UJIAN AKHIR SEMESTER (UAS)
MACHINE LEARNING

Aulia Sindi Regita Pramesti

2211010134

INSTITUT INFORMATIKA DAN BISNIS DARMAJAYA

BANDAR LAMPUNG

TAHUN 2026

Link Kaggle dataset iris

https://www.kaggle.com/datasets/taweilo/iris-dataset-elarged-with-smote

link git hub

https://github.com/auliasindirp

1. Memasukan library pada google colab

```
import pandas as pd
import numpy as np
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, precision_score, recall_score, f1_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

2. Load data dan menampilkan data

Membaca data set, menampilkan jumlah data dan fitur, serta nama fitur yang berfungsi untuk memastikan bahwa file CSV telah berhasil dibaca oleh Python dan tidak kosong.

```
# 1. MEMBACA DATASET
# Pastikan file "iris_synthetic_data.csv" berada di folder yang sama dengan script ini
try:
    data = pd.read_csv("iris_synthetic_data.csv")
    print("Dataset berhasil dimuat! ✅")
except Exception as e:
    print(f"Gagal memuat file: {e}")

# Lihat 5 data pertama
display(data)

# Cek struktur data
data.info()
```

2.1 Menampilkan jumlah data & fitur, nama fitur dan distribusi kelas target

Dataset berhasil dimuat! ✅

1 to 25 of 3000 entries Filter

index	sepal length	sepal width	petal length	petal width	label
0	5.2	3.8	1.5	0.3	Iris-setosa
1	5.3	4.1	1.5	0.1	Iris-setosa
2	4.8	3.1	1.5	0.2	Iris-setosa
3	5.2	3.7	1.5	0.2	Iris-setosa
4	4.9	3.0	1.5	0.3	Iris-setosa
5	5.1	3.7	1.5	0.4	Iris-setosa
6	5.2	3.7	1.5	0.3	Iris-setosa
7	5.4	3.4	1.6	0.2	Iris-setosa
8	4.9	3.1	1.5	0.1	Iris-setosa
9	5.2	3.7	1.5	0.3	Iris-setosa
10	5.0	3.3	1.5	0.2	Iris-setosa
11	5.0	3.4	1.5	0.2	Iris-setosa
12	5.4	3.9	1.5	0.4	Iris-setosa
13	4.8	3.1	1.5	0.2	Iris-setosa
14	4.9	3.1	1.5	0.1	Iris-setosa
15	4.4	3.1	1.1	0.1	Iris-setosa
16	5.3	3.7	1.5	0.2	Iris-setosa
17	5.2	3.4	1.5	0.2	Iris-setosa
18	4.5	3.2	1.3	0.2	Iris-setosa
19	4.9	3.0	1.5	0.2	Iris-setosa
20	4.9	3.4	1.6	0.2	Iris-setosa

11:23 AM Python 3

Mendeteksi dimensi dataset (3000 baris dan 4 fitur input).

```
Like what you see? Visit the data table notebook to learn more about interactive tables.
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal length  3000 non-null   float64
 1   sepal width   3000 non-null   float64
 2   petal length  3000 non-null   float64
 3   petal width   3000 non-null   float64
 4   label         3000 non-null   object
dtypes: float64(4), object(1)
memory usage: 117.3+ KB
```

Menampilan distribusi kelas target

```python
# 2. TAMPILKAN DISTRIBUSI KELAS TARGET
print("=== DISTRIBUSI KELAS TARGET (TEKS) ===")
distribusi = data['label'].value_counts()
print(distribusi)

# Visualisasi Distribusi Kelas Target
plt.figure(figsize=(8, 5))
sns.countplot(x='label', data=data, palette='viridis')
plt.title('Grafik Distribusi Kelas Target (Spesies Iris)')
plt.xlabel('Spesies')
plt.ylabel('Jumlah Data')
plt.show()
```

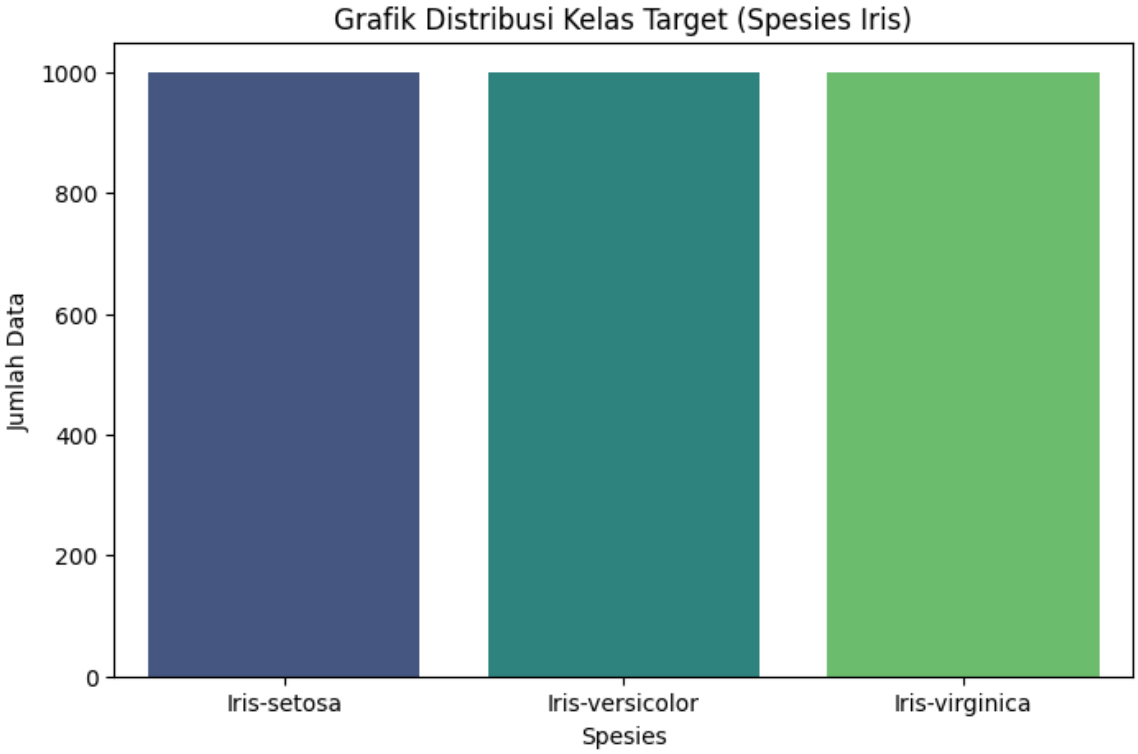
Dataset iris ini memiliki total 3000 baris data yang terbagi secara merata ke dalam tiga jenis spesies bunga Iris:

- **Iris-setosa**: 1000 data
- **Iris-versicolor**: 1000 data
- **Iris-virginica**: 1000 data

Karena setiap kelas memiliki jumlah yang sama persis (masing-masing 33,3% dari total data), dataset ini disebut sebagai Balanced Dataset (Dataset Seimbang).

```
=== DISTRIBUSI KELAS TARGET (TEKS) ===
label
Iris-setosa        1000
Iris-versicolor    1000
Iris-virginica     1000
Name: count, dtype: int64
```

Visualisasi distribusi kelas target dataset iris



3. Prepocessing

```
# 3. PRE-PROCESSING
# Menghapus duplikat agar model lebih akurat
data = data.drop_duplicates()

# Pisahkan Fitur dan Target
X = data.drop(columns=['label'])
y = data['label']

# Encode label (Teks ke Angka)
le = LabelEncoder()
y_encoded = le.fit_transform(y)
```

3.1 membagi dataset menjadi data latih (training) dan data uji (testing)

```
# 4. SPLIT DATA (70% Training, 30% Testing)
X_train_raw, X_test_raw, y_train, y_test = train_test_split(
    X, y_encoded, test_size=0.30, random_state=42
)
```

3.2 Tahap Feature Scaling (normalisasi fitur) menggunakan StandardScaler

```
# 5. FEATURE SCALING
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train_raw)
X_test = scaler.transform(X_test_raw)
```

4. Implementasi KNN

```
# 6. PEMODELAN KNN (k=5)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
print("Model KNN berhasil dilatih! ✅")
```

```
Model KNN berhasil dilatih! ✅
```

4.1 Melakukan evaluasi model KNN menggunakan data testing

```
# 7. EVALUASI MODEL
y_pred = knn.predict(X_test)

print("\n=== HASIL EVALUASI MODEL KNN ===")
print(f"Accuracy  : {accuracy_score(y_test, y_pred)*100:.2f}%")
print(f"Precision : {precision_score(y_test, y_pred, average='weighted')*100:.2f}%")
print(f"Recall    : {recall_score(y_test, y_pred, average='weighted')*100:.2f}%")
print(f"F1-Score  : {f1_score(y_test, y_pred, average='weighted')*100:.2f}%")

print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=le.classes_))
```

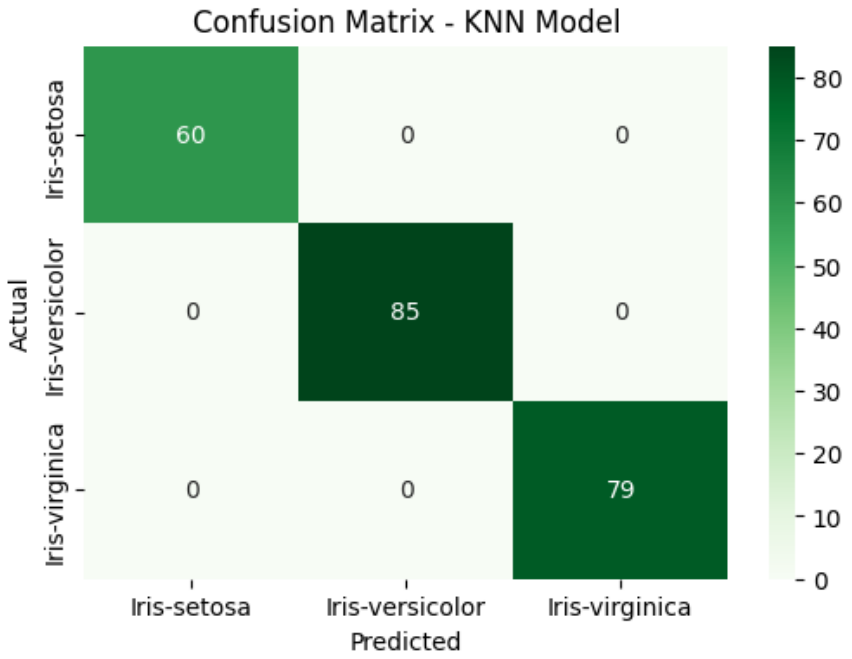
Model KNN menunjukkan performa sangat baik dengan nilai accuracy, precision, recall, dan F1-score sebesar 100%. Hal ini menandakan bahwa model mampu mengklasifikasikan seluruh data uji dengan tepat pada setiap kelas Iris.

```
=== HASIL EVALUASI MODEL KNN ===
Accuracy  : 100.00%
Precision : 100.00%
Recall    : 100.00%
F1-Score  : 100.00%

Classification Report:
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        60
Iris-versicolor       1.00      1.00      1.00        85
 Iris-virginica       1.00      1.00      1.00        79

       accuracy                           1.00       224
      macro avg       1.00      1.00      1.00       224
   weighted avg       1.00      1.00      1.00       224
```

4.2 visualisasi confusion matrix

Confusion matrix menampilkan perbandingan antara kelas sebenarnya (actual) dan kelas hasil prediksi (predicted)

```
# 8. VISUALISASI CONFUSION MATRIX
plt.figure(figsize=(6,4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Greens',
            xticklabels=le.classes_, yticklabels=le.classes_)
plt.title('Confusion Matrix - KNN Model')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



4.3 Prediksi manual / input manual

```
# 9. PREDIKSI MANUAL
print("\n--- TEST PREDIKSI MANUAL ---")
try:
    sl = float(input("Masukkan Sepal Length: "))
    sw = float(input("Masukkan Sepal Width : "))
    pl = float(input("Masukkan Petal Length: "))
    pw = float(input("Masukkan Petal Width : "))

    # Skalakan data input manual sebelum prediksi
    input_data = scaler.transform([[sl, sw, pl, pw]])
    prediksi = knn.predict(input_data)
    hasil = le.inverse_transform(prediksi)

    print(f"\n🌸 Hasil Prediksi: {hasil[0]}")
except ValueError:
    print("Error: Harap masukkan angka saja!")
```

Hasil tes prediksi manual

```
--- TEST PREDIKSI MANUAL ---
Masukkan Sepal Length: 6.2
Masukkan Sepal Width : 3
Masukkan Petal Length: 4.5
Masukkan Petal Width : 1.5

🌸 Hasil Prediksi: Iris-versicolor
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does
  warnings.warn(
```