

Model Design and Data Preprocessing

Artificial Intelligence dan Big Data | AAK2KAB3 | Kur. 2024 | 2024/2025

Model Design

Model design adalah proses memilih dan merancang model yang akan digunakan untuk memecahkan masalah tertentu. Dalam machine learning, ada dua kategori utama model yang sering digunakan:

- **Supervised Learning:** Model yang dilatih menggunakan data yang sudah dilabeli, yaitu data yang memiliki target/output yang diketahui. Contoh: Linear Regression, Decision Trees, Random Forest, SVM, dll.
- **Unsupervised Learning:** Model yang dilatih menggunakan data yang tidak memiliki label, bertujuan untuk menemukan pola atau struktur dalam data. Contoh: K-Means Clustering, PCA, DBSCAN, dll.

Pemilihan model didasarkan pada:

- **Tipe data** (numerik, kategorikal, gambar, teks)
- **Tujuan analisis** (klasifikasi, regresi, clustering, dll)
- **Kinerja yang diinginkan** (akurasi, presisi, recall, dll)

Data Preprocessing

Imputation: Mengganti nilai/data yang hilang (missing value; NaN; blank) dengan nilai pengganti.

Jenis missing value:

1. Missing Completely At Random (MCAR): tidak bergantung pada nilai yang diketahui atau nilai yang hilang itu sendiri.

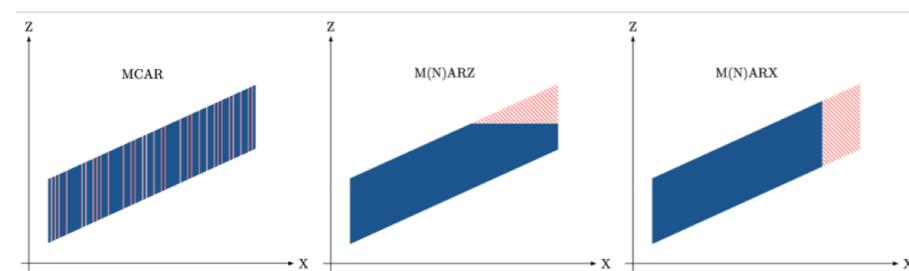
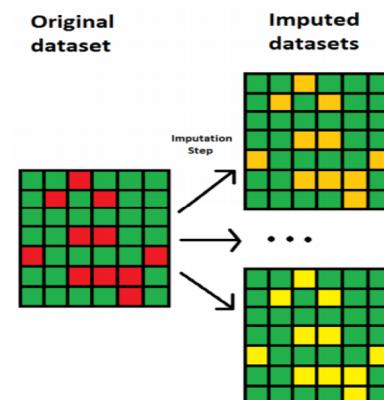
Tabel data dicetak tanpa nilai yang hilang dan seseorang secara tidak sengaja menjatuhkan beberapa tinta di atasnya sehingga beberapa sel tidak dapat dibaca lagi. Di sini, kita dapat mengasumsikan bahwa nilai yang hilang mengikuti distribusi yang sama dengan nilai yang diketahui.

2. Missing At Random (MAR): bergantung pada nilai yang diketahui tetapi tidak pada nilai yang hilang itu sendiri.

Dalam kasus sensor suhu, fakta bahwa suatu nilai hilang tidak bergantung pada suhu, tetapi mungkin bergantung pada beberapa faktor lain, misalnya pada daya baterai termometer.

3. Missing Not At Random (MNAR): bergantung pada nilai variable

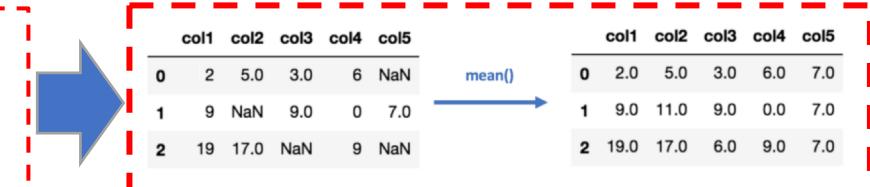
Dalam kasus sensor suhu, sensor tidak berfungsi dengan baik saat suhu lebih dingin dari 5°C.



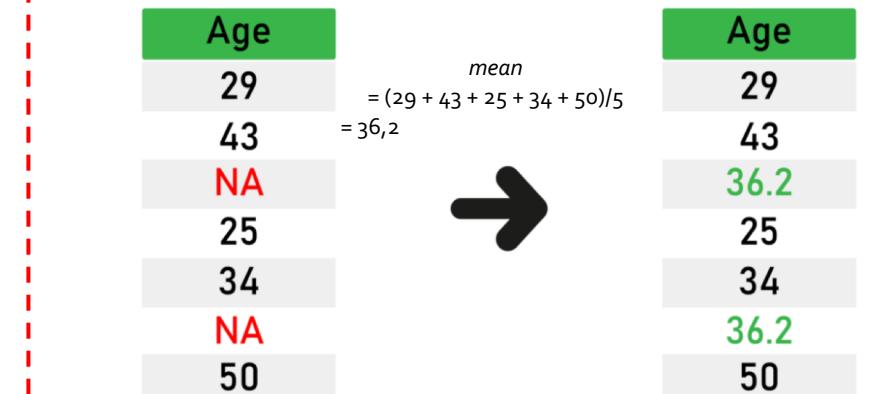
Data Preprocessing

Imputasi Mean atau Median

- Pro:
 - Mudah dan cepat.
 - Bekerja efektif untuk dataset numerik berukuran kecil.
 - Cocok untuk variabel numerik.
 - Cocok untuk data missing completely at random (MCAR).
 - Dapat digunakan dalam produksi (mis. dalam model deployment).
- Kontra:
 - Tidak memperhitungkan korelasi antar fitur, berfungsi pada tingkat kolom.
 - Kurang akurat.
 - Tidak memperhitungkan probabilitas/ketidakpastian.
 - Tidak cocok utk >5% missing data.
 - Mendistorsi variansi dan distribusi variabel asal/orijinal serta covariant variabel sisa data.



	col1	col2	col3	col4	col5		col1	col2	col3	col4	col5
0	2	5.0	3.0	6	NaN		2.0	5.0	3.0	6.0	7.0
1	9	NaN	9.0	0	7.0		9.0	11.0	9.0	0.0	7.0
2	19	17.0	NaN	9	NaN		19.0	17.0	6.0	9.0	7.0



Age	Age
29	29
43	43
NA	mean $= (29 + 43 + 25 + 34 + 50)/5$ = 36,2
25	36.2
34	25
NA	34
50	36.2
	50

Mean/Median Imputation: Mengisi nilai hilang dengan nilai rata-rata (untuk data numerik) atau median.

```
df['column_name'].fillna(df['column_name'].mean(), inplace=True)
```

Mode Imputation: Mengisi nilai hilang untuk data kategorikal dengan nilai mode.

```
df['column_name'].fillna(df['column_name'].mode()[0], inplace=True)
```

Data Preprocessing

Hands On

```
import pandas as pd
import numpy as np
```

Import pandas dan numpy

```
kolom = {'col1' : [2, 9, 19],
          'col2' : [5, np.nan, 17],
          'col3' : [3, 9, np.nan],
          'col4' : [6, 0, 9],
          'col5' : [np.nan, 7, np.nan]}
```

Masukkan data

```
data = pd.DataFrame(kolom)
```

pd.DataFrame() --> fungsi mengubah menjadi dataframe

```
data
```

	col1	col2	col3	col4	col5
0	2	5.0	3.0	6	NaN
1	9	NaN	9.0	0	7.0
2	19	17.0	NaN	9	NaN

Output

Data Preprocessing

Hands On

```
data.fillna(data.mean()) → Mengganti missing value  
dengan mean()
```

	col1	col2	col3	col4	col5
0	2	5.0	3.0	6	7.0
1	9	11.0	9.0	0	7.0
2	19	17.0	6.0	9	7.0

Output

```
umur = {'umur' : [29, 43, np.nan, 25, 34, np.nan, 50]} → Masukan  
Data
```

```
data = pd.DataFrame(umur) → Ubah ke dataframe
```

data

	umur
0	29.0
1	43.0
2	NaN
3	25.0
4	34.0
5	NaN
6	50.0

Output

Mengganti missing value dengan mean pada kolom umur

```
data.fillna(data.mean()) →
```

data.fillna() --> fungsi mengganti missing value

	umur
0	29.0
1	43.0
2	36.2
3	25.0
4	34.0
5	36.2
6	50.0

Output

Data Preprocessing

Data Cleansing (Pembersihan Data)

- Data cleaning atas data berantakan (messy data), seperti:
 - missing value,
 - format tidak konsisten
 - record tidak berbentuk baik (malformed record)
 - outlier yang berlebihan
- Lingkup Data Cleaning:
 - Membuang kolom-kolom tidak penting dalam suatu DataFrame
 - Mengubah indeks di DataFrame
 - Membersihkan kolom dengan metode .str()
 - Membersihkan semua dataset dengan fungsi DataFrame.applymap()
 - Merubah nama kolom sehingga kolom lebih mudah dikenali
 - Melewatkana baris-baris tidak penting dalam file CSV

```
df.drop(columns=to_drop, inplace=True)
```

inplace=True adalah perintah
menimpa kolom dan menyimpan
kolom/fitur yang dimanipulasi (dlm hal
ini di drop)

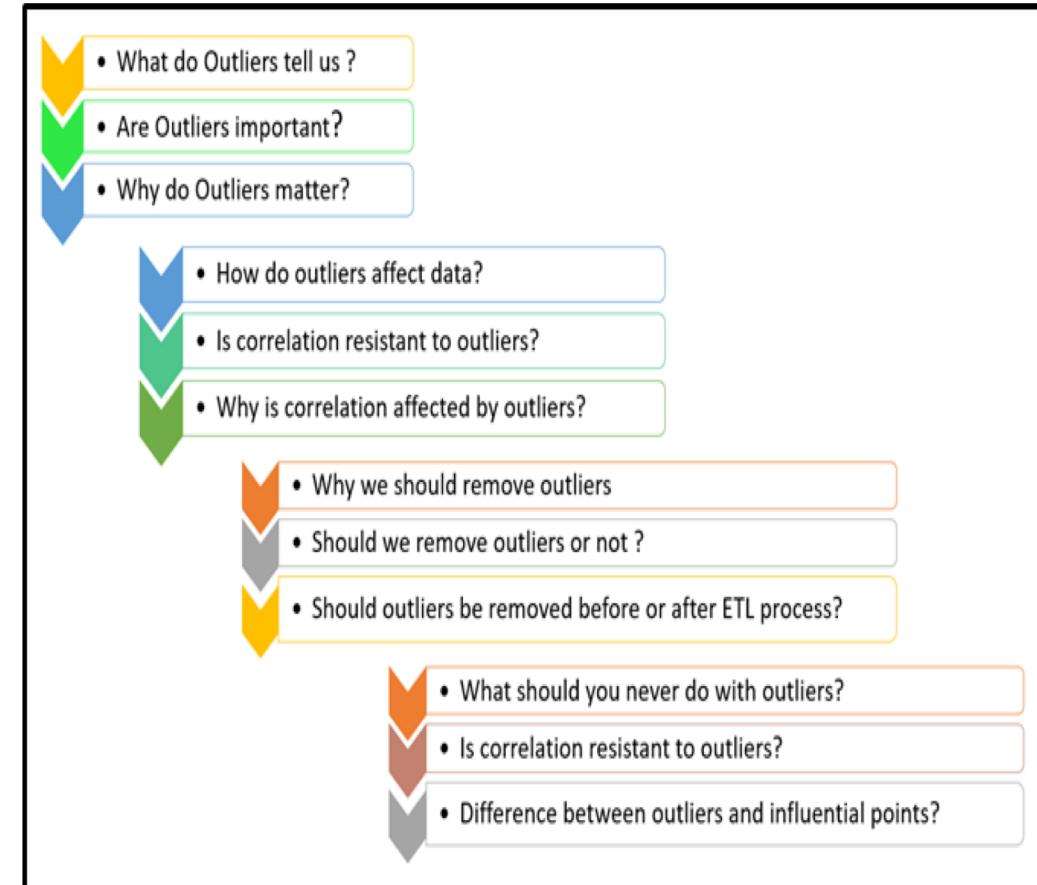
Data Preprocessing

Mendeteksi Outliers: Outliers dapat dilihat menggunakan visualisasi (seperti boxplot) atau menggunakan statistik (seperti IQR).

- Definisi *Outlier*:
 - Titik data yang sangat berbeda dari data lainnya.
 - Pengamatan yang menyimpang dari pola keseluruhan pada sampel.
- Penyebab:
 - Error percobaan, salah input, error instrumen, kesengajaan (untuk pengujian), error pemrosesan data, error sampling, kewajaran karena keanehan dalam data (bukan error).

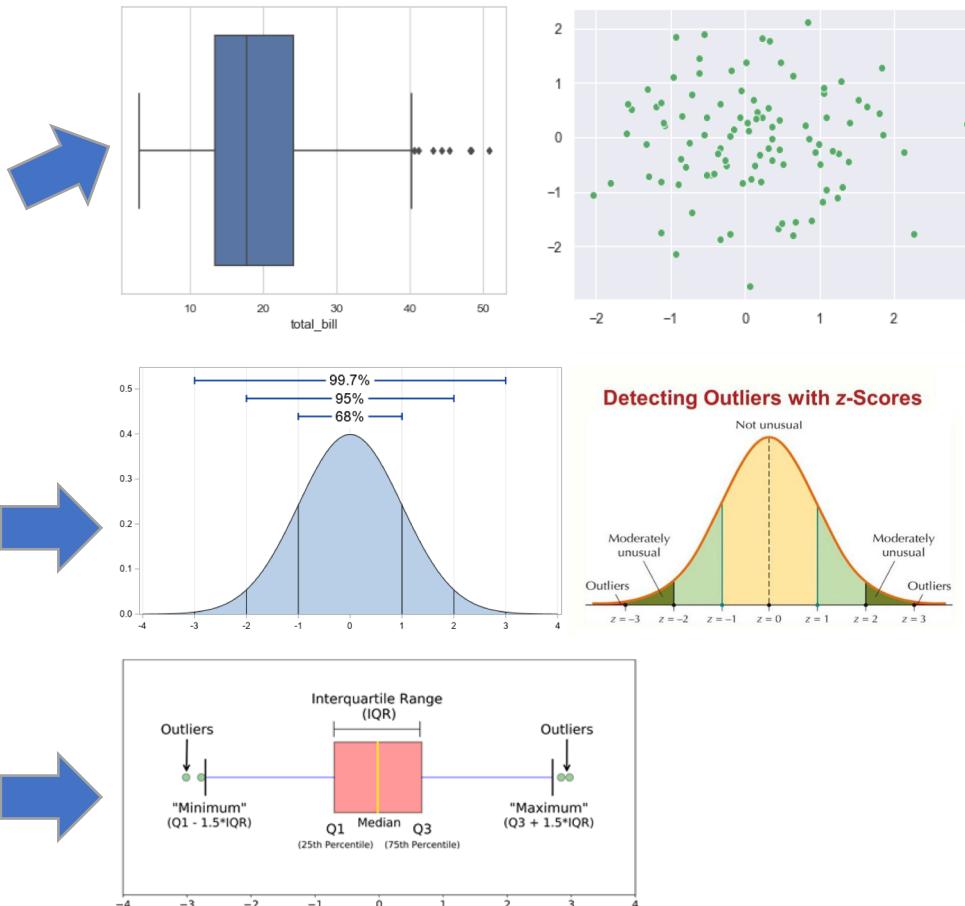
Teknik Mengatasi Outlier:

- Trimming
- Winsorizing
- Imputing
- Discretization
- Censoring
- Z-score
- Linear Regression Model



Data Preprocessing

- Visualiasi dgn Boxplot dan Scatterplot
 - Sebagian besar titik data terletak di tengah, tetapi ada satu titik yang jauh dari pengamatan lainnya; ini bisa menjadi outlier.
- Distribusi Normal
 - Dalam distribusi normal, sekitar 99,7% data berada dalam tiga standar deviasi dari mean.
 - Jika ada pengamatan yang lebih dari tiga kali standar deviasi, kemungkinan itu adalah outlier.
- Z-score
- Inter Quantile Range (IQR)



Imbalance dataset adalah kondisi di mana distribusi kelas dalam data tidak seimbang, yaitu jumlah sampel dalam satu kelas jauh lebih banyak dibandingkan kelas lainnya.

Contoh Kasus:

- **Fraud Detection**  → 98% transaksi normal, hanya 2% yang fraud.
- **Deteksi Penyakit Langka**  → 95% pasien sehat, hanya 5% yang sakit.
- **Spam Email Detection**  → 90% email normal, 10% spam.

Dalam kondisi seperti ini, model machine learning cenderung bias terhadap kelas mayoritas karena hanya sedikit belajar dari kelas minoritas. Akibatnya, meskipun akurasi tinggi, model bisa buruk dalam mendekripsi kelas minoritas.

Data Preprocessing

Up/Downsampling (Sampling Ulang)

Oversampling (Upsampling): Menambah jumlah data pada kelas minoritas dengan cara menduplikasi atau membuat data sintetis menggunakan teknik seperti **SMOTE** (Synthetic Minority Over-sampling Technique).

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
X_res, y_res = smote.fit_resample(X, y)
```

Undersampling (Downsampling): Mengurangi jumlah data pada kelas mayoritas..

```
from imblearn.under_sampling import RandomUnderSampler
rus = RandomUnderSampler()
X_res, y_res = rus.fit_resample(X, y)
```

Data Preprocessing

SMOTE (Synthetic Minority Over-sampling Technique)

```
from collections import Counter
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split

# Membuat dataset imbalance
X, y = make_classification(n_samples=1000, n_features=10, n_classes=2,
                           weights=[0.9, 0.1], random_state=42)

# Cek distribusi kelas
print("Distribusi Kelas Sebelum Penyeimbangan:", Counter(y))

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Distribusi Kelas Sebelum Penyeimbangan: Counter({0: 897, 1: 103})
```

```
from imblearn.over_sampling import SMOTE

# Oversampling dengan SMOTE
smote = SMOTE(sampling_strategy='auto', random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Cek distribusi kelas setelah SMOTE
print("Distribusi Kelas Setelah SMOTE:", Counter(y_train_resampled))

Distribusi Kelas Setelah SMOTE: Counter({1: 722, 0: 722})
```

SMOTE adalah teknik **oversampling** yang digunakan untuk menangani **imbalance dataset** dengan **membuat data sintetis** berdasarkan sampel yang sudah ada, bukan hanya menduplikasi data minoritas.

Data Preprocessing

Random Undersampling

```
from collections import Counter
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split

# Membuat dataset imbalance
X, y = make_classification(n_samples=1000, n_features=10, n_classes=2,
                           weights=[0.9, 0.1], random_state=42)

# Cek distribusi kelas
print("Distribusi Kelas Sebelum Penyeimbangan:", Counter(y))

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Distribusi Kelas Sebelum Penyeimbangan: Counter({0: 897, 1: 103})
```

```
from imblearn.under_sampling import RandomUnderSampler

# Undersampling
undersampler = RandomUnderSampler(sampling_strategy='auto', random_state=42)
X_train_resampled, y_train_resampled = undersampler.fit_resample(X_train, y_train)

# Cek distribusi kelas setelah undersampling
print("Distribusi Kelas Setelah Undersampling:", Counter(y_train_resampled))

Distribusi Kelas Setelah Undersampling: Counter({0: 78, 1: 78})
```

Random Undersampling (RUS) adalah teknik untuk menangani **imbalance dataset** dengan **mengurangi jumlah sampel dari kelas mayoritas secara acak**, sehingga jumlahnya lebih seimbang dengan kelas minoritas.

Data Preprocessing

Normalisasi dan Standarisasi Data --> teknik **penskalaan data** agar semua fitur memiliki skala yang seragam, sehingga model machine learning bekerja lebih optimal.

Min-Max Scaling: Mengubah data agar berada dalam rentang tertentu (biasanya 0-1).

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df_scaled = scaler.fit_transform(df[['column_name']])
```

Z-Score Standardization (Standarisasi): Mengubah data sehingga distribusinya memiliki mean 0 dan standar deviasi 1.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_standardized = scaler.fit_transform(df[['column_name']])
```

Data Preprocessing

Min-Max Scaling (Normalisasi)

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Contoh data
data = {'Feature1': [10, 20, 30, 40, 50],
        'Feature2': [5, 15, 25, 35, 45]}
df = pd.DataFrame(data)

# Inisialisasi MinMaxScaler
scaler = MinMaxScaler()

# Normalisasi data
df_normalized = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)

print("Data Normalisasi:\n", df_normalized)

Data Normalisasi:
 Feature1  Feature2
0      0.00      0.00
1      0.25      0.25
2      0.50      0.50
3      0.75      0.75
4      1.00      1.00
```

Normalisasi adalah proses penskalaan data ke dalam rentang tertentu, biasanya $[0,1]$ atau $[-1,1]$.

Teknik ini berguna ketika data memiliki distribusi yang tidak normal atau bervariasi dalam skala yang berbeda.

Data Preprocessing

Z-Score Scaling (Standarisasi)

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Contoh data
data = {'Feature1': [10, 20, 30, 40, 50],
        'Feature2': [5, 15, 25, 35, 45]}

# Inisialisasi StandardScaler
scaler = StandardScaler()

# Standarisasi data
df_standardized = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)

print("Data Standarisasi:\n", df_standardized)

Data Standarisasi:
   Feature1  Feature2
0 -1.414214 -1.414214
1 -0.707107 -0.707107
2  0.000000  0.000000
3  0.707107  0.707107
4  1.414214  1.414214
```

Standarisasi mengubah data sehingga memiliki rata-rata 0 dan standar deviasi 1.

Ini memastikan bahwa data memiliki distribusi yang lebih stabil dan sering digunakan untuk algoritma berbasis statistik.

Features Engineering

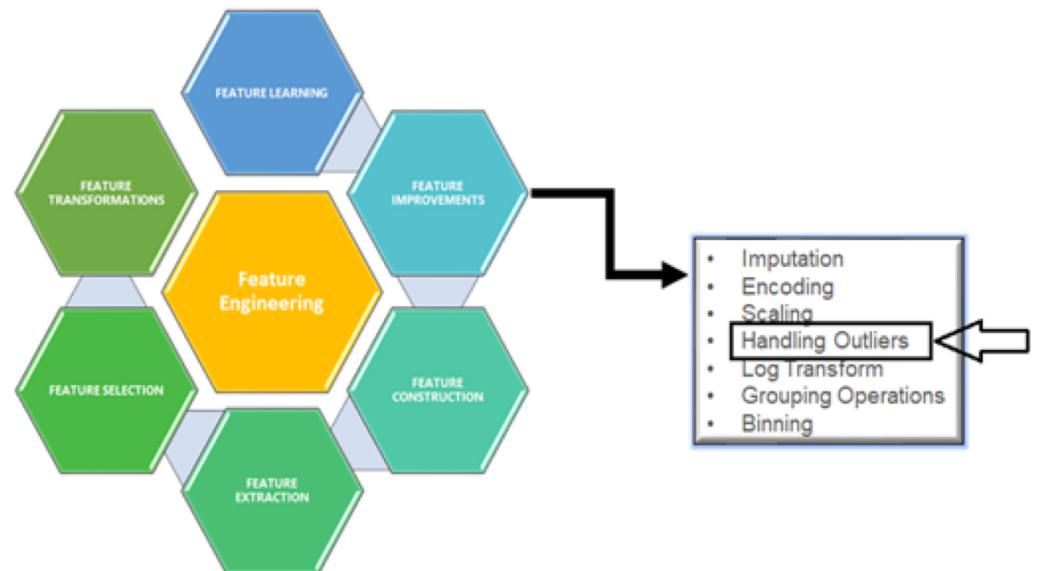
Rekayasa fitur adalah proses mengubah data mentah menjadi fitur yang lebih representatif agar dapat meningkatkan kinerja model machine learning. Proses ini mencakup pemilihan, transformasi, dan pembuatan fitur baru dari dataset.

Teknik Utama Feature Engineering

- 1. Scaling (Penskalaan)** → Menyamakan skala fitur (contoh: Min-Max Scaling, Standardization).
- 2. Transformation (Transformasi)** → Mengubah distribusi data (contoh: Log Transform, Binning).
- 3. Encoding Categorical Variables** → Mengubah data kategorikal menjadi numerik (contoh: One-Hot Encoding, Label Encoding).
- 4. Feature Extraction** → Membuat fitur baru dari fitur yang ada (contoh: usia dari tanggal lahir).
- 5. Feature Selection** → Memilih fitur yang paling relevan (contoh: korelasi, PCA, Lasso).
- 6. Feature Interaction** → Menggabungkan beberapa fitur untuk informasi tambahan (contoh: fitur baru = fitur1 × fitur2).
- 7. Handling Missing Values** → Mengisi nilai yang hilang (contoh: Mean Imputation, KNN Imputation).

Data Preprocessing

Rekayasa Fitur



Outlier

