BAB I

PENDAHULUAN

1.1 Dasar Teori

Laporan praktikum ini akan membahas mengenai salah satu algoritma dalam unsupervised learning atau clustering, yaitu k-means clustering. K-means clustering merupakan algoritma yang mengelompokkan data menjadi K cluster berdasarkan jarak terdekatnya dengan centroid masing-masing cluster.

K-Means adalah suatu metode penganalisaan data atau metode Data Mining yang melakukan proses pemodelan tanpa supervisi (unsupervised) dan merupakan salah satu metode yang melakukan pengelompokan data dengan sistem partisi. Metode k-means berusaha 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 ini berusaha untuk meminimalkan variasi antar data yang ada di dalam suatu cluster dan memaksimalkan variasi dengan data yang ada di cluster lainnya.

Pada laporan praktikum ini akan dicontohkan cara melakukan clustering menggunakan sebuah dataset dan dataset yang di-generate secara otomats menggunakan make_blobs.

1.2 Permasalahan

1.2.1 Lakukanlah clustering menggunakan dataset iris seperti yang digunakan pada praktikum sebelumnya!

1.2.2 Lakukan evaluasi hasil clustering menggunakan inertia dan silhouette coefficient!

1.3 Tujuan

1.3.1 Mahasiswa mampu melakukan clustering menggunakan dataset iris seperti yang digunakan pada praktikum sebelumnya.

1.3.2 Mahasiswa mampu melakukan evaluasi hasil clustering menggunakan inertia dan silhouette coefficient.

BAB II

PEMBAHASAN

2.1. Lakukanlah clustering menggunakan dataset iris seperti yang digunakan pada praktikum sebelumnya

Dataset yang akan digunakan adalah dataset klasifikasi Bunga Iris. Dataset ini berisi 150 pengamatan bunga Iris. Terdapat empat kolom pengukuran bunga dalam centimeter. Kolom kelima adalah spesies bunga yang diamati. Dataset ini merupakan dataset yang paling umum digunakan, sehingga sering disebut sebagai dataset "hello word" dalam *machine learning*. Informasi detail mengenai dataset tersebut dapat diakses melalui https://en.wikipedia.org/wiki/Iris_flower_data_set.

2.1.1 Import Dataset

```
%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 kamu 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.

Data yang kita olah tentu tidak elok apabila ditampilkan begitu saja dengan tabel hitam saja kepada investor atau manajemen. Bila ditampilkan dengan sejumlah grafik berwarna pasti mereka akan lebih tertarik melihatnya. Matplotlib membantu kamu untuk memvisualisasikan data dengan lebih indah dan rapi.

Ada *plot* untuk menampilkan data secara 2D atau 3D. Sehingga kamu dapat menampilkan data yang telah kamu olah 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 *plottingnya*.

Matplotlib adalah *library* paling banyak digunakan oleh *data science* untuk menyajikan datanya ke dalam visual yang lebih baik. `plt.rcParams['figure.figsize']=(16,9)` berguna mengatur gambar dengan tinggi 16 inchi dan lebar 9 inchi. Sedangkan `plt.style.use ('ggplot')` berguna untuk sistem plotting untuk membuat plot yang terlihat profesional, dengan cepat dengan kode minimal.

```
# Import dataset
# Sesuaikan dengan lokasi file iris.data di local komputer
names = ['sepal-length', 'sepal-width', 'petal-length','petal-width', 'class']
data = pd.read_csv('iris.data', names=names)
print(data.shape)
```

```
(150, 5)
```

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.

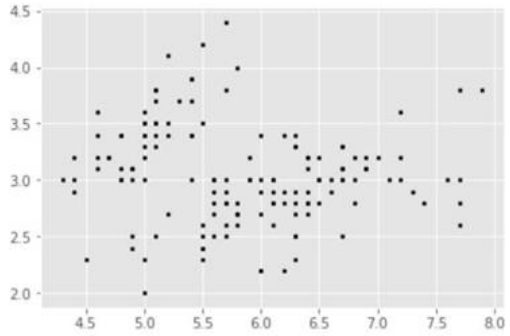
Untuk melihat 10 data pertama yang ada pada dataset dengan perintah berikut:

```
data.head(10)
```

	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

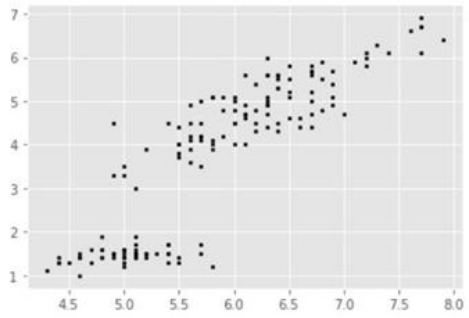
Melakukan plot dataset dengan perintah berikut:

```
In [4]: # 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[4]: <matplotlib.collections.PathCollection at 0x27b13e184e0>
```



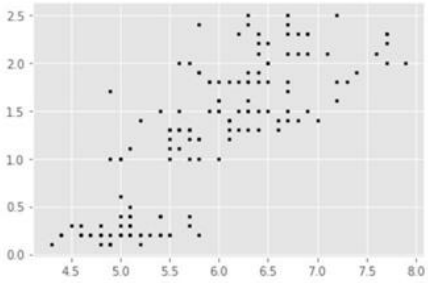
Hanya dapat melihat dengan 2 titik values karena grafik yang akan ditampilkan hanya berbentuk 2 dimensi. Gambar diatas merupakan grafik dari titik f1 yaitu isi dari kolom sepal-length dan f2 yang beisi kolom pada sepal-width.

```
In [7]: # 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)
Out[7]: <matplotlib.collections.PathCollection at 0x27b13e91dd8>
```



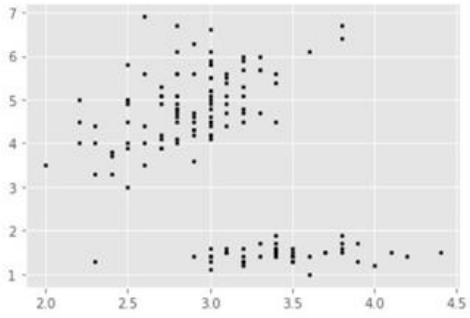
Gambar diatas merupakan graik dari titik f1 yaitu isi dari kolom sepal-length dan f3 yaitu isi dari kolom petal-length.

```
In [8]: # 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, f4)))
        plt.scatter(f1, f4, c='black', s=7)
Out[8]: <matplotlib.collections.PathCollection at 0x27b141d2940>
```



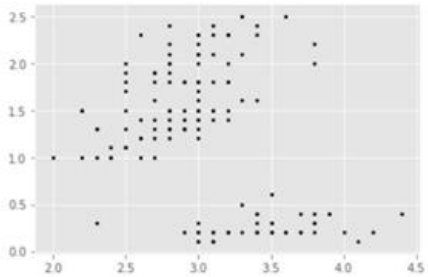
Gambar diatas merupakan graik dari titik f1 yaitu isi dari kolom sepal-length dan f4 yaitu isi dari kolom petal-width.

```
In [9]: # 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(f2, f3)))
        plt.scatter(f2, f3, c='black', s=7)
Out[9]: <matplotlib.collections.PathCollection at 0x27b141bd518>
```



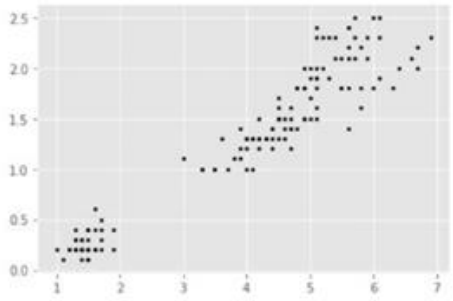
Gambar diatas merupakan graik dari titik f2 yaitu isi dari kolom sepal-width dan f3 yaitu isi dari kolom petal-length.

```
In [10]: # 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(f2, f4)))
         plt.scatter(f2, f4, c='black', s=7)
Out[10]: <matplotlib.collections.PathCollection at 0x27b1421a080>
```



Gambar diatas merupakan graik dari titik f2 yaitu isi dari kolom sepal-width dan f4 yaitu isi dari kolom petal-width.

```
In [11]: # 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(f3, f4)))
         plt.scatter(f3, f4, c='black', s=7)
Out[11]: <matplotlib.collections.PathCollection at 0x27b141bd470>
```



Gambar diatas merupakan graik dari titik f3 yaitu isi dari kolom petal-length dan f4 yaitu isi dari kolom petal-width.

2.1.2 Melakukan Clustering

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

```
In [12]: 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)

[[4.26923077 1.34230769]
 [1.464      0.244     ]
 [5.59583333 2.0375    ]]
```

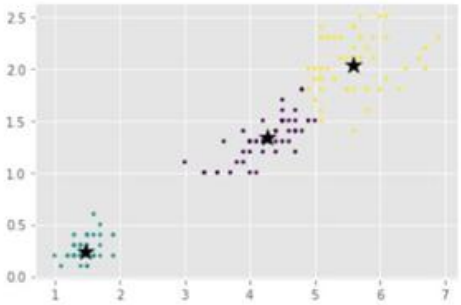
2.1.3 Plot hasil clustering

Untuk menampilkan plot hasil clustering dengan menggunakan perintah sebagai berikut:

```
In [13]: plt.scatter(X[:, 0], X[:, 1], s=7, c=labels)
         plt.scatter(C[:, 0], C[:, 1], marker='*', s=200, c='#050505')
```

Jika menggunakan perintah diatas maka akan menampilkan gambar sebagai berikut:

```
Out[13]: <matplotlib.collections.PathCollection at 0x27b16c84048>
```



2.1.4 Melakukan Clustering Menggunakan Generate Dataset

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

2.1.4.1 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)
```

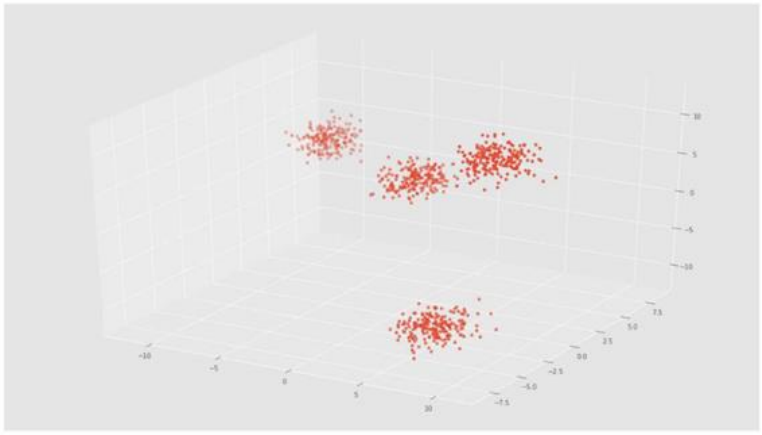
Sebelum itu, kita harus meng-import library seperti numpy, matplotlib.pyplot, Axes3D, KMeans, dan make_blobs. Serta mengatur gambar dengan tinggi 16 inchi dan lebar 9.

Lalu setelah meng-import library maka lakukan perintah seperti berikut ini:

```
In [15]: # 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])
```

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

```
Out[15]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x27b16cbf940>
```



2.1.4.2 Melakukan clustering

Selanjutnya, kita akan melakukan clustering pada dataset dengan menggunakan KMeans Clustering. Lalukanlah perintah berikut ini:

```
In [16]: # 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 fitting dan di predicting clusternya. Lalu getting the cluster centers menggunakan fungsi yang sudah ada.

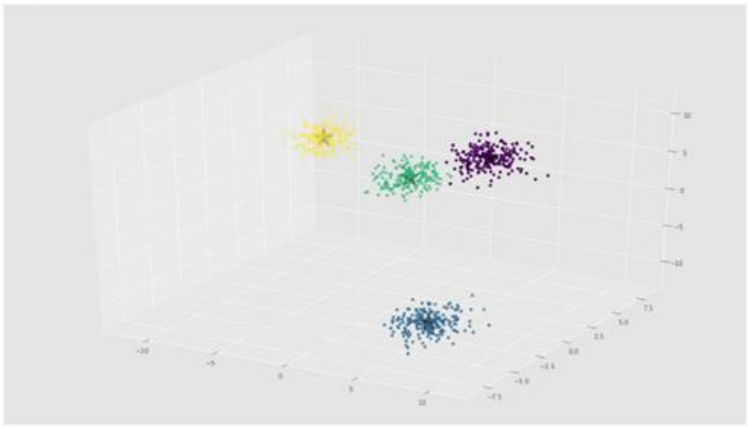
2.1.4.3 Plot hasil clustering

Selanjutnya plot hasil clustering dengan menggunakan perintah berikut ini:

```
In [17]: 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)
```

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

```
Out[17]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x27b16e4ab00>
```



2.2. Lakukan evaluasi hasil clustering menggunakan inertia dan silhouette coefficient

Pada kedua contoh di atas, kita telah mengetahui jumlah k (cluster) yang tepat untuk melakukan clustering. Pada kasus di dunia nyata, kita mungkin tidak mengetahui dengan tepat berapa jumlah cluster yang tepat untuk dataset yang akan kita cluster. Untuk itu, kita perlu melakukan evaluasi hasil clustering menggunakan beberapa kemungkinan nilai k dan menggunakan metric yang bersesuaian untuk clustering. Sebagai contoh kita akan mengevaluasi hasil clustering pada contoh yang pertama.

```
In [18]: 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: 85913.04492896626
k:  2  cost: 39161.65830010284
k:  3  cost: 10562.663939723905
k:  4  cost: 2419.578999043244
k:  5  cost: 2265.8794255556836
k:  6  cost: 2101.999686654821
k:  7  cost: 1944.4112165401432
k:  8  cost: 1802.9135742392311
k:  9  cost: 1711.7435017401704
```

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

```
In [19]: from sklearn.metrics.cluster import silhouette_score
         silhouette_score(X, labels)

Out[19]: 0.24139218360242126
```

BAB III

PENUTUP

3.1 Kesimpulan

Pada kesimpulan ini, kita dapat melakukan kegiatan praktikum ketiga mengenai salah satu algoritma dalam unsupervised learning atau clustering, yaitu k-means clustering dan cara melakukan clustering menggunakan sebuah dataset serta dataset yang di-generate secara otomats menggunakan make_blobs. K-means clustering merupakan algoritma yang mengelompokkan data menjadi K cluster berdasarkan jarak terdekatnya dengan centroid masing-masing cluster.

DAFTAR PUSTAKA

Modul Praktikum Pembelajaran Mesin (*Machine Learning*): Pertemuan 3 K-Means Clustering.

Agusta, Yudi. 2015. *K-Means.* https://yudiagusta.wordpress.com/k-means/. Diakses pada 15 November 2019.