# Praktikum & Tugas 10: Algoritma K-Nearest Neighbours (KNN)

**Rizky Hilmiawan Anggoro[1] - 01102221401**

[1] Teknik Informatika, STT Terpadu Nurul Fikri, Depok

*E-mail: rizk22140ti@student.nurulfikri.ac.id

**Abstract.** Laporan praktikum ini membahas implementasi algoritma K-Nearest Neighbours (KNN) untuk klasifikasi data melalui tiga latihan. Latihan pertama menerapkan KNN untuk mengklasifikasikan data dengan menentukan nilai k optimal menggunakan validasi silang. Latihan kedua berfokus pada evaluasi model klasifikasi melalui confusion matrix dan metrik akurasi, presisi, serta recall dalam memprediksi kelulusan mahasiswa. Latihan ketiga mengaplikasikan KNN untuk memprediksi tipe cuaca dengan tahapan preprocessing data, scaling fitur, pembagian dataset, dan evaluasi model. Hasil implementasi menunjukkan bahwa KNN mampu melakukan klasifikasi dengan efektif setelah melalui tahapan preprocessing yang tepat, termasuk handling data kategorikal dan normalisasi fitur, serta pemilihan parameter k yang optimal.

## LATIHAN 1

### 1. Persiapan Data dan Library

Project ini menggunakan library Pandas untuk mengelola data dan scikit-learn untuk implementasi KNN.

```python
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score
import numpy as np

# Data Set (raining)
data = {
    'Suhu_Udara_C': [10, 25, 15, 20, 18, 20, 22, 24],
    'Kecepatan_Angin_km/jam': [0, 0, 5, 3, 7, 10, 5, 6],
    'Klasifikasi_Marry': ['Dingin', 'Panas', 'Dingin', 'Panas', 'Dingin', 'Dingin', 'Panas', 'Panas']
}
df = pd.DataFrame(data)

# Data Fitur (X) dan Target (y)
X = df[['Suhu_Udara_C', 'Kecepatan_Angin_km/jam']]
y = df['Klasifikasi_Marry']

# Data Test yang akan diprediksi
data_test = pd.DataFrame({
    'Suhu_Udara_C': [16],
    'Kecepatan_Angin_km/jam': [3]
})
```

## 2. Klasifikasi Data Test

(Asumsi k=3)Untuk menjawab pertanyaan pertama, disini akan memprediksi persepsi Marry dengan asumsi awal k=3 (nilai ganjil yang umum digunakan).A. Inisialisasi dan Pelatihan Model (k=3

```
k_initial = 3
knn_model = KNeighborsClassifier(n_neighbors=k_initial)
knn_model.fit(X, y)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

```
prediksi_marry = knn_model.predict(data_test)
print(f"Untuk Suhu 16°C dan Angin 3 km/jam, prediksi persepsi Marry (dengan k={k_initial}) adalah: **{prediksi_marry[0]}**")
```

```
Untuk Suhu 16°C dan Angin 3 km/jam, prediksi persepsi Marry (dengan k=3) adalah: **Dingin**
```

## 3. Menentukan Nilai k yang Tepat

Kita akan menguji berbagai nilai k (misalnya dari k=1 hingga k=7) untuk mencari nilai yang memberikan akurasi terbaik menggunakan Cross-Validation (Validasi Silang).

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score

# X = df[['Suhu_Udara_C', 'Kecepatan_Angin_km/jam']] (8 baris data)
# y = df['Klasifikasi_Marry'] (4 'Dingin', 4 'Panas')

# Daftar nilai k yang akan diuji
k_values = range(1, len(X) // 2 + 1, 2)
if not k_values:
    # Kasus data sangat sedikit
    k_values = [1]

accuracies = []

# Jumlah lipatan (cv) yang aman adalah 4, karena jumlah anggota kelas terkecil adalah 4
SAFE_CV = 4

# Uji setiap nilai k menggunakan Cross-Validation
for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    # GANTI cv=len(X) menjadi cv=4
    scores = cross_val_score(knn, X, y, cv=SAFE_CV, scoring='accuracy')
    accuracies.append(scores.mean())

# Cari nilai k dengan akurasi tertinggi
best_k_index = np.argmax(accuracies)
best_k = k_values[best_k_index]
best_accuracy = accuracies[best_k_index]

print("\n## Hasil Pengujian Akurasi Berdasarkan Nilai k")
for k, acc in zip(k_values, accuracies):
    print(f"k = {k}: Akurasi rata-rata = {acc:.2f}")

print(f"\nNilai **k** yang tepat berdasarkan akurasi tertinggi adalah **k = {best_k}**")
print(f"Akurasi tertinggi yang dicapai: {best_accuracy:.2f}")
```

```
## Hasil Pengujian Akurasi Berdasarkan Nilai k
k = 1: Akurasi rata-rata = 1.00
k = 3: Akurasi rata-rata = 0.88

Nilai **k** yang tepat berdasarkan akurasi tertinggi adalah **k = 1**
Akurasi tertinggi yang dicapai: 1.00
```

**4. Klasifikasi Akhir dengan** k **Terbaik**

Setelah menemukan k yang optimal, ulangi prediksi data test dengan nilai k tersebut.

```
# Model KNN dengan k terbaik
knn_best = KNeighborsClassifier(n_neighbors=best_k)
knn_best.fit(X, y)

# Prediksi menggunakan k terbaik
prediksi_final = knn_best.predict(data_test)

print("\n## Prediksi Akhir")
print(f"Dengan nilai k optimal **k = {best_k}**:")
print(f"Untuk Suhu 16°C dan Angin 3 km/jam, prediksi persepsi Marry adalah: **{prediksi_final[0]}**")
```

```
## Prediksi Akhir
Dengan nilai k optimal **k = 1**:
Untuk Suhu 16°C dan Angin 3 km/jam, prediksi persepsi Marry adalah: **Dingin**
```

## LATIHAN 2

### 1. Persiapan Data dan Library

Kode di bawah merupakan implementasi dalam Python untuk melakukan analisis evaluasi model klasifikasi menggunakan data hasil prediksi kelulusan mahasiswa. Program dimulai dengan mengimpor library yang diperlukan seperti pandas untuk manipulasi data, sklearn untuk metrik evaluasi, serta matplotlib dan seaborn untuk visualisasi. Data disiapkan dalam bentuk list yang berisi NIM mahasiswa, hasil sebenarnya, dan hasil prediksi kelulusan, kemudian dikonversi ke dalam DataFrame pandas untuk memudahkan pengolahan dan analisis lebih lanjut. Output awal dari kode ini adalah menampilkan tabel data yang berisi ketiga informasi tersebut, yang akan menjadi dasar untuk penghitungan metrik evaluasi seperti confusion matrix, akurasi, presisi, dan recall pada tahap selanjutnya.

```
import pandas as pd
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Persiapan Data

# Data dari tabel (Nilai sebenarnya dan Nilai Prediksi)
nim = ['TI001', 'TI002', 'TI003', 'TI004', 'TI005', 'TI006', 'TI007', 'TI008', 'TI009', 'TI010']
hasil_sebenarnya = ['Lulus', 'Lulus', 'Lulus', 'Lulus', 'Lulus', 'Tidak Lulus', 'Tidak Lulus', 'Tidak Lulus', 'Tidak Lulus', 'Tidak Lulus']
hasil_prediksi = ['Lulus', 'Lulus', 'Lulus', 'Tidak Lulus', 'Tidak Lulus', 'Lulus', 'Tidak Lulus', 'Tidak Lulus', 'Tidak Lulus', 'Tidak Lulus']

# Membuat DataFrame untuk memudahkan visualisasi data
data = pd.DataFrame({
    'NIM': nim,
    'Hasil Sebenarnya': hasil_sebenarnya,
    'Hasil Prediksi': hasil_prediksi
})

print("Tabel Data:")
print(data)
```

```
Tabel Data:
    NIM Hasil Sebenarnya Hasil Prediksi
0  TI001            Lulus          Lulus
1  TI002            Lulus          Lulus
2  TI003            Lulus          Lulus
3  TI004            Lulus    Tidak Lulus
4  TI005            Lulus    Tidak Lulus
5  TI006      Tidak Lulus          Lulus
6  TI007      Tidak Lulus    Tidak Lulus
7  TI008      Tidak Lulus    Tidak Lulus
8  TI009      Tidak Lulus    Tidak Lulus
9  TI010      Tidak Lulus    Tidak Lulus
```

### 2. Pembuatan Confusion Matrix

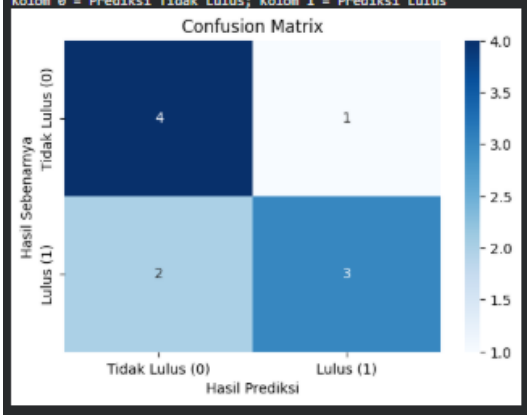
Kode di bawah merupakan implementasi pembuatan dan visualisasi confusion matrix untuk mengevaluasi performa model klasifikasi dalam memprediksi kelulusan mahasiswa. Pertama, data kategorikal ('Lulus'/'Tidak Lulus') dikonversi menjadi format numerik (1 untuk 'Lulus' dan

0 untuk 'Tidak Lulus') agar dapat diproses oleh fungsi confusion_matrix dari sklearn. confusion matrix yang dihasilkan kemudian divisualisasikan menggunakan heatmap dari seaborn dengan anotasi nilai dan label yang jelas, dimana sumbu Y merepresentasikan kondisi sebenarnya dan sumbu X merepresentasikan hasil prediksi.

Dari confusion matrix tersebut dapat diidentifikasi empat komponen evaluasi: True Negative (TN) sebanyak 4 kasus dimana mahasiswa tidak lulus diprediksi dengan benar, False Positive (FP) sebanyak 1 kasus dimana mahasiswa tidak lulus tapi diprediksi lulus, False Negative (FN) sebanyak 2 kasus dimana mahasiswa lulus tapi diprediksi tidak lulus, dan True Positive (TP) sebanyak 3 kasus dimana mahasiswa lulus diprediksi dengan benar. Visualisasi heatmap ini memudahkan dalam memahami distribusi hasil prediksi model secara keseluruhan.

```python
# 2. Pembuatan Confusion Matrix

y_true = [1 if x == 'Lulus' else 0 for x in hasil_sebenarnya]
y_pred = [1 if x == 'Lulus' else 0 for x in hasil_prediksi]

# Menghitung Confusion Matrix
# Label: 0 (Tidak Lulus), 1 (Lulus)
cm = confusion_matrix(y_true, y_pred)

print("\n--- Confusion Matrix ---")
print(cm)
print("Baris 0 = Sebenarnya Tidak Lulus; Baris 1 = Sebenarnya Lulus")
print("Kolom 0 = Prediksi Tidak Lulus; Kolom 1 = Prediksi Lulus")

# Visualisasi Confusion Matrix
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Tidak Lulus (0)', 'Lulus (1)'],
            yticklabels=['Tidak Lulus (0)', 'Lulus (1)'])
plt.title('Confusion Matrix')
plt.xlabel('Hasil Prediksi')
plt.ylabel('Hasil Sebenarnya')
plt.show()

# Berdasarkan CM:
# TN (True Negative): cm[0, 0] = 4 (Benar prediksi Tidak Lulus)
# FP (False Positive): cm[0, 1] = 1 (Salah prediksi Lulus, seharusnya Tidak Lulus)
# FN (False Negative): cm[1, 0] = 2 (Salah prediksi Tidak Lulus, seharusnya Lulus)
# TP (True Positive): cm[1, 1] = 3 (Benar prediksi Lulus)
```

```
--- Confusion Matrix ---
[[4 1]
 [2 3]]
Baris 0 = Sebenarnya Tidak Lulus; Baris 1 = Sebenarnya Lulus
Kolom 0 = Prediksi Tidak Lulus; Kolom 1 = Prediksi Lulus
```




### 3. Perhitungan Metrik

Kode dibawah menghitung tiga metrik evaluasi model klasifikasi. **Accuracy** (70%) mengukur persentase prediksi benar secara keseluruhan, yaitu 7 dari 10 prediksi benar. **Precision** (75%) menunjukkan dari semua prediksi "Lulus", 3 dari 4 prediksi tersebut benar. **Recall** (60%) mengukur kemampuan model mendeteksi kasus "Lulus" yang sebenarnya, yaitu 3 dari 5

mahasiswa yang benar-benar lulus berhasil diprediksi dengan benar. Perhitungan dilakukan secara manual menggunakan rumus dan diverifikasi dengan fungsi sklearn.

```python
# 3. Perhitungan Metrik (Langkah 2 dari soal)

# Total data
N = len(y_true)
TP = cm[1, 1]
TN = cm[0, 0]
FP = cm[0, 1]
FN = cm[1, 0]

# 3.1. Accuracy
# Accuracy = (TP + TN) / N
accuracy_manual = (TP + TN) / N
accuracy_sklearn = accuracy_score(y_true, y_pred)

# 3.2. Precision
# Precision = TP / (TP + FP)
precision_manual = TP / (TP + FP)
precision_sklearn = precision_score(y_true, y_pred)

# 3.3. Recall
# Recall = TP / (TP + FN)
recall_manual = TP / (TP + FN)
recall_sklearn = recall_score(y_true, y_pred)

print("\n--- Hasil Perhitungan Metrik (Persentase) ---")

print(f"1. Accuracy: {accuracy_manual:.4f} ({accuracy_manual*100:.2f}%)")
print(f"   (Skor Benar/Total: {TP + TN}/{N})")

print(f"2. Precision: {precision_manual:.4f} ({precision_manual*100:.2f}%)")
print(f"   (Dari semua prediksi 'Lulus', yang benar-benar Lulus: {TP}/{TP + FP})")

print(f"3. Recall: {recall_manual:.4f} ({recall_manual*100:.2f}%)")
print(f"   (Dari semua yang benar-benar 'Lulus', yang berhasil diprediksi: {TP}/{TP + FN})")

# Perbandingan dengan sklearn untuk verifikasi
# print("\nVerifikasi Sklearn:")
# print(f"Accuracy (Sklearn): {accuracy_sklearn:.4f}")
# print(f"Precision (Sklearn): {precision_sklearn:.4f}")
# print(f"Recall (Sklearn): {recall_sklearn:.4f}")
```

```
--- Hasil Perhitungan Metrik (Persentase) ---
1. Accuracy: 0.7000 (70.00%)
   (Skor Benar/Total: 7/10)
2. Precision: 0.7500 (75.00%)
   (Dari semua prediksi 'Lulus', yang benar-benar Lulus: 3/4)
3. Recall: 0.6000 (60.00%)
   (Dari semua yang benar-benar 'Lulus', yang berhasil diprediksi: 3/5)
```

**LATIHAN 3**

## 1. Persiapan Data

Kode di bawah mempersiapkan dataset untuk klasifikasi cuaca menggunakan algoritma K-Nearest Neighbors (KNN). Data disimpan dalam dictionary Python yang berisi 10 fitur prediktor seperti temperatur, kelembapan, kecepatan angin, dan lokasi, dengan target prediksi 'Weather Type' (Rainy, Cloudy, Sunny, Snowy). Dictionary tersebut kemudian dikonversi menjadi DataFrame pandas untuk memudahkan proses analisis dan pemodelan data lebih lanjut.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
import numpy as np

# 1. Persiapan Data
print("1. Persiapan Data")
# Data yang dimasukkan sesuai dengan tabel yang disediakan
data = {
    'Temperature': [14.0, 39.0, 30.0, 38.0, 27.0, 32.0, -2.0, 3.0, 3.0, 28.0],
    'Humidity': [73, 96, 64, 83, 74, 55, 97, 85, 83, 74],
    'Wind Speed': [9.5, 8.5, 7.0, 1.5, 17.0, 3.5, 8.0, 6.0, 6.0, 8.5],
    'Precipitation (%)': [82.0, 71.0, 16.0, 82.0, 66.0, 26.0, 86.0, 96.0, 66.0, 107.0],
    'Cloud Cover': ['partly cloudy', 'partly cloudy', 'clear', 'clear', 'overcast', 'overcast', 'overcast', 'partly cloudy', 'overcast', 'clear'],
    'Atmospheric Pressure': [1010.82, 1011.43, 1018.72, 1026.25, 990.67, 1010.03, 990.87, 984.46, 999.44, 1012.13],
    'UV Index': [2, 7, 5, 7, 1, 2, 1, 1, 0, 8],
    'Season': ['Winter', 'Spring', 'Spring', 'Spring', 'Winter', 'Summer', 'Winter', 'Winter', 'Winter', 'Winter'],
    'Visibility (km)': [3.5, 10.0, 5.5, 1.0, 2.5, 5.0, 4.0, 3.5, 1.0, 7.5],
    'Location': ['inland', 'inland', 'mountain', 'coastal', 'mountain', 'inland', 'inland', 'inland', 'mountain', 'coastal'],
    'Weather Type': ['Rainy', 'Cloudy', 'Sunny', 'Sunny', 'Rainy', 'Cloudy', 'Snowy', 'Snowy', 'Snowy', 'Sunny']
}
df = pd.DataFrame(data)
print("Data asli:")
print(df.head())
```

```
1. Persiapan Data
Data asli:
   Temperature  Humidity  Wind Speed  Precipitation (%)    Cloud Cover  \
0         14.0        73         9.5              82.0  partly cloudy
1         39.0        96         8.5              71.0  partly cloudy
2         30.0        64         7.0              16.0          clear
3         38.0        83         1.5              82.0          clear
4         27.0        74        17.0              66.0       overcast

   Atmospheric Pressure  UV Index  Season  Visibility (km)  Location  \
0               1010.82         2  Winter              3.5    inland
1               1011.43         7  Spring             10.0    inland
2               1018.72         5  Spring              5.5  mountain
3               1026.25         7  Spring              1.0   coastal
4                990.67         1  Winter              2.5  mountain

  Weather Type
0        Rainy
1       Cloudy
2        Sunny
3        Sunny
4        Rainy
```

## 2. Pre-processing Data

Kode di bawah melakukan preprocessing data untuk mengubah variabel kategorikal menjadi numerik. Kolom target 'Weather Type' diencode menggunakan LabelEncoder menjadi nilai numerik. Fitur kategorikal lainnya ('Cloud Cover', 'Season', 'Location') dikonversi menggunakan one-hot encoding untuk menghindari urutan arbitrer. Setelah encoding, dataset dipisahkan menjadi fitur (X) dan target (y), dengan memastikan semua fitur berupa numerik untuk kompatibilitas dengan algoritma KNN.

```python
# 2. Pre-processing Data (Mengubah Kategorikal ke Numerik)
print("\n2. Pre-processing Data")

# Menggunakan Label Encoding untuk kolom target (Weather Type)
le_target = LabelEncoder()
df['Weather Type_encoded'] = le_target.fit_transform(df['Weather Type'])
# Label mapping: 0=Cloudy, 1=Rainy, 2=Snowy, 3=Sunny (Urutan tergantung abjad)

# Menggunakan One-Hot Encoding untuk fitur kategorikal lainnya
categorical_features = ['Cloud Cover', 'Season', 'Location']
df_encoded = pd.get_dummies(df, columns=categorical_features, drop_first=True)

# Menentukan Fitur (X) dan Target (Y)
# Drop kolom string asli dan kolom target asli
X = df_encoded.drop(columns=['Weather Type', 'Weather Type_encoded'])
y = df_encoded['Weather Type_encoded']

# Memastikan semua fitur X adalah numerik (sisa kolom string harus sudah di-drop)
X = X.select_dtypes(include=np.number)

print("Fitur setelah encoding dan pembersihan:")
print(X.head())
```

```
2. Pre-processing Data
Fitur setelah encoding dan pembersihan:
   Temperature  Humidity  Wind Speed  Precipitation (%)  Atmospheric Pressure  \
0         14.0        73         9.5               82.0               1010.82
1         39.0        96         8.5               71.0               1011.43
2         30.0        64         7.0               16.0               1018.72
3         38.0        83         1.5               82.0               1026.25
4         27.0        74        17.0               66.0                990.67

   UV Index  Visibility (km)
0         2              3.5
1         7             10.0
2         5              5.5
3         7              1.0
4         1              2.5
```

### 3. Scaling Fitur

Kode di bawah melakukan scaling pada fitur numerik menggunakan StandardScaler. Proses ini mentransformasi data sehingga memiliki mean = 0 dan standar deviasi = 1, yang sangat penting untuk algoritma KNN karena sensitif terhadap perbedaan skala antar fitur. Hasil scaling disimpan dalam DataFrame baru yang mempertahankan nama kolom asli untuk memudahkan interpretasi.

```
# 3. Scaling Fitur (Penting untuk KNN)
print("\n3. Scaling Fitur Numerik")
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)

print("Fitur setelah scaling (Standar Deviasi = 1, Mean = 0):")
print(X_scaled_df.head())
```

```
3. Scaling Fitur Numerik
Fitur setelah scaling (Standar Deviasi = 1, Mean = 0):
   Temperature  Humidity  Wind Speed  Precipitation (%)  Atmospheric Pressure  \
0    -0.493846 -0.431738    0.498161           0.446267              0.415986
1     1.220898  1.407145    0.242694           0.043895              0.463523
2     0.603590 -1.151300   -0.140507          -1.967965              1.031627
3     1.152308  0.367776   -1.545577           0.446267              1.618433
4     0.397821 -0.351786    2.414165          -0.139001             -1.154287

   UV Index  Visibility (km)
0 -0.487713        -0.318943
1  1.254119         2.120034
2  0.557386         0.431511
3  1.254119        -1.257011
4 -0.836080        -0.694170
```

### 4. Membagi Data Training dan Testing

Kode di bawah membagi dataset menjadi data training (60%) dan testing (40%) menggunakan train_test_split. Parameter stratify=y memastikan distribusi kelas target proporsional di kedua subset. Dengan random_state=42, pembagian data konsisten setiap kali dijalankan. Hasilnya adalah 6 baris data training dan 4 baris data testing untuk proses modeling dan evaluasi.

```
# 4. Membagi Data Training dan Testing
print("\n4. Membagi Data (Training: 60%, Testing: 40%)") # Adjusted for stratification
# Karena data sangat kecil (10 baris), split ini hanya untuk demonstrasi.
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled_df, y, test_size=0.4, random_state=42, stratify=y # Changed test_size to 0.4
)

print(f"Jumlah data Training: {len(X_train)} baris")
print(f"Jumlah data Testing: {len(X_test)} baris")
```

```
4. Membagi Data (Training: 60%, Testing: 40%)
Jumlah data Training: 6 baris
Jumlah data Testing: 4 baris
```

### 5. Implementasi Model KNN

Kode di bawah mengimplementasikan model KNN dengan K=3. Model dilatih menggunakan data training (X_train, y_train) kemudian melakukan prediksi pada data testing (X_test). Hasil prediksi disimpan dalam y_pred untuk evaluasi performa model selanjutnya.

```python
# 5. Implementasi Model KNN
print("\n5. Implementasi Model KNN (K=3)")
# Pilih nilai K (jumlah tetangga)
k_value = 3
knn_model = KNeighborsClassifier(n_neighbors=k_value)

# Melatih model
knn_model.fit(X_train, y_train)

# Melakukan Prediksi
y_pred = knn_model.predict(X_test)
```

```
5. Implementasi Model KNN (K=3)
```

### 6. Evaluasi Model

Kode ini mengevaluasi performa model KNN dengan menghitung akurasi dan classification report. Akurasi menunjukkan persentase prediksi benar secara keseluruhan, sementara classification report memberikan metrik detail per kelas (precision, recall, f1-score). Parameter zero_division=0 menangani kasus ketika tidak ada prediksi untuk kelas tertentu.

```python
# 6. Evaluasi Model
print("\n6. Evaluasi Model")

# Menghitung Akurasi
accuracy = accuracy_score(y_test, y_pred)
print(f"Akurasi Model (K={k_value}): {accuracy:.4f}")

# Classification Report
# Catatan: Karena data sangat kecil dan stratify mungkin gagal menyeimbangkan
# kelas di data test (hanya 3 baris), hasil metrik mungkin tidak informatif.
print("\nClassification Report (Metrik per Kelas):")
print(classification_report(y_test, y_pred, target_names=le_target.classes_, zero_division=0))
```

```
6. Evaluasi Model
Akurasi Model (K=3): 0.5000

Classification Report (Metrik per Kelas):
              precision    recall  f1-score   support

      Cloudy       0.00      0.00      0.00         1
       Rainy       0.00      0.00      0.00         1
       Snowy       0.50      1.00      0.67         1
       Sunny       0.50      1.00      0.67         1

    accuracy                           0.50         4
   macro avg       0.25      0.50      0.33         4
weighted avg       0.25      0.50      0.33         4
```