

**LAPORAN TUGAS CASE BASED 2**  
**PEMBELAJARAN MESIN**

Untuk memenuhi tugas mata kuliah Pembelajaran Mesin - DDR



Disusun oleh:

Vania Amadea

1301204365

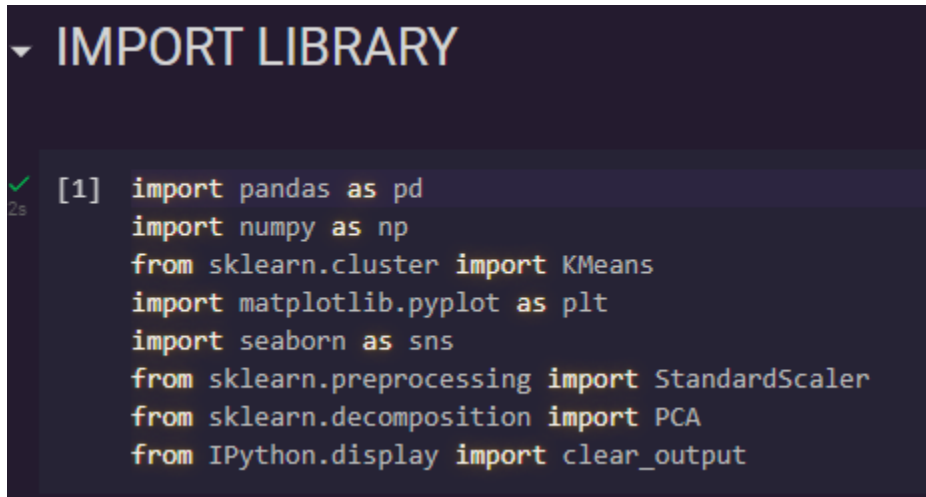
IF4408

**Program Studi S1 Informatika**  
**Fakultas Informatika**  
**Universitas Telkom**  
**Bandung**  
**2022**

\*Saya mengerjakan tugas ini dengan cara yang tidak melanggar aturan perkuliahan dan kode etik akademisi.

<b>I. Library yang digunakan</b>	<b>3</b>
<b>II. Ikhtisar kumpulan data</b>	<b>4</b>
<b>III. Preprocessing dan visualisasi data</b>	<b>6</b>
<b>IV. Elbow Method</b>	<b>23</b>
<b>V. Algoritma/metode yang diterapkan</b>	<b>24</b>
<b>VI. Evaluasi hasil</b>	<b>24</b>
<b>VII. Link</b>	<b>24</b>
<b>VIII. Reference Link</b>	<b>24</b>

## I. Library yang digunakan



```
▶ IMPORT LIBRARY

[1] import pandas as pd
    import numpy as np
    from sklearn.cluster import KMeans
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.preprocessing import StandardScaler
    from sklearn.decomposition import PCA
    from IPython.display import clear_output
```

- pandas = untuk memproses data yang meliputi pembersihan data, manipulasi data, hingga melakukan analisis data.
- numpy = untuk digunakan untuk bekerja dengan array dan juga memiliki fungsi yang bekerja dalam domain aljabar linier, transformasi fourier, dan matriks.
- sklearn.cluster KMeans = untuk menggunakan library KMeans, **tetapi ini hanya digunakan untuk MELIHAT KESAMAAN OUTPUT DENGAN SCRATCH.**
- matplotlib.pyplot = untuk melakukan visualisasi data seperti membuat plot grafik untuk satu sumbu atau lebih.
- seaborn = untuk membuat grafik dan statistik dengan menggunakan Python. Library ini dibangun berdasarkan library Matplotlib yang sudah ada. Kemudian terintegrasi dengan struktur data pada Pandas.
- sklearn.preprocessing StandardScaler = untuk data scaling.
- sklearn.decomposition PCA = untuk reduksi dimensi.
- IPython.display clear\_output = untuk clear output.

## II. Ikhtisar kumpulan data

Data yang digunakan pada Case Based 2 merupakan Water Treatment Plant Dataset sesuai dengan ketentuan soal untuk mahasiswa yang memiliki NIM ganjil. Berdasarkan situs yang menyediakan dataset ini, berasal dari pengukuran harian sensor di instalasi pengolahan air limbah perkotaan. Tujuannya adalah untuk mengklasifikasikan keadaan operasional pembangkit untuk memprediksi kesalahan melalui variabel keadaan pembangkit pada setiap tahapan proses perawatan. Domain ini telah dinyatakan sebagai domain yang tidak terstruktur.

- Isi data

```

IMPORT DATASET

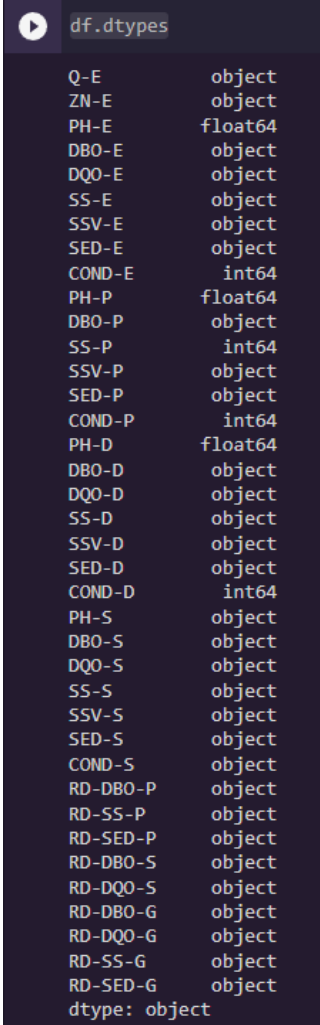
df = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/water-treatment/water-treatment_data",
names = ['Q-E', 'ZN-E', 'PH-E', 'DBO-E', 'DQO-E', 'SS-E', 'SSV-E', 'SED-E', 'COND-E', 'PH-P', 'DBO-P', 'RD-DBO-P', 'RD-SS-P', 'RD-SSV-P', 'RD-DBO-S', 'RD-DQO-S', 'RD-DBO-G', 'RD-DQO-G', 'RD-SS-G', 'RD-SSV-G', 'RD-SED-G', 'RD-COND-G', 'PH-S', 'DBO-S', 'DQO-S', 'SS-S', 'SSV-S', 'SED-S', 'COND-S', 'PH-S'])
df

```

Data asli tidak berlabel, sehingga diberi label secara manual yang didapatkan dari informasi atribut.

N.	Attrib.
1	Q-E (input flow to plant)
2	ZN-E (input Zinc to plant)
3	PH-E (input pH to plant)
4	DBO-E (input Biological demand of oxygen to plant)
5	DQO-E (input chemical demand of oxygen to plant)
6	SS-E (input suspended solids to plant)
7	SSV-E (input volatile suspended solids to plant)
8	SED-E (input sediments to plant)
9	COND-E (input conductivity to plant)
10	PH-P (input pH to primary settler)
11	DBO-P (input Biological demand of oxygen to primary settler)
12	SS-P (input suspended solids to primary settler)
13	SSV-P (input volatile suspended solids to primary settler)
14	SED-P (input sediments to primary settler)
15	COND-P (input conductivity to primary settler)
16	PH-D (input pH to secondary settler)
17	DBO-D (input Biological demand of oxygen to secondary settler)
18	DQO-D (input chemical demand of oxygen to secondary settler)
19	SS-D (input suspended solids to secondary settler)
20	SSV-D (input volatile suspended solids to secondary settler)
21	SED-D (input sediments to secondary settler)
22	COND-D (input conductivity to secondary settler)
23	PH-S (output pH)
24	DBO-S (output Biological demand of oxygen)
25	DQO-S (output chemical demand of oxygen)
26	SS-S (output suspended solids)
27	SSV-S (output volatile suspended solids)
28	SED-S (output sediments)
29	COND-S (output conductivity)
30	RD-DBO-P (performance input Biological demand of oxygen in primary settler)
31	RD-SS-P (performance input suspended solids to primary settler)
32	RD-SED-P (performance input sediments to primary settler)
33	RD-SSV-P (performance input Biological demand of oxygen to secondary settler)
34	RD-DQO-S (performance input chemical demand of oxygen to secondary settler)
35	RD-DBO-G (global performance input Biological demand of oxygen)
36	RD-DQO-G (global performance input chemical demand of oxygen)

- Informasi data  
Dapat dilihat tipe data setiap label.



```
df.dtypes
```

Q-E	object
ZN-E	object
PH-E	float64
DBO-E	object
DQO-E	object
SS-E	object
SSV-E	object
SED-E	object
COND-E	int64
PH-P	float64
DBO-P	object
SS-P	int64
SSV-P	object
SED-P	object
COND-P	int64
PH-D	float64
DBO-D	object
DQO-D	object
SS-D	object
SSV-D	object
SED-D	object
COND-D	int64
PH-S	object
DBO-S	object
DQO-S	object
SS-S	object
SSV-S	object
SED-S	object
COND-S	object
RD-DBO-P	object
RD-SS-P	object
RD-SED-P	object
RD-DBO-S	object
RD-DQO-S	object
RD-DBO-G	object
RD-DQO-G	object
RD-SS-G	object
RD-SED-G	object
dtype:	object

Berdasarkan data yang ada, harus dilakukan tahap *preprocessing* data untuk memastikan kualitas data baik sebelum digunakan ke dalam model yang dipilih. Tahap ini dapat dilakukan dengan *replace* “?” dengan nan agar bisa dilakukan pengecekan berapa banyak data null, mengganti tipe data dari object ke float, mengisi nilai null, melihat *outliers*, *outliers handling*, *scaling* data, dan reduksi dimensi.

### III. Preprocessing dan visualisasi data

- Replace “?” dengan nan

```
#replace ? to nan  
df = df.replace(['?'], np.nan)
```

- Mengganti tipe data dari object ke float

Hal ini dilakukan untuk memudahkan proses-proses selanjutnya.

```
#change data type from object to float  
df = df.apply(pd.to_numeric)
```

- Cek null / missing value dan tipe data yang sudah diubah

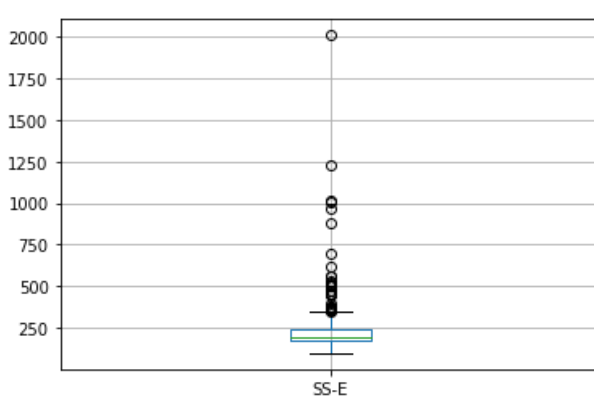
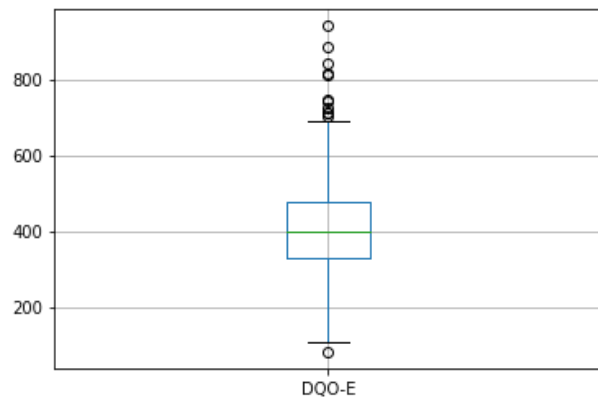
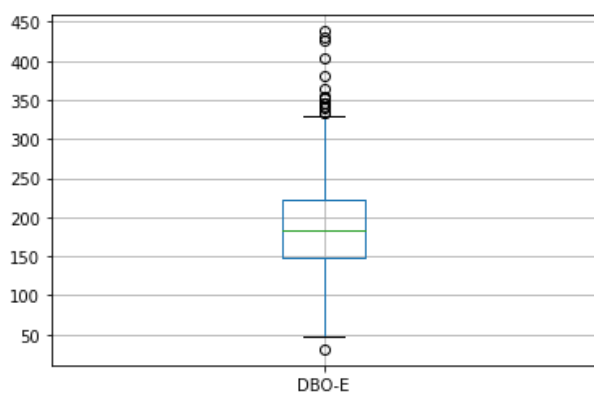
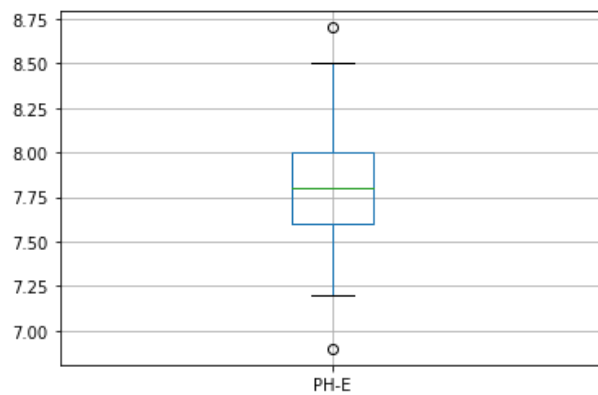
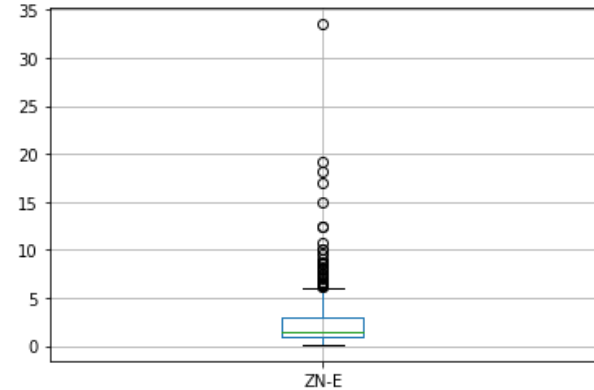
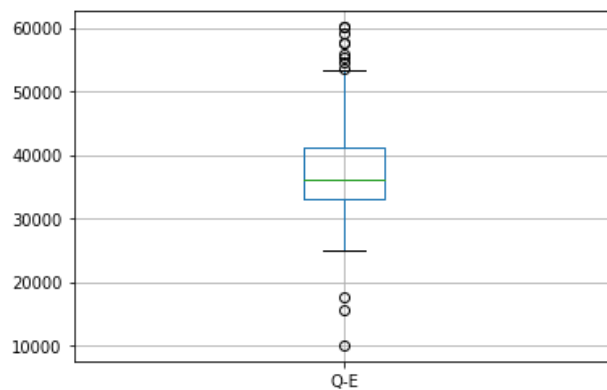
df.isna().sum()		df.dtypes	
Q-E	18	Q-E	float64
ZN-E	3	ZN-E	float64
PH-E	0	PH-E	float64
DBO-E	23	DBO-E	float64
DQO-E	6	DQO-E	float64
SS-E	1	SS-E	float64
SSV-E	11	SSV-E	float64
SED-E	25	SED-E	float64
COND-E	0	COND-E	int64
PH-P	0	PH-P	float64
DBO-P	40	DBO-P	float64
SS-P	0	SS-P	int64
SSV-P	11	SSV-P	float64
SED-P	24	SED-P	float64
COND-P	0	COND-P	int64
PH-D	0	PH-D	float64
DBO-D	28	DBO-D	float64
DQO-D	9	DQO-D	float64
SS-D	2	SS-D	float64
SSV-D	13	SSV-D	float64
SED-D	25	SED-D	float64
COND-D	0	COND-D	int64
PH-S	1	PH-S	float64
DBO-S	23	DBO-S	float64
DQO-S	18	DQO-S	float64
SS-S	5	SS-S	float64
SSV-S	17	SSV-S	float64
SED-S	28	SED-S	float64
COND-S	1	COND-S	float64
RD-DBO-P	62	RD-DBO-P	float64
RD-SS-P	4	RD-SS-P	float64
RD-SED-P	27	RD-SED-P	float64
RD-DBO-S	40	RD-DBO-S	float64
RD-DQO-S	26	RD-DQO-S	float64
RD-DBO-G	36	RD-DBO-G	float64
RD-DQO-G	25	RD-DQO-G	float64
RD-SS-G	8	RD-SS-G	float64
RD-SED-G	31	RD-SED-G	float64
dtype: int64		dtype: object	

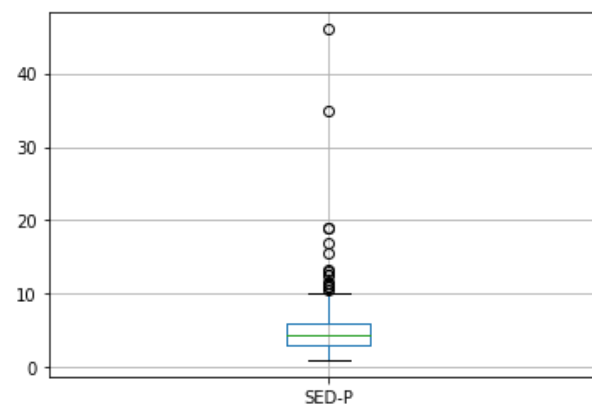
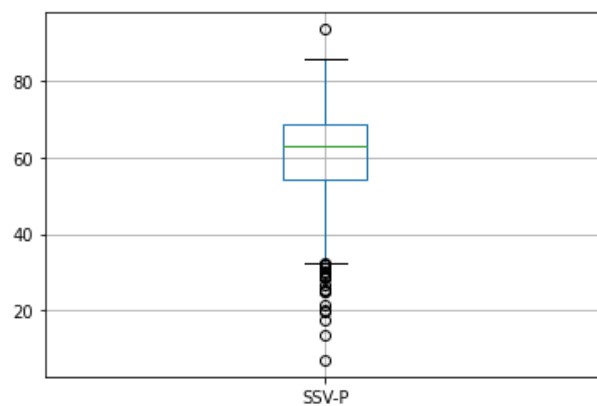
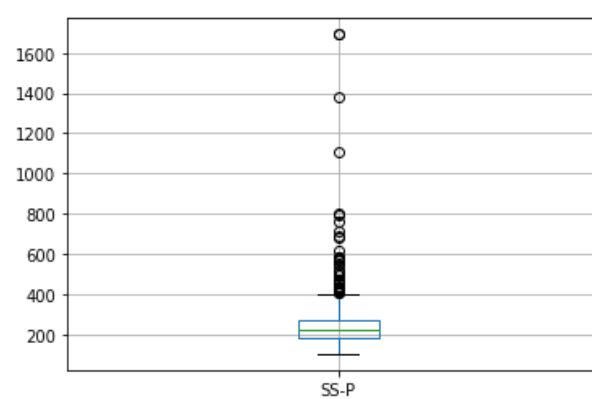
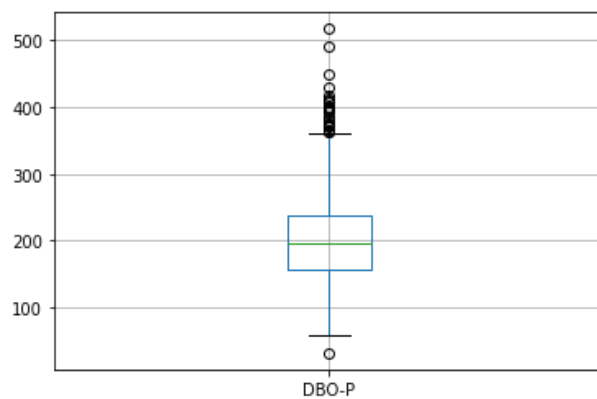
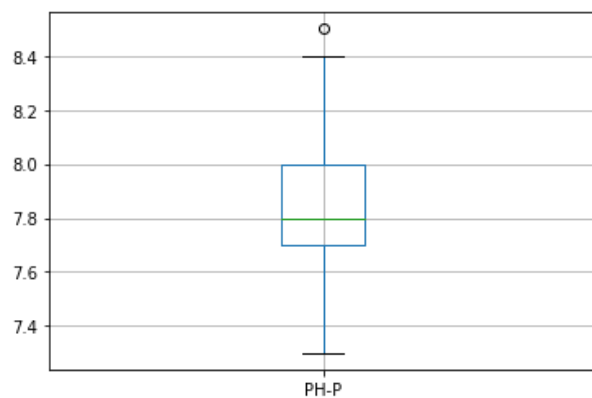
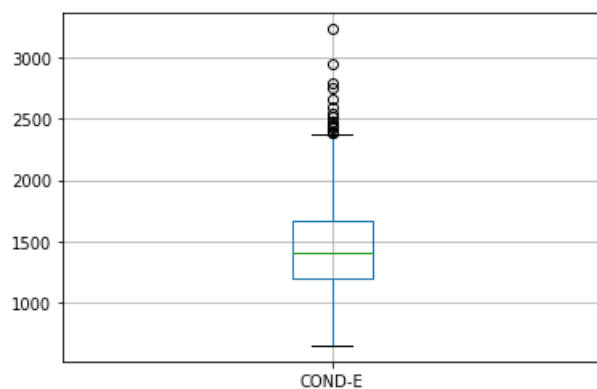
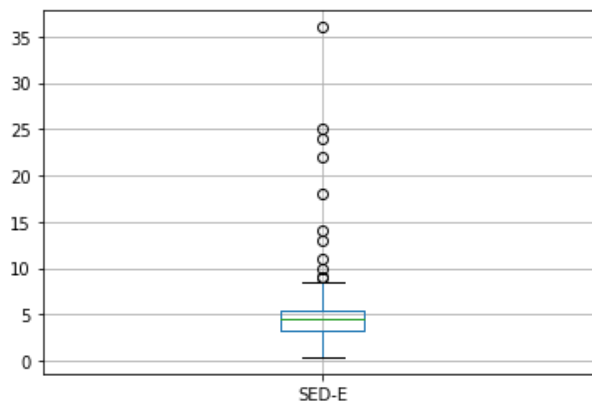
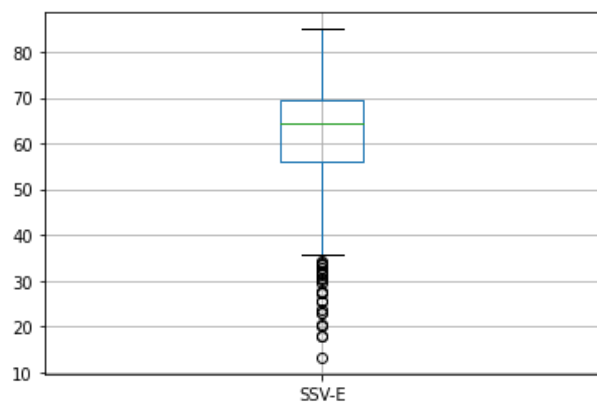
- Isi missing value dengan median  
Diisi dengan median karena data memiliki banyak *outliers*. Pengecekan *outliers* dilakukan dengan *plotting* dengan boxplot.

```
✓ [25] #fill null with median because there are so many outliers
df = df.fillna(df.median())
```

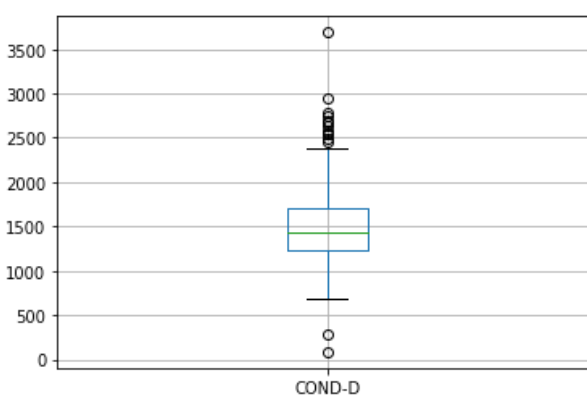
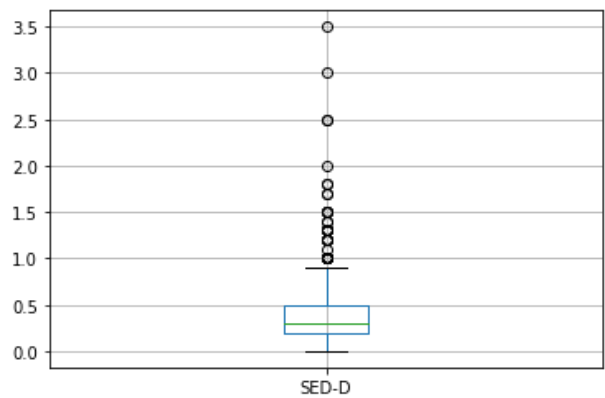
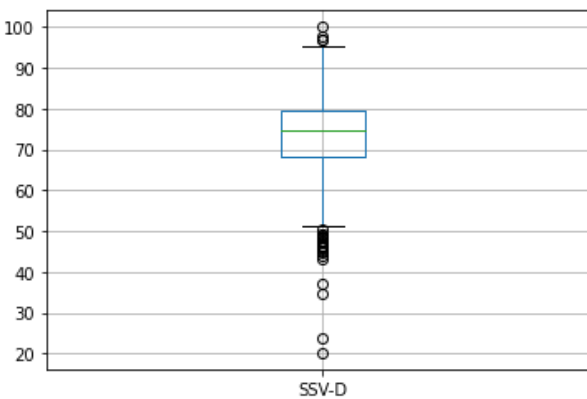
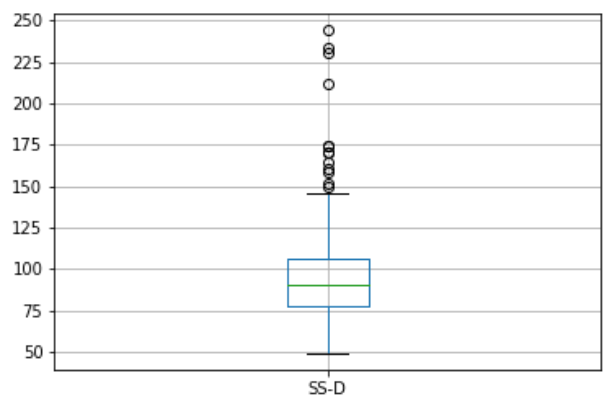
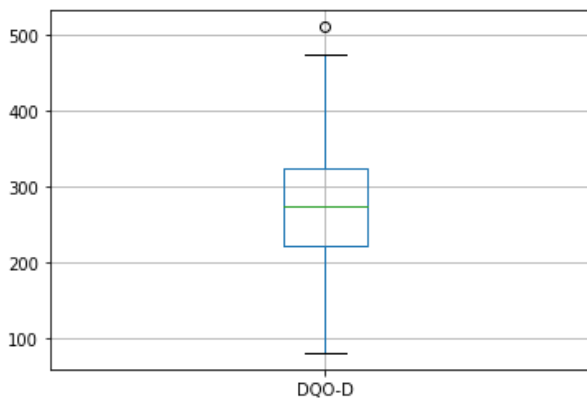
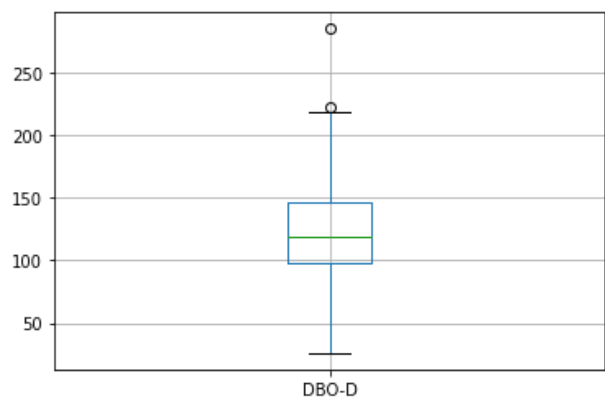
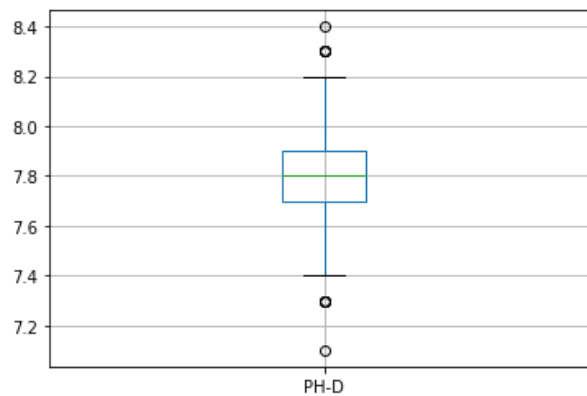
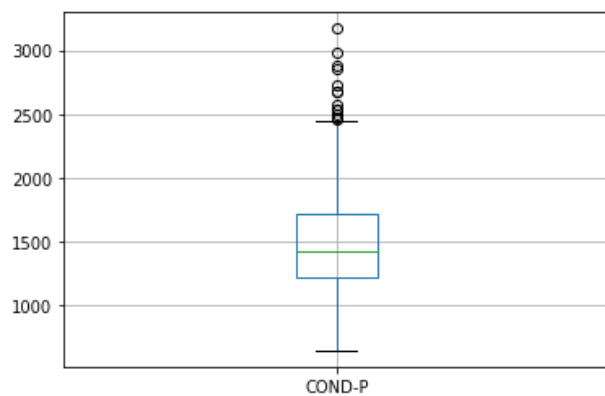
- Visualisasi data dengan boxplot untuk melihat *outliers*.

```
▶ for column in df:
    plt.figure()
    df.boxplot([column])
```

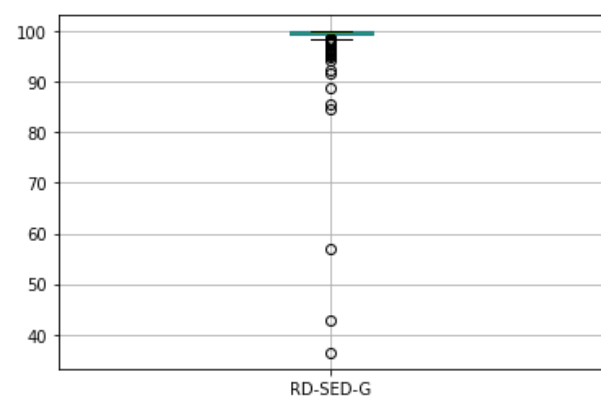
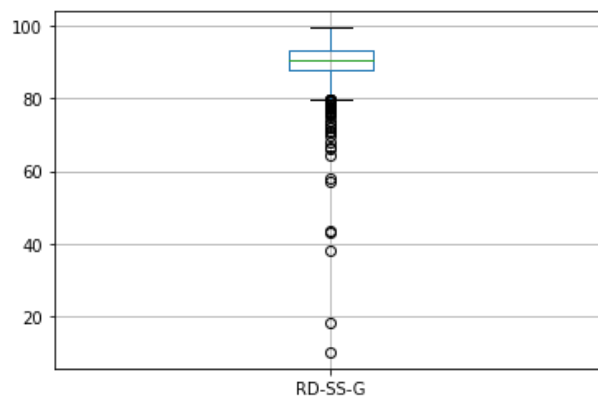
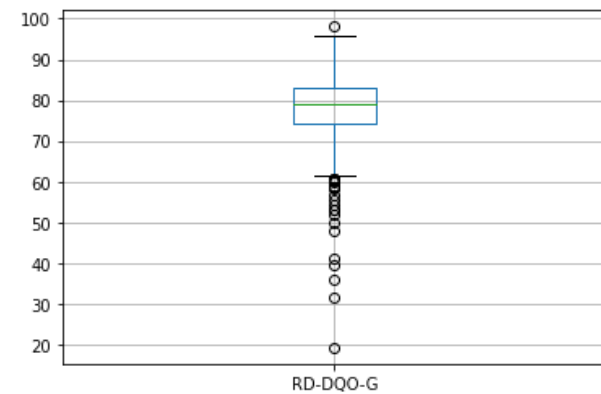
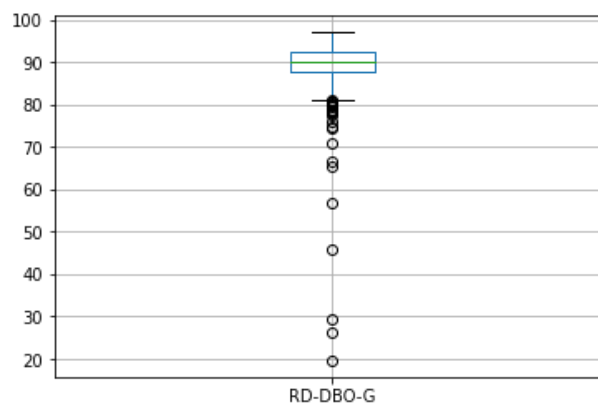
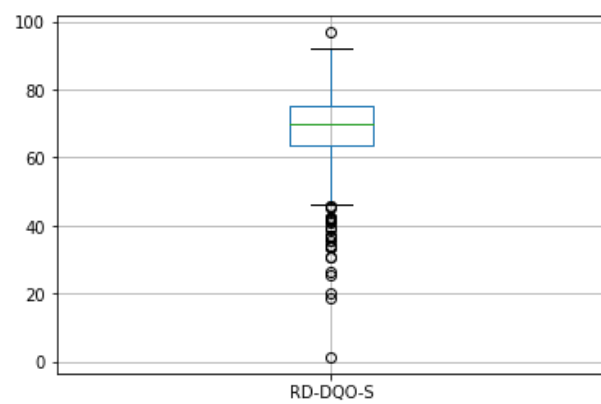
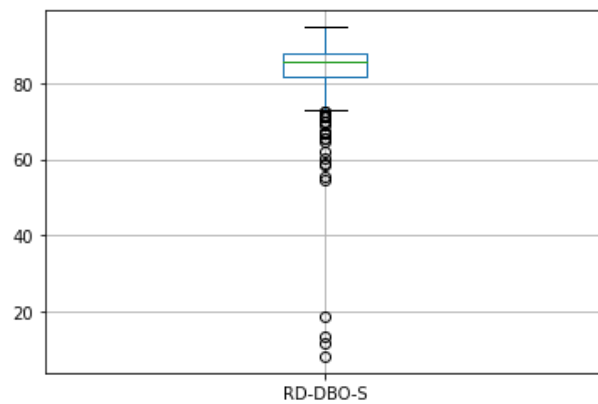
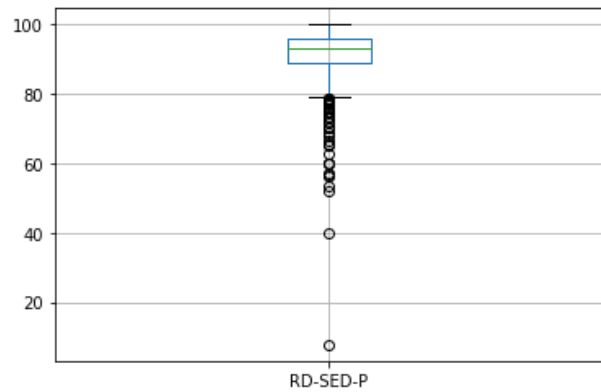
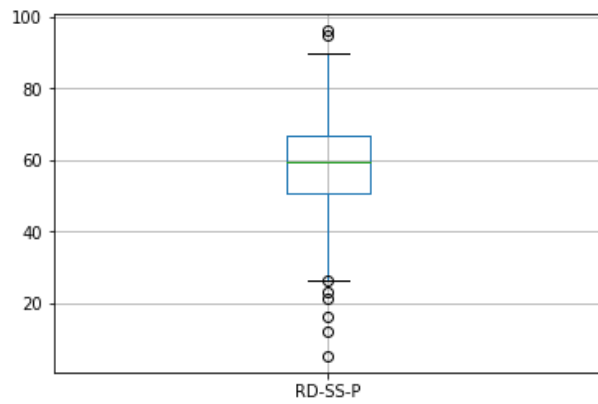












- Memeriksa kembali data apakah sudah berhasil menghilangkan *missing value* atau null.

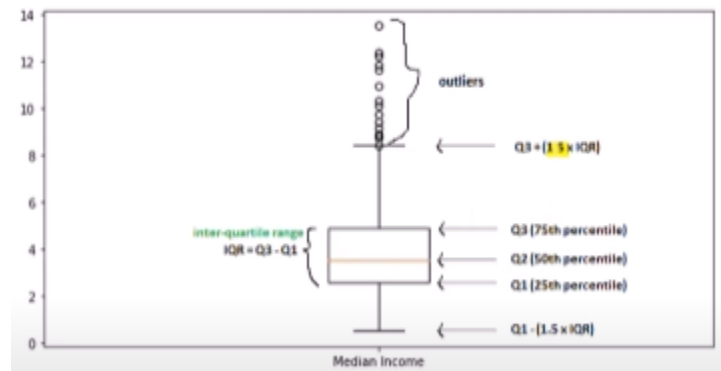
```
df.isna().sum()
```

Q-E	0
ZN-E	0
PH-E	0
DBO-E	0
DQO-E	0
SS-E	0
SSV-E	0
SED-E	0
COND-E	0
PH-P	0
DBO-P	0
SS-P	0
SSV-P	0
SED-P	0
COND-P	0
PH-D	0
DBO-D	0
DQO-D	0
SS-D	0
SSV-D	0
SED-D	0
COND-D	0
PH-S	0
DBO-S	0
DQO-S	0
SS-S	0
SSV-S	0
SED-S	0
COND-S	0
RD-DBO-P	0
RD-SS-P	0
RD-SED-P	0
RD-DBO-S	0
RD-DQO-S	0
RD-DBO-G	0
RD-DQO-G	0
RD-SS-G	0
RD-SED-G	0

dtype: int64

- *Outliers handling* dengan IQR

Hal ini dilakukan untuk mengatasi nilai pencilan atau *outliers* dengan memasukan nilai-nilai pencilan dengan nilai upper/lower bound dari teori statistika.



```
def iqr_capping(df, cols, factor):
    for col in cols:
        q1 = df[col].quantile(0.25)
        q3 = df[col].quantile(0.75)

        iqr = q3 - q1

        upper_whisker = q3 + (factor*iqr)
        lower_whisker = q1 - (factor*iqr)

        df[col] = np.where(df[col]>upper_whisker, upper_whisker,
                           np.where(df[col]<lower_whisker, lower_whisker, df[col]))
```

Keberhasilan *outliers handling* yang dilakukan dapat dilihat dari plot yang dibuat dari setiap iterasi dengan membandingkan boxplot sebelum dan sesudah *di-handling*.

```
df_cap = df.copy()
features = ["Q-E", "ZN-E", "PH-E", "DBO-E", "DQO-E", "SS-E", "SSV-E", "SED-E", "COND-E", "PH-P", "DBO-P", "SS-P", "SSV-P", "SED-P",
            "iqr_capping(df_cap, features, 1.5)

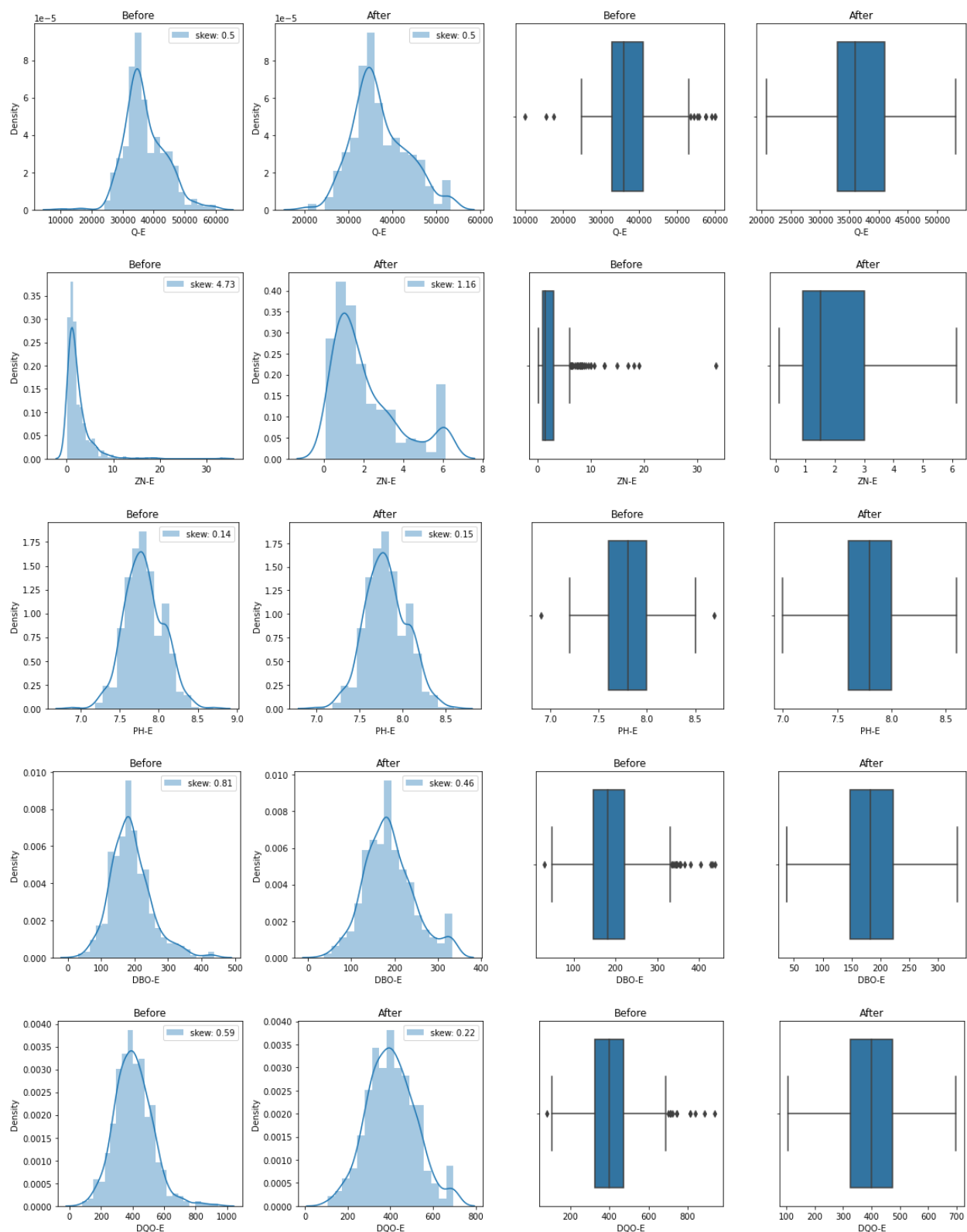
for col in features:
    plt.figure(figsize=(16,4))

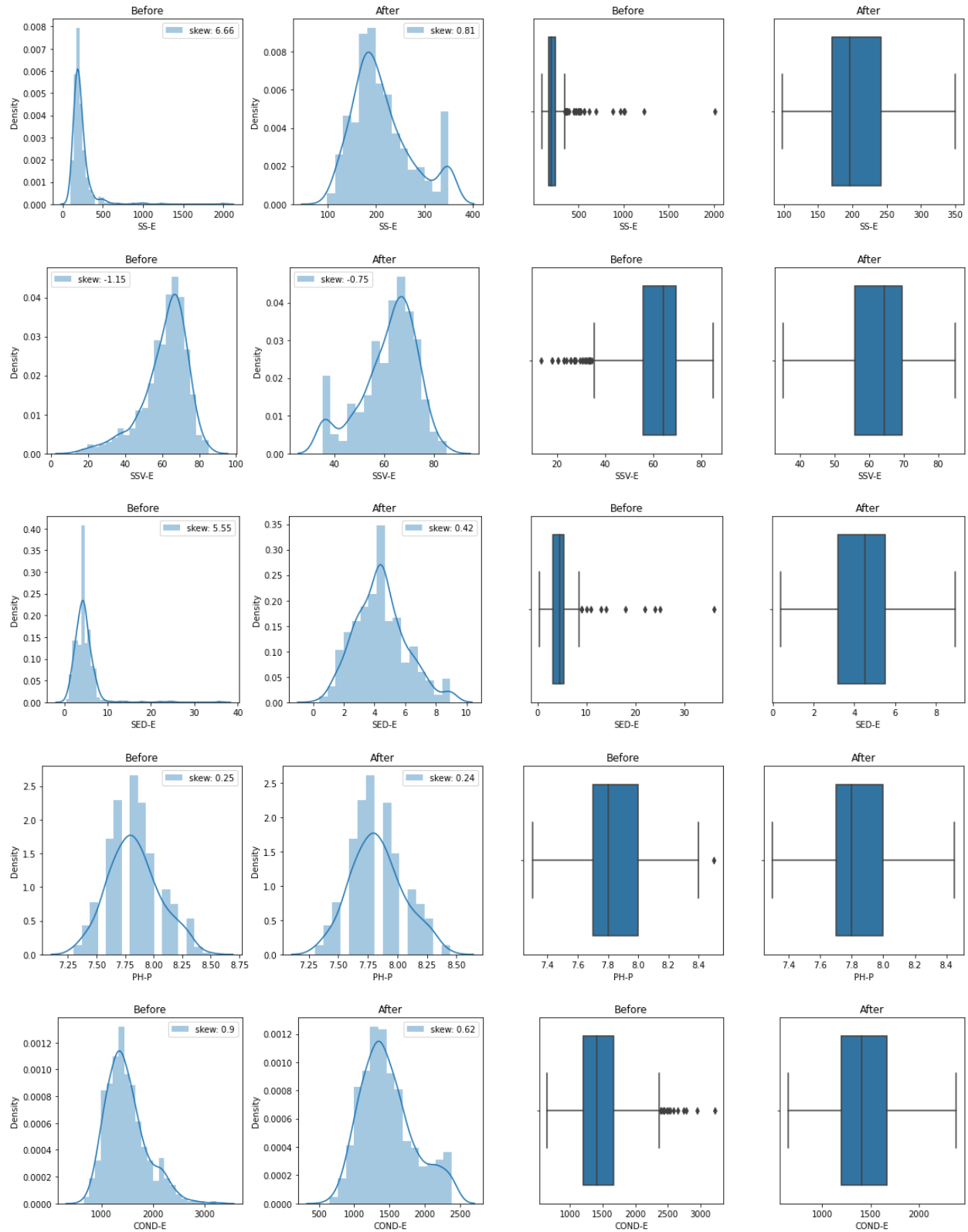
    plt.subplot(141)
    sns.distplot(df[col], label="skew: " + str(np.round(df[col].skew(),2)))
    plt.title('Before')
    plt.legend()

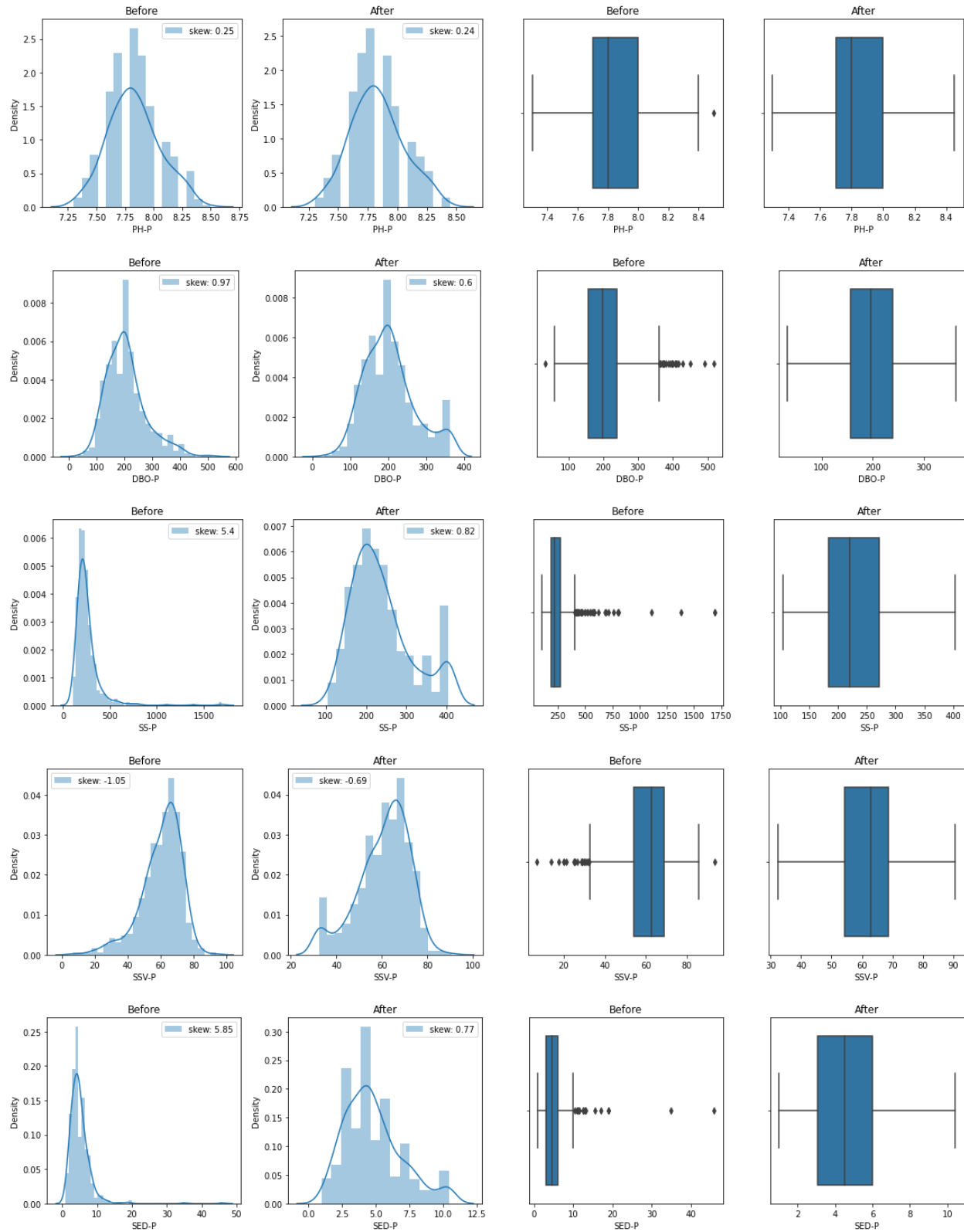
    plt.subplot(142)
    sns.distplot(df_cap[col], label="skew: " + str(np.round(df_cap[col].skew(),2)))
    plt.title('After')
    plt.legend()

    plt.subplot(143)
    sns.boxplot(df[col])
    plt.title('Before')

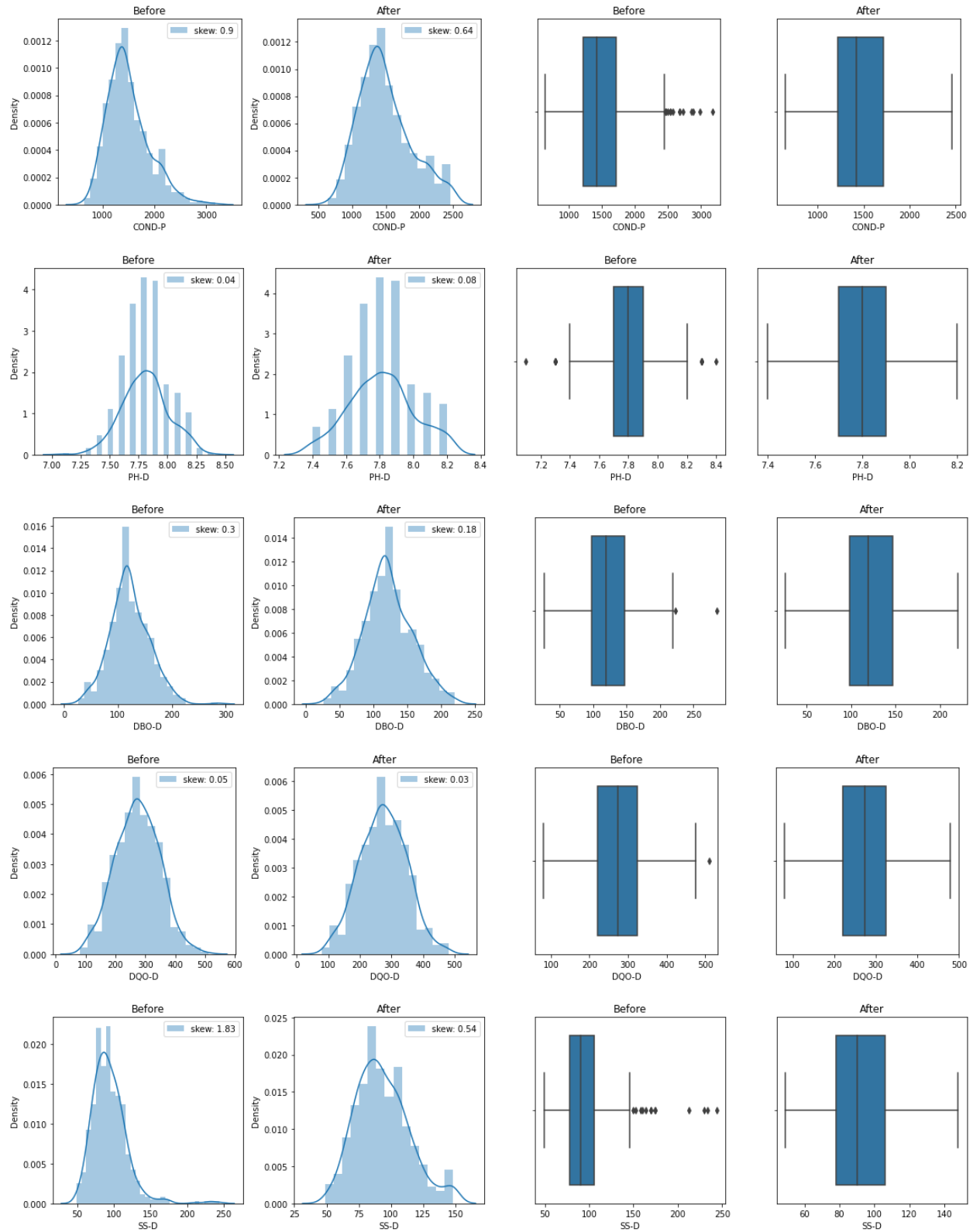
    plt.subplot(144)
    sns.boxplot(df_cap[col])
    plt.title('After')
    plt.tight_layout()
    plt.show()
```

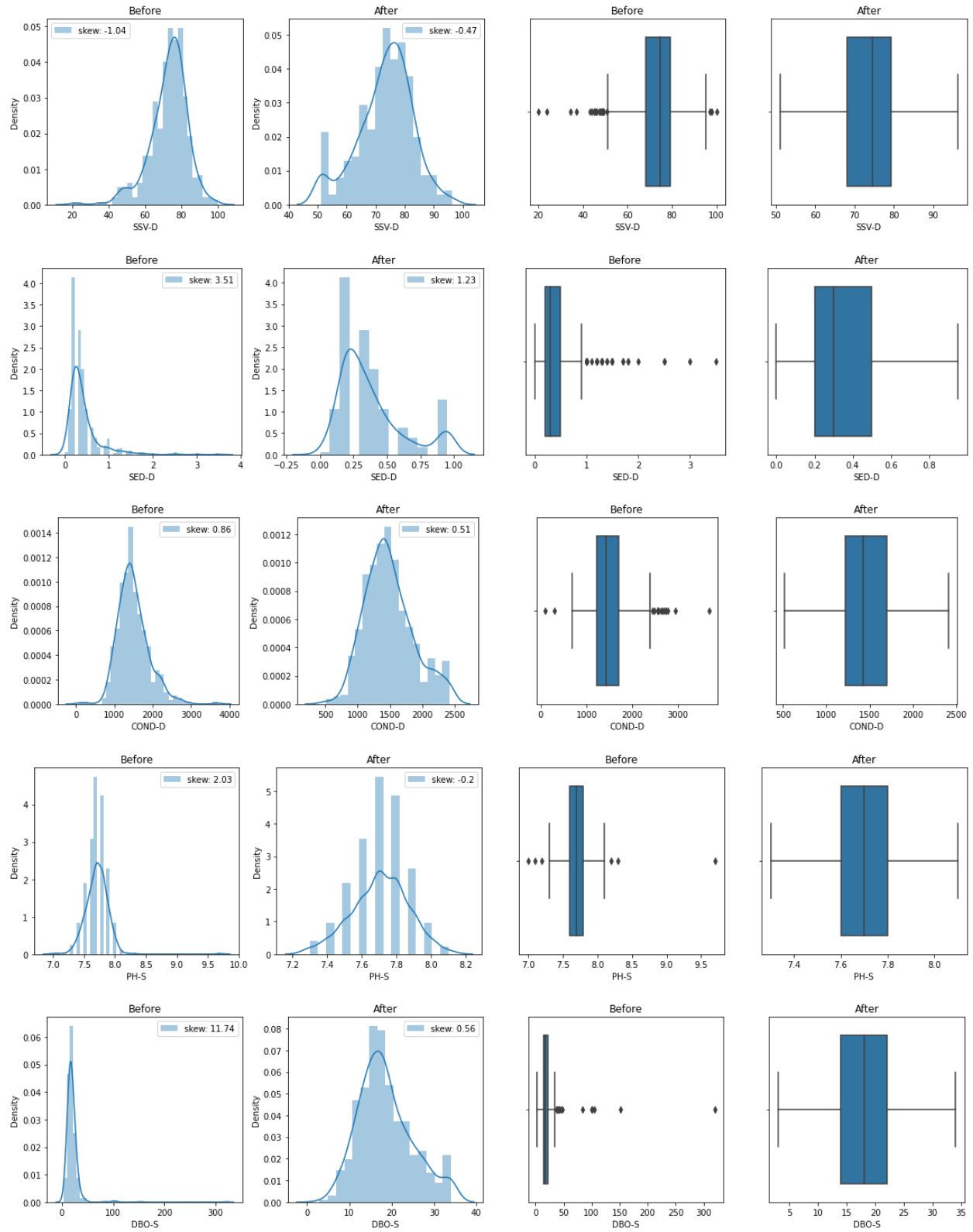


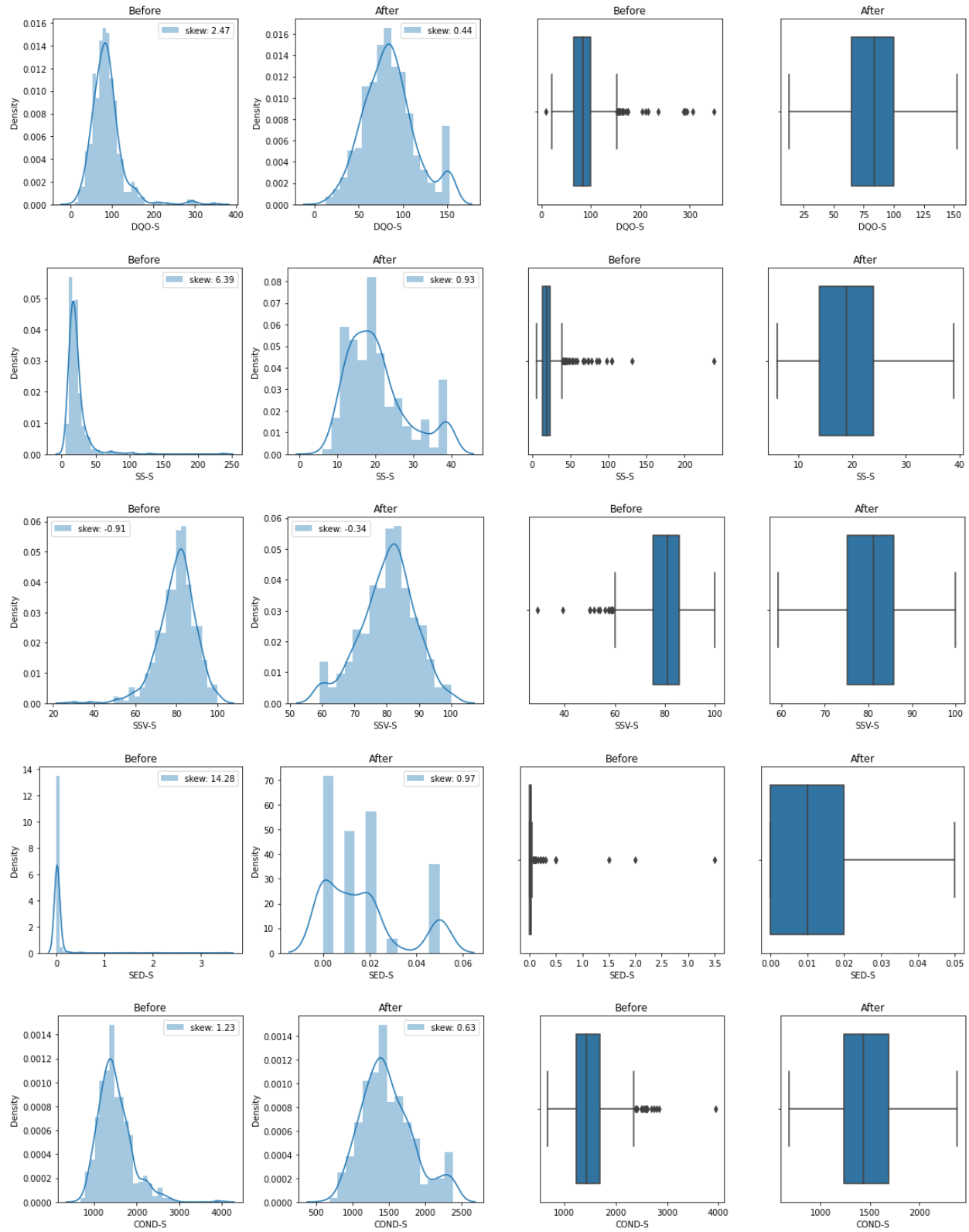


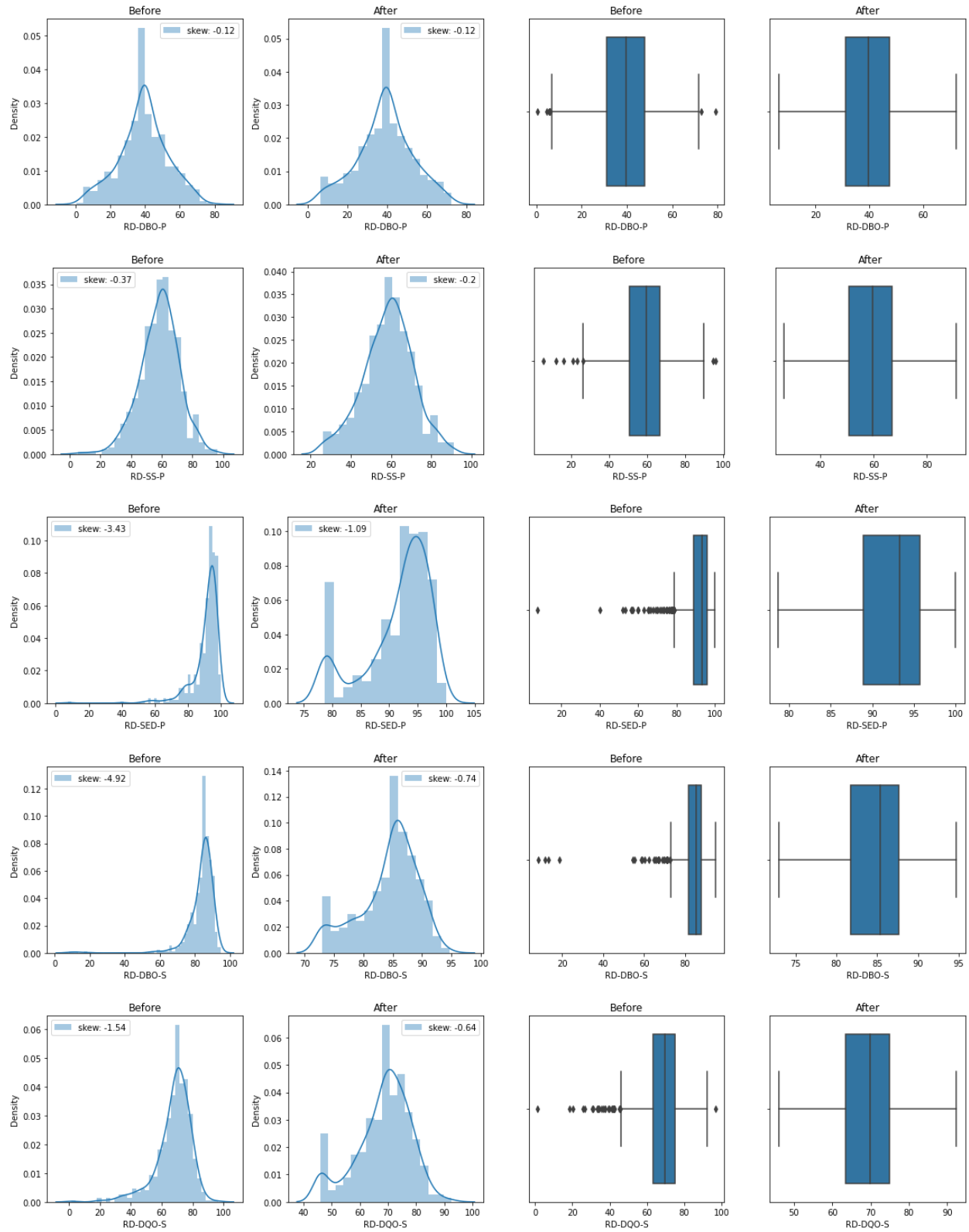


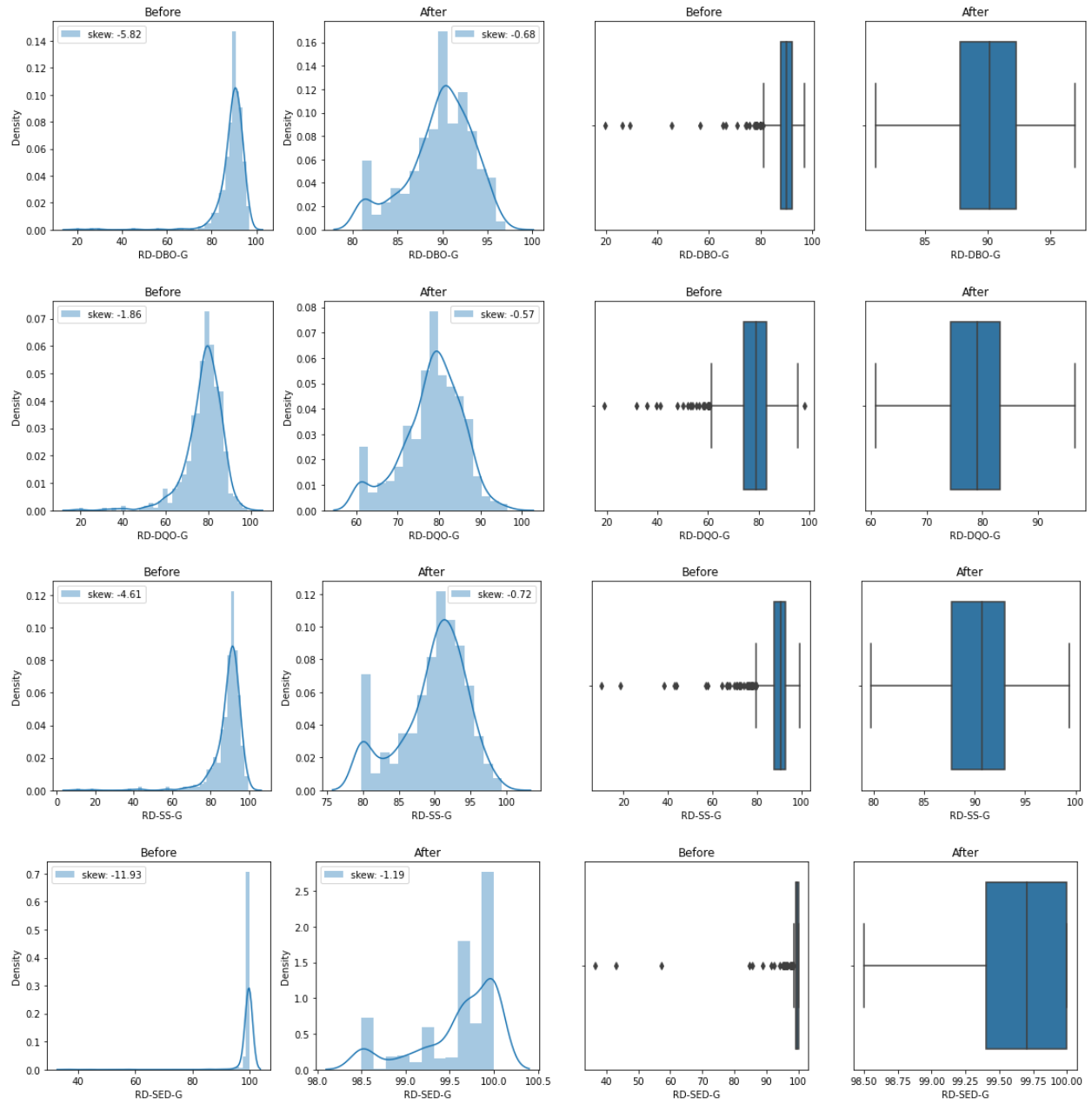












- *Data scaling*

Dilakukan feature scaling data pada dataset memiliki rentang nilai (scale) yang sama.

### DATA SCALING

```
[ ] scaler = StandardScaler()
    scaler.fit(df_cap)
    df_scaled = scaler.transform(df_cap)
```

```
x = df_scaled
```

- Reduksi dimensi

Dilakukan reduksi dimensi untuk membuat dimensi data menjadi 2 menggunakan PCA.

### DIMENSION REDUCTION USING PCA

```
pca = PCA(n_components=2)
fit_pca = pca.fit_transform(x)
pca_df = pd.DataFrame(data = fit_pca, columns = ['PCA_1', 'PCA_2'])
pca_df.head()
```

	PCA_1	PCA_2
0	1.611241	-0.843754
1	4.089042	0.074448
2	0.658219	-1.838839
3	4.181095	0.031123
4	2.378687	-0.253165

#### IV. Elbow Method

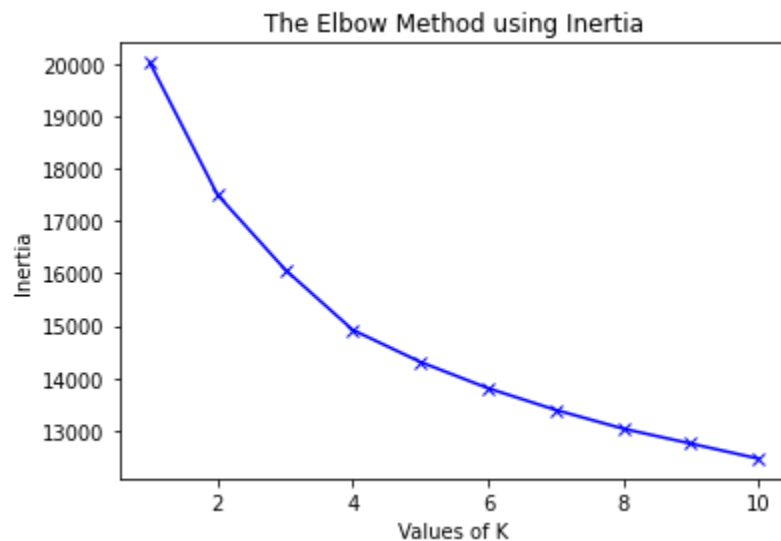
Elbow method inersia digunakan untuk mencari nilai k terbaik dengan melakukan iterasi di setiap k nya.

##### ELBOW METHOD

```
▶ K = range(1,11)
inertias = []

for i in (K):
    #kmeanModel = KMeans(n_clusters=i, init = "k-means++", random_state = 42)
    kmeanModel = KMeans(n_clusters=i).fit(x)
    kmeanModel.fit(x)
    inertias.append(kmeanModel.inertia_)

plt.plot(K, inertias, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Inertia')
plt.title('The Elbow Method using Inertia')
plt.show()
```



Dapat diambil kesimpulan bahwa, k terbaik adalah 4 dilihat dari *elbow* yang terdapat pada plot di atas.

## V. Algoritma/metode yang diterapkan

Metode *unsupervised learning* yang digunakan adalah KMeans. KMeans clustering merupakan salah satu algoritma populer yang digunakan untuk memecahkan masalah pengelompokan data. K pada K-means clustering menandakan jumlah kluster yang digunakan. Pengelompokan ini berdasarkan variabel tertentu tanpa perlu melakukan proses training karena KMeans berbasis centroid. Algoritma ini berfungsi untuk meminimalkan jumlah jarak antara titik data dan cluster yang sesuai yang dihitung dengan euclidean distance. Algoritma KMeans akan mengambil data sebagai input, menentukan centroid awal dengan acak atau ditentukan sendiri, melakukan pengelompokan/menentukan *cluster*, centroid berubah, dan mengulangi proses tersebut sampai centroids tidak lagi berubah atau iterasi sudah selesai.

```
from IPython.display import clear_output
np.random.seed(42)

def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2))

class KMeans:
    def __init__(self, K=5, max_iters=100, plot_steps=False):
        self.K = K
        self.max_iters = max_iters
        self.plot_steps = plot_steps

        # list of sample indices for each cluster
        self.clusters = [[] for _ in range(self.K)]
        # the centers (mean feature vector) for each cluster
        self.centroids = []
```



```

def predict(self, X):
    self.X = X
    self.n_samples, self.n_features = X.shape

    # initialize
    random_sample_idxs = np.random.choice(self.n_samples, self.K, replace=False)
    self.centroids = [self.X[idx] for idx in random_sample_idxs]

    # Optimize clusters
    for _ in range(self.max_iters):
        # Assign samples to closest centroids (create clusters)
        self.clusters = self._create_clusters(self.centroids)

        if self.plot_steps:
            self.plot()

        # Calculate new centroids from the clusters
        centroids_old = self.centroids
        self.centroids = self._get_centroids(self.clusters)

        # check if clusters have changed
        if self._is_converged(centroids_old, self.centroids):
            break

        if self.plot_steps:
            self.plot()

    # Classify samples as the index of their clusters
    return self._get_cluster_labels(self.clusters)

```

```

def _get_cluster_labels(self, clusters):
    # each sample will get the label of the cluster it was assigned to
    labels = np.empty(self.n_samples)

    for cluster_idx, cluster in enumerate(clusters):
        for sample_index in cluster:
            labels[sample_index] = cluster_idx
    return labels

def _create_clusters(self, centroids):
    # Assign the samples to the closest centroids to create clusters
    clusters = [[] for _ in range(self.K)]
    for idx, sample in enumerate(self.X):
        centroid_idx = self._closest_centroid(sample, centroids)
        clusters[centroid_idx].append(idx)
    return clusters

def _closest_centroid(self, sample, centroids):
    # distance of the current sample to each centroid
    distances = [euclidean_distance(sample, point) for point in centroids]
    closest_index = np.argmin(distances)
    return closest_index

def _get_centroids(self, clusters):
    # assign mean value of clusters to centroids
    centroids = np.zeros((self.K, self.n_features))
    for cluster_idx, cluster in enumerate(clusters):
        cluster_mean = np.mean(self.X[cluster], axis=0)
        centroids[cluster_idx] = cluster_mean
    return centroids

def _is_converged(self, centroids_old, centroids):
    # distances between each old and new centroids, for all centroids
    distances = [euclidean_distance(centroids_old[i], centroids[i]) for i in range(self.K)]
    return sum(distances) == 0

```

```

def plot(self):
    fig, ax = plt.subplots(figsize=(12, 8))

    for i, index in enumerate(self.clusters):
        point = self.X[index].T
        ax.scatter(*point)
        clear_output(wait = True)

    for point in self.centroids:
        ax.scatter(*point, marker="x", color="black", linewidth=2)
        clear_output(wait = True)

    plt.show()

```

```

if __name__ == "__main__":
    arr = pca_df.to_numpy()

    k = KMeans(K=4, max_iters=150, plot_steps=True)
    y_pred = k.predict(arr)

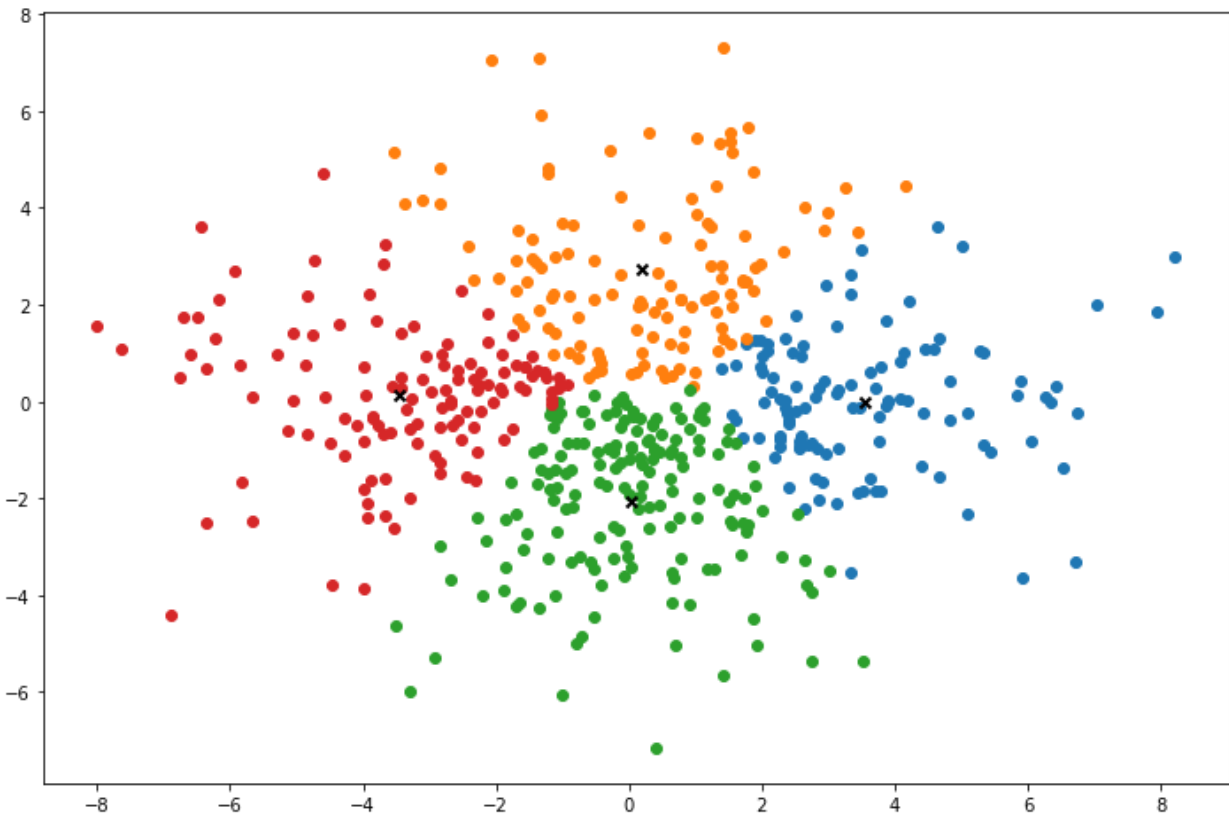
    k.plot()

```

## VI. Evaluasi hasil

	PCA_1	PCA_2	cluster
0	1.611241	-0.843754	0
1	4.089043	0.074449	3
2	0.658219	-1.838839	0
3	4.181095	0.031123	3
4	2.378687	-0.253165	3
...	...	...	...
522	-5.139850	-0.602024	2
523	-2.449418	-0.182213	2
524	-3.292510	-1.986531	2
525	-1.569625	0.227945	2
526	-3.189868	-0.848696	2

527 rows x 3 columns



Dapat dilihat dari visualisasi pengelompokan data, data berhasil dikelompokkan dengan baik.

▼ DBI SCORE FOR CLUSTERING EVALUATION

```
from sklearn.metrics import davies_bouldin_score
db_index = davies_bouldin_score(pca_df, y_pred)
print(db_index)
```

0.8712478826810504

Skor tersebut didefinisikan sebagai rata-rata ukuran kesamaan dari setiap kluster dengan kluster yang paling mirip, di mana kesamaan adalah rasio jarak dalam kluster terhadap jarak antar kluster. Dengan demikian, cluster yang jaraknya lebih jauh dan kurang tersebar akan menghasilkan skor yang lebih baik.

Skor minimum adalah nol, dengan nilai yang lebih rendah menunjukkan pengelompokan yang lebih baik. Jadi, dapat disimpulkan bahwa pengelompokan yang dilakukan sudah cukup baik jika dilihat dari DBI *score*.

## VII. Link

Presentation slide link :

[https://www.canva.com/design/DAFTwjX3Pdo/Ek5B-RfsavBy0Ju4kKPCTQ/view?utm\\_content=DAFTwjX3Pdo&utm\\_campaign=designshare&utm\\_medium=link2&utm\\_source=sharebutton](https://www.canva.com/design/DAFTwjX3Pdo/Ek5B-RfsavBy0Ju4kKPCTQ/view?utm_content=DAFTwjX3Pdo&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton)

Presentation video, docs, and colab link :

[CASE BASED 2 - VANIA AMADEA \(1301204365\)](#)

## VIII. Reference Link

- ▶ (Code) Capping outliers using the IQR method | Machine Learning
- ▶ K-Means Clustering in Python - Machine Learning From Scratch 12 - Python Tuto...
- ▶ Outlier detection and removal using IQR | Feature engineering tutorial python # 4
- ▶ algoritma KMeans Clustering [Unsupervised Learning Python pemula] - #INDOS...
- ▶ Machine Learning Tutorial Python - 13: K Means Clustering Algorithm
- ▶ Preprocessing Data|Preparation Data|Cleaning Data|Hapus Data Outlier|Menseimb...

[K-Means Clustering: Pengertian, Cara Kerja, Kelebihan, dan Kekurangannya - Trivusi ML | Introduction to Kernel PCA - GeeksforGeeks](#)

[2 Teknik Reduksi Dimensi Populer dengan Python - IlmudataPy](#)

[Pandas DataFrame to NumPy Array - Python Examples](#)

[StandardScaler, MinMaxScaler and RobustScaler techniques - ML - GeeksforGeeks.](#)

[K-Means Clustering From Scratch in Python \[Algorithm Explained\] - AskPython](#)

[Pandas DataFrame – Rename Label Index dan Columns – SkillPlus](#)

[K-means for 3 variables](#)

[Tutorial K-Means Clustering dengan Python](#)

[Davies-Bouldin Index for K-Means Clustering Evaluation in Python](#)

[sklearn.metrics.davies\\_bouldin\\_score — scikit-learn 1.1.3 documentation](#)

[2.3. Clustering — scikit-learn 1.1.3 documentation](#)

[Peningkatan Hasil Evaluasi Clustering Davies-Bouldin Index dengan Penentuan Titik Pusat Cluster Awal Algoritma K-Means](#)