_Ketinggian, df_Kemiringan], axis=1)
df_hasil
```

	Block	Perhitungan Luas Area	Perhitungan Curah Hujan	Perhitungan Jenis Tanah	Perhitungan Ketinggian	Perhitungan Kemiringan
0	Blok 1	0.75	1.00	1.0	0.333333	1.0
1	Blok 13	0.00	0.25	0.0	0.000000	0.5
2	Blok 14	1.00	0.75	0.0	0.000000	1.0
3	Blok 15	0.25	0.50	0.0	0.666667	0.5
4	Blok 16	1.00	1.00	0.0	0.666667	0.0
5	Blok 17	1.00	0.75	0.0	0.333333	1.0
6	Blok 30	0.25	0.00	1.0	0.666667	0.5
7	Blok 31	0.50	1.00	1.0	0.000000	1.0
8	Blok 3A	0.00	0.00	0.0	1.000000	0.5
9	Blok 4A	0.50	0.00	0.0	0.666667	0.0
10	Blok 7A	0.25	1.00	1.0	0.666667	0.5

Menghitung Perkalian Hasil Normalisasi Dengan Bobot Referensi

```
# Bobot pada tiap kriteria
LuasArea = 0.25
CurahHujan = 0.2
JenisTanah = 0.3
Ketinggian = 0.15
Kemiringan = 0.1
# Hasil = LuasArea + CurahHujan + JenisTanah + Ketinggian + Kemiringan
# Hasil
```

```
# df_NilaiAkhir = pd.DataFrame(columns=['Nilai Akhir', 'Keputusan Kelayakan'])
df_Keputusan = pd.DataFrame(columns=['Block', 'Nilai Akhir', 'Kelayakan'])
```

```
# Mengganti df_hasil dengan df yang sesuai dengan DataFrame Anda
for index, row in df_hasil.iterrows():
    Hasil = (row['Perhitungan Luas Area'] * LuasArea) + \
            (row['Perhitungan Curah Hujan'] * CurahHujan) + \
            (row['Perhitungan Jenis Tanah'] * JenisTanah) + \
            (row['Perhitungan Ketinggian'] * Ketinggian) + \
            (row['Perhitungan Kemiringan'] * Kemiringan)

    # Lakukan sesuatu dengan nilai Hasil, misalnya mencetaknya
    # print(f'Hasil untuk baris {index + 1}: {Hasil}')

    Kelayakan = 'Layak' if Hasil > 0.5 else 'Tidak Layak'

# df_NilaiAkhir = pd.concat([df_NilaiAkhir, pd.DataFrame({'Nilai Akhir': [Hasil], 'Keputusan Kelayakan': [Keputusan]})],
df_Keputusan = pd.concat([df_Keputusan, pd.DataFrame({'Block': [row['Block']], 'Nilai Akhir': [Hasil], 'Kelayakan': [Keputusan]})],
```

df_Keputusan

	Block	Nilai Akhir	Kelayakan
0	Blok 1	0.8375	Layak
1	Blok 13	0.1000	Tidak Layak
2	Blok 14	0.5000	Tidak Layak
3	Blok 15	0.3125	Tidak Layak
4	Blok 16	0.5500	Layak
5	Blok 17	0.5500	Layak
6	Blok 30	0.5125	Layak
7	Blok 31	0.7250	Layak
8	Blok 3A	0.2000	Tidak Layak
9	Blok 4A	0.2250	Tidak Layak
10	Blok 7A	0.7125	Layak



Perengkingan

```
df_Keputusan = df_Keputusan.sort_values(by='Nilai Akhir', ascending=False)
```

```
df_Keputusan['Rangking'] = range(1, len(df_Keputusan) + 1)
```

df_Keputusan

	Block	Nilai Akhir	Kelayakan	Rangking
0	Blok 1	0.8375	Layak	1
7	Blok 31	0.7250	Layak	2
10	Blok 7A	0.7125	Layak	3
4	Blok 16	0.5500	Layak	4
5	Blok 17	0.5500	Layak	5
6	Blok 30	0.5125	Layak	6
2	Blok 14	0.5000	Tidak Layak	7
3	Blok 15	0.3125	Tidak Layak	8
9	Blok 4A	0.2250	Tidak Layak	9
8	Blok 3A	0.2000	Tidak Layak	10
1	Blok 13	0.1000	Tidak Layak	11



Terima Kasih