LAPORAN PROGRES PROYEK

MATA KULIAH

10S3001 - KECERDASAN BUATAN

Customer Segmentation Using Gaussian Mixture Model

Disusun Oleh Kelompok 7 :

12S21001 Dhino Rayvaldo Turnip

12S21008 Tuani Putra Manurung

12S21010 Bobby Willy Siagian

12S21013 Markus Pardianto Hutagalung

12S21019 Alex Mario K. Simamora

Tautan GitHub : https://github.com/MarkusHutagalung/customer-segmentation-using-gmm

---

PROGRAM STUDI SARJANA SISTEM INFORMASI

FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO

INSTITUT TEKNOLOGI DEL

TA 2023/2024

---

1. Topik yang Dipilih

Dalam proyek kali ini topik kami adalah **Clusterisasi using Mall Custumer Segmentation Data.**

2. Algoritma

Algoritma yang kami pakai dalam topik ini adalah Gaussian Mixture Model (GMM). Algoritma GMM adalah algoritma dalam *machine learning* yang bertujuan untuk memodelkan distribusi probabilitas dari data dengan mengasumsikan bahwa data tersebut berasal dari campuran beberapa distribusi Gaussian (normal). Berikut adalah langkah-langkah terperinci dari algoritma GMM:

a) **Inisialisasi:**
- Pilih jumlah kluster (komponen Gaussian) yang diinginkan.
- Inisialisasi parameter untuk setiap kluster, termasuk mean, covariance, dan bobot (proporsi) untuk masing-masing distribusi Gaussian.

b) **Expectation-Maximization (EM) Iteration:**
- **Expectation (E-Step):**
  Hitung nilai ekspektasi responsibilitas setiap kluster terhadap setiap data point menggunakan rumus distribusi Gaussian dan bobot kluster.
- **Maximization (M-Step):**
  Perbarui parameter untuk setiap kluster (mean, covariance, dan bobot) dengan memaksimalkan likelihood berdasarkan responsibilitas yang dihitung pada langkah E.

c) **Iterasi:**
- Ulangi langkah E dan M sampai konvergensi tercapai atau kriteria berhenti lainnya.

d) **Prediksi:**
- Setelah konvergensi, atau sejumlah iterasi tertentu, tentukan kluster yang paling mungkin untuk setiap data point berdasarkan responsibilitas yang telah dihitung.

Algoritma GMM menggunakan pendekatan EM untuk melakukan optimasi dan memperbarui parameter secara iteratif. Langkah E digunakan untuk menghitung responsibilitas atau "probabilitas" bahwa setiap data point berasal dari setiap kluster, sedangkan langkah M digunakan untuk memaksimalkan likelihood dengan memperbarui parameter-parameter kluster.

3. Metode

3.1 Data Visual

Berikut ini adalah langkah-langkah untuk menampilkan tampilan visual dari dataset yang telah diberikan:

3.1.1 Import Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import plotly as py
import matplotlib.pyplot as plt
from sklearn.mixture import GaussianMixture
from sklearn.preprocessing import StandardScaler
```

Keterangan tambahan:

- "from sklearn.mixture import GaussianMixture": Scikit-learn adalah library untuk machine learning di Python. Dalam kode ini, model Gaussian Mixture Model (GMM) dari Scikit-learn digunakan untuk melakukan clustering berbasis distribusi Gaussian pada data.
- from sklearn.preprocessing import StandardScaler: StandardScaler adalah bagian dari modul preprocessing di Scikit-learn yang digunakan untuk mentransformasi data sehingga memiliki rata-rata nol dan deviasi standar satu. Ini dilakukan dalam pra-pemrosesan data sebelum menerapkan algoritma machine learning.

3.1.2 Menampilkan Data

```
In [2]: data = pd.read_csv("D:\Semester 5\CERTAN\Proyek Akhir Certan/Mall_Customers.csv")
        data.head()
```

Out[2]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

3.1.3 Menampilkan Statistik Deskriptif

```
In [12]: data.shape
Out[12]: (200, 5)
```

```
In [13]: data.describe()
```

Out[13]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

Output yang dihasilkan akan berupa jumlah baris dan kolom pada data, dan juga hasil statistik deskriptif berupa:

- Count
- Mean
- Std
- Min
- 25%
- 50%
- 75%
- Max

3.1.4 Menampilkan tipe data

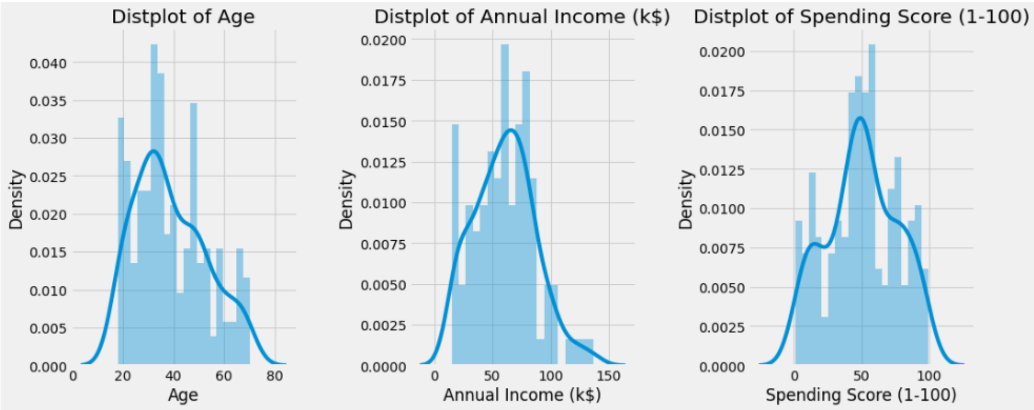
```
In [14]: data.dtypes
Out[14]: CustomerID                int64
         Gender                   object
         Age                       int64
         Annual Income (k$)        int64
         Spending Score (1-100)    int64
         dtype: object
```

3.1.5 Menampilkan Displot

```
In [17]: plt.style.use('fivethirtyeight')
```

```
In [21]: plt.figure(1 , figsize = (15 , 6))
n = 0
for x in ['Age' , 'Annual Income (k$)' , 'Spending Score (1-100)']:
    n += 1
    plt.subplot(1 , 3 , n)
    plt.subplots_adjust(hspace =0.5 , wspace = 0.5)
    sns.distplot(data[x] , bins = 20)
    plt.title('Distplot of {}'.format(x))
plt.show()
```

```
C:\Users\user\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and
will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexi
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\user\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and
will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexi
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\user\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and
will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexi
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Pada kode diatas menghasilkan sebuah figur dengan tiga subplot, masing-masing menampilkan plot distribusi (distplot) dari tiga variabel numerik ('Age', 'Annual Income (k$)', 'Spending Score (1-100)') dalam DataFrame. Setiap subplot disesuaikan dengan posisinya dalam grid 1 baris dan 3 kolom, dan memanfaatkan Seaborn untuk membuat histogram yang menggambarkan distribusi data dengan 20 bin. Pengaturan figur yang luas memastikan ruang yang cukup antar subplot. Tujuan dari plot ini adalah memberikan pemahaman visual tentang distribusi data untuk setiap variabel numerik, yang dapat membantu dalam analisis eksploratif data.

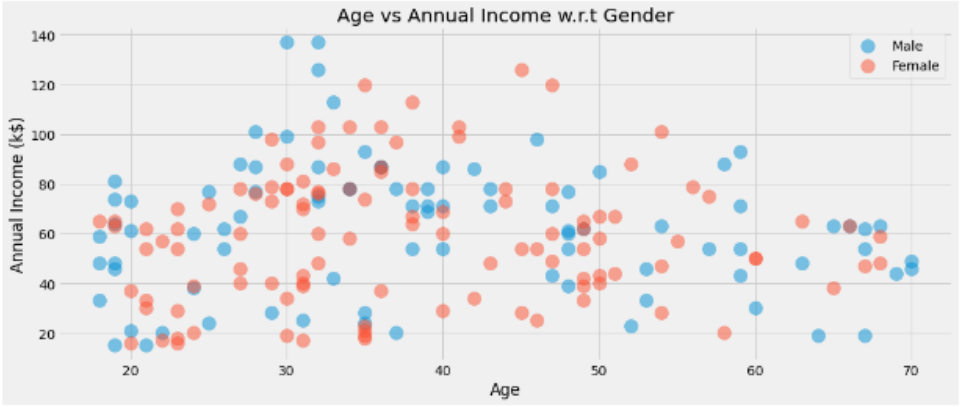
3.1.6 Menampilkan Scatter Plot & Garis Regresi

```
In [25]: plt.figure(1 , figsize = (15 , 7))
         n = 0
         for x in ['Age' , 'Annual Income (k$)' , 'Spending Score (1-100)']:
             for y in ['Age' , 'Annual Income (k$)' , 'Spending Score (1-100)']:
                 n += 1
                 plt.subplot(3 , 3 , n)
                 plt.subplots_adjust(hspace = 0.5 , wspace = 0.5)
                 sns.regplot(x = x , y = y , data = data)
                 plt.ylabel(y.split()[0]+' '+y.split()[1] if len(y.split()) > 1 else y )
         plt.show()
```

Kode diatas akan menciptakan matriks scatter plot dan garis regresi untuk mengeksplorasi hubungan antara semua pasangan variabel numerik ('Age', 'Annual Income (k$)', 'Spending Score (1-100)') dalam DataFrame. Dengan menggunakan nested loop, setiap subplot pada matriks 3x3 menunjukkan scatter plot dari dua variabel yang berbeda. Seaborn digunakan untuk menghasilkan plot tersebut, termasuk garis regresi yang dapat memberikan pandangan lebih dalam tentang tren hubungan antar variabel. Label sumbu y diberikan dengan memanfaatkan nama variabel, dan penyesuaian ruang antar subplot memastikan tata letak yang jelas dan informatif. Tujuan dari matriks ini adalah untuk memberikan gambaran visual yang komprehensif tentang pola hubungan di antara variabel-variabel numerik dalam data set.

3.1.7 Membandingkan Age & Annual Income

```
In [26]: plt.figure(1 , figsize = (15 , 6))
         for gender in ['Male' , 'Female']:
             plt.scatter(x = 'Age' , y = 'Annual Income (k$)' , data = data[data['Gender'
                         s = 200 , alpha = 0.5 , label = gender)
         plt.xlabel('Age'), plt.ylabel('Annual Income (k$)')
         plt.title('Age vs Annual Income w.r.t Gender')
         plt.legend()
         plt.show()
```



Kode ini bertujuan untuk membuat scatter plot yang membandingkan usia (Age) dengan pendapatan tahunan (Annual Income) untuk setiap kelompok gender (Male dan Female) dalam DataFrame. Melalui loop for, dua scatter plot dibuat terpisah untuk setiap gender dengan menggunakan data yang difilter berdasarkan jenis kelamin. Dalam setiap scatter plot, sumbu x mewakili usia, sumbu y mewakili pendapatan tahunan, dan poin-poin data ditandai dengan ukuran 200 dan tingkat transparansi 0.5. Label sumbu dan judul plot diberikan untuk memberikan konteks interpretasi, dan legenda menunjukkan warna yang sesuai dengan setiap kelompok gender. Dengan demikian, visualisasi ini memungkinkan pemahaman yang cepat tentang distribusi dan pola hubungan antara usia, pendapatan tahunan, dan gender dalam data pelanggan mall.

3.2 Clusterisasi Menggunakan Algoritma GMM

3.2.1 Menyimpan ke variabel cluster_data

```
In [29]: # Selecting relevant features for clustering
         cluster_data = data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]
```

Dalam kode diatas digunakan untuk memilih kolom-kolom tertentu dari DataFrame **data** dan menyimpannya dalam variabel **cluster_data**. Kolom-kolom yang dipilih untuk analisis clustering adalah 'Age' (usia), 'Annual Income (k$)' (pendapatan tahunan dalam ribu dolar), dan 'Spending Score (1-100)' (skor pengeluaran dalam rentang 1-100).

3.2.2 Standarisasi Data

```
In [30]: # Standardize the data
         scaler = StandardScaler()
         cluster_data_scaled = scaler.fit_transform(cluster_data)
```

Dalam kode yang ada diatas digunakan untuk melakukan standarisasi data. Proses standarisasi adalah suatu langkah dalam pra-pemrosesan data di mana nilai-nilai dalam setiap kolom diubah sehingga memiliki rata-rata nol dan deviasi standar satu.

3.2.3 Aplikasikan GMM (Gaussian Mixture Model)

```
In [31]: # Fit Gaussian Mixture Model (GMM)
         gmm = GaussianMixture(n_components=3, random_state=42)
         gmm.fit(cluster_data_scaled)

Out[31]: GaussianMixture(n_components=3, random_state=42)
```

Pertama-tama, objek GMM dibuat dengan mengatur jumlah komponen (kluster) menjadi tiga, dan parameter tersebut digunakan untuk inisialisasi model. Selanjutnya, model GMM dilatih menggunakan metode **fit** pada data yang telah di-standarisasi, di mana model mencoba menemukan distribusi Gaussian yang paling sesuai dengan struktur data. Hasilnya, model GMM dapat digunakan untuk melakukan prediksi kluster pada data atau menjalankan analisis lebih lanjut terkait dengan struktur kluster yang diidentifikasi dalam data tersebut. Parameter **random_state=42** digunakan untuk memastikan reproduktibilitas hasil dengan memberikan seed ke generator angka acak.

3.2.4 Prediksi Kluster

```
In [32]: # Predict clusters
         clusters = gmm.predict(cluster_data_scaled)
```

Dalam kode diatas dilakukan untuk memprediksi kluster pada data yang telah di-standarisasi menggunakan model Gaussian Mixture Model (GMM) yang telah diaplikasikan sebelumnya. Dengan menggunakan metode **predict** dari model GMM, setiap observasi dalam data diberikan label kluster yang paling sesuai berdasarkan distribusi Gaussian yang telah diidentifikasi selama proses pengaplikasian model. Label-label kluster ini, yang merepresentasikan kelompok atau segmen data tertentu, disimpan dalam variabel **clusters**

3.2.5 Tambahkan Label Kluster

```
In [33]: # Add cluster labels to the original dataframe
         data['Cluster'] = clusters
```

Kode diatas digunakan untuk menambahkan label kluster yang telah diprediksi (**clusters**) ke dalam DataFrame asli **data**. Dengan menetapkan kolom baru 'Cluster' pada DataFrame dan mengisi nilainya dengan label kluster yang sesuai untuk setiap observasi, kita dapat dengan mudah mengintegrasikan informasi kluster ke dalam dataset utama.

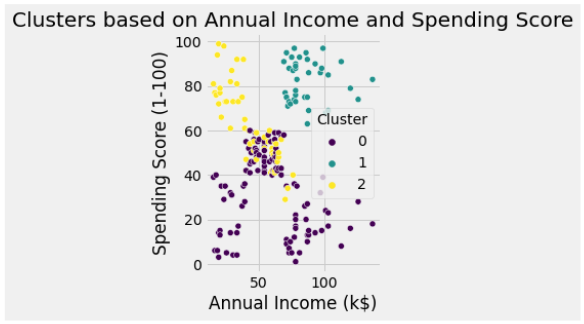
3.2.6 Visualisasi Kluster

```
In [34]: # Visualize the clusters
         plt.figure(figsize=(15, 7))

Out[34]: <Figure size 1080x504 with 0 Axes>

         <Figure size 1080x504 with 0 Axes>
```

Kode yang ada diatas digunakan untuk membuat visualisasi kluster menggunakan matplotlib dengan ukuran figur sebesar 15 unit lebar dan 7 unit tinggi. Visualisasi ini akan membantu dalam memahami pola dan distribusi kluster yang telah diprediksi.

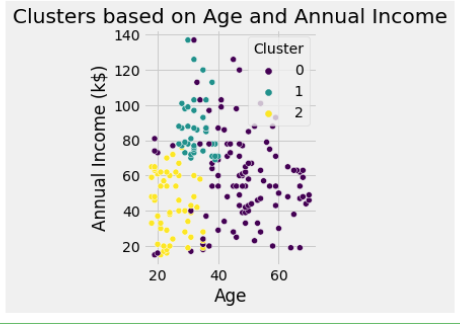
3.2.7 Analisis Klustering

```
In [35]: plt.subplot(1, 2, 1)
         sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)', hue='Cluster
         plt.title('Clusters based on Annual Income and Spending Score')
         plt.savefig('cluster_annual_income_spending_score.png')  # Save the plot
```



Pertama-tama kode ini akan membuat subplot pertama dalam figur dengan menggunakan Seaborn untuk membuat scatter plot dari dua variabel, 'Annual Income (k$)' dan 'Spending Score (1-100)', dengan warna poin data yang ditentukan oleh label kluster dari kolom 'Cluster' dalam DataFrame **data**. Pengaturan palet warna menggunakan 'viridis' dan judul plot diberikan sebagai 'Clusters based on Annual Income and Spending Score'. Selanjutnya, plot tersebut disimpan dalam format file gambar PNG dengan nama 'cluster_annual_income_spending_score.png' menggunakan fungsi **savefig**.

```
In [36]: plt.subplot(1, 2, 2)
         sns.scatterplot(x='Age', y='Annual Income (k$)', hue='Cluster', data=data, palet
         plt.title('Clusters based on Age and Annual Income')
         plt.savefig('cluster_age_annual_income.png')  # Save the plot

         plt.show()
```



Kode ini membuat subplot kedua dalam figur yang sama dengan subplot sebelumnya. Pada subplot ini, Seaborn digunakan untuk membuat scatter plot dari dua variabel, 'Age' dan 'Annual Income (k$)', dengan warna poin data yang ditentukan oleh label kluster dari kolom 'Cluster' dalam DataFrame **data**. Pengaturan palet warna menggunakan 'viridis' dan judul plot diberikan sebagai 'Clusters based on Age and Annual Income'. Selanjutnya, plot ini disimpan dalam format file gambar PNG dengan nama 'cluster_age_annual_income.png' menggunakan fungsi **savefig.**