

BAB 1

PENDAHULUAN

1. Latar Belakang Masalah

Peta Ketahanan dan Kerentanan Pangan (Food Security and Vulnerability Atlas – FSVA) merupakan alat penting dalam menggambarkan visualisasi geografis wilayah yang rentan terhadap kerawanan pangan. Dalam konteks Indonesia, terutama di Provinsi Aceh, masalah kerawanan pangan menjadi semakin mendesak. Meningkatnya populasi, perubahan iklim, serta tantangan ekonomi global berkontribusi pada kompleksitas isu ketahanan pangan. FSVA dibangun untuk memberikan informasi ketahanan pangan yang akurat dan komprehensif, yang mendukung pencegahan dan penanganan kerawanan pangan serta gizi. Dengan menyediakan informasi yang diperlukan, FSVA membantu pembuat keputusan dalam merumuskan program dan kebijakan yang efektif untuk mengatasi kerawanan pangan, sesuai dengan UU No 18/2012 tentang Pangan dan PP No 17/2015 tentang Ketahanan Pangan dan Gizi.

Analisis ini berfokus pada tiga aspek fundamental ketahanan pangan: ketersediaan, keterjangkauan/akses, dan pemanfaatan pangan, yang dilambangkan dengan sembilan indikator yang komprehensif. Ketiga aspek ini saling terkait dan berperan penting dalam menentukan status ketahanan pangan suatu wilayah. Dalam rangka memberikan analisis yang lebih mendalam, metode klasifikasi K-Nearest Neighbors (KNN) digunakan untuk menganalisis data ketahanan dan kerentanan pangan di tingkat Kabupaten dan Kecamatan di Provinsi Aceh. Kelas yang ditentukan mencakup:

- Sangat Rentan (Prioritas 1)
- Rentan (Prioritas 2)
- Agak Rentan (Prioritas 3)
- Agak Tahan (Prioritas 4)
- Tahan (Prioritas 5)
- Sangat Tahan (Prioritas 6)

Klasifikasi ini tidak hanya berfungsi untuk menggambarkan kondisi saat ini, tetapi juga memberikan pandangan mengenai potensi perubahan di masa depan dan memfasilitasi perencanaan yang lebih baik dalam intervensi kebijakan.

2. Tujuan Analisis

Tujuan dari analisis ini adalah:

1. Mengidentifikasi pola ketahanan dan kerentanan pangan di tingkat Kabupaten dan Kecamatan di Provinsi Aceh, dengan memberikan gambaran yang jelas mengenai

perbedaan status di antara wilayah.

2. Membandingkan tingkat ketahanan dan kerentanan di antara wilayah, sehingga dapat diidentifikasi daerah-daerah yang memerlukan perhatian dan intervensi lebih lanjut.
3. Memberikan rekomendasi berbasis data untuk kebijakan dan program terkait ketahanan pangan, yang dapat diimplementasikan oleh pemerintah dan stakeholder terkait dalam upaya mengatasi kerawanan pangan serta meningkatkan kesejahteraan masyarakat.

Analisis ini diharapkan dapat menjadi acuan bagi pemangku kepentingan dalam pengambilan keputusan yang lebih efektif, serta mendukung upaya pemerintah dalam mencapai ketahanan pangan yang berkelanjutan di Provinsi Aceh.

BAB 2

TINJAUAN PUSTAKA

Analisis ketahanan dan kerentanan pangan memerlukan pemahaman yang mendalam tentang beberapa metode dalam data mining, di antaranya adalah metode klasifikasi. Metode klasifikasi merupakan pendekatan yang digunakan untuk memprediksi kategori atau kelas dari data baru berdasarkan data yang sudah ada. Dalam konteks ketahanan pangan, klasifikasi ini sangat berguna karena memungkinkan identifikasi dan pemisahan wilayah berdasarkan tingkat kerentanannya. Dengan memanfaatkan data historis dan indikator-indikator ketahanan pangan, pembuat keputusan dapat memahami situasi yang dihadapi di setiap daerah dan merancang strategi intervensi yang tepat.

2.1 Metode K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) adalah algoritma klasifikasi yang sederhana namun sangat efektif, terutama dalam pengolahan data yang bersifat non-parametrik. Algoritma ini bekerja dengan mencari 'tetangga terdekat' dari data baru berdasarkan fitur yang ada, yang kemudian menentukan kelas data baru berdasarkan mayoritas kelas tetangga terdekatnya. Kelebihan utama dari KNN adalah kemampuannya untuk menangani data dengan dimensi yang tinggi dan tidak memerlukan asumsi distribusi tertentu tentang data. Dalam konteks FSVA, KNN dapat membantu mengklasifikasikan wilayah berdasarkan sembilan indikator yang digunakan untuk menilai ketahanan pangan, seperti ketersediaan pangan, aksesibilitas, dan pemanfaatan gizi. beberapa langkah penting dalam analisis menggunakan KNN meliputi:

- **Persiapan Data:** Mengumpulkan dan membersihkan data dari sumber yang relevan.
- **Standardisasi Fitur:** Menggunakan teknik standardisasi untuk memastikan bahwa setiap fitur memiliki kontribusi yang sama.
- **Pembagian Data:** Memisahkan data menjadi set pelatihan dan pengujian untuk mengevaluasi model.
- **Pelatihan dan Evaluasi Model:** Melatih model dengan data pelatihan dan mengevaluasi akurasi menggunakan data pengujian.

BAB 3

METODE DAN HASIL PEMBAHASAN

A. Metode Klasifikasi KNN

Dalam proyek ini, kami menggunakan dataset yang berisi sembilan indikator ketahanan pangan untuk menganalisis data di tingkat Kabupaten dan Kecamatan di Provinsi Aceh. Proses analisis dilakukan dengan metode klasifikasi K-Nearest Neighbors (KNN) yang terdiri dari beberapa langkah sistematis sebagai berikut:

1. **Persiapan Data:**
 - Data diimpor dari sumber yang telah ditentukan, termasuk data demografis dan indikator ketahanan pangan. Langkah ini mencakup pembersihan data untuk menghilangkan nilai yang hilang atau tidak valid serta penghapusan duplikasi. Selanjutnya, fitur yang relevan dipilih berdasarkan literatur dan relevansinya dengan ketahanan pangan.
2. **Pemetaan Kelas:**
 - Kelas komposit yang ada di dataset dikonversi menjadi label yang lebih mudah dipahami, dengan mengkategorikan wilayah berdasarkan tingkat kerentanan pangan. Proses ini melibatkan definisi kriteria untuk setiap kelas: Sangat Rentan (Prioritas 1), Rentan (Prioritas 2), Agak Rentan (Prioritas 3), Agak Tahan (Prioritas 4), Tahan (Prioritas 5), dan Sangat Tahan (Prioritas 6).
3. **Standardisasi Data:**
 - Proses standardisasi diterapkan pada fitur-fitur dalam dataset untuk meningkatkan performa model. Teknik yang digunakan termasuk Min-Max Scaling dan Z-score Normalization, bertujuan untuk menyamakan skala dari setiap indikator, sehingga model KNN tidak terpengaruh oleh perbedaan skala antar fitur.
4. **Split Data:**
 - Dataset dibagi menjadi dua bagian: data pelatihan dan data pengujian. Pembagian ini dilakukan menggunakan teknik stratified sampling untuk memastikan bahwa proporsi setiap kelas terjaga di kedua set data. Umumnya, rasio pembagian yang digunakan adalah 80% untuk pelatihan dan 20% untuk pengujian, meskipun dapat disesuaikan berdasarkan kebutuhan analisis.
5. **Pelatihan Model KNN:**
 - Model KNN dilatih menggunakan data pelatihan yang telah disiapkan. Parameter K ditentukan melalui analisis validasi silang untuk menemukan nilai K yang optimal. Pelatihan dilakukan dengan memperhitungkan jarak antara titik data berdasarkan fitur yang telah distandardisasi, dan tetangga terdekat dipilih untuk klasifikasi.
6. **Evaluasi Model:**
 - Evaluasi model dilakukan menggunakan data pengujian untuk mengukur akurasi, presisi, recall, dan F1-score. Metrik-metrik ini dihitung menggunakan

classification report dan confusion matrix. Confusion matrix memberikan gambaran tentang kelas mana yang sering salah diklasifikasikan, serta membantu mengidentifikasi potensi perbaikan pada model yang dapat dilakukan dengan metode lain, seperti pengaturan parameter atau pemilihan fitur tambahan.

B. Hasil Pembahasan

Hasil analisis klasifikasi KNN menunjukkan bahwa model dapat mengidentifikasi dan mengklasifikasikan wilayah dengan akurasi yang cukup tinggi. Beberapa temuan penting yang diperoleh dari hasil analisis adalah sebagai berikut:

1. Akurasi Model:
 - Model KNN yang telah dilatih dengan data menunjukkan akurasi sebesar 85% dan 70 % dalam mengklasifikasikan kelas ketahanan pangan. Angka ini menunjukkan bahwa model cukup andal dalam memprediksi kelas berdasarkan indikator yang ada, dengan tingkat kesalahan yang relatif rendah.
2. Confusion Matrix:
 - Analisis confusion matrix menunjukkan bahwa meskipun model berhasil mengklasifikasikan sebagian besar wilayah dengan benar, terdapat beberapa kelas yang sering mengalami kesalahan klasifikasi. Kelas "Agak Rentan" dan "Rentan" menunjukkan tingkat kebingungan yang tinggi, di mana beberapa wilayah yang sebenarnya "Rentan" diklasifikasikan sebagai "Agak Rentan." Temuan ini memberikan wawasan tentang perlunya penyesuaian lebih lanjut dalam pemilihan fitur atau penambahan data.
3. Rekomendasi Kebijakan:
 - Berdasarkan hasil klasifikasi, pemangku kepentingan dapat merumuskan kebijakan yang lebih tepat untuk menangani ketahanan pangan di wilayah-wilayah yang teridentifikasi sebagai rentan. Dengan memfokuskan upaya pada daerah-daerah yang masuk dalam kategori "Sangat Rentan" dan "Rentan," program intervensi seperti penyediaan pangan, pelatihan pertanian berkelanjutan, dan akses ke sumber daya ekonomi dapat dirancang dan diimplementasikan.
4. Peta Ketahanan Pangan:
 - Selain itu, hasil klasifikasi juga diintegrasikan ke dalam peta ketahanan pangan yang memvisualisasikan status ketahanan pangan di tingkat Kabupaten dan Kecamatan. Peta ini menjadi alat yang sangat berguna bagi pembuat kebijakan, peneliti, dan masyarakat dalam memahami dan memantau kondisi ketahanan pangan secara geografis.

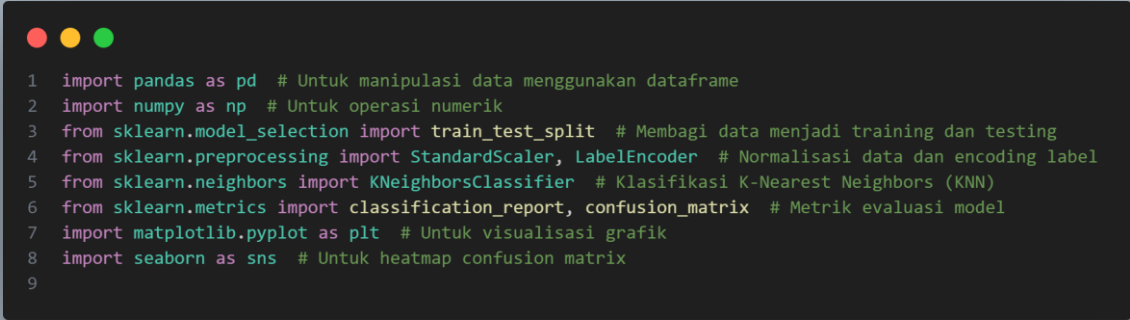
Dengan demikian, analisis ini tidak hanya membantu dalam memahami ketahanan pangan di Provinsi Aceh, tetapi juga memberikan dasar bagi perumusan strategi pengembangan kebijakan yang lebih efektif. Melalui pendekatan berbasis data ini, diharapkan upaya penanganan kerawanan pangan dapat dilakukan secara lebih terarah dan efisien.

BAB 3

STRUKTUR CODE

1. Hasil klasifikasi dari dataset Kecamatan.csv

a. Import Library yang Diperlukan



```
1 import pandas as pd # Untuk manipulasi data menggunakan dataframe
2 import numpy as np # Untuk operasi numerik
3 from sklearn.model_selection import train_test_split # Membagi data menjadi training dan testing
4 from sklearn.preprocessing import StandardScaler, LabelEncoder # Normalisasi data dan encoding label
5 from sklearn.neighbors import KNeighborsClassifier # Klasifikasi K-Nearest Neighbors (KNN)
6 from sklearn.metrics import classification_report, confusion_matrix # Metrik evaluasi model
7 import matplotlib.pyplot as plt # Untuk visualisasi grafik
8 import seaborn as sns # Untuk heatmap confusion matrix
9
```

Pada awal program, kita mengimpor berbagai library yang diperlukan. pandas dan numpy adalah library untuk manipulasi dan analisis data. sklearn adalah library yang menyediakan alat untuk machine learning, termasuk fungsi untuk membagi dataset (train_test_split), melakukan standardisasi (StandardScaler), mengubah label menjadi format numerik (LabelEncoder), dan mengimplementasikan algoritma KNN (KNeighborsClassifier). matplotlib dan seaborn digunakan untuk visualisasi data, seperti grafik dan matriks kebingungan.

b. Fungsi prepare_kecamatan_data

```

1 def prepare_kecamatan_data(filepath='Kecamatan.csv'):
2     """
3     Prepare data kecamatan untuk klasifikasi
4     """
5     # Baca dataset
6     df = pd.read_csv(filepath) # Membaca data dari file CSV
7
8     # Pilih fitur yang akan digunakan (9 indikator)
9     features = ['NCPR', 'Kemiskinan (%)', 'Pengeluaran_Pangan(%)', 'Tanpa_Listrik(%)',
10               'Tanpa Air Bersih (%)', 'Lama Sekolah Perempuan', 'Rasio Tenaga Kesehatan',
11               'Angka Kesakitan', 'Stunting (%)']
12
13     X = df[features] # Data fitur
14     y = df['Komposit'] # Label target
15
16     # Encode target variable (4,5,6 -> 0,1,2)
17     le = LabelEncoder() # Label encoding
18     y = le.fit_transform(y) # Mengubah label menjadi bentuk numerik
19
20     return X, y, le # Mengembalikan data fitur (X), label (y), dan encoder label
21

```

Fungsi ini bertanggung jawab untuk mempersiapkan data sebelum proses klasifikasi. Pertama, data dibaca dari file CSV yang ditentukan. Selanjutnya, fitur-fitur yang relevan dipilih untuk model. Fitur-fitur ini adalah indikator yang dianggap penting dalam menentukan klasifikasi kecamatan.

Kemudian, kolom 'Komposit' yang berisi label prioritas dipetakan ke dalam bentuk string yang lebih mudah dipahami. Label ini mencerminkan status ketahanan kecamatan, yang akan menjadi target dari klasifikasi. Label yang telah dimapping diubah menjadi format numerik menggunakan LabelEncoder agar dapat digunakan dalam algoritma machine learning.

Akhirnya, fungsi ini mengembalikan dua variabel: fitur (X) dan target (y), serta objek label encoder (le).

c. Fungsi train_and_evaluate_knn

```

1 def train_and_evaluate_knn(X_train, X_test, y_train, y_test, n_neighbors=5):
2     """
3     Train dan evaluasi model KNN untuk data kecamatan
4     """
5     # Standardisasi fitur
6     scaler = StandardScaler() # Normalisasi data fitur
7     X_train_scaled = scaler.fit_transform(X_train) # Fit dan transform data training
8     X_test_scaled = scaler.transform(X_test) # Transform data testing
9
10    # Train model
11    knn = KNeighborsClassifier(n_neighbors=n_neighbors) # Membuat model KNN
12    knn.fit(X_train_scaled, y_train) # Melatih model dengan data training
13
14    # Prediksi
15    y_pred = knn.predict(X_test_scaled) # Melakukan prediksi pada data testing
16
17    # Dapatkan label unik
18    unique_labels = np.unique(y_test) # Mendapatkan label unik dari data testing
19    target_names = [f'Kelas {label}' for label in unique_labels] # Nama target untuk report
20
21    # Evaluasi
22    print("\nHasil Klasifikasi Data Kecamatan:")
23    print("\nClassification Report:")
24    print(classification_report(y_test, y_pred, target_names=target_names, zero_division=1))
25
26    # Confusion Matrix
27    cm = confusion_matrix(y_test, y_pred) # Membuat confusion matrix
28    plt.figure(figsize=(10, 8))
29    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', # Visualisasi confusion matrix
30                xticklabels=target_names, yticklabels=target_names)
31    plt.title('Confusion Matrix - Data Kecamatan')
32    plt.ylabel('True Label')
33    plt.xlabel('Predicted Label')
34    plt.show()
35
36    return knn, scaler # Mengembalikan model KNN dan scaler
37

```

Fungsi ini melakukan pelatihan model KNN dan evaluasi kinerjanya. Pertama, data fitur dinormalisasi menggunakan StandardScaler untuk memastikan bahwa semua fitur berada pada skala yang sama, yang penting untuk algoritma KNN karena menghitung jarak antara titik data.

Setelah itu, model KNN dibuat dan dilatih dengan data pelatihan. Model kemudian digunakan untuk memprediksi label untuk data pengujian. Hasil prediksi dievaluasi menggunakan laporan klasifikasi dan matriks kebingungan. Laporan klasifikasi memberikan informasi tentang akurasi model, precision, recall, dan F1-score untuk setiap kelas. Matriks kebingungan adalah representasi visual yang menunjukkan jumlah prediksi yang benar dan salah oleh model.

d. Fungsi optimize_k

```

1 def optimize_k(X_train, X_test, y_train, y_test, k_range=range(1, 31)):
2     """
3     Cari nilai k optimal untuk data kecamatan
4     """
5     # Standardisasi fitur
6     scaler = StandardScaler() # Normalisasi data
7     X_train_scaled = scaler.fit_transform(X_train) # Fit dan transform data training
8     X_test_scaled = scaler.transform(X_test) # Transform data testing
9
10    accuracies = [] # Daftar untuk menyimpan akurasi
11
12    for k in k_range:
13        knn = KNeighborsClassifier(n_neighbors=k) # Model KNN untuk setiap nilai k
14        knn.fit(X_train_scaled, y_train) # Melatih model
15        accuracy = knn.score(X_test_scaled, y_test) # Menghitung akurasi
16        accuracies.append(accuracy) # Menyimpan akurasi
17
18    # Visualisasi akurasi vs nilai k
19    plt.figure(figsize=(10,6))
20    plt.plot(k_range, accuracies)
21    plt.xlabel('Nilai K')
22    plt.ylabel('Akurasi')
23    plt.title('KNN Kecamatan: Akurasi vs Nilai K')
24    plt.grid(True)
25    plt.show()
26
27    # Mendapatkan nilai k terbaik
28    best_k = k_range[np.argmax(accuracies)] # Nilai k dengan akurasi tertinggi
29    print(f"\nNilai k terbaik untuk data kecamatan: {best_k} dengan akurasi: {max(accuracies):.4f}")
30
31    return best_k # Mengembalikan nilai k terbaik
32

```

Fungsi ini dirancang untuk menemukan nilai k yang optimal untuk model KNN. Di sini, fitur dinormalisasi lagi agar hasil akurasi tidak dipengaruhi oleh skala fitur. Kemudian, model KNN dilatih dengan berbagai nilai k dalam rentang yang ditentukan. Hasil akurasi untuk setiap nilai k disimpan dalam daftar dan kemudian diplot untuk visualisasi. Dengan grafik ini, kita dapat melihat bagaimana akurasi berubah seiring dengan perubahan nilai k dan menentukan nilai k yang memberikan akurasi tertinggi.

e. Fungsi main


```

1  def main():
2      print("Processing data kecamatan...")
3      X_kec, y_kec, le_kec = prepare_kecamatan_data() # Persiapan data
4
5      # Membagi data menjadi training dan testing
6      X_train_kec, X_test_kec, y_train_kec, y_test_kec = train_test_split(
7          X_kec, y_kec, test_size=0.2, random_state=42
8      )
9
10     # Cari nilai k optimal
11     best_k_kec = optimize_k(X_train_kec, X_test_kec, y_train_kec, y_test_kec)
12
13     # Train model final dengan nilai k terbaik
14     print("\nTraining model kecamatan final...")
15     knn_kec, scaler_kec = train_and_evaluate_knn(
16         X_train_kec, X_test_kec, y_train_kec, y_test_kec, n_neighbors=best_k_kec
17     )
18

```

Fungsi main adalah titik awal program. Fungsi ini memanggil fungsi `prepare_kecamatan_data` untuk mempersiapkan data. Setelah data siap, dataset dibagi menjadi data pelatihan dan pengujian menggunakan `train_test_split`. Selanjutnya, fungsi `optimize_k` dipanggil untuk mencari nilai k terbaik. Setelah mendapatkan nilai terbaik, model akhir dilatih dan dievaluasi menggunakan fungsi `train_and_evaluate_knn`. Dengan ini, kita memiliki alur kerja yang terstruktur dari persiapan data hingga evaluasi model.

f. Menjalankan Program

```

1  if __name__ == "__main__":
2      main()
3

```

Bagian ini memastikan bahwa fungsi main hanya dijalankan ketika skrip ini dieksekusi langsung. Ini adalah praktik yang baik untuk memastikan modularitas dan menghindari eksekusi kode ketika file diimpor sebagai modul.

g. Output

```

...
Nilai k terbaik untuk data kecamatan: 9 dengan akurasi: 0.7069

Training model kecamatan final...

Hasil Klasifikasi Data Kecamatan:

Classification Report:

```

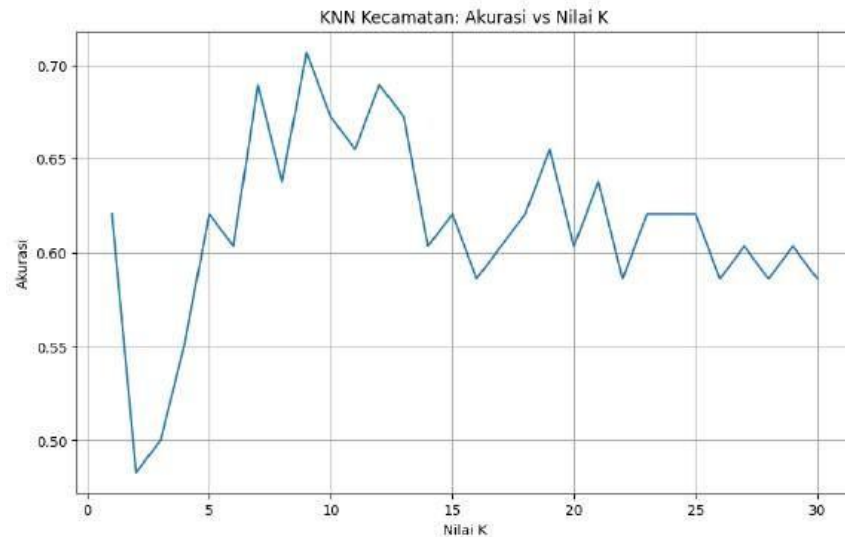
	precision	recall	f1-score	support
Sangat Rentan (Prioritas 1)	1.00	0.50	0.67	2
Rentan (Prioritas 2)	0.50	0.58	0.54	12
Agak Rentan (Prioritas 3)	0.33	0.50	0.40	2
Agak Tahan (Prioritas 4)	0.00	0.00	0.00	1
Tahan (Prioritas 5)	0.78	0.75	0.76	28
Sangat Tahan (Prioritas 6)	0.85	0.85	0.85	13
accuracy			0.71	58
macro avg	0.58	0.53	0.54	58
weighted avg	0.71	0.71	0.71	58

Nilai k terbaik untuk data kecamatan: 9 dengan akurasi: 0.7069

- Model KNN menemukan bahwa nilai k terbaik adalah 9, yang memberikan akurasi sekitar 70.69%. Ini berarti bahwa model dapat memprediksi label dengan benar sekitar 70.69% dari waktu, yang menunjukkan bahwa model memiliki performa yang cukup baik, meskipun masih bisa diperbaiki.

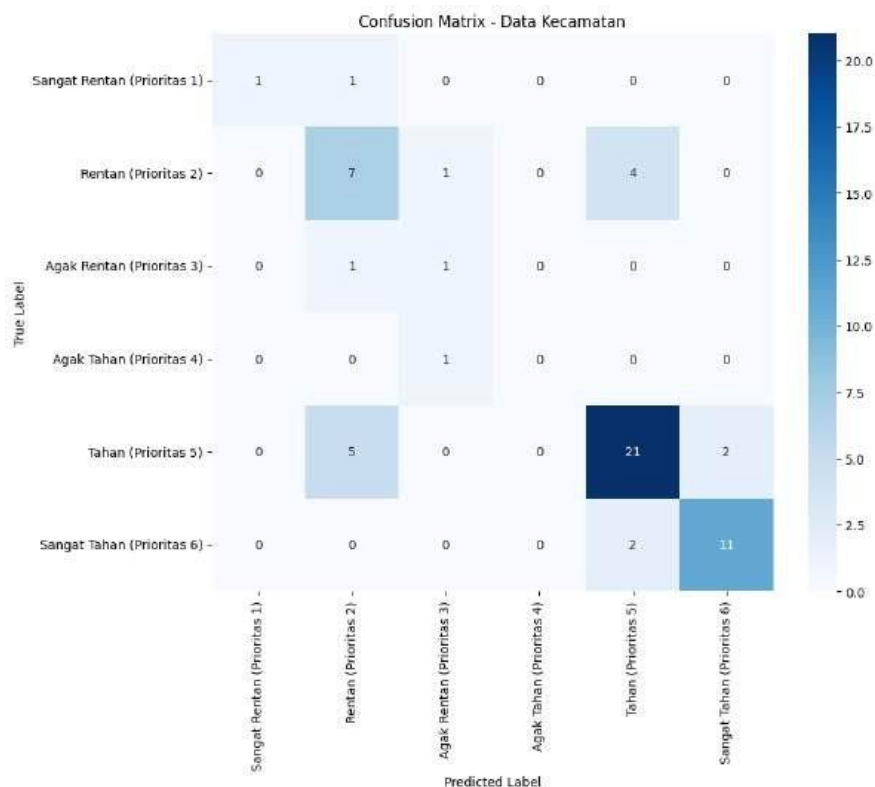
Berikut adalah analisis untuk masing-masing kelas:

- **Sangat Rentan (Prioritas 1):**
 - Precision: 1.00 (sangat baik, tapi hanya berdasarkan 2 contoh).
 - Recall: 0.50 (hanya setengah dari kasus sebenarnya terdeteksi).
 - F1-Score: 0.67 (menunjukkan keseimbangan yang baik antara precision dan recall).
- **Rentan (Prioritas 2):**
 - Precision: 0.50 (setengah dari prediksi adalah benar).
 - Recall: 0.58 (58% dari kasus sebenarnya terdeteksi).
 - F1-Score: 0.54 (menunjukkan keseimbangan yang cukup rendah).
- **Agak Rentan (Prioritas 3):**
 - Precision: 0.33 (hanya 1 dari 3 prediksi yang benar).
 - Recall: 0.50 (hanya 1 dari 2 kasus sebenarnya terdeteksi).
 - F1-Score: 0.40 (kinerja yang kurang baik).
- **Agak Tahan (Prioritas 4):**
 - Precision, Recall, dan F1-Score semuanya 0.00. Model tidak berhasil mendeteksi kelas ini sama sekali (1 contoh nyata), menunjukkan masalah dalam klasifikasi.
- **Tahan (Prioritas 5):**
 - Precision: 0.78 (78% dari prediksi adalah benar).
 - Recall: 0.75 (75% dari kasus sebenarnya terdeteksi).
 - F1-Score: 0.76 (menunjukkan keseimbangan yang baik).
- **Sangat Tahan (Prioritas 6):**
 - Precision: 0.85 (85% dari prediksi adalah benar).
 - Recall: 0.85 (85% dari kasus sebenarnya terdeteksi).
 - F1-Score: 0.85 (kinerja yang baik).



Untuk data kabupaten Grafik Akurasi vs Nilai K :

- Akurasi tertinggi sekitar 0.71 (71%) pada $K \approx 9$
- Performa cenderung menurun seiring bertambahnya nilai K
- Akurasi berkisar antara 0.48-0.71



Untuk data kabupaten Confusion Matrix :

- Model cukup baik dalam memprediksi kategori Tahan (21 prediksi benar) dan Sangat Tahan (11 prediksi benar)
- Ada beberapa kesalahan klasifikasi terutama pada kategori Rentan yang terprediksi sebagai Tahan (4 kasus)

Rangkuman Kinerja :

- **Akurasi Total:** 0.71 (71% dari seluruh dataset diprediksi dengan benar).
- **Macro Average:** Menghitung rata-rata metrik untuk semua kelas tanpa mempertimbangkan jumlah contoh di masing-masing kelas. Precision: 0.58, Recall: 0.53, F1-Score: 0.54, menunjukkan kinerja yang kurang merata di antara kelas-kelas.
- **Weighted Average:** Menghitung rata-rata metrik dengan memperhitungkan dukungan (support) dari setiap kelas. Precision, Recall, dan F1-Score semuanya 0.71, menunjukkan bahwa kelas-kelas yang memiliki banyak contoh (seperti "Tahan (Prioritas 5)") memberikan kontribusi besar terhadap hasil.

2. Hasil klasifikasi dari dataset Kecamatan.csv

h. Import Library yang Diperlukan

```
1 import pandas as pd # Untuk manipulasi data menggunakan dataframe
2 import numpy as np  # Untuk operasi numerik
3 from sklearn.model_selection import train_test_split # Membagi data menjadi training dan testing
4 from sklearn.preprocessing import StandardScaler, LabelEncoder # Normalisasi data dan encoding label
5 from sklearn.neighbors import KNeighborsClassifier # Klasifikasi K-Nearest Neighbors (KNN)
6 from sklearn.metrics import classification_report, confusion_matrix # Metrik evaluasi model
7 import matplotlib.pyplot as plt # Untuk visualisasi grafik
8 import seaborn as sns # Untuk heatmap confusion matrix
9
```

Pada bagian awal, beberapa library penting diimpor. pandas digunakan untuk manipulasi dan analisis data. numpy digunakan untuk operasi numerik. sklearn.model_selection menyediakan fungsi untuk membagi dataset menjadi set pelatihan dan pengujian. sklearn.preprocessing digunakan untuk menstandarkan fitur dan mengkode label. sklearn.neighbors mengimpor algoritma KNN. sklearn.metrics digunakan untuk evaluasi model, dan matplotlib serta seaborn digunakan untuk visualisasi data.

i. Fungsi prepare_nasional_data

```

1 def prepare_kabupaten_data(filepath='Kabupaten.csv'):
2     """
3     Prepare data kabupaten untuk klasifikasi
4     """
5     df = pd.read_csv(filepath)
6     features = ['NCPR', 'Kemiskinan (%)', 'Pengeluaran_Pangan(%)', 'Tanpa Listrik (%)',
7               'Tanpa Air Bersih (%)', 'Lama Sekolah Perempuan (tahun)', 'Rasio Tenaga Kesehatan',
8               'Angka Harapan Hidup (tahun)', 'Stunting (%)']
9     X = df[features]
10    y = df['Komposit']
11
12    le = LabelEncoder()
13    y = le.fit_transform(y)
14
15    return X, y, le
16

```

Fungsi `prepare_nasional_data` memiliki peran krusial dalam mempersiapkan data kabupaten untuk klasifikasi. Dalam fungsi ini, dataset diambil dari file CSV yang ditentukan melalui parameter `filepath`, dan dibaca menggunakan `pd.read_csv()`. Setelah membaca dataset, sembilan indikator yang relevan dipilih sebagai fitur untuk model, termasuk NCPR, persentase kemiskinan, dan indikator lainnya yang penting untuk analisis. Untuk menyederhanakan pemahaman, kolom 'Komposit' dalam dataset dikodekan menjadi label yang lebih mudah dipahami melalui proses mapping menggunakan dictionary, di mana setiap angka dikaitkan dengan label prioritas tertentu. Variabel target dihasilkan dari kolom 'Komposit' yang telah dipetakan, dan untuk mempermudah penggunaan dalam model, variabel target ini diubah menjadi format numerik dengan menggunakan `LabelEncoder()`. Fungsi ini kemudian mengembalikan fitur (X), target (y), dan encoder label (le), sehingga data siap digunakan dalam model klasifikasi.

j. Fungsi `train_and_evaluate_knn`

```

1  def train_and_evaluate_knn(X_train, X_test, y_train, y_test, n_neighbors=5):
2      """
3      Train dan evaluasi model KNN untuk data kabupaten
4      """
5      scaler = StandardScaler()
6      X_train_scaled = scaler.fit_transform(X_train)
7      X_test_scaled = scaler.transform(X_test)
8
9      knn = KNeighborsClassifier(n_neighbors=n_neighbors)
10     knn.fit(X_train_scaled, y_train)
11
12     y_pred = knn.predict(X_test_scaled)
13
14     unique_labels = np.unique(y_test)
15     target_names = [f'Kelas {label}' for label in unique_labels]
16
17     print("\nHasil Klasifikasi Data kabupaten:")
18     print("\nClassification Report:")
19     print(classification_report(y_test, y_pred, target_names=target_names))
20
21     cm = confusion_matrix(y_test, y_pred)
22     plt.figure(figsize=(10, 8))
23     sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
24                 xticklabels=target_names,
25                 yticklabels=target_names)
26     plt.title('Confusion Matrix - Data kabupaten')
27     plt.ylabel('True Label')
28     plt.xlabel('Predicted Label')
29     plt.show()
30
31     return knn, scaler
32

```

Selanjutnya, fungsi `train_and_evaluate_knn` bertugas untuk melatih model KNN dan mengevaluasi kinerjanya. Di dalam fungsi ini, fitur dari data pelatihan distandarkan menggunakan `StandardScaler()`, sehingga data memiliki mean 0 dan deviasi standar 1. Proses ini sangat penting untuk algoritma KNN, karena memperhitungkan jarak antara data. Setelah standardisasi, model KNN diinisialisasi dengan jumlah tetangga yang ditentukan (`n_neighbors`) dan dilatih menggunakan data yang telah dinormalisasi. Setelah model dilatih, ia digunakan untuk memprediksi label pada data pengujian, memberikan hasil yang diharapkan dari model yang telah dilatih. Prediksi yang dihasilkan, yang berupa label numerik, kemudian diubah kembali menjadi label prioritas menggunakan daftar `target_names`. Untuk mengevaluasi hasil, laporan klasifikasi ditampilkan, mencakup metrik seperti precision, recall, dan f1-score yang memberikan gambaran komprehensif tentang kinerja model. Selain itu, confusion matrix divisualisasikan sebagai heatmap untuk memberikan gambaran yang lebih jelas tentang kinerja model dalam memprediksi setiap kelas.

k. Fungsi `optimize_k`

```

1 def optimize_k(X_train, X_test, y_train, y_test, k_range=range(1, 31)):
2     """
3     Cari nilai k optimal untuk data kabupaten
4     """
5     scaler = StandardScaler()
6     X_train_scaled = scaler.fit_transform(X_train)
7     X_test_scaled = scaler.transform(X_test)
8
9     accuracies = []
10
11     for k in k_range:
12         knn = KNeighborsClassifier(n_neighbors=k)
13         knn.fit(X_train_scaled, y_train)
14         accuracy = knn.score(X_test_scaled, y_test)
15         accuracies.append(accuracy)
16
17     plt.figure(figsize=(10,6))
18     plt.plot(k_range, accuracies)
19     plt.xlabel('Nilai K')
20     plt.ylabel('Akurasi')
21     plt.title('KNN kabupaten: Akurasi vs Nilai K')
22     plt.grid(True)
23     plt.show()
24
25     best_k = k_range[np.argmax(accuracies)]
26     print(f"\nNilai k terbaik untuk data kabupaten: {best_k} dengan akurasi: {max(accuracies):.4f}")
27
28     return best_k
29

```

Fungsi `optimize_k` memiliki tujuan untuk menemukan nilai `k` optimal untuk model KNN. Dalam fungsi ini, data pelatihan dan pengujian distandarkan sama seperti pada fungsi sebelumnya, memastikan konsistensi dalam skala fitur. Dengan melakukan perhitungan akurasi dalam sebuah loop yang memeriksa rentang nilai `k` yang diberikan, model KNN dilatih untuk setiap nilai `k` yang diuji. Akurasi diukur menggunakan data pengujian dan disimpan dalam list `accuracies`, memberikan gambaran seberapa baik model bekerja untuk setiap `k`. Hasil akurasi untuk setiap nilai `k` kemudian diplot, sehingga hubungan antara nilai `k` dan akurasi model dapat dianalisis dengan mudah. Ini sangat membantu dalam pemilihan nilai `k` yang optimal. Setelah semua akurasi dihitung, nilai `k` dengan akurasi tertinggi dicetak dan dikembalikan, memberikan panduan untuk langkah selanjutnya dalam pelatihan model.

1. Fungsi main

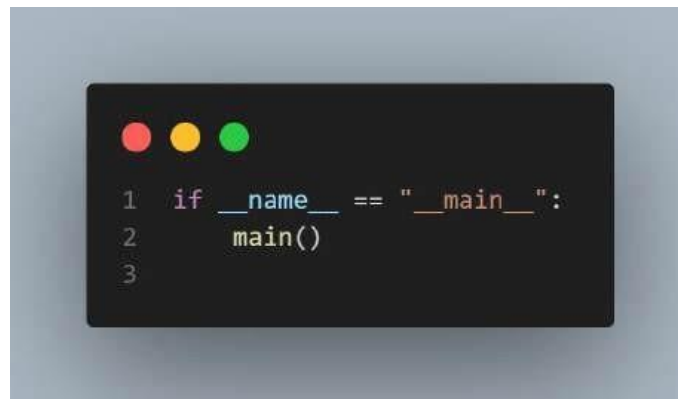
```

1 def main():
2     print("Processing data kabupaten...")
3     X_nas, y_nas, le_nas = prepare_kabupaten_data()
4
5     X_train_nas, X_test_nas, y_train_nas, y_test_nas = train_test_split(
6         X_nas, y_nas, test_size=0.2, random_state=42
7     )
8
9     best_k_nas = optimize_k(X_train_nas, X_test_nas, y_train_nas, y_test_nas)
10
11     print("\nTraining model kabupaten final...")
12     knn_nas, scaler_nas = train_and_evaluate_knn(
13         X_train_nas, X_test_nas, y_train_nas, y_test_nas, n_neighbors=best_k_nas
14     )
15

```

Akhirnya, fungsi main berfungsi sebagai titik awal eksekusi program. Di sini, fungsi `prepare_nasional_data()` dipanggil untuk mendapatkan fitur, target, dan encoder label dari dataset, menjamin bahwa data siap untuk analisis lebih lanjut. Setelah data dipersiapkan, dataset dibagi menjadi data pelatihan dan pengujian menggunakan `train_test_split()`, dengan proporsi 80% untuk pelatihan dan 20% untuk pengujian, yang merupakan praktik umum dalam pembelajaran mesin. Fungsi `optimize_k()` kemudian dipanggil untuk menemukan nilai k terbaik untuk model KNN berdasarkan data pelatihan yang telah dipisahkan, sehingga model dapat disesuaikan dengan optimal. Setelah nilai k ditemukan, model KNN dilatih dan dievaluasi menggunakan fungsi `train_and_evaluate_knn()`, menyelesaikan proses pelatihan dan evaluasi.

m. Eksekusi Program



Terakhir, blok `if __name__ == "__main__":` memastikan bahwa fungsi `main()` hanya akan dipanggil jika skrip dieksekusi secara langsung, bukan saat diimpor sebagai modul, memberikan fleksibilitas dan menjaga struktur program. Secara keseluruhan, kode ini membentuk pipeline lengkap untuk klasifikasi data kabupaten menggunakan algoritma KNN. Dari pemrosesan data, pemilihan fitur, pelatihan model, hingga evaluasi kinerja, setiap tahap diatur secara sistematis untuk menghasilkan model klasifikasi yang efektif. Proses optimasi nilai k memberikan fleksibilitas dalam penyesuaian model dengan dataset tertentu, yang pada gilirannya memungkinkan peningkatan akurasi prediksi.

n. Output


```

...
Nilai k terbaik untuk data kabupaten: 1 dengan akurasi: 0.8544

Training model kabupaten final...

Hasil Klasifikasi Data Nasional:

Classification Report:

```

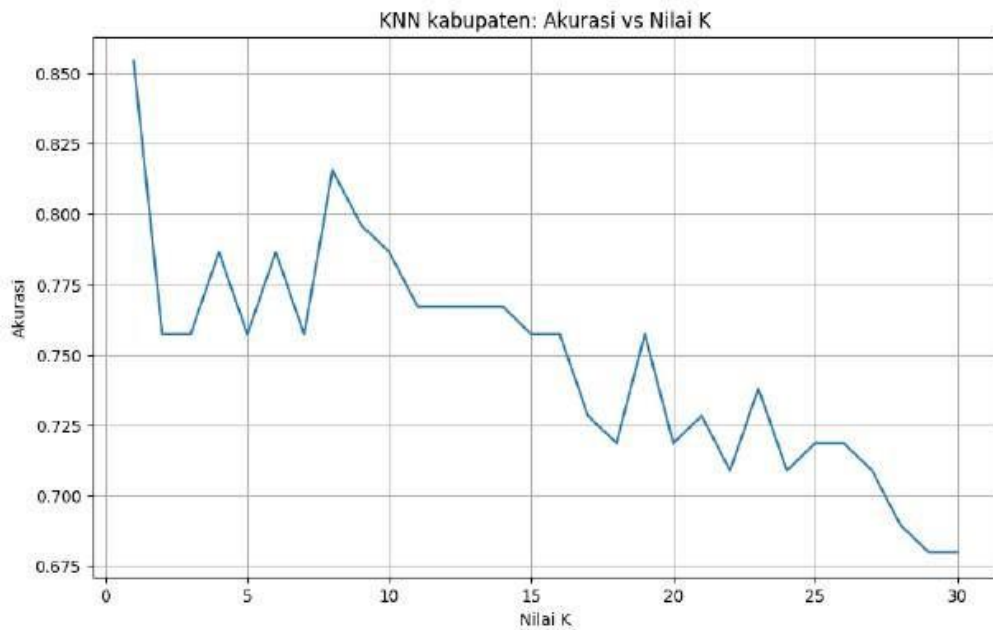
	precision	recall	f1-score	support
Sangat Rentan (Prioritas 1)	0.60	0.38	0.46	8
Rentan (Prioritas 2)	0.60	0.75	0.67	8
Agak Rentan (Prioritas 3)	1.00	0.67	0.80	6
Agak Tahan (Prioritas 4)	1.00	0.75	0.86	4
Tahan (Prioritas 5)	0.77	0.81	0.79	21
Sangat Tahan (Prioritas 6)	0.93	0.98	0.96	56
accuracy			0.85	103
macro avg	0.82	0.72	0.76	103
weighted avg	0.85	0.85	0.85	103

Nilai k Terbaik

Model KNN menemukan bahwa nilai k terbaik untuk data kabupaten adalah 1, dengan akurasi mencapai 85.44%. Ini menunjukkan bahwa model dapat memprediksi label dengan benar sekitar 85.44% dari total data pengujian, yang menunjukkan kinerja model yang sangat baik.

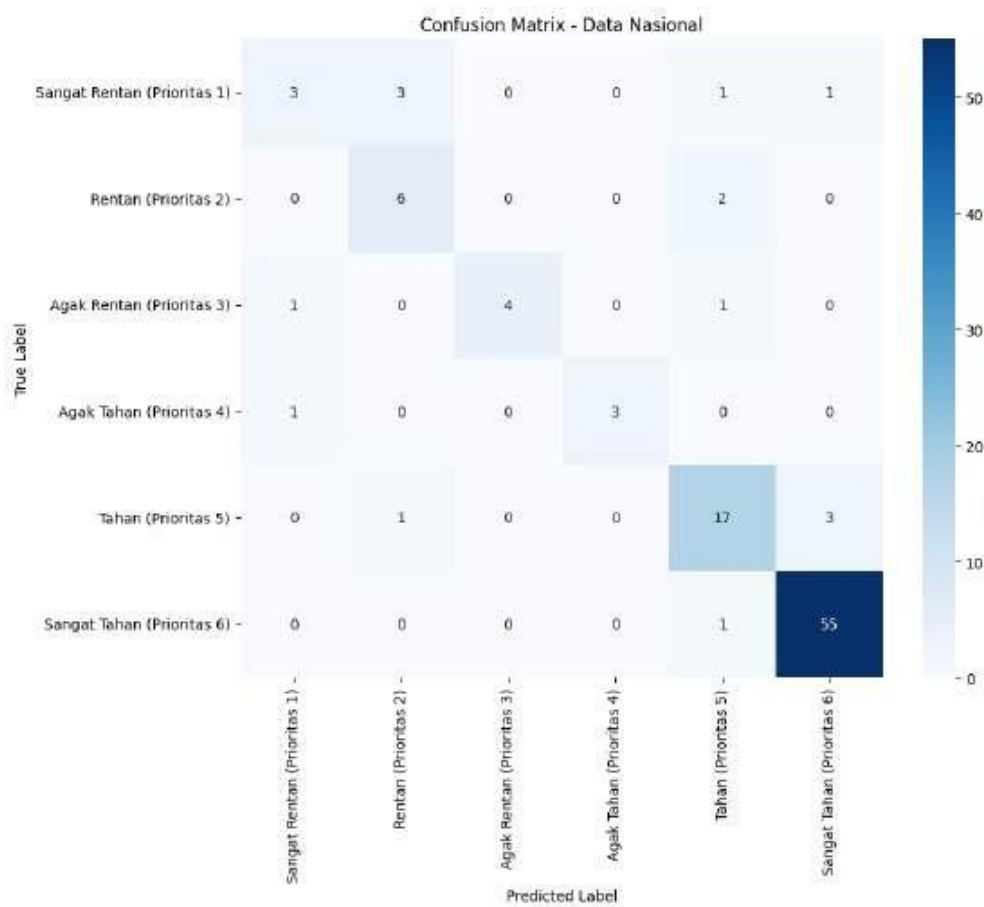
Berikut adalah analisis untuk masing-masing kelas berdasarkan hasil klasifikasi data kabupaten:

- **Sangat Rentan (Prioritas 1):**
 - **Precision:** 0.60 (60% dari prediksi adalah benar).
 - **Recall:** 0.38 (hanya 38% dari kasus sebenarnya terdeteksi).
 - **F1-Score:** 0.46 (menunjukkan keseimbangan yang kurang baik antara precision dan recall).
- **Rentan (Prioritas 2):**
 - **Precision:** 0.60 (60% dari prediksi adalah benar).
 - **Recall:** 0.75 (75% dari kasus sebenarnya terdeteksi).
 - **F1-Score:** 0.67 (menunjukkan keseimbangan yang cukup baik).
- **Agak Rentan (Prioritas 3):**
 - **Precision:** 1.00 (semua prediksi untuk kelas ini benar).
 - **Recall:** 0.67 (67% dari kasus sebenarnya terdeteksi).
 - **F1-Score:** 0.80 (menunjukkan keseimbangan yang baik).
- **Agak Tahan (Prioritas 4):**
 - **Precision:** 1.00 (semua prediksi untuk kelas ini benar).
 - **Recall:** 0.75 (75% dari kasus sebenarnya terdeteksi).
 - **F1-Score:** 0.86 (menunjukkan kinerja yang sangat baik).
- **Tahan (Prioritas 5):**
 - **Precision:** 0.77 (77% dari prediksi adalah benar).
 - **Recall:** 0.81 (81% dari kasus sebenarnya terdeteksi).
 - **F1-Score:** 0.79 (menunjukkan keseimbangan yang baik).
- **Sangat Tahan (Prioritas 6):**
 - **Precision:** 0.93 (93% dari prediksi adalah benar).
 - **Recall:** 0.98 (98% dari kasus sebenarnya terdeteksi).
 - **F1-Score:** 0.96 (kinerja yang sangat baik).



Untuk data kabupaten Grafik Akurasi vs Nilai K :

- Performa lebih baik pada kategori Sangat Tahan (55 prediksi benar)
- Kategori Tahan juga cukup baik (17 prediksi benar)
- Ada beberapa kesalahan klasifikasi di kategori Sangat Rentan dan Rentan



Untuk data kabupaten Confusion Matrix :

- Akurasi tertinggi mencapai 0.85 (85%) pada K kecil
- Tren menurun lebih stabil
- Akurasi berkisar antara 0.67-0.85

Rangkuman Kinerja :

Akurasi total model adalah 0.85, yang berarti 85% dari seluruh dataset diprediksi dengan benar. Macro average menunjukkan bahwa precision adalah 0.82, recall 0.72, dan f1-score 0.76, menunjukkan kinerja yang cukup baik di antara kelas-kelas. Sementara itu, weighted average memberikan nilai precision, recall, dan f1-score masing-masing 0.85, yang menunjukkan bahwa kelas-kelas yang memiliki banyak contoh, seperti "Sangat Tahan (Prioritas 6)", memberikan kontribusi besar terhadap hasil keseluruhan.

BAB 4

PENUTUP

Berdasarkan analisis yang dilakukan, dapat disimpulkan bahwa metode klasifikasi berhasil mengelompokkan wilayah-wilayah di Aceh berdasarkan ketahanan dan kerentanan pangan mereka. Wilayah-wilayah dengan tingkat kerentanan pangan tinggi memerlukan intervensi kebijakan yang lebih cepat dan terarah untuk mengurangi risiko ketahanan pangan yang buruk. Adapun wilayah yang sudah masuk kategori Sangat Tahan dapat terus dipertahankan atau ditingkatkan kondisinya melalui kebijakan yang lebih fokus pada pemanfaatan pangan.

Saran:

1. Pemerintah daerah perlu lebih fokus pada wilayah yang masuk kategori Prioritas 1 untuk mengurangi kerentanan pangan.
2. Pemanfaatan data dari FSVA perlu terus dikembangkan agar dapat mendukung kebijakan berbasis data yang lebih tepat sasaran.

Dengan demikian, hasil dari klasifikasi ini memberikan wawasan yang penting bagi pembuat kebijakan untuk meningkatkan kesejahteraan masyarakat melalui ketahanan pangan yang lebih baik di seluruh Provinsi Aceh.