ML_Prak03_2406011201401 54

November 15, 2022

1 PRAKTIKUM MACHINE LEARNING

2 K-MEANS CLUSTERING AND ELBOW METHOD

2.1 Nama: Anisatul Marifah

2.2 NIM: 2406120140154

2.3 Lab: B1

2.3.1 import library dan dataset

[ ]:
```
#import berbagai library yang dibutuhkan dalam proses clustering
%matplotlib inline
from copy import deepcopy
from matplotlib import pyplot as plt

import numpy as np
import pandas as pd
import seaborn as sns

from sklearn.cluster import KMeans

plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('seaborn-whitegrid')
pal = ["#EA047E","#3F0071", "#1746A2","#377D71"]
```

2.3.2 import dataset

dataset yang akan digunakan pada praktikum ini adalah Iris Dataset dari repository UCI Machine Learning. Dataset ini terdiri dari empat fitur yaitu Sepal Length, Sepal Width, Petal Length, dan Petal Width.

[ ]:
```
#import dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
```

```
names = ['sepal-l', 'sepal-w', 'petal-l', 'petal-w', 'class']
data = pd.read_csv(url, names=names)
```

[ ]:
```
#menentukan dimensi dari dataset
print (data.shape)
```

```
(150, 5)
```

[ ]:
```
#melihat 20 baris dari data teratas
print (data.head(20))
```

3 clustering Iris Dataset Fitur 1 dan 2

3.0.1 memilih fitur

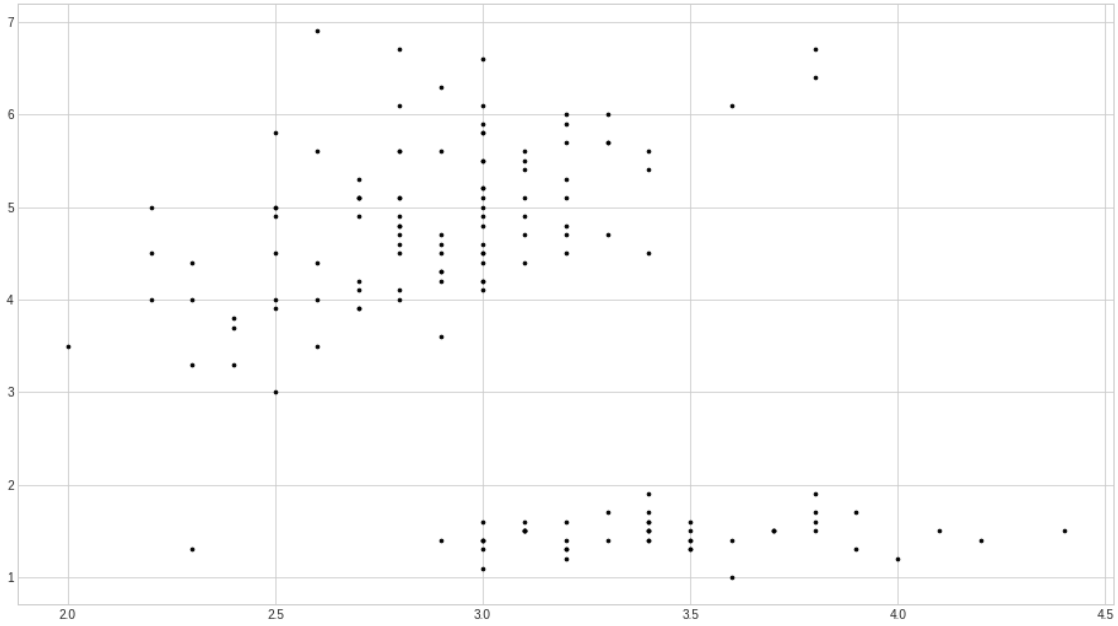
Pada bagian ini dilakukan pemilihan fitur, pada soal yang diminta adalah fitur 1 (sepal-w) dan 2 (petal-l).

Kemudian dapat dilihat scatter plot yang dihasilkan oleh dataset.

[ ]:
```
#plot dataset
f1 = data['sepal-w'].values
f2 = data['petal-l'].values

X = np.array(list(zip(f1,f2)))
plt.scatter(f1, f2, c='black', s=7)
```

[ ]:
```
<matplotlib.collections.PathCollection at 0x7fbcc40b8b90>
```



3.0.2 menentukan cluster

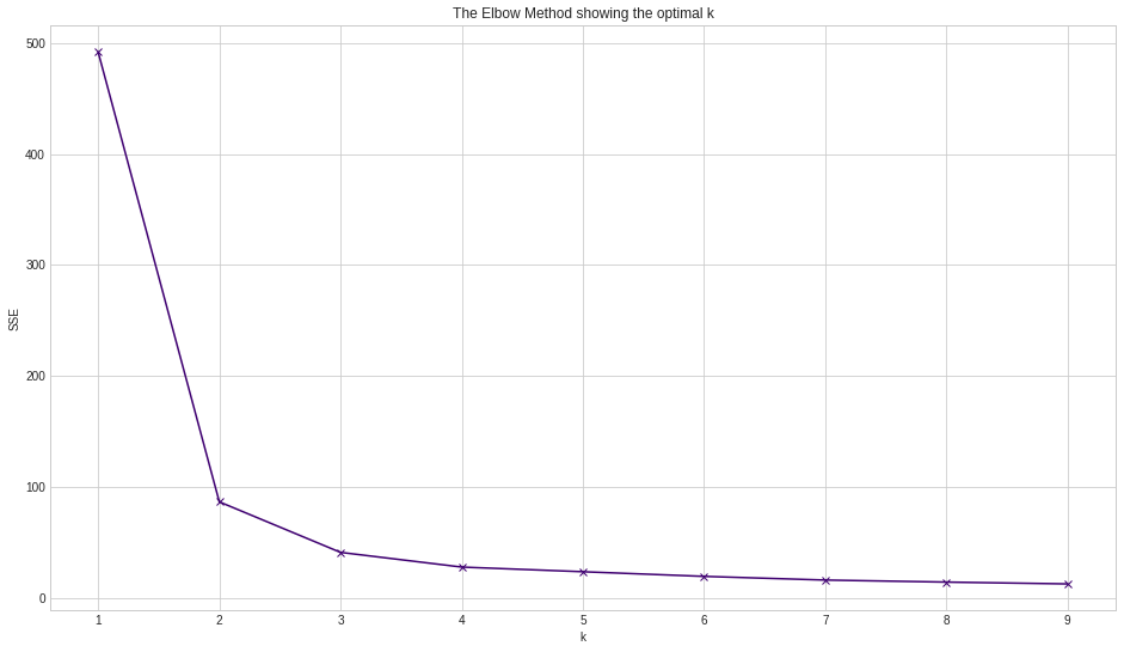
Selanjutnya yaitu menentukan cluster dengan melakukan perhitungan Sum of Squared Error untuk menentukan jumlah cluster terbaik yang nantinya akan digunakan. Nilai inertia tersebut akan disimpan dalam array SSE.

[ ]:
```
#menentukan nilai yang tepat untuk cluster
X = np.array(list(zip(f1, f2))).reshape(len(f2), 2)
SSE = []
K = range(1,10)
for k in K:
  kmeanModel = KMeans(n_clusters = k, init = 'k-means++', random_state = 0)
  kmeanModel.fit(X)
  SSE.append(kmeanModel.inertia_)
```

3.0.3 menentukan jumlah K pada elbow method untuk clustering dataset iris sesuai fitur 1, 2

dari perhitungan SSE di atas, tiap nilai di plot kan ke dalam bentuk grafik di bawah untuk membentuk elbow graph

[ ]:
```
#plot the elbow
plt.plot(K, SSE,'bx-', color = '#3F0071')
plt.xlabel('k')
plt.ylabel('SSE')
plt.title ('The Elbow Method showing the optimal k')
plt.show()
```



3.0.4 melakukan clustering

Untuk melakukan clustering dengan K-Means clustering kita akan menggunakan library sklearn.cluster.

[ ]:
```
#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)
```

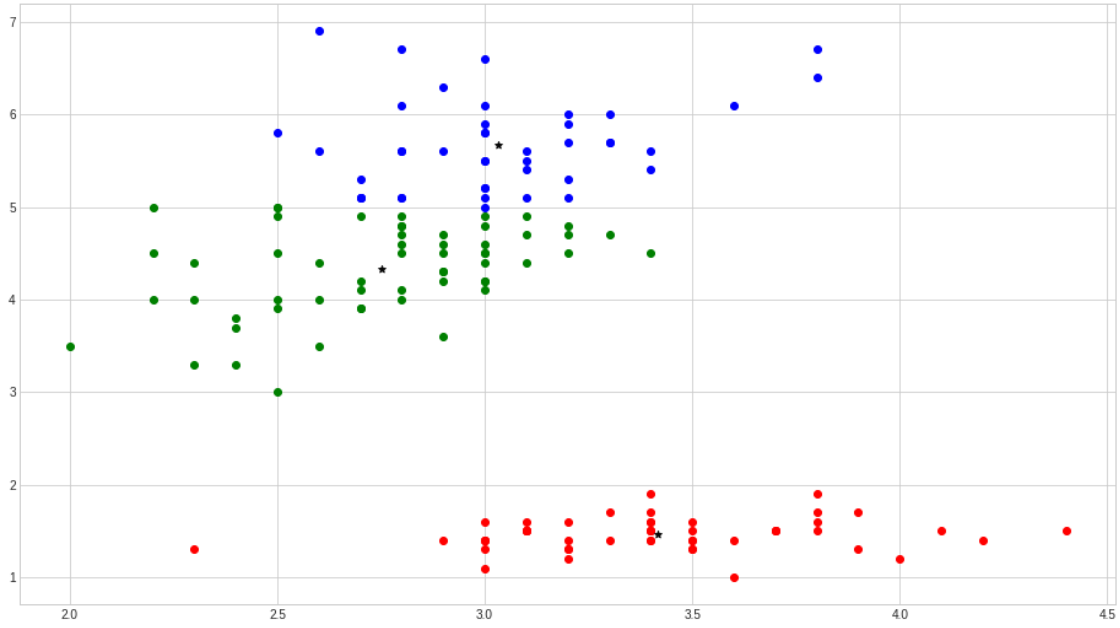
```
[[3.418      1.464     ]
 [3.03255814 5.67209302]
 [2.75087719 4.32807018]]
```

3.1 Visualisasi

```
[ ]: kmeansmodel = KMeans(n_clusters= 3, init='k-means++', random_state=0)
y_kmeans= kmeansmodel.fit_predict(X)

X = np.array(X)
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], c = 'red')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], c = 'blue')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], c = 'green')
plt.scatter(C[:, 0], C[:, 1], marker='*', c='#050505')
```

```
[ ]: <matplotlib.collections.PathCollection at 0x7f407287ead0>
```



3.2 Evaluasi

evaluasi hasil clustering menggunakan beberapa kemungkinan nilai k dan menggunakan metric yang besesuaian untuk clustering menggunakan Inertia dan Silhouette Coefficient.

```
[ ]: #inertia
for k in range (1,10):
  #menentukan jumlah clsuter
  kmeans = KMeans(n_clusters=k, random_state = 0)
  #fitting input data
  kmeans = kmeans.fit(X)
  #mendapatkan cluster labels
  labels = kmeans.predict(X)
```

```
  #menghitung jumlahan jarak antara setiap sampel dg cluster
  inertia = kmeans.inertia_
  print ("k:", k, "cost:", inertia)
  print("")
```

```
k: 1 cost: 491.8763333333334

k: 2 cost: 86.35692216280452

k: 3 cost: 40.80747409220726

k: 4 cost: 27.55509523809524

k: 5 cost: 23.36170673323093

k: 6 cost: 19.167252389373086

k: 7 cost: 15.845469144648455

k: 8 cost: 14.012562244291662

k: 9 cost: 12.38164141414142
```

3.2.1 silhouette Coefficient

masukan untuk evaluasi hasil clustering menggunakan solhouette coefficient

nilai di bawah adalah nilai silhouette dari fitur 1 dan 2 di mana jika nilai yang mendekati 1 maka nilai cluster adalah benar

Hasil pada nilai silhouette coeffisient sebesar adalah 0.593 dan lebih mendekati 1

[ ]:
```
#Evaluasi hasil cluster menggunakan silhouette coefficient
from sklearn.metrics.cluster import silhouette_score
silhouette_score(X, labels)
# silhouette score yang baik adalah yang mendekati 1
```

[ ]:
```
0.5930654335408364
```

3.3 clustering dataset iris sesuai fitur 2 dan 3

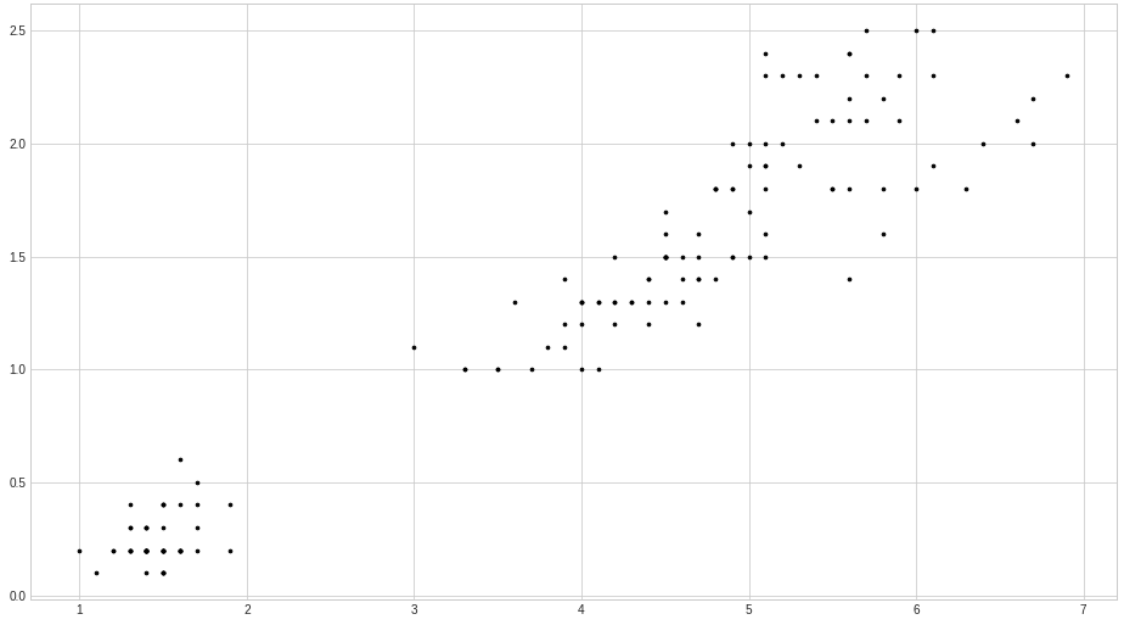
3.3.1 memilih fitur

Pada bagian ini dilakukan pemilihan fitur, pada soal yang diminta adalah fitur 2 (petal-l) dan 3 (petal-w).

Kemudian dapat dilihat scatter plot yang dihasilkan oleh dataset.

```
[ ]: f2 = data ['petal-l'].values
     f3 = data['petal-w'].values
     X = np.array(list(zip(f2, f3)))
     plt.scatter(f2, f3, c = 'black', s=7)
```

```
[ ]: <matplotlib.collections.PathCollection at 0x7fbcc9192a50>
```



3.3.2 menentukan cluster

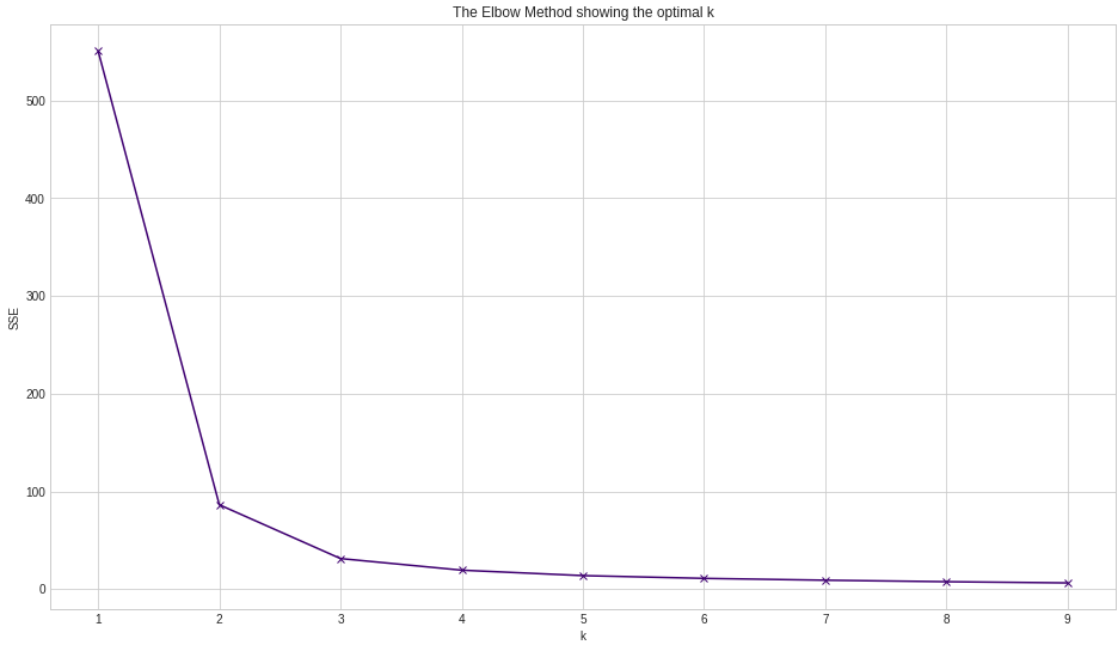
Selanjutnya yaitu menentukan cluster dengan melakukan perhitungan Sum of Squared Error untuk menentukan jumlah cluster terbaik yang nantinya akan digunakan. Nilai inertia tersebut akan disimpan dalam array SSE.

```
[ ]: #menentukan nilai yang tepat untuk cluster
     X2 = np.array(list(zip(f2, f3))).reshape(len(f3), 2)
     SSE = []
     K = range(1,10)
     for k in K:
       kmeanModel = KMeans(n_clusters = k, init = 'k-means++', random_state = 0)
       kmeanModel.fit(X)
       SSE.append(kmeanModel.inertia_)
```

3.3.3 menentukan jumlah K pada elbow method untuk clustering dataset iris sesuai fitur 2 dan 3

dari perhitungan SSE di atas, tiap nilai di plot kan ke dalam bentuk grafik di bawah untuk membentuk elbow graph

[ ]:
```
#plot the elbow
plt.plot(K, SSE,'bx-', color = '#3F0071')
plt.xlabel('k')
plt.ylabel('SSE')
plt.title ('The Elbow Method showing the optimal k')
plt.show()
```



3.3.4 melakukan clustering

Untuk melakukan clustering dengan K-Means clustering kita akan menggunakan library sklearn.cluster.

[ ]:
```
#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
Ce = kmeans.cluster_centers_
```

```
#mencetak nilai centroid
print(Ce)
```

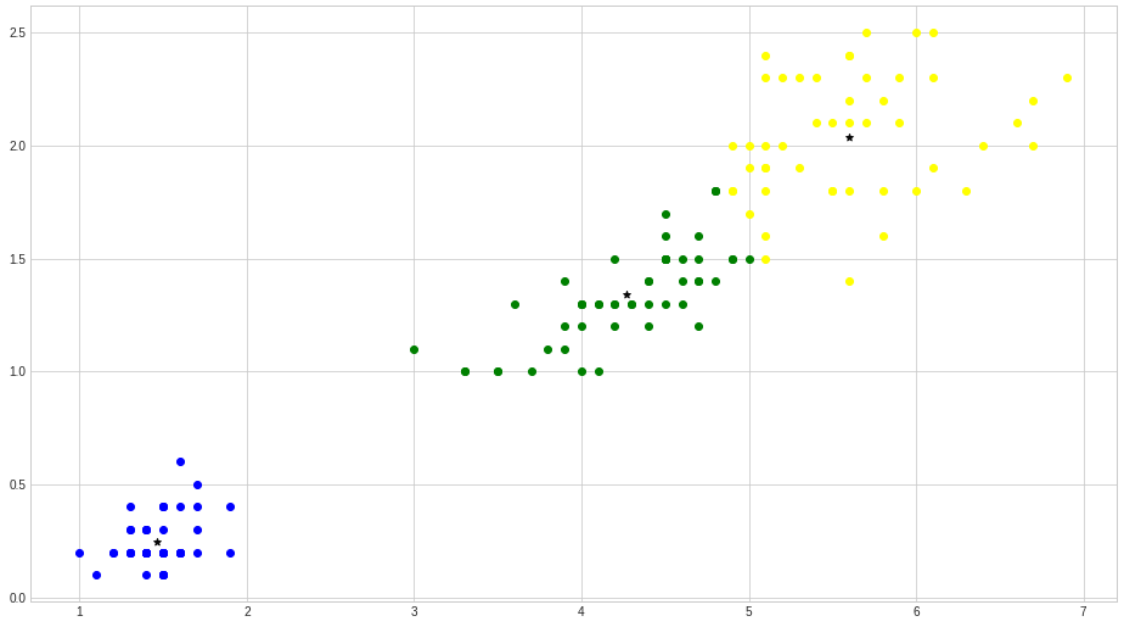
```
[[5.59583333 2.0375    ]
 [1.464      0.244     ]
 [4.26923077 1.34230769]]
```

3.4 Visualisasi

[ ]:
```
kmeansmodel = KMeans(n_clusters= 3, init='k-means++', random_state=0)
y_kmeans= kmeansmodel.fit_predict(X)

X2 = np.array(X2)
plt.scatter(X2[y_kmeans == 0, 0], X2[y_kmeans == 0, 1], c = 'blue')
plt.scatter(X2[y_kmeans == 1, 0], X2[y_kmeans == 1, 1], c = 'yellow')
plt.scatter(X2[y_kmeans == 2, 0], X2[y_kmeans == 2, 1], c = 'green')
plt.scatter(Ce[:, 0], Ce[:, 1], marker='*', c='#050505')
```

[ ]:
```
<matplotlib.collections.PathCollection at 0x7fbcc9141a90>
```



3.5 Evaluasi

di bawah ini merupakan nilai inertia dan silhouette coefficient

[ ]:
```
#inertia
for k in range (1,10):
  #menentukan jumlah clsuter
  kmeans = KMeans(n_clusters=k, random_state = 0)
  #fitting input data
  kmeans = kmeans.fit(X2)
  #mendapatkan cluster labels
  labels = kmeans.predict(X2)
  #menghitung jumlahan jarak antara setiap sampel dg cluster
  inertia = kmeans.inertia_
  print ("k:", k, "cost:", inertia)
  print("")
```

```
k: 1 cost: 550.6434666666669

k: 2 cost: 86.40394533571003

k: 3 cost: 31.38775897435897

k: 4 cost: 19.499400899685114

k: 5 cost: 13.933308757908755

k: 6 cost: 11.107174889156015

k: 7 cost: 9.225808730158732

k: 8 cost: 7.696685296574769

k: 9 cost: 6.472894541406307
```

3.5.1 silhouette Coefficient

masukan untuk evaluasi hasil clustering menggunakan solhouette coefficient

nilai di bawah adalah nilai silhouette dari soal 1 di mana jika nilai yang mendekati 1 maka nilai cluster adalah benar

Hasil pada nilai silhouette coeffisient sebesar adalah 0.3986 dan lebih mendekati 1

[ ]:
```
#Evaluasi hasil cluster menggunakan silhouette coefficient
from sklearn.metrics.cluster import silhouette_score
silhouette_score(X2, labels)
# silhouette score yang baik adalah yang mendekati 1
```

[ ]:
```
0.5864134072932062
```

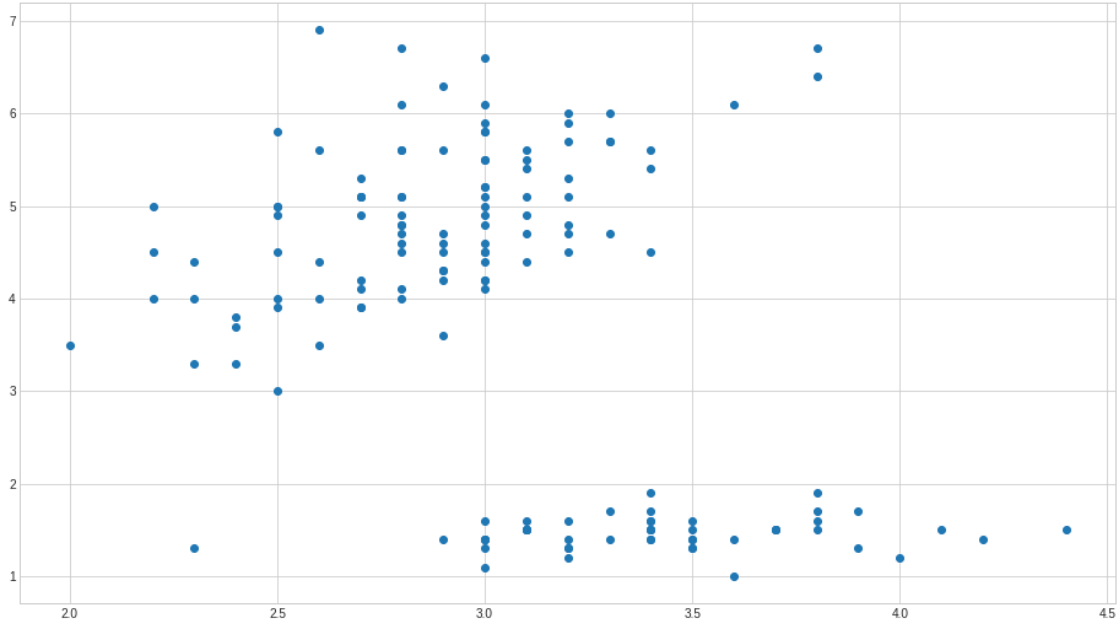
3.6 Tunjukan hasil clustering untuk K 1 sampai 6 dengan colormap ”coolwarm” dan 3 colormap lainnya

Nilai row dan column ditentukan oleh nilai k. jika nilai k 1 maka berada di baris 1 kolom 1

[ ]:
```
from sklearn.datasets import load_iris
iris = load_iris()
features = iris.data.T
```

4 Fitur 1 dan Fitur 2

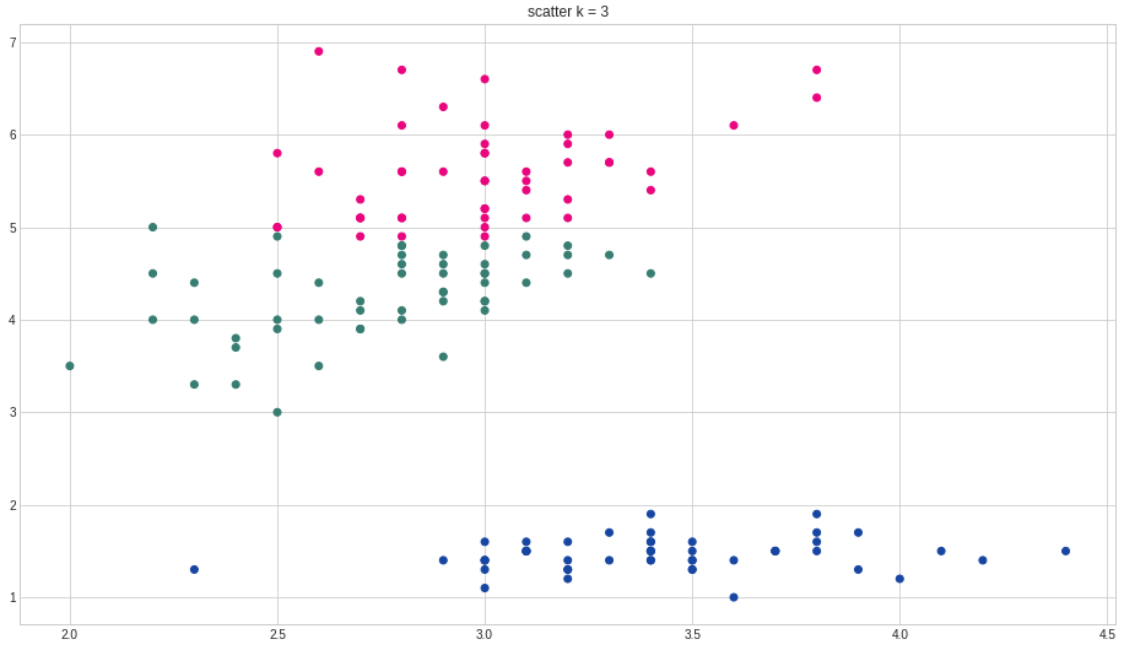
[ ]:
```
# plotting data
plt.plot()
plt.scatter(f1, f2)
plt.show()
```



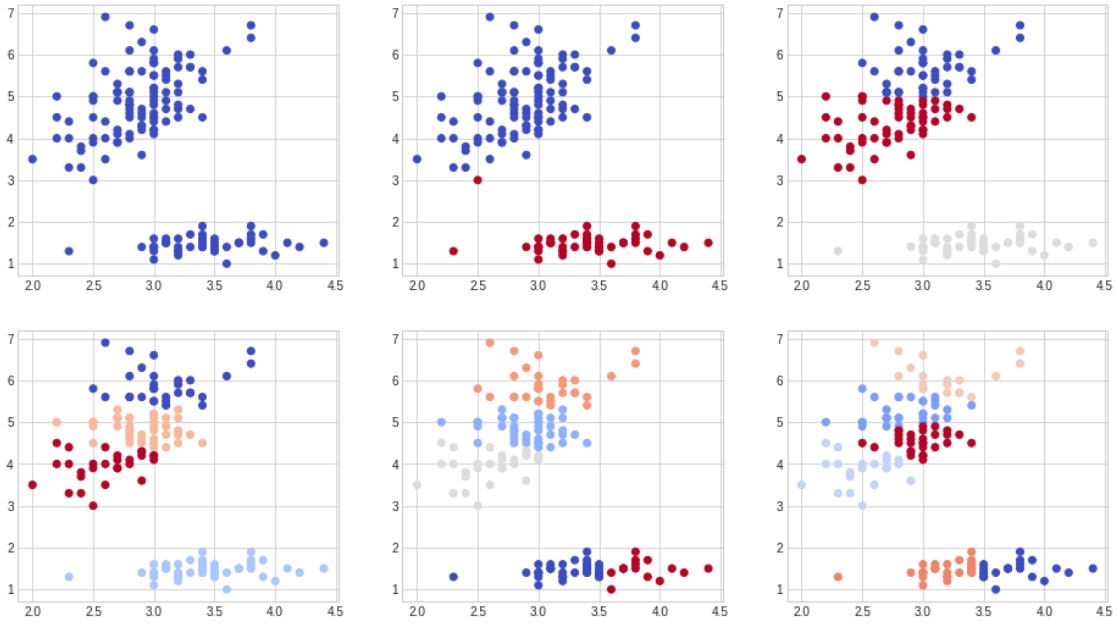
[ ]:
```
#melihat bentuk cluster
y_pred = KMeans(n_clusters=3).fit_predict(X)
plt.plot
color = {0 : '#EA047E',
         1 : '#1746A2',
         2 : '#377D71'}

label_color = [color[l] for l in y_pred]
plt.scatter(f1,f2, c=label_color)
```

```
plt.title("scatter k = 3")
plt.show()
```

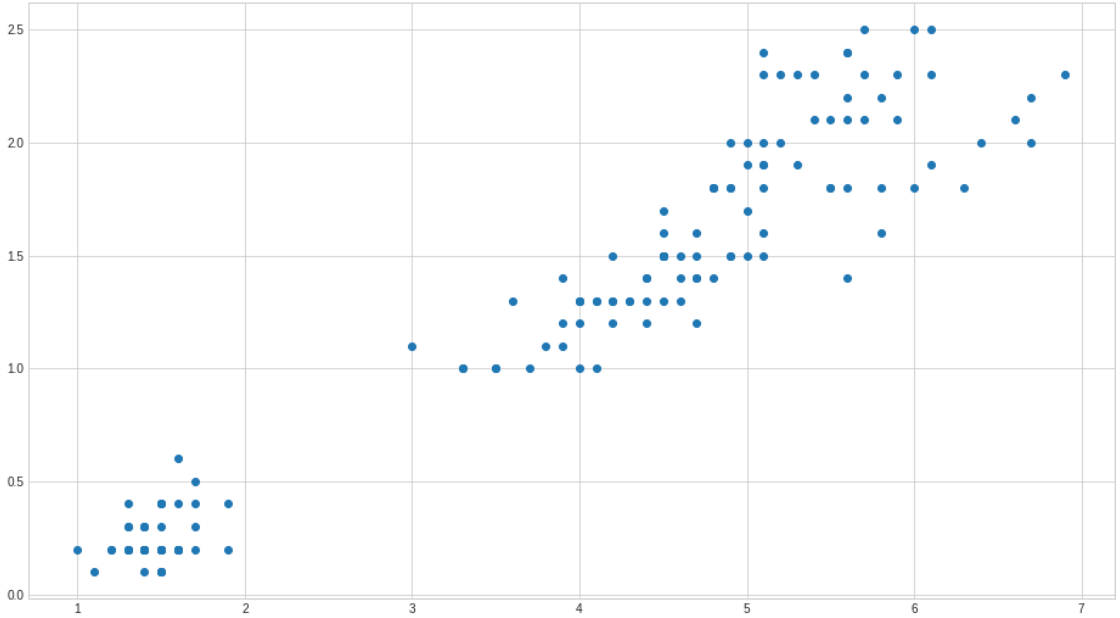


[ ]:
```
#hasil clustering K = 1 sampai 6
figure, ax=plt.subplots(2, 3)
K = range(1, 7)
for k in K:
  if(k<4):
    row = 0
    column = k - 1
  else:
    row = 1
    column = k - 4
  kmeanModel = KMeans(n_clusters = k).fit(X)
  y_pred = kmeanModel.fit_predict(X)
  ax[row][column].scatter(f1, f2, c = y_pred, cmap = 'coolwarm')
plt.show()
```

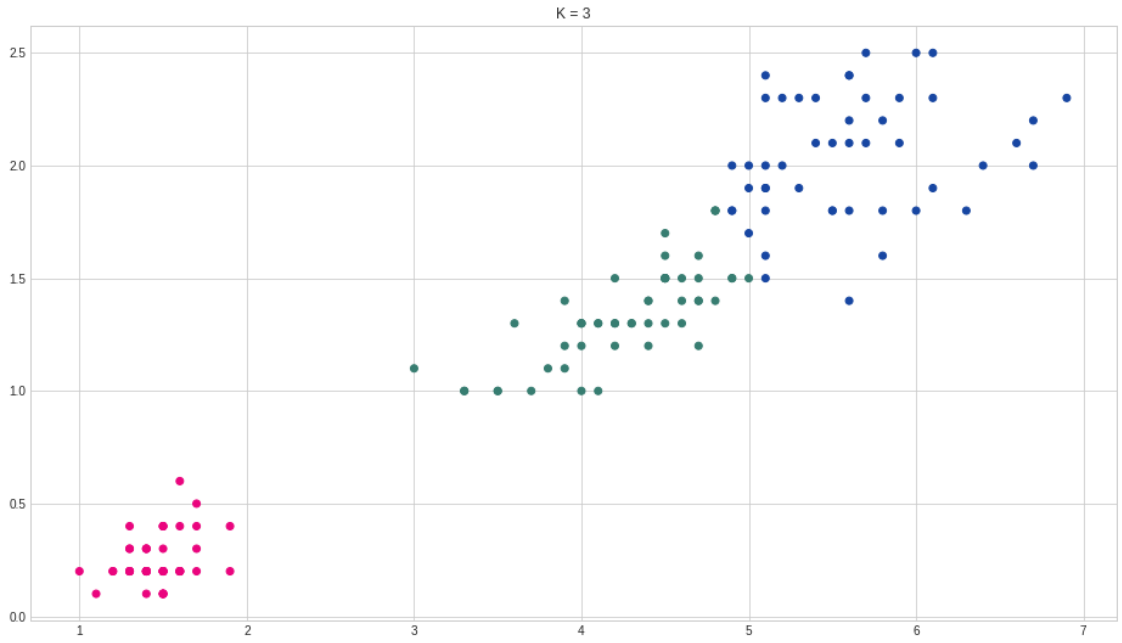


5 Fitur 2 dan 3

[ ]:
```
# plot data
plt.plot()
plt.scatter(f2, f3)
plt.show()
```

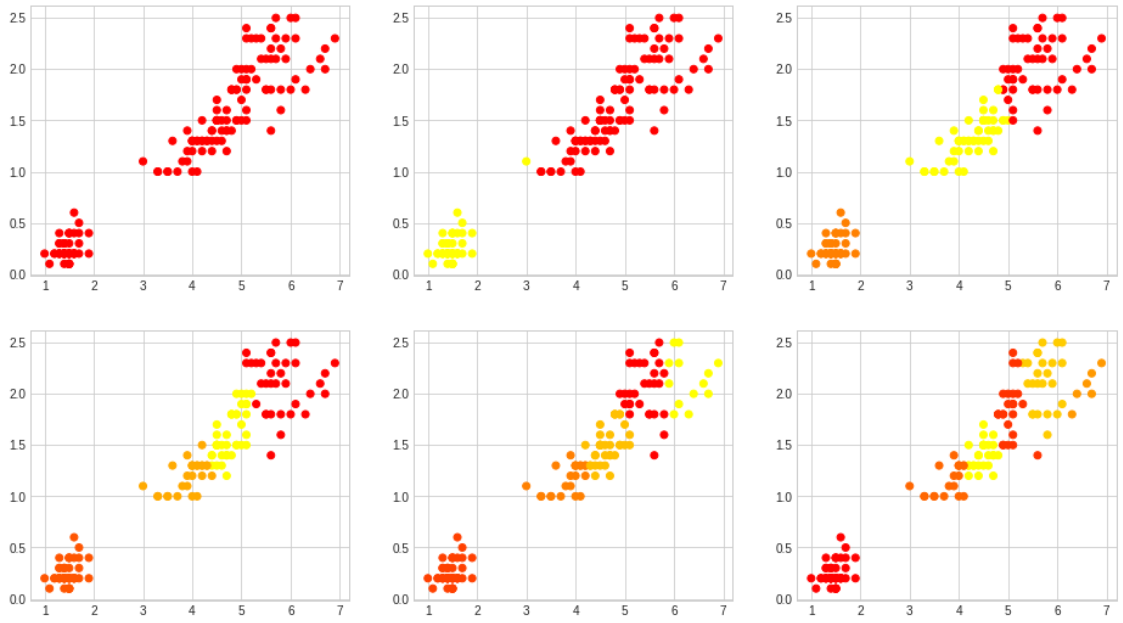


[ ]:
```
# melihat bentuk cluster
y_pred = KMeans(n_clusters=3).fit_predict(X2)
plt.plot
color = {0 : '#EA047E',
         1 : '#1746A2',
         2 : '#377D71'}
label_color = [color[l] for l in y_pred]
plt.scatter(f2, f3, c=label_color)
plt.title("K = 3")
plt.show()
```



[ ]:
```
# hasil clustering K = 1 sampai K = 6
fig, ax = plt.subplots(2,3)
K=range(1,7)
for k in K:
  if(k<4):
    row=0
    column=k-1
  else:
    row=1
    column=k-4
  kmeanmodel = KMeans(n_clusters=k).fit(X2)
  y_pred = kmeanmodel.fit_predict(X2)
  ax[row][column].scatter(f2,f3, c=y_pred, cmap='autumn')
```

```
plt.show()
```



Link google collab: https://colab.research.google.com/drive/13e5Zm9yoIpHS6WZB_f5cIRquLeS8vuDS