Praktikum 9
Algoritma Clustering K-Means

Tujuan

Setelah mengikuti praktikum ini, mahasiswa diharapkan mampu:

1. Membentuk cluster dari sebuah dataset
2. Menampilkan visualisasi cluster yang terbentuk

Dasar Teori

Clustering merupakan proses pembentukan cluster atau kelompok dari sekumpulan data, dimana data-data yang berada pada satu cluster tertentu memiliki kemiripan yang tinggi, sedangkan data-data yang berada pada cluster yang berbeda memiliki kemiripan yang rendah.

Pengelompokan data pada proses clustering didasarkan pada suatu kriteria pengelompokkan, yaitu kemiripan data. Kemiripan data dapat diukur dengan beberapa cara, salah satunya adalah menggunakan ukuran jarak. Dengan menggunakan ukuran jarak, maka data yang berada dalam satu cluster memiliki jarak yang kecil, sedangkan

data pada cluster yang berbeda memiliki jarak yang besar. Ukuran jarak yang dapat digunakan pada clustering adalah *Euclidean distance*. Persamaan *Euclidean distance* adalah

$$d = \sqrt{\sum_{i}^{n} \left(x_i - y_i \right)^2}$$

Apabila data berupa kategori, maka perhitungan jarak dapat menggunakan Simple Matching Distance. Persamaan jarak Simple Matching Distance adalah

$$\delta(x, y) = \begin{cases} 0 & if \quad x = y \\ 1 & if \quad x \neq y \end{cases}$$

$$d_{sim}(x, y) = \sum_{j=1}^{d} \delta\left(x_j, y_j \right)$$

Simple Matching Distance biasa disebut juga dengan Hamming Distance. Definisi dan contoh dapat dipelajari pada link berikut.

K-Means merupakan algoritma clutering non-hirarkis yang bertujuan membagi sekumpulan data menjadi *k* cluster. Tahapan-tahapan dalam algoritma K-Means adalah:

1. Tentukan banyaknya cluster/kelompok
2. Alokasikan data ke dalam kelompok secara acak.
3. Hitung pusat cluster dari data yang ada di masing-masing cluster.
4. Alokasikan masing-masing data ke centroid terdekat.
5. Kembali ke langkah 3, jika masih ada data yang berpindah cluster atau ada perubahan nilai centroid di atas nilai ambang/Threshold yang ditentukan, atau jika perubahan nilai pada fungsi objektif masih di atas nilai ambang atau dengan batasan iterasi max.
6. Selesai

Praktikum

1. Import Data

Unduh dataset yang akan digunakan pada praktikum kali ini. Anda dapat menggunakan aplikasi wget untuk mendowload dataset dan menyimpannya dalam Google Colab. Jalankan cell di bawah ini untuk mengunduh dataset

```
! wget https://dataset-ppm.s3.amazonaws.com/iris.csv
```

Setelah dataset berhasil diunduh, langkah berikutnya adalah membaca dataset dengan memanfaatkan fungsi **readcsv** dari library **pandas**. Lakukan pembacaan berkas csv ke dalam dataframe dengan nama **data** menggunakan fungsi readcsv. Jangan lupa untuk melakukan import library pandas terlebih dahulu

```
import pandas as pd
import numpy as np
data = pd.read_csv('iris.csv')
```

Cek isi dataset Anda dengan menggunakan perintah head()

```
data.head()
```

2. Preprocessing

Sebelum dilakukan proses clustering, perlu dilakukan beberapa tahapan pemrosesan awal data. Tahapan pertama adalah memisahkan kelas data dari dataframe, karena algoritma K-Means tidak memerlukan informasi kelas. Pisahkan kelas data ke dalam sebuah variabel bernama **kelas**.

```
kelas = data.pop('species')
```

Pada praktikum kali ini, kita akan memvisualisasikan data hasil clustering menggunakan scatter plot. Scatter plot hanya dapat digambarkan dalam bentuk 2 dimensi, dengan demikian data yang digunakan juga hanya terdiri dari 2 fitur. Hapus fitur **petal_length** dan **petal_width** pada data, sehingga fitur yang digunakan adalah **sepal_length** dan **sepal_width**

```
del data['petal_length']
del data['petal_width']
```

Salah satu tahap pada algoritma K-Means adalah perhitungan jarang Euclidean. Agar tidak ada fitur yang mendominasi pada perhitungan jarak, maka perlu dilakukan normalisasi terlebih dahulu agar rentang nilai setiap fitur sama. Implementasikan metode normalisasi min-max menggunakan fungsi bernama **minmax** pada cell di bawah ini. (Catatan : Anda pernah membuat fungsi ini di Praktikum 2)

```
def minmax(df_input):
  list_fitur = df_input.columns[:-1]
  for fitur in list_fitur:
    max = df_input[fitur].max()
    min = df_input[fitur].min()
    df_input[fitur] = (df_input[fitur]-min)/(max-min)
  return df_input
```

Lakukan normalisasi data dan simpan hasilnya pada variabel bernama **data_normal**

```
data_normal = minmax(data)
```

Tampilkan isi dataframe **data_normal**

```
data_normal
```

3. **Visualisasi Data**

Visualisasi data dilakukan untuk memahami struktur dari data. Informasi ini dapat digunakan untuk mengetahui cara pembentukan cluster yang baik. Visualisasikan **data_normal** menggunakan scatter plot. Gunakan library matplotlib.pyplot

```
import matplotlib.pyplot as plt
warna =
{'Iris-setosa':'red','Iris-versicolor':'green','Iris-virginica':'blue'}
plt.scatter(data_normal['sepal_length'],data_normal['sepal_width'],c=kelas.
map(warna))
plt.plot()
```

Fungsi scatter pada matplotlib menerima input berupa nilai variabel x, nilai variabel y, dan informasi kelas yang akan digunakan untuk membedakan warna elemen scatter plot.

4. **K-Means Clustering**

Sebelum mengimplementasikan algoritma K-Means, perlu dilakukan implementasi perhitungan jarak. K-Means pada praktikum ini menggunakan perhitungan jarak Euclidean. Implementasikan perhitungan jarak Euclidean menggunakan fungsi bernama **euclidean** pada cell di bawah ini

```
def euclidean(data1,data2):
  jarak = np.square(data1-data2)
  jarak = np.sum(jarak)
  return np.sqrt(jarak)
```

Langkah selanjutnya adalah implementasi algoritma K-Means. Tahapan-tahapan dalam algoritma K-Means adalah:

1. Penentuan keanggotaan cluster secara acak
2. Hitung centroid berdasarkan data yang ada di masing-masing cluster

3. Alokasikan masing-masing data ke centroid terdekat
4. Kembali ke langkah 2 selama kondisi berhenti belum terpenuhi. Iterasi berhenti jika salah satu kondisi berikut terpenuhi:

    a. Tidak ada perubahan keanggotaan cluster
    b. Iterasi melebihi parameter *iter_max*

Implementasikan algoritma K-Means menggunakan fungsi bernama **kmeans** pada cell di bawah ini. Fungsi **kmeans** memiliki 3 parameter input, yaitu data, K, dan iter_max

```
from random import randint
import copy
def kmeans(data_input,k,iter_max=200):
  n_data = data_input.shape[0]
  n_fitur = data_input.shape[1]
  iter = 1
  cluster = np.array([randint(0,k-1) for i in range(n_data)])
  centroid = np.zeros((k,n_fitur))
  cluster_tidak_berubah = False
  while cluster_tidak_berubah==False and iter<iter_max:
    jarak = np.zeros((n_data,k))
    for i in range(k):
          centroid[i,:]=data_input.iloc[np.where(cluster==i)[0]].mean().tolist()
    for i in range(n_data):
      for j in range(k):
        jarak[i][j]=euclidean(data_input.iloc[i],centroid[j])
    cluster_baru = [np.argmin(jarak[i]) for i in range(n_data)]
    cluster_tidak_berubah = np.array_equal(cluster_baru,cluster)
    cluster = np.array(copy.deepcopy(cluster_baru))
    iter+=1
```

Pada fungsi tersebut, variabel **cluster** berisi keanggotan cluster dari masing-masing data. Variabel **cluster** awalnya bernilai acak dengan rentang 0 sampai dengan k-1. Variabel **centroid** merupakan centroid masing,masing cluster dengan struktur data berupa matriks berdimensi k x banyak fitur. Variabel **cluster_tidak_berubah** merupakan variabel Boolean yang menandakan apakah keanggotaan cluster yang terbentuk pada iterasi sekarang sama dengan keanggotaan cluster yang terbentuk pada iterasi sebelumnya.

Proses clustering terjadi di dalam looping while. Looping akan terus dilakukan selama masih terjadi perubahan keanggotaan cluster (cluster_tidak_berubah = False) dan iterasi kurang dari iterasi maksimum (iter<iter_max). Tahapan pertama dalam looping

adalah menghitung centroid, yang merupakan rata-rata data pada masing-masing cluster. Selanjutnya, jarak setiap data dengan setiap centroid dihitung dan disimpan pada variabel **jarak**. Keanggotaan cluster yang baru diperoleh dari indeks jarak dengan nilai minimum dari masing-masing data ke centroid dan disimpan pada variabel **cluster_baru**. Setelah itu, dilakukan perbandingan dengan variabel **cluster**, untuk mengecek apakah ada perubahan keanggotaan cluster pada interasi sebelumnya. Jika ada perubahan, variabel **cluster_tidak_berubah** berisi nilai **False**, sedangkan jika tidak ada perubahan maka variabel **cluster_tidak_berubah** berisi nilai **True**.

Lakukan proses clustering dengan memanggil fungsi **kmeans** dengan parameter input berupa **data_normal** dan k=3.

```
cluster = kmeans(data_normal,3)
```

Tampilkan keanggotaan cluster pada masing-masing data

```
print(cluster)
```

Tampilkan hasil clustering menggunakan scatter plot

```
import matplotlib.pyplot as plt
plt.scatter(data_normal['sepal_length'],data_normal['sepal_width'],c=cluster)
plt.plot()
```

Tugas

Pada tugas kali ini Anda mengidentifikasikan pasangan fitur apa yang menghasilkan cluster yang ideal pada K-Means menggunakan data Iris. Langkah yang harus Anda lakukan adalah:

1. Buatlah plot untuk setiap pasangan fitur yang mungkin.
2. LIhatlah plot yang terbentuk. Identifikasikan pasangan fitur apa yang dapat membedakan ketiga kelas dengan baik. Hal ini ditandai dengan plot yang memiliki warna-warna (kelas) yang saling berpisah atau tidak bercampur.
3. Lakukan clustering menggunakan K-Means dengan k=3, apakah cluster yang dihasilkan juga terpisah dengan baik?

Petunjuk pengerjaan soal:

1. Klik link berikut, pastikan Anda login menggunakan **akun student UB**. https://colab.research.google.com/drive/1G9Nq0z37tjFpz4yPj_R3dA7oa07CaH-_?usp=sharing

2. Klik tombol Copy to Drive

3. Beri nama file Praktikum 9 - Nama - NIM
4. Isilah cell yang kosong
5. Download file *.ipynb dengan cara klik **File -> Download .ipynb**
6. Kumpulkan file *.ipynb ke asisten