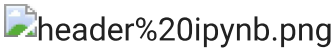


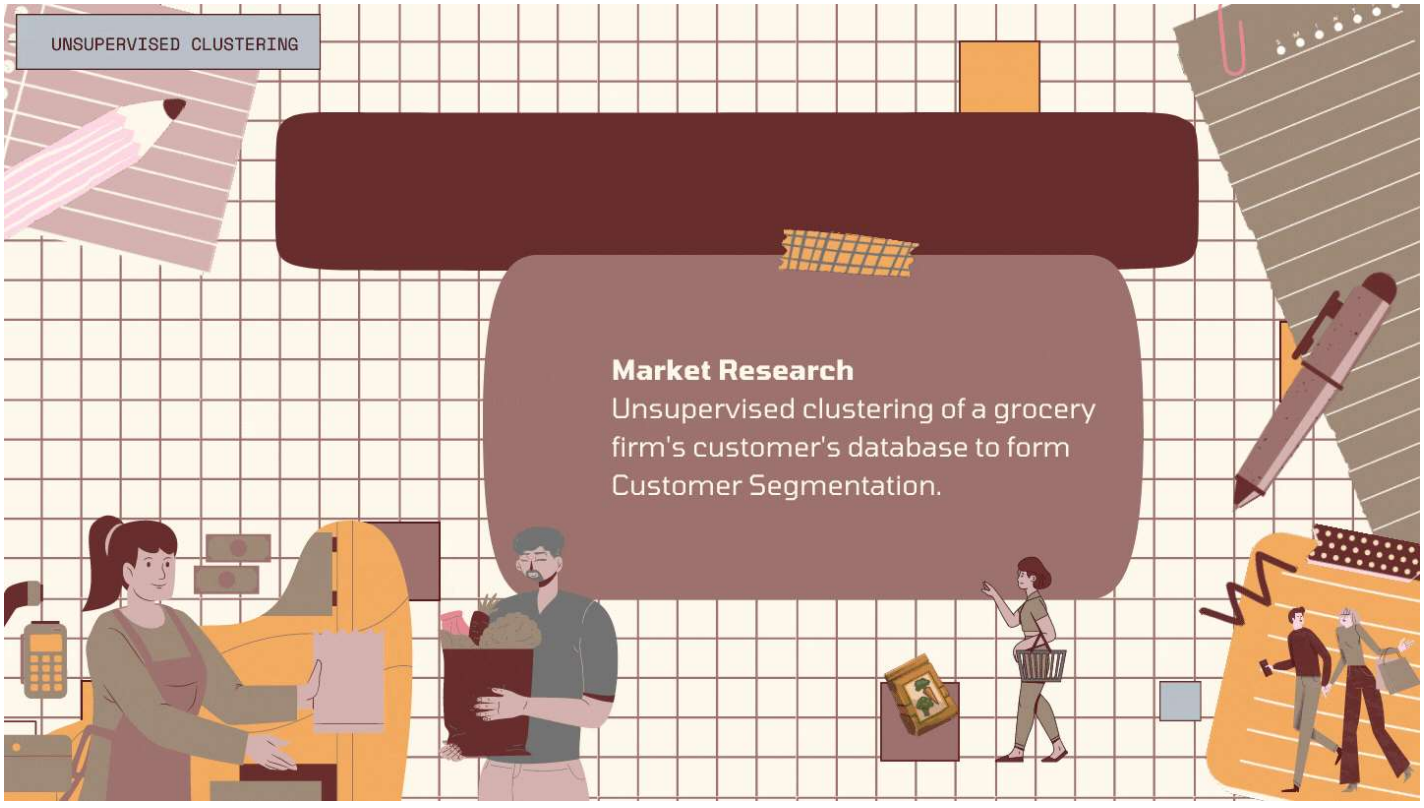
Hands-On

Hands-On ini digunakan pada kegiatan Microcredential Associate Data Scientist 2021

Tugas Mandiri Pertemuan 15

Pertemuan 15 (limabelas) pada Microcredential Associate Data Scientist 2021 menyampaikan materi mengenai Membangun Model (Clustering). silakan Anda kerjakan Latihan 1 s/d 10. Output yang anda lihat merupakan panduan yang dapat Anda ikuti dalam penulisan code :)

Customer Segmentation



Dalam Kasus ini, kita akan melakukan pengelompokan data tanpa pengawasan /unsupervised clustering pada catatan pelanggan dari database perusahaan bahan makanan. Segmentasi pelanggan/Customer segmentation adalah praktik memisahkan pelanggan ke dalam kelompok-kelompok yang mencerminkan kesamaan di antara pelanggan di setiap cluster. Kita akan membagi pelanggan menjadi beberapa segmen untuk mengoptimalkan signifikansi setiap pelanggan bagi

bisnis. Untuk memodifikasi produk sesuai dengan kebutuhan dan perilaku pelanggan yang berbeda. Ini juga membantu bisnis untuk memenuhi kekhawatiran berbagai jenis pelanggan.

TABLE OF CONTENTS

IMPORTING LIBRARIES

Latihan (1)

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
%cd /content/drive/My Drive/tugas15
!ls
```

```
Mounted at /content/drive
/content/drive/My Drive/tugas15
marketing_campaign.csv
```

```
# Import library numpy untuk operasi fungsi aritmatika
import numpy as np

# import library pandas untuk operasi dataframe
import pandas as pd

# Import library matplotlib dan seaborn untuk visualisasi
import seaborn as sns
import matplotlib.pyplot as plt
```

```
%matplotlib inline
plt.style.use('seaborn')

# Import library Axes3D untuk vizualisasi 3 Dimensi
import mpl_toolkits
from mpl_toolkits.mplot3d import Axes3D

# import library datetime untuk operasi yang berhubungan dengan waktu.
from datetime import datetime

# import library Label encoder untuk mengubah setiap nilai dalam kolom menjadi angka yang ber
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing

# import library StandardScaler untuk menskalakan nilai kolom jika terdpaat perbedaan skala,
from sklearn.preprocessing import StandardScaler

# import library PCA adalah prosedur statistik yang mengekstrak fitur-fitur terpenting dari s
from sklearn.decomposition import PCA

# import library KElbowVisualizer untuk mengimplementasikan metode "elbow/siku" untuk data sc
from yellowbrick.cluster import KElbowVisualizer

# import library KMeans metode adalah teknik unsupervised machine learning yang digunakan unt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
from sklearn import metrics


# import library AgglomerativeClustering untuk melakukan pengelompokan data menggunakan botto
from sklearn.cluster import AgglomerativeClustering


# import library metrics untuk mengimplementasikan fungsi yang menilai kesalahan prediksi unt
from sklearn.metrics import confusion_matrix

# me-non aktifkan peringatan pada python
import warnings
warnings.filterwarnings('ignore')
```

LOADING DATA

Latihan (2)

```
#Load the dataset dan tampilkan data nya
```

```
df = pd.read_csv('/content/drive/MyDrive/tugas15/marketing_campaign.csv', sep='\t')
```

```
df.head()
```

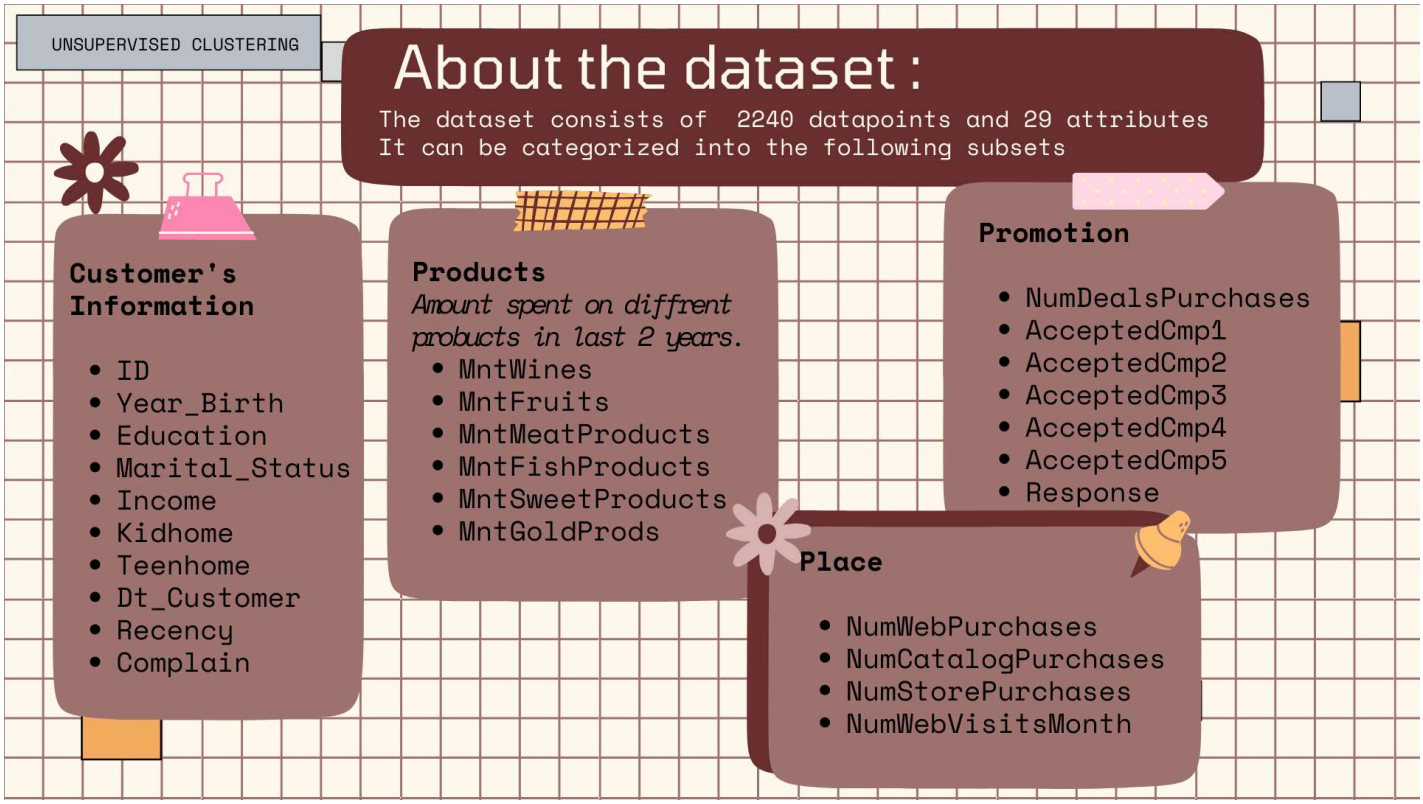
	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2012
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2014
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-2013
3	6182	1984	Graduation	Together	26646.0	1	0	10-02-2014
4	5324	1981	PhD	Married	58293.0	1	0	19-01-2014

```
print('Jumlah data :',len(df))
```

```
Jumlah data : 2240
```

```
df.describe()
```

	ID	Year_Birth	Income	Kidhome	Teenhome	Recency	
count	2240.000000	2240.000000	2216.000000	2240.000000	2240.000000	2240.000000	22
mean	5592.159821	1968.805804	52247.251354	0.444196	0.506250	49.109375	3
std	3246.662198	11.984069	25173.076661	0.538398	0.544538	28.962453	3
min	0.000000	1893.000000	1730.000000	0.000000	0.000000	0.000000	
25%	2828.250000	1959.000000	35303.000000	0.000000	0.000000	24.000000	
50%	5458.500000	1970.000000	51381.500000	0.000000	0.000000	49.000000	1
75%	8427.750000	1977.000000	68522.000000	1.000000	1.000000	74.000000	5
max	11191.000000	1996.000000	666666.000000	2.000000	2.000000	99.000000	14



Untuk informasi lebih lanjut tentang atribut data disini.

DATA CLEANING

Di bagian ini

- Data Cleaning
- Feature Engineering

Untuk mendapatkan pemahaman penuh tentang langkah-langkah apa yang harus kita ambil untuk membersihkan dataset. Mari kita lihat informasi dalam data.

Latihan (3)

```
# Melihat Informasi lebih detail mengenai struktur DataFrame dapat dilihat menggunakan fungsi
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   2240 non-null   int64
 1   Year_Birth           2240 non-null   int64
 2   Education            2240 non-null   object
 3   Marital_Status       2240 non-null   object
```

```
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64
 6   Teenhome             2240 non-null   int64
 7   Dt_Customer          2240 non-null   object
 8   Recency              2240 non-null   int64
 9   MntWines             2240 non-null   int64
 10  MntFruits            2240 non-null   int64
 11  MntMeatProducts      2240 non-null   int64
 12  MntFishProducts      2240 non-null   int64
 13  MntSweetProducts     2240 non-null   int64
 14  MntGoldProds         2240 non-null   int64
 15  NumDealsPurchases    2240 non-null   int64
 16  NumWebPurchases      2240 non-null   int64
 17  NumCatalogPurchases  2240 non-null   int64
 18  NumStorePurchases    2240 non-null   int64
 19  NumWebVisitsMonth    2240 non-null   int64
 20  AcceptedCmp3         2240 non-null   int64
 21  AcceptedCmp4         2240 non-null   int64
 22  AcceptedCmp5         2240 non-null   int64
 23  AcceptedCmp1         2240 non-null   int64
 24  AcceptedCmp2         2240 non-null   int64
 25  Complain             2240 non-null   int64
 26  Z_CostContact        2240 non-null   int64
 27  Z_Revenue            2240 non-null   int64
 28  Response             2240 non-null   int64
dtypes: float64(1), int64(25), object(3)
memory usage: 507.6+ KB
```

Dari output di atas, kita dapat menyimpulkan dan mencatat bahwa:

- Ada nilai yang hilang/missing value dalam kolom income
- Dt_Customer yang menunjukkan tanggal pelanggan bergabung dengan database tidak diuraikan sebagai DateTime
- Ada beberapa fitur kategoris dalam dataframe; karena ada beberapa fitur bertipe object. Jadi kita perlu mengkodekannya ke dalam bentuk numerik nanti.

Pertama-tama, untuk nilai yang hilang, kita hanya akan menghapus baris yang memiliki nilai pendapatan yang hilang.

```
# menghapus missing values
df.dropna(inplace=True)
```

```
df.isnull().sum()
```

```
ID                   0
Year_Birth           0
Education            0
Marital_Status       0
Income               0
Kidhome              0
Teenhome             0
```

```
Dt_Customer            0
Recency                0
MntWines               0
MntFruits              0
MntMeatProducts        0
MntFishProducts        0
MntSweetProducts       0
MntGoldProds           0
NumDealsPurchases      0
NumWebPurchases        0
NumCatalogPurchases    0
NumStorePurchases      0
NumWebVisitsMonth      0
AcceptedCmp3           0
AcceptedCmp4           0
AcceptedCmp5           0
AcceptedCmp1           0
AcceptedCmp2           0
Complain               0
Z_CostContact          0
Z_Revenue              0
Response               0
dtype: int64
```

```
df.shape
```

```
(2216, 29)
```

Pada langkah selanjutnya, kita akan membuat fitur dari "**Dt_Customer**" yang menunjukkan jumlah hari pelanggan terdaftar di database perusahaan. Namun, untuk membuatnya tetap sederhana, kita mengambil nilai ini relatif terhadap pelanggan terbaru dalam catatan.

Jadi untuk mendapatkan nilai, kita harus memeriksa tanggal rekaman terbaru dan terlama.

```
df["Dt_Customer"] = pd.to_datetime(df["Dt_Customer"])
dates=[]
for i in df['Dt_Customer']:
    i=i.date()
    dates.append(i)
print("Tanggal pendaftaran pelanggan terbaru dalam catatan",max(dates))
print("Tanggal pendaftaran pelanggan terlama dalam catatan",min(dates))
```

```
Tanggal pendaftaran pelanggan terbaru dalam catatan 2014-12-06
Tanggal pendaftaran pelanggan terlama dalam catatan 2012-01-08
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2216 entries, 0 to 2239
Data columns (total 29 columns):
```

```
 #   Column               Non-Null Count  Dtype         
---  ------               --------------  -----         
 0   ID                   2216 non-null   int64         
 1   Year_Birth           2216 non-null   int64         
 2   Education            2216 non-null   object        
 3   Marital_Status       2216 non-null   object        
 4   Income               2216 non-null   float64       
 5   Kidhome              2216 non-null   int64         
 6   Teenhome             2216 non-null   int64         
 7   Dt_Customer          2216 non-null   datetime64[ns]
 8   Recency              2216 non-null   int64         
 9   MntWines             2216 non-null   int64         
 10  MntFruits            2216 non-null   int64         
 11  MntMeatProducts      2216 non-null   int64         
 12  MntFishProducts      2216 non-null   int64         
 13  MntSweetProducts     2216 non-null   int64         
 14  MntGoldProds         2216 non-null   int64         
 15  NumDealsPurchases    2216 non-null   int64         
 16  NumWebPurchases      2216 non-null   int64         
 17  NumCatalogPurchases  2216 non-null   int64         
 18  NumStorePurchases    2216 non-null   int64         
 19  NumWebVisitsMonth    2216 non-null   int64         
 20  AcceptedCmp3         2216 non-null   int64         
 21  AcceptedCmp4         2216 non-null   int64         
 22  AcceptedCmp5         2216 non-null   int64         
 23  AcceptedCmp1         2216 non-null   int64         
 24  AcceptedCmp2         2216 non-null   int64         
 25  Complain             2216 non-null   int64         
 26  Z_CostContact        2216 non-null   int64         
 27  Z_Revenue            2216 non-null   int64         
 28  Response             2216 non-null   int64         
dtypes: datetime64[ns](1), float64(1), int64(25), object(2)
memory usage: 519.4+ KB
```

Membuat fitur (**"Customer_For"**) dari jumlah hari pelanggan mulai berbelanja di toko relatif terhadap tanggal terakhir yang tercatat

```
# Membuat fitur "Customer_For"
from datetime import date
days = []
d1 = max(df.Dt_Customer) # membawanya menjadi pelanggan terbaru
for i in df.Dt_Customer:
    delta = d1 - i
    days.append(delta.days)
df["Customer_For"] = days
df["Customer_For"] = pd.to_numeric(df["Customer_For"], errors="coerce")


df.head()
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer
0	5524	1957	Graduation	Single	58138.0	0	0	2012-04-09
1	2174	1954	Graduation	Single	46344.0	1	1	2014-08-03
2	4141	1965	Graduation	Together	71613.0	0	0	2013-08-21
3	6182	1984	Graduation	Together	26646.0	1	0	2014-10-02
4	5324	1981	PhD	Married	58293.0	1	0	2014-01-19

```
df.Dt_Customer[3]
```

```
Timestamp('2014-10-02 00:00:00')
```

```
d1
```

```
Timestamp('2014-12-06 00:00:00')
```

Sekarang kita akan mengeksplorasi nilai unik dalam fitur kategoris untuk mendapatkan gambaran yang jelas tentang data.

```
print("Total kategori dalam fitur Marital_Status:\n\n", df["Marital_Status"].value_counts(),
print("Total kategori dalam fitur Education:\n\n", df["Education"].value_counts())
```

```
Total kategori dalam fitur Marital_Status:

 Married     857
Together    573
Single      471
Divorced    232
Widow        76
Alone         3
Absurd        2
YOLO          2
Name: Marital_Status, dtype: int64

Total kategori dalam fitur Education:

 Graduation    1116
PhD            481
Master         365
2n Cycle       200
Basic           54
Name: Education, dtype: int64
```

Latihan (4)

Pada step berikutnya, kita akan melakukan langkah-langkah berikut untuk merekayasa beberapa fitur baru:

- Ekstrak "**Age**" dari pelanggan dengan "**Year_Birth**" yang menunjukkan tahun lahir orang yang bersangkutan.
- Buat fitur lain "**Spent**" yang menunjukkan jumlah total yang dibelanjakan oleh pelanggan dalam berbagai kategori selama rentang waktu dua tahun.
- Buat fitur lain "**Living_With**" dari "**Marital_Status**" untuk mengekstrak situasi kehidupan pasangan.
- Buat fitur "**Children**" untuk menunjukkan jumlah anak dalam rumah tangga, anak-anak dan remaja.
- Untuk mendapatkan kejelasan lebih lanjut tentang rumah tangga, Membuat fitur yang menunjukkan "**Family_Size**"
- Buat fitur "**Is_Parent**" untuk menunjukkan status orang tua
- Terakhir, kita akan membuat tiga kategori di "**Education**" dengan menyederhanakan penghitungan nilainya.
- Menjatuhkan beberapa fitur yang berlebihan / redundant features

```
#Feature Engineering

# Usia pelanggan hari ini
age=[]
for i in df.Year_Birth:
  age.append(int((datetime.now().year - df.Year_Birth[i])))
df['Age']=  age

# Total pengeluaran untuk berbagai macam item
df['Spent'] = df['MntWines']+df['MntFruits']+df['MntMeatProducts']+df['MntFishProducts']+df['


# situasi kehidupan dari status pernikahan "Alone"
df['Living_With']= df['Marital_Status']

# Fitur yang menunjukkan jumlah anak yang tinggal di rumah tangga
df['Children']= df['Kidhome'] + df['Teenhome']

# Fitur untuk total anggota dalam rumah tangga
df['Family_Size'] = df['Kidhome']+df['Teenhome']+df['Recency']

# Fitur yang berkaitan dengan orang tua
df['Is_Parent'] = df['Marital_Status']
```

```
df['Collected'] = '2014-12-07'
df['Collected'] = pd.to_datetime(df['Collected'])
df['Days_is_client'] = (df['Collected'] - df['Dt_Customer']).dt.days
# Untuk kejelasan produk/analisis lainnya
df['NumAllPurchases'] = df['NumWebPurchases']+df['NumCatalogPurchases']+df['NumStorePurchases
df['AverageCheck'] = round((df['Spent'] / df['NumAllPurchases']), 1)
df['ShareDealsPurchases'] = round((df['NumDealsPurchases'] / df['NumAllPurchases']) * 100, 1)
df['TotalAcceptedCmp'] = df['AcceptedCmp1']+df['AcceptedCmp2']+df['AcceptedCmp3']+df['Accepte


data=df.query("NumAllPurchases!=0")


df['Collected'] = '2014-12-07'
df['Collected'] = pd.to_datetime(df['Collected'])


df.head()
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer
0	5524	1957	Graduation	Single	58138.0	0	0	2012-04-09
1	2174	1954	Graduation	Single	46344.0	1	1	2014-08-03
2	4141	1965	Graduation	Together	71613.0	0	0	2013-08-21
3	6182	1984	Graduation	Together	26646.0	1	0	2014-10-02
4	5324	1981	PhD	Married	58293.0	1	0	2014-01-19

```
# Drop / Menjatuhkan beberapa fitur yang berlebihan / redundant features
df.drop(['ID', 'Year_Birth', 'Kidhome', 'Teenhome', 'Dt_Customer', 'Z_CostContact', 'Z_Revenu


df.head()
```

	Education	Marital_Status	Income	Recency	MntWines	MntFruits	MntMeatProducts	Mn

```
df['Income'] = np.where(df['Income'] > 120000, 120000, df['Income'])
df['AverageCheck'] = np.where(df['AverageCheck'] > 200, 200, df['AverageCheck'])


df["ActiveDays"]=df["Days_is_client"]-df['Recency']
```

3	Graduation	Together	26646.0	26	11	4	20	

```
df.head()
```

	Education	Marital_Status	Income	Recency	MntWines	MntFruits	MntMeatProducts	Mn
0	Graduation	Single	58138.0	58	635	88	546	
1	Graduation	Single	46344.0	38	11	1	6	
2	Graduation	Together	71613.0	26	426	49	127	
3	Graduation	Together	26646.0	26	11	4	20	
4	PhD	Married	58293.0	94	173	43	118	

LATIHAN (5)-CORRELATION

```
#correlation matrix
df.corr()
```

	Income	Recency	MntWines	MntFruits	MntMeatProducts	MntFisl
Income	1.000000	0.005380	0.705958	0.521466	0.692041	
Recency	0.005380	1.000000	0.015721	-0.005844	0.022518	
MntWines	0.705958	0.015721	1.000000	0.387024	0.568860	
MntFruits	0.521466	-0.005844	0.387024	1.000000	0.547822	
MntMeatProducts	0.692041	0.022518	0.568860	0.547822	1.000000	
MntFishProducts	0.533968	0.000551	0.397721	0.593431	0.573574	
MntSweetProducts	0.537344	0.025110	0.390326	0.571606	0.535136	
MntGoldProds	0.400995	0.017663	0.392731	0.396487	0.359446	
NumDealsPurchases	-0.115594	0.002115	0.008886	-0.134512	-0.121308	
NumWebPurchases	0.476057	-0.005641	0.553786	0.302039	0.307090	
NumCatalogPurchases	0.693520	0.024081	0.634753	0.486263	0.734127	
NumStorePurchases	0.651649	-0.000434	0.640012	0.458491	0.486006	
NumWebVisitsMonth	-0.652339	-0.018564	-0.321978	-0.418729	-0.539484	
AcceptedCmp3	-0.014412	-0.032257	0.061463	0.014424	0.018438	
AcceptedCmp4	0.225139	0.017566	0.373143	0.006396	0.091618	
AcceptedCmp5	0.406084	-0.000482	0.473550	0.212871	0.376867	
AcceptedCmp1	0.335143	-0.021061	0.351417	0.191816	0.313076	
AcceptedCmp2	0.106588	-0.001400	0.206185	-0.009980	0.043521	
Complain	-0.030810	0.013637	-0.039470	-0.005324	-0.023782	
Customer_For	-0.028761	0.030777	0.148720	0.059609	0.071345	

LATIHAN (6 & 7)

CLUSTERING

Sekarang kita telah mengurangi atribut menjadi tiga dimensi, kita akan melakukan pengelompokan melalui pengelompokan Agglomerative. Pengelompokan aglomeratif adalah metode pengelompokan hierarkis. Ini melibatkan penggabungan contoh sampai jumlah cluster yang diinginkan tercapai.

Langkah-langkah yang dilakukan dalam Clustering

- Metode Elbow untuk menentukan jumlah cluster yang akan dibentuk
- Clustering melalui Agglomerative Clustering
- Memeriksa cluster yang terbentuk melalui scatter plot

Selanjutnya, mari kita lihat korelasi di antara fitur-fiturnya.

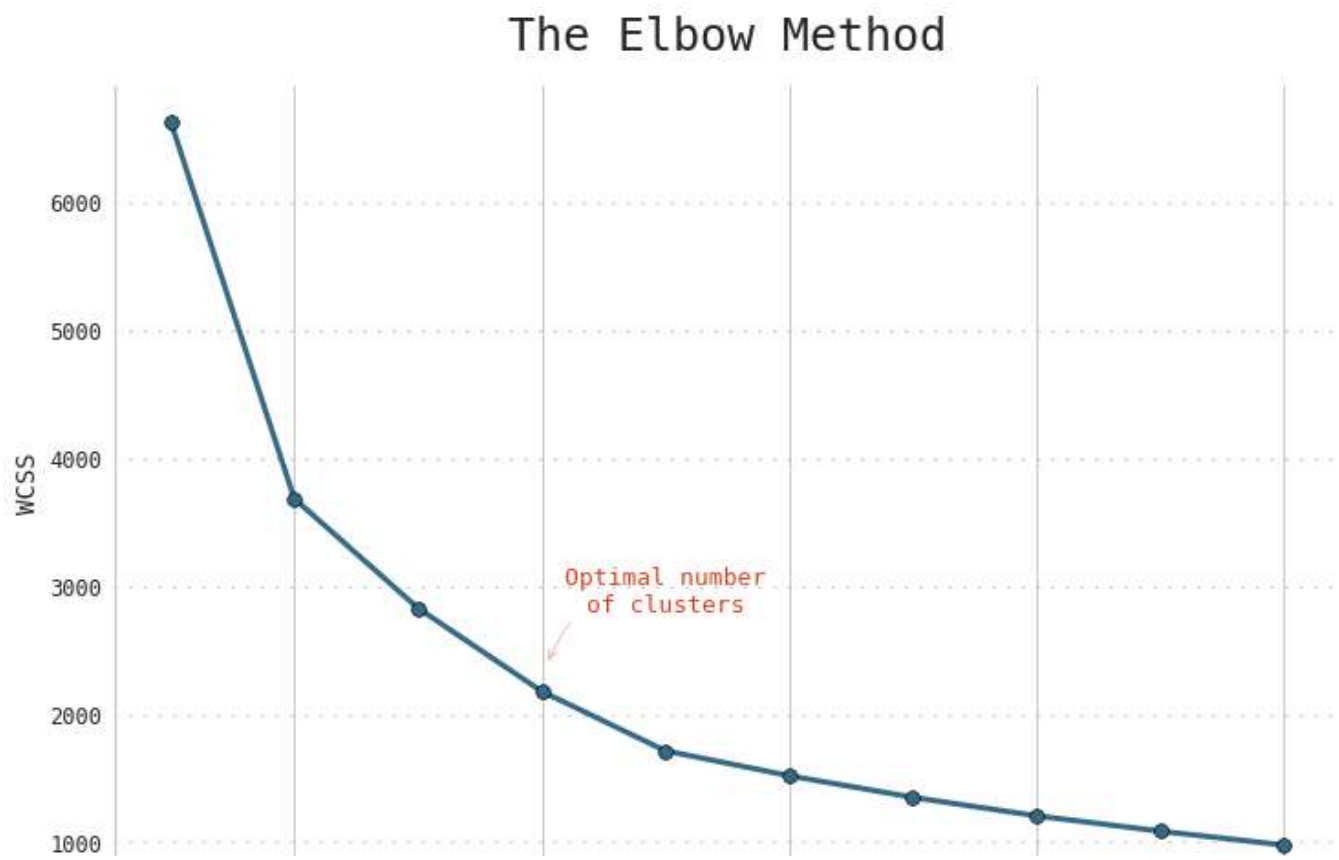
(Tidak termasuk atribut kategoris pada saat ini)

LATIHAN 8

```
data_clustring=data[['AverageCheck', 'Days_is_client', 'NumAllPurchases']].copy()
for i in data_clustring.columns:
    data_clustring[i]=StandardScaler().fit_transform(np.array(data_clustring[[i]]))
```

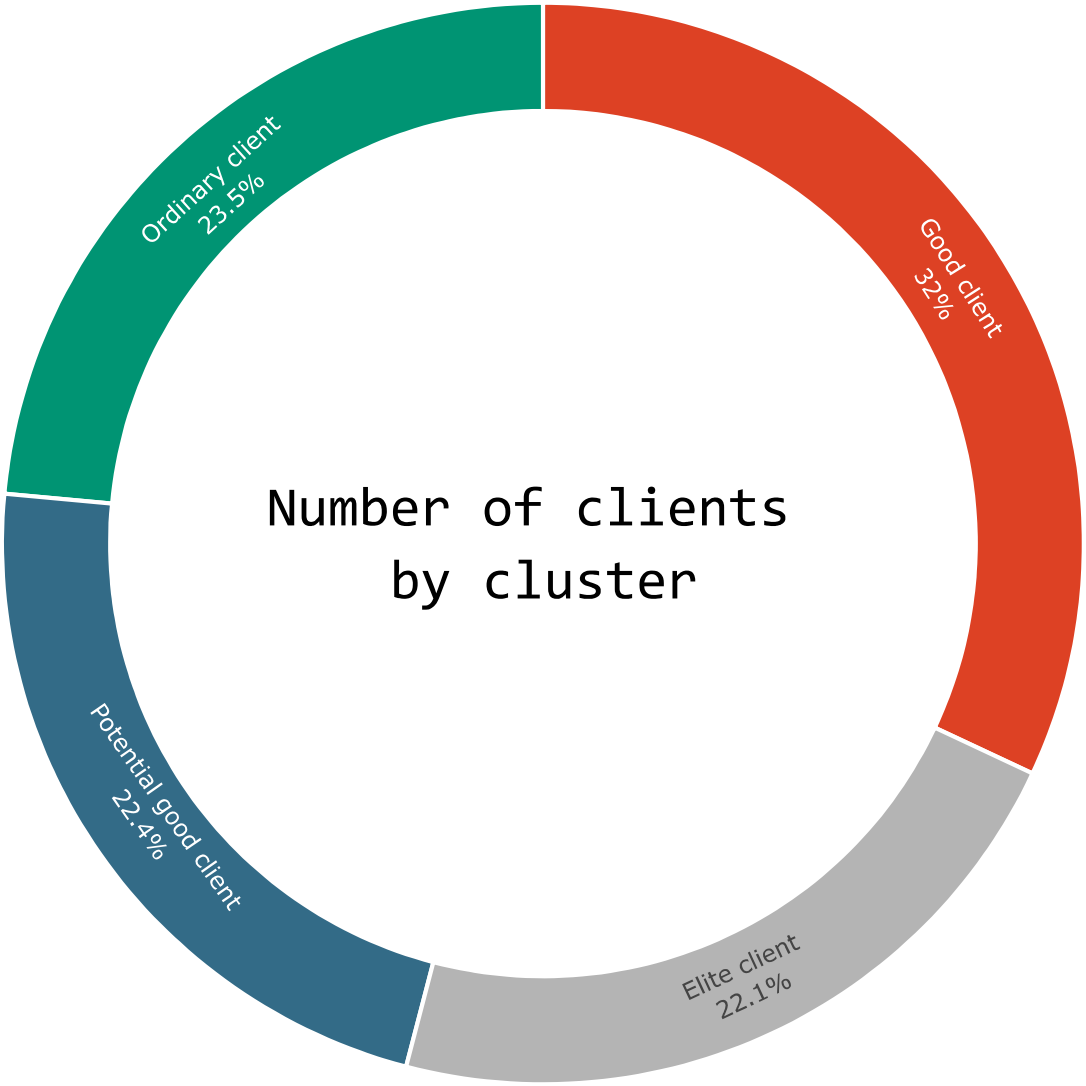
LATIHAN 9

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    km = KMeans(n_clusters = i, random_state = 228)
    km.fit(data_clustring)
    wcss.append(km.inertia_)

plt.figure(figsize = (12, 8))
plt.title('The Elbow Method', size = 25, y = 1.03, fontname = 'monospace')
plt.grid(color = 'gray', linestyle = ':', axis = 'y', alpha = 0.8, zorder = 0,  dashes = (1,7
a = sns.lineplot(x = range(1, 11), y = wcss, color = '#336b87', linewidth = 3)
sns.scatterplot(x = range(1, 11), y = wcss, color = '#336b87', s = 60, edgecolor = 'black', z
plt.ylabel('WCSS', size = 14, fontname = 'monospace')
plt.xlabel('Number of clusters', size = 14, fontname = 'monospace')
plt.xticks(size = 12, fontname = 'monospace')
plt.yticks(size = 12, fontname = 'monospace')

for j in ['right', 'top']:
    a.spines[j].set_visible(False)
a.spines['bottom'].set_linewidth(1.3)
a.spines['left'].set_linewidth(1.3)

plt.annotate('''Optimal number
of clusters''', xy = (4.05, 2400), xytext = (5, 2800),
             arrowprops = dict(facecolor = 'steelblue', arrowstyle = "->", connectionstyle =
             fontsize = 13, fontfamily = 'monospace', ha = 'center', color = '#dd4124')

plt.show()
```




```
from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components = 4, covariance_type = 'spherical', max_iter = 3000, rando
labels = gmm.predict(data_clustring)

data['Cluster'] = labels
data_re_clust = {
    0: 'Ordinary client',
    1: 'Elite client',
    2: 'Good client',
    3: 'Potential good client'
}
data['Cluster'] = data['Cluster'].map(data_re_clust)


import plotly.express as px
fig = px.pie(data['Cluster'].value_counts().reset_index(), values = 'Cluster', names = 'index
fig.update_traces(textposition = 'inside',
                  textinfo = 'percent + label',
                  hole = 0.8,
                  marker = dict(colors = ['#dd4124','#009473', '#336b87', '#b4b4b4'], line =
                  hovertemplate = 'Clients: %{value}')

fig.update_layout(annotations = [dict(text = 'Number of clients <br>by cluster',
                                      x = 0.5, y = 0.5, font_size = 28, showarrow = False,
                                      font_family = 'monospace',
                                      font_color = 'black')],
                  showlegend = False)
```

```
fig.show()
```



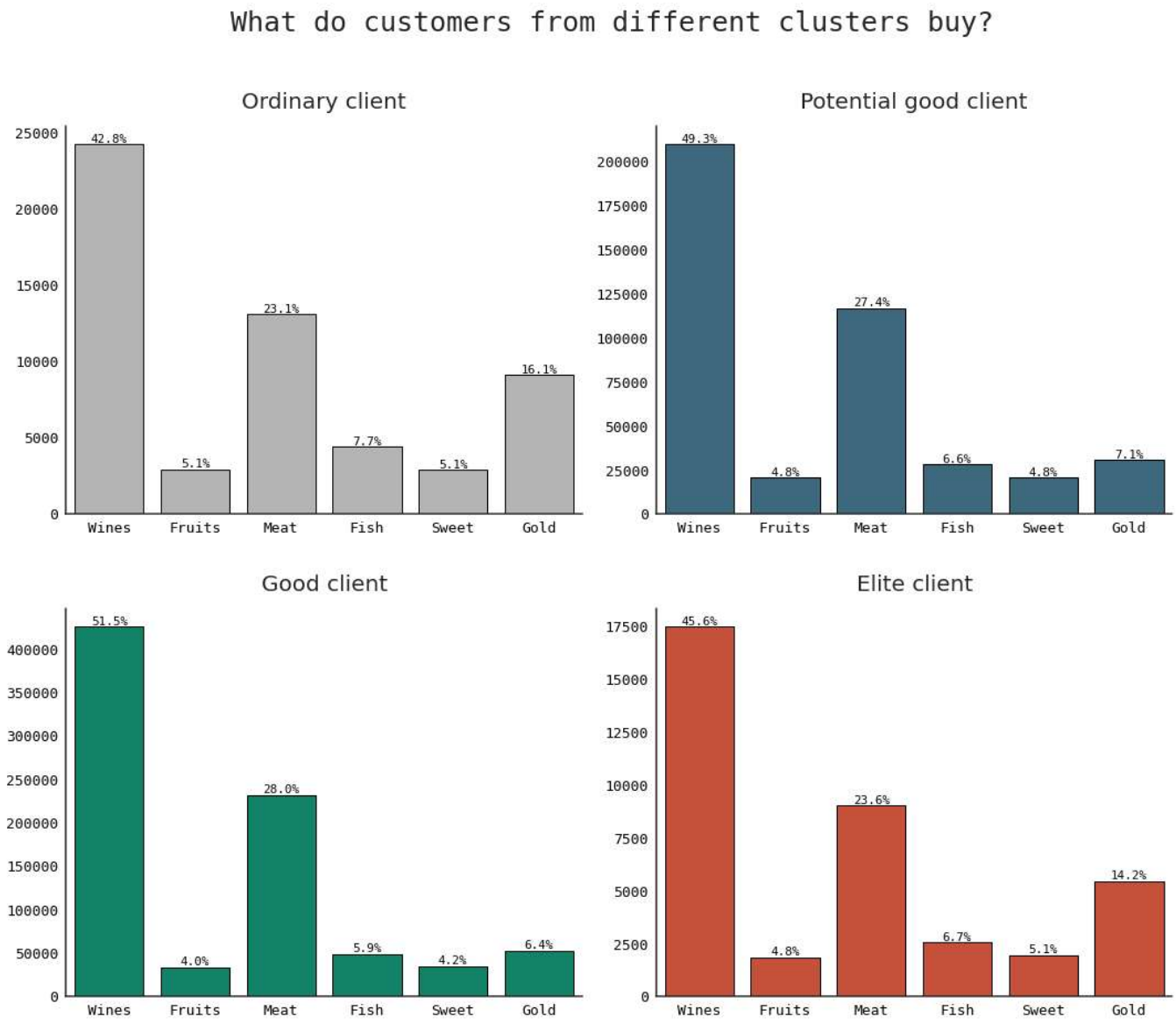
LATIHAN 10

```
import plotly.graph_objs as go
plot = go.Figure()

colors = ['#b4b4b4', '#dd4124', '#009473', '#336b87']
names = ['Ordinary client', 'Elite client', 'Good client', 'Potential good client']

for i in range(4):
    cl = names[i]
```

```
        plot.add_trace(go.Scatter3d(x = data.query("Cluster == @cl")['NumAllPurchases'],
                                    y = data.query("Cluster == @cl")['AverageCheck'],
                                    z = data.query("Cluster == @cl")['Days_is_client'],
                                    mode = 'markers',
                                    name = names[i],
                                    marker = dict(
                                        size = 2,
                                        color = colors[i],
                                        opacity = 0.6)))

    plot.update_traces(hovertemplate = 'Purchases: %{x} <br>Average Check: %{y} <br>Days is clien

    plot.update_layout(width = 800, height = 800, autosize = True, showlegend = False,
                       scene = dict(xaxis = dict(title = 'Count of purchases', titlefont_color =
                                    yaxis = dict(title = 'Average check', titlefont_color = 'blac
                                    zaxis = dict(title = 'Days is client', titlefont_color = 'bla
                       font = dict(family = "monospace", color  = 'black', size = 12),
                       title_text = 'Customers clusters', title_x = 0.5)
```

Customers clusters


```
fig = plt.figure(figsize = (15, 12))
k = 1

for i in cl:
    ass = data.groupby(['Cluster']).agg({'Wines': 'sum', 'Fruits': 'sum', 'Meat': 'sum', 'Fis
    plt.subplot(2, 2, k)
    plt.title(i, size = 20, x = 0.5, y = 1.03)
    a = sns.barplot(data = ass, x = 'Category', y = i, color = colors[i],
                    linestyle = "-", linewidth = 1,
                    edgecolor = "black")
    plt.xticks(fontname = 'monospace', size = 13, color = 'black')
    plt.yticks(fontname = 'monospace', size = 13, color = 'black')
    plt.xlabel('')
    plt.ylabel('')
    for p in a.patches:
        height = p.get_height()
        a.annotate(f'{round((height / sum(ass[i])) * 100, 1)}%', (p.get_x() + p.get_width() /
                   ha = 'center', va = 'center',
                   size = 11,
                   xytext = (0, 5),
                   textcoords = 'offset points',
                   fontname = 'monospace', color = 'black')

    for j in ['right', 'top']:
        a.spines[j].set_visible(False)
    for j in ['bottom', 'left']:
        a.spines[j].set_linewidth(1.5)
    k += 1

plt.figtext(0.05, -0.05, 'Conclusion', fontname = 'monospace', size = 22, color = '#dd4124')
plt.figtext(0.05, -0.14, '''We are definitely dealing with a store that sells very good wine,
of buyers. In general, there are no major differences, except that customers from ordinary an
good clusters are more likely to buy gold and customers from good and elite clusters are more
buy meat more often.''', fontname = 'monospace', size = 14, color = 'black')

plt.figtext(0.2, 1.05, 'What do customers from different clusters buy?', fontname = 'monospac
fig.tight_layout(h_pad = 3)
plt.show()
```



Conclusion

We are definitely dealing with a store that sells very good wine, which is almost equally bought by all clusters of buyers. In general, there are no major differences, except that customers from ordinary and potential good clusters are more likely to buy gold and customers from good and elite clusters are more likely to buy meat more often.

```
fig = plt.figure(figsize = (15, 10))
plt.title('Participation of customer clusters in marketing campaigns', size = 25, x = 0.5, y
plt.grid(color = 'gray', linestyle = ':', axis = 'y', alpha = 0.8, zorder = 0,  dashes = (1,7
a = sns.barplot(x = 'Cmp', y = 'value', hue = 'Cluster',
            data = data.groupby(['Cluster']).agg({'AcceptedCmp1': 'sum','AcceptedCmp2': 'sum'
                                'AcceptedCmp3': 'sum','AcceptedCmp4': 'sum',
                                'AcceptedCmp5': 'sum', 'Response': 'sum'}).stack().reset_index
            dodge = False, palette = ['#dd4124', '#009473', '#b4b4b4', '#336b87'])
plt.xticks(fontname = 'monospace', size = 16, color = 'black')
plt.yticks(fontname = 'monospace', size = 16, color = 'black')
plt.xlabel('')
plt.ylabel('')
for j in ['right', 'top']:
    a.spines[j].set_visible(False)
for j in ['bottom', 'left']:
    a.spines[j].set_linewidth(1.5)

plt.figtext(0.12, 0.01, 'Conclusion', fontname = 'monospace', size = 22, color = '#dd4124')
plt.figtext(0.12, -0.14, '''The best marketing campaigns were the first, fifth and last. The
campaign. The cluster of elite clients shows the greatest activity. This is a rather strange
trend, because clients with low income are more likely to participate in promotions. Ordinary
customers were not generally interested in any campaign. I dare to assume that we have
information about a fairly good store for wealthy people. In the future, based on these
results, company can plan campaigns more correctly.''', fontname = 'monospace', size = 14, co

plt.show()
```

```
fig = plt.figure(figsize = (15, 10))
plt.title('Participation of customer clusters in marketing campaigns', size = 25, x = 0.5, y
plt.grid(color = 'gray', linestyle = ':', axis = 'y', alpha = 0.8, zorder = 0,  dashes = (1,7
a = sns.barplot(x = 'Cmp', y = 'value', hue = 'Cluster',
            data = data.groupby(['Cluster']).agg({'AcceptedCmp1': 'sum','AcceptedCmp2': 'sum'
                                'AcceptedCmp3': 'sum','AcceptedCmp4': 'sum',
                                'AcceptedCmp5': 'sum', 'Response': 'sum'}).stack().reset_index
            dodge = False, palette = ['#dd4124', '#009473', '#b4b4b4', '#336b87'])
plt.xticks(fontname = 'monospace', size = 16, color = 'black')
plt.yticks(fontname = 'monospace', size = 16, color = 'black')
plt.xlabel('')
plt.ylabel('')
for j in ['right', 'top']:
    a.spines[j].set_visible(False)
for j in ['bottom', 'left']:
    a.spines[j].set_linewidth(1.5)

plt.figtext(0.12, 0.01, 'Conclusion', fontname = 'monospace', size = 22, color = '#dd4124')
plt.figtext(0.12, -0.14, '''The best marketing campaigns were the first, fifth and last. The
```

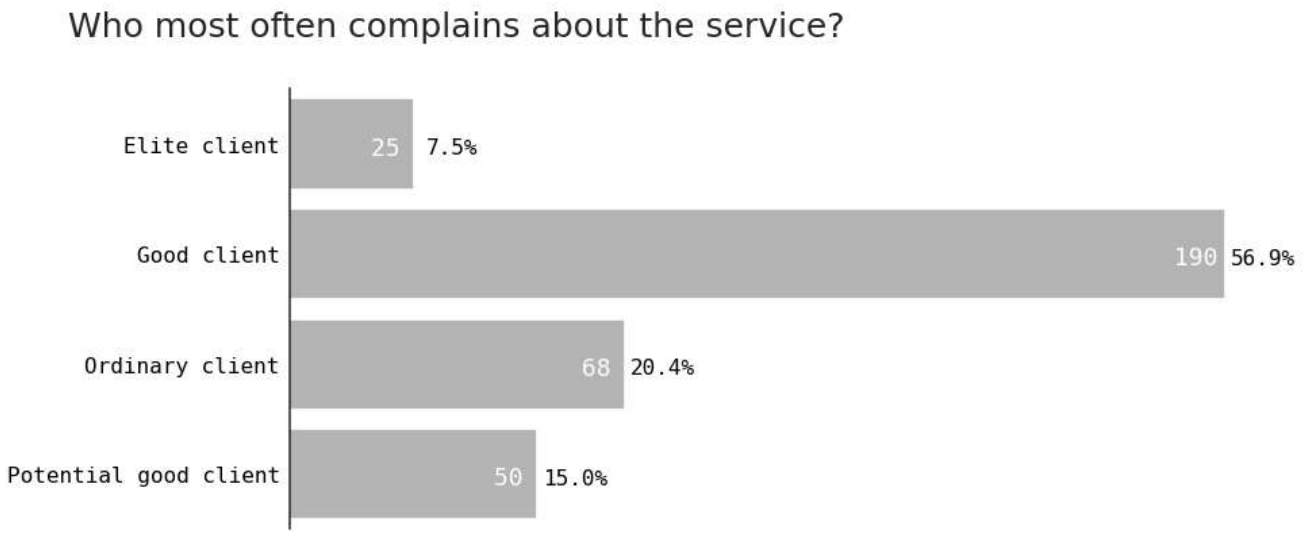
```
campaign. The cluster of elite clients shows the greatest activity. This is a rather strange
trend, because clients with low income are more likely to participate in promotions. Ordinary
customers were not generally interested in any campaign. I dare to assume that we have
information about a fairly good store for wealthy people. In the future, based on these
results, company can plan campaigns more correctly.''', fontname = 'monospace', size = 14, co

plt.show()
```

Participation of customer clusters in marketing campaigns

Sekarang kita memiliki beberapa fitur baru, mari kita lihat statistik data.

```
fig = plt.figure(figsize = (13, 6))
plt.title('Who most often complains about the service?', size = 25, x = 0.17, y = 1.1)
a = sns.barplot(data = data.groupby(['Cluster']).agg({'Response': 'sum'}).reset_index(),
                x = 'Response', y = 'Cluster')
plt.xticks([])
plt.yticks(fontname = 'monospace', size = 16, color = 'black')
plt.xlabel('')
plt.ylabel('')

for p in a.patches:
    width = p.get_width()
    plt.text(width - 7, p.get_y() + 0.55*p.get_height(), f'{width: .0f}',
             ha = 'center', va = 'center', fontname = 'monospace', fontsize = 18, color = 'wh
    plt.text(8 + width, p.get_y() + 0.55*p.get_height(), f'{round((width / 334) * 100, 1)}%',
             ha = 'center', va = 'center', fontname = 'monospace', fontsize = 16, color = 'bl
    if p.get_width() == 189:
        p.set_color('#dd4124')
    elif p.get_width() == 54:
        p.set_color('#009473')
    elif p.get_width() == 66:
        p.set_color('#336b87')
    else:
        p.set_color('#b4b4b4')
for j in ['right', 'top', 'bottom']:
    a.spines[j].set_visible(False)
a.spines['left'].set_linewidth(1.5)

l1 = lines.Line2D([-0.1, 0.95], [0, 0], transform = fig.transFigure, figure = fig, color = 'b
fig.lines.extend([l1])

plt.figtext(-0.05, -0.1, 'What is the percentage of dissatisfied customers for each cluster?'

x = 0
cl = 0
colors = ['#dd4124', '#009473', '#336b87', '#b4b4b4']
for i in round(data.groupby(['Cluster']).agg({'Response': 'mean'}).reset_index()['Response']
    plt.figtext(x, -0.18, f'{i}%', fontname = 'monospace', size = 23, color = colors[cl])
    x += 0.25
    cl += 1

l2 = lines.Line2D([-0.1, 0.95], [-0.25, -0.25], transform = fig.transFigure, figure = fig, co
fig.lines.extend([l2])

plt.figtext(-0.05, -0.33, 'Conclusion', fontname = 'monospace', size = 23, color = 'black')
plt.figtext(-0.05, -0.48, '''More than half of the complaints come from a cluster of elite cu
from this cluster have complained in the last two years, which is a big indicator.
```

```
The number of complaints for the remaining clusters is within the normal range.''', fontname

plt.show()
```



What is the percentage of dissatisfied customers for each cluster?

5.1% 26.9% 13.1% 10.1%

Conclusion

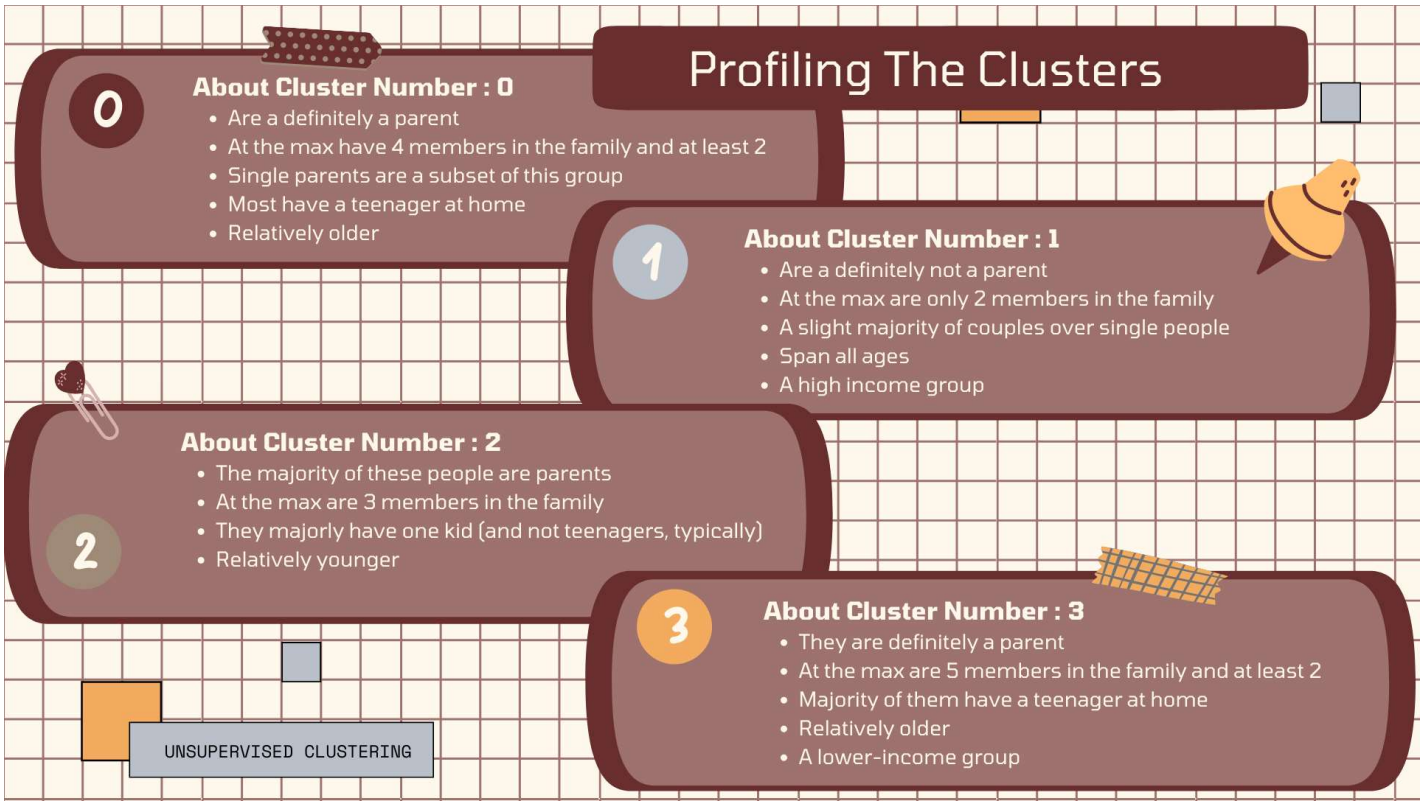
More than half of the complaints come from a cluster of elite customers, more than 1/4 of the customers from this cluster have complained in the last two years, which is a big indicator.
The number of complaints for the remaining clusters is within the normal range.

```
df.head()
```

```
    Education  Marital_Status   Income  Recency  MntWines  MntFruits  MntMeatProducts  Mn
```

Hal-hal yang perlu diperhatikan:

Informasi berikut dapat disimpulkan tentang pelanggan di cluster yang berbeda.



CONCLUSION

Dalam kasus ini, kita melakukan unsupervised clustering. Kita memang menggunakan pengurangan dimensi/dimensionality reduction diikuti oleh agglomerative clustering. Kita datang dengan 4 cluster dan selanjutnya menggunakannya dalam membuat profil pelanggan dalam cluster sesuai dengan struktur keluarga dan pendapatan/pengeluaran mereka(income/spending).

Ini dapat digunakan dalam merencanakan strategi pemasaran yang lebih baik!

Terimakasih Semoga dapat menambah akan pemahaman kalian!

END

✓ 0 d selesai pada 23.35