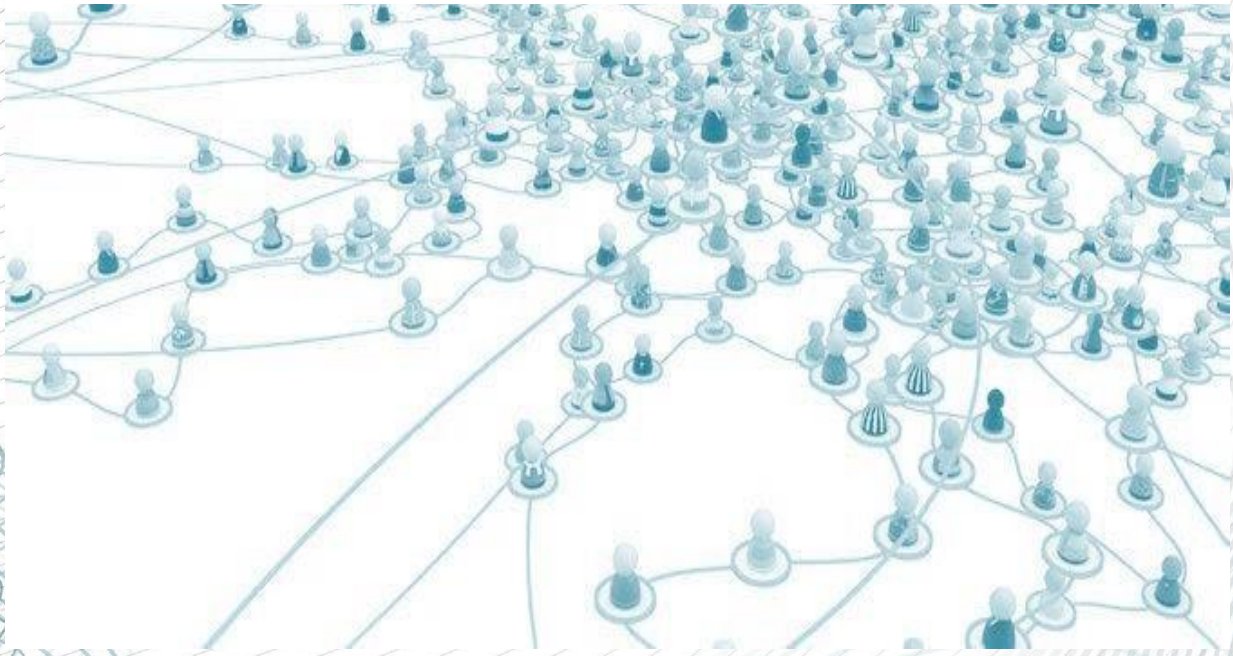
MODUL PRAKTIKUM

SD3107-Pembelajaran Mesin



Program Studi Sains Data
Fakultas Sains
Institut Teknologi Sumatera

2024

MODUL 4

Klasterisasi K-Means

A. Konsep Dasar

Klasterisasi adalah teknik Data Mining (penambangan data) dan Machine Learning (pembelajaran mesin) yang melibatkan pengelompokan sekumpulan objek sehingga mereka yang berada dalam kelompok yang sama (disebut Cluster) relatif memiliki karakteristik serupa dibandingkan di luar kelompok. Pengelompokan (Clustering) dalam konteks Machine Learning memiliki makna secara sederhana, yaitu sebuah konsep mesin belajar mengelompokkan sebuah data yang tidak memiliki keterangan kelas (data tidak berlabel).

K-Means Clustering adalah proses untuk melakukan pengklasteran n data ke dalam k klaster. Asumsinya adalah semua objek atribut membentuk sebuah ruang vektor fitur (numerik). Tujuannya adalah membagi data ke dalam sejumlah klaster **K** yang ditentukan di awal, di mana setiap klaster direpresentasikan oleh pusatnya yang disebut **centroid**. Algoritma ini bekerja dengan prinsip iteratif untuk memperbarui posisi centroid dan menentukan anggota klaster sampai hasilnya stabil. Salah satu kelebihan dari K-Means adalah kesederhanaannya yang membuatnya mudah digunakan dan efisien dalam menemukan Cluster dengan bentuk elips atau bola. Namun, K-Means juga memiliki kelemahan yaitu tidak efektif dalam menangani Cluster dengan ukuran atau bentuk yang tidak terdefinisi dengan jelas. Meskipun begitu, algoritma ini masih menjadi pilihan utama dalam banyak kasus Clustering karena kepraktisannya.

Algoritma K-Means Clustering diinisiasi dengan menentukan jumlah klaster. Dilanjutkan dengan menentukan titik centroid secara acak. Banyaknya centroid yang ditentukan sama dengan jumlah klaster. Untuk setiap titik-titik yang diberikan oleh data, tempatkan ke klaster yang terdekat. dalam hal ini, kedekatan titik ditentukan berdasarkan jarak. Jarak yang dapat digunakan adalah jarak Euclid

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

B. Tujuan Praktikum

I. Tujuan Instruksional Umum

Praktikum bertujuan untuk menerapkan teori Klasterisasi *K-Means* dalam pembelajaran mesin.

II. Tujuan Instruksional Khusus

1. Mahasiswa mampu menguasai konsep dasar Klasterisasi *K-Means.*
2. Mahasiswa mampu menyelesaikan studi kasus menggunakan metode Klasterisasi *K-Means.*
3. Mahasiswa mampu menganalisis hasil dari studi kasus menggunakan metode Klasterisasi *K-Means.*

C. Dataset dan bahasa pemrograman Python

C.1 Dataset dalam praktikum

Dalam praktikum modul 4 ini kita akan menggunakan satu dataset. Dataset merupakan sekumpulan data yang digunakan sebagai masukan bagi algoritma pembelajaran mesin, seperti K-Means Clustering. Dataset yang digunakan dalam contoh codingan dalam praktikum ini terdiri dari koordinat titik-titik dalam dua dimensi (X dan Y), yang menjadi fitur untuk mengelompokkan data. Setiap titik data memiliki nilai X dan Y yang akan digunakan oleh algoritma untuk menentukan klaster mana data tersebut akan dimasukkan.

1) Dataset Numerik

Titik	X	Y
A	1	3
B	3	3
C	4	3
D	5	3
E	1	2
F	4	2
G	1	1
H	2	1

Dataset ini terdiri dari delapan titik dengan dua fitur (X dan Y) yang menggambarkan posisi setiap titik dalam koordinat dua dimensi. Algoritma K-Means akan menggunakan data ini untuk menghitung jarak dari setiap titik ke centroid yang sudah

ditentukan atau yang diperbarui dalam proses iterasi, kemudian mengelompokkan titik-titik tersebut berdasarkan jarak terdekat ke centroid.

C.2 Bahasa Pemrograman Python

Pada praktikum kali ini, kita akan belajar dalam konteks K-Means Clustering, library yang akan digunakan dalam praktikum inisebagai berikut:

1) NumPy: Digunakan untuk menangani operasi matematis seperti manipulasi array dan komputasi numerik dengan efisiensi tinggi. Dalam contoh praktikum ini, NumPy digunakan untuk membuat array yang merepresentasikan dataset (koordinat X dan Y dari titik-titik data).

2) Pandas : library untuk pengolahan data berbasis tabel (DataFrame).

D. Implementasi Klasterisasi K-Means

Pada praktikum ini kita akan menggunakan Google colab yang dapat diakses menggunakan tautan https://colab.research.google.com/. Untuk dalam praktikum ini kita akan menyelesaikan sebuah studi kasus sebagai berikut.

Titik	X	Y
A	1	3
B	3	3
C	4	3
D	5	3
E	1	2
F	4	2
G	1	1
H	2	1

- Misalkan ditentukan bahwa dari data di atas, terdapat dua klaster.
- Inisiasi juga Centroid untuk klaster 1 adalah G(1,1), dan Centroid untuk klaster 2 adalah H(2,1)
- Untuk setiap titik data lain, bandingkan mana yang lebih dekat terhadap centroid yang tersedia

Maka dalam hal ini dapat kita selesaikan sebagai berikut.

1) Import pustaka atau library yang akan digunakan

```
#Import pustaka atau library yang akan digunakan
import numpy as np
import pandas as pd
```

2) Masukkan dataset yang akan digunakan.

```
#Masukkan dataset yang akan digunakan
# Data titik (X,Y)
data = {
    'A': [1, 3],
    'B': [3, 3],
    'C': [4, 3],
    'D': [5, 3],
    'E': [1, 2],
    'F': [4, 2],
    'G': [1, 1],
    'H': [2, 1]
}

df = pd.DataFrame(data)
```

3) Inisialisasi centroid awal

```
# Inisialisasi  centroid awal
centroid_1 = np.array([1, 1])  # G(1,1)
centroid_2 = np.array([2, 1])  # H(2,1)
```

4) Fungsi untuk menghitung jarak Euclidean

```python
# Fungsi untuk menghitung jarak Euclidean
def euclidean_distance(point, centroid):
    return np.sqrt(np.sum((point - centroid) ** 2))

# List untuk menyimpan hasil
results = []
```

5) Menentukan klaster untuk setiap titik berdasarkan jarak ke centroid

```python
# Menghitung jarak dan menentukan klaster untuk setiap titik
for point, coordinates in data.items():
    coordinates = np.array(coordinates)

    # Menghitung jarak ke masing-masing centroid
    distance_to_centroid_1 = euclidean_distance(coordinates, centroid_1)
    distance_to_centroid_2 = euclidean_distance(coordinates, centroid_2)

    # Menentukan klaster terdekat
    if distance_to_centroid_1 < distance_to_centroid_2:
        nearest_centroid = "C1"
    else:
        nearest_centroid = "C2"

    # Menyimpan hasil
    results.append([point, coordinates[0], coordinates[1], round(distance_to_centroid_1, 2), round(distance_to_centroid_2, 2), nearest_centroid])
```

6) Output hasil

```python
# Membuat dataframe untuk menampilkan tabel
df = pd.DataFrame(results, columns=['Titik', 'X', 'Y', 'Jarak C1', 'Jarak C2', 'Klaster'])

# Menampilkan tabel
print(df)
```

Perintah di atas akan menghasilkan output sebagai berikut:

```
Titik  X  Y  Jarak C1  Jarak C2 Klaster
    A  1  3      2.00      2.24      C1
    B  3  3      2.83      2.24      C2
    C  4  3      3.61      2.83      C2
    D  5  3      4.47      3.61      C2
    E  1  2      1.00      1.41      C1
    F  4  2      3.16      2.24      C2
    G  1  1      0.00      1.00      C1
    H  2  1      1.00      0.00      C2
```

Selanjutnya didapatkan klaster seperti diatas sebagai berikut.

Titik	X	Y	Jarak C1	Jarak C2	Klaster
A	1	3	2.00	2.24	C1
B	3	3	2.83	2.24	C2
C	4	3	3.61	2.83	C2
D	5	3	4.47	3.61	C2
E	1	2	1.00	1.41	C1
F	4	2	3.16	2.24	C2
G	1	1	0.00	1.00	C1
H	2	1	1.00	0.00	C2

- Dari klaster yang sudah diperoleh, dihitung ulang centroid dengan cara mencari rerata untuk setiap titik yang berada di dalam klaster.
- Klaster 1: ((1,1)+(1,2)+(1,3))/3 = (1,2)
- Klaster 2: ((3,3)+(4,3)+(5,3)+(4,2)+(2,1))/5 = (3.6, 2.4)
- Dengan centroid yang baru, terdapat perubahan anggota klaster yaitu klaster 1 : {A, E, G, H} dan klaster 2 : {B, C, D, F}

Maka dalam hal ini dapat kita selesaikan sebagai berikut.

1) Import pustaka atau library yang akan digunakan

```
#Import pustaka atau library yang akan digunakan
import numpy as np
import pandas as pd
```

2) Masukkan dataset yang akan digunakan.

```
#Masukkan dataset yang akan digunakan
# Data titik (X,Y)
data = {
    'A': [1, 3],
    'B': [3, 3],
    'C': [4, 3],
    'D': [5, 3],
    'E': [1, 2],
    'F': [4, 2],
    'G': [1, 1],
    'H': [2, 1]
}

df = pd.DataFrame(data)
```

3) Inisialisasi centroid awal

```
# Inisialisasi centroid baru
centroid_1 = np.array([1.2, 1.2])  # Centroid 1 baru (1.2, 1.2)
centroid_2 = np.array([3.6, 2.4])  # Centroid 2 baru (3.6, 2.4)
```

4) Fungsi untuk menghitung jarak Euclidean

```
# Fungsi untuk menghitung jarak Euclidean
def euclidean_distance(point, centroid):
    return np.sqrt(np.sum((point - centroid) ** 2))

# List untuk menyimpan hasil
results = []
```

5) Menentukan klaster untuk setiap titik berdasarkan jarak ke centroid

```python
# Menghitung jarak setiap titik ke centroid
for index, row in df.iterrows():
    point = row['Titik']
    coordinates = np.array([row['X'], row['Y']])

    distance_to_centroid_1 = euclidean_distance(coordinates, centroid_1)
    distance_to_centroid_2 = euclidean_distance(coordinates, centroid_2)

    # Menentukan klaster terdekat
    if distance_to_centroid_1 < distance_to_centroid_2:
        nearest_centroid = "C1"
    else:
        nearest_centroid = "C2"

    # Menyimpan hasil
    results.append([point, coordinates[0], coordinates[1], round(distance_to_centroid_1, 2), round(distance_to_centroid_2, 2), nearest_centroid])
```

6) Membuat Dataframe

```python
# Membuat dataframe untuk menampilkan tabel
df_results = pd.DataFrame(results, columns=['Titik', 'X', 'Y', 'Jarak C1', 'Jarak C2', 'Klaster'])

# Menampilkan tabel
print(df_results)
```

Perintah di atas akan menghasilkan output sebagai berikut:

```
Titik  X  Y  Jarak C1  Jarak C2 Klaster
    A  1  3      1.81      2.67      C1
    B  3  3      2.55      0.85      C2
    C  4  3      3.33      0.72      C2
    D  5  3      4.20      1.52      C2
    E  1  2      0.82      2.63      C1
    F  4  2      2.91      0.57      C2
    G  1  1      0.28      2.95      C1
    H  2  1      0.82      2.13      C1
```

7) Menghitung centroid baru berdasarkan klaster

```python
# Menghitung centroid baru berdasarkan klaster
klaster_1 = df_results[df_results['Klaster'] == 'C1']
klaster_2 = df_results[df_results['Klaster'] == 'C2']

centroid_1_new = np.array([klaster_1['X'].mean(), klaster_1['Y'].mean()])
centroid_2_new = np.array([klaster_2['X'].mean(), klaster_2['Y'].mean()])

print(f"Centroid baru Klaster 1: {centroid_1_new}")
print(f"Centroid baru Klaster 2: {centroid_2_new}")
```

Perintah di atas akan menghasilkan output sebagai berikut:

```
Centroid baru Klaster 1: [1.25 1.75]
Centroid baru Klaster 2: [4.   2.75]
```