Modul Praktikum 4
Data Mining

PROGRAM STUDI SAINS DATA
FAKULTAS SAINS
INSTITUT TEKNOLOGI SUMATERA
2024

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 several data reduction techniques can be used in data mining:

Data Sampling: This technique involves selecting a subset of the data to work with, rather than using the entire dataset. This can be useful for reducing the size of a dataset while still preserving the overall trends and patterns in the data.

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.

Data Compression: This technique involves using techniques such as lossy or lossless compression to reduce the size of a dataset.

Data Discretization: This technique involves converting continuous data into discrete data by partitioning the range of possible values into intervals or bins.

Feature Selection: This technique involves selecting a subset of features from the dataset that are most relevant to the task at hand.

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:

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.

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.

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.

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.

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.**

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 principal component analysis
2. Understanding the concept of principal component analysis in Python
3. Learning the concept of data normalization
4. Understanding the concept of data normalization in Python

Principal Component Analysis

Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation that converts a set of correlated variables to a set of uncorrelated variables. PCA works on the condition that while the data in a higher dimensional space is mapped to data in a lower dimension space, the variance of the data in the lower dimensional space should be maximum. The main goal of Principal Component Analysis (PCA) is to reduce the dimensionality of a dataset while preserving the most important patterns or relationships between the variables without any prior knowledge of the target variables. In another words, PCA is used to reduce the dimensionality of a data set by finding a new set of variables, smaller than the original set of variables, retaining most of the sample's information, and useful for the regression and classification of data.

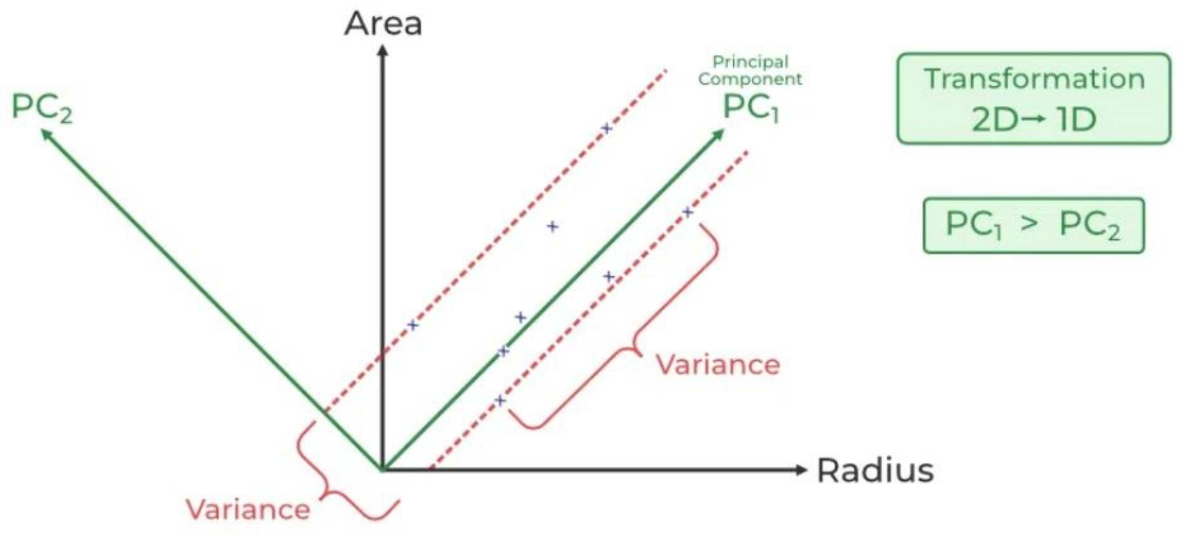
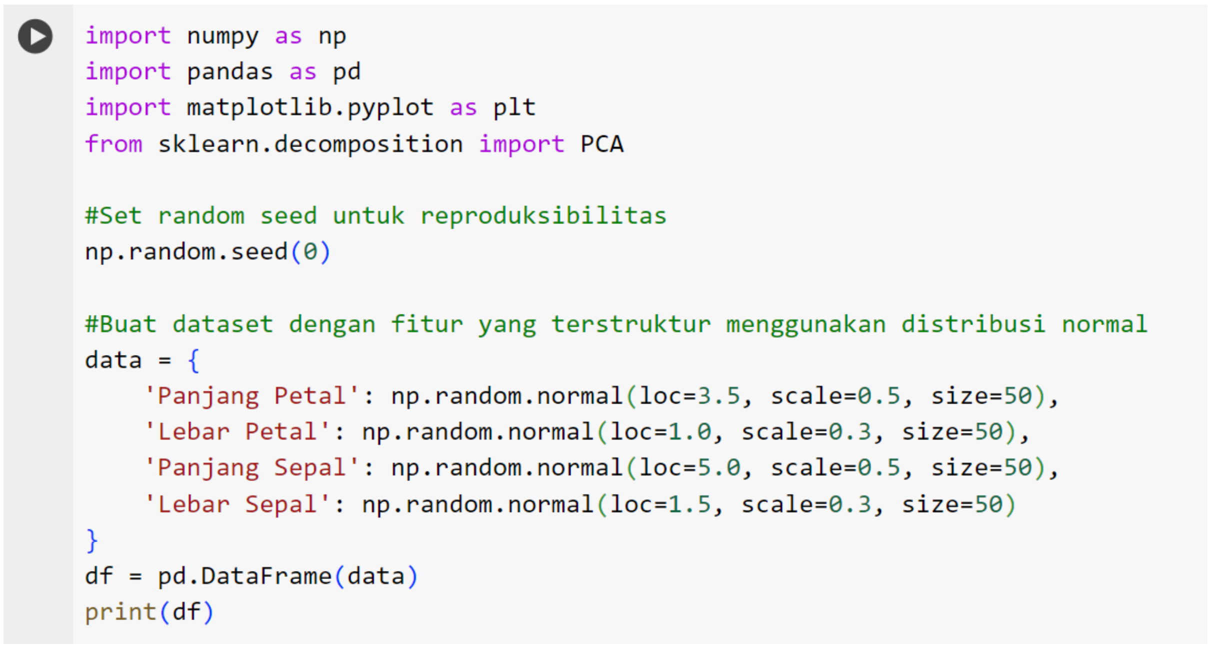


Figure 1. Principal Component Analysis

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)
```

Then, visualize the dataset.

```
#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.

```
#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()
```

Data Normalization

In this practical work, we will learn about Data Normalization using Min-Max Normalization, Z-score Normalization, and Normalization by Decimal Scaling.

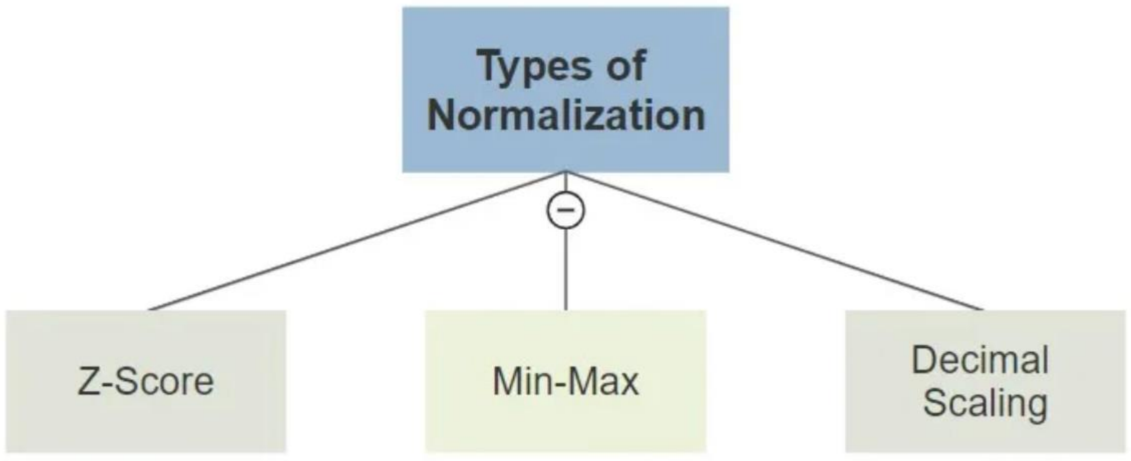
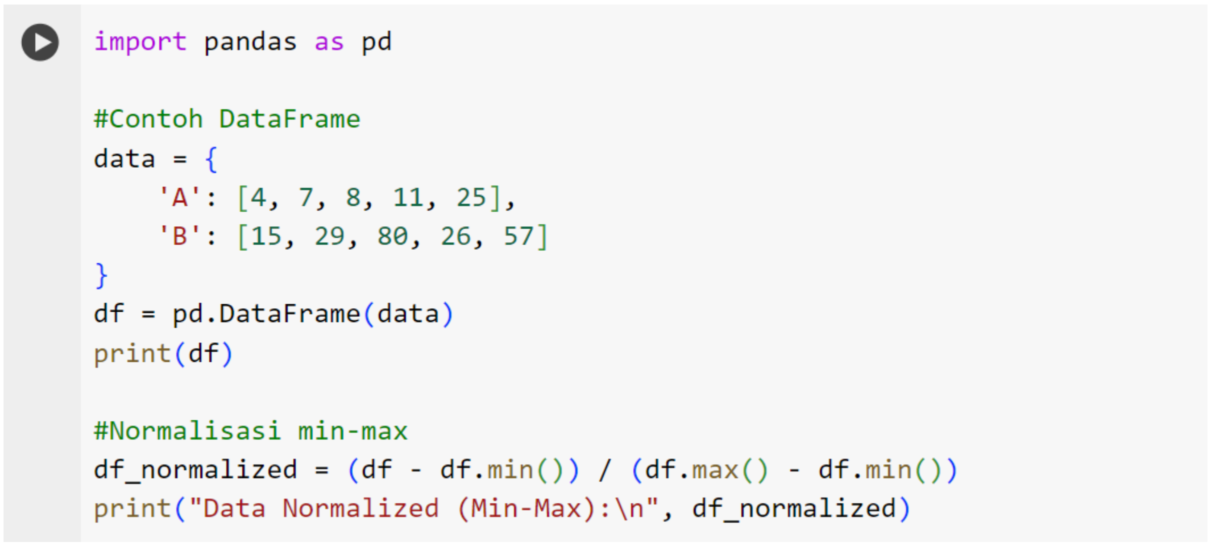


Figure 2. Types of Normalization

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.



```
import pandas as pd

#Contoh DataFrame
data = {
    'A': [4, 7, 8, 11, 25],
    '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)
```

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)
```

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.

```python
#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)
```

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

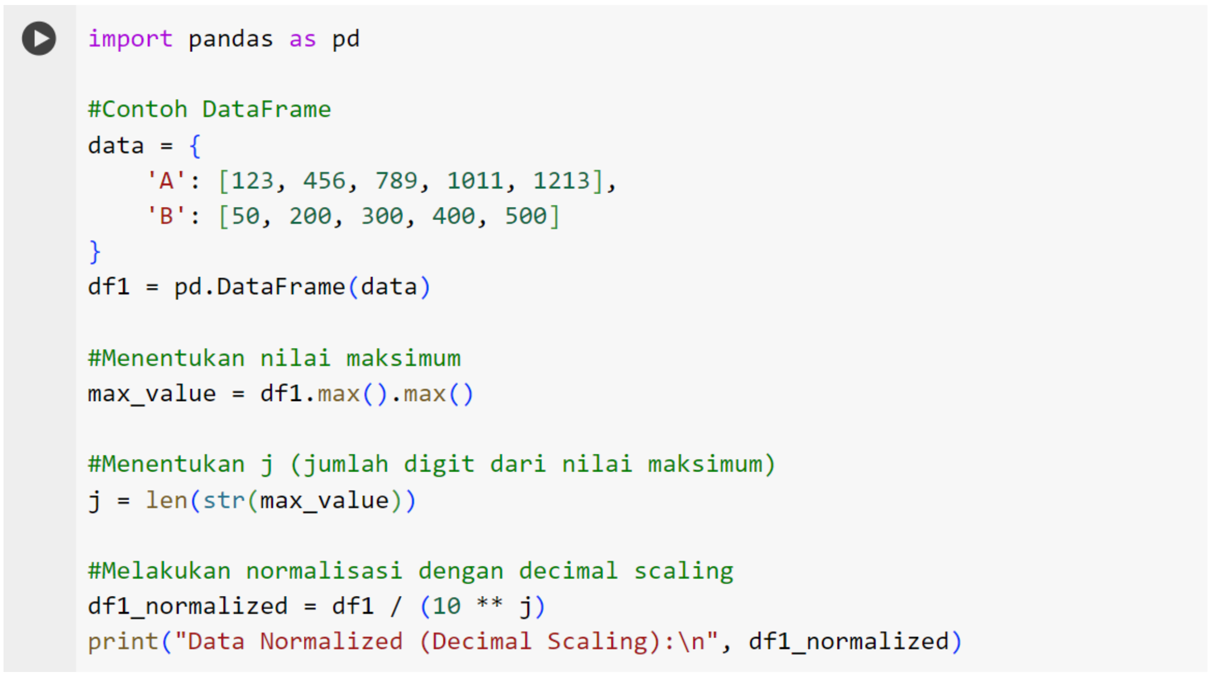
```python
#Inisialisasi StandardScaler
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

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

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 $Max(|v'|) < 1$.

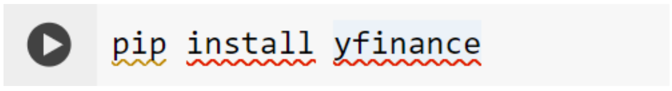


```python
import pandas as pd

#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)
```

Preparing for the Individual Task

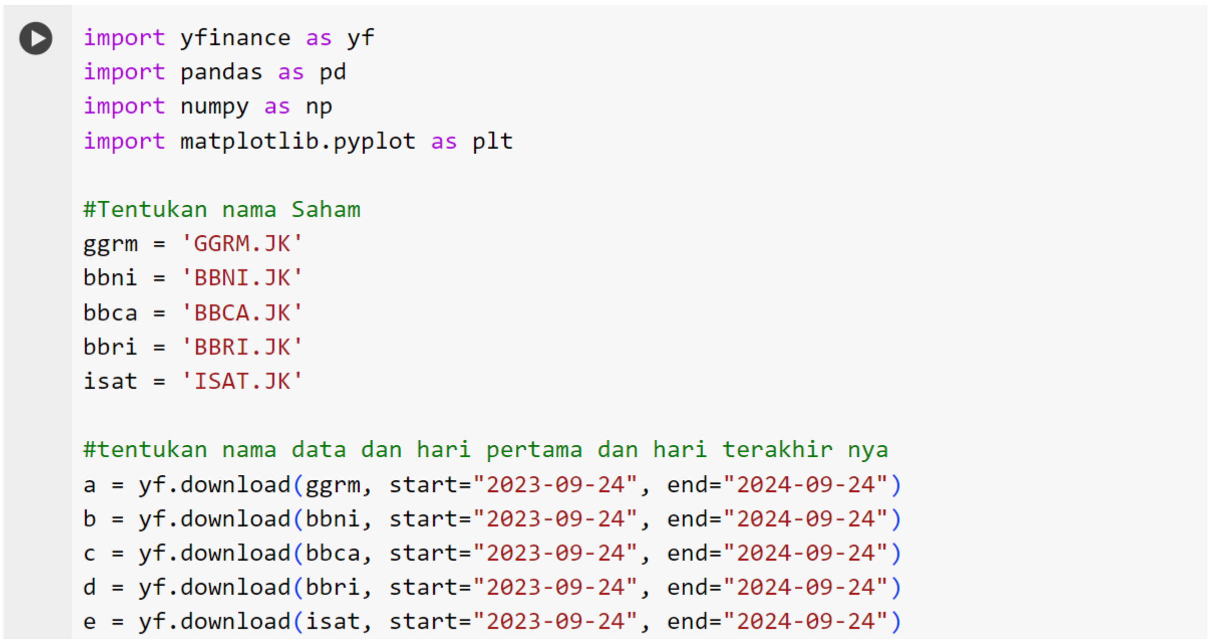
For the individual task, you need to prepare some of these steps.

- Install Yahoo Finance to Python.



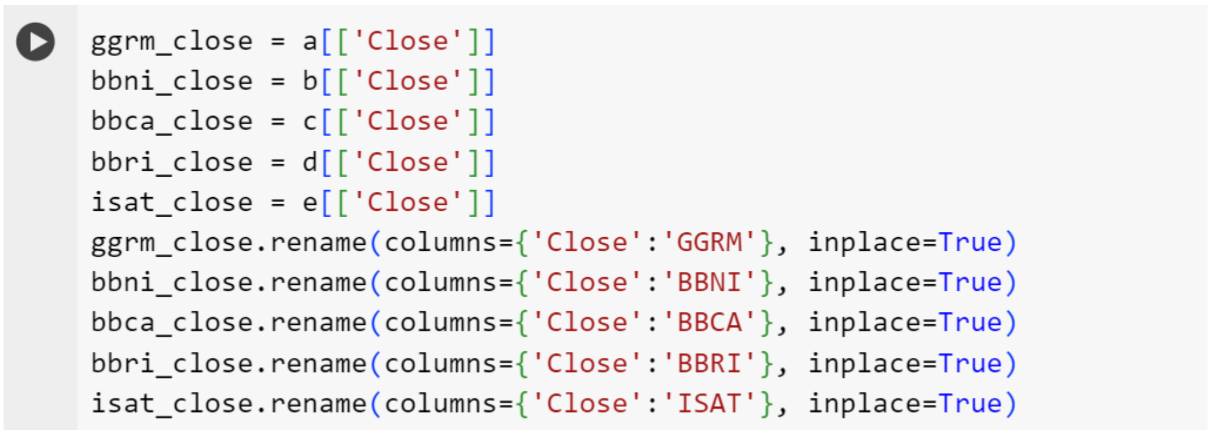
```
pip install yfinance
```

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

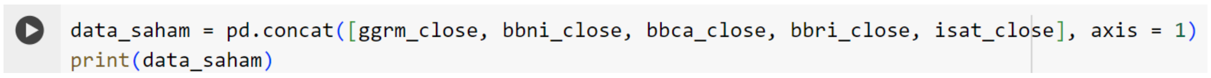


```
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

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

#tentukan nama data dan hari pertama dan hari terakhir nya
a = yf.download(ggrm, start="2023-09-24", end="2024-09-24")
b = yf.download(bbni, start="2023-09-24", end="2024-09-24")
c = yf.download(bbca, start="2023-09-24", end="2024-09-24")
d = yf.download(bbri, start="2023-09-24", end="2024-09-24")
e = yf.download(isat, start="2023-09-24", end="2024-09-24")
```

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

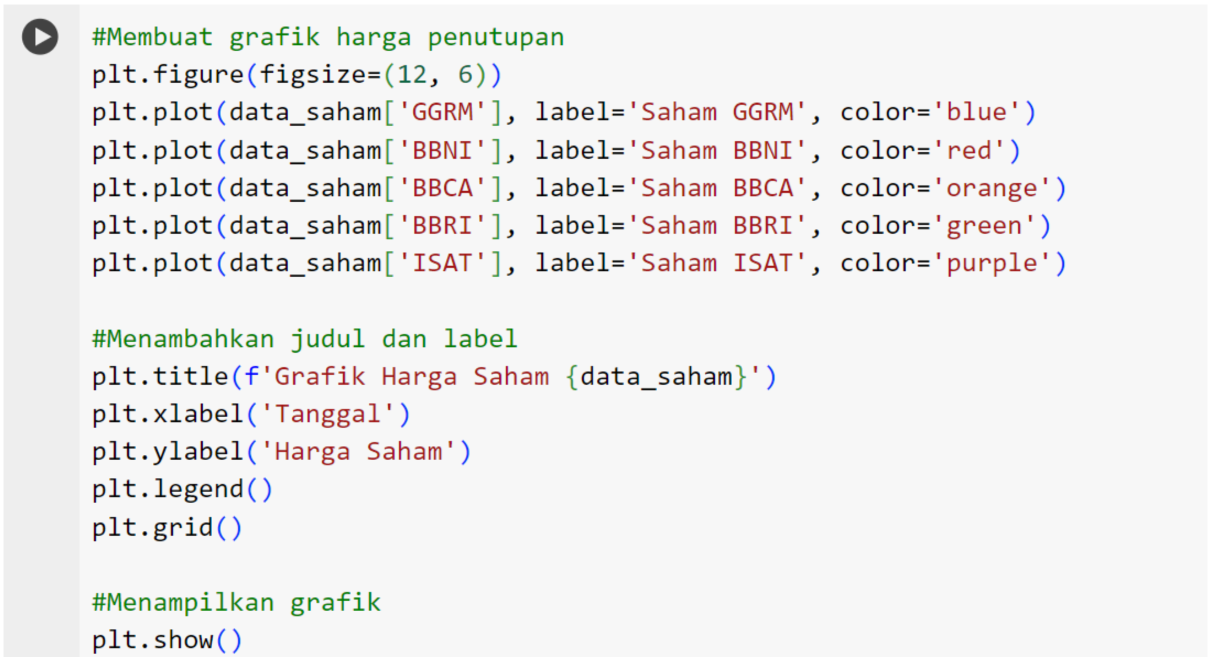


```
ggrm_close = a[['Close']]
bbni_close = b[['Close']]
bbca_close = c[['Close']]
bbri_close = d[['Close']]
isat_close = e[['Close']]
ggrm_close.rename(columns={'Close':'GGRM'}, inplace=True)
bbni_close.rename(columns={'Close':'BBNI'}, inplace=True)
bbca_close.rename(columns={'Close':'BBCA'}, inplace=True)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



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

- Then, visualize the dataset using a graphic.

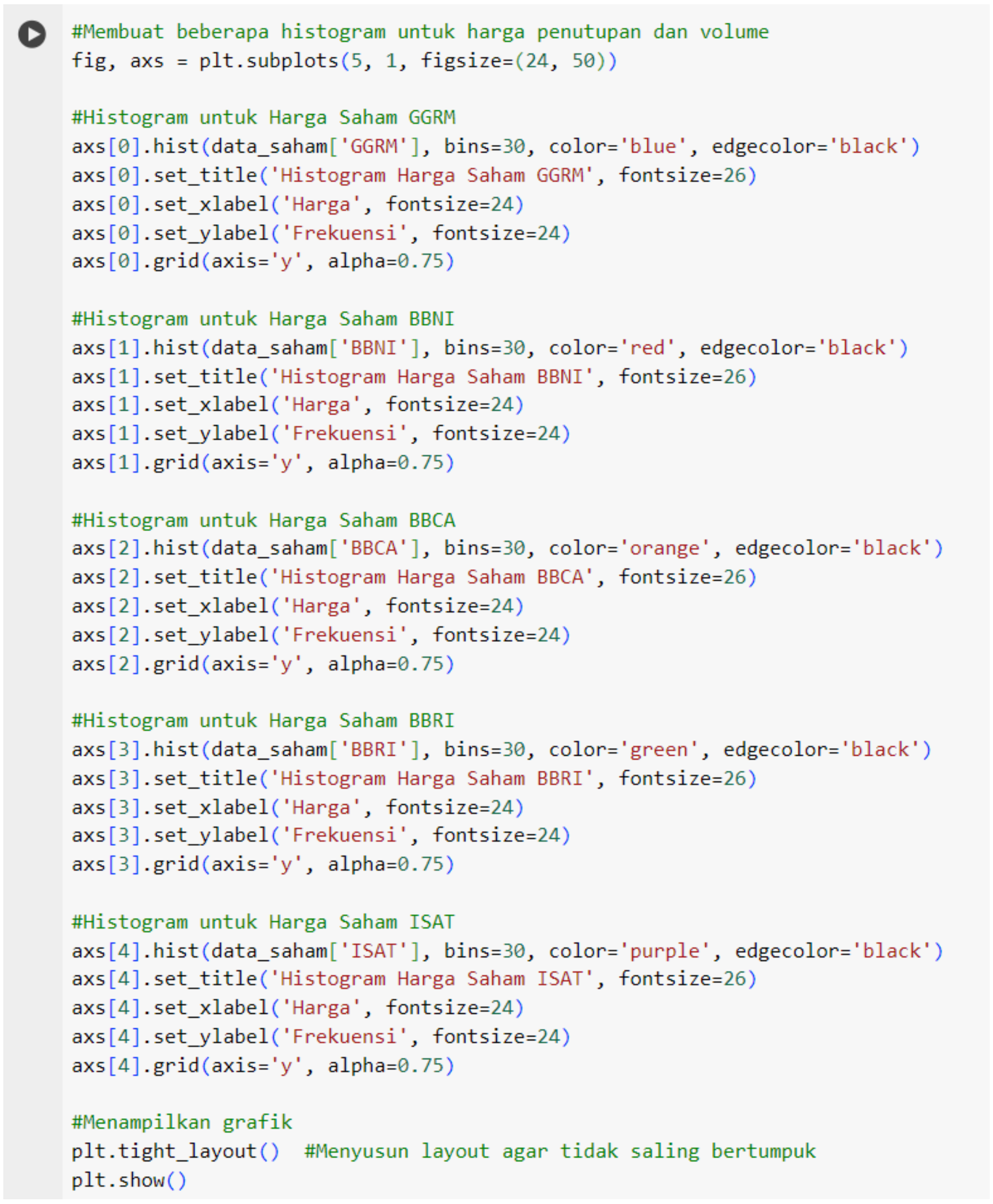


```
#Membuat grafik harga penutupan
plt.figure(figsize=(12, 6))
plt.plot(data_saham['GGRM'], label='Saham GGRM', color='blue')
plt.plot(data_saham['BBNI'], label='Saham BBNI', color='red')
plt.plot(data_saham['BBCA'], label='Saham BBCA', color='orange')
plt.plot(data_saham['BBRI'], label='Saham BBRI',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

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

#Menampilkan grafik
plt.show()
```

For easier coding in making the graph, you can copy and paste this source code.

```
#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['ISAT'], label='Saham ISAT', color='purple')

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

#Menampilkan grafik
plt.show()
```

- Make histogram for dataset.



```
#Membuat beberapa histogram untuk harga penutupan dan volume
fig, axs = plt.subplots(5, 1, figsize=(24, 50))

#Histogram untuk Harga Saham GGRM
axs[0].hist(data_saham['GGRM'], bins=30, color='blue', edgecolor='black')
axs[0].set_title('Histogram Harga Saham GGRM', fontsize=26)
axs[0].set_xlabel('Harga', fontsize=24)
axs[0].set_ylabel('Frekuensi', fontsize=24)
axs[0].grid(axis='y', alpha=0.75)

#Histogram untuk Harga Saham BBNI
axs[1].hist(data_saham['BBNI'], bins=30, color='red', edgecolor='black')
axs[1].set_title('Histogram Harga Saham BBNI', fontsize=26)
axs[1].set_xlabel('Harga', fontsize=24)
axs[1].set_ylabel('Frekuensi', fontsize=24)
axs[1].grid(axis='y', alpha=0.75)

#Histogram untuk Harga Saham BBCA
axs[2].hist(data_saham['BBCA'], bins=30, color='orange', edgecolor='black')
axs[2].set_title('Histogram Harga Saham BBCA', fontsize=26)
axs[2].set_xlabel('Harga', fontsize=24)
axs[2].set_ylabel('Frekuensi', fontsize=24)
axs[2].grid(axis='y', alpha=0.75)

#Histogram untuk Harga Saham BBRI
axs[3].hist(data_saham['BBRI'], bins=30, color='green', edgecolor='black')
axs[3].set_title('Histogram Harga Saham BBRI', fontsize=26)
axs[3].set_xlabel('Harga', fontsize=24)
axs[3].set_ylabel('Frekuensi', fontsize=24)
axs[3].grid(axis='y', alpha=0.75)

#Histogram untuk Harga Saham ISAT
axs[4].hist(data_saham['ISAT'], bins=30, color='purple', edgecolor='black')
axs[4].set_title('Histogram Harga Saham ISAT', fontsize=26)
axs[4].set_xlabel('Harga', fontsize=24)
axs[4].set_ylabel('Frekuensi', fontsize=24)
axs[4].grid(axis='y', alpha=0.75)

#Menampilkan grafik
plt.tight_layout()  #Menyusun layout agar tidak saling bertumpuk
plt.show()
```

For easier coding in making the graph, you can copy and paste this source code.

```
#Membuat beberapa histogram untuk harga penutupan dan volume
fig, axs = plt.subplots(5, 1, figsize=(24, 50))

#Histogram untuk Harga Saham GGRM
axs[0].hist(data_saham['GGRM'], bins=30, color='blue',
edgecolor='black')
axs[0].set_title('Histogram Harga Saham GGRM', fontsize=26)
axs[0].set_xlabel('Harga', fontsize=24)
axs[0].set_ylabel('Frekuensi', fontsize=24)
axs[0].grid(axis='y', alpha=0.75)

#Histogram untuk Harga Saham BBNI
axs[1].hist(data_saham['BBNI'], bins=30, color='red',
edgecolor='black')
axs[1].set_title('Histogram Harga Saham BBNI', fontsize=26)
axs[1].set_xlabel('Harga', fontsize=24)
axs[1].set_ylabel('Frekuensi', fontsize=24)
axs[1].grid(axis='y', alpha=0.75)

#Histogram untuk Harga Saham BBCA
axs[2].hist(data_saham['BBCA'], bins=30, color='orange',
edgecolor='black')
axs[2].set_title('Histogram Harga Saham BBCA', fontsize=26)
axs[2].set_xlabel('Harga', fontsize=24)
axs[2].set_ylabel('Frekuensi', fontsize=24)
axs[2].grid(axis='y', alpha=0.75)

#Histogram untuk Harga Saham BBRI
axs[3].hist(data_saham['BBRI'], bins=30, color='green',
edgecolor='black')
axs[3].set_title('Histogram Harga Saham BBRI', fontsize=26)
axs[3].set_xlabel('Harga', fontsize=24)
axs[3].set_ylabel('Frekuensi', fontsize=24)
axs[3].grid(axis='y', alpha=0.75)

#Histogram untuk Harga Saham ISAT
axs[4].hist(data_saham['ISAT'], bins=30, color='purple',
edgecolor='black')
axs[4].set_title('Histogram Harga Saham ISAT', fontsize=26)
axs[4].set_xlabel('Harga', fontsize=24)
axs[4].set_ylabel('Frekuensi', fontsize=24)
axs[4].grid(axis='y', alpha=0.75)

#Menampilkan grafik
plt.tight_layout()  #Menyusun layout agar tidak saling
bertumpuk
plt.show()
```