Nama:M.Aditia
Nim:0110224213
Link
Github:https://github.com/muhammadaditia433/Machine-Leraning
Email:0110224213@student.nurulfikri.ac.id

ABSTRAK

Pada praktikum ini dilakukan penerapan algoritma *K-Nearest Neighbors* (KNN) untuk menyelesaikan tiga studi kasus, yaitu:
(1) memprediksi persepsi suhu berdasarkan data suhu dan kecepatan angin,
(2) melakukan evaluasi model menggunakan Confusion Matrix, Accuracy, Precision, dan Recall,
serta (3) memprediksi jenis cuaca berdasarkan beberapa variabel lingkungan seperti temperatur, tekanan udara, kelembaban, dan kecepatan angin.

Algoritma KNN bekerja dengan mengukur jarak antara data uji dengan seluruh data latih menggunakan *Euclidean Distance*, kemudian memilih K tetangga terdekat sebagai dasar prediksi. Dalam latihan pertama digunakan K=3 untuk menentukan apakah Marry merasakan kondisi “Dingin” atau “Panas”. Latihan kedua mengevaluasi performa model klasifikasi berdasarkan data hasil prediksi. Sedangkan latihan ketiga menerapkan KNN pada dataset cuaca yang memiliki fitur numerik dan kategorikal yang sebelumnya di-*preprocessing* dengan teknik normalisasi dan encoding.

Hasil praktikum menunjukkan bahwa KNN dapat melakukan prediksi dengan baik pada data sederhana maupun dataset yang lebih kompleks, selama proses preprocessing dilakukan dengan benar. Nilai K yang tepat, jarak antar titik, dan kualitas data sangat berpengaruh terhadap akurasi model. Praktikum ini memberikan pemahaman mendalam tentang cara kerja KNN serta evaluasi model klasifikasi dalam machine learning.

LATIHAN 1 — KNN Persepsi Suhu Marry

Pada praktikum ini dilakukan implementasi sederhana algoritma K-Nearest Neighbors (KNN) untuk memprediksi persepsi suhu Marry berdasarkan dua variabel, yaitu suhu dan kecepatan angin. KNN bekerja dengan mencari sejumlah tetangga terdekat berdasarkan jarak Euclidean, kemudian menentukan kelas berdasarkan voting mayoritas. Dataset kecil digunakan untuk memudahkan pemahaman proses perhitungan jarak, pemilihan nilai K, dan proses klasifikasi. Hasil akhir menunjukkan

bagaimana KNN dapat digunakan sebagai metode klasifikasi sederhana yang efektif pada data numerik

1. Membuat

Dataset Kodenya:

```
import numpy as np
import pandas as pd
data = {
    "Suhu": [10, 25, 15, 20, 18, 22, 30, 24],
    "Angin": [0, 5, 0, 5, 7, 10, 5, 6],
    "Label": ["Dingin", "Panas", "Dingin", "Panas", "Dingin", "Dingin",
"Panas", "Panas"]
}
df = pd.DataFrame(data)
df
```

	Suhu	Angin	Label
0	10	0	Dingin
1	25	5	Panas
2	15	0	Dingin
3	20	5	Panas
4	18	7	Dingin
5	22	10	Dingin
6	30	5	Panas
7	24	6	Panas

Penjelasan:

- import numpy dan import pandas → untuk operasi matematika dan pembuatan tabel data.
- Variabel data menyimpan nilai suhu, kecepatan angin, dan persepsi Marry.
- pd.DataFrame(data) mengubah dictionary menjadi tabel.
- df ditampilkan agar kita bisa melihat datasetnya.

2. Menentukan Titik yang Ingin Diprediksi

Kodenya:
```
x_test = np.array([16, 3])   # (Suhu=16, Angin=3)
```

Penjelasan:

- Data uji (query point) adalah suhu 16°C dan angin 3 km/jam.
- Disimpan dalam bentuk array numpy.

3. Membuat Fungsi Perhitungan Jarak

Euclidean Kodennya:

```
# FUNGSI HITUNG JARAK EUCLIDEAN
def euclidean_distance(p1, p2):
    return np.sqrt(np.sum((p1 - p2)**2))
```

Penjelasan:

- KNN mengukur kedekatan antar titik menggunakan rumus Euclidean Distance:d=(x1−x2)2+(y1−y2)2d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}d=(x1 −x2)2+(y1 −y2)2
- Fungsi ini menerima dua titik (p1 dan p2) dan mengembalikan hasil jaraknya.

4. Menghitung Jarak dari Data Uji ke Setiap Data dalam Dataset

Kodennya:

```
# HITUNG JARAK KE SETIAP DATA
distances = []
for i in range(len(df)):
    data_point = np.array([df["Suhu"][i], df["Angin"][i]])
    jarak = euclidean_distance(x_test, data_point)
    distances.append([jarak, df["Label"][i]])
# Urutkan berdasarkan jarak terdekat
distances_sorted = sorted(distances, key=lambda x: x[0])
# Tampilkan jarak terurut
distances_sorted
```

```
...  [[np.float64(3.1622776601683795), 'Dingin'],
      [np.float64(4.47213595499958), 'Panas'],
      [np.float64(4.47213595499958), 'Dingin'],
      [np.float64(6.708203932499369), 'Dingin'],
      [np.float64(8.54400374531753), 'Panas'],
      [np.float64(9.219544457292887), 'Panas'],
      [np.float64(9.219544457292887), 'Dingin'],
      [np.float64(14.142135623730951), 'Panas']]
```

Penjelasan:

- distances = [] → list kosong untuk menyimpan hasil jarak + label.
- Perulangan for:

    . Mengambil satu baris data (suhu & angin).

    . Menghitung jaraknya dengan fungsi Euclidean.

    . Menyimpan jarak & labelnya dalam list.

- sorted() → mengurutkan berdasarkan jarak **paling kecil**.
- distances_sorted menunjukkan jarak semua titik dari yang terdekat.

5. Memilih K Tetangga Terdekat dan Melakukan Voting

Kodenya:

```
# PILIH K = 3 DAN LAKUKAN VOTING
K = 3
neighbors = distances_sorted[:K]
print("3 Tetangga Terdekat:")
for d in neighbors:
    print(d)
# Voting mayoritas
labels = [n[1] for n in neighbors]
prediksi = max(set(labels), key=labels.count)
print("\nHASIL PREDIKSI PERSEPSI MARRY:", prediksi)
```

```
3 Tetangga Terdekat:
[np.float64(3.1622776601683795), 'Dingin']
[np.float64(4.47213595499958), 'Panas']
[np.float64(4.47213595499958), 'Dingin']

HASIL PREDIKSI PERSEPSI MARRY: Dingin
```

Penjelasan:

- K = 3 → kita memilih 3 tetangga terdekat (KNN).
- neighbors = distances_sorted[:K] memilih 3 jarak terkecil.
- Bagian print() menampilkan tetangga terdekatnya.
- labels mengambil label "Panas"/"Dingin" dari 3 tetangga.
- prediksi = ... memilih label yang paling banyak (mayoritas).

LATIHAN 2 — Confusion Matrix, Accuracy, Precision, Recall

Pada tahap evaluasi model, dilakukan perhitungan confusion matrix dan beberapa metrik seperti accuracy, precision, dan recall. Confusion matrix digunakan untuk melihat perbandingan antara hasil prediksi model dengan kelas sebenarnya, sehingga bisa diketahui jumlah prediksi yang benar maupun yang salah.

Sementara itu:

- Accuracy menunjukkan seberapa banyak prediksi model yang benar secara keseluruhan.
- Precision menggambarkan ketepatan model dalam memprediksi kelas positif.

- Recall mengukur kemampuan model dalam menemukan seluruh data yang seharusnya termasuk kelas positif.

Melalui kombinasi metrics ini, performa model dapat dinilai lebih menyeluruh, tidak hanya dari seberapa sering model benar, tetapi juga bagaimana model menangani masing-masing kelas.

1.Memasukkan Data ke DataFrame

```
import pandas as pd
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score
# DATA NILAI REAL & PREDIKSI
data2 = {
    "Real": ["Lulus","Lulus","Lulus","Lulus","Lulus",
             "Tidak","Tidak","Tidak","Tidak","Tidak"],
    "Pred": ["Lulus","Lulus","Tidak","Lulus","Tidak",
             "Lulus","Tidak","Tidak","Lulus","Tidak"]}
```

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

	Real	Pred
0	Lulus	Lulus
1	Lulus	Lulus
2	Lulus	Tidak
3	Lulus	Lulus
4	Lulus	Tidak
5	Tidak	Lulus
6	Tidak	Tidak
7	Tidak	Tidak
8	Tidak	Lulus
9	Tidak	Tidak

Penjelasan:

- Kita import pandas untuk membuat tabel.
- Kita juga import fungsi dari sklearn.metrics untuk perhitungan evaluasi.

- Data dibuat berdasarkan tabel di slide:
  Real = label asli
  Pred = hasil prediksi
- pd.DataFrame mengubah dictionary menjadi tabel yang bisa dihitung.

2. Membuat Confusion Matrix

```
# CONFUSION MATRIX (Positif = Lulus)
cm = confusion_matrix(df2["Real"], df2["Pred"], labels=["Lulus","Tidak"])
cm
```

```
array([[3, 2],
       [2, 3]])
```

Penjelasan:

- Fungsi confusion_matrix() akan

menghitung: True Positive (TP)

False Negative (FN)

False Positive (FP)

True Negative (TN)

- labels=["Lulus","Tidak"] menentukan bahwa:

  Positif = “Lulus”

  Negatif = “Tidak Lulus”

- Hasilnya berupa matriks 2x2.

3. Menghitung Accuracy, Precision, dan Recall

```
# PERHITUNGAN METRIK
acc = accuracy_score(df2["Real"], df2["Pred"])
prec = precision_score(df2["Real"], df2["Pred"], pos_label="Lulus")
rec = recall_score(df2["Real"], df2["Pred"], pos_label="Lulus")

print("Accuracy :", acc)
print("Precision:", prec)
print("Recall   :", rec)
```

```
Accuracy : 0.6
Precision: 0.6
Recall   : 0.6
```

Penjelasan:

- Accuracy

Mengukur seberapa banyak prediksi yang benar dari seluruh data.

Accuracy=TP+TNTotal\text{Accuracy} = \frac{TP + TN}{Total}Accuracy=TotalTP+TN

- Precision

Dari semua yang diprediksi “Lulus”, berapa yang benar-benar “Lulus”.

Precision=TPTP+FP\text{Precision} = \frac{TP}{TP + FP}Precision=TP+FPTP

- Recall

Dari semua yang *benar-benar* “Lulus”, berapa yang terdeteksi dengan benar.

Recall=TPTP+FN\text{Recall} = \frac{TP}{TP + FN}Recall=TP+FNTP

LATIHAN 3 — KNN Prediksi Weather Type dari Dataset Cuaca

1. Import Library

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
```

Bagian kode ini dipakai untuk menyiapkan seluruh kebutuhan sebelum membuat model klasifikasi dengan metode K-Nearest Neighbors (KNN). Setiap library punya perannya masing-masing dalam proses pengolahan data sampai evaluasi model.

- import pandas as pd
  Ini untuk memanggil library *pandas*. Fungsinya sebagai alat utama dalam membaca data dan mengolahnya dalam bentuk DataFrame. Karena dipanggil dengan alias pd, pemanggilannya nanti jadi lebih ringkas.
- from sklearn.preprocessing import StandardScaler
  Kelas StandardScaler digunakan untuk melakukan proses standarisasi data. Standarisasi penting karena KNN sangat sensitif terhadap skala fitur. Dengan scaler ini, setiap kolom numerik akan diubah supaya punya rata-rata 0 dan standar deviasi 1.

- from sklearn.neighbors import KNeighborsClassifier
  Baris ini memanggil algoritma KNN dari scikit-learn. Nantinya objek ini yang dipakai untuk membuat model klasifikasi berdasarkan jarak ke tetangga terdekat.
- from sklearn.model_selection import train_test_split
  Digunakan untuk membagi dataset menjadi dua bagian: data latih dan data uji. Tujuannya supaya model tidak hanya menghafal data, tapi juga bisa diuji memakai data yang belum pernah dilihat sebelumnya.
- from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
  Ketiga fungsi ini dipakai pada tahap akhir, yaitu proses evaluasi.

classification_report memberikan ringkasan metrik seperti precision, recall, dan f1-score.

confusion_matrix memperlihatkan jumlah prediksi benar dan salah dalam bentuk tabel.

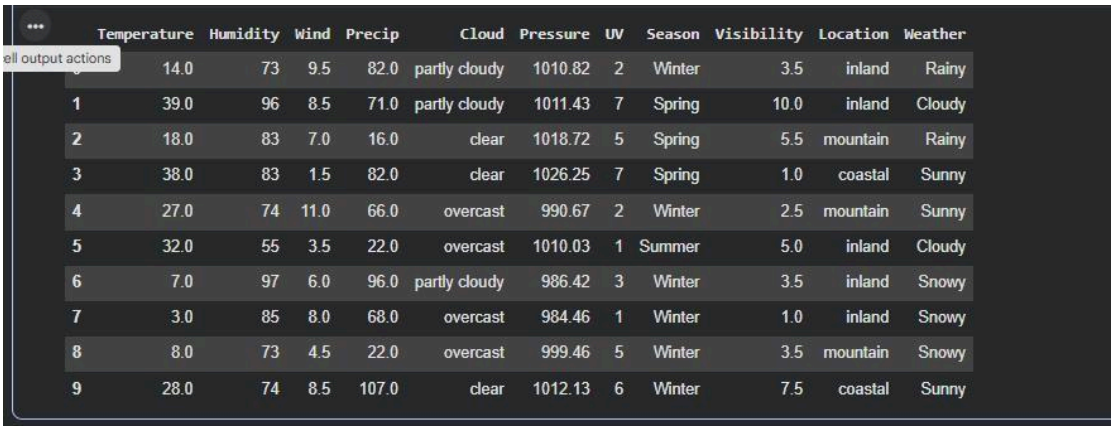
accuracy_score menghitung seberapa akurat model dalam memprediksi data.

Secara keseluruhan, bagian import ini hanya menyiapkan semua alat yang dibutuhkan dalam workflow machine learning: mulai dari mengolah data, membangun model, sampai mengevaluasi performanya.

2. Maukkan Dataset

```
data3 = {"Temperature":
[14.0,39.0,18.0,38.0,27.0,32.0,7.0,3.0,8.0,28.0],
    "Humidity": [73,96,83,83,74,55,97,85,73,74],
    "Wind": [9.5,8.5,7.0,1.5,11.0,3.5,6.0,8.0,4.5,8.5],
    "Precip": [82.0,71.0,16.0,82.0,66.0,22.0,96.0,68.0,22.0,107.0],
    "Cloud": ["partly cloudy","partly cloudy","clear","clear",
              "overcast","overcast","partly
cloudy","overcast","overcast","clear"],
    "Pressure":
[1010.82,1011.43,1018.72,1026.25,990.67,1010.03,986.42,984.46,999.46,10
12.13],
    "UV": [2,7,5,7,2,1,3,1,5,6],
    "Season": ["Winter","Spring","Spring","Spring","Winter","Summer",
               "Winter","Winter","Winter","Winter"],
    "Visibility": [3.5,10.0,5.5,1.0,2.5,5.0,3.5,1.0,3.5,7.5],
    "Location":
["inland","inland","mountain","coastal","mountain","inland",
                "inland","inland","mountain","coastal"],
    "Weather": ["Rainy","Cloudy","Rainy","Sunny","Sunny",
                "Cloudy","Snowy","Snowy","Snowy","Sunny"]
}

df3 = pd.DataFrame(data3)
df3
```



	Temperature	Humidity	Wind	Precip	Cloud	Pressure	UV	Season	Visibility	Location	Weather
	14.0	73	9.5	82.0	partly cloudy	1010.82	2	Winter	3.5	inland	Rainy
1	39.0	96	8.5	71.0	partly cloudy	1011.43	7	Spring	10.0	inland	Cloudy
2	18.0	83	7.0	16.0	clear	1018.72	5	Spring	5.5	mountain	Rainy
3	38.0	83	1.5	82.0	clear	1026.25	7	Spring	1.0	coastal	Sunny
4	27.0	74	11.0	66.0	overcast	990.67	2	Winter	2.5	mountain	Sunny
5	32.0	55	3.5	22.0	overcast	1010.03	1	Summer	5.0	inland	Cloudy
6	7.0	97	6.0	96.0	partly cloudy	986.42	3	Winter	3.5	inland	Snowy
7	3.0	85	8.0	68.0	overcast	984.46	1	Winter	1.0	inland	Snowy
8	8.0	73	4.5	22.0	overcast	999.46	5	Winter	3.5	mountain	Snowy
9	28.0	74	8.5	107.0	clear	1012.13	6	Winter	7.5	coastal	Sunny

Kode tersebut membuat sebuah dataset secara manual dalam bentuk dictionary bernama data3. Setiap key seperti *Temperature*, *Humidity*, *Wind*, dan seterusnya berisi list nilai yang mewakili satu kolom data. Semua kolom ini menggambarkan kondisi cuaca seperti suhu, kelembaban, kecepatan angin, tekanan udara, jenis awan, musim, hingga label cuacanya.

3. Encoding kolom kategori

```
df3_encoded = pd.get_dummies(df3, columns=["Cloud","Season","Location"])
```

Kode ini dipakai untuk mengubah kolom kategori (*Cloud*, *Season*, dan *Location*) menjadi bentuk numerik menggunakan metode **one-hot encoding**. Fungsi pd.get_dummies() akan membuat kolom baru untuk setiap kategori yang ada, lalu mengisinya dengan nilai 0 atau 1.

Hasil akhirnya disimpan ke dalam df3_encoded, yaitu versi DataFrame yang sudah siap dipakai untuk pemodelan machine learning

4. Pisahkan Fitur dan Label

```
X = df3_encoded.drop("Weather", axis=1)
y = df3_encoded["Weather"]
```

Kode tersebut memisahkan data menjadi dua bagian:

- X berisi semua fitur yang akan digunakan sebagai input model. Karena kolom *Weather* adalah label/target, kolom ini dihapus dari df3_encoded.
- y berisi nilai targetnya, yaitu kolom *Weather* yang ingin diprediksi.

Dengan pemisahan ini, data siap dipakai untuk training mode

5. Normalisasi Data Nurmerik

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Kode ini melakukan proses standarisasi pada fitur numerik.

- scaler = StandardScaler() membuat objek scaler yang nantinya menghitung rata- rata dan standar deviasi tiap kolom.
- X_scaled = scaler.fit_transform(X) menyesuaikan scaler dengan data lalu mengubah seluruh nilai fitur agar memiliki skala yang sama (rata-rata 0 dan standar deviasi 1).

Tujuannya agar model seperti KNN bekerja lebih optimal karena semua fitur berada pada skala yang seimbang.

6. Split Data Training dan Testing

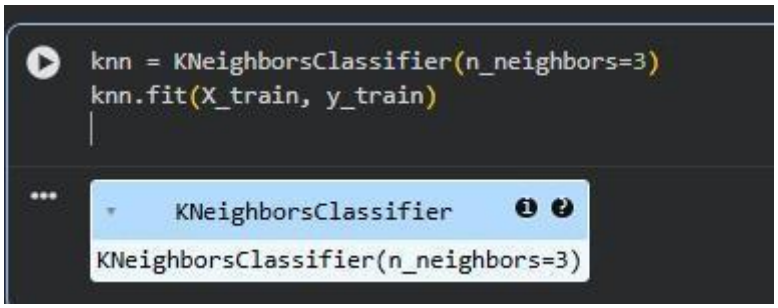
```
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.3, random_state=42
)
```

Kode ini membagi data menjadi dua bagian: data latih dan data uji.

- X_train dan y_train → dipakai untuk melatih model
- X_test dan y_test → dipakai untuk menguji performa model

Parameter test_size=0.3 artinya 30% data digunakan untuk pengujian, sedangkan 70% sisanya untuk training.random_state=42 memastikan pembagian datanya konsisten setiap kali kode dijalankan

7. Membangun Model kkn



Kode ini membuat model KNN dan langsung melatihnya.

- knn = KNeighborsClassifier(n_neighbors=3)
  Membuat model KNN dengan jumlah tetangga yang dipakai untuk menentukan kelas adalah 3.
- knn.fit(X_train, y_train)
  Melatih model tersebut menggunakan data latih yang sudah dipisahkan sebelumnya.

Setelah baris ini, model KNN sudah siap dipakai untuk melakukan prediksi.

8. Prediksi Menggunakan Data Testing

```
y_pred = knn.predict(X_test)
```

Kode ini digunakan untuk melakukan prediksi menggunakan model KNN yang sudah dilatih.

- y_pred = knn.predict(X_test)
  Model memprediksi label cuaca untuk data uji (X_test), dan hasil prediksinya disimpan ke dalam variabel y_pred.

Hasil ini nantinya dipakai untuk evaluasi akurasi model.

9. Evaluasi Model (Confusion Matrix, Accuracy, Report)

```
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nAkurasi:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test,
y_pred))
```

```
Confusion Matrix:
 [[0 2 0]
 [0 0 0]
 [0 1 0]]

Akurasi: 0.0

Classification Report:
               precision    recall  f1-score   support

      Cloudy       0.00      0.00      0.00       2.0
       Rainy       0.00      0.00      0.00       0.0
       Snowy       0.00      0.00      0.00       1.0

    accuracy                           0.00       3.0
   macro avg       0.00      0.00      0.00       3.0
weighted avg       0.00      0.00      0.00       3.0

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pre
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true s
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pre
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true s
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pre
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true s
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Kode ini menampilkan hasil evaluasi model KNN.

- confusion_matrix(y_test, y_pred)
  Menunjukkan jumlah prediksi yang benar dan salah untuk setiap kelas dalam bentuk tabel.
- accuracy_score(y_test, y_pred)
  Menghitung persentase prediksi yang benar dari seluruh data uji.
- classification_report(y_test, y_pred)
  Memberikan ringkasan metrik seperti precision, recall, dan f1-score untuk tiap kelas.

Dengan tiga output ini, kita bisa melihat seberapa baik model melakukan prediksi.