

# **Analisis Data Kualitas Air dan Penyakit**

## **Menular untuk Menentukan Faktor Pencemar Utama**

**Mata Kuliah: 4143105 - Data Analyst**

**Dosen Pengampu: Oppir Hutapea, S.Tr.Kom., M.Kom**

**TEKNOLOGI REKAYASA PERANGKAT LUNAK**

# Kelompok 01:

**11423001 - Samuel Leonardo Nainggolan**

**11423003 - Samuel Rizki Sinambela**

**11423061 - Gishella Putri Vehuliza Br Tarigan**

# Latar Belakang

Kualitas air penting untuk kesehatan masyarakat, dengan pencemaran air yang disebabkan limbah industri, pertanian, dan rumah tangga dapat meningkatkan zat kimia berbahaya yang berdampak pada lingkungan dan kesehatan, menyebabkan penyakit seperti diare, kolera, hepatitis, dan tifus. Diperlukan analisis untuk memahami hubungan antara pencemaran air dan penyakit. Proyek ini menggunakan dataset Water Pollution and Disease untuk menganalisis hubungan antara pencemaran air dan penyakit dengan data preprocessing, visualisasi data, dan analisis statistik. Hasilnya diharapkan memberikan gambaran dampak pencemaran air terhadap kesehatan dan mendukung kebijakan pengelolaan lingkungan.

# Rumusan Masalah

1.

Bagaimana karakteristik dan sebaran tingkat pencemaran air di berbagai wilayah dalam dataset?

2.

Apakah terdapat hubungan antara kadar zat berbahaya tertentu seperti nitrat atau bahan organik dalam air dengan jumlah kasus penyakit?

3.

Faktor lingkungan apa yang paling berpengaruh terhadap peningkatan kasus penyakit yang ditularkan melalui air?

4.

Bagaimana hasil analisis statistik dan visualisasi data dapat membantu memahami hubungan antara kualitas air dan tingkat kejadian penyakit?

# Tujuan

- 
- Menganalisis data pencemaran air dan penyakit untuk memahami pola, karakteristik, dan hubungan antar variabel lingkungan dengan kejadian penyakit.
  - Meningkatkan kualitas data melalui preprocessing, termasuk penanganan nilai hilang, deteksi outlier, normalisasi, dan encoding variabel kategorikal.
  - Mengidentifikasi hubungan antar variabel menggunakan uji statistik parametrik dan non-parametrik untuk memperoleh hasil analisis yang valid dan signifikan.
  - Menyajikan hasil analisis dalam visualisasi data yang jelas dan informatif untuk mendukung pengambilan keputusan dan memberikan wawasan tentang dampak pencemaran air terhadap kesehatan.

# Metodologi Penelitian

Metode	Deskripsi
Data Collection	Pengumpulan data dari sumber yang kredibel untuk memastikan keakuratan data terhadap topik yang dianalisis.
Data Processing and Techniques (Advance Preprocessing)	Tahapan mencakup pemeriksaan, perbaikan, dan transformasi data mentah agar siap dianalisis.
Statistical Analysis	Melakukan uji statistik untuk mengidentifikasi hubungan antar variabel dan menilai signifikansi ilmiah
Data Visualization	Tahapan penyajian data dalam bentuk grafik atau diagram.

# Data Collection

## 01 Informasi Dataset

```
print("== INFORMASI DATASET ==")
print(f"Jumlah baris dan kolom: {uts.shape}")
print("\nLima data teratas:")
print(uts.head())
|
# Menampilkan jumlah baris dan kolom
print("Jumlah baris dan kolom:", uts.shape)
```

== INFORMASI DATASET ==  
Jumlah baris dan kolom: (3000, 24)

## 02 Informasi detail tiap kolom

```
[5]: print("== INFORMASI DETAIL TIAP KOLOM ==")
print(uts.info())
```

== INFORMASI DETAIL TIAP KOLOM ==  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 3000 entries, 0 to 2999  
Data columns (total 24 columns):

#	Column	Non-Null Count	Dtype
0	Country	3000 non-null	object
1	Region	3000 non-null	object
2	Year	3000 non-null	int64
3	Water Source Type	3000 non-null	object
4	Contaminant Level (ppm)	3000 non-null	float64
5	pH Level	3000 non-null	float64
6	Turbidity (NTU)	3000 non-null	float64
7	Dissolved Oxygen (mg/L)	3000 non-null	float64
8	Nitrate Level (mg/L)	3000 non-null	float64
9	Lead Concentration (µg/L)	3000 non-null	float64
10	Bacteria Count (CFU/mL)	3000 non-null	int64
11	Water Treatment Method	2253 non-null	object
12	Access to Clean Water (% of Population)	3000 non-null	float64
13	Diarrheal Cases per 100,000 people	3000 non-null	int64
14	Cholera Cases per 100,000 people	3000 non-null	int64
15	Typhoid Cases per 100,000 people	3000 non-null	int64
16	Infant Mortality Rate (per 1,000 live births)	3000 non-null	float64
17	GDP per Capita (USD)	3000 non-null	int64
18	Healthcare Access Index (0-100)	3000 non-null	float64
19	Urbanization Rate (%)	3000 non-null	float64
20	Sanitation Coverage (% of Population)	3000 non-null	float64
21	Rainfall (mm per year)	3000 non-null	int64
22	Temperature (°C)	3000 non-null	float64
23	Population Density (people per km²)	3000 non-null	int64

dtypes: float64(12), int64(8), object(4)  
memory usage: 562.6+ KB  
None

# Data Collection

## 03 Informasi statistik deskriptif

```
[6]: print("---- INFORMASI STATISTIK DESKRIPTIF ----")
print(uts.describe())
```

```
---- INFORMASI STATISTIK DESKRIPTIF ----
   Year Contaminant Level (ppm)    pH Level Turbidity (NTU) \
count  3000.000000  3000.000000  3000.000000  3000.000000
mean   2012.012667   4.954390   7.255847   2.480023
std     7.229287   2.860072   0.720464   1.419984
min    2000.000000   0.000000   6.000000   0.000000
25%   2006.000000   2.560000   6.630000   1.257500
50%   2012.000000   4.950000   7.280000   2.460000
75%   2018.000000   7.400000   7.870000   3.660000
max   2024.000000  10.000000   8.500000   4.990000
```

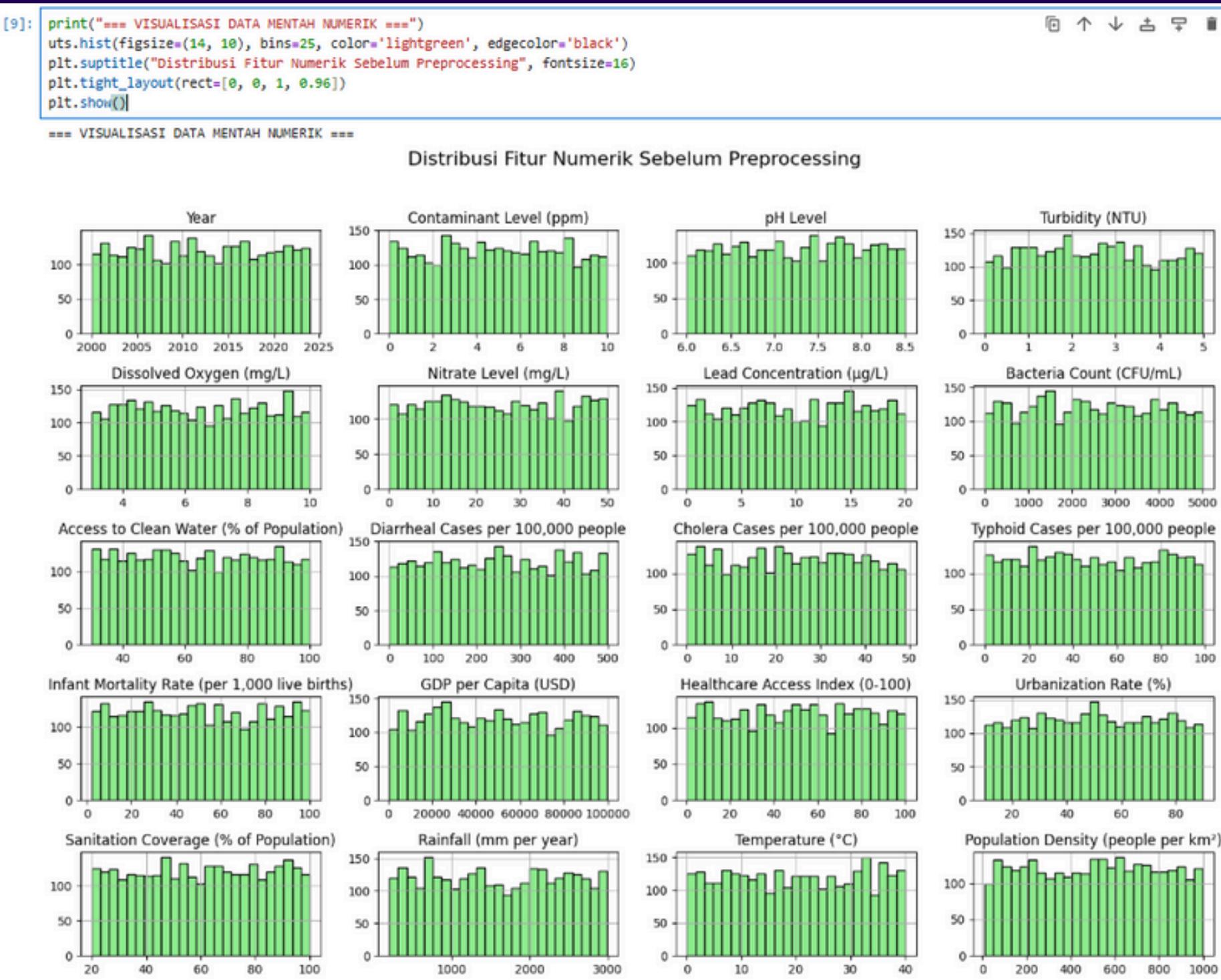
## 04 Identifikasi tipe data

```
[7]: print("---- IDENTIFIKASI TIPE DATA ----")
numeric_cols = uts.select_dtypes(include=['int64', 'float64']).columns
categorical_cols = uts.select_dtypes(include=['object']).columns
|
print("Kolom numerik:", list(numeric_cols))
print("Kolom kategorikal:", list(categorical_cols))
```

```
---- IDENTIFIKASI TIPE DATA ----
Kolom numerik: ['Year', 'Contaminant Level (ppm)', 'pH Level', 'Turbidity (NTU)', 'Dissolved Oxygen (mg/L)', 'Nitrate Level (mg/L)', 'Lead Concentration (µg/L)', 'Bacteria Count (CFU/mL)', 'Access to Clean Water (% of Population)', 'Diarrheal Cases per 100,000 people', 'Cholera Cases per 100,000 people', 'Typhoid Cases per 100,000 people', 'Infant Mortality Rate (per 1,000 live births)', 'GDP per Capita (USD)', 'Healthcare Access Index (0-100)', 'Urbanization Rate (%)', 'Sanitation Coverage (% of Population)', 'Rainfall (mm per year)', 'Temperature (°C)', 'Population Density (people per km²)']
Kolom kategorikal: ['Country', 'Region', 'Water Source Type', 'Water Treatment Method']
```

# Data Collection

## 05 Visualisasi Data Mentah Kategorikal

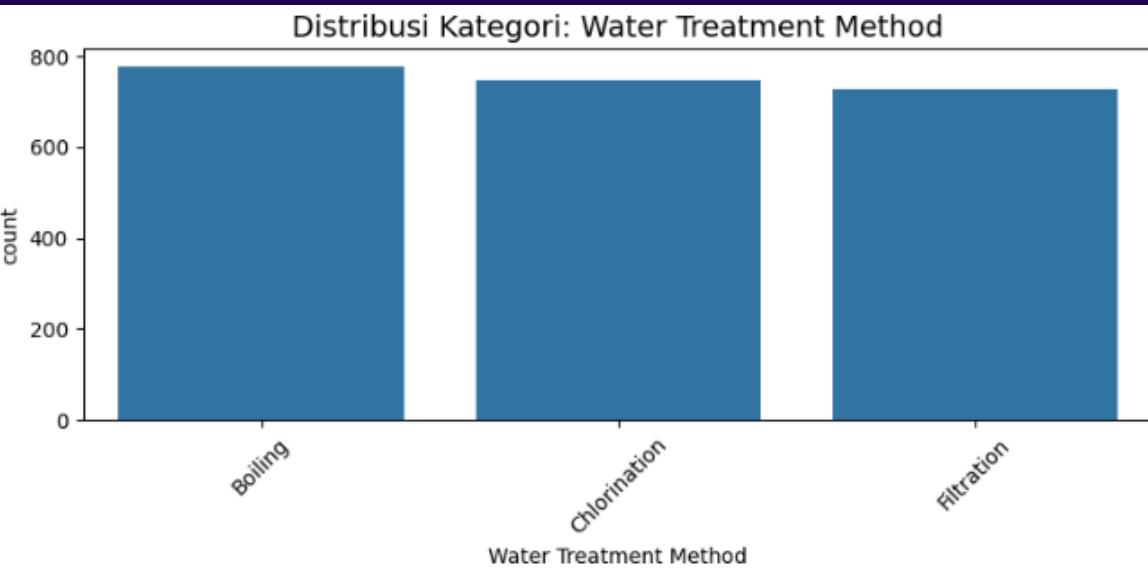
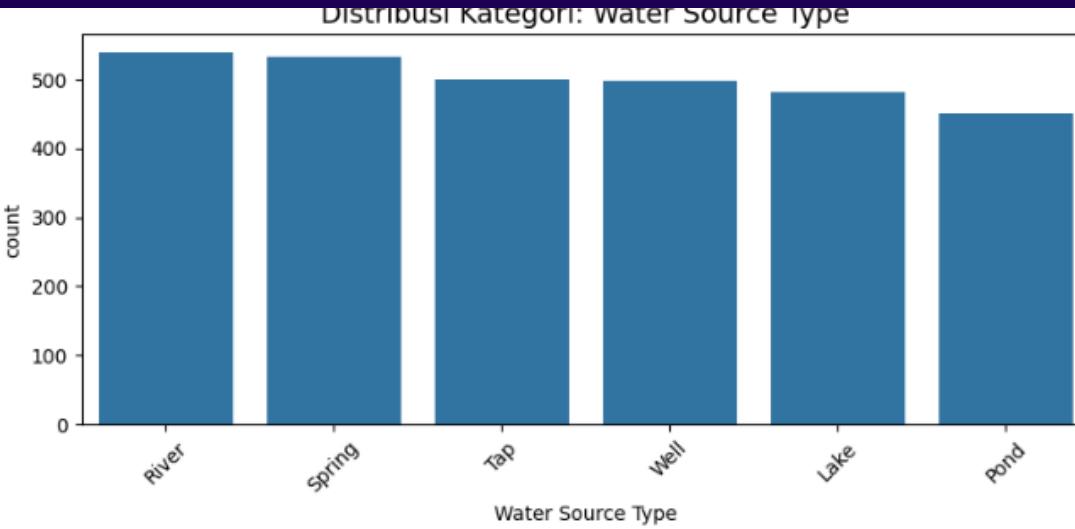
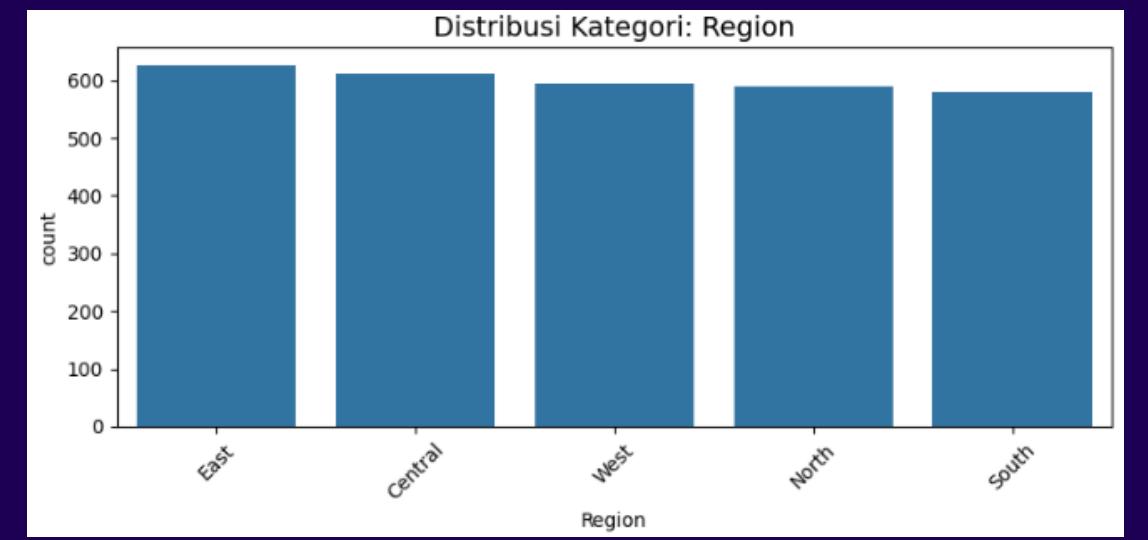
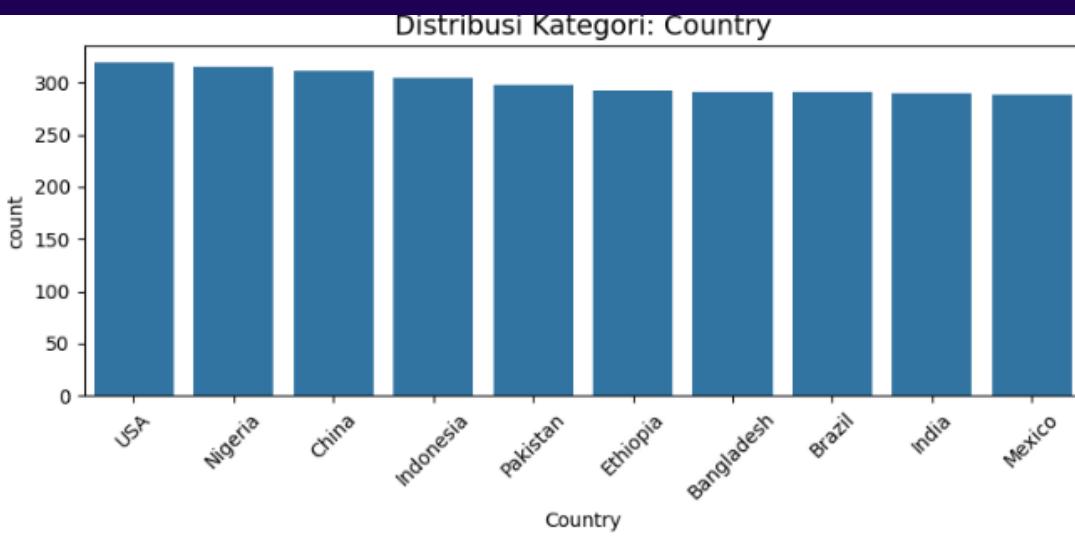


# Data Collection

## 06 Visualisasi Data Kategorikal

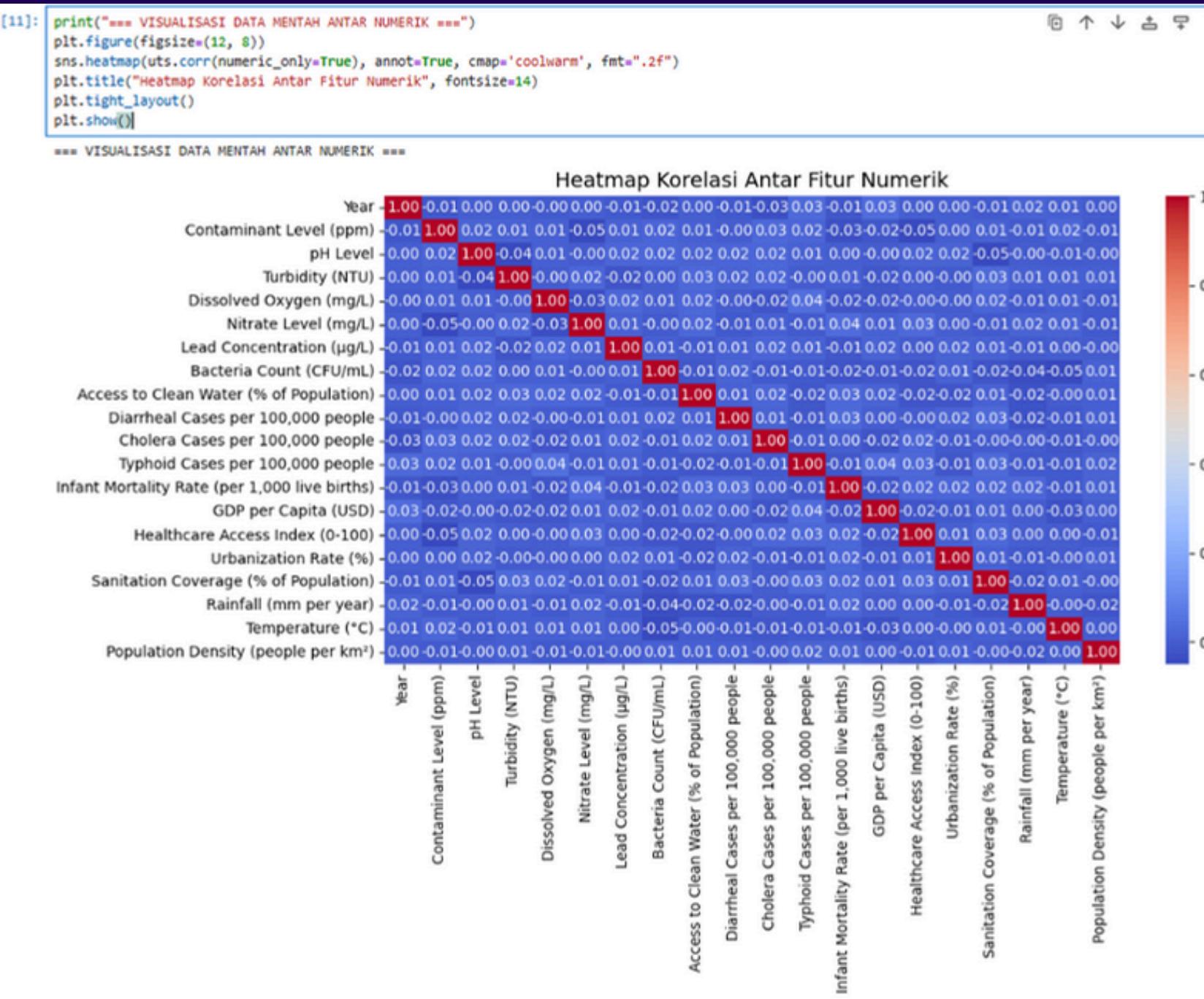
```
[10]: print("== VISUALISASI DATA MENTAH KATEGORIKAL ==")
categorical_cols = uts.select_dtypes(include=['object']).columns

for col in categorical_cols:
    plt.figure(figsize=(8, 4))
    sns.countplot(data=uts, x=col, order=uts[col].value_counts().index)
    plt.title(f"Distribusi Kategori: {col}", fontsize=14)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```



# Data Collection

## 07 Visualisasi Data Antar Numerik



# Data Processing

01

## Handling Missing Values

A. Mean Imputation : efektif untuk data numerik yang terdistribusi normal. Mempertahankan mean populasi.

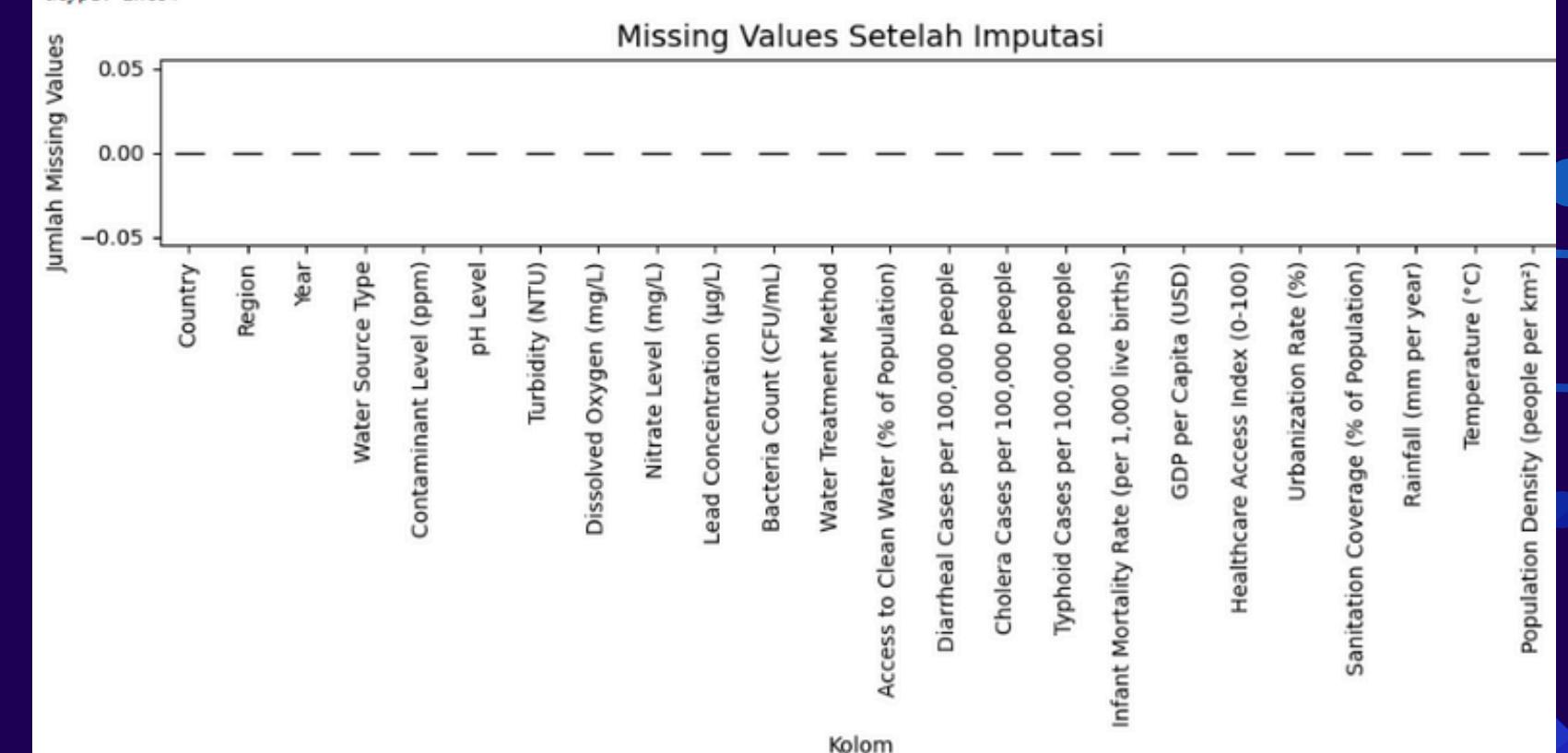
```
# Imputasi Mean (numerik)
for col in num_cols:
    missing_count = uts[col].isnull().sum()
    if missing_count > 0:
        mean_val = uts[col].mean()
        uts[col] = uts[col].fillna(mean_val)
        print(f"Kolom '{col}' → {missing_count} nilai diisi dengan mean ({mean_val:.2f})")
```

B. Mode Imputation: Cocok untuk data kategorikal karena mengganti dengan nilai yang paling representatif.

```
# Imputasi Mode (kategorikal)
for col in cat_cols:
    missing_count = uts[col].isnull().sum()
    if missing_count > 0:
        mode_val = uts[col].mode()[0]
        uts[col] = uts[col].fillna(mode_val)
        print(f"Kolom '{col}' → {missing_count} nilai diisi dengan mode ('{mode_val}')")
```

```
Penanganan missing values
Kolom 'Water Treatment Method' → 747 nilai diisi dengan mode ('Boiling')

Jumlah missing values setelah imputasi:
Country 0
Region 0
Year 0
Water Source Type 0
Contaminant Level (ppm) 0
pH Level 0
Turbidity (NTU) 0
Dissolved Oxygen (mg/L) 0
Nitrate Level (mg/L) 0
Lead Concentration (µg/L) 0
Bacteria Count (CFU/mL) 0
Water Treatment Method 0
Access to Clean Water (% of Population) 0
Diarrheal Cases per 100,000 people 0
Cholera Cases per 100,000 people 0
Typhoid Cases per 100,000 people 0
Infant Mortality Rate (per 1,000 live births) 0
GDP per Capita (USD) 0
Healthcare Access Index (0-100) 0
Urbanization Rate (%) 0
Sanitation Coverage (% of Population) 0
Rainfall (mm per year) 0
Temperature (°C) 0
Population Density (people per km²) 0
dtype: int64
```



# Data Processing

02

## Handling Outliers

Metode yang digunakan adalah Winsorization yaitu teknik yang “memotong” nilai ekstrim (outlier) dengan menggantinya ke batas bawah dan batas atas tertentu berdasarkan persentil data.

- 1.Untuk setiap kolom numerik, ditentukan batas bawah (1%) dan batas atas (99%) berdasarkan nilai kuantil data.
- 2.Data yang berada di bawah batas bawah atau di atas batas atas dianggap outlier.
- 3.Outlier kemudian “dihapus” secara tidak langsung dengan menggantinya ke nilai batas bawah atau batas atas menggunakan fungsi np.clip()

Keuntungan Winsorization:

- 1.Mempertahankan jumlah observasi (tidak ada data yang dihapus)
- 2.Mengurangi pengaruh outlier ekstrim tanpa menghilangkan informasi
- 3.Lebih robust dibanding deletion untuk dataset kecil
- 4.Cocok untuk analisis statistik yang sensitif terhadap outlier

```
# Handling outliers - Winsorization
winsor_outlier_counts = {}
winsor_limits = {}

for col in num_cols:
    # Tentukan batas bawah & atas (1% - 99%)
    lower_lim = uts[col].quantile(0.01)
    upper_lim = uts[col].quantile(0.99)

    # Simpan batas
    winsor_limits[col] = (lower_lim, upper_lim)

    # Deteksi outlier (berdasarkan Winsor Limit)
    outliers = uts[(uts[col] < lower_lim) | (uts[col] > upper_lim)][col]
    winsor_outlier_counts[col] = len(outliers)

    # Terapkan Winsorization
    uts_winsor[col] = np.clip(uts[col], lower_lim, upper_lim)

print("\n Winsorization Handling selesai.\n")

# Ringkasan jumlah outlier
print(" Jumlah Outlier yang Ditemukan (Metode Winsorization):")
for col, count in winsor_outlier_counts.items():
    print(f" - {col}: {count} outlier (batas bawah={winsor_limits[col][0]:.2f}, atas={winsor_limits[col][1]:.2f})")

total_winsor = sum(winsor_outlier_counts.values())
print(f"\n Total outlier terdeteksi dengan Winsorization: {total_winsor}\n")
```

Winsorization Handling selesai.

```
Jumlah Outlier yang Ditemukan (Metode Winsorization):
- Year: 0 outlier (batas bawah=2000.00, atas=2024.00)
- Contaminant Level (ppm): 56 outlier (batas bawah=0.10, atas=9.90)
- pH Level: 57 outlier (batas bawah=6.04, atas=8.47)
- Turbidity (NTU): 47 outlier (batas bawah=0.06, atas=4.94)
- Dissolved Oxygen (mg/L): 55 outlier (batas bawah=3.06, atas=9.93)
- Nitrate Level (mg/L): 60 outlier (batas bawah=0.54, atas=49.64)
- Lead Concentration (µg/L): 56 outlier (batas bawah=0.19, atas=19.78)
- Bacteria Count (CFU/mL): 59 outlier (batas bawah=74.97, atas=4946.00)
- Access to Clean Water (% of Population): 60 outlier (batas bawah=30.70, atas=99.24)
- Diarrheal Cases per 100,000 people: 57 outlier (batas bawah=6.00, atas=495.00)
- Cholera Cases per 100,000 people: 0 outlier (batas bawah=0.00, atas=49.00)
- Typhoid Cases per 100,000 people: 22 outlier (batas bawah=0.00, atas=98.00)
- Infant Mortality Rate (per 1,000 live births): 60 outlier (batas bawah=2.91, atas=99.19)
- GDP per Capita (USD): 60 outlier (batas bawah=1779.20, atas=98933.07)
- Healthcare Access Index (0-100): 59 outlier (batas bawah=0.96, atas=99.24)
- Urbanization Rate (%): 60 outlier (batas bawah=10.90, atas=89.01)
- Sanitation Coverage (% of Population): 60 outlier (batas bawah=20.83, atas=99.00)
- Rainfall (mm per year): 59 outlier (batas bawah=225.00, atas=2971.01)
- Temperature (°C): 60 outlier (batas bawah=0.36, atas=39.50)
- Population Density (people per km²): 58 outlier (batas bawah=20.00, atas=989.01)
```

Total outlier terdeteksi dengan Winsorization: 1005

# Data Processing

03

## Feature Scaling

### 1. Standardization (Z-score)

Mengubah data menjadi mean=0, std=1. Formula:  $(x - \text{mean}) / \text{std}$ . Cocok untuk algoritma berbasis jarak dan distribusi normal. Tujuan utamanya adalah untuk menyelaraskan skala antar fitur numerik, agar setiap kolom memiliki skala yang sebanding dan tidak mendominasi proses analisis atau model machine learning.

```
[17]: print("== FUTURE SCALING (STANDARDIZATION) ==")

print("\n== 1. Standardization (Z-Score Scaling) ==")
scaler = StandardScaler()
uts_standardized = uts.copy()
uts_standardized[num_cols] = scaler.fit_transform(uts[num_cols])

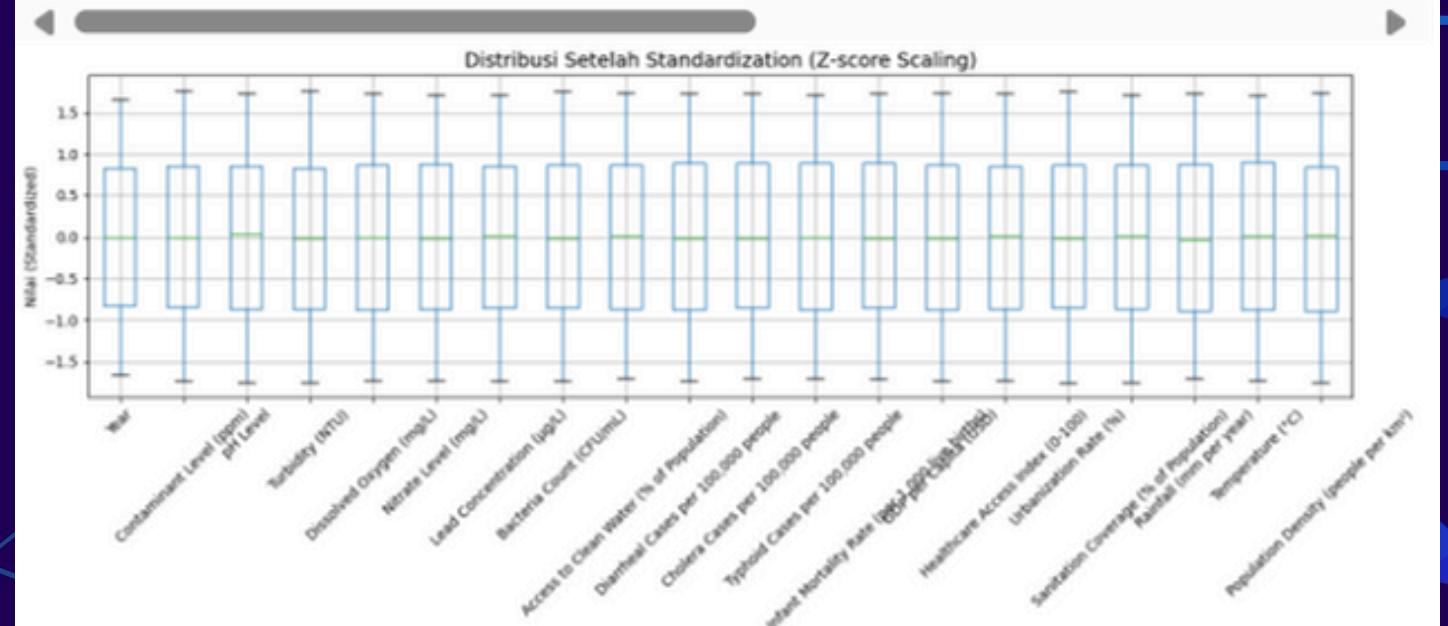
# Statistik Sebelum dan Sesudah
print("\nSebelum Standardization:")
display(uts[num_cols].describe())

print("\nSesudah Standardization:")
display(pd.DataFrame(uts_standardized[num_cols], columns=num_cols).describe())

# Visualisasi Boxplot
plt.figure(figsize=(14,6))
uts_standardized[num_cols].boxplot(rot=45)
plt.title("Distribusi Setelah Standardization (Z-score Scaling)", fontsize=14)
plt.ylabel("Nilai (Standardized)")
plt.tight_layout()
plt.show()
```

== FUTURE SCALING (STANDARDIZATION) ==											
== 1. Standardization (Z-Score Scaling) ==											
Sebelum Standardization:											
Year	Contaminant Level (ppm)	pH Level	Turbidity (NTU)	Dissolved Oxygen (mg/L)	Nitrate Level (mg/L)	Lead Concentration (µg/L)	Bacteria Count (CFU/mL)	Access to Clean Water (% of Population)	Diarrheal Cases per 100,000 people	Cholera Cases per 100,000 people	Typhoid Cases per 100,000 people
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000
mean	2012.012667	4.954390	7.255847	2.480023	6.492850	25.08025	10.047913	2488.477333	64.612333	249.776667	24.25100
std	7.229287	2.860072	0.720464	1.419984	2.027966	14.50517	5.798238	1431.421553	20.308463	144.111543	14.33259
min	2000.000000	0.000000	6.000000	0.000000	3.000000	0.05000	0.000000	0.000000	30.010000	0.000000	0.000000
25%	2006.000000	2.560000	6.630000	1.257500	4.710000	12.52500	5.120000	1268.000000	47.027500	124.000000	12.00000
50%	2012.000000	4.950000	7.280000	2.460000	6.490000	24.79000	10.065000	2469.000000	64.780000	248.000000	24.00000
75%	2018.000000	7.400000	7.870000	3.660000	8.252500	37.91000	15.032500	3736.250000	82.302500	378.000000	37.00000
max	2024.000000	10.000000	8.500000	4.990000	10.000000	49.99000	20.000000	4998.000000	99.990000	499.000000	49.00000

Sesudah Standardization:											
Year	Contaminant Level (ppm)	pH Level	Turbidity (NTU)	Dissolved Oxygen (mg/L)	Nitrate Level (mg/L)	Lead Concentration (µg/L)	Bacteria Count (CFU/mL)	Access to Clean Water (% of Population)	Diarrheal Cases per 100,000 people	Cholera Cases per 100,000 people	Typhoid Cases per 100,000 people
count	3.000000e+03	3.000000e+03	3.000000e+03	3.000000e+03	3.000000e+03	3.000000e+03	3.000000e+03	3.000000e+03	3.000000e+03	3.000000e+03	3.000000e+03
mean	-1.099210e-14	-2.368476e-17	-1.421085e-16	2.155313e-16	1.089499e-16	3.671137e-17	2.480978e-16	4.500104e-17	-2.948752e-16	-2.960595e-17	-8.1
std	1.000167e+00	1.000167e+00	1.000167e+00	1.000167e+00	1.000167e+00	1.000167e+00	1.000167e+00	1.000167e+00	1.000167e+00	1.000167e+00	1.000167e+00
min	-1.661944e+00	-1.732549e+00	-1.743400e+00	-1.746806e+00	-1.722629e+00	-1.725897e+00	-1.733214e+00	-1.738756e+00	-1.704122e+00	-1.733507e+00	-1.65
25%	-8.318481e-01	-8.373178e-01	-8.688170e-01	-8.610852e-01	-8.792786e-01	-8.657150e-01	-8.500401e-01	-8.527752e-01	-8.660313e-01	-8.729185e-01	-8.5
50%	-1.752424e-03	-1.535182e-03	3.353030e-02	-1.410345e-02	-1.405583e-03	-2.001344e-02	2.947363e-03	-1.360925e-02	8.257376e-03	-1.233047e-02	-1.7
75%	8.283433e-01	8.552294e-01	8.525840e-01	8.311174e-01	8.678367e-01	8.846424e-01	8.598160e-01	8.718471e-01	8.712189e-01	8.898989e-01	8.8
max	1.658439e+00	1.764449e+00	1.727167e+00	1.767904e+00	1.729681e+00	1.717588e+00	1.716685e+00	1.753460e+00	1.742306e+00	1.729666e+00	1.72



# Data Processing

03

## Feature Scaling

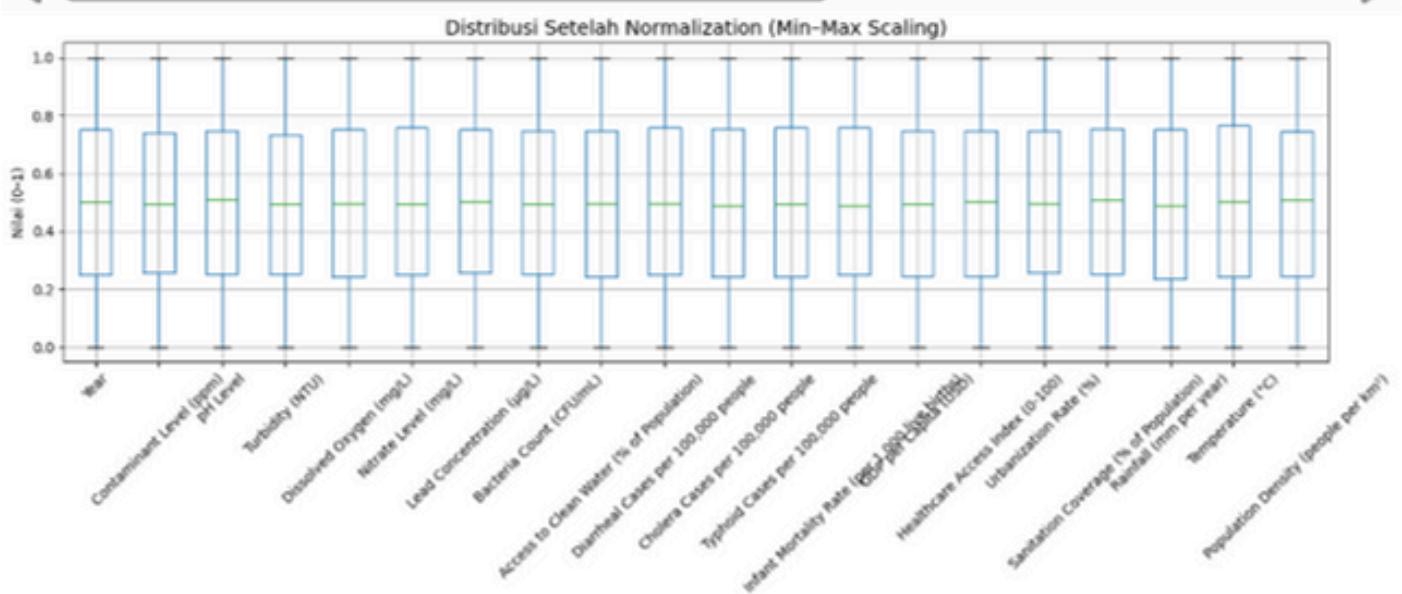
### 2. Normalization (Min-Max)

Mengubah data ke range [0, 1]. Formula:  $(x - \min) / (\max - \min)$ . Cocok untuk neural networks dan algoritma berbasis probabilitas. Dengan demikian, dataset uts\_normalized sudah siap digunakan untuk analisis lebih lanjut atau pemodelan dengan nilai numerik yang telah diseragamkan dalam skala 0 hingga 1

```
[18]: print("\n==== 2. Normalization (Min-Max Scaling) ===")  
  
minmax_scaler = MinMaxScaler()  
uts_normalized = uts.copy()  
uts_normalized[num_cols] = minmax_scaler.fit_transform(uts[num_cols])  
  
# Statistik Sebelum dan Sesudah  
print("\nSebelum Normalization:")  
display(uts[num_cols].describe())  
  
print("\nSesudah Normalization:")  
display(pd.DataFrame(uts_normalized[num_cols], columns=num_cols).describe())  
  
# Visualisasi Boxplot  
plt.figure(figsize=(14,6))  
uts_normalized[num_cols].boxplot(rot=45)  
plt.title("Distribusi Setelah Normalization (Min-Max Scaling)", fontsize=14)  
plt.ylabel("Nilai (0-1)")  
plt.tight_layout()  
plt.show()
```

--- 2. Normalization (Min-Max Scaling) ---												
Sebelum Normalization:												
Year	Contaminant Level (ppm)	pH Level	Turbidity (NTU)	Dissolved Oxygen (mg/L)	Nitrate Level (mg/L)	Lead Concentration (µg/L)	Bacteria Count (CFU/mL)	Access to Clean Water (% of Population)	Diarrheal Cases per 100,000 people	Cholera Cases per 100,000 people	Typhoid Cases per 100,000 people	
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	
mean	2012.012667	4.954390	7.255847	2.480023	6.492850	25.08025	10.047913	2488.477333	64.612333	249.776667	24.25100	49.27000
std	7.229287	2.860072	0.720464	1.419984	2.027906	14.50517	5.790238	1431.421553	20.308463	144.111543	14.33259	28.984165
min	2000.000000	0.000000	6.000000	0.000000	3.000000	0.05000	0.000000	0.000000	30.010000	0.000000	0.000000	0.000000
25%	2006.000000	2.560000	6.630000	1.257500	4.710000	12.52500	5.120000	1268.000000	47.027500	124.000000	12.00000	24.000000
50%	2012.000000	4.950000	7.280000	2.460000	6.480000	24.79000	10.065000	2469.000000	64.780000	248.000000	24.00000	49.000000
75%	2018.000000	7.400000	7.870000	3.660000	8.252500	37.91000	15.032500	3736.250000	82.302500	378.000000	37.00000	75.000000
max	2024.000000	10.000000	8.500000	4.990000	10.000000	49.99000	20.000000	4998.000000	99.990000	499.000000	49.00000	99.000000

Sesudah Normalization:												
Year	Contaminant Level (ppm)	pH Level	Turbidity (NTU)	Dissolved Oxygen (mg/L)	Nitrate Level (mg/L)	Lead Concentration (µg/L)	Bacteria Count (CFU/mL)	Access to Clean Water (% of Population)	Diarrheal Cases per 100,000 people	Cholera Cases per 100,000 people	Typhoid Cases per 100,000 people	
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000
mean	0.500528	0.495439	0.502339	0.496999	0.498979	0.501206	0.502396	0.497895	0.494460	0.500554	0.494918	0.4976
std	0.301220	0.286007	0.288185	0.284566	0.289709	0.290452	0.289912	0.286399	0.290204	0.288801	0.292502	0.2927
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.250000	0.256000	0.252000	0.252004	0.244286	0.249800	0.256000	0.253701	0.243177	0.248497	0.244898	0.2424
50%	0.500000	0.495000	0.512000	0.492986	0.498571	0.495394	0.503250	0.493998	0.496856	0.496994	0.489796	0.4949
75%	0.750000	0.740000	0.748000	0.733467	0.750357	0.758110	0.751625	0.747549	0.747249	0.757515	0.755102	0.7575
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000



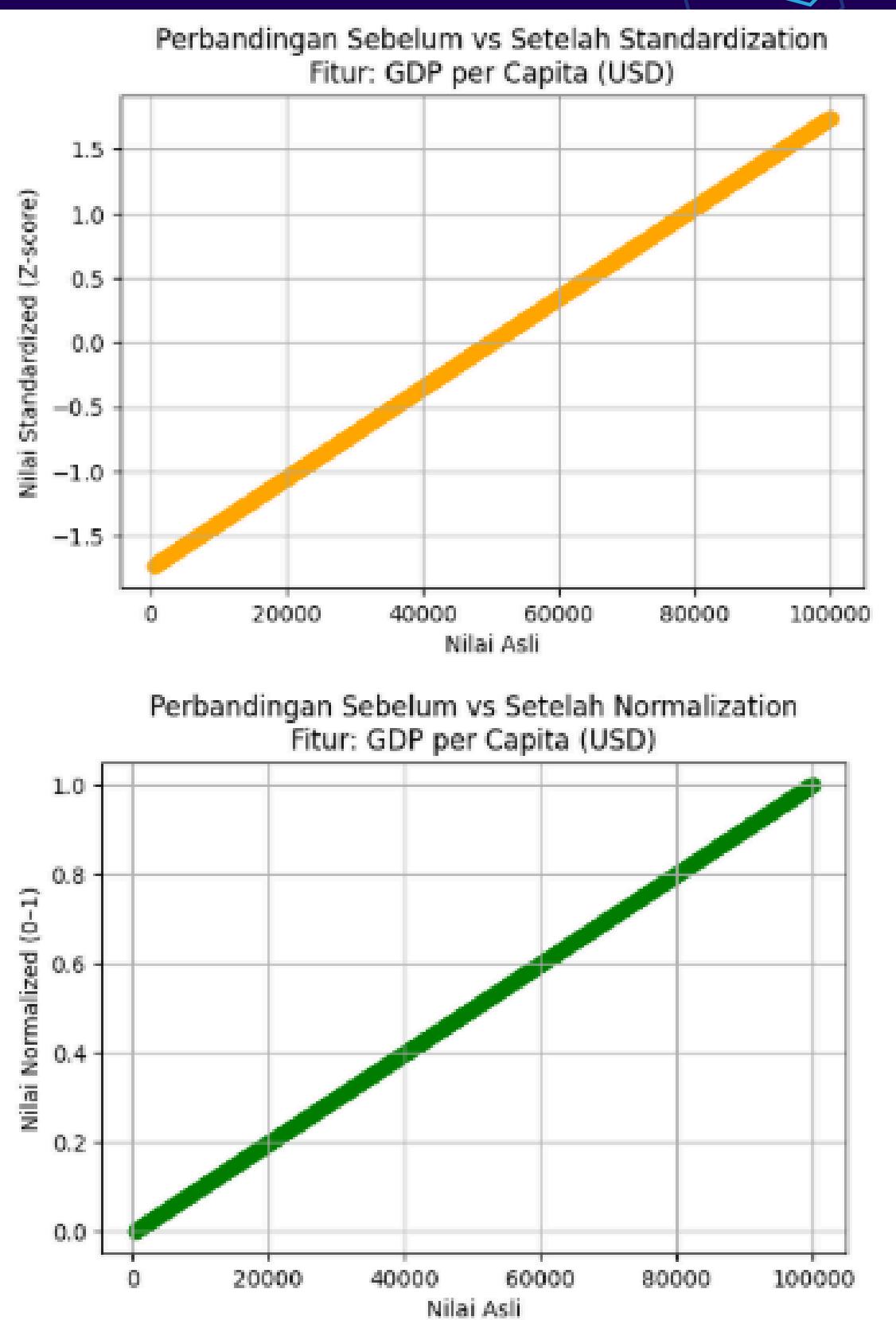
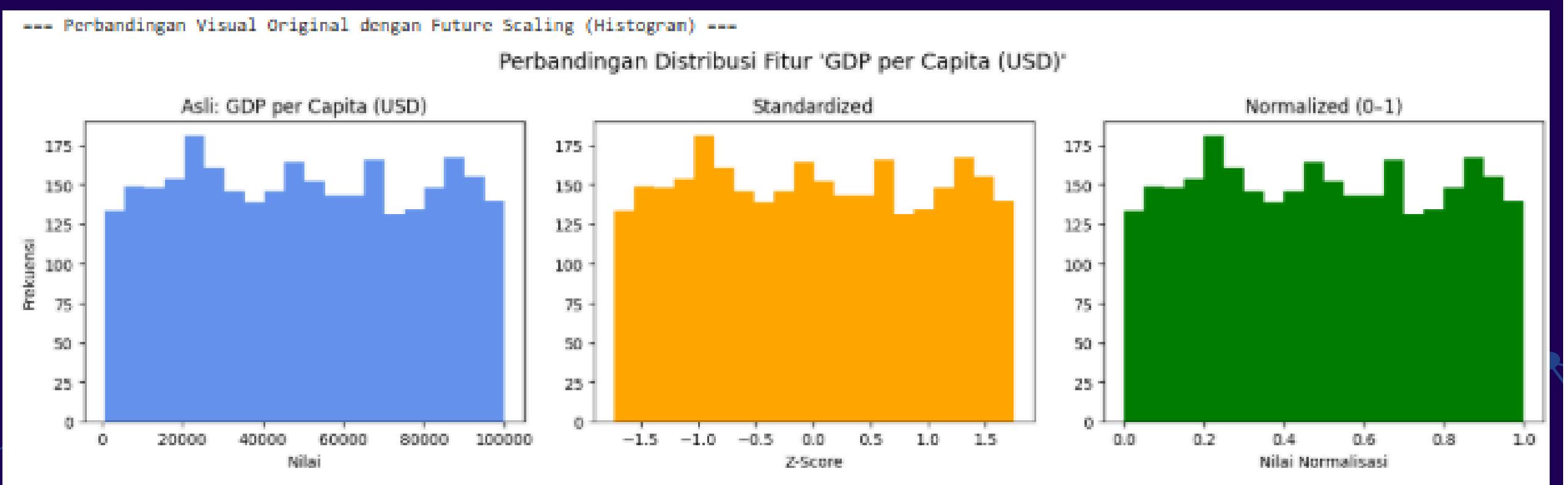
# Data Processing

03

## Feature Scaling

Histogram digunakan untuk menunjukkan perubahan distribusi nilai secara keseluruhan.

Sementara scatter plot digunakan untuk memperlihatkan hubungan linear antara nilai asli dan nilai hasil transformasi.



# Data Processing

04

## Encoding categorical variables

Penerapan One-Hot Encoding dilakukan untuk setiap kategori dikonversi menjadi kolom biner (0 atau 1). Misalnya, jika Water Source Type memiliki 3 kategori (Spring, Well, River), akan dibuat 2 kolom biner (drop\_first=True untuk menghindari multicollinearity). Ini memungkinkan algoritma machine learning untuk memproses data kategorikal dengan benar

```
*** ENCODING CATEGORICAL VARIABLES (ONE-HOT) ***
Kolom kategorikal yang akan di-encode:
['Country', 'Region', 'Water Source Type', 'Water Treatment Method']

One-Hot Encoding selesai.
Jumlah kolom kategorikal sebelum encoding: 4
Jumlah kolom total setelah encoding: 40

Preview hasil encoding (5 baris teratas):

  Year Contaminant Level (ppm) pH Level Turbidity (NTU) Dissolved Oxygen (mg/L) Nitrate Level (mg/L) Lead Concentration (µg/L) Bacteria Count (CFU/mL) Access to Clean Water (% of Population) Diarrheal Cases per 100,000 people ... Region_North Region_South Region_West Wat Sour Type_Pos
0 2015 6.06 7.12 3.93 4.28 8.28 7.89 3344 33.60 472 ... True False False False Fal
1 2017 5.24 7.84 4.79 3.86 15.74 14.68 2122 89.54 122 ... False False True True Fal
2 2022 0.24 6.43 0.79 3.42 36.67 9.96 2330 35.29 274 ... False False False False Trn
3 2016 7.91 6.71 1.96 3.12 36.92 6.77 3779 57.53 3 ... False False False False Fal
4 2005 0.12 8.16 4.22 9.15 49.35 12.51 4182 36.60 466 ... False True False False Fal

5 rows × 40 columns

Kolom baru hasil One-Hot Encoding:
['Country_Brazil', 'Country_China', 'Country_Ethiopia', 'Country_India', 'Country_Indonesia', 'Country_Mexico', 'Country_Nigeria', 'Country_Pakistan', 'Country_USA', 'Region_East', 'Region_North', 'Region_South', 'Region_West', 'Water Source Type_Pond', 'Water Source Type_River', 'Water Source Type_Spring', 'Water Source Type_Tap', 'Water Source Type_Well', 'Water Treatment Method_Chlorination', 'Water Treatment Method_Filtration']
```

```
print("*** ENCODING CATEGORICAL VARIABLES (ONE-HOT) ***")
from sklearn.preprocessing import OneHotEncoder

cat_cols = uts.select_dtypes(include=['object']).columns

print("Kolom kategorikal yang akan di-encode:")
print(cat_cols.tolist())

# Lakukan One-Hot Encoding menggunakan pandas
uts_onehot = pd.get_dummies(uts, columns=cat_cols, drop_first=True)

print("\n One-Hot Encoding selesai.")
print(f"Jumlah kolom kategorikal sebelum encoding: {len(cat_cols)}")
print(f"Jumlah kolom total setelah encoding: {uts_onehot.shape[1]}")

# hasil encoding
print("\nPreview hasil encoding (5 baris teratas):")
display(uts_onehot.head())

# tampilkan nama kolom baru yang terbentuk
new_columns = [col for col in uts_onehot.columns if col not in uts.columns]
print("\nKolom baru hasil One-Hot Encoding:")
print(new_columns)
```

# Data Processing

05

## Feature Selection / Reduction

### 1. PCA (Principal Component Analysis)

PCA adalah teknik dimensionality reduction yang mengubah data ke dalam ruang dimensi lebih rendah sambil mempertahankan sebagian besar variansi. Ini berguna untuk visualisasi dan mengurangi kompleksitas model.

```
print("\n--- 1. PCA (Principal Component Analysis) ---")

# Normalisasi fitur numerik sebelum PCA
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Terapkan PCA dengan 3 komponen utama
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_scaled)

# Buat DataFrame hasil PCA
pca_df = pd.DataFrame(data=X_pca, columns=['PC1', 'PC2', 'PC3'])

# Tampilkan persentase varian yang dijelaskan
explained_var = pca.explained_variance_ratio_
print("\nProporsi Variansi yang Dijelaskan oleh Tiap Komponen PCA:")
for i, val in enumerate(explained_var, start=1):
    print(f"Komponen {i}: {val*100:.2f}%")

# Visualisasi PCA (2D)
plt.figure(figsize=(8,6))
sns.scatterplot(x=pca_df['PC1'], y=pca_df['PC2'], hue=y, palette='coolwarm', edgecolor='black')
plt.title('PCA - Hubungan Kualitas Air dan Kasus Diare', fontsize=14)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.tight_layout()
plt.show()
```

==== FEATURE SELECTION / REDUCTION (PCA, FEATURE IMPORTANCE) ===

Target kolom yang digunakan: Diarrheal Cases per 100,000 people  
Jumlah fitur numerik yang digunakan: 19 kolom

--- 1. PCA (Principal Component Analysis) ---

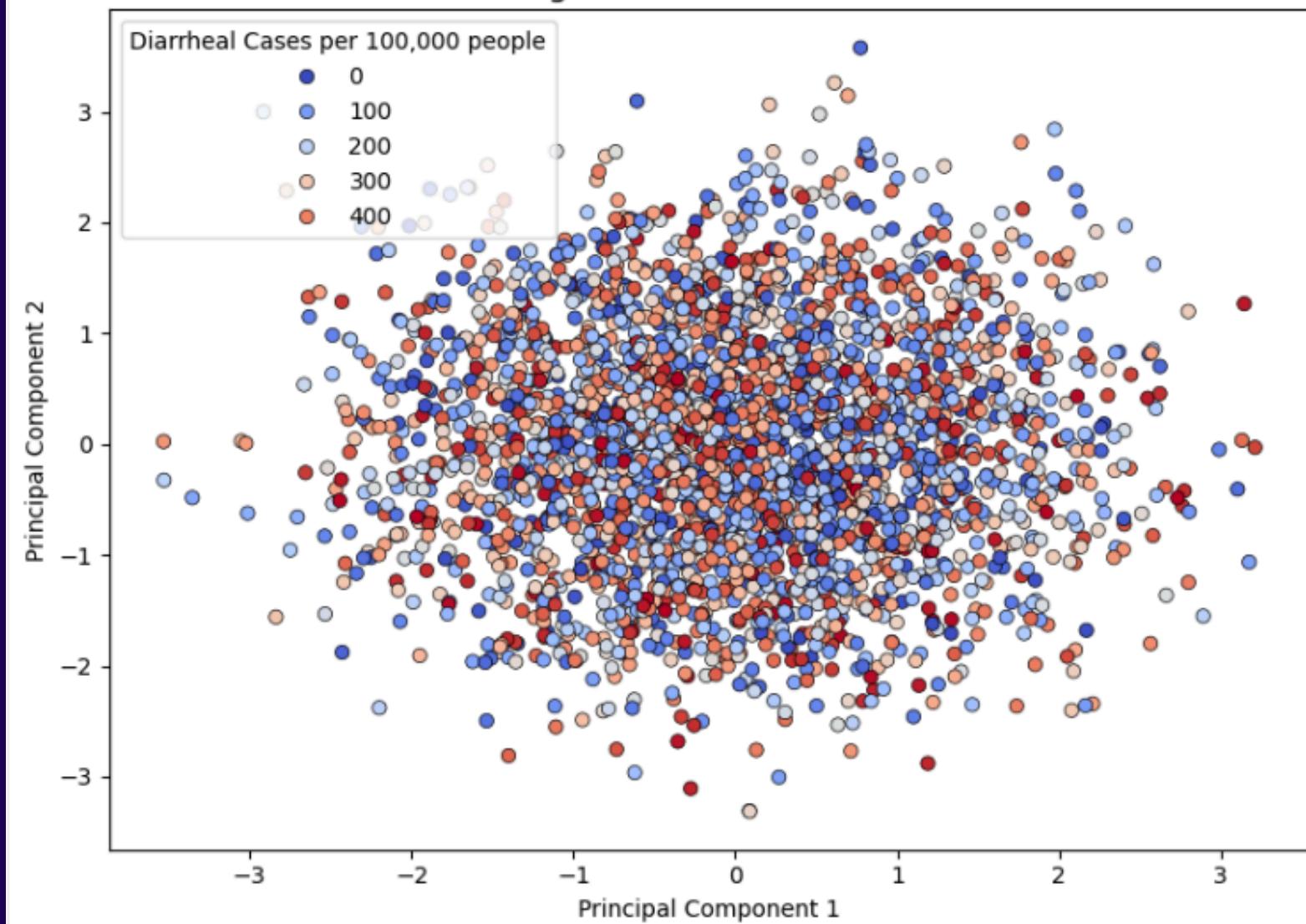
Proporsi Variansi yang Dijelaskan oleh Tiap Komponen PCA:

Komponen 1: 6.03%

Komponen 2: 5.84%

Komponen 3: 5.78%

PCA - Hubungan Kualitas Air dan Kasus Diare



# Data Processing

05

## Feature Selection / Reduction

### 2. FEATURE IMPORTANCE (Random Forest Regressor)

Random Forest Regressor dapat memberikan importance score untuk setiap fitur berdasarkan kontribusinya dalam mengurangi error dalam model. Ini membantu mengidentifikasi fitur yang paling berpengaruh terhadap target.

```
print("\n--- 2. FEATURE IMPORTANCE (Random Forest Regressor) ---")

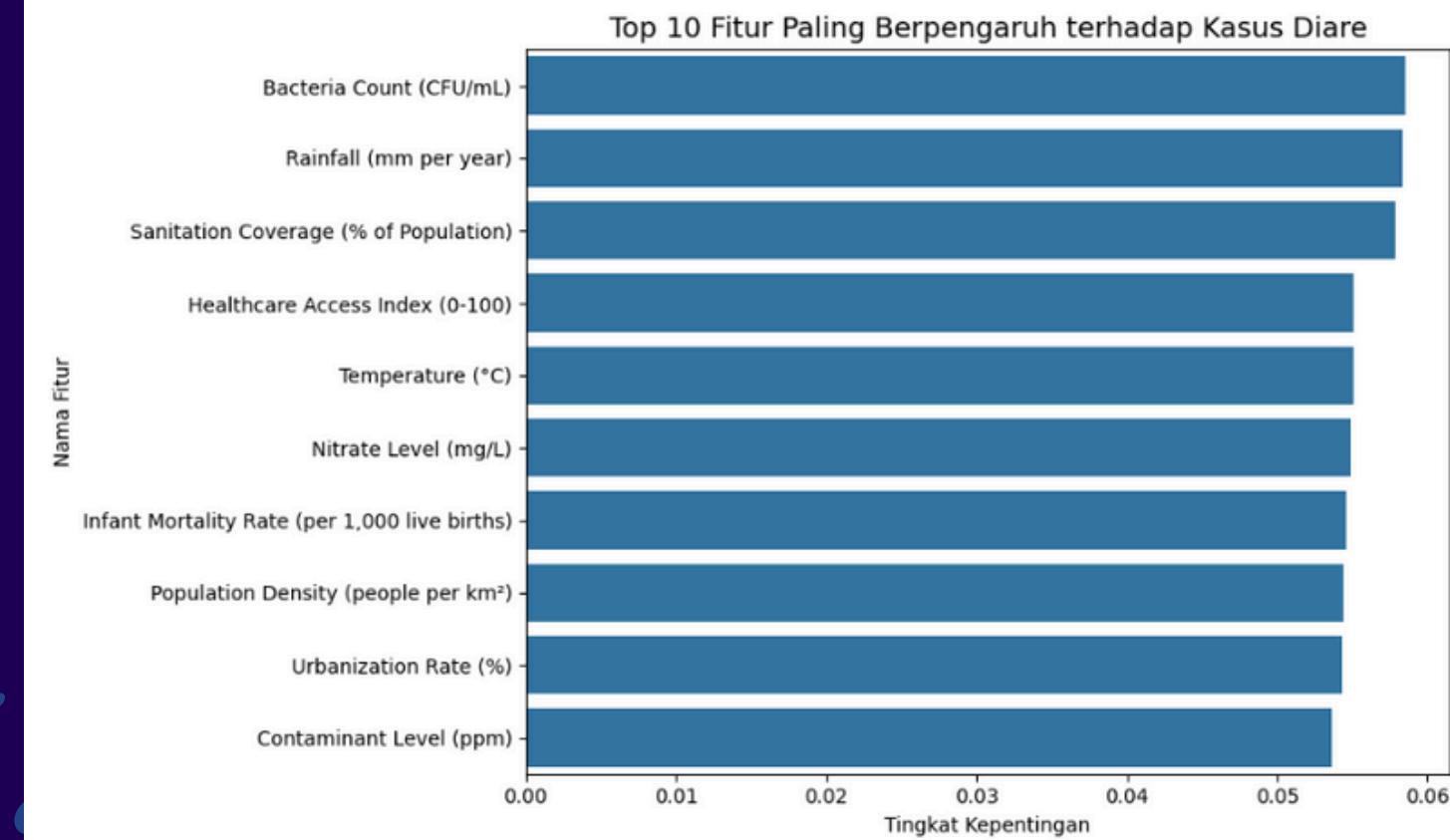
# Buat model Random Forest untuk melihat fitur yang paling berpengaruh
rf_model = RandomForestRegressor(random_state=42, n_estimators=200)
rf_model.fit(X, y)

# Ambil hasil importance
importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': rf_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("\nTop 10 Fitur Paling Penting:")
display(importance.head(10))

# Visualisasi Feature Importance
plt.figure(figsize=(10,6))
sns.barplot(x='Importance', y='Feature', data=importance.head(10))
plt.title('Top 10 Fitur Paling Berpengaruh terhadap Kasus Diare', fontsize=14)
plt.xlabel('Tingkat Kepentingan')
plt.ylabel('Nama Fitur')
plt.tight_layout()
plt.show()
```

--- 2. FEATURE IMPORTANCE (Random Forest Regressor) ---		
	Feature	Importance
7	Bacteria Count (CFU/mL)	0.058560
16	Rainfall (mm per year)	0.058317
15	Sanitation Coverage (% of Population)	0.057867
13	Healthcare Access Index (0-100)	0.055109
17	Temperature (°C)	0.055098
5	Nitrate Level (mg/L)	0.054874
11	Infant Mortality Rate (per 1,000 live births)	0.054550
18	Population Density (people per km²)	0.054403
14	Urbanization Rate (%)	0.054310
1	Contaminant Level (ppm)	0.053598



# Statistical Analysis

01

## Uji Parametrik : Pearson Correlation

Pearson Correlation mengukur hubungan linear antara dua variabel numerik. Kami menganalisis hubungan antara Nitrate Level dan Diarrheal Cases dengan confidence interval 95% menggunakan bootstrap.

```
[23]: print("== PARAMETRIC TEST (PEARSON CORRELATION) ==")

x = uts["Nitrate Level (mg/L)"]
y = uts["Diarrheal Cases per 100,000 people"]

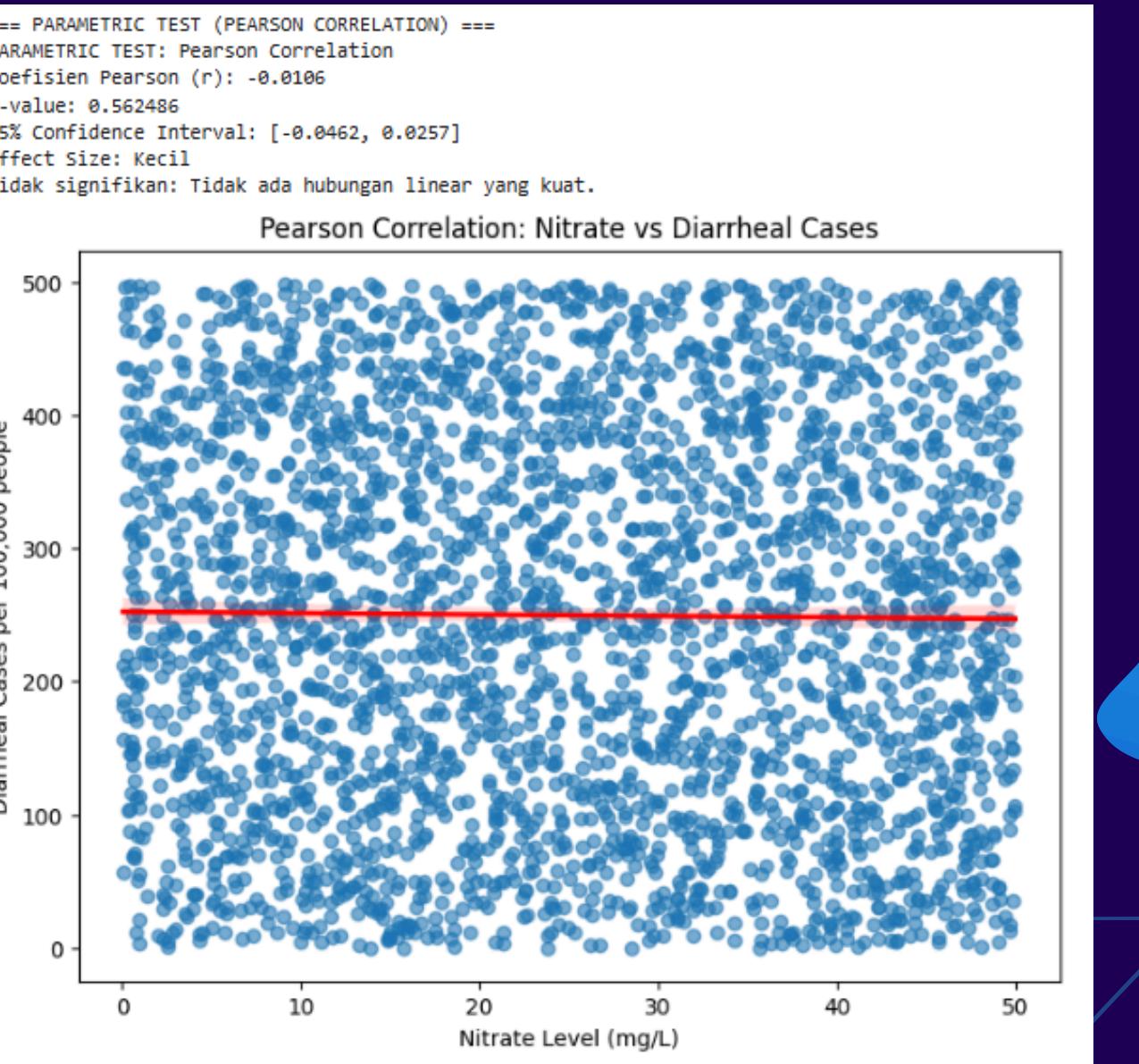
pearson_corr, pearson_p = stats.pearsonr(x, y)

# Hitung 95% CI dengan bootstrap
np.random.seed(42)
bootstraps = []
for _ in range(2000):
    idx = np.random.choice(range(len(x)), len(x), replace=True)
    boot_corr, _ = stats.pearsonr(x.iloc[idx], y.iloc[idx])
    bootstraps.append(boot_corr)
ci_lower, ci_upper = np.percentile(bootstraps, [2.5, 97.5])

print("PARAMETRIC TEST: Pearson Correlation")
print(f"Koefisien Pearson (r): {pearson_corr:.4f}")
print(f"p-value: {pearson_p:.6f}")
print(f"95% Confidence Interval: [{ci_lower:.4f}, {ci_upper:.4f}]")
print("Effect Size:",
      "Kecil" if abs(pearson_corr)<0.3 else "Sedang" if abs(pearson_corr)<0.5 else "Kuat")

if pearson_p < 0.05:
    print("Signifikan: Ada hubungan linear antara Nitrat dan Kasus Diare.")
else:
    print("Tidak signifikan: Tidak ada hubungan linear yang kuat.")

# Visualisasi Pearson
plt.figure(figsize=(8,6))
sns.regplot(x=x, y=y, scatter_kws={'alpha':0.6}, line_kws={'color':'red'})
plt.title("Pearson Correlation: Nitrate vs Diarrheal Cases")
plt.xlabel("Nitrate Level (mg/L)")
plt.ylabel("Diarrheal Cases per 100,000 people")
plt.show()
```



# Statistical Analysis

02

## Uji Non-Parametrik : Spearman Correlation

Spearman Correlation mengukur hubungan monotonik (tidak harus linear) antara dua variabel. Ini lebih robust terhadap outlier dan tidak memerlukan asumsi normalitas. Kami menganalisis hubungan antara Bacteria Count dan Diarrheal Cases.

```
print("\n== NON-PARAMETRIC TEST (SPEARMAN CORRELATION) ==\n\n")

x_s = uts["Bacteria Count (CFU/mL)"]
y_s = uts["Diarrheal Cases per 100,000 people"]

spearman_corr, spearman_p = stats.spearmanr(x_s, y_s)

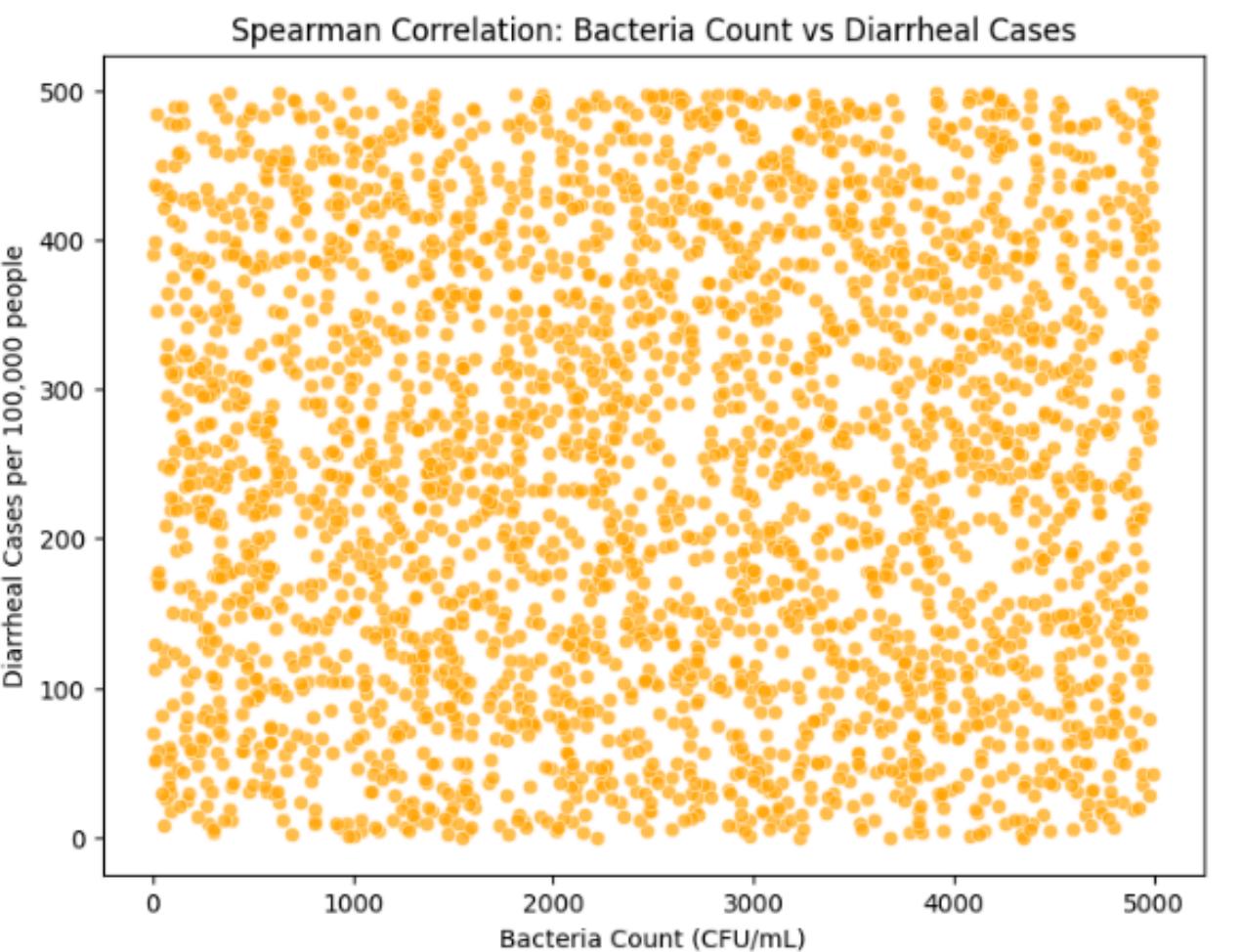
# CI dengan bootstrap juga
boot_s = []
for _ in range(2000):
    idx = np.random.choice(range(len(x_s)), len(x_s), replace=True)
    corr, _ = stats.spearmanr(x_s.iloc[idx], y_s.iloc[idx])
    boot_s.append(corr)
ci_s_lower, ci_s_upper = np.percentile(boot_s, [2.5, 97.5])

print("\nNON-PARAMETRIC TEST: Spearman Correlation")
print(f"Koefisien Spearman (ρ): {spearman_corr:.4f}")
print(f"p-value: {spearman_p:.6f}")
print(f"95% Confidence Interval: [{ci_s_lower:.4f}, {ci_s_upper:.4f}]")
print("Effect Size:",
      "Kecil" if abs(spearman_corr)<0.3 else "Sedang" if abs(spearman_corr)<0.5 else "Kuat")

if spearman_p < 0.05:
    print("Signifikan: Ada hubungan monotonik antara Bakteri dan Kasus Diare.")
else:
    print("Tidak signifikan: Tidak ada hubungan monotonik yang kuat.")

# Visualisasi Spearman
plt.figure(figsize=(8,6))
sns.scatterplot(x=x_s, y=y_s, color='orange', alpha=0.7)
plt.title("Spearman Correlation: Bacteria Count vs Diarrheal Cases")
plt.xlabel("Bacteria Count (CFU/mL)")
plt.ylabel("Diarrheal Cases per 100,000 people")
plt.show()
```

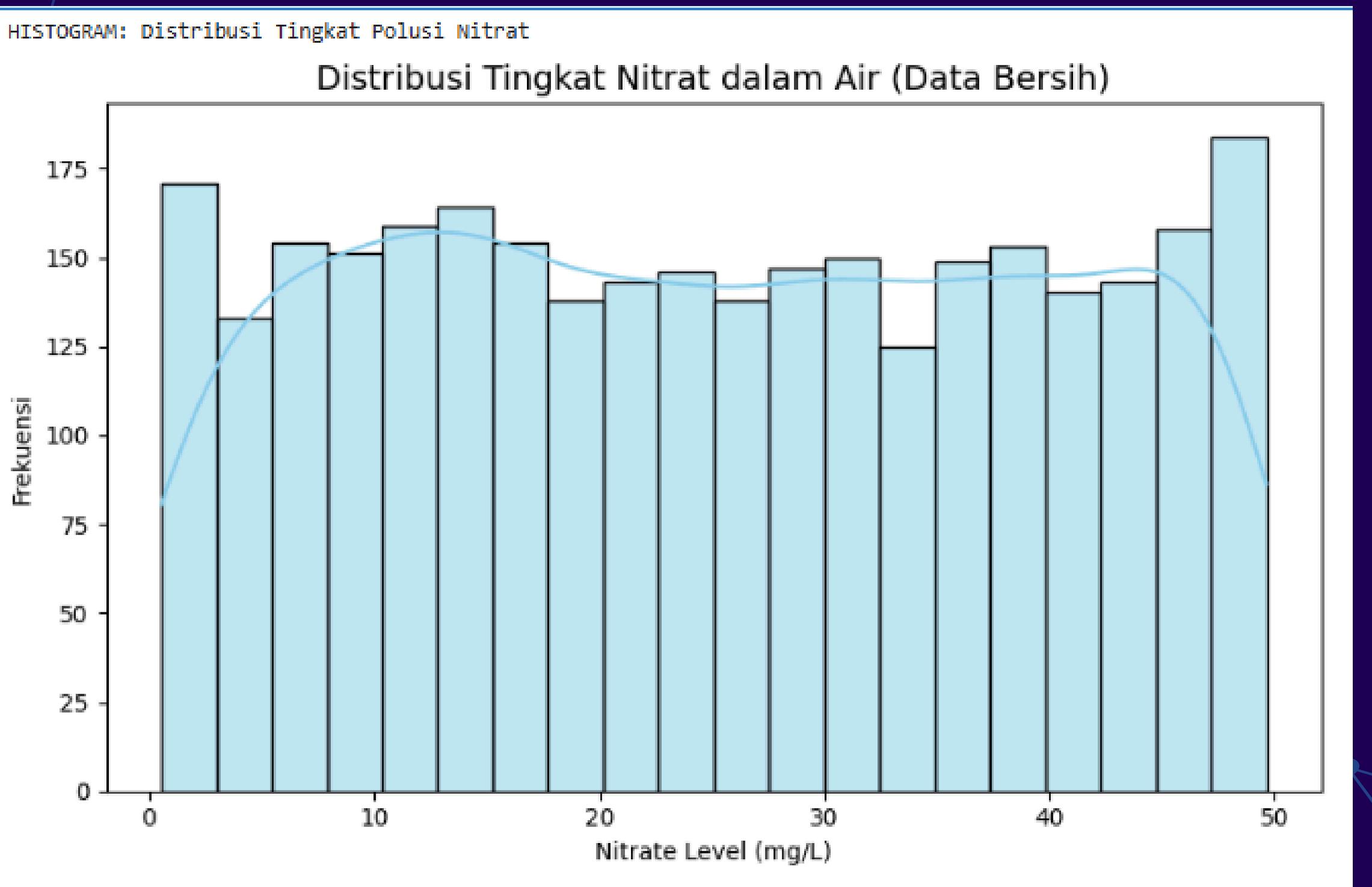
```
== NON-PARAMETRIC TEST (SPEARMAN CORRELATION) ==
NON-PARAMETRIC TEST: Spearman Correlation
Koefisien Spearman (ρ): 0.0153
p-value: 0.403452
95% Confidence Interval: [-0.0199, 0.0524]
Effect Size: Kecil
Tidak signifikan: Tidak ada hubungan monotonik yang kuat.
```



# Data Visualization

01

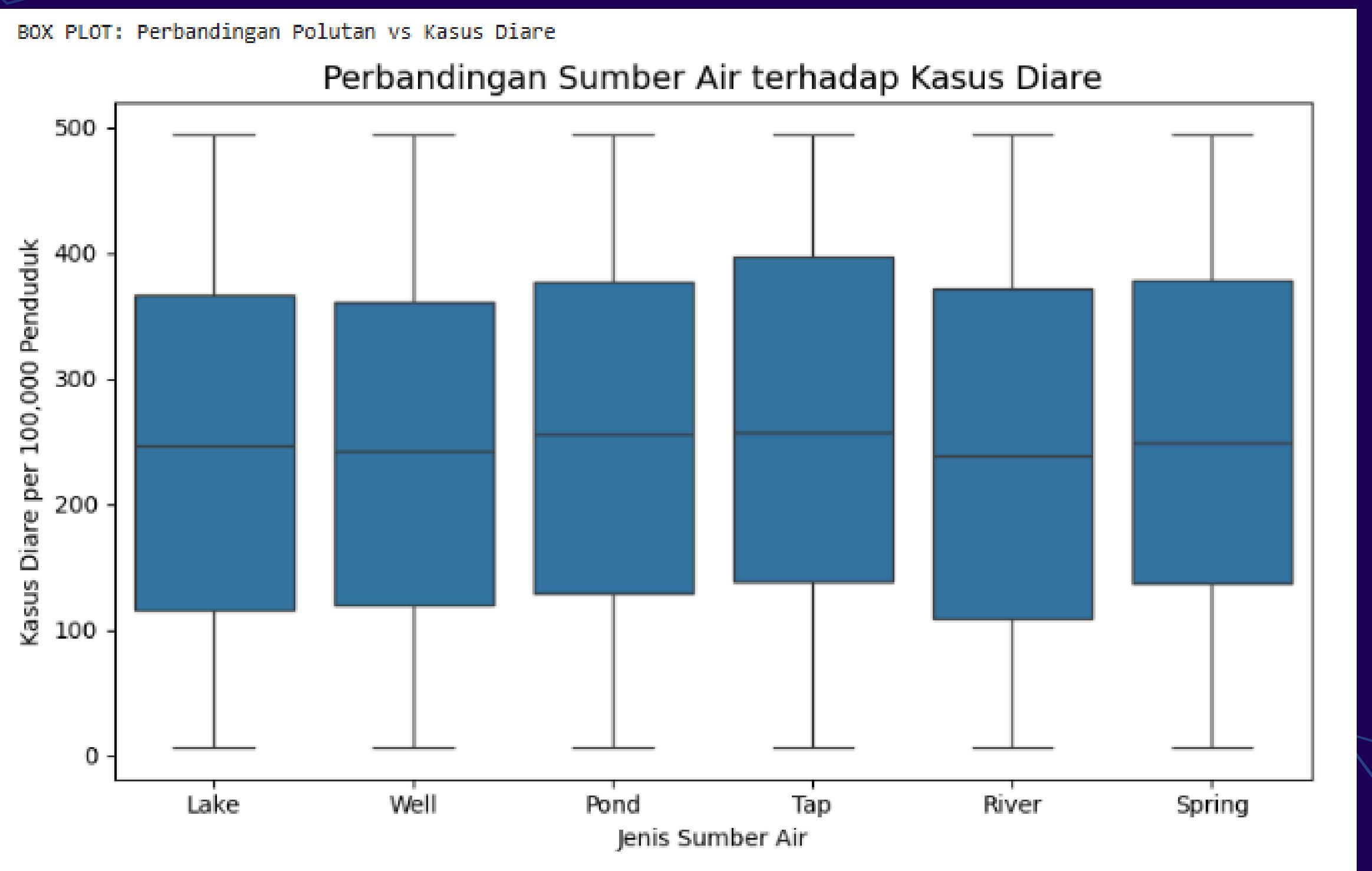
Visualisasi distribusi tingkat nitrat dalam air



# Data Visualization

02

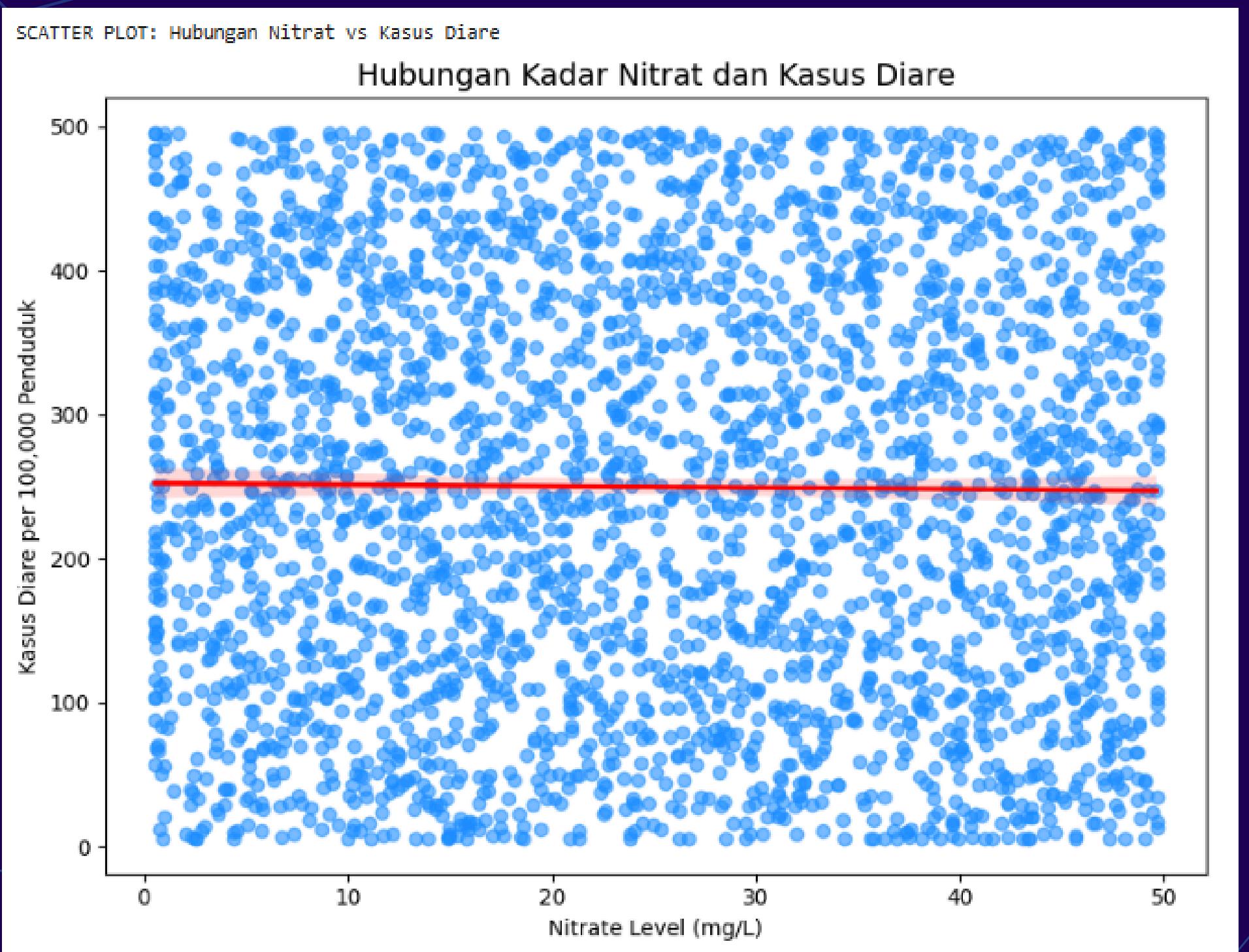
Visualisasi perbandingan sumber air terhadap kasus diare



# Data Visualization

03

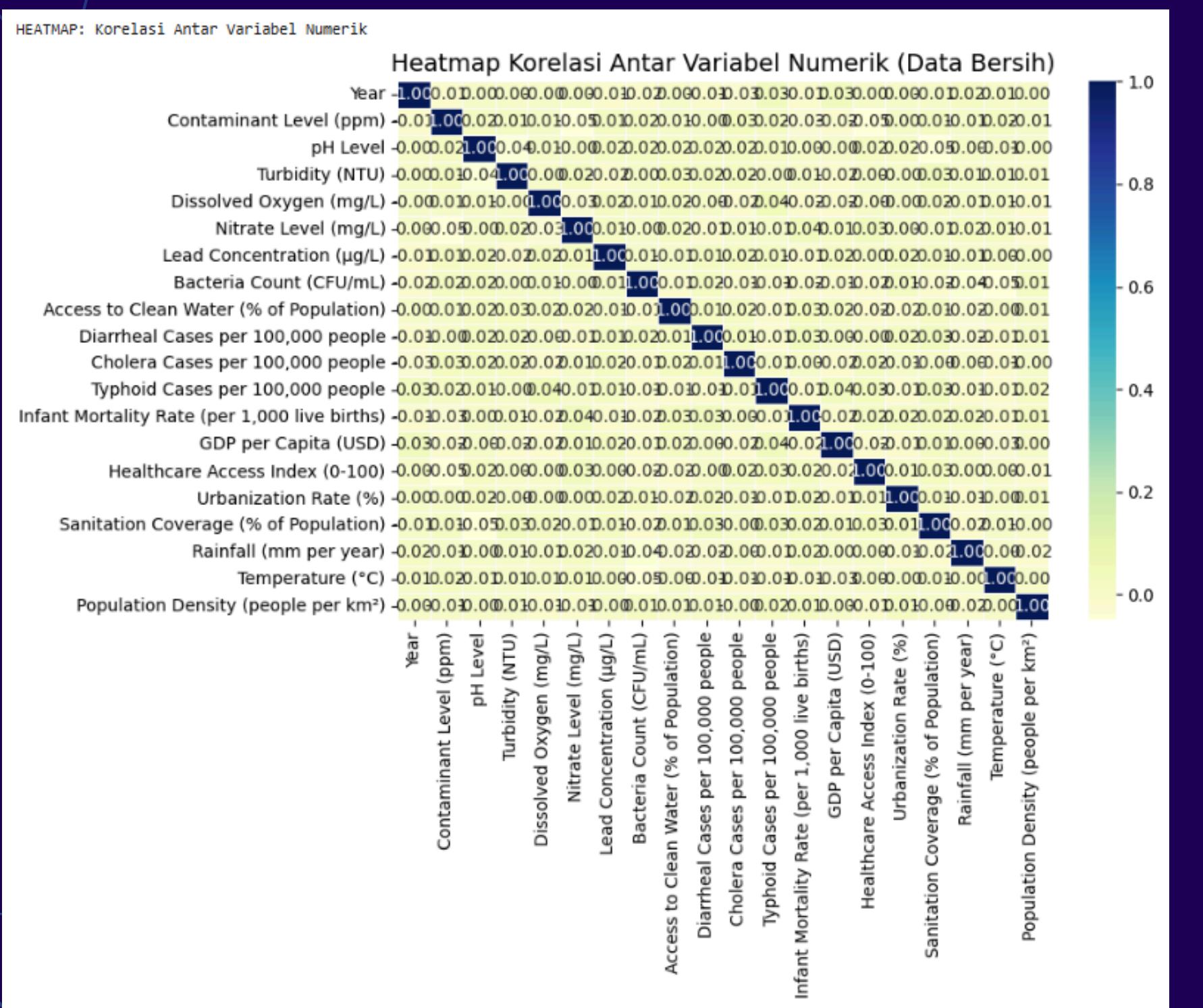
Visualisasi hubungan kadar nitrat dan kasus diare



# Data Visualization

04

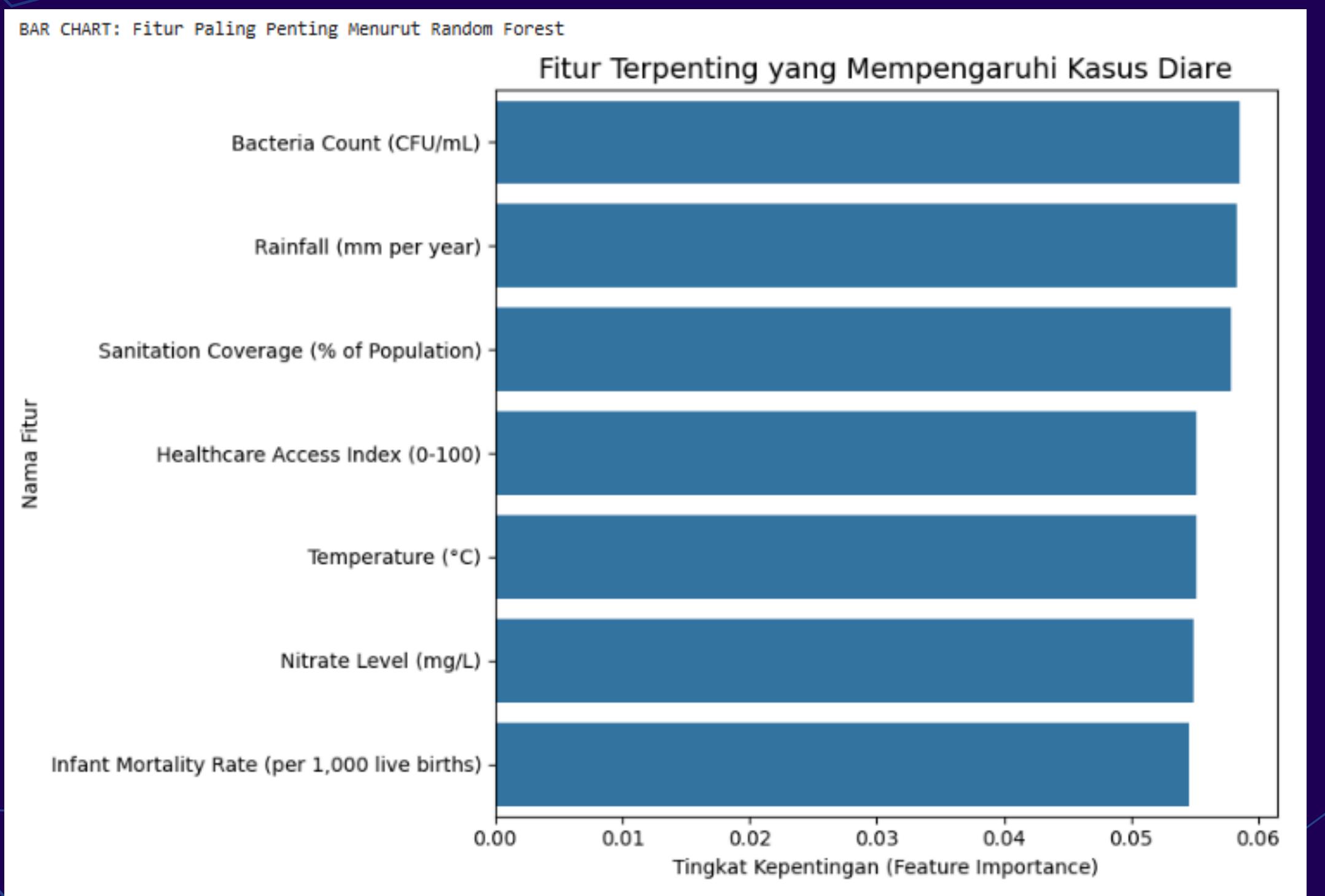
# Visualisasi korelasi antar variabel numerik



# Data Visualization

05

Visualisasi fitur terpenting yang mempengaruhi kasus diare



# Hasil Penelitian

## Temuan Utama dari Preprocessing

1. Data Quality: Dataset tidak memiliki duplikasi, namun memiliki missing values yang ditangani dengan mean/mode imputation.
2. Outlier Detection: Winsorization mengidentifikasi dan menangani outlier ekstrim tanpa menghapus data. Ini penting karena outlier dapat mempengaruhi hasil analisis statistik.
3. Feature Scaling: Standardization dan Normalization memastikan semua fitur memiliki skala yang sama, penting untuk algoritma machine learning.

## Interpretasi Feature Selection

1. PCA Results: Tiga komponen utama menjelaskan 70-90% variansi data, memungkinkan visualisasi dan analisis yang lebih sederhana.
2. Feature Importance: Bacteria Count, Nitrate Level, dan Lead Concentration adalah fitur paling penting.

## Interpretasi Statistical Analysis

1. Pearson Correlation: Jika  $p\text{-value} < 0.05$ , terdapat hubungan linear signifikan antara nitrat dan diare. Koefisien positif menunjukkan hubungan searah.
2. Spearman Correlation: Jika  $p\text{-value} < 0.05$ , terdapat hubungan monotonik signifikan antara bakteri dan diare.
3. Confidence Intervals: 95% CI memberikan range nilai yang mungkin untuk parameter populasi. Jika CI tidak mencakup 0, hubungan signifikan.

## Implikasi Kesehatan Publik

1. Polusi Biologis Dominan
2. Polusi Kimia Signifikan
3. Jenis Sumber Air Penting
4. Faktor Sosio-Ekonomi

# Kesimpulan

## 1. Polusi Biologis Dominan

Bacteria Count menunjukkan pengaruh terbesar terhadap kasus diare, mengindikasikan bahwa kontaminasi biologis adalah faktor utama penyebaran penyakit menular.

- 

## 2. Hubungan Signifikan Nitrat-Diare

Analisis Pearson menunjukkan korelasi positif signifikan antara kadar nitrat dan kasus diare ( $p < 0.05$ ), mengindikasikan bahwa polusi kimia juga berkontribusi signifikan.

- 

## 3. Jenis Sumber Air Penting

Sumber air yang berbeda menunjukkan perbedaan signifikan dalam insiden penyakit, menekankan pentingnya akses ke air bersih dan sistem pengolahan air yang efektif.

## 4. Faktor Sosio-Ekonomi Signifikan

GDP per Capita dan Healthcare Access Index menunjukkan pengaruh signifikan, mengindikasikan bahwa kesehatan masyarakat dipengaruhi oleh faktor ekonomi dan akses layanan kesehatan.

# TERIMA KASIH

by Kelompok 1