Laporan Praktikum

K-Means Clustering

Nama : Fauzia Nur Fitria

NIM : 24060118130118

DEPARTEMEN ILMU KOMPUTER/INFORMATIKA

FAKULTAS SAINS DAN MATEMATIKA

UNIVERSITAS DIPONEGORO

SEMARANG

2020

Bab I
Pendahuluan

A. Dasar Teori

K-Means Clustering adalah suatu metode penganalisaan data atau metode Data Mining yang melakukan proses pemodelan secara *unsupervised* dan merupakan salah satu metode yang melakukan pengelompokan data dengan sistem partisi. Metode dalam K-Means Clustering adalah mengelompokkan data yang ada ke dalam beberapa kelompok, dimana data dalam satu kelompok mempunyai karakteristik yang sama satu sama lainnya dan mempunyai karakteristik yang berbeda dengan data yang ada di dalam kelompok yang lain. Dengan kata lain, metode K-Means Clustering bertujuan untuk meminimalisasikan *objective function* yang di-set dalam proses *clustering* dengan cara meminimalkan variasi antar data yang ada di dalam suatu cluster dan memaksimalkan variasi dengan data yang ada di cluster lainnya.

Data *clustering* menggunakan metode K-Means Clustering ini secara umum dilakukan dengan algoritma dasar sebagai berikut:

1. Pilih K buah titik *centroid* secara acak.
2. Kelompokkan data sehingga terbentuk K buah *cluster* dengan titik *centroid* dari setiap *cluster* merupakan titik *centroid* yang telah dipilih sebelumnya.
3. Perbaharui nilai titik *centroid*.
4. Ulangi langkah 2 dan 3 sampai nilai dari titik *centroid* tidak lagi berubah.

B. Rumusan Masalah

1. Bagaimana cara melakukan *clustering* pada dataset iris?
2. Bagaimana cara melakukan evaluasi hasil *clustering* menggunakan *inertia* dan *silhouette coefficient*?

Bab II

Pembahasan

A. Melakukan *Clustering* Menggunakan Dataset Iris

1. *Import Database*

```
%matplotlib inline
from copy import deepcopy
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
```

Sebelum melakukan *clustering*, perlu dilakukan *import* dataset dengan meng-*import* numpy, pandas, dan pyplot. Numpy memiliki kegunaan untuk operasi vektor dan matriks. Fiturnya hampir sama dengan MATLAB dalam mengelola *array* dan *array* multidimensi. Numpy merupakan salah satu *library* yang digunakan oleh *library* lain seperti Scikit-Learn untuk keperluan analisis data.

Dengan menggunakan sistem dataframe, dapat memuat sebuah *file* ke dalam tabel virtual ala *spreadsheet* dengan menggunakan Pandas. Dengan menggunakan Pandas dapat mengolah suatu data dan mengolahnya seperti *join*, *distinct*, *group by*, agregasi, dan teknik seperti pada SQL. Hanya saja dilakukan pada tabel yang dimuat dari *file* ke RAM.

Fungsi dari matplotlib dalam data adalah untuk memvisualisasikan data dengan lebih indah dan rapi, sehingga data terlihat lebih berwarna tidak hanya warna hitam saja.

Kemudian fungsi *plot* untuk menampilkan data secara 2D atau 3D, sehingga dapat ditampilkan data yang telah diolah sesuai kebutuhan. Matplotlib pun terintegrasi dengan iPython Notebook atau Jupyter dimana kamu dapat membuat sebuah buku interaktif yang dapat diberi penjelasan dan kode yang disisipkan begitupun hasil *plotting*-nya.

Matplotlib adalah *library* paling banyak digunakan oleh *data science* untuk menyajikan datanya ke dalam visual yang lebih baik. *Source code* `plt.rcParams['figure.figsize']=(16,9)` berfungsi untuk mengatur gambar dengan tinggi 16 inchi dan lebar 9 inchi. Sedangkan *source code* `plt.style.use`

`('ggplot')` berfungsi untuk sistem *plotting* untuk membuat *plot* yang terlihat profesional, dengan cepat dengan kode minimal.

```
# Import dataset
url = "http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width']
data = pd.read_csv(url, names=names)
print(data.shape)
```

```
(150, 4)
```

Dataset yang digunakan diberikan nama label sesuai dengan variabel names dan data tersebut dibaca dan di print dimensinya. Dimensi dari dataset merupakan gambaran singkat mengenai banyaknya jumlah baris yang menunjukkan banyaknya sampel data dan jumlah kolom yang menunjukkan atribut data dari dataset terkait.

Untuk melihat 10 data pertama, digunakan *source code* seperti berikut :

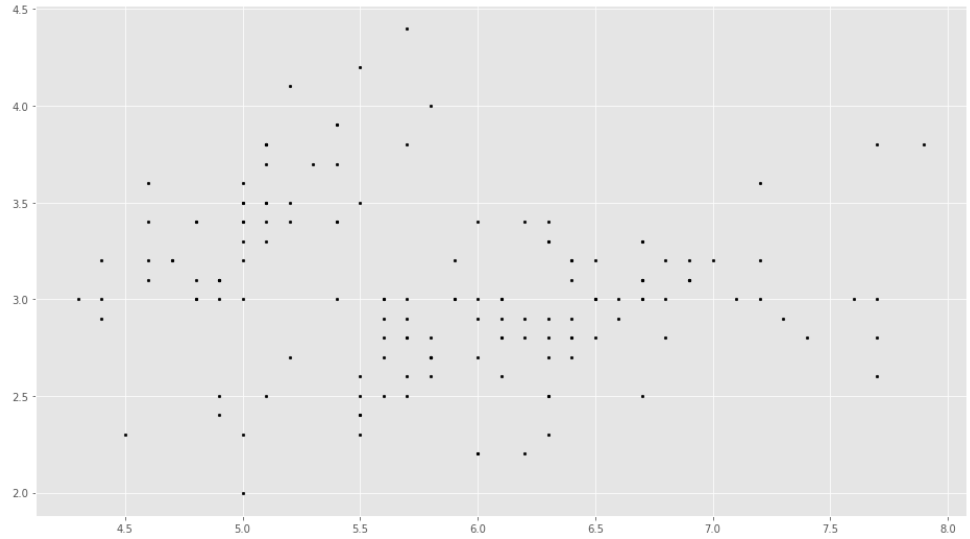
```
print(dataset.head(20))
```

```
    sepal-length  sepal-width  petal-length  petal-width        class
0            5.1          3.5           1.4          0.2  Iris-setosa
1            4.9          3.0           1.4          0.2  Iris-setosa
2            4.7          3.2           1.3          0.2  Iris-setosa
3            4.6          3.1           1.5          0.2  Iris-setosa
4            5.0          3.6           1.4          0.2  Iris-setosa
5            5.4          3.9           1.7          0.4  Iris-setosa
6            4.6          3.4           1.4          0.3  Iris-setosa
7            5.0          3.4           1.5          0.2  Iris-setosa
8            4.4          2.9           1.4          0.2  Iris-setosa
9            4.9          3.1           1.5          0.1  Iris-setosa
```

Kemudian untuk melakukan *plot* dataset dengan perintah berikut :

```
In [20]: # Plot dataset
         f1 = data['sepal-length'].values
         f2 = data['sepal-width'].values
         f3 = data['petal-length'].values
         f4 = data['petal-width'].values
         f5 = data['class'].values
         X = np.array(list(zip(f1, f2)))
         plt.scatter(f1, f2, c='black', s=7)
```

```
Out[20]: <matplotlib.collections.PathCollection at 0xe4b990b700>
```

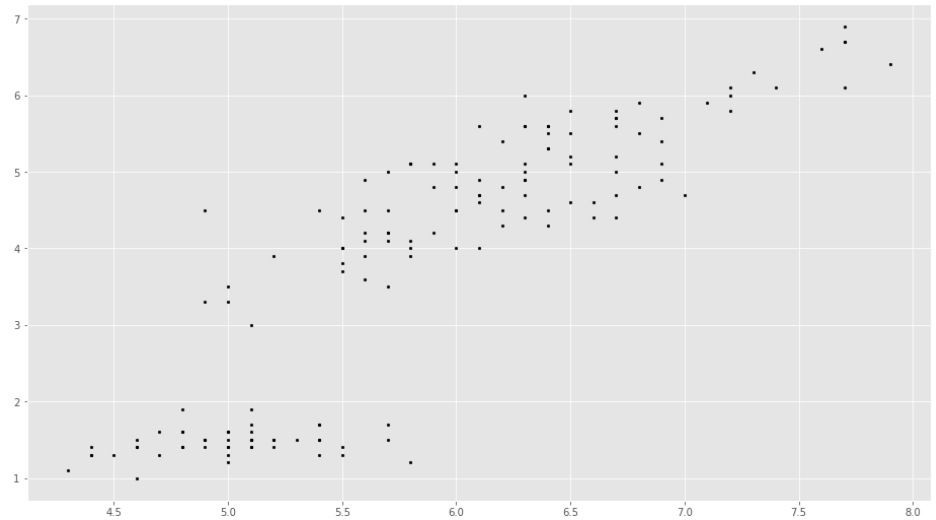


Dari perintah tersebut, hanya dapat terlihat dengan 2 titik values karena grafik yang akan ditampilkan hanya berbentuk 2 dimensi. Gambar diatas merupakan grafik dari titik f1 yaitu isi dari kolom sepal-length dan f2 yang berisi kolom pada sepal-width.

Kemudian, untuk menampilkan gambar grafik plot dataset dari titik f1 yaitu isi dari kolom sepal-length dan f3 yang berisi kolom petal-length

```
# Plot dataset
f1 = data['sepal-length'].values
f2 = data['sepal-width'].values
f3 = data['petal-length'].values
f4 = data['petal-width'].values
f5 = data['class'].values
X = np.array(list(zip(f1, f3)))
plt.scatter(f1, f3, c='black', s=7)
```

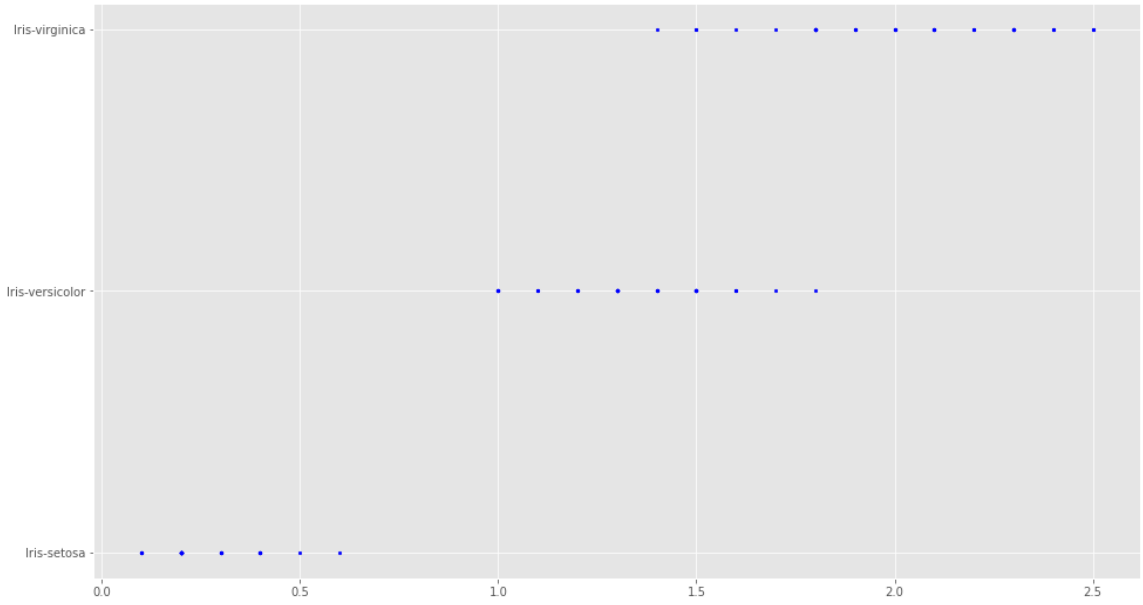
```
<matplotlib.collections.PathCollection at 0xe4adc7eeb0>
```



Kemudian, untuk menampilkan gambar grafik plot dataset dari titik f4 yaitu isi dari kolom petal-width dan f5 yang berisi kolom class

```
# Plot dataset
f1 = data['sepal-length'].values
f2 = data['sepal-width'].values
f3 = data['petal-length'].values
f4 = data['petal-width'].values
f5 = data['class'].values
X = np.array(list(zip(f4, f5)))
plt.scatter(f4, f5, c='blue', s=7)
```

```
<matplotlib.collections.PathCollection at 0xe4adb334c0>
```



2. Melakukan *Clustering*

Untuk melakukan *clustering* dengan K-Means clustering akan digunakan *library* sklearn.cluster. Lalu menentukan jumlah cluster = 3 serta *fitting input* data menggunakan fit(x), kemudian mendapatkan *cluster labels* dengan cara predict(x) dari data yang sudah ada.

```
from sklearn.cluster import KMeans
# Menentukan jumlah cluster
kmeans = KMeans(n_clusters=3)
# Fitting input data
kmeans = kmeans.fit(X)
# Mendapatkan cluster labels
labels = kmeans.predict(X)
# Mendapatkan nilai centroid
C = kmeans.cluster_centers_
# Mencetak nilai centroid
print(C)
```

```
[[5.77358491 2.69245283]
 [6.81276596 3.07446809]
 [5.006      3.418     ]]
```

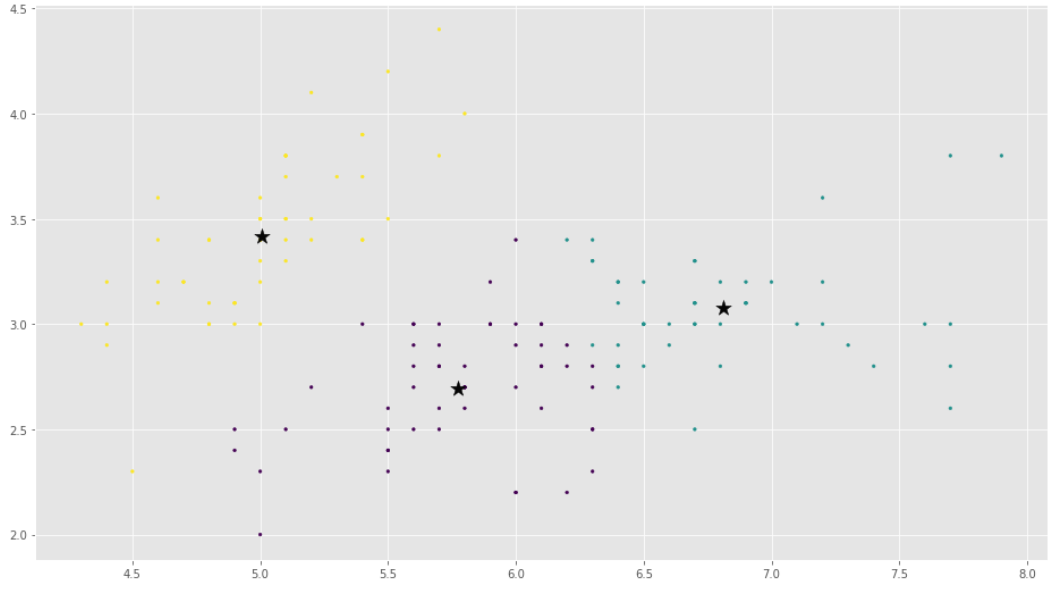
3. *Plot* Hasil *Clustering*

Untuk menampilkan *plot* hasil *clustering* dengan menggunakan perintah sebagai berikut:

```
In [29]: plt.scatter(X[:, 0], X[:, 1], s=7, c=labels)
         plt.scatter(C[:, 0], C[:, 1], marker='*', s=200, c='#050505')
```

```
Out[29]: <matplotlib.collections.PathCollection at 0xe4b9975fd0>
```

Lalu hasil grafik dari perintah diatas adalah sebagai berikut :



4. Melakukan *Clustering* Menggunakan *Generate* Dataset

Selanjutnya, dataset yang akan di-*cluster* di-*generate* secara otomatis menggunakan make_blobs.

a. Men-*generate* Dataset

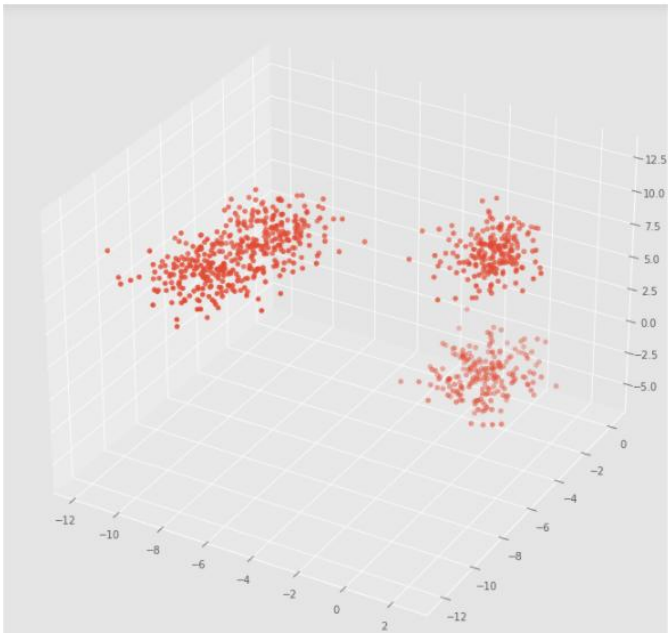
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
plt.rcParams['figure.figsize'] = (16, 9)
```

Sebelum itu, harus meng-*import library* terlebih dahulu seperti numpy, matplotlib.pyplot, Axes3D, KMeans, dan make_blobs serta mengatur gambar dengan tinggi 16 inchi dan lebar 9 inchi. Lalu setelah meng-*import library* maka lakukan perintah seperti berikut ini:

```
# Men-generate dataset yang terkelompok dalam 4 cluster
X, y = make_blobs(n_samples=800, n_features=3, centers=4)
fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(X[:, 0], X[:, 1], X[:, 2])
```

```
Out[31]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0xe4b9f99b50>
```

Perintah diatas akan men-*generate* dataset yang terkelompok dalam 4 *cluster*. Berikut adalah *output* dari perintah diatas:



b. Melakukan *Clustering*

Selanjutnya, akan dilakukan *clustering* pada dataset dengan menggunakan KMeans Clustering dengan melakukan perintah berikut :

```
# Initializing KMeans
kmeans = KMeans(n_clusters=4)
# Fitting with inputs
kmeans = kmeans.fit(X)
# Predicting the clusters
labels = kmeans.predict(X)
# Getting the cluster centers
C = kmeans.cluster_centers_
```

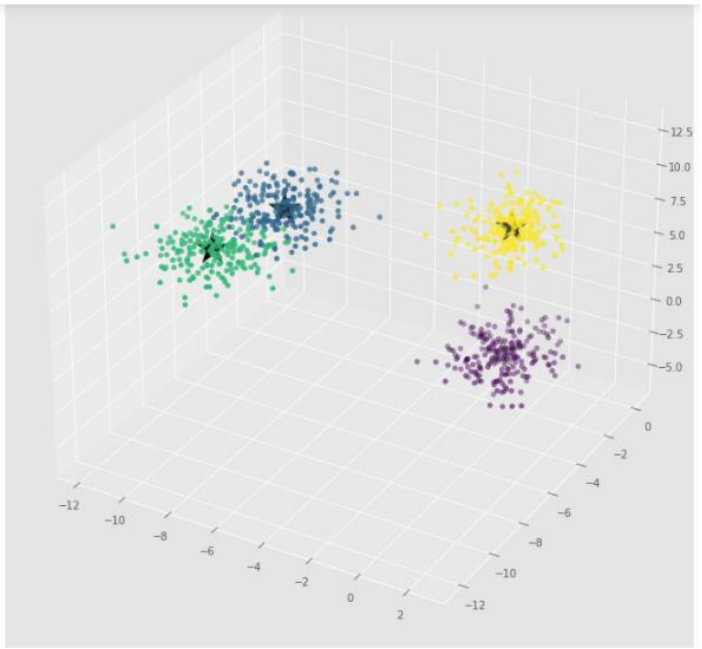
Jumlah *clustering* menggunakan kMeans adalah 4 lalu di lakukan *fitting* dan di-*predicting cluster*-nya. Lalu *getting the cluster centers* menggunakan fungsi yang sudah ada.

c. *Plot* Hasil *Clustering*

Selanjutnya *plot* hasil *clustering* dengan menggunakan perintah berikut ini :

```
In [33]: fig = plt.figure()
         ax = Axes3D(fig)
         ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=y)
         ax.scatter(C[:, 0], C[:, 1], C[:, 2], marker='*', c='#050505', s=1000)
Out[33]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0xe4bc2494c0>
```

Setelah melakukan perintah di atas maka akan menampilkan *output* grafik 3 dimensi sebagai berikut:



B. Melakukan Evaluasi Hasil *Clustering* Menggunakan *Inertia* dan *Silhouette Coefficient*

Pada contoh sebelumnya, telah diketahui jumlah k (*cluster*) yang tepat untuk melakukan *clustering*. Namun, pada kenyataannya tidak selalu ditemukan jumlah *cluster* yang tepat pada dataset yang akan di *cluster*. Oleh karena itu, perlu dilakukan evaluasi hasil *clustering* menggunakan beberapa kemungkinan nilai k dan menggunakan *metric* yang bersesuaian untuk *clustering*. Contohnya, akan dilakukan evaluasi hasil *clustering* pada contoh pertama.

```
for k in range (1, 10):
 # Menentukan jumlah cluster
 kmeans = KMeans(n_clusters=k, random_state=1)
 # Fitting input data
 kmeans = kmeans.fit(X)
 # Mendapatkan cluster labels
 labels = kmeans.predict(X)
 # Menghitung jumlahan jarak antara setiap sampel dengan cluster centroid-nya (SSE)
 interia = kmeans.inertia_
 print("k:",k, " cost:", interia)
```

```
k: 1  cost: 40528.62442971582
k: 2  cost: 14301.635258100045
k: 3  cost: 3219.544739496614
k: 4  cost: 2295.673833270753
k: 5  cost: 2121.31383059987
k: 6  cost: 1963.734892195495
k: 7  cost: 1826.175267465443
k: 8  cost: 1676.0429334162575
k: 9  cost: 1593.5928602003344
```

Dilakukan perulangan untuk menentukan jumlah cluster, melakukan *fitting* input data, mendapatkan *cluster* label, dan menghitung jumlahan jarak antar setiap *sampel* dengan *cluster centroid*-nya (SSE). Lalu untuk mencari evaluasi hasil *cluster* dengan menggunakan *silhouette coefficient* seperti berikut:

```
In [38]: from sklearn.metrics.cluster import silhouette_score
         silhouette_score(X, labels)

Out[38]: 0.2344017022682533
```

Bab III
Kesimpulan

A. Kesimpulan

Pada praktikum kali ini, dapat diketahui cara melakukan *clustering* menggunakan sebuah dataset dan cara melakukan *clustering* dengan dataset yang di-*generate* secara otomatis menggunakan make_blobs. Salah satu algoritma *unsupervised learning* atau *clustering* yang digunakan dalam praktikum ini adalah K-Means Clustering.

K-Means Clustering merupakan algoritma yang mengelompokkan data menjadi k *cluster* berdasarkan jarak terdekatnya dengan *centroid* dari masing-masing *cluster*.