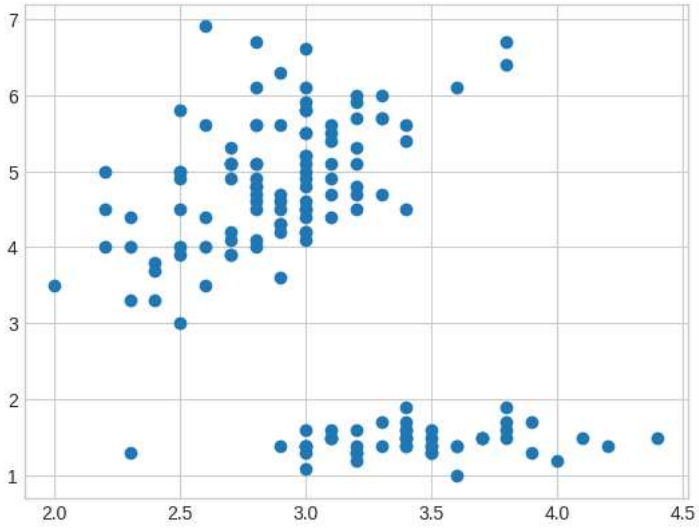
Agung Surya Permana

```
%matplotlib inline
#%matplotlib inline sets the backend of matplotlib to the 'inline' backend
import matplotlib.pyplot as plt
#Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy
plt.style.use("seaborn-whitegrid")
#Seaborn style on matplotlib plot, menentukan jenis graph. untuk jenis graph lain bisa dilihat di https://python-graph-gallery.com/199-matplot
import numpy as np
from sklearn.cluster import KMeans
```

```
<ipython-input-1-056deb4bdaa2>:5: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as th
  plt.style.use("seaborn-whitegrid")
```

Kode tersebut digunakan untuk membuat visualisasi menggunakan Matplotlib dengan gaya "seaborn-whitegrid" dan melakukan pengelompokan data dengan algoritma K-Means dari scikit-learn. %matplotlib inline menunjukkan bahwa plot akan ditampilkan langsung di Jupyter Notebook.

```
# membuat data set
from sklearn.datasets import load_iris
iris = load_iris()
features=iris.data.T
#mengambil data dari fitur data iris.
plt.plot()
plt.scatter(features[1], features[2]) #2 fitur yang akan dipakai
plt.show()
#catatan: jika ingin mencetak/ mmengetahui data iris, di baris paling bawah bisa diketik iris. (atau iris.data , atau iris.target_names)
```



Kode ini membuat dataset menggunakan Iris dataset dari scikit-learn. Selanjutnya, mengambil dua fitur dari dataset tersebut (sepal length dan sepal width) untuk membuat scatter plot menggunakan Matplotlib. Scatter plot ini memvisualisasikan hubungan antara dua fitur tersebut untuk setiap sampel dalam dataset Iris. plt.show() digunakan untuk menampilkan plot. Jika ingin melihat data Iris, dapat menggunakan iris.data atau iris.target_names.

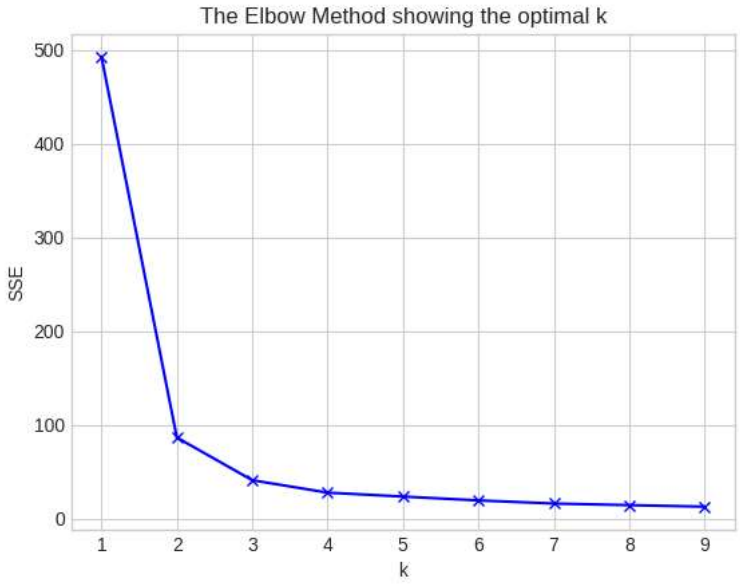
```
import warnings
warnings.filterwarnings('ignore')

X = np.array(list(zip(features[1], features[2]))).reshape(len(features[1]), 2)
SSE = []
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k).fit(X)
    kmeanModel.fit(X)
    SSE.append(kmeanModel.inertia_)
```

Kode ini melakukan pengelompokan (clustering) data menggunakan algoritma K-Means untuk berbagai jumlah klaster (k) dari 1 hingga 9. SSE (Sum of Squared Errors) dihitung untuk setiap jumlah klaster dan disimpan dalam daftar `SSE`. SSE mencerminkan seberapa jauh setiap titik data dalam suatu klaster dari pusat klasternya.

- `warnings.filterwarnings('ignore')`: Mengabaikan peringatan untuk kejadian tertentu selama eksekusi program.
- `X = np.array(list(zip(features[1], features[2]))).reshape(len(features[1]), 2)`: Membentuk array NumPy dari dua fitur (sepal length dan sepal width) dari dataset Iris.
- `SSE = []`: Membuat list kosong untuk menyimpan SSE.
- `K = range(1,10)`: Menetapkan rentang jumlah klaster (k) dari 1 hingga 9.
- Loop `for k in K`: Iterasi melalui setiap nilai k.
    - `kmeanModel = KMeans(n_clusters=k).fit(X)`: Membuat model K-Means dengan jumlah klaster k, dan melatih model tersebut dengan data X.
    - `kmeanModel.fit(X)`: Melatih kembali model untuk mendapatkan atribut-inertia_ yang menunjukkan SSE.
    - `SSE.append(kmeanModel.inertia_)`: Menambahkan nilai SSE ke dalam list SSE untuk nilai k saat ini.

Hasilnya adalah list SSE yang mencerminkan seberapa baik data dapat dikelompokkan dengan jumlah klaster yang berbeda. Dengan plot SSE terhadap jumlah klaster, Anda dapat mencari elbow point (titik siku) yang menunjukkan jumlah klaster yang optimal untuk dataset.

```
# Plot the elbow
plt.plot(K, SSE, 'bx-')
plt.xlabel('k')
plt.ylabel('SSE')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```

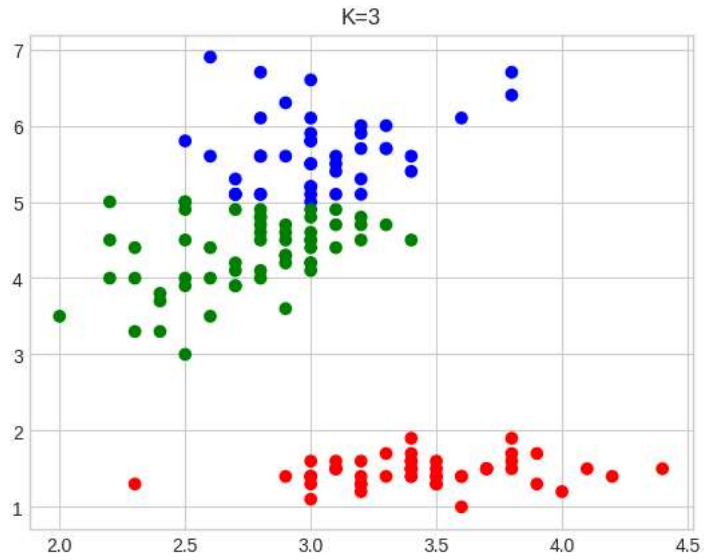


Kode ini membuat plot untuk metode elbow, yang membantu menentukan jumlah klaster yang optimal dalam algoritma K-Means.

- `plt.plot(K, SSE, 'bx-')`: Membuat plot dengan sumbu x berisi nilai k (jumlah klaster) dan sumbu y berisi SSE. 'bx-' menunjukkan bahwa plot menggunakan marker kotak (b) dan garis (-).
- `plt.xlabel('k')`: Menetapkan label sumbu x sebagai 'k' (jumlah klaster).
- `plt.ylabel('SSE')`: Menetapkan label sumbu y sebagai 'SSE' (Sum of Squared Errors).
- `plt.title('The Elbow Method showing the optimal k')`: Menetapkan judul plot sebagai 'The Elbow Method showing the optimal k'.
- `plt.show()`: Menampilkan plot.

Hasilnya adalah grafik yang memvisualisasikan bagaimana SSE (Sum of Squared Errors) berubah seiring dengan peningkatan jumlah klaster (k). Pada titik di mana penurunan SSE mulai melambat, disebut "elbow", itulah jumlah klaster yang sering dianggap sebagai pilihan yang optimal untuk membagi data.

```
y_pred = KMeans(n_clusters=3).fit_predict(X)
plt.plot
LABEL_COLOR_MAP = {0 : 'r',
                   1 : 'g',
                   2 : 'b'
                   }

label_color = [LABEL_COLOR_MAP[l] for l in y_pred]
plt.scatter(features[1],features[2], c=label_color)
plt.title("K=3")
plt.show()
```

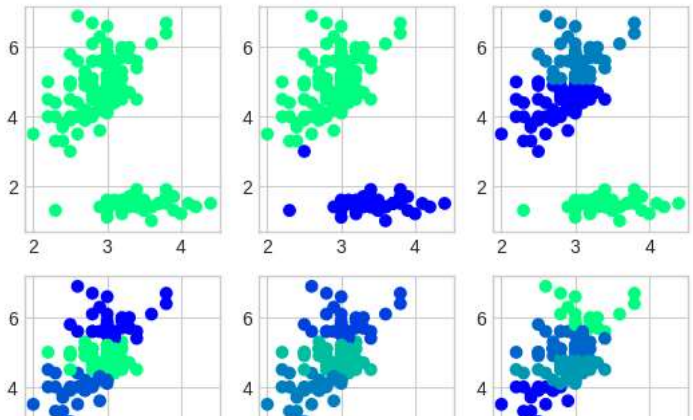


Kode ini menggunakan model K-Means dengan jumlah klaster (k) sebanyak 3 untuk memprediksi klaster dari data yang telah diambil fiturnya.

- `y_pred = KMeans(n_clusters=3).fit_predict(X)` : Melakukan prediksi klaster menggunakan model K-Means dengan jumlah klaster 3, dan menyimpan hasil prediksi dalam `y_pred` .
- `LABEL_COLOR_MAP` : Membuat peta warna untuk label klaster, di mana klaster 0 akan memiliki warna merah ('r'), klaster 1 akan memiliki warna hijau ('g'), dan klaster 2 akan memiliki warna biru ('b').
- `label_color = [LABEL_COLOR_MAP[l] for l in y_pred]` : Membuat daftar warna sesuai dengan label klaster yang diprediksi.
- `plt.scatter(features[1],features[2], c=label_color)` : Membuat scatter plot dari fitur sepal length dan sepal width, diwarnai berdasarkan label klaster yang diprediksi.
- `plt.title("K=3")` : Menetapkan judul plot sebagai "K=3".
- `plt.show()` : Menampilkan plot.

Hasilnya adalah scatter plot yang memvisualisasikan pemisahan data menjadi tiga klaster berdasarkan prediksi model K-Means dengan jumlah klaster sebanyak 3. Setiap klaster ditandai dengan warna yang berbeda sesuai dengan peta warna yang telah ditentukan.

```
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(features[1],features[2], c=y_pred,cmap='winter_r')

plt.show()
```



Kode ini membuat subplot 2x3 dan memvisualisasikan hasil pengelompokan data menggunakan algoritma K-Means dengan jumlah klaster (k) dari 1 hingga 6.

- `figure, ax = plt.subplots(2,3)` : Membuat objek subplot 2x3, yang berarti terdapat 2 baris dan 3 kolom subplot dalam satu gambar.
- `K = range(1, 7)` : Menetapkan rentang nilai k dari 1 hingga 6.
- Loop `for k in K` : Iterasi melalui setiap nilai k.
  - Pengaturan baris (`row`) dan kolom (`column`) dilakukan berdasarkan nilai k. Jika k kurang dari 4, subplot akan berada di baris pertama (row=0) dan kolom ke-(k-1). Jika k lebih besar atau sama dengan 4, subplot akan berada di baris kedua (row=1) dan kolom ke-(k-4).
  - `kmeanModel = KMeans(n_clusters=k).fit(X)` : Membuat dan melatih model K-Means dengan jumlah klaster k.
  - `y_pred = kmeanModel.fit_predict(X)` : Memprediksi klaster untuk setiap sampel dalam dataset.
  - `ax[row][column].scatter(features[1], features[2], c=y_pred, cmap='winter_r')` : Membuat scatter plot pada subplot yang sesuai, dengan warna yang menunjukkan klaster yang diprediksi. Penggunaan `cmap='winter_r'` mengatur peta warna yang digunakan.
- `plt.show()` : Menampilkan gambar dengan semua subplot, di mana setiap subplot menunjukkan hasil pengelompokan data untuk nilai k yang berbeda dari 1 hingga 6.