Modul 2
Praktikum Data Mining

PROGRAM STUDI SAINS DATA
FAKULTAS SAINS
INSTITUT TEKNOLOGI SUMATERA
2025

Tujuan Praktikum

1. Mahasiswa memahami konsep data cleaning dan data integration.
2. Mampu menerapkan teknik pembersihan data (missing value, outlier, inkonsistensi, duplikat).
3. Mampu mengintegrasikan dataset dengan merge dan concatenate.
4. Mampu melakukan analisis korelasi data numerik dan kategorik.

Data Cleaning

Data Cleaning adalah proses penting dalam data mining yang bertujuan:

1. Memastikan kualitas data.
2. Menghapus atau memperbaiki nilai yang tidak valid, hilang, atau tidak konsisten.
3. Meningkatkan akurasi model machine learning.

Contoh masalah pada data:

a. Nilai hilang (*missing values*)

   Beberapa cara mengatasi nilai yang hilang

   1. Abaikan (hapus data / record yang mengandung nilai kosong)
   2. Isi nilai kosong secara manual
   3. Imputasi nilai secara otomatis (mean, median, modus, *end-of-tail*, suka-suka, *random sample*)

b. Data duplikat

c. Kesalahan format (misalnya penulisan tanggal tidak konsisten)

d. Outlier (nilai ekstrem)

Data Integration

Data Integration (Integrasi Data) adalah proses menggabungkan data dari berbagai sumber ke dalam satu tampilan yang koheren dan konsisten. Proses ini penting dalam data mining karena:

1. Data sering tersebar di berbagai file, database, atau sistem.
2. Integrasi yang baik meningkatkan kualitas analisis data dan hasil prediksi.

Contoh kasus:

1. Menggabungkan data pelanggan dari sistem penjualan dan sistem keuangan.
2. Menggabungkan data karyawan dari HR dan sistem kehadiran.

Prosedur praktikum minggu 2 sebagai berikut:

➢ **Data Cleaning**

1. Import library yang diperlukan untuk data cleaning

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Input dataset praktikum

```
data = {
    "ID": [1, 2, 3, 4, 5, 6],
    "Nama": ["Andi", "Budi", "Cici", "Dedi", None, "Fina"],
    "Umur": [23, 25, np.nan, 22, 200, np.nan],  # missing + outlier
    "Gaji": [4000, 4200, 4100, None, 4300, 3900]
}
df = pd.DataFrame(data)
print("Data Awal:\n", df)
```

```
Data Awal:
    ID  Nama   Umur    Gaji
0   1  Andi   23.0  4000.0
1   2  Budi   25.0  4200.0
2   3  Cici    NaN  4100.0
3   4  Dedi   22.0     NaN
4   5  None  200.0  4300.0
5   6  Fina    NaN  3900.0
```

3. Menampilkan jumlah nilai yang hilang di setiap kolom

```
df.isnull().sum() #menampilkan jumlah nilai yang hilang di setiap kolom
```

	0
ID	0
Nama	1
Umur	2
Gaji	1

dtype: int64

4. Menghapus baris untuk menghapus data kosong

```
#Menghapus baris untuk menghapus data kosong
df_dropped = df.dropna()
print("\nData setelah menghapus baris dengan nilai yang hilang:\n", df_dropped)
```

```
Data setelah menghapus baris dengan nilai yang hilang:
    ID  Nama  Umur    Gaji
0   1  Andi  23.0  4000.0
1   2  Budi  25.0  4200.0
```

5. Menangani data yang hilang menggunakan mean

```
# Mean
df_mean = df.copy()
df_mean["Umur"].fillna(df_mean["Umur"].mean(), inplace=True)
df_mean["Gaji"].fillna(df_mean["Gaji"].mean(), inplace=True)

print("\nHasil Imputasi - Mean:\n", df_mean)
```

```
Hasil Imputasi - Mean:
    ID  Nama   Umur    Gaji
0   1  Andi   23.0  4000.0
1   2  Budi   25.0  4200.0
2   3  Cici   67.5  4100.0
3   4  Dedi   22.0  4100.0
4   5  None  200.0  4300.0
5   6  Fina   67.5  3900.0
```

6. Menangani data yang hilang menggunakan median

```
# Median
df_median = df.copy()
df_median["Umur"].fillna(df_median["Umur"].median(), inplace=True)
df_median["Gaji"].fillna(df_median["Gaji"].median(), inplace=True)
print("\nHasil Imputasi - Median:\n", df_median)
```

```
Hasil Imputasi - Median:
    ID  Nama   Umur    Gaji
0   1  Andi   23.0  4000.0
1   2  Budi   25.0  4200.0
2   3  Cici   24.0  4100.0
3   4  Dedi   22.0  4100.0
4   5  None  200.0  4300.0
5   6  Fina   24.0  3900.0
```

7. Menangani data yang hilang menggunakan modus

```
# Modus
df_mode = df.copy()
df_mode["Umur"].fillna(df_mode["Umur"].mode()[0], inplace=True)
df_mode["Gaji"].fillna(df_mode["Gaji"].mode()[0], inplace=True)
print("\nHasil Imputasi - Modus:\n", df_mode)
```

```
Hasil Imputasi - Modus:
    ID  Nama   Umur    Gaji
0   1  Andi   23.0  4000.0
1   2  Budi   25.0  4200.0
2   3  Cici   22.0  4100.0
3   4  Dedi   22.0  3900.0
4   5  None  200.0  4300.0
5   6  Fina   22.0  3900.0
```

8. Menangani data yang hilang menggunakan konstanta atau nilai suka-suka

```
# Konstanta
df_const = df.copy()
df_const["Umur"].fillna(30, inplace=True)
df_const["Gaji"].fillna(4000, inplace=True)
print("\nHasil Imputasi - Konstanta:\n", df_const)
```

```
Hasil Imputasi - Konstanta:
    ID  Nama   Umur    Gaji
0   1  Andi   23.0  4000.0
1   2  Budi   25.0  4200.0
2   3  Cici   30.0  4100.0
3   4  Dedi   22.0  4000.0
4   5  None  200.0  4300.0
5   6  Fina   30.0  3900.0
```

9. Menangani data yang hilang menggunakan *End-of-Tail*

```
# End of Tail
df_tail = df.copy()
umur_fill = df_tail["Umur"].mean() + 3 * df_tail["Umur"].std()
df_tail["Umur"].fillna(umur_fill, inplace=True)
gaji_fill = df_tail["Gaji"].mean() - 3 * df_tail["Gaji"].std()
df_tail["Gaji"].fillna(gaji_fill, inplace=True)
print("\nHasil Imputasi - End of Tail:\n", df_tail)
```

```
Hasil Imputasi - End of Tail:
    ID  Nama        Umur         Gaji
0   1  Andi   23.000000  4000.000000
1   2  Budi   25.000000  4200.000000
2   3  Cici  332.526414  4100.000000
3   4  Dedi   22.000000  3625.658351
4   5  None  200.000000  4300.000000
5   6  Fina  332.526414  3900.000000
```

10. Menangani data yang hilang menggunakan Random Sample

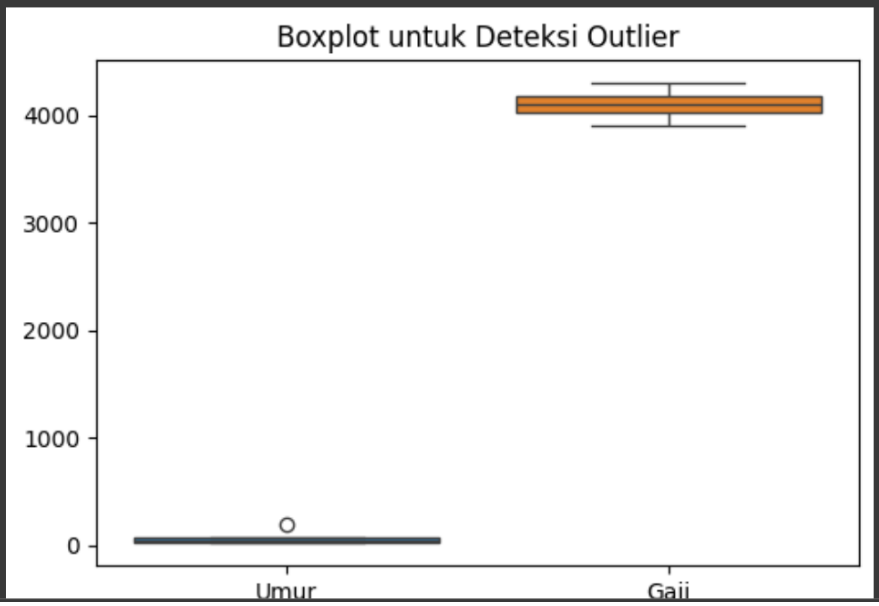
```
# Random Sample
df_random = df.copy()
df_random["Umur"] = df_random["Umur"].apply(
    lambda x: np.random.choice(df_random["Umur"].dropna()) if pd.isnull(x) else x
)
df_random["Gaji"] = df_random["Gaji"].apply(
    lambda x: np.random.choice(df_random["Gaji"].dropna()) if pd.isnull(x) else x
)
print("\nHasil Imputasi - Random Sample:\n", df_random)
```

```
Hasil Imputasi - Random Sample:
    ID  Nama   Umur    Gaji
0   1  Andi   23.0  4000.0
1   2  Budi   25.0  4200.0
2   3  Cici   22.0  4100.0
3   4  Dedi   22.0  4300.0
4   5  None  200.0  4300.0
5   6  Fina   25.0  3900.0
```

11. Deteksi outlier menggunakan visualisasi

```
# Gunakan data hasil imputasi mean
df_clean = df_mean.copy()

plt.figure(figsize=(6,4))
sns.boxplot(data=df_clean[["Umur", "Gaji"]])
plt.title("Boxplot untuk Deteksi Outlier")
plt.show()
```



12. Menangani outlier

```
Q1 = df_clean["Umur"].quantile(0.25)
Q3 = df_clean["Umur"].quantile(0.75)
IQR = Q3 - Q1
batas_bawah = Q1 - 1.5 * IQR
batas_atas = Q3 + 1.5 * IQR

df_no_outlier = df_clean[(df_clean["Umur"] >= batas_bawah) & (df_clean["Umur"] <= batas_atas)]
print("\nData tanpa Outlier:\n", df_no_outlier)
```

```
Data tanpa Outlier:
    ID  Nama  Umur    Gaji
0   1  Andi  23.0  4000.0
1   2  Budi  25.0  4200.0
2   3  Cici  67.5  4100.0
3   4  Dedi  22.0  4100.0
5   6  Fina  67.5  3900.0
```

13. Menangani data yang tidak konsisten

a. Kasus penulisan nama kota yang berbeda

```
data_kota = {
    "ID": [1, 2, 3, 4, 5],
    "Kota": ["Jakarta", "jakarta", "JKT", "Bandung", "bdg"]
}
df_kota = pd.DataFrame(data_kota)
df_kota["Kota"] = df_kota["Kota"].str.lower()
mapping_kota = {"jakarta": "Jakarta", "jkt": "Jakarta", "bandung": "Bandung", "bdg": "Bandung"}
df_kota["Kota"] = df_kota["Kota"].replace(mapping_kota)
print("\nData Kota Setelah Normalisasi:\n", df_kota)
```

```
Data Kota Setelah Normalisasi:
    ID     Kota
0   1  Jakarta
1   2  Jakarta
2   3  Jakarta
3   4  Bandung
4   5  Bandung
```

b. Kasus format tanggal yang berbeda

```
data_tanggal = {
    "ID": [1, 2, 3],
    "Tanggal": ["2025-01-05", "05/01/2025", "Jan 5, 2025"]
}
df_tgl = pd.DataFrame(data_tanggal)
df_tgl["Tanggal"] = pd.to_datetime(df_tgl["Tanggal"], errors="coerce")
print("\nData Tanggal Setelah Normalisasi:\n", df_tgl)
```

```
Data Tanggal Setelah Normalisasi:
    ID    Tanggal
0   1 2025-01-05
1   2        NaT
2   3        NaT
```

14. Mengatasi data duplikat

```
data_duplikat = {
    "ID": [1, 2, 2, 3, 4, 4],
    "Nama": ["Andi", "Budi", "Budi", "Cici", "Dedi", "Dedi"],
    "Umur": [23, 25, 25, 22, 24, 24]
}
df_dup = pd.DataFrame(data_duplikat)
print("\nJumlah Duplikat:", df_dup.duplicated().sum())
df_no_dup = df_dup.drop_duplicates(keep="first")
print("\nData Setelah Hapus Duplikat:\n", df_no_dup)
```

```
Jumlah Duplikat: 2

Data Setelah Hapus Duplikat:
    ID  Nama  Umur
0   1  Andi    23
1   2  Budi    25
3   3  Cici    22
4   4  Dedi    24
```

➢ Data Integration

1. Menggabungkan 2 datasets

```
data_gaji = {
    "ID": [1, 2, 3, 4, 5, 6],
    "Bonus": [500, 600, 550, 650, 700, 580],
    "Departemen": ["IT", "HR", "Finance", "IT", "HR", "Finance"]
}
df_gaji = pd.DataFrame(data_gaji)

df_merge = pd.merge(df_no_outlier, df_gaji, on="ID", how="inner")
print("\nData Hasil Merge:\n", df_merge)
```

```
Data Hasil Merge:
    ID  Nama  Umur    Gaji  Bonus Departemen
0   1  Andi  23.0  4000.0    500         IT
1   2  Budi  25.0  4200.0    600         HR
2   3  Cici  67.5  4100.0    550    Finance
3   4  Dedi  22.0  4100.0    650         IT
4   6  Fina  67.5  3900.0    580    Finance
```

2. Menambahkan baris pada datasets

```
data_cabang = {
    "ID": [7, 8],
    "Nama": ["Gilang", "Hana"],
    "Umur": [28, 27],
    "Gaji": [4500, 4700],
    "Bonus": [600, 650],
    "Departemen": ["IT", "HR"]
}
df_cabang = pd.DataFrame(data_cabang)

df_final = pd.concat([df_merge, df_cabang], ignore_index=True)
print("\nData Setelah Concatenate:\n", df_final)
```

```
Data Setelah Concatenate:
    ID    Nama  Umur    Gaji  Bonus Departemen
0   1    Andi  23.0  4000.0    500         IT
1   2    Budi  25.0  4200.0    600         HR
2   3    Cici  67.5  4100.0    550    Finance
3   4    Dedi  22.0  4100.0    650         IT
4   6    Fina  67.5  3900.0    580    Finance
5   7  Gilang  28.0  4500.0    600         IT
6   8    Hana  27.0  4700.0    650         HR
```

3. Analisis korelasi menggunakan Pearson Correlation dan Covariance (Data Numerik)

```
data_num = {
    "Umur": [23, 25, 22, 30, 28, 35, 40, 41, 29, 33],
    "Gaji": [4000, 4200, 3900, 5200, 5000, 6000, 7500, 7800, 4900, 5500],
    "Pengeluaran": [2000, 2100, 1800, 3000, 2800, 3500, 4000, 4200, 2700, 3100]
}
df_num = pd.DataFrame(data_num)

print("\nPearson Correlation:\n", df_num.corr(method="pearson"))
print("\nCovariance:\n", df_num.cov())
```

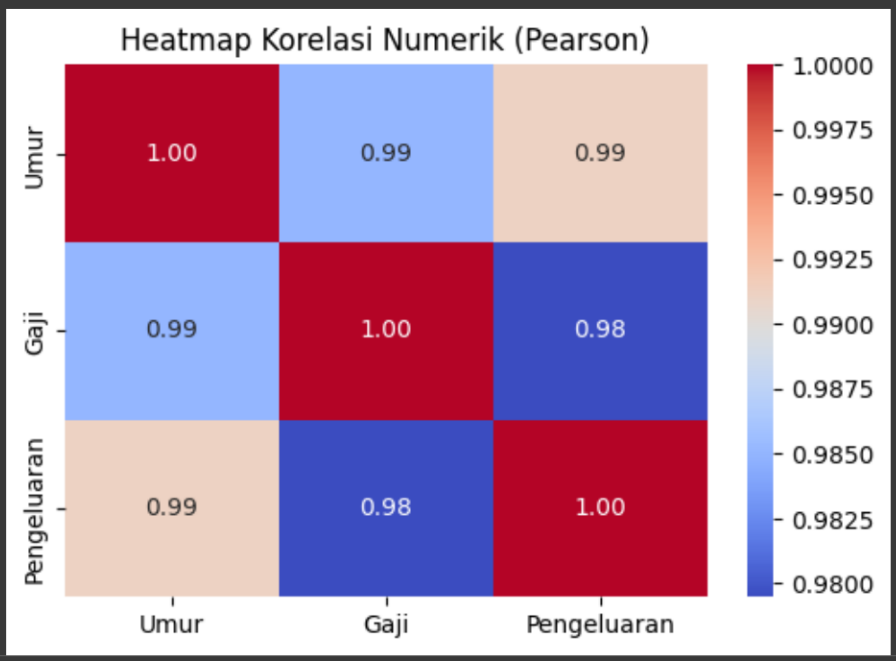
```
Pearson Correlation:
                Umur      Gaji  Pengeluaran
Umur        1.000000  0.985049     0.991163
Gaji        0.985049  1.000000     0.979480
Pengeluaran 0.991163  0.979480     1.000000

Covariance:
                    Umur          Gaji   Pengeluaran
Umur          43.822222  8.866667e+03  5.364444e+03
Gaji        8866.666667  1.848889e+06  1.088889e+06
Pengeluaran 5364.444444  1.088889e+06  6.684444e+05
```

4. Visualisasi Pearson menggunakan Heatmap

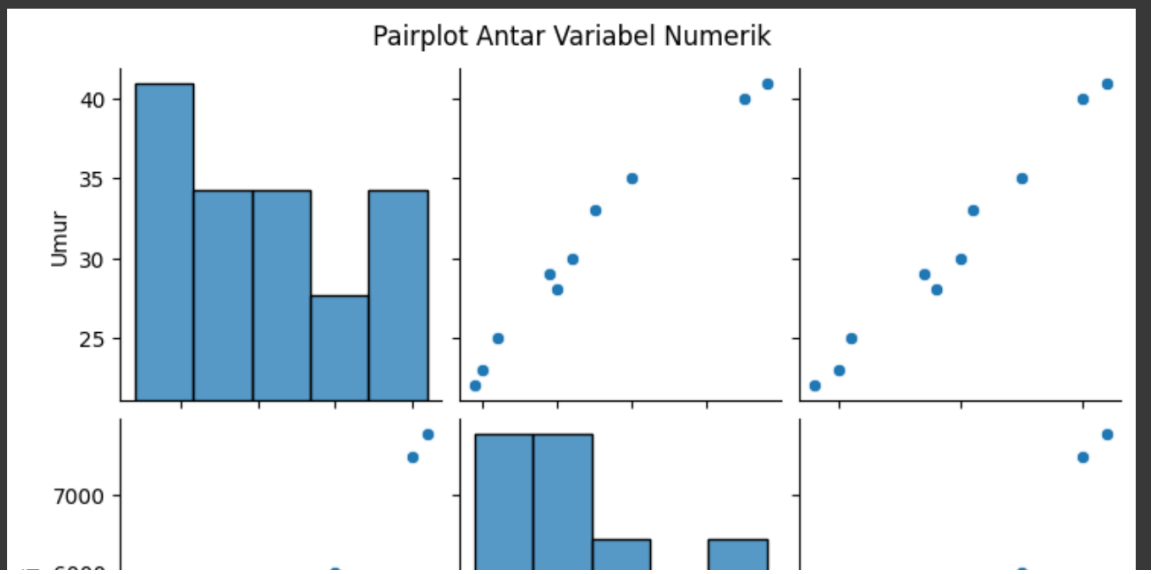
```
df_num = pd.DataFrame(data_num)

# Heatmap korelasi Pearson
plt.figure(figsize=(6,4))
sns.heatmap(df_num.corr(method="pearson"), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Heatmap Korelasi Numerik (Pearson)")
plt.show()
```



5. Visualisasi Pairplot untuk melihat hubungan numerik

```
sns.pairplot(df_num)
plt.suptitle("Pairplot Antar Variabel Numerik", y=1.02)
plt.show()
```



6. Analisis korelasi menggunakan Chi-Square test (Data Nominal)

```
import scipy.stats as stats

# Contoh data kategorik
data_cat = {
    "Gender": ["Pria", "Wanita", "Pria", "Wanita", "Pria", "Wanita", "Pria", "Wanita", "Pria", "Wanita"],
    "Pembelian": ["Ya", "Tidak", "Ya", "Ya", "Tidak", "Tidak", "Ya", "Ya", "Tidak", "Ya"]
}

df_cat = pd.DataFrame(data_cat)
print("Data Kategorik:\n", df_cat)

# --- Buat tabel kontingensi ---
contingency_table = pd.crosstab(df_cat["Gender"], df_cat["Pembelian"])
print("\nTabel Kontingensi:\n", contingency_table)

# --- Uji Chi-Square ---
chi2, p, dof, expected = stats.chi2_contingency(contingency_table)

print("\nChi-Square Test Result:")
print("Chi2 Statistic:", chi2)
print("p-value:", p)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:\n", expected)

# --- Interpretasi hasil ---
if p < 0.05:
    print("\nKesimpulan: Ada hubungan signifikan antara Gender dan Pembelian.")
else:
    print("\nKesimpulan: Tidak ada hubungan signifikan antara Gender dan Pembelian.")
```

```
Data Kategorik:
    Gender Pembelian
0    Pria        Ya
1  Wanita     Tidak
2    Pria        Ya
3  Wanita        Ya
4    Pria     Tidak
5  Wanita     Tidak
6    Pria        Ya
7  Wanita        Ya
8    Pria     Tidak
9  Wanita        Ya

Tabel Kontingensi:
 Pembelian  Tidak  Ya
Gender
Pria           2   3
Wanita         2   3

Chi-Square Test Result:
Chi2 Statistic: 0.0
p-value: 1.0
Degrees of Freedom: 1
Expected Frequencies:
 [[2. 3.]
 [2. 3.]]

Kesimpulan: Tidak ada hubungan signifikan antara Gender dan Pembelian.
```

7. Visualisasi Mosaic Plot pada Chi-Square test

```
from statsmodels.graphics.mosaicplot import mosaic
# Mosaic Plot
plt.figure(figsize=(6,4))
mosaic(df_cat, ["Gender", "Pembelian"])
plt.title("Mosaic Plot: Gender vs Pembelian")
plt.show()
```

```
<Figure size 600x400 with 0 Axes>
```