18/
12/
24.

DETEKSI AKTIVITAS TRANSAKSI ANOMALI

CLUSTERING DENGAN DBSCAN

GCD - IF A PAGI

TUJUAN

Mendeteksi aktivitas transaksi yang tidak wajar

Contoh: pencucian uang, rekening transaksi bodong atau penyalahgunaan rekening

Membagi nasabah dalam kelompok tertentu

Contoh: penawaran promosi

DATASET

SUMBER DATASET

kaggle.com/datasets/shivamb/bank-customer-segmentation

UKURAN DATASET

1.048.567 baris data, 9 kolom fitur

FITUR-FITUR

TransactionID, CustomerID, CustomerDOB, CustGender, CustLocation, CustAccountBalance, TransactionDate, TransactionTime, TransactionAmount (INR)

[7]:
```
raw_data = pd.read_csv("bank_transactions.csv")
raw_data.sample(frac = 1).head()
```

[7]:

	TransactionID	CustomerID	CustomerDOB	CustGender	CustLocation	CustAccountBalance	TransactionDate	TransactionTime	TransactionAmount (INR)
313104	T313105	C9018770	22/10/91	F	GURGAON	161.23	10/8/16	163943	3602.0
617525	T617526	C8816635	12/6/96	M	HOSHIARPUR	20643.59	26/8/16	134531	810.0
4801	T4802	C3439934	19/10/65	M	NEW DELHI	3293.25	22/9/16	145327	557.0
136431	T136432	C6216950	1/1/1800	M	NEW DELHI	446739.29	5/8/16	122615	1100.0
573848	T573849	C4731579	10/8/89	M	BURDWAN	9019.95	22/8/16	112342	150.0

GCD - IF A PAGI

DATASET PREPROCESSING

Drop “NaN” and invalid value

Before

```
[10]: # cek apakah ada dataset yang hilang
      raw_data.isna().sum()

[10]: TransactionID              0
      CustomerID                 0
      CustomerDOB             3397
      CustGender              1100
      CustLocation             151
      CustAccountBalance      2369
      TransactionDate            0
      TransactionTime            0
      TransactionAmount (INR)    0
      dtype: int64
```

```
[8]: raw_data.shape

[8]: (1048567, 9)
```

After

```
[12]: # Drop row(s) with nan value
      preprocessed_data = raw_data.dropna()
      display(preprocessed_data.isna().sum())

      TransactionID              0
      CustomerID                 0
      CustomerDOB                0
      CustGender                 0
      CustLocation               0
      CustAccountBalance         0
      TransactionDate            0
      TransactionTime            0
      TransactionAmount (INR)    0
      dtype: int64
```

```
[13]: preprocessed_data.shape

[13]: (1041614, 9)
```

DATASET PREPROCESSING

Drop “NaN” and invalid value

[15]:
```python
for data in preprocessed_data["CustomerDOB"].unique():
  date, month, year = data.split("/")
  if int(year) > 100:  # remove more than two digit year
    print(data)
    preprocessed_data = preprocessed_data[preprocessed_data["CustomerDOB"] != data]
```

```
1/1/1800
```

[16]:
```python
preprocessed_data.drop(preprocessed_data[preprocessed_data['CustGender']=='T'].index, inplace=True)
preprocessed_data.shape
```

[16]:
```
(985322, 9)
```

GCD - IF A PAGI

DATASET PREPROCESSING

Format Data Type & Feature Encoding

```
Nama kolom: CustomerID (884265, object)
['C1010011' 'C1010012' 'C1010014' ... 'C9099919' 'C9099941' 'C9099956']
Nama kolom: CustomerDOB (17255, object)
['10/1/94' '4/4/57' '26/11/96' ... '18/7/65' '15/5/42' '24/10/44']
Nama kolom: CustGender (4, object)
['F' 'M' nan 'T']
Nama kolom: CustLocation (9356, object)
['JAMSHEDPUR' 'JHAJJAR' 'MUMBAI' ... 'KARANJIA'
 'NR HERITAGE FRESH HYDERABAD' 'IMPERIA THANE WEST']
Nama kolom: CustAccountBalance (161329, float64)
[0.00000000e+00 1.00000000e-02 3.00000000e-02 ... 6.97993296e+07
 8.22446299e+07 1.15035495e+08]
Nama kolom: TransactionDate (55, object)
['1/8/16' '1/9/16' '10/8/16' '10/9/16' '11/8/16' '11/9/16' '12/8/16'
 '12/9/16' '13/8/16' '13/9/16' '14/8/16' '14/9/16' '15/8/16' '15/9/16'
 '16/10/16' '16/8/16' '17/8/16' '18/8/16' '18/9/16' '19/8/16' '2/8/16'
 '2/9/16' '20/8/16' '21/10/16' '21/8/16' '22/8/16' '22/9/16' '23/8/16'
 '23/9/16' '24/8/16' '25/8/16' '25/9/16' '26/8/16' '26/9/16' '27/8/16'
 '27/9/16' '28/8/16' '29/8/16' '3/8/16' '3/9/16' '30/8/16' '30/9/16'
 '31/8/16' '4/8/16' '4/9/16' '5/8/16' '5/9/16' '6/8/16' '6/9/16' '7/8/16'
 '7/9/16' '8/8/16' '8/9/16' '9/8/16' '9/9/16']
Nama kolom: TransactionTime (81918, int64)
[     0      1      2 ... 235957 235958 235959]
Nama kolom: TransactionAmount (INR) (93024, float64)
[0.00000000e+00 1.00000000e-02 2.00000000e-02 ... 9.91132220e+05
 1.38000288e+06 1.56003499e+06]
```

DATASET PREPROCESSING

Format Data Type & Feature Encoding

[19]:
```
# convert type of columns TransactionDate, CustomerDOB from string to datetime
preprocessed_data['TransactionDate'] = pd.to_datetime(preprocessed_data['TransactionDate'], format='%d/%m/%y', errors='coerce')
preprocessed_data['CustomerDOB'] = pd.to_datetime(preprocessed_data['CustomerDOB'], format='%d/%m/%y', errors='coerce')

# encode 'CustGender' to numeric format (F = 0, M = 1)
preprocessed_data['CustGender'] = preprocessed_data['CustGender'].map({'F': 0, 'M': 1})

# calculate 'CustomerAge' based on 'CustomerDOB'
preprocessed_data['CustomerAge'] = preprocessed_data['TransactionDate'].dt.year - preprocessed_data['CustomerDOB'].dt.year
preprocessed_data = preprocessed_data[preprocessed_data['CustomerAge'] > 0]

np.array(sorted(preprocessed_data['CustomerAge'].unique()))
```

[19]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47], dtype=int32)
```

DATASET PREPROCESSING

Make/ Group Transaction Summary by Customer

[48]:
```
# Calculate transaction frequency, total, and average per customer
transaction_summary = preprocessed_data.groupby('CustomerID').agg({
    # 'TransactionAmount (INR)': ['count', 'sum', 'mean'],
    'TransactionAmount (INR)': ['count', 'sum'],
    'TransactionDate': ['min', 'max'],
}).reset_index()

# Flatten the MultiIndex columns
transaction_summary.columns = [
    'CustomerID',
    'TransactionCount',
    'TotalAmount',
    # 'AverageAmount',
    'FirstTransactionDate',
    'LastTransactionDate',
]

# No need to calculate Age cause it's already in the dataset
transaction_summary['Age'] = preprocessed_data.groupby('CustomerID')['CustomerAge'].first().values

# Add customer gender
transaction_summary['Gender'] = preprocessed_data.groupby('CustomerID')['CustGender'].first().values

# Add customer account balance
transaction_summary['AccBalance'] = preprocessed_data.groupby('CustomerID')['CustAccountBalance'].first().values

# Add recency (days since last transaction)
transaction_summary['Recency'] = transaction_summary['LastTransactionDate'].apply(
    lambda date: (datetime.now() - pd.to_datetime(date)).days
)

# Group by customer and calculate the date range
transaction_summary['DayRange'] = (transaction_summary['LastTransactionDate'] - transaction_summary['FirstTransactionDate']).dt.days
```

DATASET PREPROCESSING

Make/ Group Transaction Summary by Customer

[18]:
```
# Check the results
transaction_summary.head(10)
```

[18]:

	CustomerID	TransactionCount	TotalAmount	FirstTransactionDate	LastTransactionDate	Age	Gender	AccBalance	Recency	DayRange
0	C1010011	2	5106.0	2016-08-09	2016-09-26	24	0	32500.73	3003	48
1	C1010012	1	1499.0	2016-08-14	2016-08-14	22	1	24204.49	3046	0
2	C1010014	2	1455.0	2016-08-01	2016-08-07	24	0	38377.14	3053	6
3	C1010018	1	30.0	2016-09-15	2016-09-15	26	0	496.18	3014	0
4	C1010028	1	557.0	2016-08-29	2016-08-29	28	0	296828.37	3031	0
5	C1010031	2	1864.0	2016-08-03	2016-08-04	32	1	1754.10	3056	1
6	C1010035	2	750.0	2016-08-01	2016-08-27	24	1	7284.42	3033	26
7	C1010036	1	208.0	2016-08-26	2016-08-26	20	1	355430.17	3034	0
8	C1010037	1	19680.0	2016-08-09	2016-08-09	35	1	95859.17	3051	0
9	C1010038	1	100.0	2016-09-07	2016-09-07	24	0	1290.76	3022	0

DATASET PREPROCESSING

Feature Selection

[19]:

```python
# Select features for clustering
columns = [
  'Age',
  # 'Gender',
  # 'DayRange',
  'AccBalance',
  # 'TransactionCount',
  'TotalAmount',
  # 'AverageAmount',
  # 'Recency',
]
features = transaction_summary[columns]
features.sample(frac = 1).head()
```

[19]:

	Age	AccBalance	TotalAmount
435550	40	86685.34	400.00
139609	21	3972.83	4200.00
592661	32	8897.50	201.73
465142	22	11719.55	90.00
534332	23	13232.97	400.00

Use 3% Subset of Dataset

[20]:

```python
# Process only 3% of data cause lack of resource
features = features.sample(frac = 0.03, random_state = 1000)
features.head()
```

[22]:

```python
data_length = len(scaled_features)
data_length
```

[22]: 23988

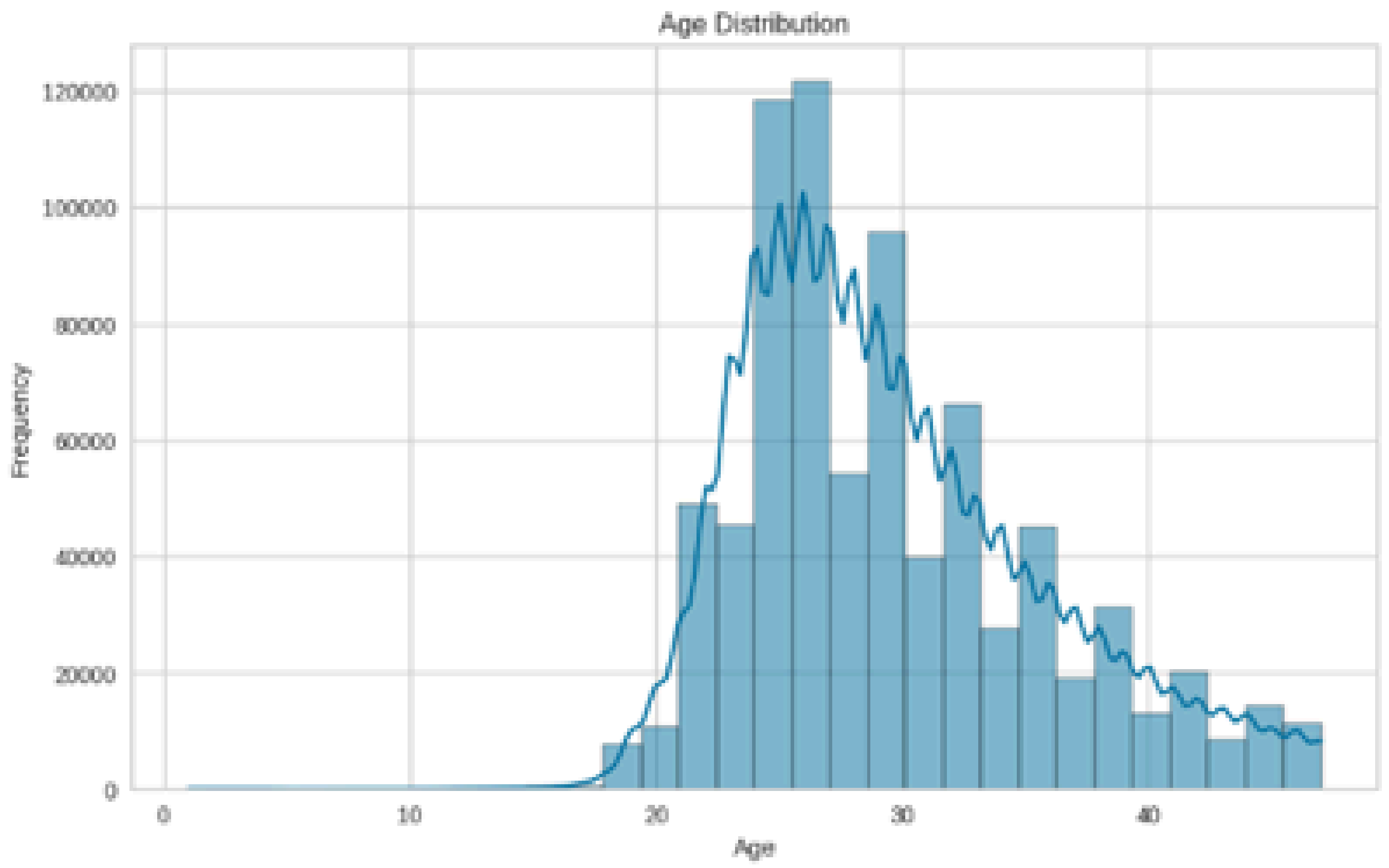
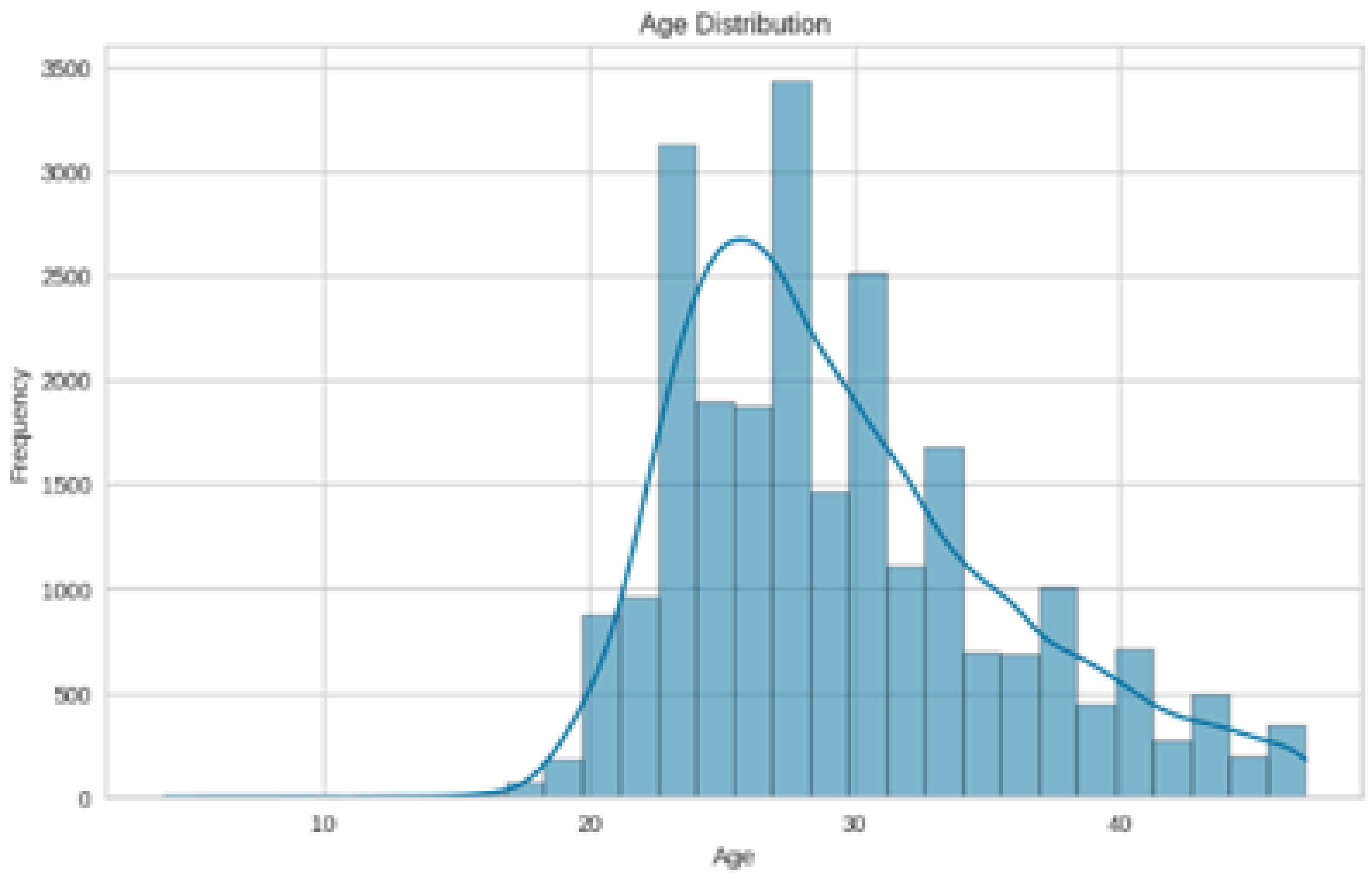
DATASET PREPROCESSING

Normalization (Standarization)

```
[20]: # Scale features
      scaler = StandardScaler()
      scaled_features = scaler.fit_transform(features)

      scaled_features[:5]
```

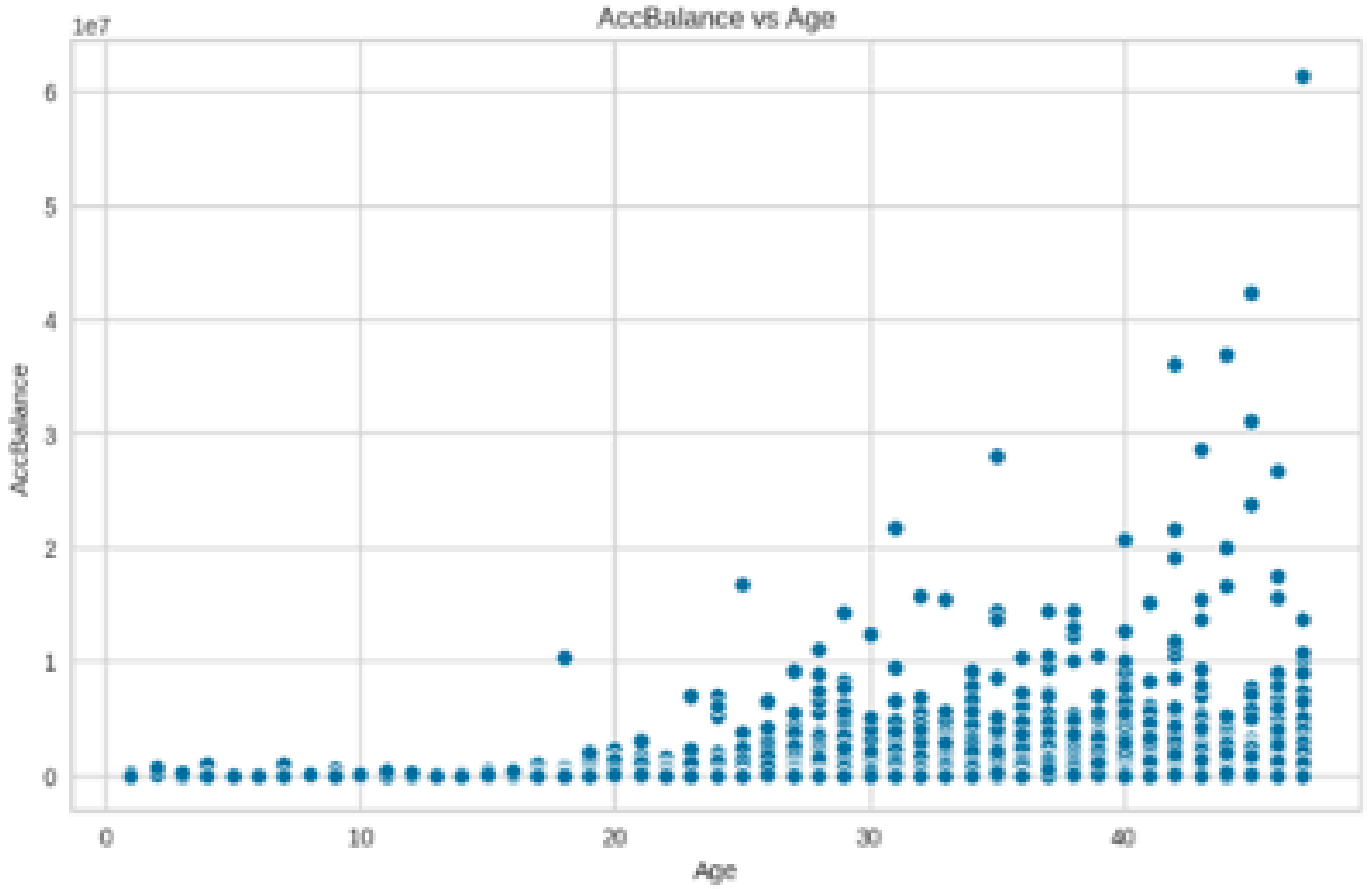
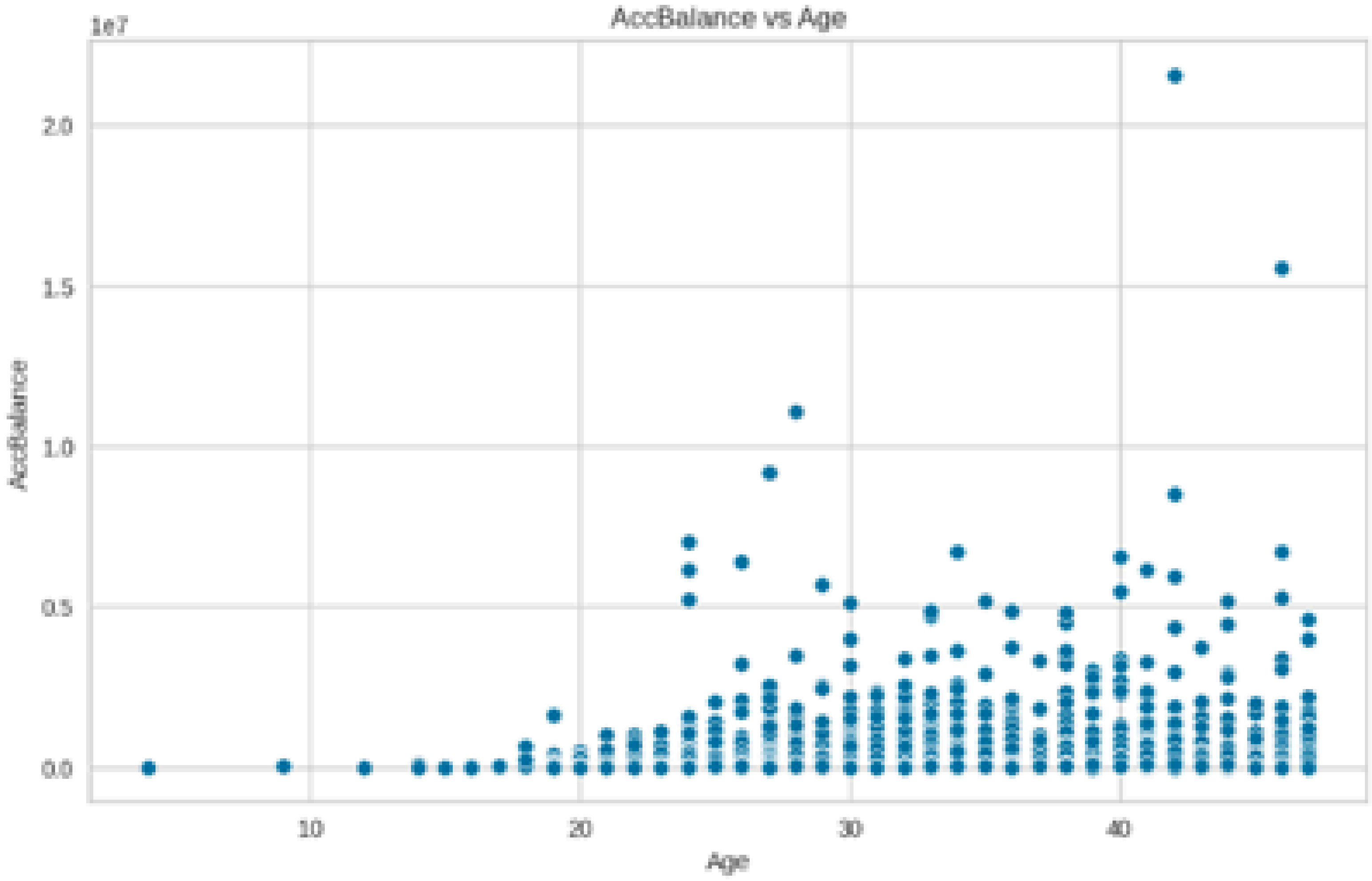
```
[20]: array([[-0.90450846, -1.61764198,  0.07265365,  1.98048597,  0.55887532],
             [-1.23043586,  0.61818376, -0.14032966, -0.39000265, -0.01252418],
             [-0.90450846, -1.61764198,  0.16512084,  1.98048597, -0.01949439],
             [-0.57858106, -1.61764198, -0.19294148, -0.39000265, -0.24523442],
             [-0.25265366, -1.61764198,  0.46465811, -0.39000265, -0.16175022]])
```

PERBANDINGAN DISTRIBUSI DATA

<table>
<tr><th>Keseluruhan Dataset</th><th>3% dari Dataset</th></tr>
<tr><td colspan="2">Distribusi Umur</td></tr>
<tr><td>
</td><td>
</td></tr>
</table>

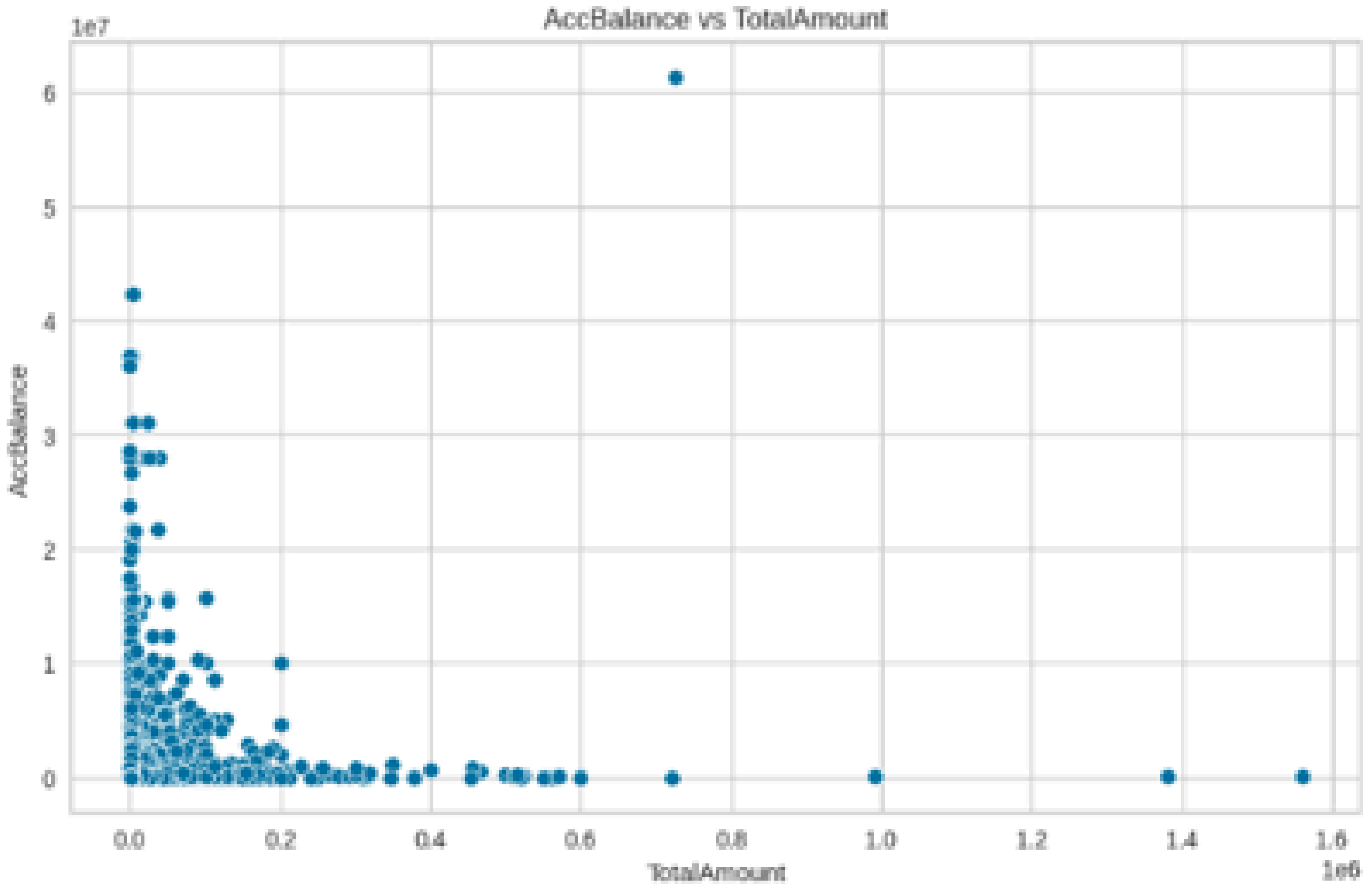
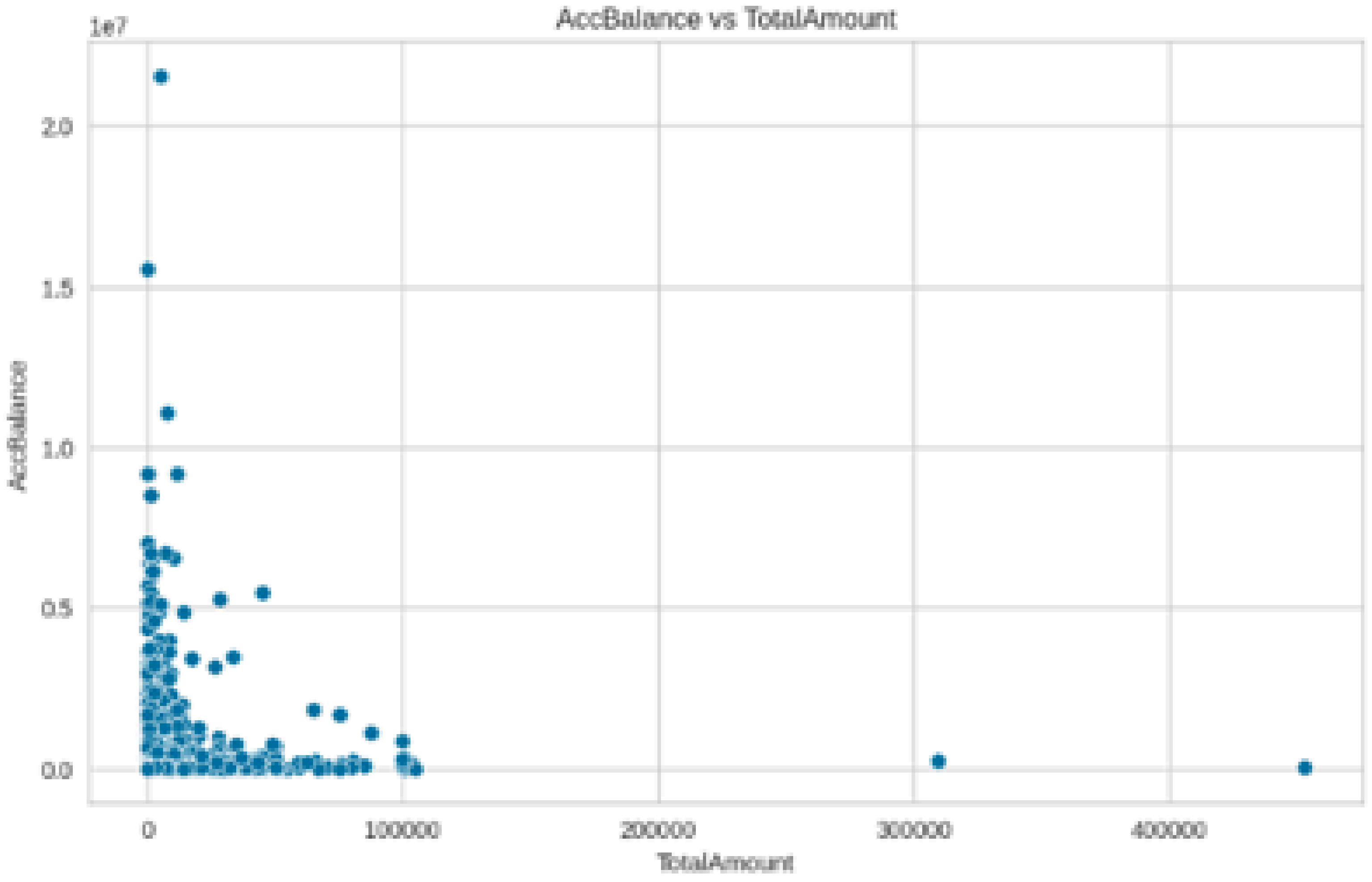
GCD - IF A PAGI

PERBANDINGAN DISTRIBUSI DATA

<table>
<tr><th>Keseluruhan Dataset</th><th>3% dari Dataset</th></tr>
<tr><td colspan="2">Distribusi Saldo vs Umur</td></tr>
<tr><td>
</td><td>
</td></tr>
</table>

GCD - IF A PAGI

PERBANDINGAN DISTRIBUSI DATA

<table>
<tr><th>Keseluruhan Dataset</th><th>3% dari Dataset</th></tr>
<tr><td colspan="2">Distribusi Saldo vs Total Transaksi</td></tr>
<tr><td>
</td><td>
</td></tr>
</table>

GCD - IF A PAGI

MODELLING

PENCARIAN HYPERPARAMETER TERBAIK

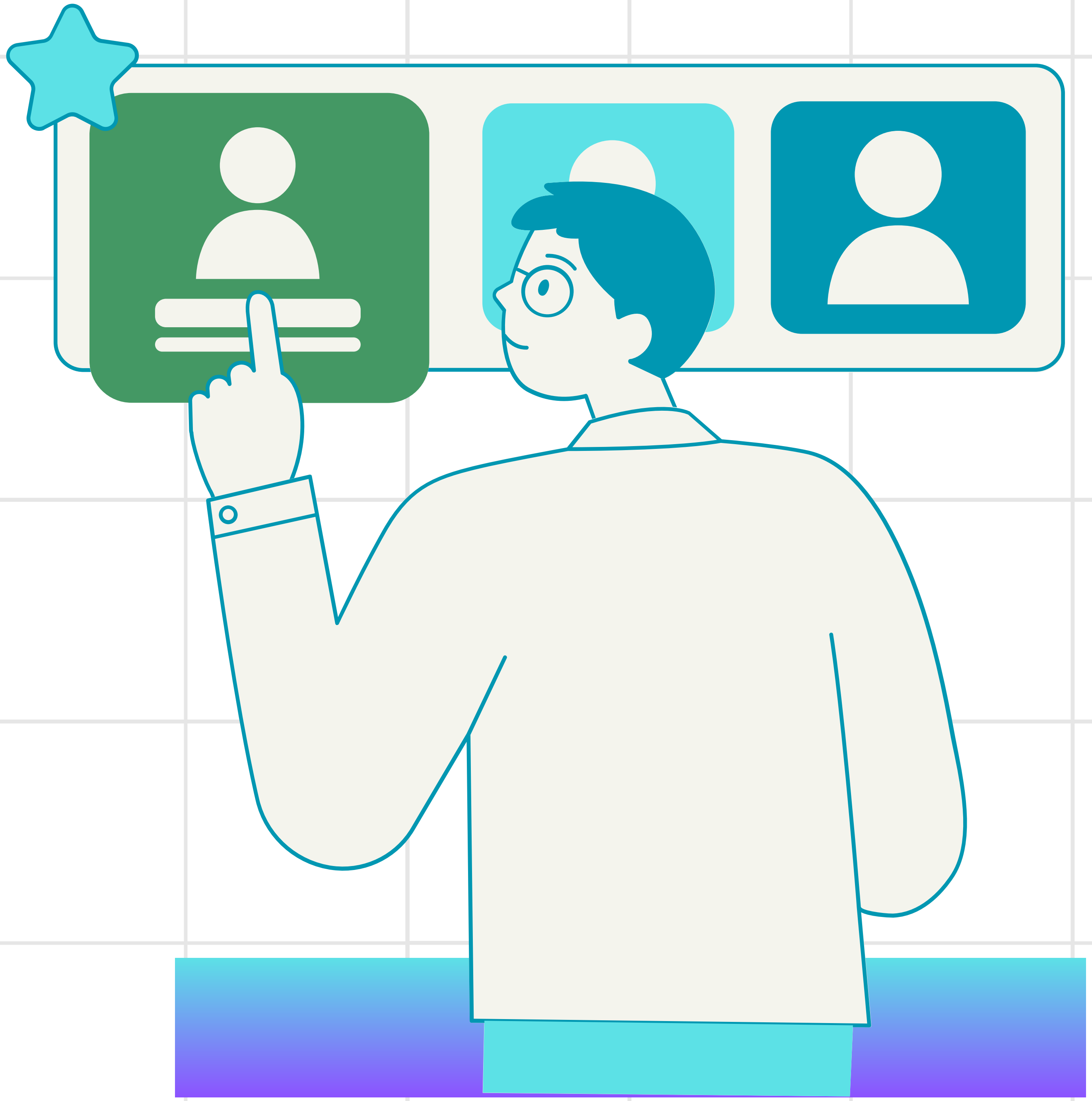
```
eps_range = np.arange(0.5, 3.5, 0.5)  # Example range for eps
min_samples_range = np.arange(10, 30, 5)  # Example range for min_samples

results = []

for eps in eps_range:
  for min_samples in min_samples_range:
    dbscan = DBSCAN(eps=eps, min_samples=min_samples)
    dbscan.fit(scaled_features)

    # label cluster
    labels_dbscan = dbscan.labels_

    # menghitung jumlah elemen dari tiap cluster
    print(f"{eps}, {min_samples}; Done, Next", end=" --> ")
    if len(np.unique(labels_dbscan)) > 1:  # Check if more than one cluster is found
      # Remove the 'ignore_index' argument as it's not supported by silhouette_score
      score = silhouette_score(scaled_features, labels_dbscan)
      results.append([eps, min_samples, score, np.unique(labels_dbscan, return_counts=True)])
print("Finish.")

result_df = pd.DataFrame(results, columns=['eps', 'min_samples', 'silhouette_score', 'clusters'])
display(result_df)
```

```
0.5, 10; Done, Next --> 0.5, 15; Done, Next --> 0.5, 20; Done, Next --> 0.5, 25; Done, Next --> 1.0, 10; Done, Next --> 1.0, 15; Done, Next --> 1.0,
20; Done, Next --> 1.0, 25; Done, Next --> 1.5, 10; Done, Next --> 1.5, 15; Done, Next --> 1.5, 20; Done, Next --> 1.5, 25; Done, Next --> 2.0, 10; D
one, Next --> 2.0, 15; Done, Next --> 2.0, 20; Done, Next --> 2.0, 25; Done, Next --> 2.5, 10; Done, Next --> 2.5, 15; Done, Next --> 2.5, 20; Done,
Next --> 2.5, 25; Done, Next --> 3.0, 10; Done, Next --> 3.0, 15; Done, Next --> 3.0, 20; Done, Next --> 3.0, 25; Done, Next --> Finish.
```

MODELLING

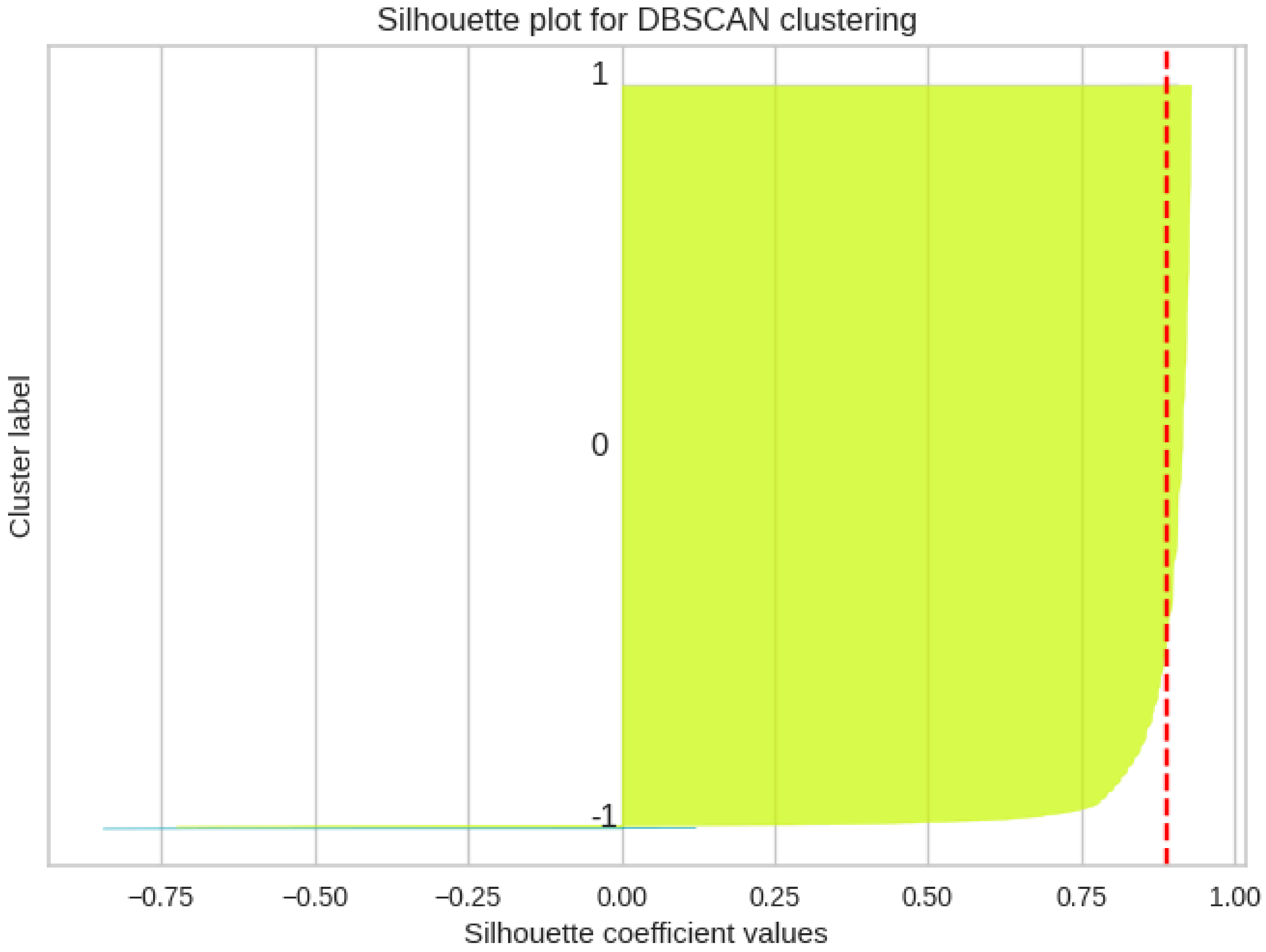
PERBANDINGAN SILHOUETTE SCORE

	eps	min_samples	silhouette_score	clusters
0	0.5	10	0.695793	([-1, 0, 1, 2, 3], [316, 23623, 31, 8, 10])
1	0.5	15	0.803552	([-1, 0], [425, 23563])
2	0.5	20	0.789556	([-1, 0], [505, 23483])
3	0.5	25	0.780932	([-1, 0], [561, 23427])
4	1.0	10	0.889953	([-1, 0], [126, 23862])
5	1.0	15	0.881662	([-1, 0], [150, 23838])
6	1.0	20	0.876581	([-1, 0], [164, 23824])
7	1.0	25	0.870994	([-1, 0], [181, 23807])
8	1.5	10	0.888455	([-1, 0, 1], [57, 23918, 13])
9	1.5	15	0.912663	([-1, 0], [78, 23910])
10	1.5	20	0.902984	([-1, 0], [99, 23889])
11	1.5	25	0.896720	([-1, 0], [116, 23872])
12	2.0	10	0.930488	([-1, 0], [41, 23947])
13	2.0	15	0.888391	([-1, 0, 1], [49, 23924, 15])
14	2.0	20	0.918941	([-1, 0], [67, 23921])
15	2.0	25	0.917989	([-1, 0], [69, 23919])
16	2.5	10	0.916101	([-1, 0, 1], [17, 23961, 10])
17	2.5	15	0.930769	([-1, 0], [41, 23947])
18	2.5	20	0.924035	([-1, 0], [56, 23932])
19	2.5	25	0.922682	([-1, 0], [59, 23929])
20	3.0	10	0.916025	([-1, 0, 1], [14, 23966, 8])
21	3.0	15	0.934758	([-1, 0], [33, 23955])
22	3.0	20	0.932207	([-1, 0], [39, 23949])
23	3.0	25	0.927700	([-1, 0], [48, 23940])



PLOT SILHOUETTE VALUE

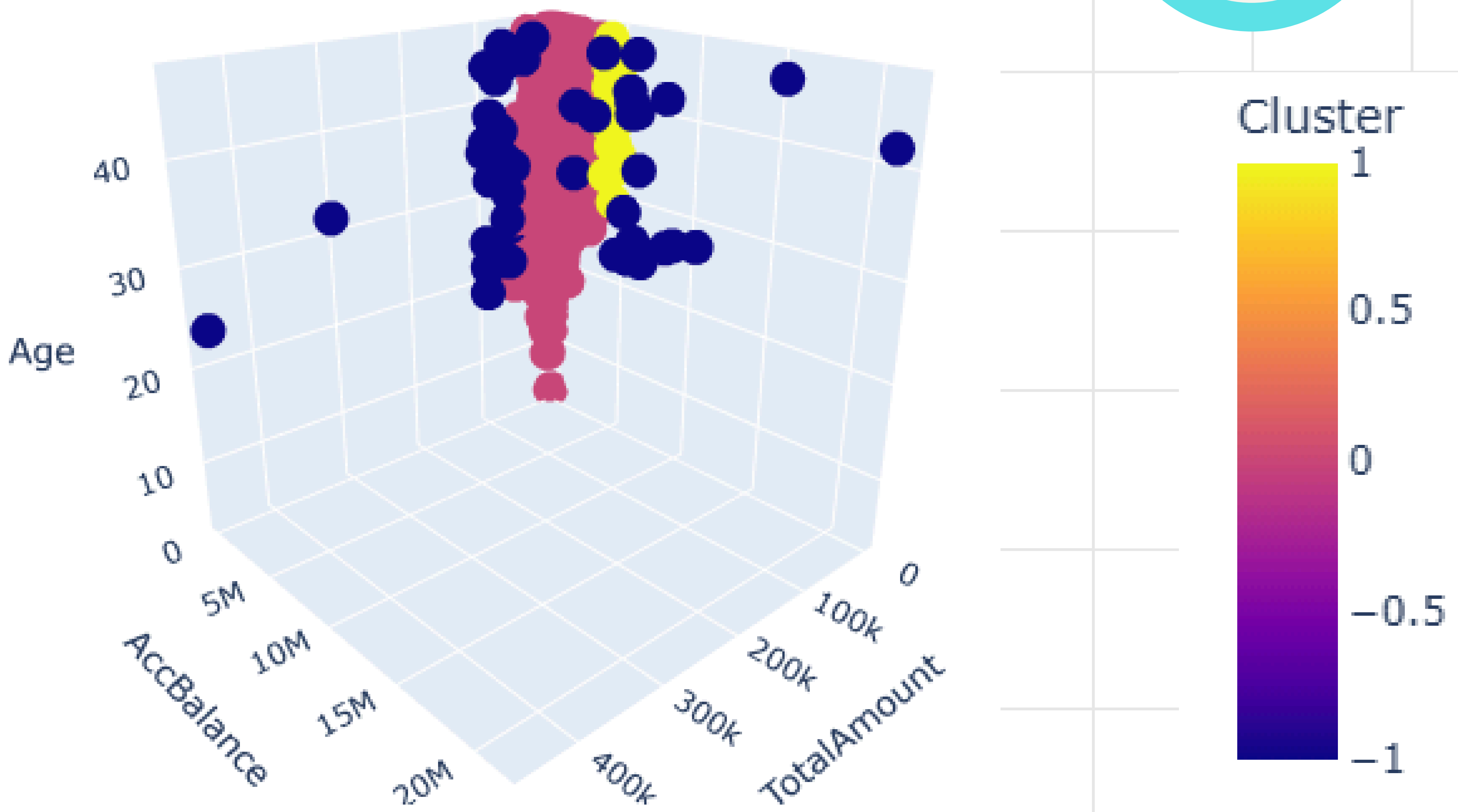
```
[30]: display(np.unique(labels_dbscan, return_counts=True))
      silhouette_score(scaled_features, labels_dbscan)

      (array([-1,  0,  1]), array([   49, 23924,    15]))
[30]: np.float64(0.8883913099110776)
```



GCD - IF A PAGI

VISUALISASI CLUSTER

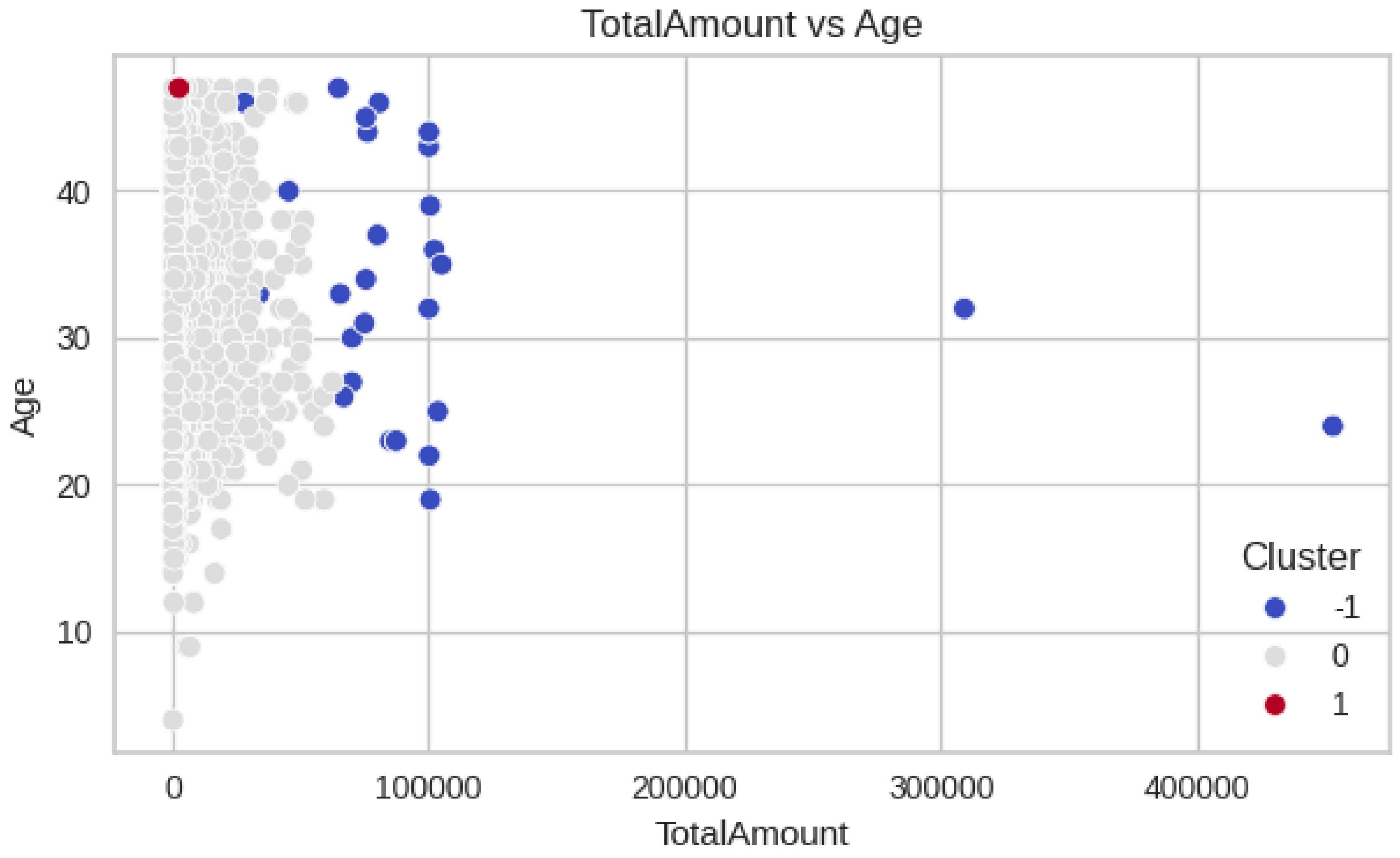
3D - Age, AccBalance, TotalAmount

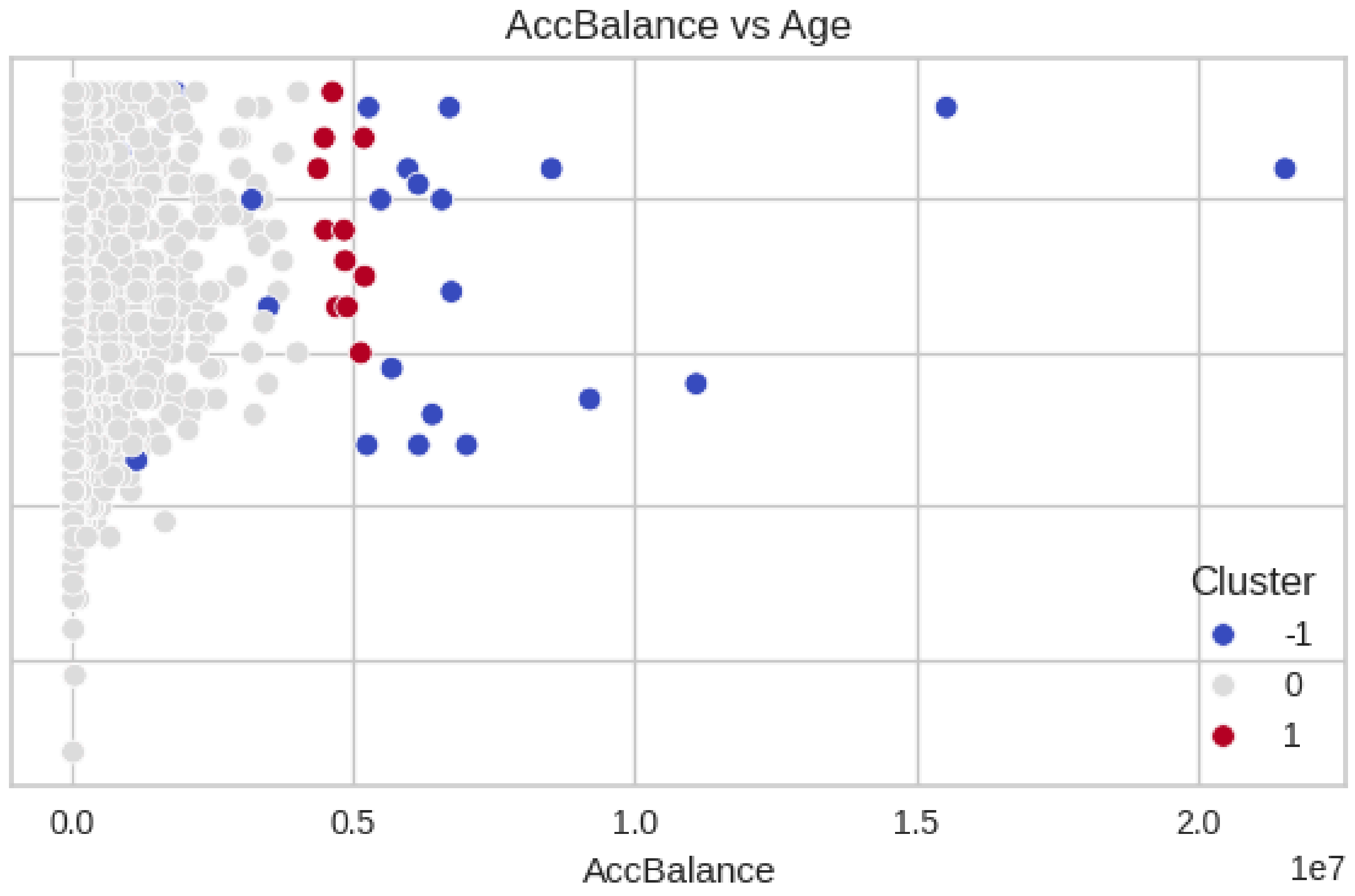


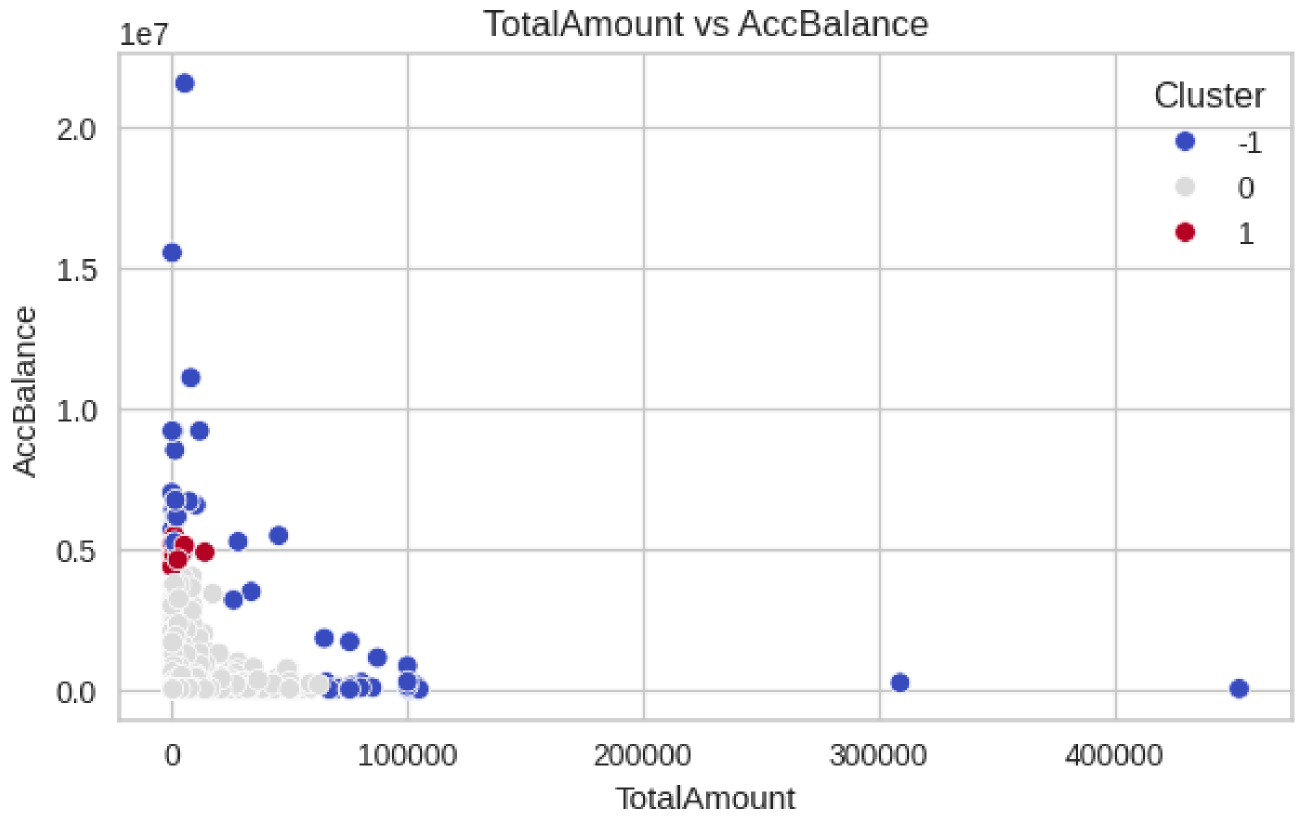
GCD - IF A PAGI

VISUALISASI CLUSTER

2D - Age + TotalAmount vs Age + AccBalance

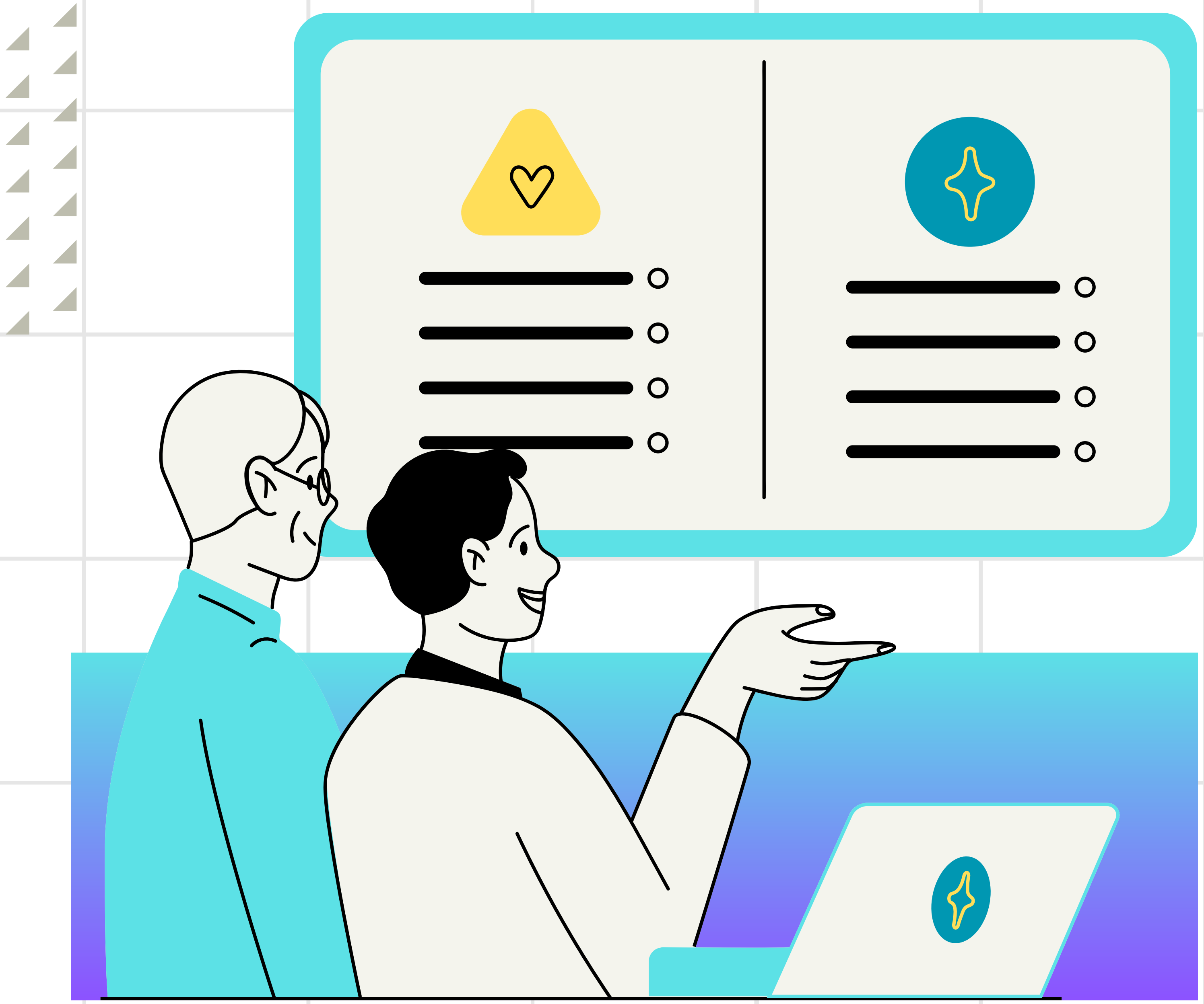


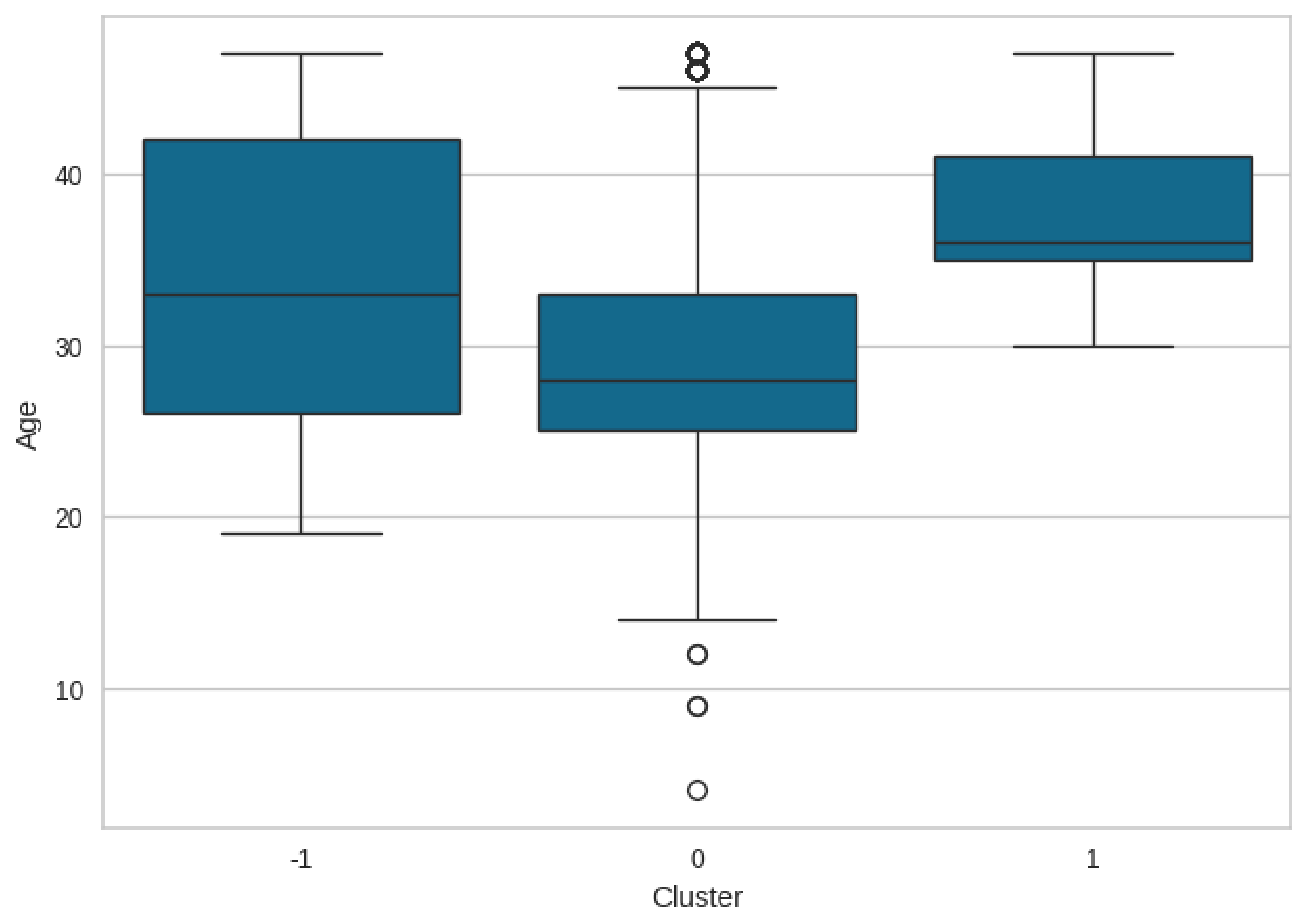




GCD - IF A PAGI

DISTRIBUSI DATA PER CLUSTER

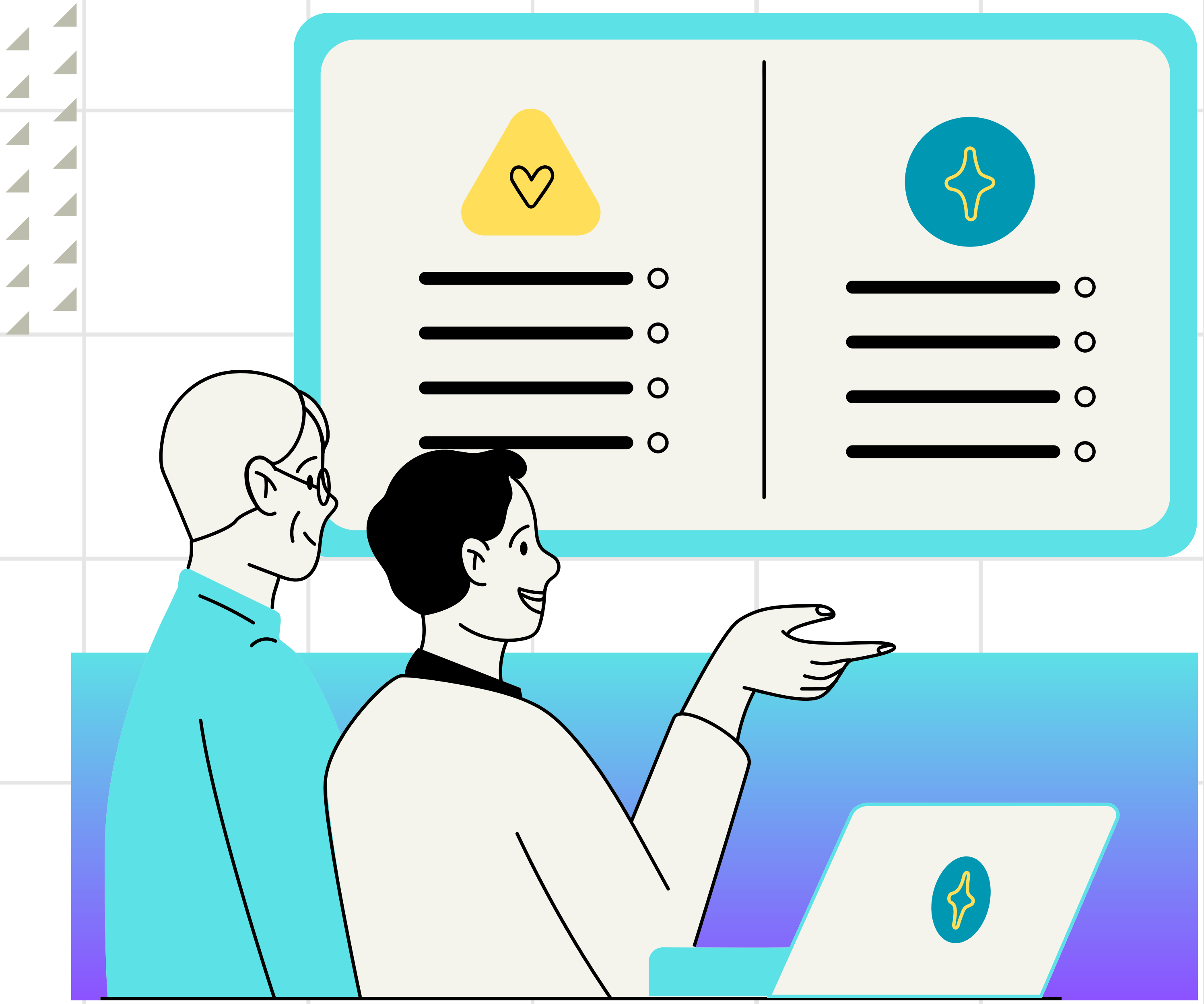


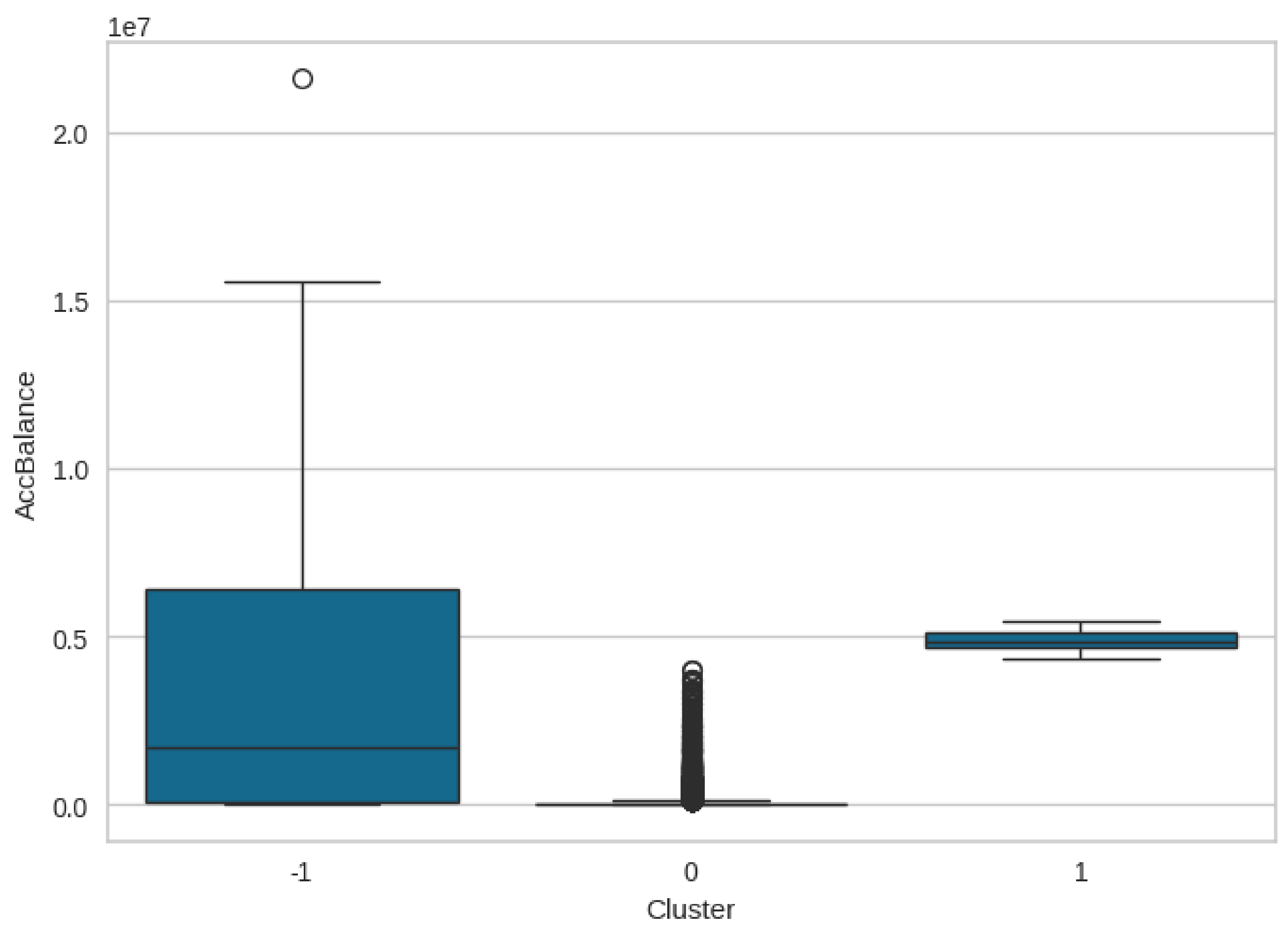


Distribusi USIA Nasabah saat Transaksi dilakukan

GCD - IF A PAGI

DISTRIBUSI DATA PER CLUSTER

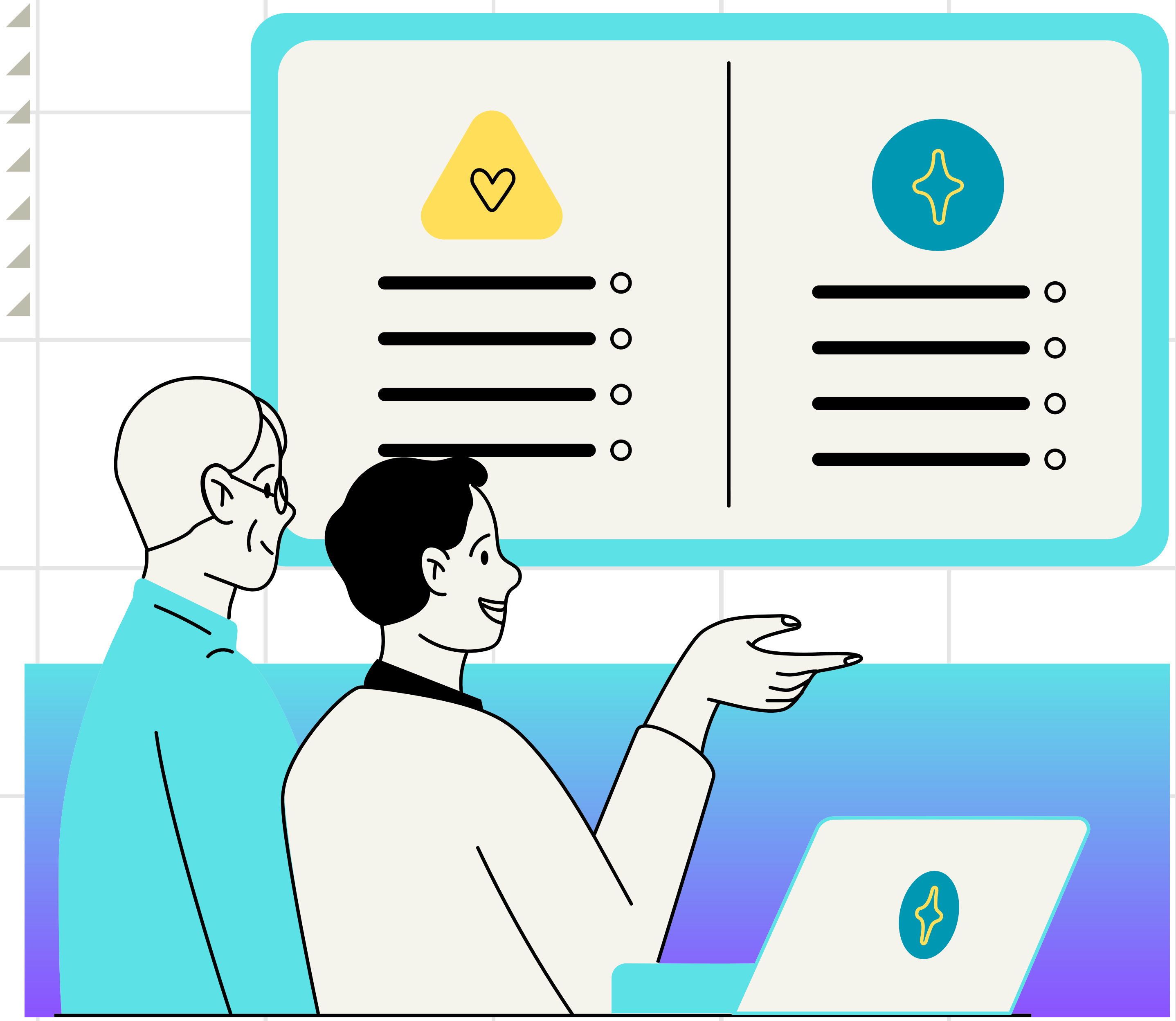


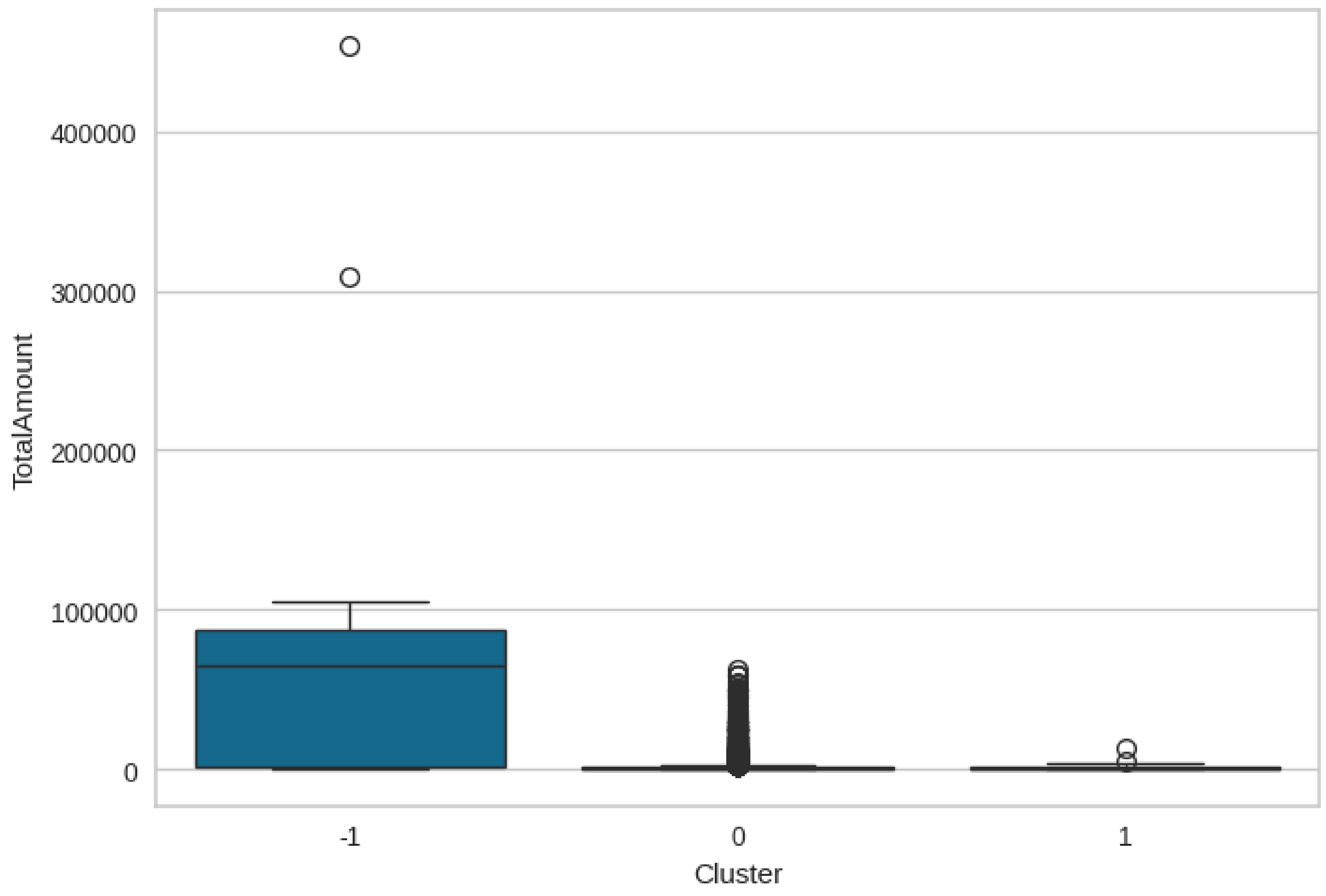


Distribusi SALDO AWAL Nasabah

GCD - IF A PAGI

DISTRIBUSI DATA PER CLUSTER

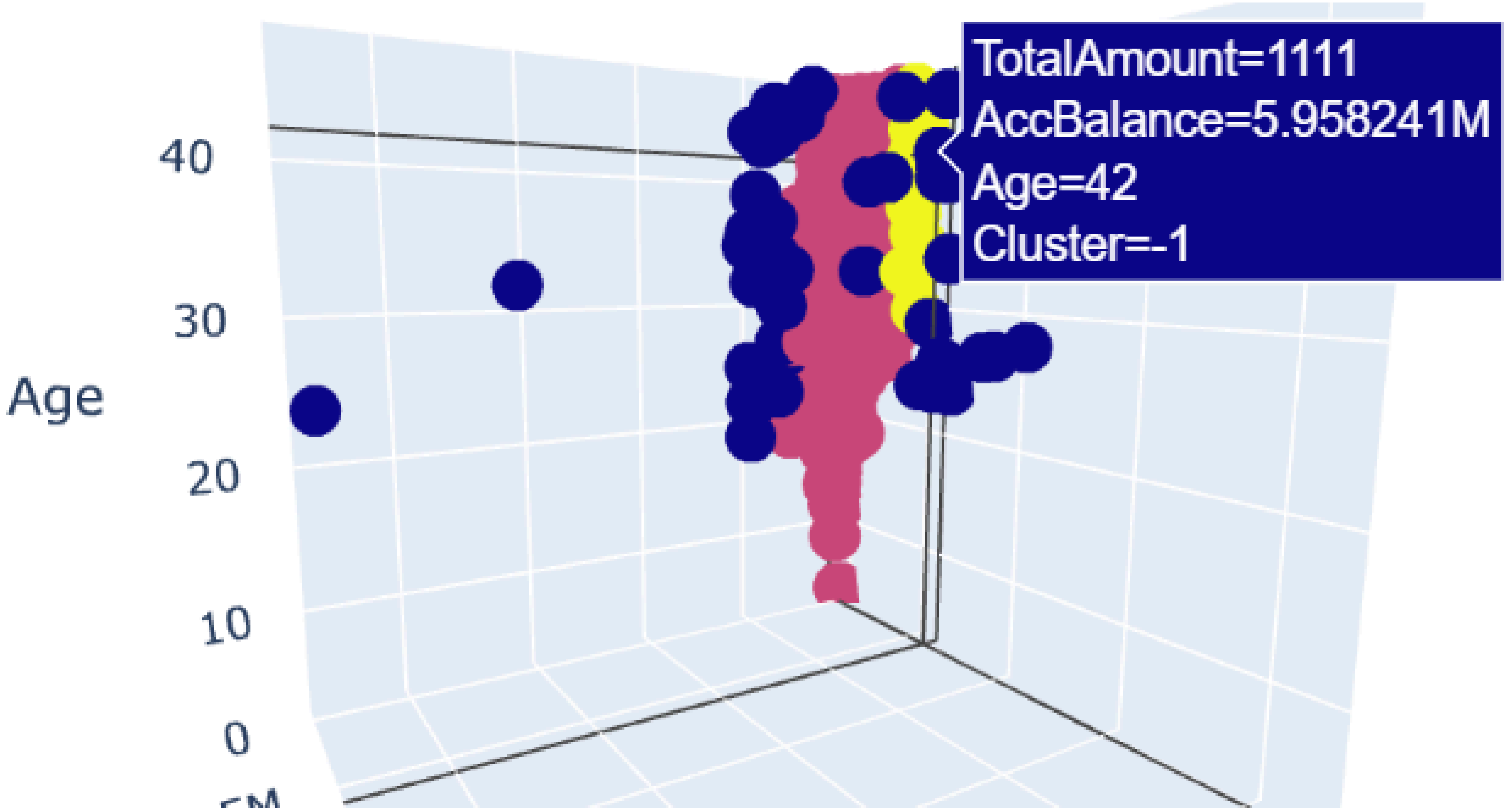




Distribusi TOTAL TRANSAKSI Nasabah

GCD - IF A PAGI

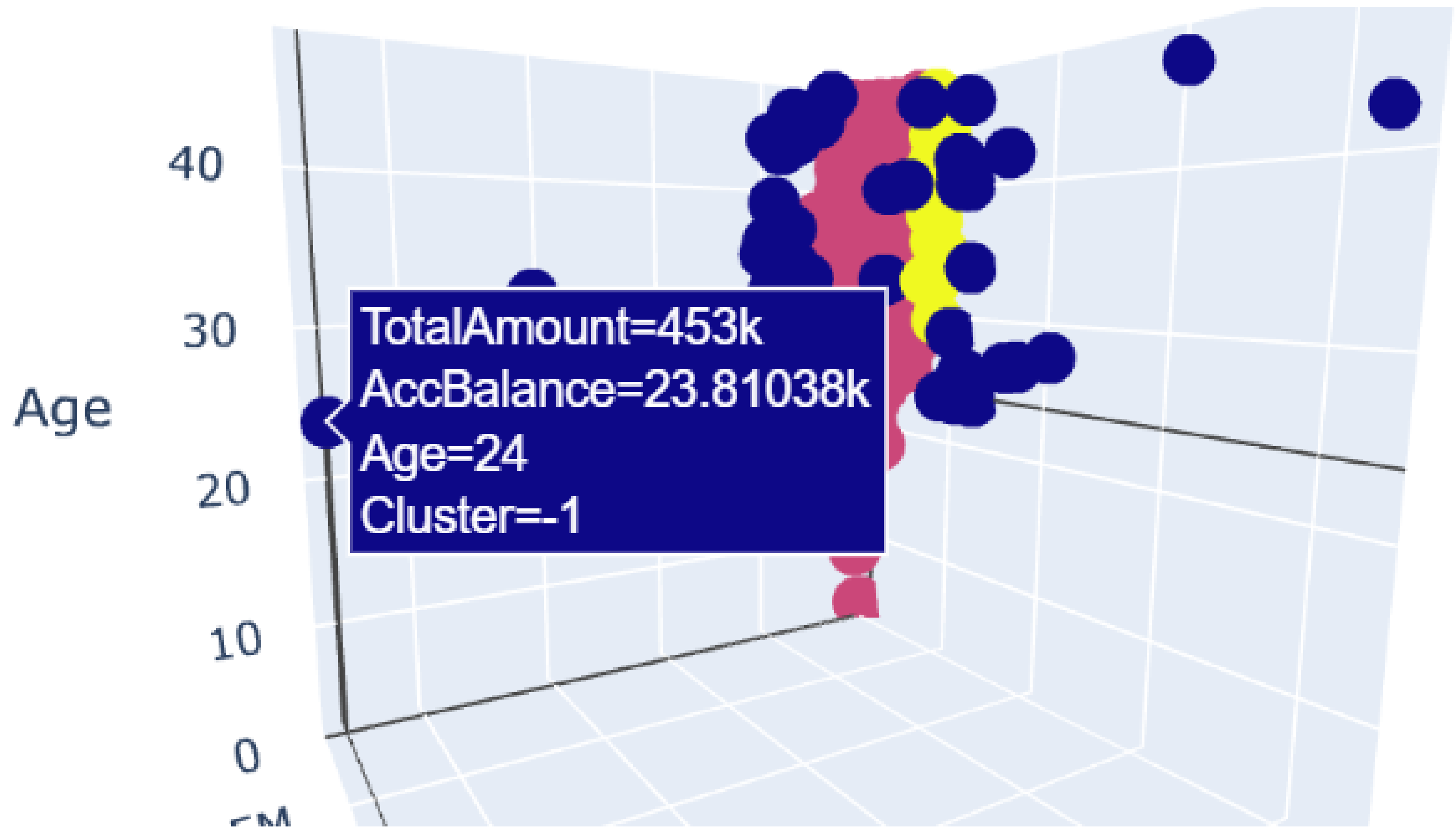
CONTOH TRANSAKSI ANOMALI



Saldo Awal	**: 5,95M INR**
Total Transaksi	**: 1.111 INR**
Usia	**: 42 tahun**
Cluster	**: -1 (outlier)**

GCD - IF A PAGI

CONTOH TRANSAKSI ANOMALI



Saldo Awal	: 23,81k INR
Total Transaksi	: 453k INR
Usia	: 24 tahun
Cluster	: -1 (outlier)

GCD - IF A PAGI

TANTANGAN & KESIMPULAN

01
Keterbatasan Sumber Daya
Kesulitan mengakses server untuk menjalankan coding yang sudah dibuat

02
Mencari Jumlah Cluster Terbaik
Menemukan kombinasi hyperparatemer terbaik dengan looping manual

03
Kesimpulan
DBSCAN cocok digunakan untuk memisahkan outliers, contohnya pada data transaksi anomali

THANK YOU!

[Akses Notebook GCD - Clustering with DBSCAN disini]

GCD - IF A PAGI

18/12/24.