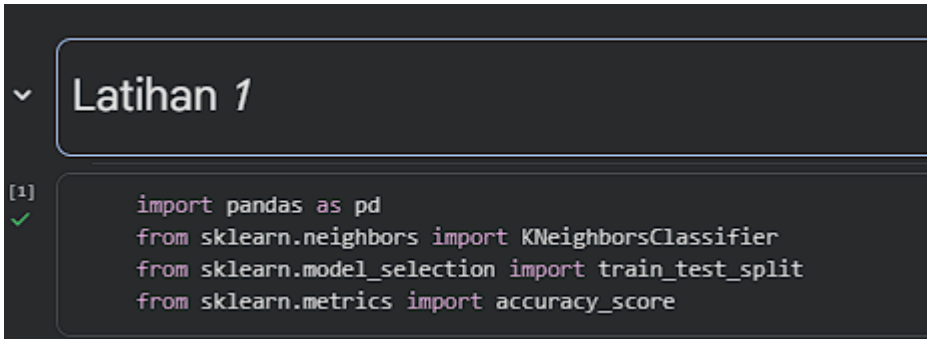
Nama: Noer Muhammad Ayub

NIM: 0110222142

Praktikum: Praktikum K-Nearest Neighbors (KNN)

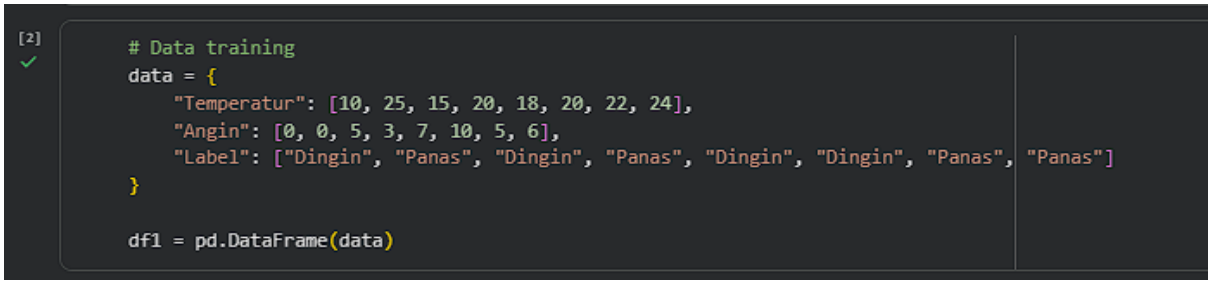
Import Library



Bagian import library ini digunakan untuk mempersiapkan seluruh komponen yang dibutuhkan dalam proses pembuatan model machine learning k-nearest neighbors.

Pandas untuk mengolah data (read_csv, rename, describe, dll), KNN Classifier untuk model, train_test_split untuk membagi data, accuracy_score untuk evaluasi.

Data training



Kode tersebut berfungsi untuk membuat dataset secara manual, dengan 3 kolom yaitu: Temperatur, Angin, dan Label. Dengan isi kolom yang ada pada kurung kotak. pd.DataFrame() mengubahnya menjadi struktur tabel yang lebih rapi sehingga bisa diproses oleh algoritma machine learning

Fitur dan label

Kode tersebut berfungsi untuk memisahkan data fitur dan data label. Kolom Temperatur dan Angin menjadi X karena keduanya digunakan sebagai input model. Sementara y berisi kolom Label, yaitu kategori cuaca yang ingin diprediksi: “Dingin” atau “Panas”.

Split Data

```
[4]
# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42
)
```

Kode ini dipakai untuk membagi data menjadi data latih dan data uji. Variabel X_train dan y_train digunakan untuk melatih model, sedangkan X_test dan y_test dipakai untuk menguji hasil prediksi model. Nilai test_size=0.25 berarti 25% data menjadi data uji, dan random_state=42 membuat pembagian datanya tetap konsisten setiap kali dijalankan.

Mencari K terbaik

```
[5]
0s
# Mencari k terbaik
best_k = None
best_acc = 0

for k in range(1, len(X_train) + 1):
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    acc = accuracy_score(y_test, pred)
    if acc > best_acc:
        best_acc = acc
        best_k = k

print("K terbaik:", best_k)
print("Akurasi terbaik:", best_acc)

...
K terbaik: 1
Akurasi terbaik: 1.0
```

Kode ini digunakan untuk mencari nilai k terbaik pada algoritma KNN. Prosesnya dilakukan dengan mencoba berbagai nilai k, mulai dari 1 hingga jumlah data latih. Setiap k diuji dengan melatih model, memprediksi data uji, lalu menghitung akurasinya. Jika akurasi lebih baik dari sebelumnya, nilai k tersebut disimpan sebagai yang terbaik. Hasil akhirnya menampilkan k terbaik dan akurasi tertinggi yang diperoleh.

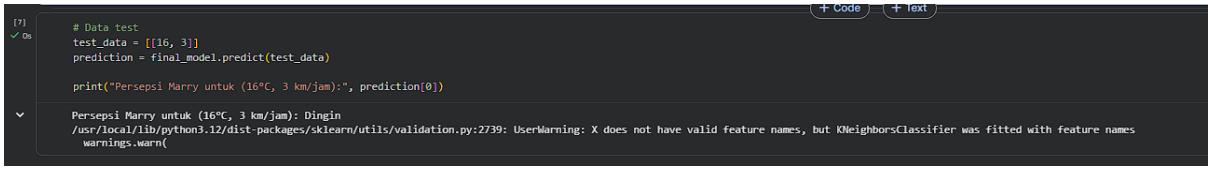
Training final model dengan k terbaik

```
[6]
0s
# Training final model dengan k terbaik
final_model = KNeighborsClassifier(n_neighbors=best_k)
final_model.fit(X, y)

KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)
```

Kode ini digunakan untuk melatih model akhir menggunakan nilai k terbaik yang sudah ditemukan sebelumnya. Model KNN dibuat dengan n_neighbors=best_k, lalu dilatih

menggunakan seluruh data X dan y. Dengan cara ini, model akhir memanfaatkan semua data yang tersedia untuk menghasilkan performa prediksi yang lebih optimal.

Data test

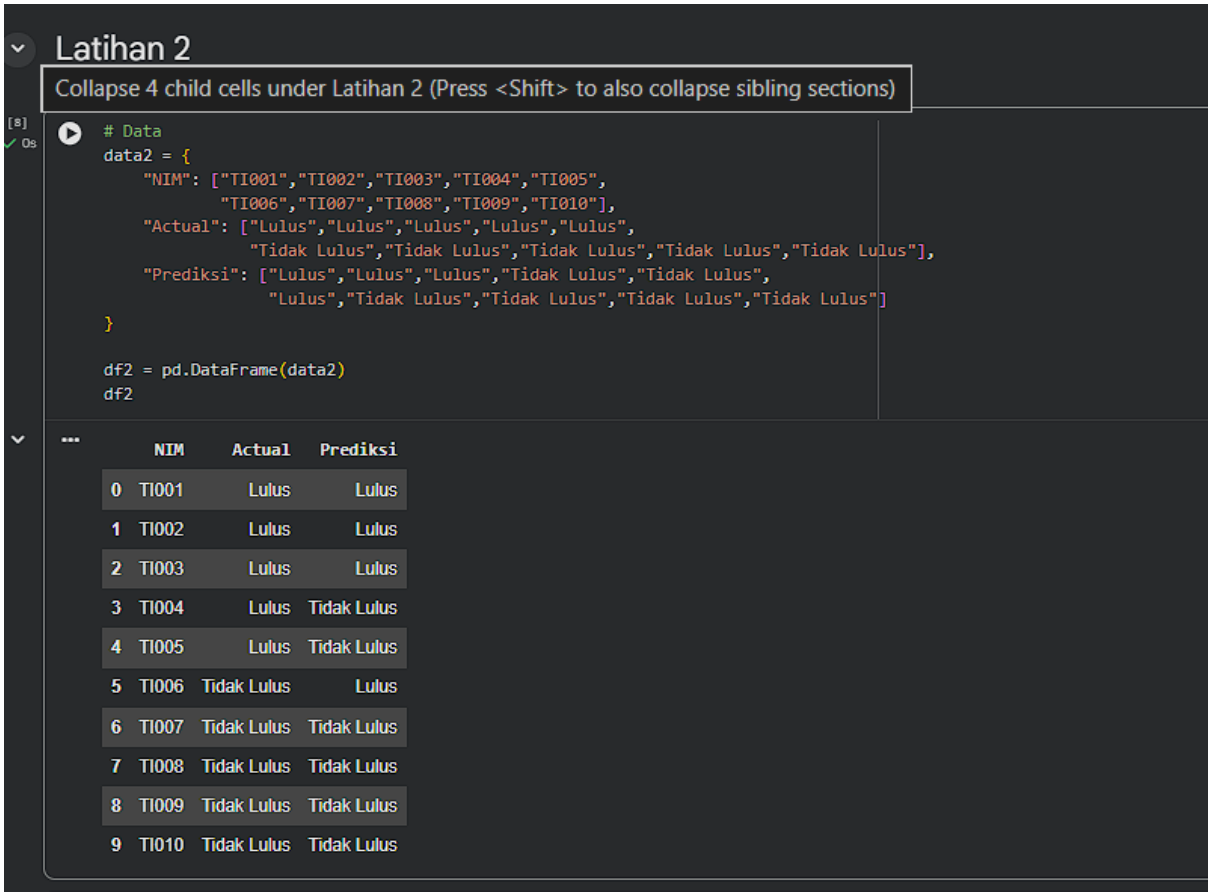


```
# Data test
test_data = [[16, 3]]
prediction = final_model.predict(test_data)

print("Persepsi Marry untuk (16°C, 3 km/jam):", prediction[0])
```

```
Persepsi Marry untuk (16°C, 3 km/jam): Dingin
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but KNeighborsClassifier was fitted with feature names
  warnings.warn(
```

Kode ini digunakan untuk melakukan prediksi terhadap data baru. Nilai 16 sebagai temperatur dan 3 sebagai angin dimasukkan sebagai *test_data*. Model yang sudah dilatih kemudian memprediksi kategori cuacanya melalui final_model.predict(). Hasilnya menunjukkan bagaimana Persepsi Marry menilai kondisi cuaca berdasarkan data tersebut.

Latihan 2



Latihan 2

Collapse 4 child cells under Latihan 2 (Press <Shift> to also collapse sibling sections)

```
# Data
data2 = {
    "NIM": ["TI001","TI002","TI003","TI004","TI005",
            "TI006","TI007","TI008","TI009","TI010"],
    "Actual": ["Lulus","Lulus","Lulus","Lulus","Lulus",
               "Tidak Lulus","Tidak Lulus","Tidak Lulus","Tidak Lulus","Tidak Lulus"],
    "Prediksi": ["Lulus","Lulus","Lulus","Tidak Lulus","Tidak Lulus",
                 "Lulus","Tidak Lulus","Tidak Lulus","Tidak Lulus","Tidak Lulus"]
}

df2 = pd.DataFrame(data2)
df2
```

	NIM	Actual	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

Kode ini digunakan untuk membuat dataset kedua yang berisi tiga kolom: NIM, Actual, dan Prediksi. Kolom Actual menunjukkan status kelulusan sebenarnya, sementara kolom Prediksi berisi hasil perkiraan model. Seluruh data dimasukkan ke dalam dictionary lalu diubah menjadi bentuk tabel menggunakan pd.DataFrame(data2), sehingga lebih mudah dianalisis untuk evaluasi akurasi model.

Encode Label

```
[9]
✓ 0s
# Encode Label
mapping = {"Lulus": 1, "Tidak Lulus": 0}

df2["Actual_enc"] = df2["Actual"].map(mapping)
df2["Prediksi_enc"] = df2["Prediksi"].map(mapping)

df2
```

	NIM	Actual	Prediksi	Actual_enc	Prediksi_enc
0	TI001	Lulus	Lulus	1	1
1	TI002	Lulus	Lulus	1	1
2	TI003	Lulus	Lulus	1	1
3	TI004	Lulus	Tidak Lulus	1	0
4	TI005	Lulus	Tidak Lulus	1	0
5	TI006	Tidak Lulus	Lulus	0	1
6	TI007	Tidak Lulus	Tidak Lulus	0	0
7	TI008	Tidak Lulus	Tidak Lulus	0	0
8	TI009	Tidak Lulus	Tidak Lulus	0	0
9	TI010	Tidak Lulus	Tidak Lulus	0	0

Kode ini digunakan untuk mengubah nilai kategori menjadi bentuk angka agar lebih mudah dihitung. Mapping dibuat dengan aturan: Lulus = 1 dan Tidak Lulus = 0. Kolom *Actual* dan *Prediksi* kemudian dikonversi ke angka menggunakan. map(mapping) dan disimpan sebagai Actual_enc dan Prediksi_enc. Hasil encode ini nantinya digunakan untuk evaluasi seperti menghitung akurasi atau membuat confusion matrix.

Menghitung hasil evaluasi model

```
[10]
✓ 0s
from sklearn.metrics import confusion_matrix, accuracy_score,precision_score, recall_score

y_true = df2["Actual_enc"]
y_pred = df2["Prediksi_enc"]

cm = confusion_matrix(y_true, y_pred)
accuracy = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)

print("Confusion Matrix:\n", cm)
print('\nPersentase Dari:')
print("Accuracy :", accuracy)
print("Precision:", precision)
print("Recall   :", recall)
```

```
Confusion Matrix:
 [[4 1]
 [2 3]]

Persentase Dari:
Accuracy : 0.7
Precision: 0.75
Recall   : 0.6
```

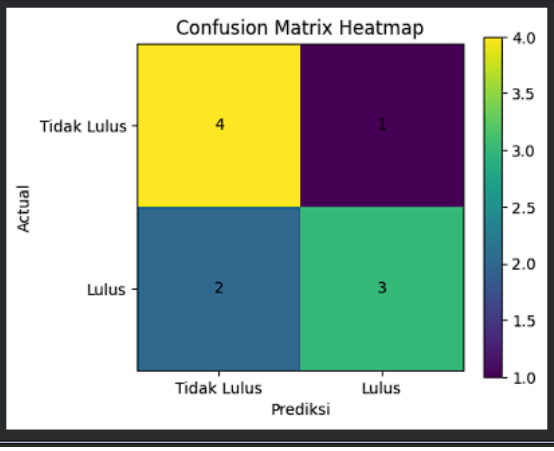
Kode ini digunakan untuk menghitung hasil evaluasi model berdasarkan data aktual dan data prediksi. Nilai Actual_enc dijadikan sebagai *y_true*, sedangkan Prediksi_enc sebagai *y_pred*. Dengan fungsi dari *sklearn*, dihitung beberapa metrik evaluasi, yaitu:

- Confusion Matrix, untuk melihat jumlah prediksi benar dan salah.
- Accuracy, persentase keseluruhan prediksi yang tepat.
- Precision, seberapa tepat model saat memprediksi “Lulus”.
- Recall, seberapa banyak data “Lulus” yang berhasil dikenali model.

Hasil evaluasi tersebut kemudian ditampilkan untuk mengetahui performa model secara keseluruhan.

menampilkan heatmap

```
[11]
✓ 0s
import matplotlib.pyplot as plt

plt.figure(figsize=(5,4))
plt.imshow(cm)
plt.title("Confusion Matrix Heatmap")
plt.xlabel("Prediksi")
plt.ylabel("Actual")
plt.xticks([0,1], ["Tidak Lulus", "Lulus"])
plt.yticks([0,1], ["Tidak Lulus", "Lulus"])
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        plt.text(j, i, cm[i, j], ha="center", va="center")
plt.colorbar()
plt.tight_layout()

plt.show()
```



Kode ini digunakan untuk menampilkan heatmap confusion matrix agar hasil evaluasi model lebih mudah dilihat. plt.imshow(cm) menampilkan matriks dalam bentuk warna, lalu ditambahkan judul, label sumbu, dan nama kategori pada sumbu X dan Y. Setiap nilai dalam matriks dituliskan di tengah kotaknya menggunakan plt.text(). Warna pada heatmap dibantu oleh plt.colorbar(). Dengan visualisasi ini, kita dapat melihat pola prediksi benar dan salah secara lebih jelas.

Latihan 3

```
Latihan 3

[12]
✓
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
```

Kode ini digunakan untuk mengimpor berbagai library yang diperlukan dalam proses machine learning. LabelEncoder dan StandardScaler digunakan untuk mengubah data kategori menjadi angka dan menstandarkan skala data. train_test_split dipakai untuk membagi data menjadi latih dan uji, sementara KNeighborsClassifier adalah algoritma KNN yang digunakan untuk membuat model. confusion_matrix dan classification_report digunakan untuk evaluasi hasil prediksi. matplotlib.pyplot dan seaborn membantu dalam membuat visualisasi seperti grafik dan heatmap.

Mengambil data dari google sheet

```
[15]
✓ 4s
df3 = pd.read_excel("https://docs.google.com/spreadsheets/d/1iG5HxB_odohh6Wit992g9MPvL5xapr64/export?format=xlsx")
df3.head()
```

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Location	Weather Type
0	14	73	9.5	82	partly cloudy	1010.82	2	Winter	3.5	inland	Rainy
1	39	96	8.5	71	partly cloudy	1011.43	7	Spring	10.0	inland	Cloudy
2	30	64	7.0	16	clear	1018.72	5	Spring	5.5	mountain	Sunny
3	38	83	1.5	82	clear	1026.25	7	Spring	1.0	coastal	Sunny
4	27	74	17.0	66	overcast	990.67	1	Winter	2.5	mountain	Rainy

Kode ini digunakan untuk mengambil data langsung dari Google Sheets dalam bentuk file Excel. pd.read_excel() membaca file dari link yang sudah dikonversi ke format *export*. Hasilnya disimpan dalam variabel df3, lalu df3.head() menampilkan beberapa baris pertama dari dataset untuk memastikan bahwa data berhasil dimuat dengan benar.

Mengubah Kolom Kategori Menjadi Angka

```
[16]
✓
categorical_cols = ["Cloud Cover", "Season", "Location", "Weather Type"]

le_dict = {}
for col in categorical_cols:
    le = LabelEncoder()
    df3[col] = le.fit_transform(df3[col])
    le_dict[col] = le
```

Kode ini digunakan untuk mengubah kolom-kolom kategori menjadi bentuk angka agar bisa diproses oleh model machine learning. Daftar kolom kategori disimpan dalam categorical_cols, lalu setiap kolom diproses menggunakan LabelEncoder(). Nilai kategori diubah menjadi angka melalui fit_transform(), dan encoder untuk tiap kolom disimpan dalam le_dict agar bisa digunakan kembali jika diperlukan. Dengan cara ini, semua data kategori pada df3 menjadi numerik dan siap digunakan dalam pemodelan.

Memisahkan data fitur dan label

```
[17]
✓
X = df3.drop("Weather Type", axis=1)
y = df3["Weather Type"]
```

Kode ini digunakan untuk memisahkan data fitur dan label. Bagian X berisi seluruh kolom kecuali *Weather Type*, karena kolom-kolom inilah yang menjadi input untuk model. Sementara y berisi kolom Weather Type, yaitu kategori cuaca yang ingin diprediksi. Dengan pemisahan ini, data siap digunakan dalam proses pelatihan model machine learning.

Standarisasi Data dengan StandardScaler

```
[18]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Proses ini mengubah setiap fitur numerik agar memiliki rata-rata 0 dan standar deviasi 1, sehingga semua variabel berada pada skala yang sama dan model KNN dapat bekerja lebih akurat.

Train-Test Split (Pembagian Data)

```
[19]
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)
```

Kode ini membagi data menjadi data latih (train) dan data uji (test). Sebanyak 80% data digunakan untuk melatih model, dan 20% digunakan untuk menguji performa model. Parameter random_state=42 digunakan agar pembagian selalu sama setiap kali dijalankan.

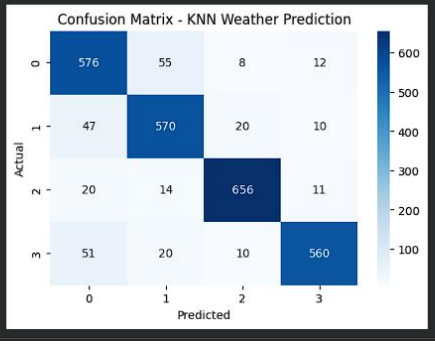
Training model KNN

```
[20]
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)

KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

Kode ini membuat model K-Nearest Neighbors (KNN) dengan jumlah tetangga k = 3. Kemudian, model tersebut dilatih (fit) menggunakan data latih X_train sebagai fitur dan y_train sebagai label target. Model akan mempelajari pola hubungan antara input dan kategori *Weather Type*.

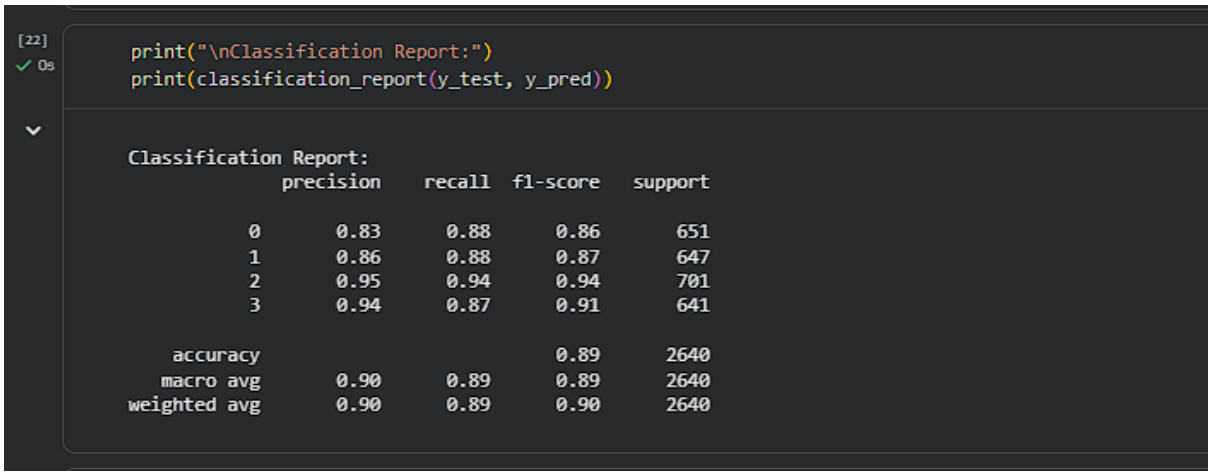
Memprediksi & Menampilkan Confusion Matrix

```
[21]
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix - KNN Weather Prediction")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



Kode ini menggunakan model untuk memprediksi label cuaca pada data uji (X_test). Hasil prediksi (y_pred) kemudian dibandingkan dengan label asli (y_test) untuk membuat confusion matrix.

Menampilkan Classification Report



```
[22]
✓ 0s
```

```
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.88      0.86       651
           1       0.86      0.88      0.87       647
           2       0.95      0.94      0.94       701
           3       0.94      0.87      0.91       641

    accuracy                           0.89      2640
   macro avg       0.90      0.89      0.89      2640
weighted avg       0.90      0.89      0.90      2640
```

Kodingan di atas digunakan untuk menampilkan classification report dari hasil prediksi model KNN. Perintah classification_report(y_test, y_pred) otomatis menghitung metrik penting seperti precision, recall, f1-score, dan support untuk setiap kelas cuaca. Output ini memberikan gambaran lengkap tentang seberapa baik model mengenali tiap kategori, bukan hanya keseluruhan performanya. Ini membantu menilai apakah model sudah seimbang dalam memprediksi semua kelas atau masih bias pada kelas tertentu.