Laporan Tugas Mata Kuliah Machine Learning

Data Preparation dari sumber Open Source

disusun untuk memenuhi tugas
mata kuliah Machine Learning A

Oleh :

Fazhira Rizky Harmayani	2308107010012
Cut Dahliana	2308107010027
Naufal Aqil	2208107010043
Hidayat Nur Hakim	2208107010063
Riska Haqika Situmorang	2208107010086

JURUSAN INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SYIAH KUALA

2025

A. Deskripsi Dataset

Nama Dataset :PRSA Data - Aotizhongxin

Sumber Dataset : Dataset di ambil dari Kaggle PRSA Data Aotizhongxin (2013-2017). Data ini diperoleh dari stasiun pemantauan kualitas udara di Aotizhongxin. Informasi dikumpulkan secara periodik dengan berbagai variabel yang mencerminkan tingkat pencemaran udara serta kondisi atmosfer.

Deskripsi Singkat :Dataset berisikan data polusi udara dari kota Aotizhongxin, mencakup berbagai parameter kualitas udara serta faktor lingkungan yang dapat mempengaruhi tingkat polusi. Data ini dapat digunakan untuk menganalisis tren polusi, mengidentifikasi korelasi dengan kondisi kesehatan masyarakat, dan mengevaluasi dampak kebijakan lingkungan terhadap kualitas udara.

Jumlah Data : Terdiri dari 35.064 baris dan 18 kolom

Fitur dalam Dataset : Dataset ini memiliki 18 kolom yang terdiri dari identifikasi waktu, yaitu tahun (**year**), bulan (**month**), hari (**day**), dan jam (**hour**) pencatatan. Parameter kualitas udara yang diukur meliputi **PM2.5** (partikel udara ≤2.5µm), **PM10** (partikel udara ≤10µm), **SO2** (sulfur dioksida), **NO2** (nitrogen dioksida), **CO** (karbon monoksida), dan **O3** (ozon). Faktor lingkungan yang dicatat meliputi suhu udara dalam derajat Celsius (**TEMP**), tekanan udara dalam hPa (**PRES**), titik embun dalam derajat Celsius (**DEWP**), curah hujan dalam mm (**RAIN**), kecepatan angin dalam m/s (**WSPM**), serta arah angin (**wd**) dengan nilai seperti NE, ENE, SW, dan lainnya. informasi tambahan mencakup nomor urut data dalam dataset (**No**) dan nama stasiun pemantauan (**station**) yang dalam hal ini adalah Aotizhongxin.

Format Data : CSV

B. Data Loading

- Library yang digunakan

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
```

Kode ini menggunakan beberapa library penting untuk analisis dan pemrosesan data. Pandas digunakan untuk manipulasi data, sementara NumPy membantu dalam perhitungan numerik seperti kuartil untuk deteksi outlier. Matplotlib dan Seaborn digunakan untuk visualisasi data, seperti membuat boxplot untuk mendeteksi outlier. Terakhir, MinMaxScaler dari Scikit-learn digunakan untuk normalisasi data agar berada dalam rentang 0 hingga 1.

- Memuat Dataset

```
[2]  # Memuat dataset
     df = pd.read_csv("PRSA_Data_Aotizhongxin.csv")
```

Fungsi read_csv() dari pustaka Pandas digunakan untuk membaca file CSV. Fungsi ini mengkonversi data dari file CSV tersebut menjadi sebuah DataFrame Pandas, yang merupakan struktur data tabular yang memudahkan manipulasi dan analisis data.

- Ukuran datanya tidak terlalu besar sehingga tidak menjadi tantangan dalam memuat datasetnya

C. Data Understanding

- Menampilkan sejumlah baris pertama dari sebuah dataframe

```
[3]  df.head()
```

	No	year	month	day	hour	PM2.5	PM10	SO2	NO2	CO	O3	TEMP	PRES	DEWP	RAIN	wd	WSPM
0	1	2013	3	1	0	4.0	4.0	4.0	7.0	300.0	77.0	-0.7	1023.0	-18.8	0.0	NNW	4.4
1	2	2013	3	1	1	8.0	8.0	4.0	7.0	300.0	77.0	-1.1	1023.2	-18.2	0.0	N	4.7
2	3	2013	3	1	2	7.0	7.0	5.0	10.0	300.0	73.0	-1.1	1023.5	-18.2	0.0	NNW	5.6
3	4	2013	3	1	3	6.0	6.0	11.0	11.0	300.0	72.0	-1.4	1024.5	-19.4	0.0	NW	3.1
4	5	2013	3	1	4	3.0	3.0	12.0	12.0	300.0	72.0	-2.0	1025.2	-19.5	0.0	N	2.0

Secara default, fungsi ini menampilkan 5 baris pertama. Ini berguna untuk mendapatkan gambaran awal tentang struktur data dan memastikan bahwa data telah dimuat dengan benar.

- Menampilkan Dimensi DataFrame

```
[5]  df.shape
     (35064, 18)
```

Dataset terdiri dari 35.064 bariis dan 18 kolom

- Menampilkan statistik untuk kolom numerik secara default.

```
[7]  df.describe()
```

	No	year	month	day	hour	PM2.5	P
count	35064.000000	35064.000000	35064.000000	35064.000000	35064.000000	34139.000000	34346.000
mean	17532.500000	2014.662560	6.522930	15.729637	11.500000	82.773611	110.060
std	10122.249256	1.177213	3.448752	8.800218	6.922285	82.135694	95.223
min	1.000000	2013.000000	1.000000	1.000000	0.000000	3.000000	2.000
25%	8766.750000	2014.000000	4.000000	8.000000	5.750000	22.000000	38.000
50%	17532.500000	2015.000000	7.000000	16.000000	11.500000	58.000000	87.000
75%	26298.250000	2016.000000	10.000000	23.000000	17.250000	114.000000	155.000
max	35064.000000	2017.000000	12.000000	31.000000	23.000000	898.000000	984.000

- Menampilkan tipe data dari setiap kolom dalam DataFrame

```
print(df.dtypes)
```

```
No            int64
year          int64
month         int64
day           int64
hour          int64
PM2.5       float64
PM10        float64
SO2         float64
NO2         float64
CO          float64
O3          float64
TEMP        float64
PRES        float64
DEWP        float64
RAIN        float64
wd           object
WSPM        float64
station      object
dtype: object
```

Kolom bertipe int64 (misalnya year, month, day, hour): Berisi angka bulat. Kolom bertipe float64 (misalnya PM2.5, TEMP, PRES): Berisi angka desimal. dan Kolom bertipe object (misalnya wd, station): Biasanya berisi teks atau kategori. Mengetahui tipe data dalam dataset sangat penting karena menentukan langkah preprocessing yang diperlukan, seperti konversi tipe data atau encoding untuk kolom kategori.

- Memisahkan kolom non-numerik dan numerik

```
[8]  # Simpan kolom non-numerik ke variabel terpisah
     df_categorical = df[['wd', 'station']].copy()
     # Pisahkan hanya kolom numerik untuk analisis korelasi
     df_numeric = df.select_dtypes(include=['number'])
```

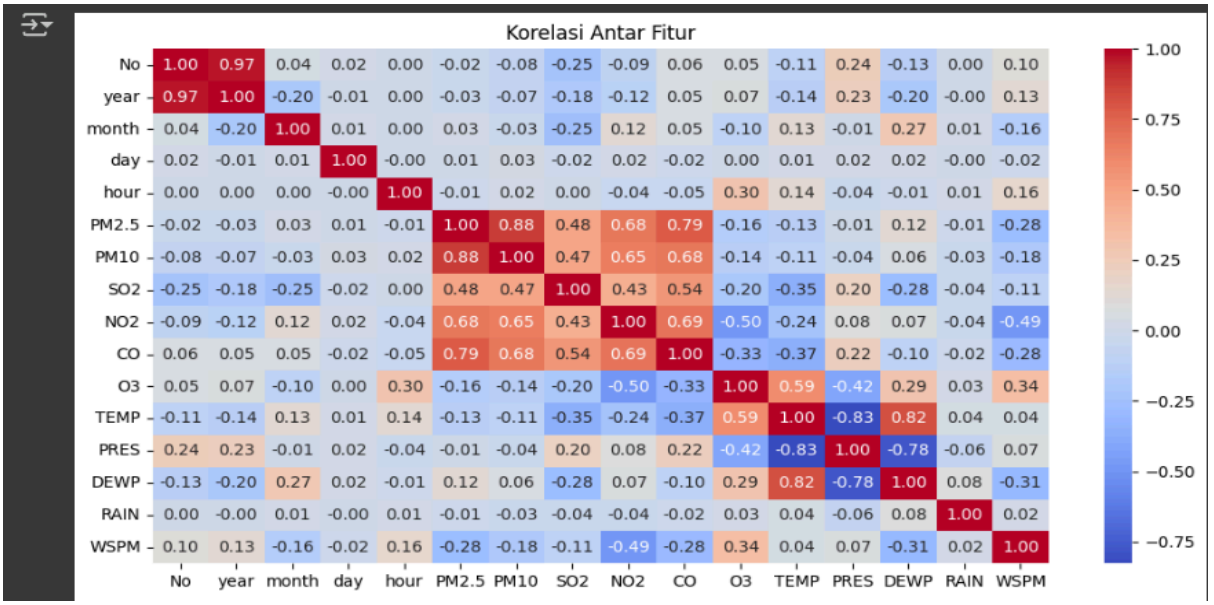
Analisis korelasi hanya bisa dilakukan pada data numerik, sehingga kolom kategori harus dipisah.

- Menghitung Korelasi Antar Fitur Numerik dan Menampilkan Heatmap Korelasi

```
[9]  # Hitung korelasi antar fitur numerik
     correlation_matrix = df_numeric.corr(method='pearson')

     # Tampilkan heatmap korelasi
     plt.figure(figsize=(12, 6))
     sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
     plt.title("Korelasi Antar Fitur")
     plt.show()
```

Menghitung korelasi antara kolom-kolom numerik menggunakan korelasi Pearson. Korelasi Pearson mengukur hubungan linear antara dua variabel, dengan nilai antara -1 hingga 1:



Heatmap menunjukkan bahwa beberapa fitur memiliki korelasi tinggi, seperti **PM2.5 dan PM10 (0.88)** serta **CO dan NO2 (0.69)**, yang menunjukkan bahwa polutan ini sering muncul bersama, kemungkinan dari sumber yang sama seperti emisi kendaraan. Korelasi negatif juga ditemukan, misalnya **PRES dan TEMP (-0.83)**, yang berarti tekanan tinggi cenderung dikaitkan dengan suhu rendah. Sementara itu, beberapa fitur memiliki korelasi lemah, seperti **bulan dan NO2 (-0.12)** serta **day dan PM2.5 (0.01)**, menunjukkan bahwa faktor musiman dan pola harian tidak terlalu berpengaruh terhadap polusi udara.

- Mengecek Missing Value

```
[10] # Mengecek jumlah nilai kosong
     print(df.isnull().sum())

No              0
year            0
month           0
day             0
hour            0
PM2.5         925
PM10          718
SO2           935
NO2          1023
CO           1776
O3           1719
TEMP           20
PRES           20
DEWP           20
RAIN           20
wd             81
WSPM           14
station         0
dtype: int64
```

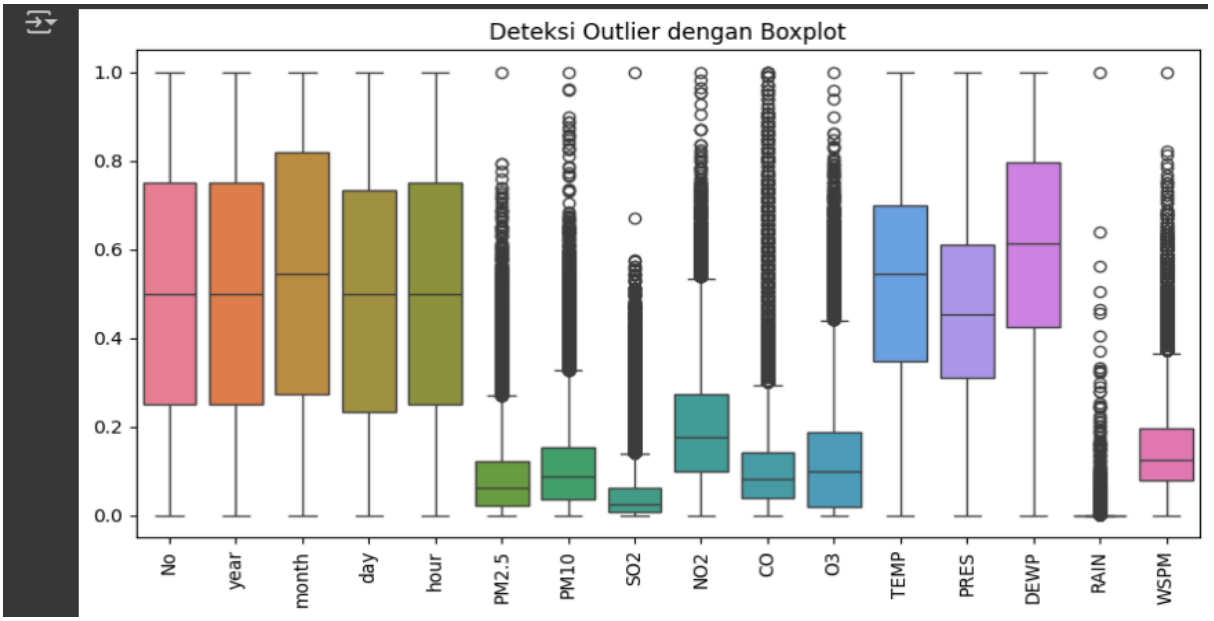
DApat dilihat dari gambar diatas terdapat beberapa missing values terutama pada kolom polutan udara (PM2.5, PM10, SO2, NO2, CO, O3) Beberapa kolom memeiliki jumlah missing values yang realtif kecil seperti WSPM.

- Mengecek Data Duplikat

```
[11] df_numeric.duplicated().sum()
0
```

Dapat dilihat dari gambar bahwa dataset ini tidak terdapat data duplikat

- Mendeteksi Outlier



Dari visualisasi boxplot, terlihat bahwa beberapa fitur seperti **PM2.5, PM10, SO2, NO2, CO, O3, RAIN, dan WSPM** memiliki banyak **outlier**, yang ditandai dengan titik-titik di luar batas (whiskers)

- Menentukan kolom yang akan dicek outliernya

```
# Daftar kolom yang akan dicek outliernya
outlier_cols = ["PM2.5", "PM10", "SO2", "NO2", "CO", "O3", "RAIN", "WSPM"]

# Fungsi untuk menghitung jumlah outlier
def count_outliers_iqr(df_numeric, cols):
    outlier_counts = {}
    for col in cols:
        Q1 = np.quantile(df_numeric[col], 0.25)
        Q3 = np.quantile(df_numeric[col], 0.75)
        IQR = Q3 - Q1
        min_IQR = Q1 - 1.5 * IQR
        max_IQR = Q3 + 1.5 * IQR

        # Hitung jumlah outlier
        outliers = df_numeric[(df_numeric[col] < min_IQR) | (df_numeric[col] > max_IQR)]
        outlier_counts[col] = len(outliers)

    return outlier_counts

# Hitung jumlah outlier pada dataset
outlier_counts = count_outliers_iqr(df_numeric, outlier_cols)
print(outlier_counts)
```

```
{'PM2.5': 1768, 'PM10': 1162, 'SO2': 3054, 'NO2': 566, 'CO': 2607, 'O3': 1491, 'RAIN': 1380, 'WSP
```

Hasil perhitungan outlier dengan metode IQR menunjukkan bahwa **SO2 (3054)** dan **CO (2607)** memiliki jumlah outlier tertinggi, kemungkinan akibat aktivitas industri atau cuaca ekstrem. **PM2.5 (1768)** dan **PM10 (1162)** juga menunjukkan banyak nilai ekstrem, mencerminkan perubahan drastis kualitas udara. **O3 (1491)** dan **RAIN (1380)** mengalami fluktuasi tinggi, sedangkan **WSPM (1742)** menunjukkan variasi signifikan pada kecepatan angin. **NO2 (566)** memiliki jumlah outlier lebih sedikit, tetapi tetap menunjukkan variabilitas data.

- Menghitung persentase outlier untuk setiap fitur dalam dataset.

```
total_data = len(df_numeric)
outlier_counts = {'PM2.5': 1768, 'PM10': 1162, 'SO2': 3167, 'NO2': 566, 'CO': 2607, 'O3': 1491,

for col, count in outlier_counts.items():
    percentage = (count / total_data) * 100
    print(f"{col}: {count} outliers ({percentage:.2f}%)")
```

```
PM2.5: 1768 outliers (5.04%)
PM10: 1162 outliers (3.31%)
SO2: 3167 outliers (9.03%)
NO2: 566 outliers (1.61%)
CO: 2607 outliers (7.43%)
O3: 1491 outliers (4.25%)
RAIN: 1380 outliers (3.94%)
WSPM: 1742 outliers (4.97%)
```

Kategori Outlier Berdasarkan Persentase Sedikit (<5%) → Bisa dihapus langsung:

- PM10 (3.31%), NO2 (1.61%), O3 (4.25%), RAIN (3.94%), WSPM (4.97%)

Sedang (5-10%) → Perlu dipertimbangkan (hapus atau imputasi):

- PM2.5 (5.04%), SO2 (9.03%), CO (7.43%)

D. Data Preparation

- Normalisasi data

```
[28] # Inisialisasi scaler untuk masing-masing dataset
     scaler = MinMaxScaler()

     # Daftar kolom numerik yang akan dinormalisasi
     num_cols = df_numeric.select_dtypes(include=['float64', 'int64']).columns.tolist()

     # Normalisasi dataset train (fit dan transform di train)
     df_numeric[num_cols] = scaler.fit_transform(df_numeric[num_cols])
```

Normalisasi diperlukan karena dataset memiliki variabel dengan skala berbeda (misalnya, konsentrasi polutan dalam µg/m³, suhu dalam °C, dan tekanan dalam hPa), yang dapat menyebabkan algoritma machine learning lebih condong ke fitur dengan nilai besar.

- Menangani Missing Value

```
# Kolom dengan missing values kecil (<100) diimputasi (mengisi)
# Mengisi kolom numerik dengan median tanpa inplace
df_numeric['WSPM'] = df_numeric['WSPM'].fillna(df_numeric['WSPM'].median())


# Disarankan untuk diimputasi dengan median atau interpolasi, karena data polutan udara biasanya
# Mengisi kolom dengan missing values sedang menggunakan median
cols_with_median = ['PM2.5', 'PM10', 'SO2', 'NO2', 'CO', 'O3', 'TEMP', 'PRES', 'DEWP', 'RAIN']
df_numeric[cols_with_median] = df_numeric[cols_with_median].apply(lambda x: x.fillna(x.median()))

# Mengecek kembali jumlah missing values setelah imputasi
print("Missing values setelah penanganan:")
print(df_numeric.isnull().sum())
```

```
Missing values setelah penanganan:
No       0
year     0
month    0
day      0
hour     0
PM2.5    0
PM10     0
SO2      0
NO2      0
CO       0
O3       0
TEMP     0
PRES     0
DEWP     0
RAIN     0
WSPM     0
dtype: int64
```

Penjelasan Bagaimana Menangani Missing Value

1) Kategori Missing Values:
   - Jumlah kecil (< 100) → WSPM (14)
   - Jumlah sedang (100 - 2000) → PM2.5 (925), PM10 (718), SO2 (935), NO2 (1023), O3 (1719)
   - Jumlah besar (> 2000) → CO (1776)

   Jika missing values kecil (<1% dari total data), imputasi lebih disarankan. Jika banyak (>5-10%), perlu dipertimbangkan apakah dihapus atau diimputasi.
2) Metode Pengisian Missing Values
   - Mode untuk Data Kategorikal : Kolom wd (arah angin) → diisi dengan mode (nilai paling sering muncul) agar tetap konsisten.
   - Median untuk Data Numerik : Kolom WSPM (kecepatan angin) → diisi dengan median, lebih tahan terhadap outlier dan Kolom PM2.5, PM10, SO2, dll. → diisi dengan median agar lebih stabil.
3) Kenapa Kolom CO Tidak Dihapus?
   - Kehilangan Data Jika Dihapus

Missing values CO = 17,76% (1.776 dari 35.064 baris). Jika dihapus, informasi dari kolom lain juga akan hilang. Solusi: Lebih baik diimputasi daripada dihapus.

- Kenapa Menggunakan Median?

  Alternatif imputasi: Mean (Rata-rata) → Tidak cocok karena rentan terhadap outlier. Median (Nilai Tengah) → Lebih robust dan stabil. Kesimpulan: Median lebih cocok untuk CO dibandingkan mean.

- Kapan Kolom Sebaiknya Dihapus?

  Jika missing values > 40-50%, kolom lebih baik dihapus. Jika hanya 17,76%, masih layak diimputasi.

- Menangani Outlier

```python
def cap_outliers(df_numeric, columns):
    for column in columns:
        Q1 = df_numeric[column].quantile(0.25)
        Q3 = df_numeric[column].quantile(0.75)
        IQR = Q3 - Q1

        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Ganti nilai outlier dengan batas bawah dan atas
        df_numeric[column] = df_numeric[column].clip(lower=lower_bound, upper=upper_bound)

    return df_numeric

# Kolom numerik yang mengandung outlier
feature_num = ['PM2.5', 'PM10', 'SO2', 'NO2', 'CO', 'O3', 'RAIN', 'WSPM']

# Menangani outlier dengan metode capping
df_numeric = cap_outliers(df_numeric, feature_num)
```

Penjelasan

Capping (pemotongan outlier) menggunakan metode Interquartile Range (IQR).

1) Menghitung Q1 dan Q3
   - Q1 (kuartil pertama): Nilai persentil ke-25 dari data.
   - Q3 (kuartil ketiga): Nilai persentil ke-75 dari data.
   - IQR (Interquartile Range): Rentang antara Q1 dan Q3.
2) Menentukan batas bawah dan atas menggunakan rumus:

   Lower Bound=Q1−1.5×IQR

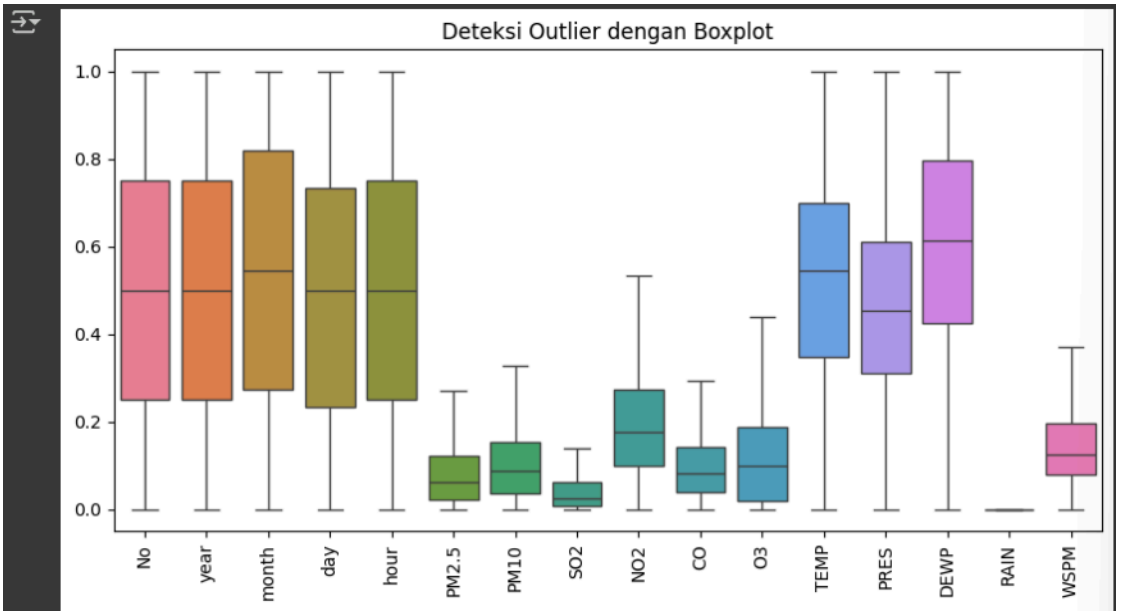
   Upper Bound=Q3+1.5×IQR
3) Mengganti nilai yang berada di luar batas tersebut dengan batas bawah atau batas atas menggunakan fungsi clip():

```python
# Ganti nilai outlier dengan batas bawah dan atas
df_numeric[column] = df_numeric[column].clip(lower=lower_bound, upper=upper_bound)
```

kode ini mengatasi outlier dengan menggantinya ke nilai yang masih dalam batas wajar. Dengan metode ini, distribusi data tetap lebih stabil dibandingkan dengan:

Menghapus data yang memiliki outlier (yang bisa menyebabkan kehilangan informasi). dan Melakukan imputasi dengan nilai rata-rata atau median.

- Menampilakan boxplot outlier nya telah ditangani



dapat dilihat bahwa semua nilai yang sebelumnya dianggap sebagai outlier telah berhasil dikonversi ke dalam rentang yang wajar berdasarkan batas bawah (Q1 - 1.5 × IQR) dan batas atas (Q3 + 1.5 × IQR). Dengan kata lain, tidak ada lagi data yang berada di luar batas yang ditentukan.

E. Kesimpulan

Kesimpulan dari tugas ini adalah memastikan bahwa data yang digunakan siap untuk tahap pemodelan dengan melalui serangkaian proses eksplorasi dan pembersihan. Dimulai dari understanding data, yaitu memahami struktur dataset, distribusi data, dan pola nilai yang ada untuk mendapatkan wawasan awal mengenai karakteristik dataset. Setelah itu, dilakukan preprocessing yang mencakup penanganan missing value agar data tidak memiliki kekosongan yang dapat mempengaruhi analisis, pemeriksaan dan penghapusan data duplikat untuk memastikan tidak ada redundansi, serta normalisasi data jika diperlukan agar skala variabel lebih seragam. Selain itu, dilakukan deteksi dan penanganan outlier untuk mengurangi pengaruh nilai ekstrem yang dapat menyebabkan bias dalam analisis dan model. Setelah semua tahap ini selesai, data siap digunakan untuk tahap berikutnya, yaitu modeling, di mana data yang telah bersih dan terstruktur dapat digunakan untuk membangun model dengan performa yang optimal.