Modul 3
Praktikum Data Mining

PROGRAM STUDI SAINS DATA
FAKULTAS SAINS
INSTITUT TEKNOLOGI SUMATERA
2025

Data Reduction

In data mining, a technique called data reduction is employed to minimize the size of a dataset while maintaining the most crucial information. This may be helpful in cases where the dataset is too big to handle well or contains a significant number of redundant or unnecessary information. There are two kinds of data reductions:

1. Dimensionality Reduction

   This technique involves reducing the number of features in the dataset, either by removing features that are not relevant or by combining multiple features into a single feature. Some of dimensionality reduction techniques are Wavelet transformation, Principal component analysis (PCA), feature selection, and attribute subset selection.

2. Numerosity Reduction

   Reduce data volume by choosing alternative, smaller forms of data representation. The numerosity reduction can be done using parametric and non parametrics methods. Parametric method (e.g., Regression) assumes the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers). Non parametric methods do not assume models (e.g., Histograms, Clustering, and Sampling).

Data reduction is an important step in data mining, as it can help to improve the efficiency and performance of machine learning algorithms by reducing the size of the dataset. However, it is important to be aware of the trade-off between the size and accuracy of the data, and carefully assess the risks and benefits before implementing it.

Advantages of Data Reduction in Data Mining

- ✓ Improved efficiency: Data reduction can help to improve the efficiency of machine learning algorithms by reducing the size of the dataset. This can make it faster and more practical to work with large datasets.
- ✓ Improved performance: Data reduction can help to improve the performance of machine learning algorithms by removing irrelevant or redundant information from the dataset. This can help to make the model more accurate and robust.
- ✓ Reduced storage costs: Data reduction can help to reduce the storage costs associated with large datasets by reducing the size of the data.
- ✓ Improved interpretability: Data reduction can help to improve the interpretability of the results by removing irrelevant or redundant information from the dataset.

Data Transformation

In the context of data mining, data transformation is the act of transforming unprocessed data into an appropriate format for analysis and modeling. Data transformation aims to prepare the data for data mining to extract valuable knowledge and insights. Data transformation is a crucial phase in the data mining process because it guarantees that the data is error -and inconsistency-free and in a format that can be used for analysis and modeling. By lowering the dimensionality of the data and scaling it to a common range of values, data transformation can also aid in enhancing the performance of data mining algorithms. Several techniques for transforming data include:

1. Data Smoothing

   Using certain techniques, the process of "data smoothing" is performed to eliminate noise from the dataset. Data smoothing makes it possible to draw attention to significant aspects in the dataset. Data can be altered during collection in order to remove or lessen variation or other types of noise. The idea behind data smoothing is that it can recognize small changes to aid in the prediction of various patterns and trends.

2. Attribute Construction

   Using the provided set of attributes, new attributes are constructed and applied to aid in the mining process. This streamlines the initial data and improves mining effectiveness.

3. Data Aggregation

   The process of storing and displaying data in an abridged format is called data aggregation. To include the data from many data sources into a description of the data analysis, the data may come from a number of different sources. This is an important phase since the volume and caliber of data used have a significant impact on how accurate data analysis findings are. To obtain meaningful findings, a sufficient amount of high-quality and precise data must be gathered.

4. Data Generalization

   Data generalization refers to the process of transforming low-level attributes into high-level ones by using the concept of hierarchy. Data generalization is applied to categorical data where they have a finite but large number of distinct values.

5. Data Normalization

Data normalization involves converting all data variables into a given range usually between [0,1]. Techniques that are used for normalization are Min-Max Normalization, Z-Score Normalization, and Normalization by Decimal Scaling.

6. Data Discretization

   Data discretization refers to the process of transforming continuous data into a set of data intervals. This is a very helpful method that can help you study and understand the data more easily and increase the effectiveness of any applied algorithm.

Advantages of Data Transformation in Data Mining

- ✓ Improves Data Quality: Data transformation helps to improve the quality of data by removing errors, inconsistencies, and missing values.
- ✓ Facilitates Data Integration: Data transformation enables the integration of data from multiple sources, which can improve the accuracy and completeness of the data.
- ✓ Improves Data Analysis: Data transformation helps to prepare the data for analysis and modeling by normalizing, reducing dimensionality, and discretizing the data.
- ✓ Increases Data Security: Data transformation can be used to mask sensitive data, or to remove sensitive information from the data, which can help to increase data security.
- ✓ Enhances Data Mining Algorithm Performance: Data transformation can improve the performance of data mining algorithms by reducing the dimensionality of the data and scaling the data to a common range of values.

Practical Work Objectives

1. Learning the concept of data reduction (PCA, histogram, and sampling) and data transformation (min-max normalization, z-score normalization, and normalization by decimal scaling).
2. Understanding the concept of data reduction and data transformation in Python

The week 3 practical procedure is as follows:

- ➢ **PCA**

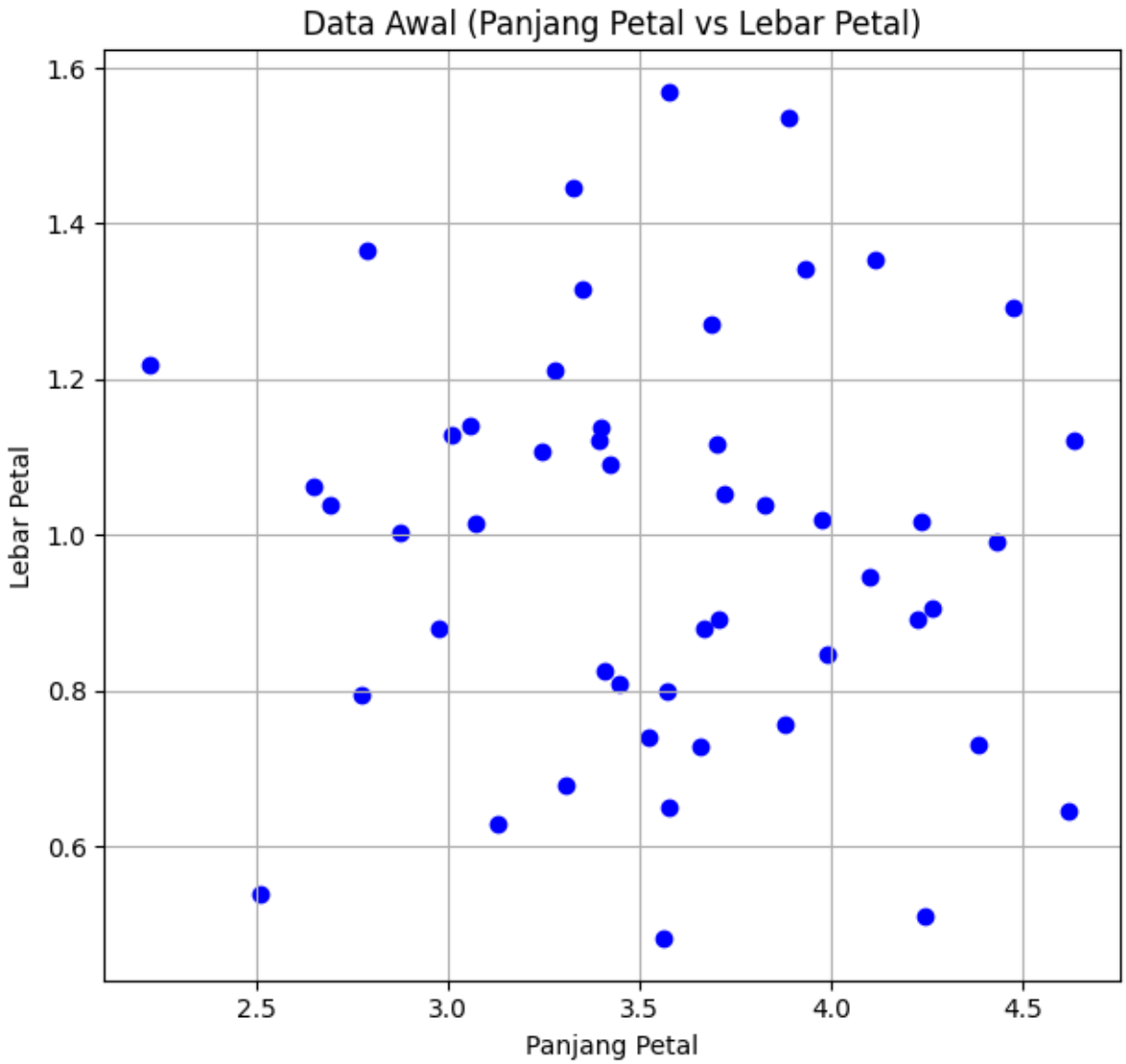
  In this practical work, we will learn about PCA using the Sklearn library in Python. Before heading to the practical work, you have to import some libraries and generate a dataset about data iris using normal distribution. The data iris contains 4 features, as in this source code.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

#Set random seed untuk reproduksibilitas
np.random.seed(0)

#Buat dataset dengan fitur yang terstruktur menggunakan distribusi normal
data = {
    'Panjang Petal': np.random.normal(loc=3.5, scale=0.5, size=50),
    'Lebar Petal': np.random.normal(loc=1.0, scale=0.3, size=50),
    'Panjang Sepal': np.random.normal(loc=5.0, scale=0.5, size=50),
    'Lebar Sepal': np.random.normal(loc=1.5, scale=0.3, size=50)
}
df = pd.DataFrame(data)
print(df)
```

```
    Panjang Petal  Lebar Petal  Panjang Sepal  Lebar Sepal
0        4.382026     0.731360       5.941575     1.479528
1        3.700079     1.116071       4.326120     2.014003
2        3.989369     0.846758       4.364758     1.276574
3        4.620447     0.645810       5.484698     1.252068
4        4.433779     0.991545       4.413438     1.470464
5        3.011361     1.128500       5.971811     1.300957
6        3.975044     1.019955       4.793191     1.837991
7        3.424321     1.090742       4.626273     1.176021
8        3.448391     0.809703       5.961471     1.155759
9        3.705299     0.891178       5.740257     1.368654
10       3.572022     0.798262       5.933779     1.350590
11       4.227137     0.892134       5.453022     2.078860
12       3.880519     0.756056       4.569387     1.784826
13       3.560838     0.482115       5.955032     1.526265
14       3.721932     1.053228       4.865998     1.132369
15       3.666837     0.879466       5.401228     1.753309
16       4.247040     0.510940       5.473626     1.199935
17       3.397421     1.138835       4.922495     1.036569
18       3.656534     0.727810       5.307040     1.856409
19       3.072952     1.015584       5.461103     1.595083
20       2.223505     1.218727       5.188213     1.776258
21       3.826809     1.038695       4.450300     1.595618
22       3.932218     1.341820       5.149119     1.757049
23       3.128917     0.629552       5.663193     1.304692
24       4.634877     1.120702       4.652716     1.189727
25       2.772817     0.794557       4.925183     1.704478
```

Then, visualize the dataset you got. Please note that every practitioner may have a different dataset, as the dataset is generated using a random generator.

```
#Visualisasikan dataset
plt.figure(figsize=(12, 6))

#Scatter plot untuk data asli (Panjang Petal vs Lebar Petal)
plt.subplot(1, 2, 1)
plt.scatter(df['Panjang Petal'], df['Lebar Petal'], color='blue', marker='o')
plt.title('Data Awal (Panjang Petal vs Lebar Petal)')
plt.xlabel('Panjang Petal')
plt.ylabel('Lebar Petal')
plt.grid()
plt.tight_layout()
plt.show()
```

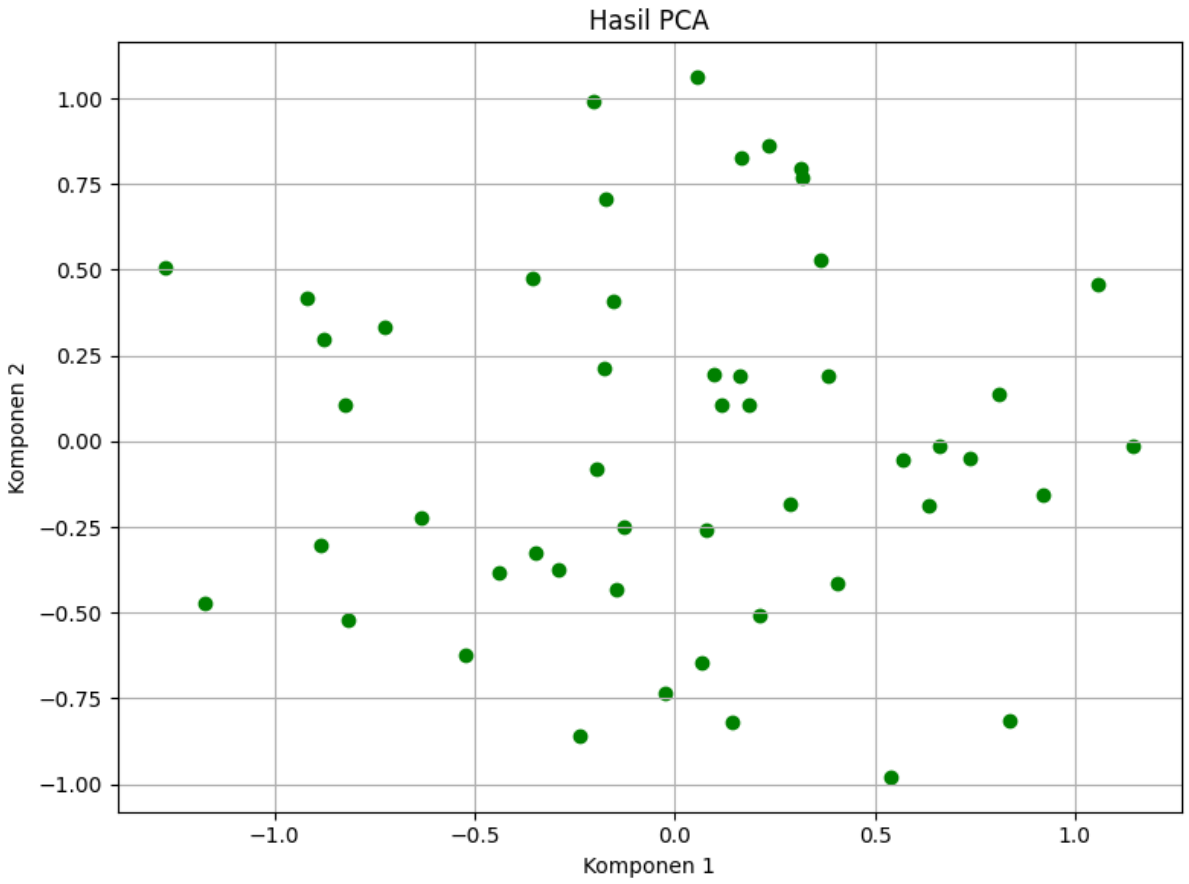


Then, we can do the feature decomposition using Principal Component Analysis and visualize the PCA by using this source code.

```python
#Lakukan PCA menggunakan scikit-learn
pca = PCA(n_components=2)
X_pca = pca.fit_transform(df)

#Buat DataFrame untuk hasil PCA
df_pca = pd.DataFrame(data=X_pca, columns=['Komponen 1', 'Komponen 2'])
print(df_pca)

#Visualisasikan hasil PCA
plt.figure(figsize=(8, 6))
plt.scatter(df_pca['Komponen 1'], df_pca['Komponen 2'], color = 'green')
plt.title('Hasil PCA')
plt.xlabel('Komponen 1')
plt.ylabel('Komponen 2')
plt.grid()

plt.tight_layout()
plt.show()
```

The results obtained should be as follows:

```
    Komponen 1  Komponen 2
0     1.059129    0.459223
1    -0.237823   -0.858861
2     0.143784   -0.818869
3     1.145992   -0.012863
4     0.540615   -0.978556
5    -0.202119    0.990609
6     0.210453   -0.505841
7    -0.288690   -0.376070
8     0.235043    0.862233
9     0.363469    0.529974
10    0.318145    0.767525
11    0.662264   -0.016113
12    0.065761   -0.644798
13    0.316406    0.797746
14    0.079932   -0.260667
15    0.163401    0.188812
16    0.811725    0.136223
17   -0.194343   -0.082501
18    0.118299    0.107496
19   -0.354797    0.474843
20   -1.271430    0.505892
21   -0.023972   -0.735307
22    0.287115   -0.183107
23   -0.172455    0.708021
24    0.835651   -0.812990
25   -0.822898    0.104640
26   -0.127709   -0.249508
```

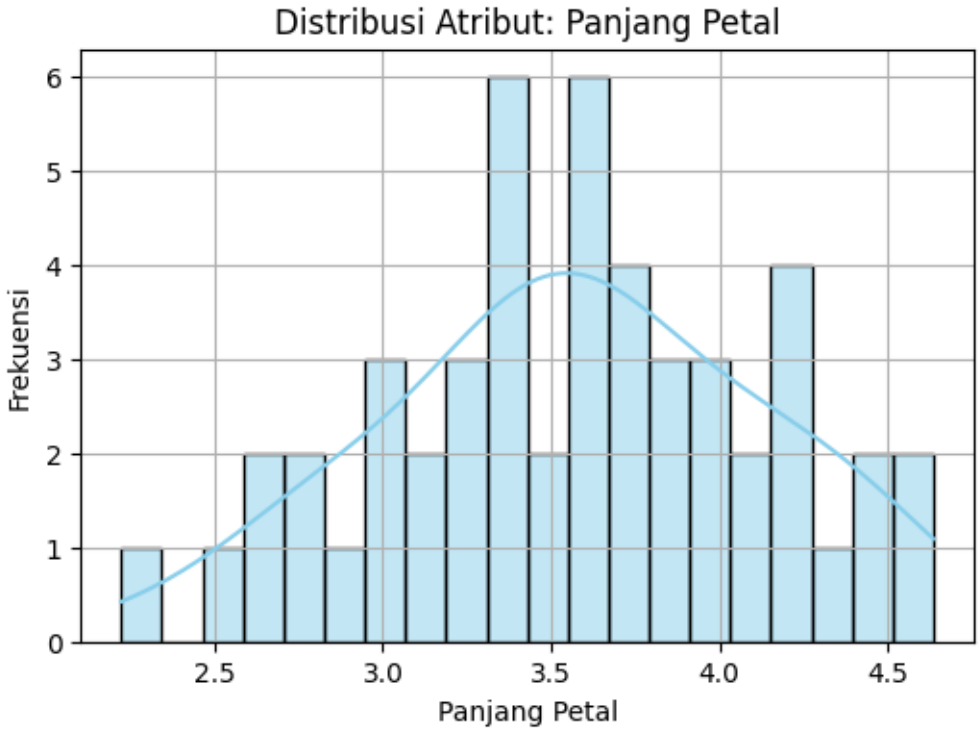


➢ **Histogram Analysis**

Represents data distribution using ranges of values (bins) and their frequency of occurrence. To make histogram you can use this source code.

```
import seaborn as sns

# Histogram untuk setiap kolom fitur
for col in df.columns:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[col], kde=True, bins=20, color='skyblue')
    plt.title(f'Distribusi Atribut: {col}')
    plt.xlabel(col)
    plt.ylabel('Frekuensi')
    plt.grid(True)
    plt.show()
```

One of the results that you will get is as below.



The histogram of petal length is nearly normally distributed.

➢ **Sampling**

Simple random sampling can be done using this source code.

```
# Sampling 30% data secara acak
sampled_data = df.sample(frac=0.3)
print("Jumlah data setelah sampling:", len(sampled_data))
print(sampled_data)
```

```
Jumlah data setelah sampling: 15
    Panjang Petal  Lebar Petal  Panjang Sepal  Lebar Sepal
15       3.666837     0.879466       5.401228     1.753309
25       2.772817     0.794557       4.925183     1.704478
39       3.348849     1.316336       4.453469     1.962904
16       4.247040     0.510940       5.473626     1.199935
22       3.932218     1.341820       5.149119     1.757049
29       4.234679     1.016850       5.203731     1.505244
4        4.433779     0.991545       4.413438     1.470464
1        3.700079     1.116071       4.326120     2.014003
18       3.656534     0.727810       5.307040     1.856409
12       3.880519     0.756056       4.569387     1.784826
13       3.560838     0.482115       5.955032     1.526265
17       3.397421     1.138835       4.922495     1.036569
40       2.975724     0.879047       4.254371     1.112143
26       3.522879     0.738761       4.782423     1.258977
32       3.056107     1.139699       4.662834     1.306914
```

Sampling without replacement can be done using this source code.

```
# Sampling 30% of the data without replacement
sampled_data_no_replacement = df.sample(frac=0.3, random_state=42, replace=False)
print("Number of data points after sampling without replacement:", len(sampled_data_no_replacement))
print(sampled_data_no_replacement)
```

```
Number of data points after sampling without replacement: 15
    Panjang Petal  Lebar Petal  Panjang Sepal  Lebar Sepal
13       3.560838     0.482115       5.955032     1.526265
39       3.348849     1.316336       4.453469     1.962904
30       3.577474     0.650455       4.615042     1.393802
45       3.280963     1.211972       5.472240     1.448536
17       3.397421     1.138835       4.922495     1.036569
48       2.693051     1.038074       4.342046     2.148971
26       3.522879     0.738761       4.782423     1.258977
25       2.772817     0.794557       4.925183     1.704478
32       3.056107     1.139699       4.662834     1.306914
19       3.072952     1.015584       5.461103     1.595083
12       3.880519     0.756056       4.569387     1.784826
4        4.433779     0.991545       4.413438     1.470464
37       4.101190     0.946023       4.895851     1.515650
8        3.448391     0.809703       5.961471     1.155759
3        4.620447     0.645810       5.484698     1.252068
```

Sampling with replacement can be done using this source code.

```
# Sampling 30% of the data with replacement
sampled_data_with_replacement = df.sample(frac=0.3, random_state=42, replace=True)
print("Number of data points after sampling with replacement:", len(sampled_data_with_replacement))
print(sampled_data_with_replacement)
```

```
Number of data points after sampling with replacement: 15
    Panjang Petal  Lebar Petal  Panjang Sepal  Lebar Sepal
38       3.306337     0.678774       5.198003     1.278131
28       4.266390     0.906534       5.336147     1.363340
14       3.721932     1.053228       4.865998     1.132369
42       2.646865     1.062482       5.083337     1.488215
7        3.424321     1.090742       4.626273     1.176021
20       2.223505     1.218727       5.188213     1.776258
38       3.306337     0.678774       5.198003     1.278131
18       3.656534     0.727810       5.307040     1.856409
22       3.932218     1.341820       5.149119     1.757049
10       3.572022     0.798262       5.933779     1.350590
10       3.572022     0.798262       5.933779     1.350590
23       3.128917     0.629552       5.663193     1.304692
35       3.578174     1.568767       5.338217     1.019383
39       3.348849     1.316336       4.453469     1.962904
23       3.128917     0.629552       5.663193     1.304692
```

It can be observed that the results obtained from random sampling, sampling without replacement, and sampling with replacement differ. It was because the sampling had a different process.

- **Min-Max Normalization**

$$v' = \frac{v - min_A}{max_A - min_A}$$

Suppose that: min_A is the minimum and max_A is the maximum of an attribute. v is the value you want to plot in the new range. v' is the new value you get after normalizing the old value. Normalization can be done using the formula above. This source code can be used to find the normalization of the datasets in Python.

```python
import pandas as pd

#Contoh DataFrame
data = {
    'A': [75, 68, 80, 90, 69],
    'B': [15, 29, 80, 26, 57]
}
df = pd.DataFrame(data)
print(df)

#Normalisasi min-max
df_normalized = (df - df.min()) / (df.max() - df.min())
print("Data Normalized (Min-Max):\n", df_normalized)
```

```
    A   B
0  75  15
1  68  29
2  80  80
3  90  26
4  69  57
Data Normalized (Min-Max):
          A         B
0  0.318182  0.000000
1  0.000000  0.215385
2  0.545455  1.000000
3  1.000000  0.169231
4  0.045455  0.646154
```

Min-Max Normalization can also be done using MinMaxScaler in Python.

```python
from sklearn.preprocessing import MinMaxScaler

#Inisialisasi MinMaxScaler
scaler = MinMaxScaler()

#Normalisasi menggunakan Scikit-learn
df_normalized1 = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
print("Data Normalized (Min-Max) dengan Scikit-learn:\n", df_normalized1)
```

```
Data Normalized (Min-Max) dengan Scikit-learn:
          A         B
0  0.318182  0.000000
1  0.000000  0.215385
2  0.545455  1.000000
3  1.000000  0.169231
4  0.045455  0.646154
```

- **Z-Score Normalization**

$$v' = \frac{v - mean(A)}{Standard\ Deviation\ (A)}$$

In Z-score Normalization (or zero-mean normalization) the values of an attribute A, are normalized based on the mean of A and its standard deviation. A value v of attribute A is normalized to v'. Source code for Z-score Normalization in Python using the formula above can be seen below.

```
#Menghitung rata-rata (mean) dan deviasi standar (standard deviation)
mean = df.mean()
std_dev = df.std()

#Menghitung Z-score menggunakan rumus
df_zscore1 = (df - mean) / std_dev

print("Data Normalized (Z-Score):\n", df_zscore1)
```

```
Data Normalized (Z-Score):
          A         B
0 -0.155268 -0.994070
1 -0.931610 -0.466912
2  0.399261  1.453451
3  1.508321 -0.579874
4 -0.820704  0.587405
```

Z-score Normalization also can be done using the StandardScaler in Python.

```
#Inisialisasi StandardScaler
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

#Melakukan standarisasi
df_standardized = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
print("Data Normalized (Z-Score) dengan Scikit-learn:\n", df_standardized)
```

```
Data Normalized (Z-Score) dengan Scikit-learn:
          A         B
0 -0.173595 -1.111404
1 -1.041571 -0.522023
2  0.446388  1.625007
3  1.686354 -0.648319
4 -0.917575  0.656739
```

- **Normalization by Decimal Scaling**

  It normalizes the values of an attribute by changing the position of their decimal points. The number of points by which the decimal point is moved can be determined by the absolute maximum value of attribute A. A value v of attribute A is normalized to v' by computing

$$v' = \frac{v}{10^j}$$

  where j is the smallest integer such that $Max(|v'|) < 1$.

```python
import pandas as pd

#Contoh DataFrame
data = {
    'A': [123, 456, 789, 1011, 1213],
    'B': [50, 200, 300, 400, 500]
}
df1 = pd.DataFrame(data)

#Menentukan nilai maksimum
max_value = df1.max().max()

#Menentukan j (jumlah digit dari nilai maksimum)
j = len(str(max_value))

#Melakukan normalisasi dengan decimal scaling
df1_normalized = df1 / (10 ** j)
print("Data Normalized (Decimal Scaling):\n", df1_normalized)
```

```
Data Normalized (Decimal Scaling):
         A      B
0  0.0123  0.005
1  0.0456  0.020
2  0.0789  0.030
3  0.1011  0.040
4  0.1213  0.050
```

Preparation for Individual Tasks

For the individual task, you need to prepare some of these steps.

- Install Yahoo Finance to Python.

```
pip install yfinance
```

- Then download the data that you want to use, this one is only for example.

```
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#Tentukan nama Saham
ggrm = 'GGRM.JK'
bbni = 'BBNI.JK'
bbca = 'BBCA.JK'
bbri = 'BBRI.JK'
bbsi = 'BBSI.JK'

#tentukan nama data dan hari pertama dan hari terakhir nya
a = yf.download(ggrm, start="2024-09-26", end="2025-09-26")
b = yf.download(bbni, start="2024-09-26", end="2025-09-26")
c = yf.download(bbca, start="2024-09-26", end="2025-09-26")
d = yf.download(bbri, start="2024-09-26", end="2025-09-26")
e = yf.download(bbsi, start="2024-09-26", end="2025-09-26")
```

- Do some preparation to merge and get the combined dataset.

```
print(a.head())
print(b.head())
print(c.head())
print(d.head())
print(e.head())
```

```
ggrm_close = a[['Close']]
bbni_close = b[['Close']]
bbca_close = c[['Close']]
bbri_close = d[['Close']]
bbsi_close = e[['Close']]
```

```
ggrm_close = a[['Close']]
bbni_close = b[['Close']]
bbca_close = c[['Close']]
bbri_close = d[['Close']]
bbsi_close = e[['Close']]
ggrm_close.rename(columns={'Close':'GGRM'}, inplace=True)
bbni_close.rename(columns={'Close':'BBNI'}, inplace=True)
bbca_close.rename(columns={'Close':'BBCA'}, inplace=True)
bbri_close.rename(columns={'Close':'BBRI'}, inplace=True)
bbsi_close.rename(columns={'Close':'BBSI'}, inplace=True)
```

```
data_saham = pd.concat([ggrm_close, bbni_close, bbca_close, bbri_close, bbsi_close], axis = 1)
print(data_saham)
```

```
Price             GGRM         BBNI          BBCA         BBRI    BBSI
Ticker          GGRM.JK      BBNI.JK       BBCA.JK      BBRI.JK BBSI.JK
Date
2024-09-26  15284.047852  5139.621582  10329.081055  4619.627930  4270.0
2024-09-27  15260.388672  5001.953125  10280.814453  4665.366699  4270.0
2024-09-30  15047.453125  4910.174316   9967.080078  4528.150391  4270.0
2024-10-01  15236.728516  4933.119141  10184.280273  4642.497070  4270.0
2024-10-02  15213.069336  4910.174316  10136.013672  4519.002441  4260.0
...                  ...          ...           ...          ...     ...
2025-09-19  10900.000000  4270.000000   7800.000000  4250.000000  4050.0
2025-09-22  13075.000000  4210.000000   7725.000000  4160.000000  4000.0
2025-09-23  14825.000000  4200.000000   7875.000000  4140.000000  4220.0
2025-09-24  13700.000000  4190.000000   7775.000000  4170.000000  4400.0
2025-09-25  14150.000000  4190.000000   7700.000000  4070.000000  4380.0

[235 rows x 5 columns]
```

❖ Then, visualize the dataset using a graphic.

```
#Membuat grafik harga penutupan
plt.figure(figsize=(12, 6))
plt.plot(data_saham['GGRM'], label='Saham GGRM', color='blue')
plt.plot(data_saham['BBNI'], label='Saham BBNI', color='red')
plt.plot(data_saham['BBCA'], label='Saham BBCA', color='orange')
plt.plot(data_saham['BBRI'], label='Saham BBRI', color='green')
plt.plot(data_saham['BBSI'], label='Saham BBSI', color='purple')

#Menambahkan judul dan label
plt.title(f'Grafik Harga Saham')
plt.xlabel('Tanggal')
plt.ylabel('Harga Saham')
plt.legend()
plt.grid()

#Menampilkan grafik
plt.show()
```