Minggu 7

Image Processing and Unsupervised Learning

Tujuan Praktikum

1. Memahami dan menerapkan *feature extraction* pada data citra dengan menggunakan Python
2. Memahami dan menerapkan transformasi fitur pada data citra dengan menggunakan Python
3. Memahami dan menerapkan unsupervised learning pada data citra dengan menggunakan Python
4. Mampu untuk mendapatkan pengetahuan dan insight dari data citra dengan menggunakan Python

Dasar Teori

1. Ekstraksi Fitur
   - Fitur adalah bagian atau pola objek dalam citra untuk membantu mengidentifikasi citra. Ada 2 macam fitur dalam citra:
     a. Fitur alami: fitur yang merupakan bagian dari gambar, misal: kecerahan dan tepi dari objek
     b. Fitur buatan: fitur yang diperoleh dengan operasi tertentu pada gambar, misal: histogram dari graylevel.

   - Vektor fitur adalah representasi ringkas dari suatu gambar (atau objek di dalam gambar)
   - Vektor fitur direpresentasikan sebagai array $n \times 1$ yang mengkodekan n fitur dari suatu gambar
   - Vektor fitur numerik x dapat direpresentasikan sebagai berikut:

     $$x = (x_1, x_2, \ldots, x_n)^T$$

     dimana : n = jumlah fitur dan T = operasi transpose
   - Fitur digunakan untuk proses selanjutnya (pengenalan pola, klasifikasi, dll)

- Jika fitur bagus maka hasil akan bagus, dan sebaliknya.
- Ekstraksi fitur pada citra antara lain: ekstraksi fitur warna, tekstur, dan bentuk
- Ekstraksi fitur warna:
  a. RGB
  b. HSV
  c. Grayscale
  d. Binary

  Implementasi dengan python:

```python
import cv2
import numpy as np
import sys

#membaca citra
img_bgr = cv2.imread("./lenna.png")

# #Fitur Warna

# #a. BGR
# mengubah dari 3D menjadi 2D dengan ukuran n x m, dimana n adalah jumlah image dan m adalah jumlah fitur
featRGB = img_bgr.reshape(1,-1)

# # b.HSV
img_hsv = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2HSV)
featHSV = img_hsv.reshape(1,-1)

# # c. Gray
img_gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
featGray = img_gray.reshape(1,-1)

# # d. gray to binary, dengan threshold 127
(threshold, img_bin) = cv2.threshold(img_gray, 127, 255, cv2.THRESH_BINARY)
featBIN = img_bin.reshape(1,-1)


print("Ukuran Citra: ", img_bgr.shape)
print("Ukuran Fitur RGB: ", featRGB.shape)
print("Ukuran Fitur Gray: ", featGray.shape)
print("Ukuran Fitur HSV: ", featHSV.shape)
print("Ukuran Fitur Binary: ", featBIN.shape)
```

  Output :

```
Ukuran Citra:  (330, 330, 3)
Ukuran Fitur RGB:  (1, 326700)
Ukuran Fitur Gray:  (1, 108900)
Ukuran Fitur HSV:  (1, 326700)
Ukuran Fitur Binary:  (1, 108900)
```

  e. Color statistics

  Implementasi dengan python:

```python
import cv2
import numpy as np
from scipy.stats import skew, kurtosis

img = cv2.imread('lenna.png')
(blue, green, red) = cv2.split(img)
blue, green, red = blue.ravel(), green.ravel(), red.ravel()

mean_blue = np.mean(blue)
mean_green = np.mean(green)
mean_red = np.mean(red)
print("Rata-rata warna biru: {:.2f}".format(mean_blue))
print("Rata-rata warna hijau: {:.2f}".format(mean_green))
print("Rata-rata warna merah: {:.2f}".format(mean_red))

std_blue = np.std(blue)
std_green = np.std(green)
std_red = np.std(red)
print("Standar deviasi warna biru: {:.2f}".format(std_blue))
print("Standar deviasi warna hijau: {:.2f}".format(std_green))
print("Standar deviasi warna merah: {:.2f}".format(std_red))

skew_blue = skew(blue, axis=0, bias=True)
skew_green = skew(green, axis=0, bias=True)
skew_red = skew(red, axis=0, bias=True)
print("Skewness warna biru: {:.2f}".format(skew_blue))
print("Skewness warna hijau: {:.2f}".format(skew_green))
print("Skewness warna merah: {:.2f}".format(skew_red))

kurtosis_blue = kurtosis(blue, axis=0, bias=True)
kurtosis_green = kurtosis(green, axis=0, bias=True)
kurtosis_red = kurtosis(red, axis=0, bias=True)
print("kurtosis warna biru: {:.2f}".format(kurtosis_blue))
print("kurtosis warna hijau: {:.2f}".format(kurtosis_green))
print("kurtosis warna merah: {:.2f}".format(kurtosis_red))
```

Output:

```
Rata-rata warna biru: 105.41
Rata-rata warna hijau: 99.05
Rata-rata warna merah: 180.22
Standar deviasi warna biru: 33.65
Standar deviasi warna hijau: 52.72
Standar deviasi warna merah: 48.98
Skewness warna biru: 0.68
Skewness warna hijau: 0.23
Skewness warna merah: -0.70
kurtosis warna biru: -0.16
kurtosis warna hijau: -0.75
kurtosis warna merah: -0.79
```

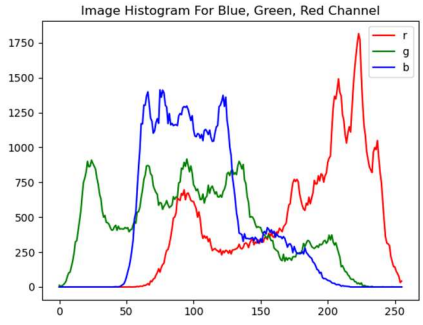
f. Histogram

Histogram Warna $H(w)$ adalah menyatakan frekuensi munculnya warna w

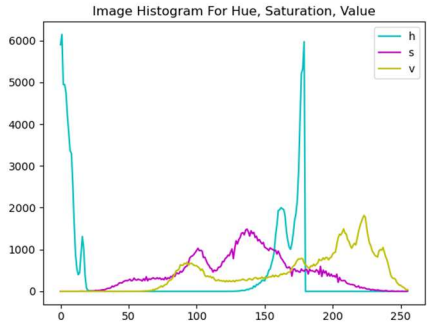
```python
# importing libraries
import cv2
import numpy as np
from matplotlib import pyplot as plt

def show_img(path):

    img = cv2.imread(path)
    b, g, r = img[:,:,0], img[:,:,1], img[:,:,2]
    hist_b = cv2.calcHist([b],[0],None,[256],[0,256])
    hist_g = cv2.calcHist([g],[0],None,[256],[0,256])
    hist_r = cv2.calcHist([r],[0],None,[256],[0,256])
    plt.plot(hist_r, color='r', label="r")
    plt.plot(hist_g, color='g', label="g")
    plt.plot(hist_b, color='b', label="b")
    plt.title('Image Histogram For Blue, Green, Red Channel')
    plt.legend()
    plt.show()
    img2 = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    h, s, v = img2[:,:,0], img2[:,:,1], img2[:,:,2]
    hist_h = cv2.calcHist([h],[0],None,[256],[0,256])
    hist_s = cv2.calcHist([s],[0],None,[256],[0,256])
    hist_v = cv2.calcHist([v],[0],None,[256],[0,256])
    plt.plot(hist_h, color='c', label="h")
    plt.plot(hist_s, color='m', label="s")
    plt.plot(hist_v, color='y', label="v")
    plt.title('Image Histogram For Hue, Saturation, Value')
    plt.legend()
    plt.show()

    return hist_r,hist_g, hist_b, hist_h, hist_s, hist_v

hist_r,hist_g, hist_b, hist_h, hist_s, hist_v = show_img('lenna.png')
```

Output:

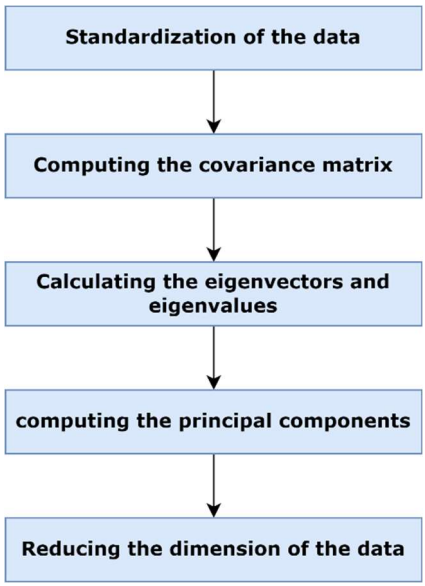
Histogram RGB:



Histogram HSV:



2. Transformasi Citra
   - Proses merubah fitur ke representasi lain sehingga lebih mudah dipahami oleh mesin. Alasan melakukan transformasi fitur:
     a. Mendapatkan fitur penting
     b. Mereduksi dimensi fitur
     c. Memudahkan untuk visualisasi data
   - Beberapa cara transformasi fitur:
     a. Transformasi wavelet
     b. Discrete Wavelet Transform (DWT)
     c. **Principal Component Analysis (PCA)**
   - **PCA** merupakan salah satu tenik transformasi data yang digunakan untuk mengidentifikasi korelasi dan pola dalam dataset sehingga dapat ditransformasika menjadi dataset baru yang memiliki dimensi lebih rendah tanpa kehilangan informasi penting.
   - Langkah-langkah melakukan PCA:



- PCA dapat digunakan untuk visualisasi dan reduksi dimensi citra.
- Implementasi PCA untuk visualisasi:

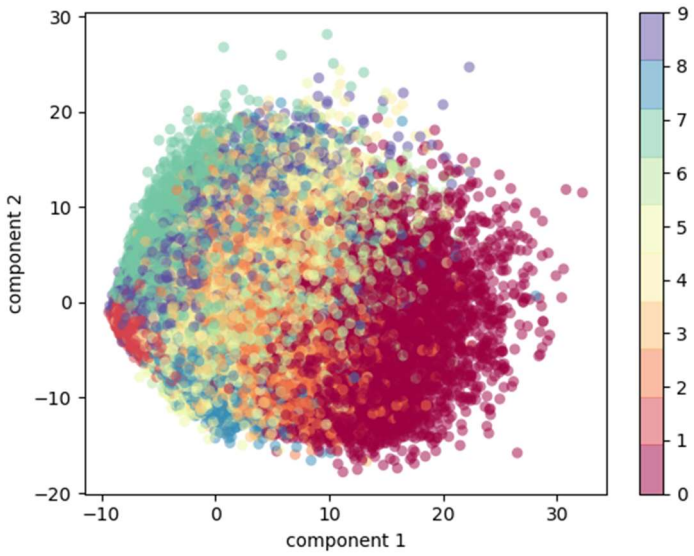
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.datasets import mnist
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

(X_train,y_train), (X_test,y_test) = mnist.load_data()
print("ukuran data train", X_train.shape)

# Reshape-ing X to a 2D array for PCA and then k-means
# We will only be using X for clustering
X = X_train.reshape(-1,X_train.shape[1]*X_train.shape[2])
print("ukuran data train (fitur): ", X.shape)

# To perform PCA we must first change the mean to 0 and
# variance to 1 for X using StandardScalar
X_scaling = StandardScaler().fit_transform(X)


# PCA for visualization
pca = PCA(2)
projected = pca.fit_transform(X_scaling)
print(projected.shape)

plt.scatter(projected[:,0], projected[:,1], c=y_train, edgecolor="none", alpha=0.5, cmap=plt.cm.get_cmap('Spectral', 10))
plt.xlabel('component 1')
plt.ylabel('component 2')
plt.colorbar()
plt.show()
```

Output:

```
ukuran data train (60000, 28, 28)
ukuran data train (fitur):  (60000, 784)
```

Data mnist yang disimpan sebagai data train berjumlah 60000 citra, dengan tiap citra berukuran 28 x 28 (grayscale), jika kita menggunakan fitur tersebut, kita harus

mengubahnya menjadi array 2D. Sehingga ukuran data train yang sebelumnya (60000, 28,28) menjadi (60000, 784) yang berarti pada tiap citra terdapat 784 fitur. Dengan data ini kita bisa melakukan transformasi fitur PCA.



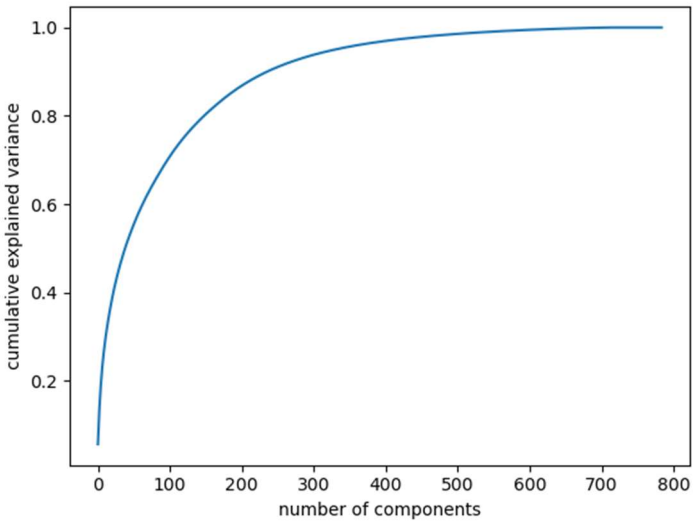
Gambar diatas merupakan visualisasi untuk data mnist digit, dimana terdapat 10 kelas dalam dataset tersebut. Jika PCA digunakan untuk visualisasi 2D, maka jumlah komponen yang digunakan hanya 2, namun jika PCA digunakan untuk visualisasi 3D, maka jumlah komponen yang digunakan adalah 3. Dari visualisasi PCA diatas dapat dilihat bahwa data antar kelas saling tumpang tindih.

- Implementasi PCA untuk reduksi dimensi citra:

  Untuk mereduksi dimensi kita bisa melihat grafik kumulatif dari variance dengan menggunakan kode berikut:

```
pca = PCA().fit(X_scaling)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.show()
```

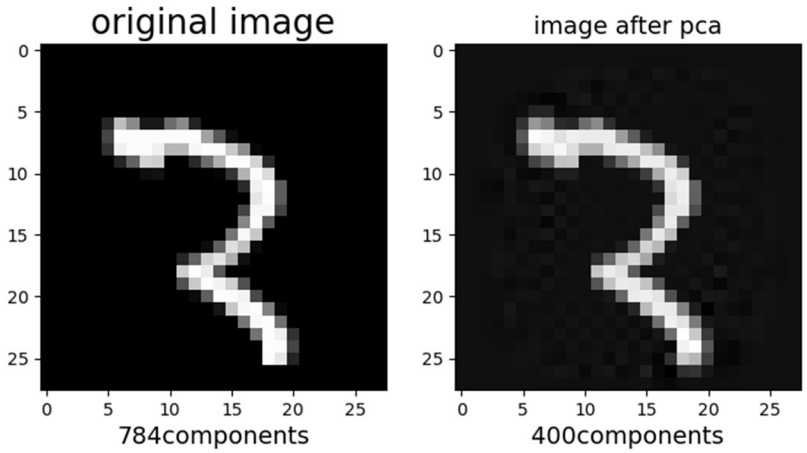
  Output:



Bisa kita lihat pada grafik kumulatif dari variance, nilai kumulatif dari variance stabil pada nilai (kira-kira) antara 0.95 hingga 0.98. kita juga bisa melihat bahwa dengan menggunakan menggunakan komponen 400 atau 784 komponen, nilai kumulatif dari variance tidak jauh berbeda. Kita bisa menggunakan jumlah komponen yang dipilih atau nilai variance yang dipilih sebagai variabel untuk PCA.

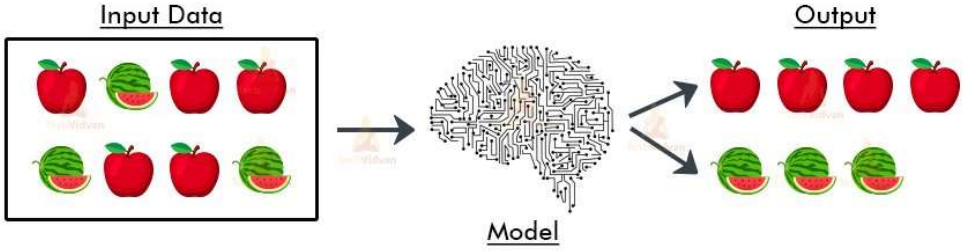
```
# biasanya nilai variance yang dipilih antara 95% - 98%
# --> misal pillih variance = 0.98 atau n_components = 400
pca = PCA(n_components=400)
X_pca = pca.fit_transform(X)
print("dimensi data setelah PCA:", X_pca.shape)

approximation = pca.inverse_transform(X_pca)
plt.figure(figsize=(8,4))
n=500
plt.subplot(1,2,1)
plt.imshow(X[n].reshape(X_train.shape[1], X_train.shape[2]), cmap=plt.get_cmap('gray'))
plt.xlabel(str(X.shape[1])+'components', fontsize=14)
plt.title('original image', fontsize=20)
plt.subplot(1,2,2)
plt.imshow(approximation[n].reshape(X_train.shape[1], X_train.shape[2]), cmap=plt.get_cmap('gray'))
plt.xlabel(str(X_pca.shape[1])+'components', fontsize=14)
plt.title('image after pca', fontsize=14)
plt.show()
```

Output:



Gambar diatas merupakan perbandingan citra asli (jumlah dimensi yang lengkap sesuai dengan jumlah fitur) dengan citra yang sudah direduksi dimensinya dengan PCA (menjadi 400 komponen). Kita bisa melihat meskipun kualitas citra menurun (kompresi citra) namun citra yang telah direduksi dimensinya masih memiliki informasi penting yang dimiliki citra asli.

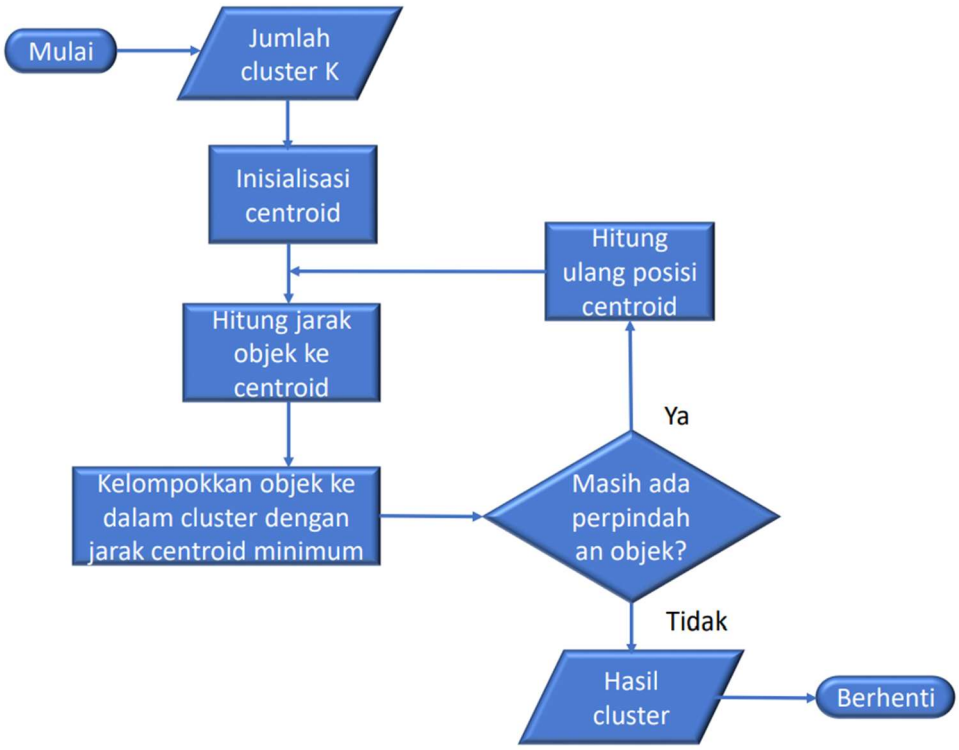
3. Unsupervised Learning – Clustering



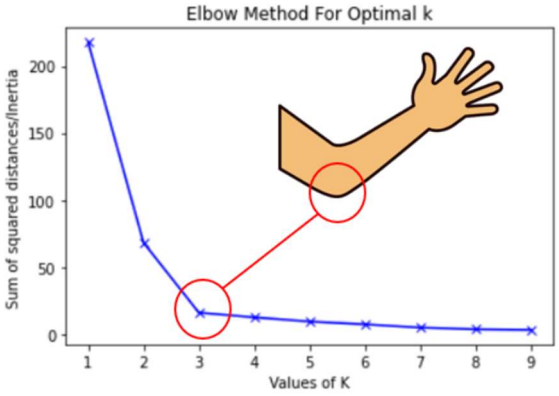
- **Unsupervised Learning** mengidentifikasi pola dalam kumpulan data yang berisi titik data yang tidak **ditandai** (kelas/label)
- Algoritma Clustering
  - **K-means**
  - Hierarchical Clustering
  - DBSCAN
  - Fuzzy C-Means
  - Local Outlier Factor

- dll

- langkah-langkah K-means



- Salah satu tahap yang penting dalam unsupervised learning adalah menentukan jumlah cluster. Salah satu caranya adalah dengan menggunakan elbow method. Cost function *j* berupa nilai inertia, yang merupakan jumlah kuadrat jarak sampel ke pusat cluster terdekat. Kita perlu memilih jumlah cluster *k* secara manual dengan mempertimbangkan trade-off antara nilai inertia dan jumlah cluster, untuk itu kita biasa menggunakan elbow method (atau metode siku), dengan menentukan titik siku dalam grafik inertia, setelah titik tersebut, peningkatan/penurunan nilai inertia tidak signifikan.

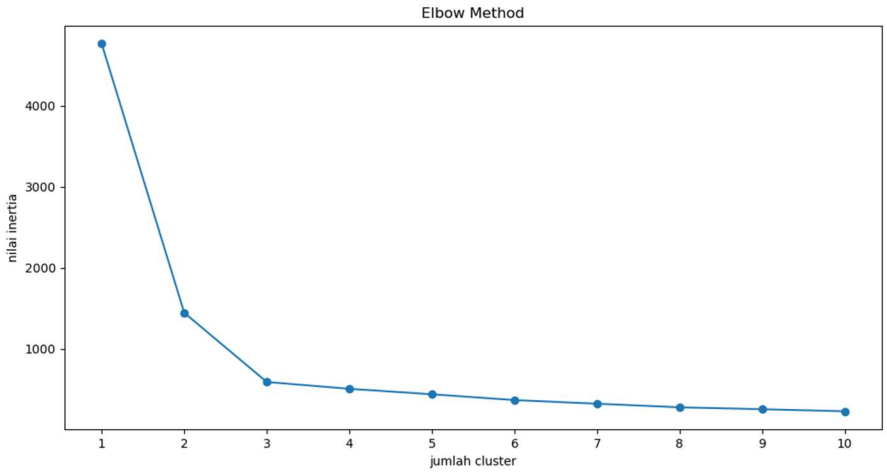


Line plot between K and inertia

- Implementasi untuk mencari jumlah cluster yang optimal:

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets.samples_generator import make_blobs

# membuat dataset untuk percobaan
X, y = make_blobs(n_samples=300, cluster_std=1, random_state=12)
plt.scatter(X[:, 0], X[:,1])
plt.show()

inertias = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(X)
    inertias.append(kmeans.inertia_)

plt.figure(figsize=(12,6))
plt.plot(range(1,11), inertias, marker='o')
plt.xlabel('jumlah cluster')
plt.ylabel('nilai inertia')
plt.title('Elbow Method')
plt.xticks(list(range(1,11)))
plt.show()
```

Output:

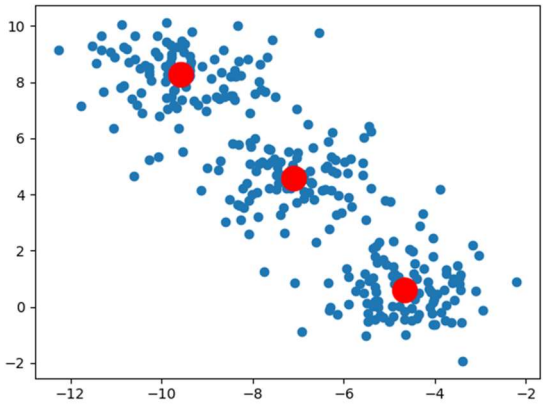


Dari grafik tersebut, kita bisa memilih jumlah cluster 3, karena pada jumlah cluster tersebut terdapat titik siku antara nilai inertia dan jumlah cluster, kemudian setelah cluster 3 peningkatan/penurunan nilai inertia tidak signifikan.

- Implementasi K-means clustering dengan jumlah cluster yang dipilih

```
kmeans = KMeans(n_clusters=3)
pred=y = kmeans.fit_predict(X)
plt.scatter(X[:, 0], X[:,1], cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=300, c='red')
plt.show()
```

Output:



Tugas Praktikum

1. Gunakan dataset fashion mnist dari library keras dan berikan penjelasan mengenai dataset tersebut.
2. Lakukan transformasi fitur dengan menggunakan PCA dan berikan alasan pemilihan jumlah komponen PCA.
3. Bandingkan citra asli dengan citra yang sudah ditransformasi dengan menggunakan PCA, apakah berbeda? Mengapa?
4. Lakukan clustering dari dataset citra yang sudah mengalami transformasi dengan PCA.
5. Lakukan analisis dan interpretasi dari hasil clustering dan berikan juga penjelasan untuk tiap cluster (jumlah anggota tiap cluster, karakteristik tiap cluster, dll).
6. Dokumentasikan setiap langkah dan berikan keterangan untuk setiap langkah yang dilakukan

7. Kumpulkan dalam format .ipynb atau jika tidak memungkinkan kumpulkan dalam format pdf. Format penamaan file: NAMA_NIM_IMAGECLUSTERING