Retail Customer Segmentation with K-Means Clustering Model

Khoerul Anam

Deskripsi Project

- This project aims to segment customers in the retail industry based on characteristics and purchasing behavior.
- Using machine learning model (unsupervised learning), K-Means Clustering.
- The main goal is to turn raw data into insights to improve business strategies.

Tools

python

matplotlib

Yellowbrick

scikit learn

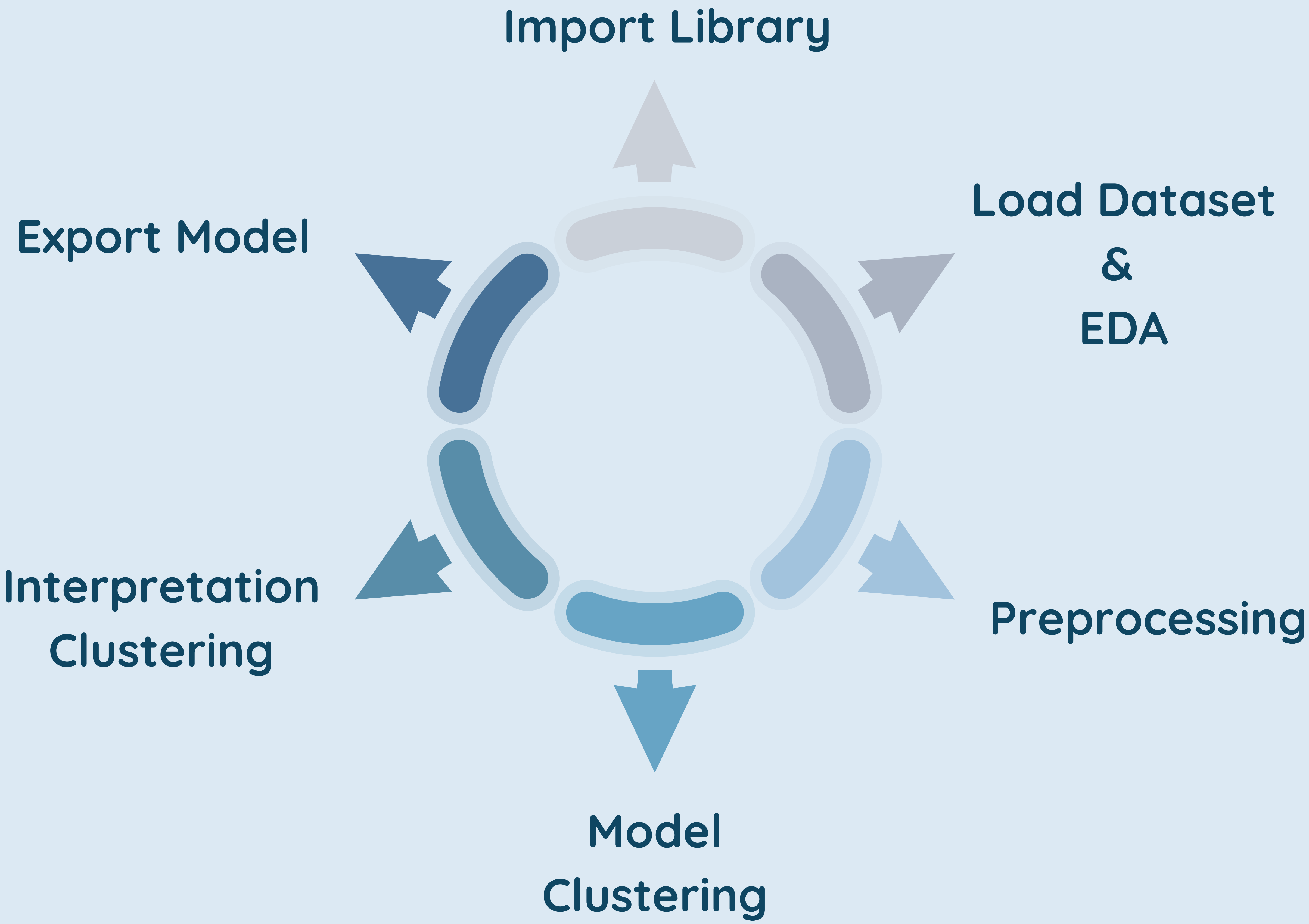
CSV

pandas

NumPy

seaborn

K-Means Clustering Steps



Import Library

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')


from sklearn.preprocessing import MinMaxScaler, StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split


from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer
from sklearn.metrics import silhouette_score

import joblib
```

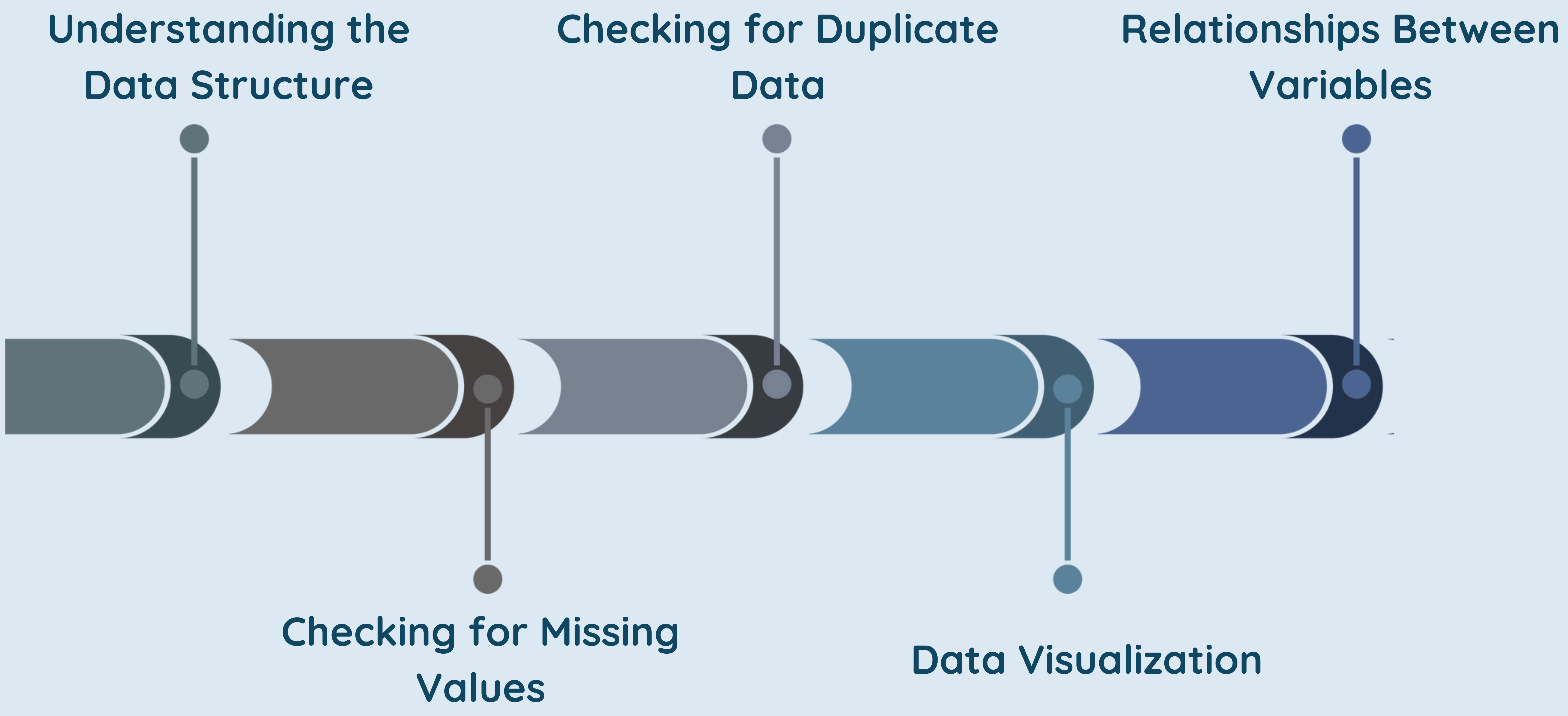
- Pandas: Used for data manipulation, particularly with DataFrames
- NumPy: Used for numerical computations, particularly with arrays.
- Scikit-learn: A machine learning library that includes tools for data modeling, preprocessing, and evaluation
- Matplotlib/Seaborn: Used for data visualization
- Joblib: Used for saving and loading machine learning models

Load Dataset

This dataset includes customer transaction data, such as the number of transactions, customer age, and product categories purchased. This data will be used for customer segmentation based on their purchasing behavior.

```
  3   print(df.head())

  TransactionID AccountID  TransactionAmount      TransactionDate  \
0      TX000001   AC00128              14.09  2023-04-11 16:29:14
1      TX000002   AC00455             376.24  2023-06-27 16:44:19
2      TX000003   AC00019             126.29  2023-07-10 18:16:08
3      TX000004   AC00070             184.50  2023-05-05 16:32:11
4      TX000005   AC00411              13.45  2023-10-16 17:51:24

  TransactionType   Location DeviceID      IP Address MerchantID Channel  \
0           Debit  San Diego  D000380  162.198.218.92       M015     ATM
1           Debit    Houston  D000051     13.149.61.4       M052     ATM
2           Debit       Mesa  D000235  215.97.143.157       M009  Online
3           Debit    Raleigh  D000187  200.13.225.150       M002  Online
4          Credit    Atlanta  D000308    65.164.3.100       M091  Online

   CustomerAge CustomerOccupation  TransactionDuration  LoginAttempts  \
0         70.0             Doctor                 81.0            1.0
1         68.0             Doctor                141.0            1.0
2         19.0            Student                 56.0            1.0
3         26.0            Student                 25.0            1.0
4          NaN            Student                198.0            1.0

   AccountBalance PreviousTransactionDate
0         5112.21     2024-11-04 08:08:08
1        13758.91     2024-11-04 08:09:35
2         1122.35     2024-11-04 08:07:04
3         8569.06     2024-11-04 08:09:06
4         7429.40     2024-11-04 08:06:39
```

Exploratory Data Analysis



Exploratory Data Analysis

```
3  print(df.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2537 entries, 0 to 2536
Data columns (total 16 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   TransactionID            2508 non-null   object
 1   AccountID                2516 non-null   object
 2   TransactionAmount        2511 non-null   float64
 3   TransactionDate          2509 non-null   object
 4   TransactionType          2507 non-null   object
 5   Location                 2507 non-null   object
 6   DeviceID                 2507 non-null   object
 7   IP Address               2517 non-null   object
 8   MerchantID               2514 non-null   object
 9   Channel                  2510 non-null   object
 10  CustomerAge              2519 non-null   float64
 11  CustomerOccupation       2514 non-null   object
 12  TransactionDuration      2511 non-null   float64
 13  LoginAttempts            2516 non-null   float64
 14  AccountBalance           2510 non-null   float64
 15  PreviousTransactionDate  2513 non-null   object
dtypes: float64(5), object(11)
memory usage: 317.3+ KB
None
```

Identify Data Structure and Type

```
3  print(df.describe())

       TransactionAmount  CustomerAge  TransactionDuration  LoginAttempts
count        2511.000000  2519.000000          2511.000000    2516.000000
mean          297.656468    44.678444           119.422939       1.121622
std           292.230367    17.837359            70.078513       0.594469
min             0.260000    18.000000            10.000000       1.000000
25%            81.310000    27.000000            63.000000       1.000000
50%           211.360000    45.000000           112.000000       1.000000
75%           413.105000    59.000000           161.000000       1.000000
max          1919.110000    80.000000           300.000000       5.000000

       AccountBalance
count     2510.000000
mean      5113.438124
std       3897.975861
min        101.250000
25%       1504.727500
50%       4734.110000
75%       7672.687500
max      14977.990000
```

Descriptive Statistics

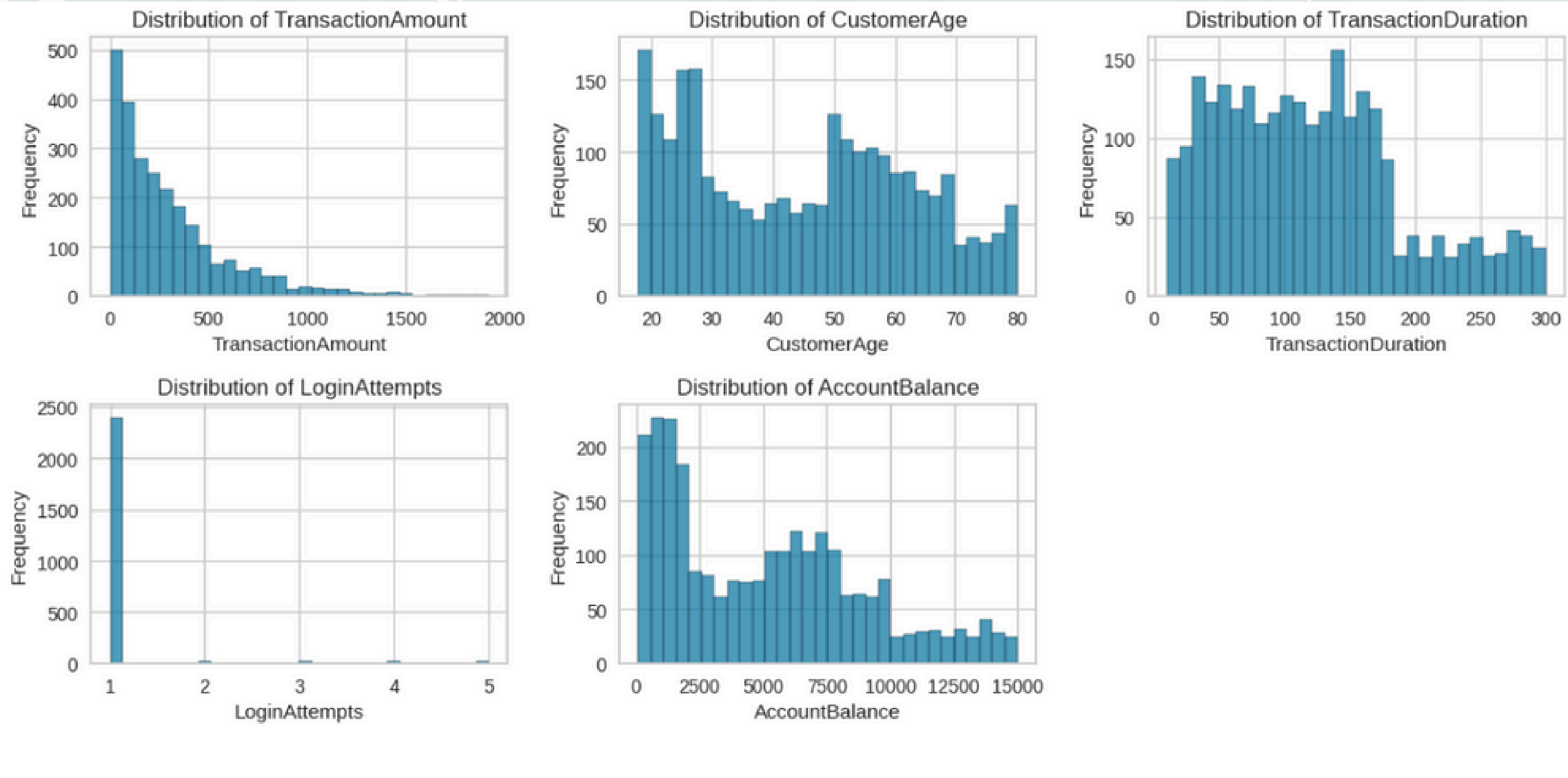
```
Missing values before handling:
TransactionID              29
AccountID                  21
TransactionAmount          26
TransactionDate            28
TransactionType            30
Location                   30
DeviceID                   30
IP Address                 20
MerchantID                 23
Channel                    27
CustomerAge                18
CustomerOccupation         23
TransactionDuration        26
LoginAttempts              21
AccountBalance             27
PreviousTransactionDate    24
```

Missing Value

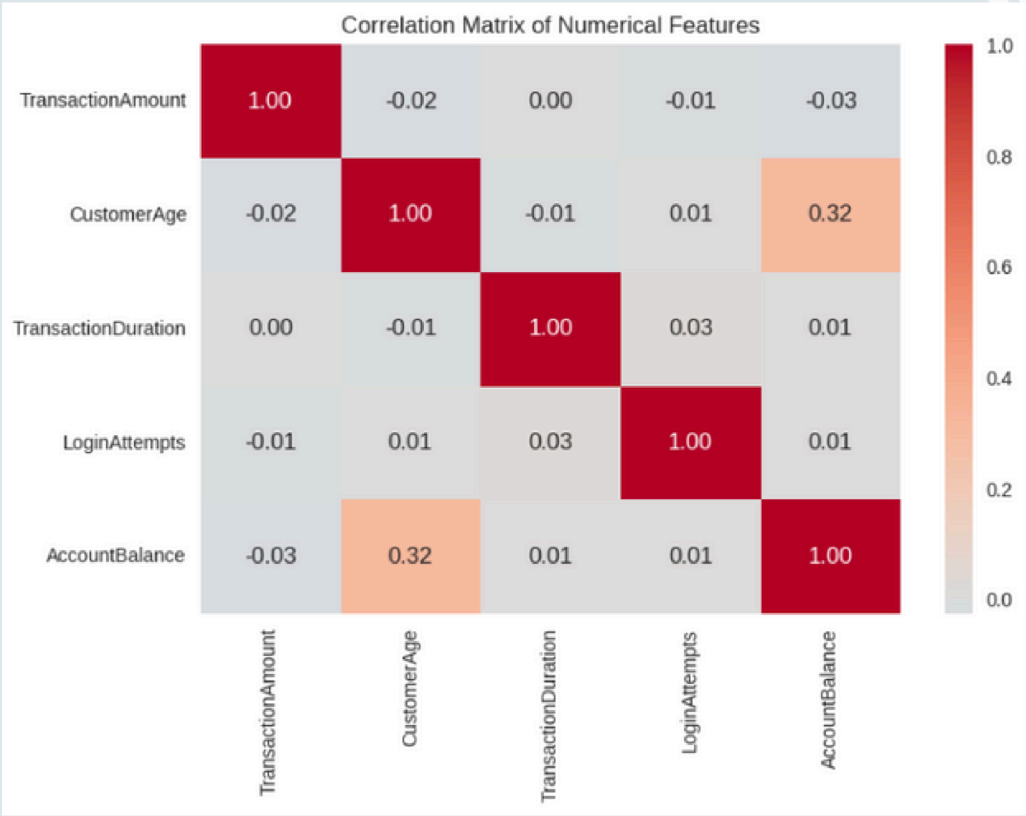
```
3  print("\nDuplicate rows before handling:")
4  print(df.duplicated().sum())

Duplicate rows before handling:
21
```

Duplicate Data

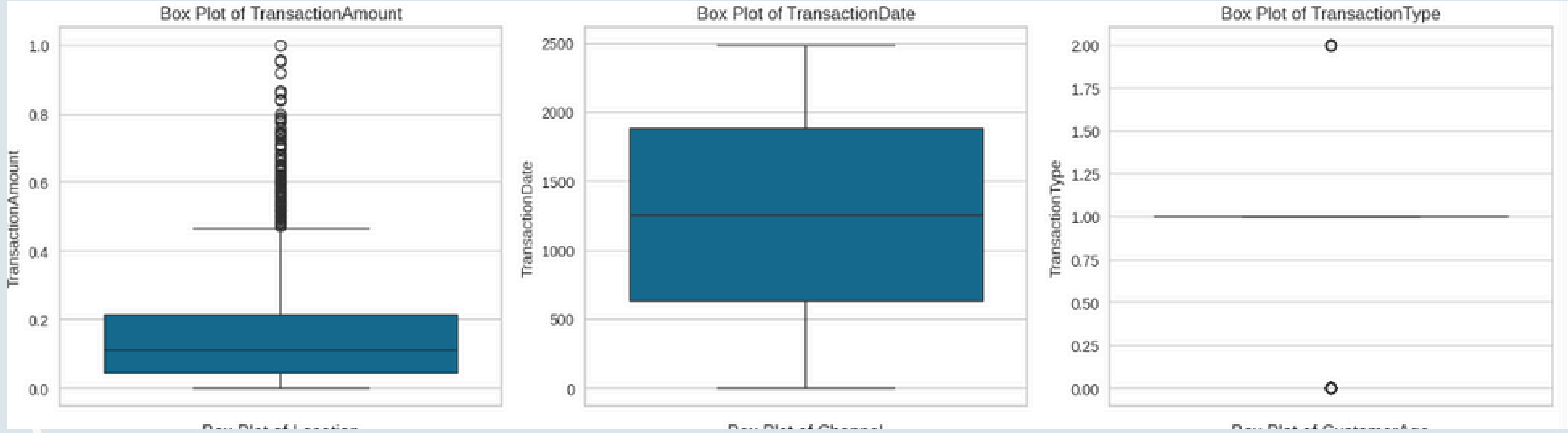
Exploratory Data Analysis



Histogram Visualization



Correlation Between Features



Outlier Visualization

Preprocessing

```
# Melakukan drop pada kolom yang memiliki keterangan id dan IP Address
id_columns = ['TransactionID', 'AccountID', 'DeviceID', 'IP Address', 'MerchantID']
print(f"Kolom yang akan di-drop: {id_columns}")
```

```
Kolom yang akan di-drop: ['TransactionID', 'AccountID', 'DeviceID', 'IP Address', 'MerchantID']
```

Removing Irrelevant Columns

```

if missing_after.sum() == 0:
    print("✅ Tidak ada missing values yang perlu ditangani!")
else:
    # Jika ada missing values, lakukan imputasi
    print("⚠ Melakukan imputasi untuk missing values...")
    df_processed = df_processed.fillna(df_processed.mean())
    print("✅ Imputasi selesai!")

```

Filling Missing Values

Removing Duplicates:

```
# Menghapus data duplikat menggunakan drop_duplicates().

print("\nMenghapus data duplikat menggunakan drop_duplicates():")
before_drop = len(df_processed)
df_processed = df_processed.drop_duplicates()
after_drop = len(df_processed)
print(f"Data sebelum menghapus duplikat: {before_drop}")
print(f"Data setelah menghapus duplikat: {after_drop}")
print(f"Jumlah data duplikat yang dihapus: {before_drop - after_drop}")
```

```
Menghapus data duplikat menggunakan drop_duplicates():
Data sebelum menghapus duplikat: 2537
Data setelah menghapus duplikat: 2515
Jumlah data duplikat yang dihapus: 22
```

Preprocessing

```
✅ Feature encoding selesai! Menampilkan hasil dengan head():
   TransactionAmount  TransactionDate  TransactionType  Location  Channel  \
0           0.007207              680                1        36        0
1           0.195940             1178                1        15        0
2           0.065680             1262                1        23        2
3           0.096016              818                1        33        2
4           0.006874             1939                0         1        2

   CustomerAge  CustomerOccupation  TransactionDuration  LoginAttempts  \
0     0.838710                   0             0.244828            0.0
1     0.806452                   0             0.451724            0.0
2     0.016129                   3             0.158621            0.0
3     0.129032                   3             0.051724            0.0
4          NaN                   3             0.648276            0.0

   AccountBalance  PreviousTransactionDate
0        0.336832                      105
1        0.918055                      192
2        0.068637                       41
3        0.569198                      163
4        0.492591                       16
```

feature encoding
Using LabelEncoder()

```python
if not numerical_cols_scaled:
        print("Tidak ada fitur numerik untuk penanganan outlier.")
else:
    for col in numerical_cols_scaled:
        try:
            # Hitung IQR pada data yang diskalakan
            Q1 = df_processed_cleaned[col].quantile(0.25)
            Q3 = df_processed_cleaned[col].quantile(0.75)
            IQR = Q3 - Q1

            # Tentukan batas atas dan bawah untuk outlier
            # Menggunakan 1.5 * IQR adalah ambang batas umum
            lower_bound = Q1 - 1.5 * IQR
            upper_bound = Q3 + 1.5 * IQR

            # Lakukan capping: ganti nilai di luar batas dengan batas itu sendiri
            df_processed_cleaned[col] = df_processed_cleaned[col].clip(lower=lower_bound, upper=upper_bound)
```

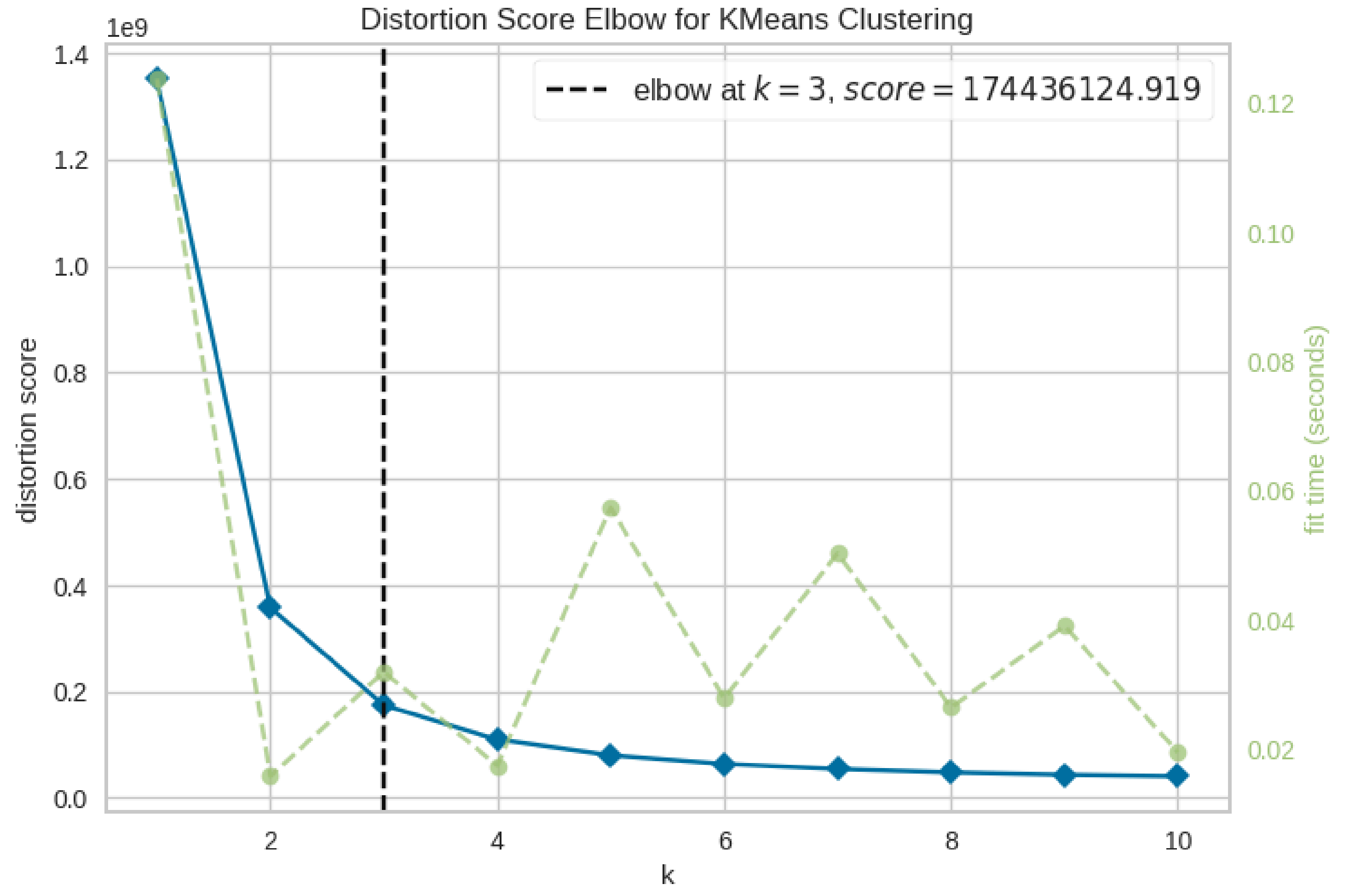
Handling Outlier Using IQR

feature scaling Using MinMaxScaler()

```python
# Menggunakan MinMaxScaler
scaler = MinMaxScaler()
df_processed[numerical_cols] = scaler.fit_transform(df_processed[numerical_cols])

print("\nFeature scaling selesai! Menampilkan hasil dengan head():")
print(df_processed[numerical_cols].head())

```

```
Feature scaling selesai! Menampilkan hasil dengan head():
   TransactionAmount  CustomerAge  AccountBalance  TransactionDuration  \
0           0.007207     0.838710        0.336832             0.244828
1           0.195940     0.806452        0.918055             0.451724
2           0.065680     0.016129        0.068637             0.158621
3           0.096016     0.129032        0.569198             0.051724
4           0.006874          NaN        0.492591             0.648276

   LoginAttempts
0            0.0
1            0.0
2            0.0
3            0.0
4            0.0
```

Elbow Method



Elbow Method determines the optimal number of clusters in the K-Means algorithm by plotting the relationship between the number of clusters (k) and the distortion score (SSE). This graph shows a sharp decline in SSE at the beginning, indicating that adding more clusters improves clustering quality. However, after k=3, the decline becomes flatter, suggesting that adding more clusters does not significantly improve. Thus, k=3 is the optimal number of clusters for this data.

K-Means Clustering

```
Melatih model K-Means dengan 3 cluster...
Penambahan label cluster ke DataFrame df_processed berhasil.
5 baris pertama df_processed dengan kolom 'Cluster' baru:
   TransactionAmount  TransactionDate  TransactionType  Location  Channel  \
0           0.007207              680                1        36        0
1           0.195940             1178                1        15        0
2           0.065680             1262                1        23        2
3           0.096016              818                1        33        2
4           0.006874             1939                0         1        2

   CustomerAge  CustomerOccupation  TransactionDuration  LoginAttempts  \
0     0.838710                   0             0.244828            0.0
1     0.806452                   0             0.451724            0.0
2     0.016129                   3             0.158621            0.0
3     0.129032                   3             0.051724            0.0
4     0.430297                   3             0.648276            0.0

   AccountBalance  PreviousTransactionDate  Target  Cluster
0        0.336832                      105       1        1
1        0.918055                      192       0        0
2        0.068637                       41       0        0
3        0.569198                      163       1        1
4        0.492591                       16       2        2

Jumlah anggota di setiap cluster:
Cluster
0    836
1    839
2    840
Name: count, dtype: int64
```

Using the K-Means model applied to this dataset, the model clusters the data into 3 groups based on relevant characteristics. Each cluster has a relatively balanced size, indicating that the model has effectively grouped the data.

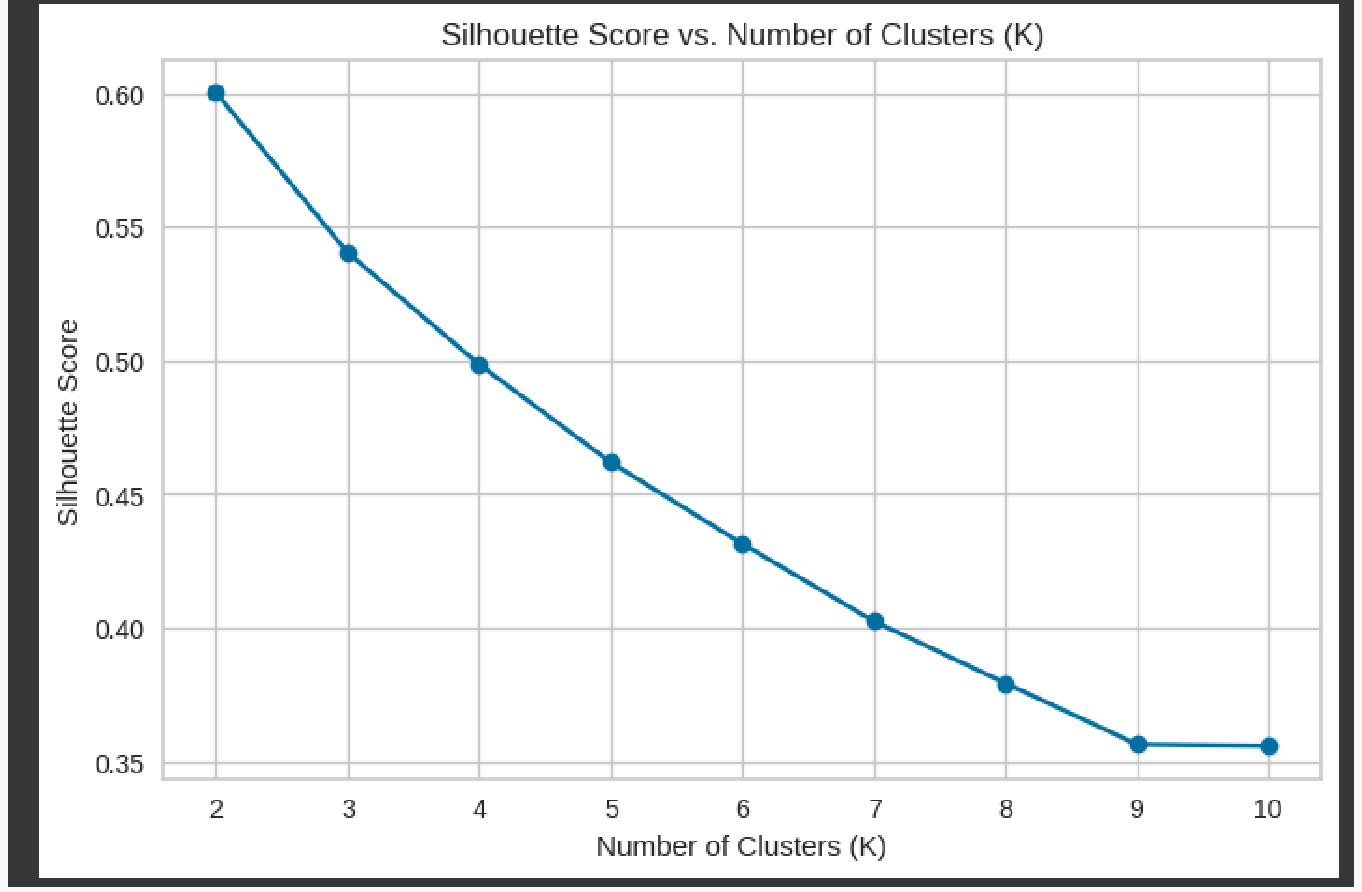
Cluster 0: 836 data

Cluster 1: 839 data

Cluster 2: 840 data

Silhouette Score

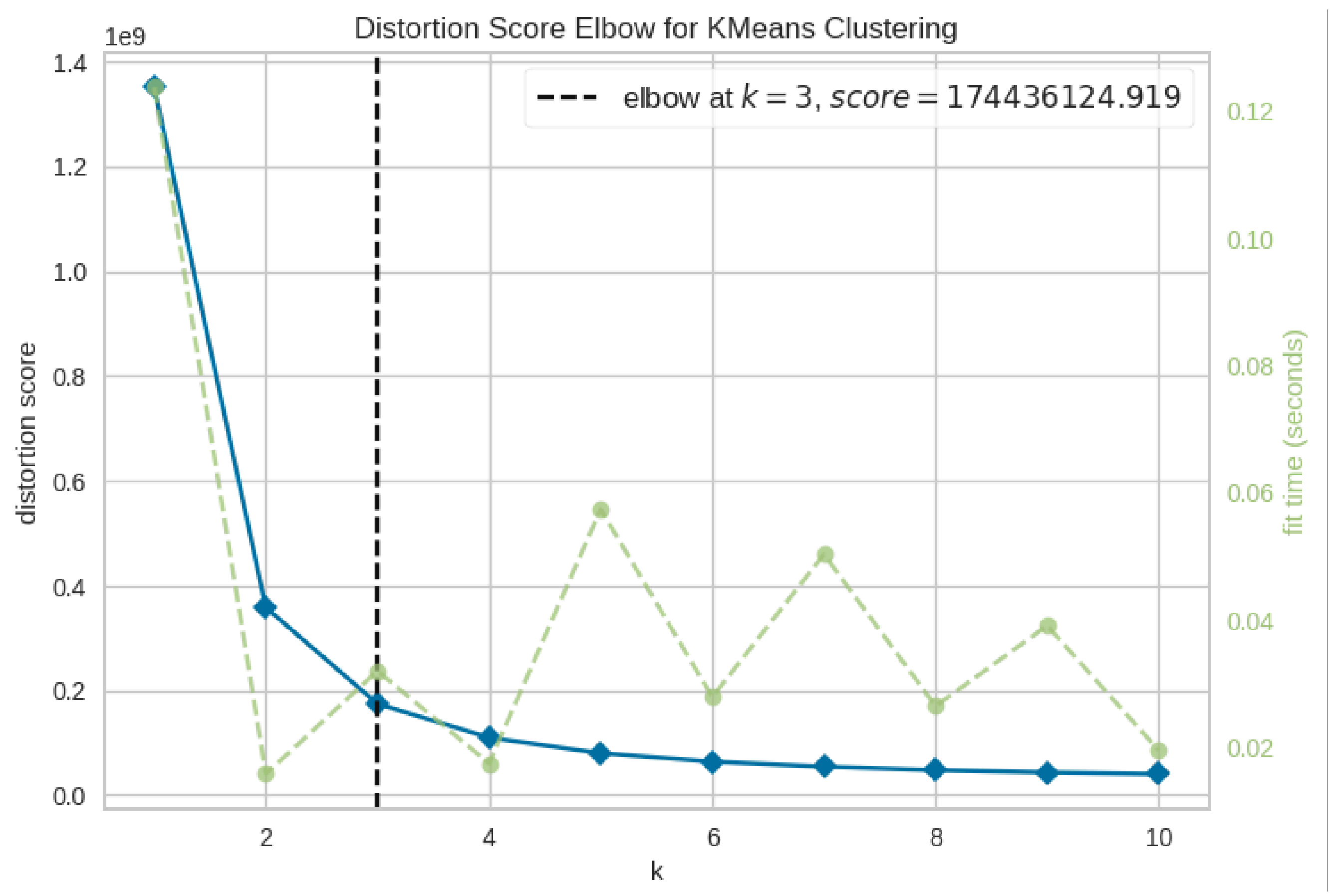
```
--- Menghitung Silhouette Score untuk Berbagai Nilai K ---
Silhouette Score untuk k=2: 0.6003
Silhouette Score untuk k=3: 0.5405
Silhouette Score untuk k=4: 0.4986
Silhouette Score untuk k=5: 0.4619
Silhouette Score untuk k=6: 0.4314
Silhouette Score untuk k=7: 0.4025
Silhouette Score untuk k=8: 0.3793
Silhouette Score untuk k=9: 0.3564
Silhouette Score untuk k=10: 0.3558
```

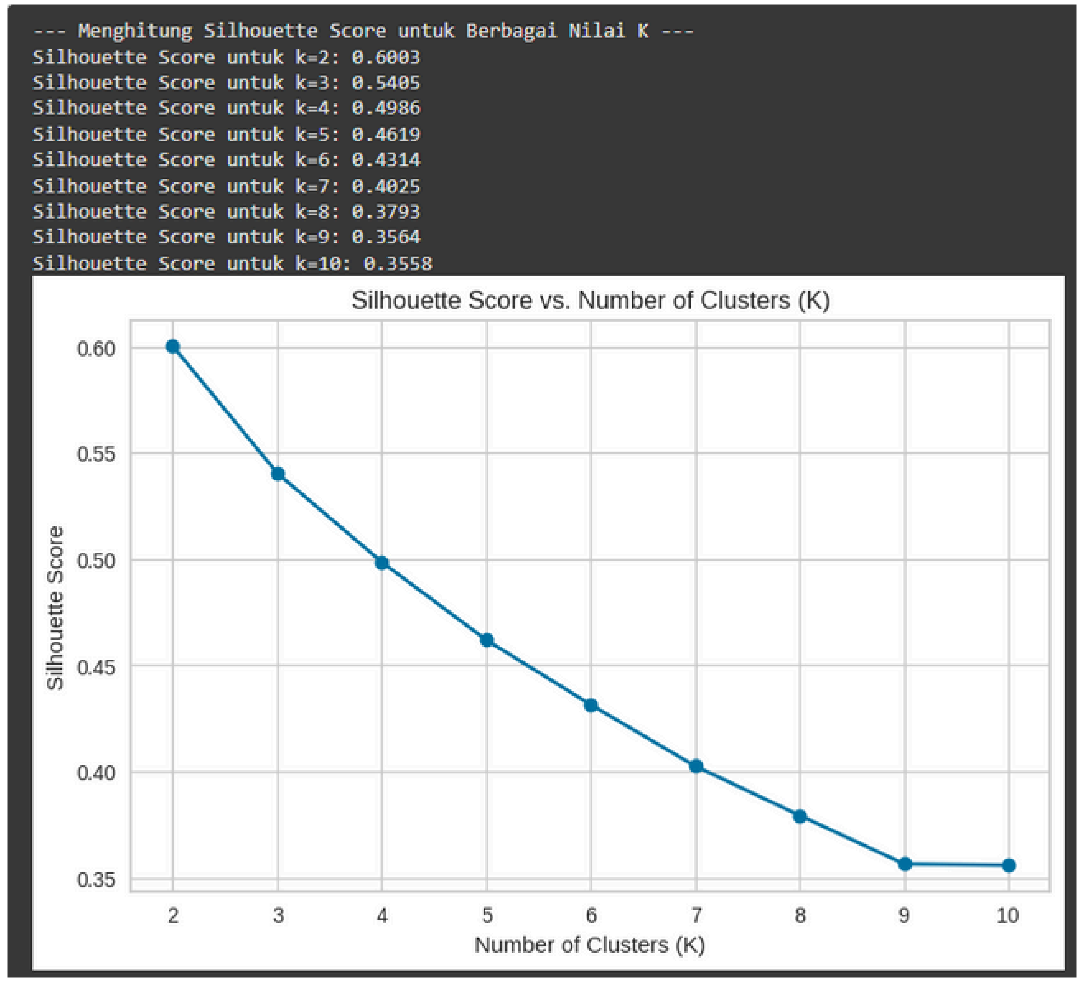


It is an evaluation metric that measures how well the data is clustered. The Silhouette Score, with a **score of 0.6003**, gives the best result at **k=2**, indicating that the data in **2 clusters** is clearly separated, providing better separation between clusters.

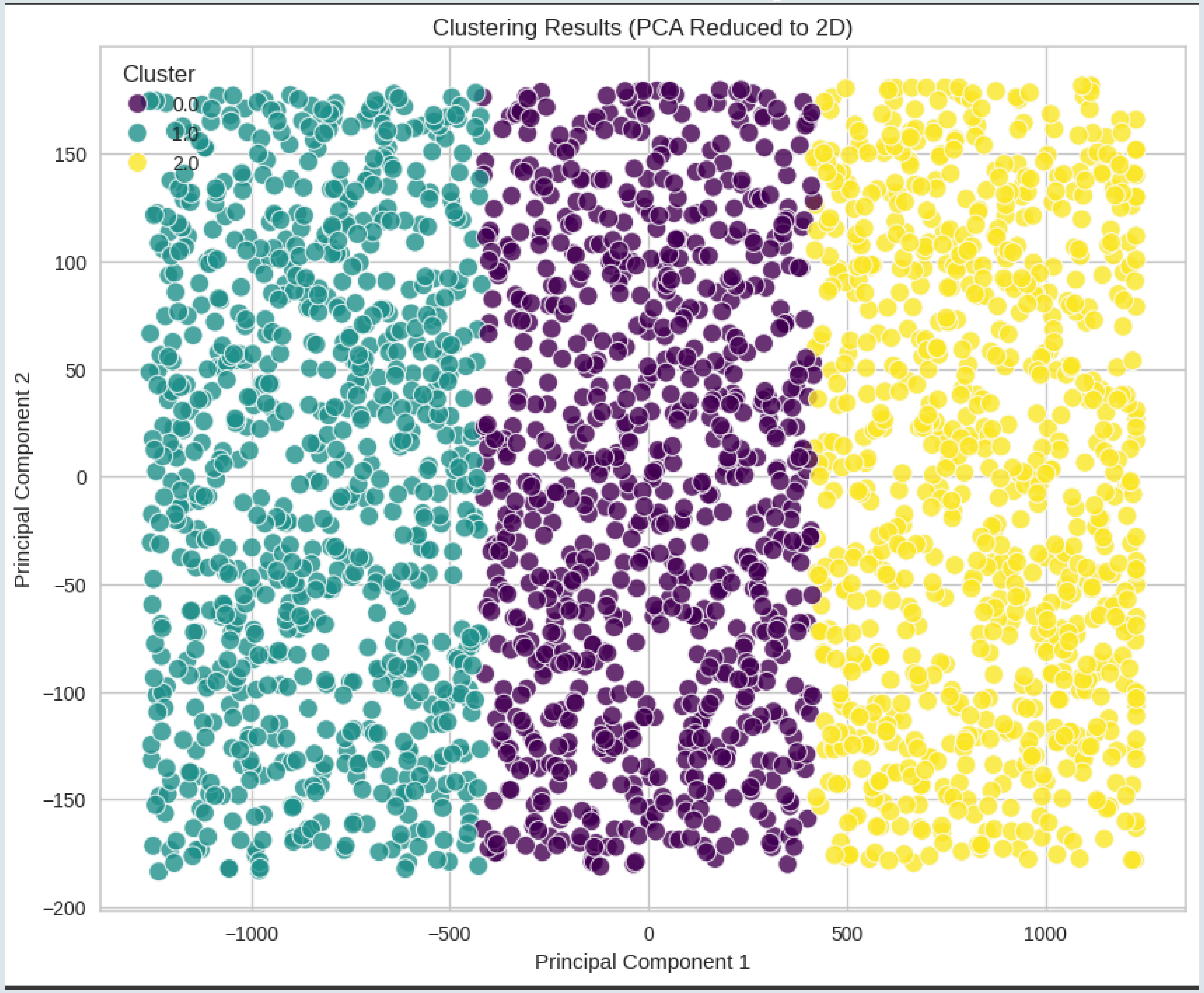
Silhouette Score and Elbow Method Results

The Elbow Method shows the optimal number of clusters at **k=3,** while the Silhouette Score reaches its highest value at **k=2.** Although **k=2** provides better results in terms of cluster separation, **k=3 is chosen** because it offers **more detailed segmentation,** which is easier to apply in making strategic decisions, especially for deeper analysis.





Clustering Results



The clustering results performed using the K-Means algorithm are applied to data that has been processed and reduced in dimensions using **PCA (Principal Component Analysis)** to visualize the data in 2D space. The clustering results show that the K-Means model successfully grouped the data into **three distinct clusters**.

Interpretation of Results

Cluster 0
Active & Mature Customers

Customers in this cluster are older and have moderate transaction activity and account balance. They log in with the least effort and are more active on the platform. Recommendation: Maintain engagement with relevant offers, improve login processes, and offer products suitable for the more mature age group

Cluster 1
Young, Less Active & Low Transactions

This cluster is dominated by young customers with low transaction value and short transaction durations. They tend to have high login efforts, indicating potential access issues. Recommendation: Implement re-engagement campaigns, improve login experience, and offer entry-level products suitable for their transaction patterns.

Cluster 2
Affluent Customers with High Transactions & Balances

Customers in this cluster have the highest transactions and account balances, with longer transaction durations. They are in moderate recency and age, making them a high-value segment. Recommendation: Focus on premium services, investment products, exclusive loyalty programs, and cross-selling high-value products.

Export Data dan Model

```python
output_filename = 'data_clustering.csv'
try:
    df_modeling.to_csv(output_filename, index=False)
    print(f"Data hasil clustering berhasil disimpan ke file '{output_filename}'")
except Exception as e:
    print(f"Gagal menyimpan file CSV: {e}")
```

```
Data hasil clustering berhasil disimpan ke file 'data_clustering.csv'
```

Export Data

```python
# Menyimpan model menggunakan joblib

try:
    joblib.dump(model_kmeans, "model_clustering")
    print("✅ Model clustering berhasil disimpan sebagai 'model_clustering")
except Exception as e:
    print(f"❌ Gagal menyimpan model: {e}")
```

```
✅ Model clustering berhasil disimpan sebagai 'model_clustering
```

Export Clustering Model

Conclusion

This project aimed to segment customers using transaction data. The clustering successfully divided customers into three distinct segments: Active & Mature Customers, Young & Less Active Customers with Small Transactions, and Affluent Customers with High Transactions & Balances. Each segment has unique characteristics in terms of age, frequency, value, and transaction duration, allowing for more effective business strategy adjustments. The clustering model created is also saved for future use.

Thank you

Feel free to reach out for further inquiries or collaborations.

khoerulanam231@gmail.com

https://www.linkedin.com/in/khoerul-anam-a7b627221/

https://github.com/khoerul-anam