

PRAKTIKUM
SISTEM CERDAS DAN PENDUKUNG KEPUTUSAN
SEMESTER GENAP T.A 2024/2025

LAPORAN PROYEK AKHIR



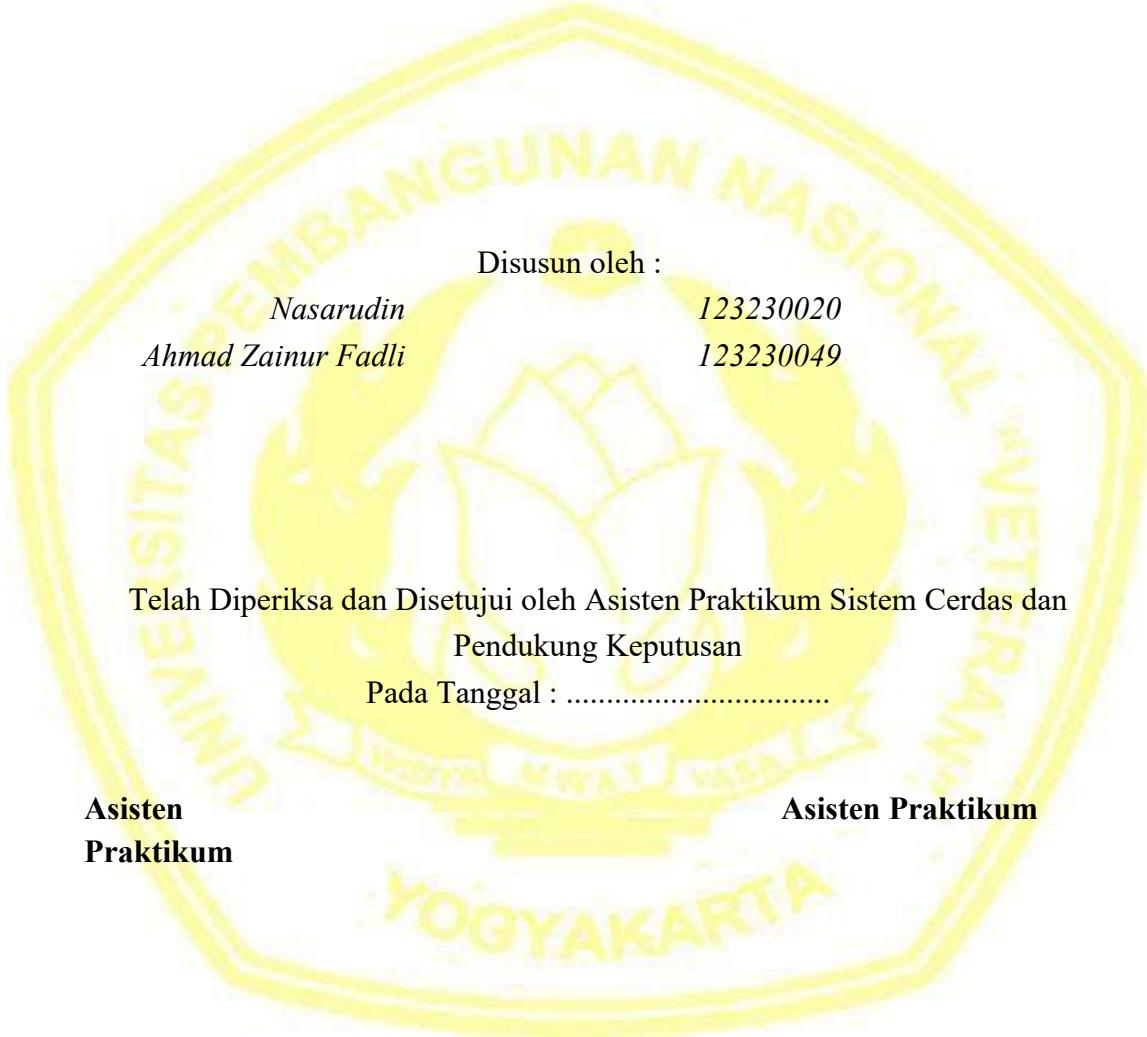
DISUSUN OLEH :

NIM	: - 123230020
	- 123230049
NAMA	: - Nasarudin
	- Ahmad Zainur Fadi
PLUG	: IF-G
NAMA ASISTEN	: - Arvidion Havas Oktavian
	- Panca Aulia Rahman

PROGRAM STUDI INFORMATIKA
JURUSAN INFORMATIKA
FAKULTAS TEKNIK INDUSTRI
UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"
YOGYAKARTA
2025

HALAMAN PENGESAHAN

LAPORAN PROYEK AKHIR



Disusun oleh :

Nasarudin

123230020

Ahmad Zainur Fadli

123230049

Telah Diperiksa dan Disetujui oleh Asisten Praktikum Sistem Cerdas dan
Pendukung Keputusan
Pada Tanggal :

Asisten
Praktikum

Asisten Praktikum

Arvidion Havas Oktavian
NIM. 123220067

Panca Aulia Rahman
NIM. 1

KATA PENGANTAR

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa yang senantiasa mencerahkan rahmat dan hidayah-Nya sehingga saya dapat menyelesaikan praktikum Sistem Cerdas dan Pendukung Keputusan serta laporan proyek akhir praktikum yang berjudul SPK Pemilihan Tanaman. Adapun laporan ini berisi tentang proyek akhir yang saya pilih dari hasil pembelajaran selama praktikum berlangsung.

Tidak lupa ucapan terimakasih kepada asisten dosen yang selalu membimbing dan mengajari saya dalam melaksanakan praktikum dan dalam menyusun laporan ini. Laporan ini masih sangat jauh dari kesempurnaan, oleh karena itu kritik serta saran yang membangun saya harapkan untuk menyempurnakan laporan akhir ini.

Atas perhatian dari semua pihak yang membantu penulisan ini, saya ucapkan terimakasih. Semoga laporan ini dapat dipergunakan seperlunya.

Yogyakarta, 2 Juni 2025

Penyusun

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN	II
KATA PENGANTAR.....	III
DAFTAR ISI	IV
SPK PEMILIHAN TANAMAN	1
BAB I PENDAHULUAN	2
1.1 Latar Belakang Masalah	2
1.2 Tujuan Proyek Akhir	2
1.3 Manfaat Proyek Akhir	2
BAB II PEMBAHASAN.....	3
2.1 Dasar Teori	3
2.2 Deskripsi Umum Proyek Akhir	3
2.3 Inti Pembahasan.....	4
BAB III JADWAL PENGERJAAN DAN PEMBAGIAN TUGAS	18
3.1 Jadwal Pengerjaan	18
3.2 Pembagian Tugas.....	18
BAB IV KESIMPULAN DAN SARAN.....	19
4.1 Kesimpulan.....	19
4.2 Saran	19
DAFTAR PUSTAKA.....	20

SPK PEMILIHAN TANAMAN

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Pemilihan jenis tanaman yang sesuai dengan kondisi lingkungan seperti Nitrogen, Fosfor, Kalium, Suhu, Kelembapan, pH Tanah, Curah Hujan merupakan faktor penting dalam keberhasilan budidaya tanaman. Sayangnya, proses penentuan ini seringkali tidak mudah dilakukan secara manual karena banyaknya variabel yang perlu dipertimbangkan secara bersamaan.

Oleh karena itu, diperlukan suatu alat bantu berupa Sistem Pendukung Keputusan (SPK) yang dapat membantu pengguna memilih tanaman yang paling sesuai berdasarkan beberapa kriteria yang telah ditentukan. Salah satu metode yang cocok untuk digunakan dalam SPK adalah *Simple Additive Weighting* (SAW), yaitu metode pengambilan keputusan multikriteria yang bekerja dengan menjumlahkan nilai kriteria yang telah dinormalisasi dan diberi bobot.

Melalui metode ini, sistem dapat memberikan hasil rekomendasi yang objektif dan sistematis, sehingga mempermudah proses pengambilan keputusan dalam memilih jenis tanaman yang optimal.

1.2 Tujuan Proyek Akhir

Tujuan dari proyek ini adalah:

- Membuat sistem pendukung keputusan pemilihan tanaman menggunakan metode SAW.
- Mengimplementasikan metode SAW dalam bentuk program Python.
- Membantu pengguna memilih tanaman berdasarkan kriteria tertentu secara efektif.

1.3 Manfaat Proyek Akhir

Manfaat dari proyek ini antara lain:

- Mempermudah pengguna dalam menentukan tanaman yang sesuai dengan kondisi lahan tertentu.
- Meningkatkan akurasi dan efisiensi dalam proses pemilihan tanaman melalui pendekatan berbasis sistem pendukung keputusan.
- Menjadi sarana pembelajaran dalam menerapkan konsep metode SAW dalam pengembangan sistem berbasis Python

BAB II

PEMBAHASAN

2.1 Dasar Teori

Metode *Simple Additive Weighting* (SAW) merupakan salah satu teknik pengambilan keputusan yang cukup populer dalam Sistem Pendukung Keputusan (SPK), terutama untuk masalah dengan banyak kriteria (MADM – *Multiple Attribute Decision Making*). Prinsip dasar dari metode ini adalah menjumlahkan nilai alternatif yang telah dikalikan dengan bobot masing-masing kriteria.

Sebelum melakukan penjumlahan, data nilai alternatif perlu dinormalisasi terlebih dahulu agar setiap kriteria berada pada skala yang sebanding. Jenis kriteria dibagi menjadi dua:

- Benefit (semakin besar nilainya, semakin baik),
- Cost (semakin kecil nilainya, semakin baik).

Langkah-langkah umum dalam metode SAW antara lain:

1. Menentukan daftar kriteria yang akan digunakan.
2. Memberikan nilai kecocokan antara alternatif dan tiap kriteria.
3. Melakukan proses normalisasi terhadap matriks keputusan berdasarkan tipe kriteria.
4. Mengalikan nilai normalisasi dengan bobot kriteria.
5. Menjumlahkan hasil perkalian untuk mendapatkan nilai akhir setiap alternatif.

Alternatif yang memperoleh nilai tertinggi akan menjadi rekomendasi utama dalam pengambilan keputusan.

2.2 Deskripsi Umum Proyek Akhir

Proyek akhir ini dibuat untuk menghasilkan sistem berbasis Python yang mampu memberikan rekomendasi tanaman terbaik berdasarkan kondisi lingkungan yang dimasukkan oleh pengguna. Sistem ini memanfaatkan metode SAW untuk menghitung dan menentukan tanaman yang paling sesuai dari beberapa alternatif.

Beberapa kriteria yang menjadi dasar pengambilan keputusan dalam sistem ini adalah:

- Nitrogen
- Fosfor
- Kalium
- Suhu
- Kelembapan
- pH Tanah
- Curah Hujan

Tiap kriteria memiliki bobot tertentu yang mencerminkan tingkat kepentingannya. Sebagian kriteria, seperti Nitrogen, Fosfor, Kalium, merupakan atribut *benefit*, sementara lainnya merupakan atribut *cost*.

2.3 Inti Pembahasan

Proyek ini adalah sebuah Sistem Pendukung Keputusan (SPK) yang bertujuan untuk membantu pengguna dalam memilih jenis tanaman yang paling sesuai untuk dibudidayakan berdasarkan kondisi lahan tertentu. SPK ini diimplementasikan sebagai aplikasi web menggunakan framework Streamlit di Python. Metode yang digunakan untuk pengambilan keputusan adalah *Simple Additive Weighting* (SAW).

Pengguna dapat memasukkan data kondisi lahannya, seperti kandungan Nitrogen (N), Fosfor (P), Kalium (K), suhu, kelembapan, pH tanah, dan curah hujan. Berdasarkan input tersebut dan bobot kriteria yang telah ditentukan secara statis, aplikasi akan menghitung skor preferensi untuk setiap jenis tanaman yang ada dalam dataset dan memberikan rekomendasi tanaman dengan skor tertinggi.

A. Listing Program

```
import streamlit as st
import pandas as pd
import numpy as np

st.set_page_config(page_title="SPK Pemilihan Tanaman",
layout="wide")

# Styling
st.markdown("""
<style>
    .main-header {
        text-align: center;
        padding: 2rem 0;
        background: linear-gradient(90deg, #4CAF50, #8BC34A);
        border-radius: 10px;
        margin-bottom: 2rem;
        color: white;
    }

    .recommendation-card {
        background: linear-gradient(135deg, #11998e 0%, #38ef7d
100%);
        padding: 2rem;
        border-radius: 20px;
        color: white;
        text-align: center;
        box-shadow: 0 10px 30px rgba(17, 153, 142, 0.3);
        margin: 2rem 0;
    }

    .input-section {
        background: #f8f9fa;
        padding-top: 1rem;
        border-radius: 15px;
    }
</style>
""")
```

```

        margin-bottom: 2rem;
        border: 1px solid #e9ecf;
    }

    .input-section h3 {
        color: #2e7d32;
        margin-bottom: 1rem;
        text-align: center;
    }

    .stButton > button {
        background: linear-gradient(90deg, #4CAF50, #8BC34A);
        color: white;
        border: none;
        padding: 0.75rem 2rem;
        border-radius: 25px;
        font-weight: bold;
        font-size: 1.1rem;
        width: 100%;
        transition: all 0.3s ease;
    }
</style>
"""", unsafe_allow_html=True)

# Header
st.markdown("""
<div class="main-header">
    <h1>🌿 Sistem Pendukung Keputusan Pemilihan Tanaman</h1>
    <p>Temukan tanaman yang paling cocok untuk lahan Anda dengan
metode SAW (Simple Additive Weighting)</p>
</div>
""", unsafe_allow_html=True)

# Fungsi SAW
def saw(alternatif_matrix, nilai_input, bobot, atribut):
    selisih = np.abs(alternatif_matrix - nilai_input)
    m, n = selisih.shape
    normalisasi = np.zeros((m, n))

    for j in range(n):
        if atribut[j] == 1: # Benefit
            normalisasi[:, j] = selisih[:, j] / np.max(selisih[:, j])
        else: # Cost
            normalisasi[:, j] = np.min(selisih[:, j]) / selisih[:, j]

    hasil_preferensi = np.dot(normalisasi, bobot)
    return hasil_preferensi, normalisasi

# Input User
def user_input(df, deskripsi_kriteria):
    st.markdown("""
<div class="input-section">
    <h3>Input Kondisi Lahan Anda</h3>
</div>
""", unsafe_allow_html=True)

    input_user = {}

```

```

col1, col2, col3 = st.columns(3)
kriteria_cols = [
    (['N', 'P', 'K'], col1),
    (['temperature', 'humidity'], col2),
    (['ph', 'rainfall'], col3)
]

for kriteria_group, kolom in kriteria_cols:
    with kolom:
        for kriteria in kriteria_group:
            if kriteria in deskripsi_kriteria:
                min_val, max_val = float(df[kriteria].min()), float(df[kriteria].max())
                default_val = round((min_val + max_val) / 2, 2)
                step = 0.1 if kriteria in ['temperature', 'humidity', 'ph', 'rainfall'] else 1.0

                input_user[kriteria] = st.slider(
                    label=f'{deskripsi_kriteria[kriteria]}',
                    min_value=min_val,
                    max_value=max_val,
                    value=default_val,
                    step=step,
                    key=f"slider_{kriteria}"
                )
            else:
                input_user[kriteria] = st.number_input(
                    label=f'{deskripsi_kriteria[kriteria]}',
                    min_value=0.0,
                    max_value=1.0,
                    value=0.5
                )

    return input_user

# Hasil
def tampilan_hasil(preferensi, labels, crop_avg, input_user, normalisasi):
    df_hasil = pd.DataFrame({
        'Tanaman': labels,
        'Nilai Preferensi': preferensi
    }).sort_values(by='Nilai Preferensi', ascending=False).reset_index(drop=True)
    df_hasil.index += 1

    st.markdown(f"""
    <div class="recommendation-card">
        <h2>🌿 Rekomendasi Terbaik</h2>
        <h1>{df_hasil.iloc[0]['Tanaman']}</h1>
        <p>Skor Preferensi: <strong>{df_hasil.iloc[0]['Nilai Preferensi']:.4f}</strong></p>
        <p>Tanaman ini paling cocok dengan kondisi lahan Anda!</p>
    </div>
    """, unsafe_allow_html=True)

    tab1, tab2, tab3, tab4, tab5 = st.tabs([
        '📋 Kondisi Lahan',
        '⚖️ Bobot Kriteria',
        '🔢 Matriks Data',
        '📈 Normalisasi SAW',
        '📊 Semua Hasil'
    ])

    #Kondisi Lahan

```

```

with tab1:
    st.markdown("### 📄 Data Kondisi Lahan yang Diinput:")
    df_input = pd.DataFrame(input_user, index=["Nilai"])
    df_input_renamed = df_input.rename(columns={
        'N': "Nitrogen (N)",
        'P': "Fosfor (P)",
        'K': "Kalium (K)",
        'temperature': "Suhu (°C)",
        'humidity': "Kelembapan (%)",
        'ph': "pH Tanah",
        'rainfall': "Curah Hujan (mm)"
    })
    st.dataframe(df_input_renamed.T, use_container_width=True)

#Bobot Kriteria
with tab2:
    st.markdown("### ⚖️ Bobot Kriteria yang Digunakan:")
    df_bobot = pd.DataFrame({
        'Kriteria': [
            "Nitrogen (N)", "Fosfor (P)", "Kalium (K)",
            "Suhu", "Kelembapan", "pH Tanah", "Curah Hujan"
        ],
        'Bobot': [0.15, 0.15, 0.15, 0.2, 0.1, 0.15, 0.1],
        'Jenis': ["Benefit", "Benefit", "Benefit", "Cost",
        "Cost", "Cost", "Cost"]
    })
    st.dataframe(df_bobot, use_container_width=True,
    hide_index=True)

#Matriks Data Rata-rata
with tab3:
    st.markdown("### 📈 Data Rata-rata Kondisi Optimal Tanaman:")
    crop_avg_display = pd.DataFrame(
        crop_avg.values.round(2),
        columns=[
            "Nitrogen (N)", "Fosfor (P)", "Kalium (K)",
            "Suhu (°C)", "Kelembapan (%)", "pH Tanah", "Curah Hujan (mm)"
        ],
        index=crop_avg.index
    )
    st.dataframe(crop_avg_display, use_container_width=True)

#Normalisasi SAW
with tab4:
    st.markdown("### 📈 Matriks Hasil Normalisasi (SAW):")
    df_normalisasi = pd.DataFrame(
        normalisasi.round(4),
        columns=[
            "Nitrogen (N)", "Fosfor (P)", "Kalium (K)",
            "Suhu (°C)", "Kelembapan (%)", "pH Tanah", "Curah Hujan (mm)"
        ],
        index=labels
    )
    st.dataframe(df_normalisasi, use_container_width=True)

```

```

#Semua Hasil
with tab5:
    st.markdown("### 📈 Semua Hasil Penilaian:")
    df_hasil_display = df_hasil.copy()
    df_hasil_display['Ranking'] = df_hasil_display.index
    df_hasil_display['Skor (%)'] = (df_hasil_display['Nilai Preferensi'] * 100).round(2)
    st.dataframe(
        df_hasil_display[['Ranking', 'Tanaman', 'Nilai Preferensi', 'Skor (%)']],
        use_container_width=True,
        hide_index=True
    )

# Main App
def main():
    dataset = pd.read_csv("crop_recommendation.csv")

    kriteria = ['N', 'P', 'K', 'temperature', 'humidity', 'ph',
    'rainfall']
    deskripsi_kriteria = {
        'N': "🧪 Nitrogen (N) di Tanah (kg/ha)",
        'P': "🧪 Fosfor (P) di Tanah (kg/ha)",
        'K': "🧪 Kalium (K) di Tanah (kg/ha)",
        'temperature': "🌡️ Suhu (°C)",
        'humidity': "💧 Kelembapan Relatif (%)",
        'ph': "🔬 pH Tanah",
        'rainfall': "🌧️ Curah Hujan (mm)"
    }

    crop_avg = dataset.groupby('label')[kriteria].mean()
    alternatif_matrix = crop_avg.values
    label_tanaman = crop_avg.index.tolist()

    bobot = np.array([0.15, 0.15, 0.15, 0.2, 0.1, 0.15, 0.1])
    atribut = [1, 1, 1, 0, 0, 0, 0]

    input_user = user_input(dataset, deskripsi_kriteria)

    coll, col2, col3 = st.columns([1, 2, 1])
    with col2:
        if st.button("🔍 Analisis & Cari Rekomendasi",
key="analyze_button"):
            nilai_input = np.array([input_user[k] for k in kriteria])
            skor_preferensi, normalisasi = saw(alternatif_matrix,
nilai_input, bobot, atribut)
            tampilan_hasil(skor_preferensi, label_tanaman,
crop_avg, input_user, normalisasi)

    if __name__ == "__main__":
        main()

```

B. Penjelasan Perbagian

1) Import library

```
import streamlit as st
import pandas as pd
import numpy as np

st.set_page_config(page_title="SPK Pemilihan Tanaman",
layout="wide")
```

Fungsi ini mengatur halaman dan mengimpor library yang diperlukan

2) Custom CSS untuk Tampilan

```
st.markdown("""
<style>
    .main-header {
        text-align: center;
        padding: 2rem 0;
        background: linear-gradient(90deg, #4CAF50, #8BC34A);
        border-radius: 10px;
        margin-bottom: 2rem;
        color: white;
    }

    .recommendation-card {
        background: linear-gradient(135deg, #11998e 0%, #38ef7d
100%);
        padding: 2rem;
        border-radius: 20px;
        color: white;
        text-align: center;
        box-shadow: 0 10px 30px rgba(17, 153, 142, 0.3);
        margin: 2rem 0;
    }

    .input-section {
        background: #f8f9fa;
        padding-top: 1rem;
        border-radius: 15px;
        margin-bottom: 2rem;
        border: 1px solid #e9ecef;
    }

    .input-section h3 {
        color: #2e7d32;
        margin-bottom: 1rem;
        text-align: center;
    }

    .stButton > button {
        background: linear-gradient(90deg, #4CAF50, #8BC34A);
        color: white;
        border: none;
        padding: 0.75rem 2rem;
        border-radius: 25px;
        font-weight: bold;
        font-size: 1.1rem;
    }
</style>
""")
```

```

        width: 100%;
        transition: all 0.3s ease;
    }
</style>
"""", unsafe_allow_html=True)

```

Menambahkan tampilan visual agar aplikasi lebih menarik.

3) Fungsi SAW (Simple Additive Weighting)

```

def saw(alternatif_matrix, nilai_input, bobot, atribut):
    selisih = np.abs(alternatif_matrix - nilai_input)
    m, n = selisih.shape
    normalisasi = np.zeros((m, n))

    for j in range(n):
        if atribut[j] == 1: # Benefit
            normalisasi[:, j] = selisih[:, j] /
np.max(selisih[:, j])
        else: # Cost
            normalisasi[:, j] = np.min(selisih[:, j]) /
selisih[:, j]

    hasil_preferensi = np.dot(normalisasi, bobot)
    return hasil_pREFERENSI, normalisasi

```

Penjelasan:

- alternatif_matrix: data rata-rata tiap tanaman.
- nilai_input: data kondisi lahan user.
- bobot: tingkat kepentingan tiap kriteria.
- atribut: tipe kriteria (1 = benefit, 0 = cost).

Proses:

- Hitung selisih antara data tanaman dan kondisi lahan.
- Normalisasi nilai selisih:
 - Benefit: nilai / max(nilai)
 - Cost: min(nilai) / nilai
- Hitung skor preferensi: normalisasi × bobot

4) Fungsi Input User

```

def user_input(df, deskripsi_kriteria):
    st.markdown("""
    <div class="input-section">
        <h3>Input Kondisi Lahan Anda</h3>
    </div>
    """", unsafe_allow_html=True)

    input_user = {}
    col1, col2, col3 = st.columns(3)
    kriteria_cols = [
        ([['N', 'P', 'K'], col1],
         [['temperature', 'humidity'], col2],
         [['ph', 'rainfall'], col3])
    ]

    for kriteria_group, kolom in kriteria_cols:

```

```

        with kolom:
            for kriteria in kriteria_group:
                if kriteria in deskripsi_kriteria:
                    min_val, max_val =
float(df[kriteria].min()), float(df[kriteria].max())
                    default_val = round((min_val + max_val) / 2,
2)
                    step = 0.1 if kriteria in ['temperature',
'humidity', 'ph', 'rainfall'] else 1.0

                    input_user[kriteria] = st.slider(
                        label=f'{deskripsi_kriteria[kriteria]}',
                        min_value=min_val,
                        max_value=max_val,
                        value=default_val,
                        step=step,
                        key=f"slider_{kriteria}"
                    )

    return input_user

```

Penjelasan:

Fungsi ini menampilkan slider input untuk 7 kriteria : Nitrogen (N), Fosfor (P), Kalium (K), Suhu, Kelembapan, pH tanah, dan Curah hujan. Pengguna memilih nilai dalam rentang min-max yang diambil dari dataset. Dan mengembalikan nilai-nilai yang diinput oleh pengguna.

5) Fungsi Tampilkan Hasil

```

def tampilkan_hasil(preferensi, labels, crop_avg, input_user,
normalisasi):
    df_hasil = pd.DataFrame({
        'Tanaman': labels,
        'Nilai Preferensi': preferensi
    }).sort_values(by='Nilai Preferensi',
ascending=False).reset_index(drop=True)
    df_hasil.index += 1

    st.markdown(f"""
    <div class="recommendation-card">
        <h2>🌱 Rekomendasi Terbaik</h2>
        <h1>{df_hasil.iloc[0]['Tanaman']}</h1>
        <p>Skor Preferensi: <strong>{df_hasil.iloc[0]['Nilai Preferensi']:.4f}</strong></p>
        <p>Tanaman ini paling cocok dengan kondisi lahan Anda!</p>
    </div>
    """, unsafe_allow_html=True)

    tab1, tab2, tab3, tab4, tab5 = st.tabs([
        '📋 Kondisi Lahan',
        '⚖️ Bobot Kriteria',
        '🔢 Matriks Data',
        '📈 Normalisasi SAW',
        '📊 Semua Hasil'
    ])

```

```

#Kondisi Lahan
with tab1:
    st.markdown("### 📄 Data Kondisi Lahan yang Diinput:")
    df_input = pd.DataFrame(input_user, index=["Nilai"])
    df_input_renamed = df_input.rename(columns={
        'N': "Nitrogen (N)",
        'P': "Fosfor (P)",
        'K': "Kalium (K)",
        'temperature': "Suhu (°C)",
        'humidity': "Kelembapan (%)",
        'ph': "pH Tanah",
        'rainfall': "Curah Hujan (mm)"
    })
    st.dataframe(df_input_renamed.T,
use_container_width=True)

#Bobot Kriteria
with tab2:
    st.markdown("### 💡 Bobot Kriteria yang Digunakan:")
    df_bobot = pd.DataFrame({
        'Kriteria': [
            "Nitrogen (N)", "Fosfor (P)", "Kalium (K)",
            "Suhu", "Kelembapan", "pH Tanah", "Curah Hujan"
        ],
        'Bobot': [0.15, 0.15, 0.15, 0.2, 0.1, 0.15, 0.1],
        'Jenis': ["Benefit", "Benefit", "Benefit", "Cost",
"Cost", "Cost", "Cost"]
    })
    st.dataframe(df_bobot, use_container_width=True,
hide_index=True)

#Matriks Data Rata-rata
with tab3:
    st.markdown("### 📊 Data Rata-rata Kondisi Optimal
Tanaman:")
    crop_avg_display = pd.DataFrame(
        crop_avg.values.round(2),
        columns=[
            "Nitrogen (N)", "Fosfor (P)", "Kalium (K)",
            "Suhu (°C)", "Kelembapan (%)", "pH Tanah",
            "Curah Hujan (mm)"
        ],
        index=crop_avg.index
    )
    st.dataframe(crop_avg_display, use_container_width=True)

#Normalisasi SAW
with tab4:
    st.markdown("### 📈 Matriks Hasil Normalisasi (SAW):")
    df_normalisasi = pd.DataFrame(
        normalisasi.round(4),
        columns=[
            "Nitrogen (N)", "Fosfor (P)", "Kalium (K)",
            "Suhu (°C)", "Kelembapan (%)", "pH Tanah",
            "Curah Hujan (mm)"
        ],
        index=labels
    )

```

```

st.dataframe(df_normalisasi, use_container_width=True)

#Semua Hasil
with tab5:
    st.markdown("### 📈 Semua Hasil Penilaian:")
    df_hasil_display = df_hasil.copy()
    df_hasil_display['Ranking'] = df_hasil_display.index
    df_hasil_display['Skor (%)'] = (df_hasil_display['Nilai Preferensi'] * 100).round(2)
    st.dataframe(
        df_hasil_display[['Ranking', 'Tanaman', 'Nilai Preferensi', 'Skor (%)']],
        use_container_width=True,
        hide_index=True
    )

```

Penjelasan:

- Menampilkan hasil analisis kepada pengguna.
- Menampilkan kartu rekomendasi terbaik dengan tanaman yang memiliki skor preferensi tertinggi.

- Menggunakan tabs untuk menyajikan informasi detail dalam beberapa bagian:
 - Kondisi Lahan yang Diinput.
 - Bobot Kriteria yang Digunakan.
 - Matriks Data Rata-rata Tanaman.
 - Matriks Hasil Normalisasi SAW.
 - Semua Hasil Peringkat Tanaman.

6) Fungsi Utama Aplikasi

```

def main():
    dataset = pd.read_csv("crop_recommendation.csv")

    kriteria = ['N', 'P', 'K', 'temperature', 'humidity', 'ph',
    'rainfall']
    deskripsi_kriteria = {
        'N': "🧪 Nitrogen (N) di Tanah (kg/ha)",
        'P': "🧪 Fosfor (P) di Tanah (kg/ha)",
        'K': "🧪 Kalium (K) di Tanah (kg/ha)",
        'temperature': "🌡️ Suhu (°C)",
        'humidity': "💧 Kelembapan Relatif (%)",
        'ph': "⼟ pH Tanah",
        'rainfall': "🌧️ Curah Hujan (mm)"
    }

    crop_avg = dataset.groupby('label')[kriteria].mean()
    alternatif_matrix = crop_avg.values
    label_tanaman = crop_avg.index.tolist()

    bobot = np.array([0.15, 0.15, 0.15, 0.2, 0.1, 0.15, 0.1])
    atribut = [1, 1, 1, 0, 0, 0, 0]

```

```

input_user = user_input(dataset, deskripsi_kriteria)

col1, col2, col3 = st.columns([1, 2, 1])
with col2:
    if st.button("🔍 Analisis & Cari Rekomendasi",
key="analyze_button"):
        nilai_input = np.array([input_user[k] for k in
kriteria])
        skor_preferensi, normalisasi =
saw(alternatif_matrix, nilai_input, bobot, atribut)
        tampilkan_hasil(skor_preferensi, label_tanaman,
crop_avg, input_user, normalisasi)

if __name__ == "__main__":
    main()

```

Penjelasan:

- Fungsi utama yang mengatur alur kerja aplikasi.
- Memuat Data: Membaca file crop_recommendation.csv ke dalam DataFrame Pandas.
- Persiapan Data dan Kriteria:
 - Mendefinisikan daftar kriteria dan deskripsinya.
 - Menghitung crop_avg: DataFrame yang berisi nilai rata-rata setiap kriteria untuk setiap jenis tanaman. Ini akan menjadi matriks alternatif dalam metode SAW.
 - Menentukan bobot kriteria dan tipe atribut benefit/cost.
- Input Pengguna: Memanggil user_input untuk mendapatkan input dari pengguna.
- Tombol Analisis: Jika tombol "Analisis & Cari Rekomendasi" diklik:
 - Mengubah input pengguna menjadi array NumPy.
 - Memanggil fungsi saw untuk melakukan perhitungan.
 - Memanggil tampilkan_hasil untuk menampilkan hasilnya.

C. Hasil Program

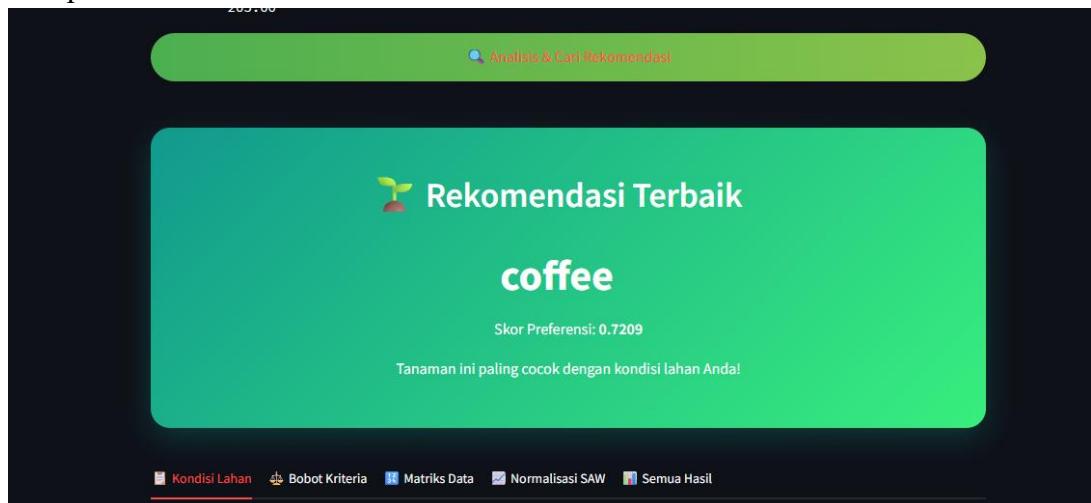
1) Tampilan Header dan Input



Gambar 2.1

Bagian atas akan menampilkan header utama aplikasi. Di bawahnya, terdapat bagian "Input Kondisi Lahan Anda". Input tersebut berisi slider untuk kriteria tertentu (N, P, K, Suhu, Kelembapan, pH, dan Curah Hujan). Pengguna dapat menggeser slider ini untuk memasukkan nilai. Di bagian tengah bawah area input, terdapat tombol "Analisis & Cari Rekomendasi".

2) Tampilan Hasil Rekomendasi



Gambar 2.2

Setelah pengguna menyesuaikan slider dan mengklik tombol "Analisis & Cari Rekomendasi", halaman akan diperbarui. Akan menampilkan "Rekomendasi Terbaik" dalam sebuah card yang berisi nama tanaman dengan skor preferensi tertinggi. Di bawah card ini, akan ada beberapa tab untuk melihat detail lebih lanjut.

3) Tampilan Tab Detail

- Tab Kondisi Lahan

Data Kondisi Lahan yang Diinput:	
	Nilai
Nitrogen (N)	70
Fosfor (P)	75
Kalium (K)	105
Suhu (°C)	26.25
Kelembapan (%)	57.12
pH Tanah	6.72
Curah Hujan (mm)	159.39

Gambar 2.3

Menampilkan tabel nilai-nilai kriteria yang telah diinput oleh pengguna.

- Tab Bobot Kriteria

The screenshot shows a software interface with a dark header bar containing five tabs: 'Kondisi Lahan', 'Bobot Kriteria' (highlighted in orange), 'Matriks Data', 'Normalisasi SAW', and 'Semua Hasil'. Below the header is a section titled 'Bobot Kriteria yang Digunakan:' with a small icon. A table follows, with columns 'Kriteria', 'Bobot', and 'Jenis'. The data is as follows:

Kriteria	Bobot	Jenis
Nitrogen (N)	0.15	Benefit
Fosfor (P)	0.15	Benefit
Kalium (K)	0.15	Benefit
Suhu	0.2	Cost
Kelembapan	0.1	Cost
pH Tanah	0.15	Cost
Curah Hujan	0.1	Cost

Gambar 2.4

Menampilkan tabel bobot statis yang digunakan untuk setiap kriteria beserta jenis atributnya (Benefit/Cost).

- Tab Matrix Data

The screenshot shows a software interface with a dark header bar containing five tabs: 'Kondisi Lahan', 'Bobot Kriteria' (highlighted in blue), 'Matriks Data' (highlighted in orange), 'Normalisasi SAW', and 'Semua Hasil'. Below the header is a section titled 'Data Rata-rata Kondisi Optimal Tanaman:' with a small icon. A table follows, with columns 'label', 'Nitrogen (N)', 'Fosfor (P)', 'Kalium (K)', 'Suhu (°C)', 'Kelembapan (%)', 'pH Tanah', and 'Curah Hujan (mm)'. The data is as follows:

label	Nitrogen (N)	Fosfor (P)	Kalium (K)	Suhu (°C)	Kelembapan (%)	pH Tanah	Curah Hujan (mm)
apple	20.8	134.22	199.89	22.63	92.33	5.93	112.65
banana	100.23	82.01	50.05	27.38	80.36	5.98	104.63
blackgram	40.02	67.47	19.24	29.97	65.12	7.13	67.88
chickpea	40.09	67.79	79.92	18.87	16.86	7.34	80.06
coconut	21.98	16.93	30.59	27.41	94.84	5.98	175.69
coffee	101.2	28.74	29.94	25.54	58.87	6.79	158.07
cotton	117.77	46.24	19.56	23.99	79.84	6.91	80.4
grapes	23.18	132.53	200.11	23.85	81.88	6.03	69.61
jute	78.4	46.86	39.99	24.96	79.64	6.73	174.79
kidneybeans	20.75	67.54	20.05	20.12	21.61	5.75	105.92

Gambar 2.5

Menampilkan tabel data rata-rata kondisi optimal (nilai N, P, K, Suhu, Kelembapan, pH Tanah, dan Curah Hujan) untuk setiap jenis tanaman yang ada dalam dataset. Ini adalah matriks alternatif yang digunakan dalam perhitungan SAW.

- Tab Normalisasi SAW

	Nitrogen (N)	Fosfor (P)	Kalium (K)	Suhu (°C)	Kelembapan (%)	pH Tanah	Curah Hujan (mm)
apple	0.9604	1	0.9977	0.1819	0.0497	0.005	0.0283
banana	0.5901	0.1184	0.5778	0.5842	0.0753	0.0054	0.0242
blackgram	0.5852	0.1272	0.9017	0.1768	0.2188	0.0096	0.0145
chickpea	0.5838	0.1217	0.2637	0.0892	0.0435	0.0064	0.0167
coconut	0.9373	0.9806	0.7824	0.5675	0.0464	0.0053	0.0812
coffee	0.609	0.7812	0.7892	0.9277	1	0.0563	1
cotton	0.9325	0.4856	0.8983	0.2911	0.077	0.0205	0.0168
grapes	0.9139	0.9715	1	0.2742	0.0707	0.0057	0.0147
jute	0.164	0.4752	0.6835	0.5096	0.0777	0.3097	0.0859
kidneybeans	0.9614	0.126	0.8932	0.1073	0.0493	0.0041	0.0248

Gambar 2.6

Menampilkan matriks hasil normalisasi dari perhitungan SAW. Nilai-nilai ini menunjukkan skor kecocokan (setelah normalisasi) setiap tanaman terhadap setiap kriteria berdasarkan input pengguna.

- Tab Semua Hasil

Ranking	Tanaman	Nilai Preferensi	Skor (%)
1	coffee	0.7209	72.09
2	mungbean	0.5632	56.32
3	coconut	0.5321	53.21
4	watermelon	0.5296	52.96
5	grapes	0.497	49.7
6	orange	0.493	49.3
7	apple	0.4886	48.86
8	mothbeans	0.4627	46.27
9	mango	0.4401	44.01
10	pomegranate	0.4341	43.41

Gambar 2.7

Menampilkan tabel lengkap yang berisi semua tanaman, diurutkan berdasarkan nilai preferensi (Skor SAW) dari yang tertinggi ke terendah, beserta peringkatnya.

BAB III

JADWAL PENGERJAAN DAN PEMBAGIAN TUGAS

3.1 Jadwal Penggerjaan

No	Kegiatan	2025									
		Minggu									
		1	2	3	4	1	2	3	4	1	
1	Menentukan Judul dan Mencari dataset			✓							
2	Membuat program				✓	✓					
3	Membuat laporan					✓					
4	Mempresentasikan hasil					✓					

Tabel 3.1

3.2 Pembagian Tugas

Nasarudin	Ahmad Zainur Fadli
1. Mencari dataset 2. Implementasi Fungsi Inti SAW 3. Membuat laporan	1. Implementasi Fungsi Tampilan Hasil 2. Implementasi Fungsi User Input 3. Implementasi Fungsi Main 4. Implementasi UI dengan streamlit dan css

Tabel 3.2

BAB IV

KESIMPULAN DAN SARAN

4.1 Kesimpulan

Berdasarkan hasil pengembangan dan implementasi sistem rekomendasi tanaman berbasis metode *Simple Additive Weighting* (SAW), dapat disimpulkan bahwa:

1. Metode SAW mampu digunakan secara efektif dalam sistem pendukung keputusan untuk memilih alternatif terbaik berdasarkan banyak kriteria. Proses normalisasi dan pembobotan memberikan hasil perhitungan yang objektif dan terstruktur.
2. Sistem yang dibangun dalam bahasa Python dapat menghitung nilai akhir dari masing-masing alternatif tanaman secara otomatis, berdasarkan input kondisi lingkungan seperti jenis tanah, suhu, kelembapan, kebutuhan air, dan intensitas cahaya.
3. Alternatif dengan skor tertinggi dari hasil perhitungan SAW secara otomatis ditampilkan sebagai rekomendasi terbaik. Hal ini membantu pengguna dalam menentukan pilihan tanaman yang paling sesuai dengan kondisi aktual di lapangan.
4. Proyek ini juga berfungsi sebagai media pembelajaran dalam memahami penerapan logika pengambilan keputusan berbasis multikriteria serta pengembangan sistem SPK sederhana berbasis bahasa pemrograman.

4.2 Saran

Adapun beberapa saran untuk pengembangan lebih lanjut dari sistem ini adalah sebagai berikut:

1. Penambahan lebih banyak data tanaman dan kriteria agar sistem dapat memberikan rekomendasi yang lebih akurat dan fleksibel sesuai dengan kebutuhan pengguna dari berbagai wilayah.
2. Perlu dilakukan validasi dan pengujian lebih lanjut terhadap sistem menggunakan data kondisi lapangan yang riil untuk mengetahui sejauh mana keakuratan hasil rekomendasi.

DAFTAR PUSTAKA

- Administrator. (2020). Mengenal Web Programming. Diakses pada 11 April 2020, dari <http://technopark.surakarta.go.id/id/media-publik/komputer-teknologi-informasi/187-mengenal-web-programming>
- Fishburn, P. C. (1967). Additive Utilities with Incomplete Product Sets: Applications to Priorities and Assignments. *Operations Research*, 15(3), 537–542.
- MacCrimmon, K. R. (1968). Decision Making Among Multiple-Attribute Alternatives: A Survey and Consolidated Approach. RAND Corporation.
- Nofriansyah, D. (2020). Penerapan Metode SAW untuk Sistem Pendukung Keputusan. *Jurnal Teknologi dan Sistem Informasi*, 1(1), 12–20.
- Suryadi, K., & Ramdhani, M. A. (2016). Metode dan Aplikasi Sistem Pendukung Keputusan. Bandung: Pustaka Setia.
- UPN “Veteran” Yogyakarta. (2024). Modul Praktikum Sistem Cerdas dan Pendukung Keputusan: BAB VI - SPK dengan Simple Additive Weighting. Fakultas Teknik Industri, Prodi Informatika.

