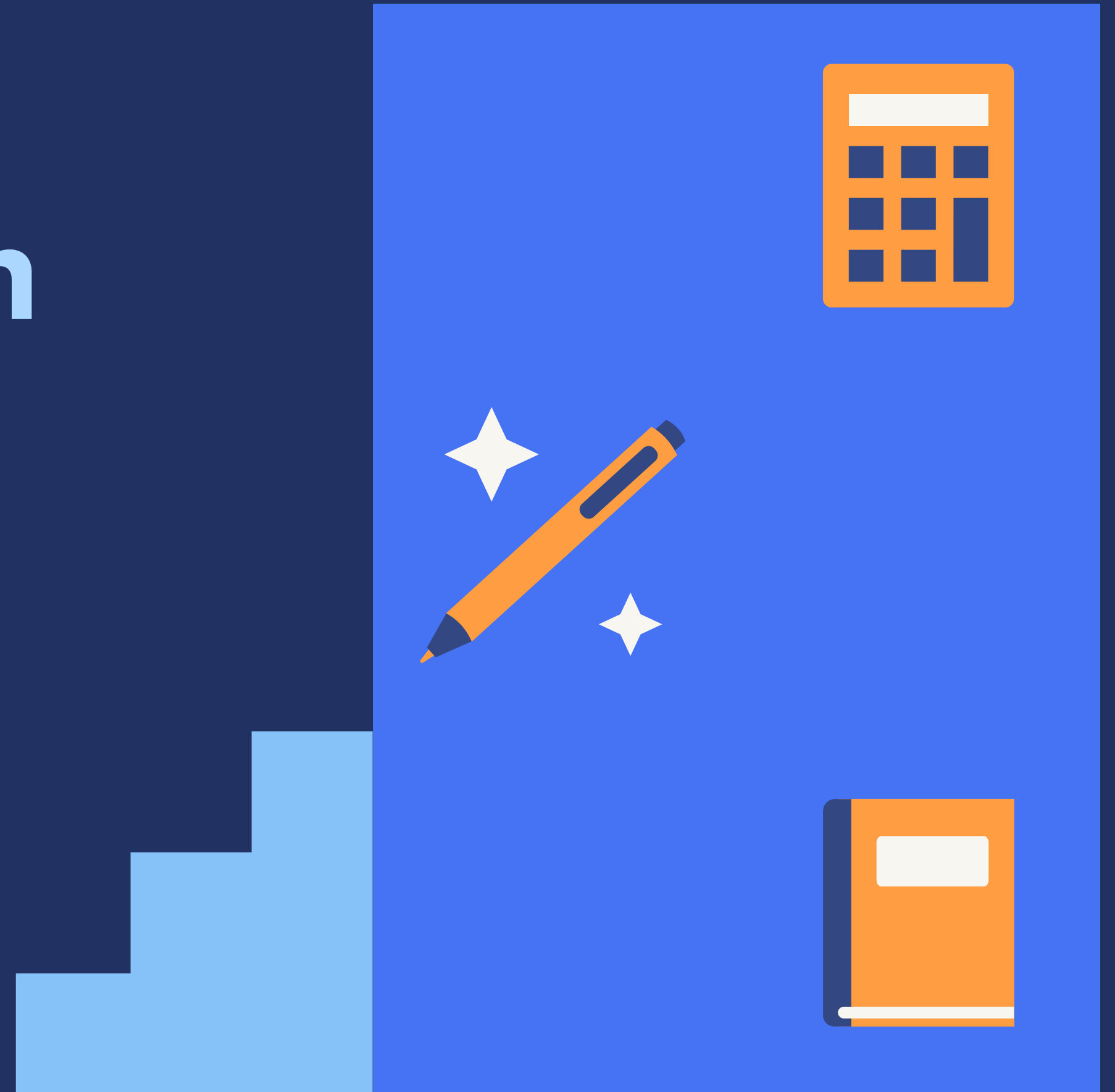


Pra-Pemrosesan Data

PERTEMUAN 3



Mengapa Data Diproses di Awal?

- Di lapangan, data yang dimiliki (*raw data*) biasanya kotor dan berantakan.
- Beberapa kasus yang sering dihadapi, antara lain:
 - **Tidak lengkap:** nilai-nilai atribut kosong (*missing data*) yang disebabkan karena responden tidak mau mengisi survey, adanya *human error* saat pengisian data, dan faktor lainnya.
Misal, pekerjaan=" " di mana seharusnya diisi misal "staff keuangan".
 - **Noisy:** memuat error atau memuat *outliers* (data yang secara nyata berbeda dengan data-data yang lain).
Misal, Salary="-10" di mana seharusnya salary tidak pernah negatif (yang mungkin negatif contohnya adalah *cashflow*)
 - **Tak-konsisten:** memuat perbedaan dalam kode atau nama
Contoh:
 - ulang tahun = "03/07/1997" harus konsisten *date/month/year*
 - rating sebelumnya "1,2,3", sekarang rating "A, B, C"
 - perbedaan antara duplikasi *record* (data yang sama muncul dua kali atau lebih)
- **Data yang baik akan menghasilkan hasil yang baik.**
- Tahapan *preprocessing* membantu di dalam memperbaiki presisi dan kinerja dalam analisis data dan mencegah kesalahan dalam prosesnya.

Mengapa Data Kotor?

- Ketidaklengkapan data datang dari:
 1. Nilai data tidak tersedia saat dikumpulkan
 2. Perbedaan pertimbangan waktu antara saat data dikumpulkan dan saat data dianalisa.
 3. Masalah manusia, hardware, dan software
- Noisy data datang dari proses data:
 1. Pengumpulan
 2. Pemasukan (*entry*)
 3. Transmisi
- Tidak konsistennya data datang dari:
 1. Sumber data yang berbeda
 2. Pelanggaran kebergantungan fungsional

Tujuan Pemrosesan Data

- Menghasilkan hasil *mining* yang berkualitas
- Data *warehouse* membutuhkan integrasi yang konsisten
- Data extraction, cleaning, and transformation merupakan salah satu tahapan untuk membangun gudang data

Tahapan Pra-Pemrosesan Data



Preprocessing terdiri dari beberapa aspek:

➤ **Data cleaning**

- Data kosong (*data imputation*)
- Meminimumkan Noise
- Mengatasi data yang tidak konsisten
- Mengatasi *outliers*

➤ **Data integration**

- Penggabungan dari beberapa sumber data seperti database, kubus data, atau file

➤ **Data transformation**

- Normalisasi dan agregasi data sehingga menjadi sama

➤ **Data Reduction**

- Pengurangan dimensi
- Pengurangan angka
- Kompresi data

➤ **Data Discretization**

- Bagian dari reduksi data
- Mengganti atribut numerik dengan atribut nominal. Termasuk juga pengurangan sejumlah nilai atribut kontinu dengan membagi rentang interval atribut.

Data Cleaning (Pembersihan Data)

Data di dunia nyata sering kali tidak lengkap, bermasalah, dan tidak konsisten. Banyak bagian data yang mungkin tidak relevan atau hilang. Pembersihan data dilakukan untuk menangani aspek ini.

Metode pembersihan data bertujuan untuk mengisi nilai yang hilang, menghaluskan noise sekaligus mengidentifikasi outlier, dan memperbaiki perbedaan data. Data yang tidak bersih dapat membingungkan data dan model. Oleh karena itu, menjalankan data melalui berbagai metode Pembersihan/Pembersihan Data merupakan langkah Pra-pemrosesan Data yang penting.

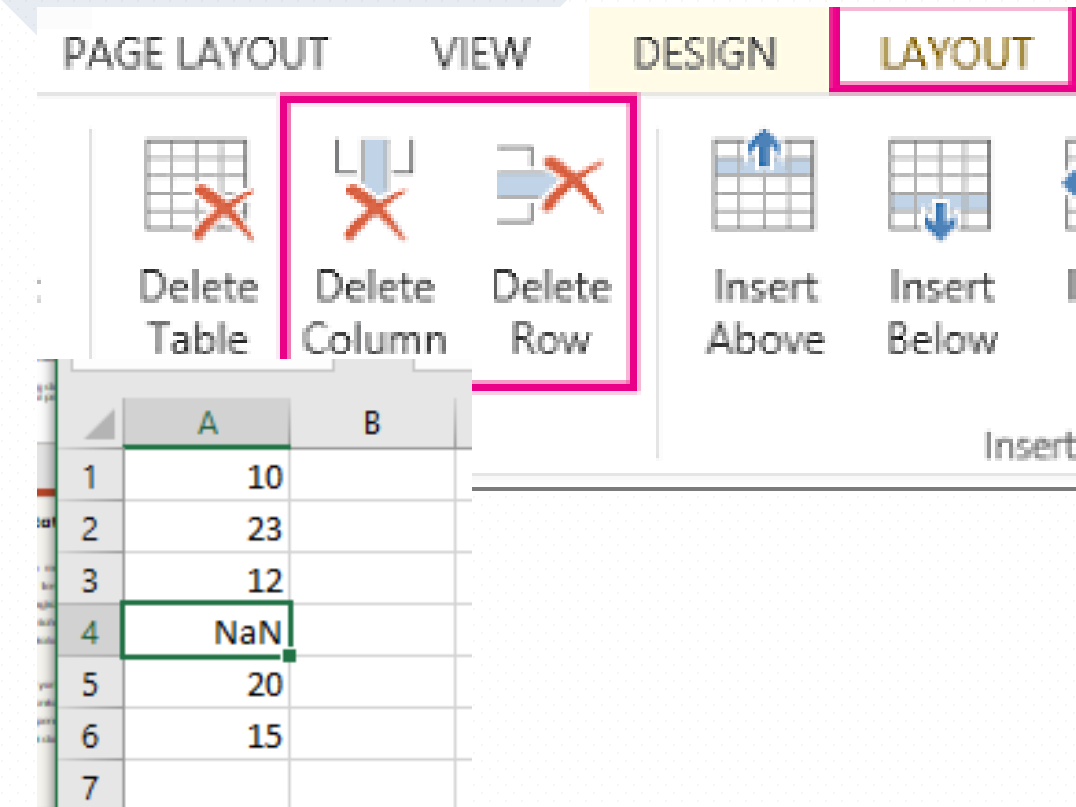


Missing Data Imputation

- Jika datasetnya lengkap (tidak ada nilai kosong), maka idealnya tidak perlu dilakukan *missing data imputation*.
- Tujuan dari *missing data imputation* (pengisian nilai kosong) adalah menghasilkan dataset yang lengkap, sehingga bisa digunakan untuk proses analisis lebih lanjut.
- Adanya nilai kosong di dataset menyebabkan beberapa masalah, antara lain:
 - Beberapa algoritma *machine learning* tidak bisa dijalankan jika masih ada data kosong.
 - Data kosong dapat mendistorsi distribusi dari variabel yang terpengaruh.
- Sebelum kita lakukan langkah-langkah saat menemukan data kosong, maka perlu diketahui penyebab mengapa ada data kosong di dataset kita.
- Dalam menentukan metode yang tepat untuk mengatasi nilai kosong, maka sangat penting untuk mengetahui bagaimana data tersebut didapatkan.
- Dengan mengetahui metode diduplikatnya data, kita bisa menilai secara objektif mengapa terdapat nilai kosong di dataset.
- Dengan demikian, bisa dipilih metode missing data imputation yang tepat sesuai kondisi yang sebenarnya terjadi di lapangan sebagai penyebab adanya missing data.

Metode Missing Data Imputation

- **Menghapus baris/kolom**: Jika baris/kolom memiliki nilai NaN maka tidak ada nilai pada baris/kolom tersebut, sehingga baris/kolom tersebut harus segera dihilangkan. Atau jika % baris/kolom sebagian besar hilang, katakanlah lebih dari 65%, maka dapat memilih untuk menghapus baris/kolom tersebut.
- **Memeriksa data duplikat**: Jika baris/kolom yang sama diulang maka dapat dihilangkan dengan mempertahankan data pertama, sehingga pada saat menjalankan algoritma pembelajaran mesin (Machine Learning), agar tidak bias pada objek data tertentu.



A	B	C
Data 1	Data 2	Apakah Data 2 Unik?
Amar Joni	Rika Wulandari	Unik
Syahrul	Retno Santi	Unik
Parjo	Akbar Joni	Unik
Ardi Bintara	Amar Joni	Duplikat
Angga Saputra	Rendra Saputra	Unik
Ratna Juwita	Permana	Duplikat
Lani Salina	Ratna Juwita	Duplikat
Joko Wardani	Andi Arsil	Unik
Permana	Kumbara	Unik

Metode Missing Data Imputation

- **Memperkirakan data yang hilang:** Jika hanya sebagian kecil dari nilai yang hilang, metode interpolasi dasar dapat digunakan untuk mengisi kekosongan tersebut. Namun, pendekatan paling umum untuk menangani data yang hilang adalah dengan mengisinya dengan nilai mean, median, atau mode fitur.
 - **Tipe data Numerik (angka)**
 - **Mengganti dengan nilai mean (jika *normal*) atau median (jika *skewed*)** → Perhitungan mean/median dihitung di *training set*, kemudian nilai mean/median diisikan untuk *training* dan *test set*.
 - Mengganti dengan nilai akhir dari distribusi (mirip dengan tahap kedua di atas) → Formula: **Mean \pm 3*SD** jika berdistribusi normal dan jika distribusi skewed maka **Q1 - IQR*3** (*lower limit*) atau **Q3 + IQR*3** (*upper limit*).
 - Menghilangkan baris-baris data yang hilang (dilakukan jika baris yang hilang adalah random dan kurang dari 5% total data)
 - **Tipe data Categorical (kategori)**
 - **Mengganti dengan nilai kategori yang sering muncul (modus / *most frequent*)** → diasumsikan terjadi karena MCAR
 - Mengganti dengan kategori baru, seperti 'kosong', '*missing*', dll
 - Menghilangkan baris-baris data yang hilang (dilakukan jika baris yang hilang adalah random dan kurang dari 5% total data)

Metode Missing Data Imputation

Program: Tipe data *Categorical* (kategori)

```
In [2]: #input library
import pandas as pd #mengubah dimensi data, membuat tabel, memeriksa data, membaca data, dsb
import numpy as np #memudahkan operasi perhitungan tipe data numerik seperti penjumlahan, perkalian, pengurangan, dsb
from sklearn.impute import SimpleImputer #memanggil fungsi SimpleImpuler yang terdapat pada library
```

```
In [3]: # Upload Data in a DataFrame
df = pd.read_csv("data latihan imputation.csv")
print(df)
```

	Nama	Usia	IPK	Lulus_tepat_waktu
0	Rina	23	3.9	Ya
1	Ilham	28	NaN	Ya
2	Uni	25	2.7	Tidak
3	Diah	24	3.1	Ya
4	Fitri	22	3.6	Ya
5	Andre	23	NaN	Tidak
6	Ma'ruf	21	3.3	Ya
7	Affan	25	2.5	Tidak
8	Nu'man	24	2.9	Tidak
9	Khalif	19	3.4	NaN

Metode Missing Data Imputation

Program: Tipe data *Categorical* (kategori)

```
# Imputer object using the mean strategy and
# missing_values type for imputation
imputer = SimpleImputer(missing_values = np.NaN,
                        strategy = 'mean')

# Fitting the data to the imputer object
imputer = imputer.fit(data['IPK'].values.reshape(-1,1))

# Imputing the data
data = imputer.transform(data['IPK'].values.reshape(-1,1))

print(data)
```

Metode Missing Data Imputation

Program: Type data *Categorical* (kategori)

```
In [36]: df.isna().sum() #menampilkan fitur yang memiliki missing value
```

```
Out[36]: Nama                0  
Usia                0  
IPK                2  
Lulus_tepat_waktu    1  
dtype: int64
```

```
In [4]: df['Lulus_tepat_waktu'].fillna(df['Lulus_tepat_waktu'].value_counts().idxmax(),inplace=True)  
df
```

Out[4]:

	Nama	Usia	IPK	Lulus_tepat_waktu
0	Rina	23	3.9	Ya
1	Ilham	28	NaN	Ya
2	Uni	25	2.7	Tidak
3	Diah	24	3.1	Ya
4	Fitri	22	3.6	Ya
5	Andre	23	NaN	Tidak
6	Ma'ruf	21	3.3	Ya
7	Affan	25	2.5	Tidak
8	Nu'man	24	2.9	Tidak
9	Khalif	19	3.4	Ya

Metode Missing Data Imputation

Program: Tipe data Numerik

```
In [36]: # Hitung nilai rata-rata
rata_rata_IPK = df['IPK'].mean()

# Imputasi missing value dengan mean
df['IPK'] = df['IPK'].fillna(rata_rata_IPK)

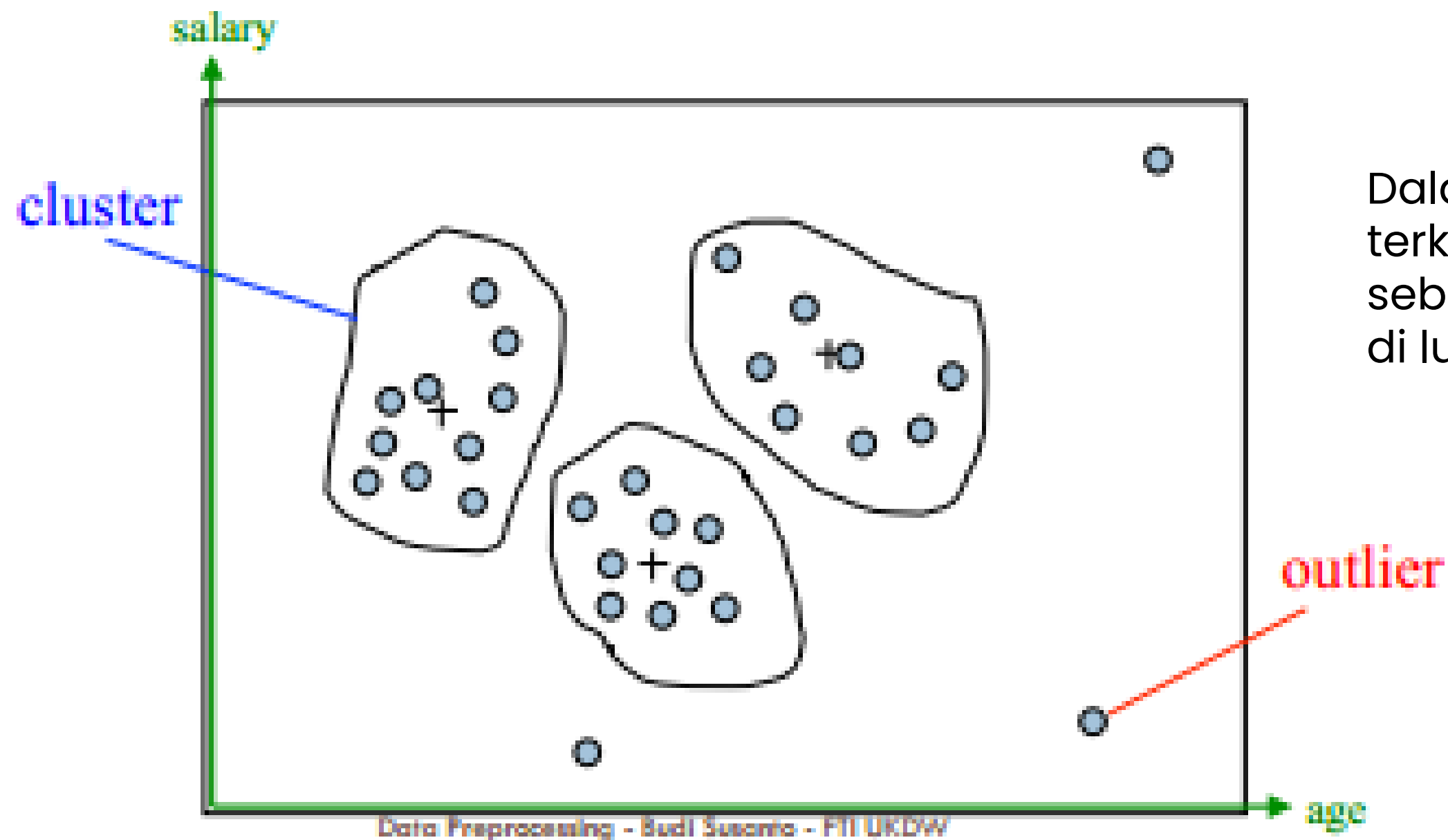
# Cetak dataframe
print(df)
```

	Nama	Usia	IPK	Lulus_tepat_waktu
0	Rina	23	3.900	Ya
1	Ilham	28	3.175	Ya
2	Uni	25	2.700	Tidak
3	Diah	24	3.100	Ya
4	Fitri	22	3.600	Ya
5	Andre	23	3.175	Tidak
6	Ma'ruf	21	3.300	Ya
7	Affan	25	2.500	Tidak
8	Nu'man	24	2.900	Tidak
9	Khalif	19	3.400	Ya

Noisy Data

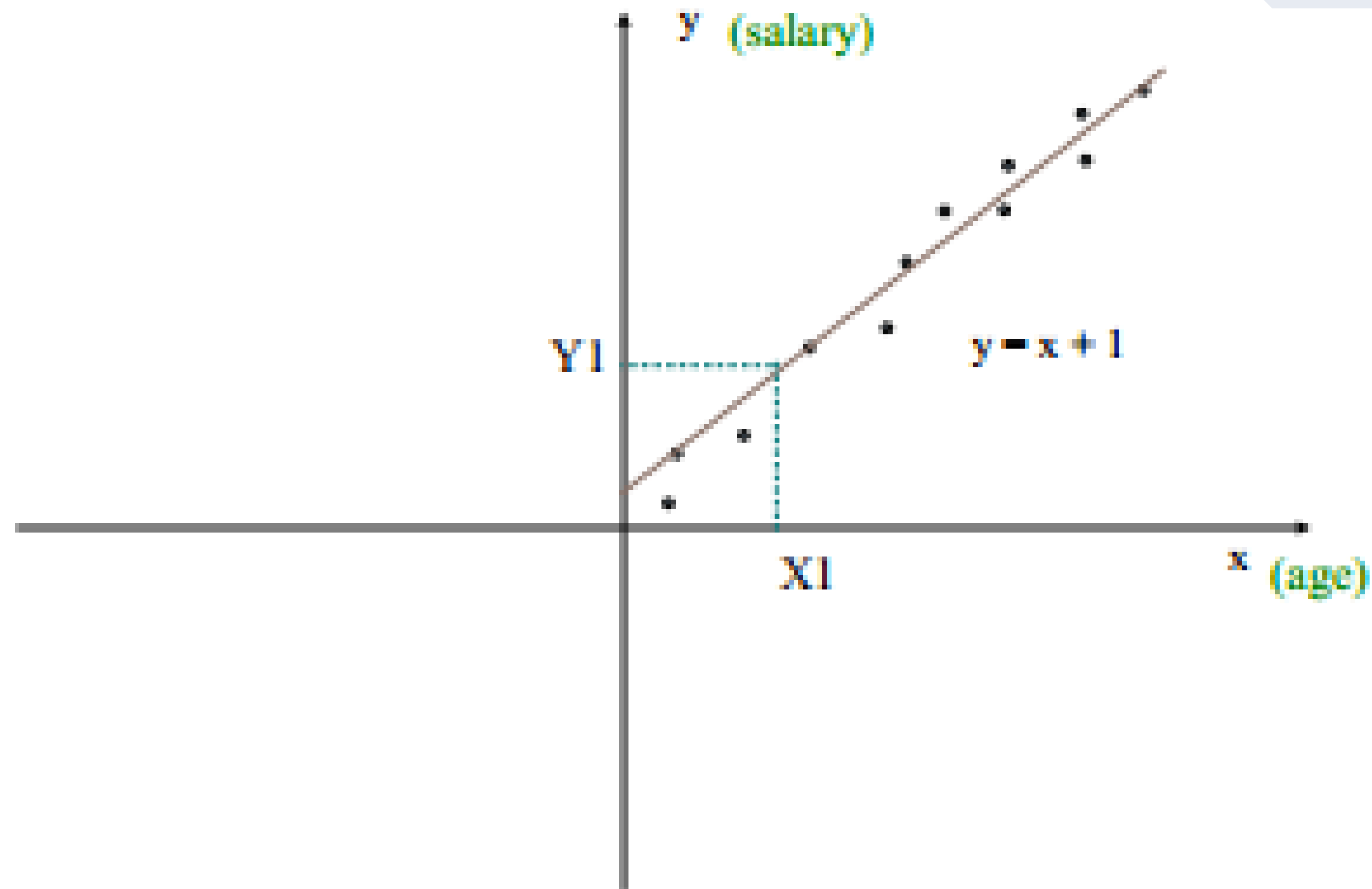
Cara mengetahui *outlier* : Klastering (*clustering*), Regresi Linear

Mendeteksi Outlier dengan Clustering



Dalam metode Clustering, data terkait dikelompokkan dalam sebuah cluster. Outlier berada di luar kelompok.

Mendeteksi Outlier dengan Regresi



Mengatasi Outlier

- **Outliers** adalah *data points* (nilai baris dalam *feature*/kolom tertentu) yang berbeda dari baris-baris lainnya dalam *feature*/kolom yang sama dan diduga berasal dari sumber/mekanisme data yang berbeda.
- Tidak selamanya outlier harus dibuang. Ini semua ditentukan oleh pengamatan terhadap data dan konteks masalah yang dihadapi.
- Aturan dasarnya, jika outliers disebabkan karena kesalahan (*error*) karena metode pengukuran, maka sebaiknya outliers tersebut dibuang terlebih dahulu sebelum datanya dianalisis lebih lanjut.

Mengatasi Outlier (2)

- **Trimming**
 - Langsung menghapus data point (baris) dari dataset (prosesnya cepat tapi harus hati-hati)
- **Missing data**
 - Memberlakukan outlier sebagai *missing data* dan lakukan *missing data imputation*
- **Discretization**
 - Mentransformasi data kontinu menjadi data diskrit dengan rentang tertentu. Dengan demikian, data outlier akan menjadi bagian dari data urutan di belakangnya.
 - Misal: Usia 16–70 th diganti menjadi → 16–20 th, 21–25 th, 30–35 th, dst
- **Censoring**
 - Jika lebih besar atau lebih kecil dari batas tertentu, maka ganti dengan nilai yang ditentukan.

Data Transformation

Tujuan diadakan transformasi data agar data lebih efisien dalam proses data mining dan mungkin juga agar pola yang dihasilkan lebih mudah dipahami.

Hal-hal yang termasuk transformasi data:

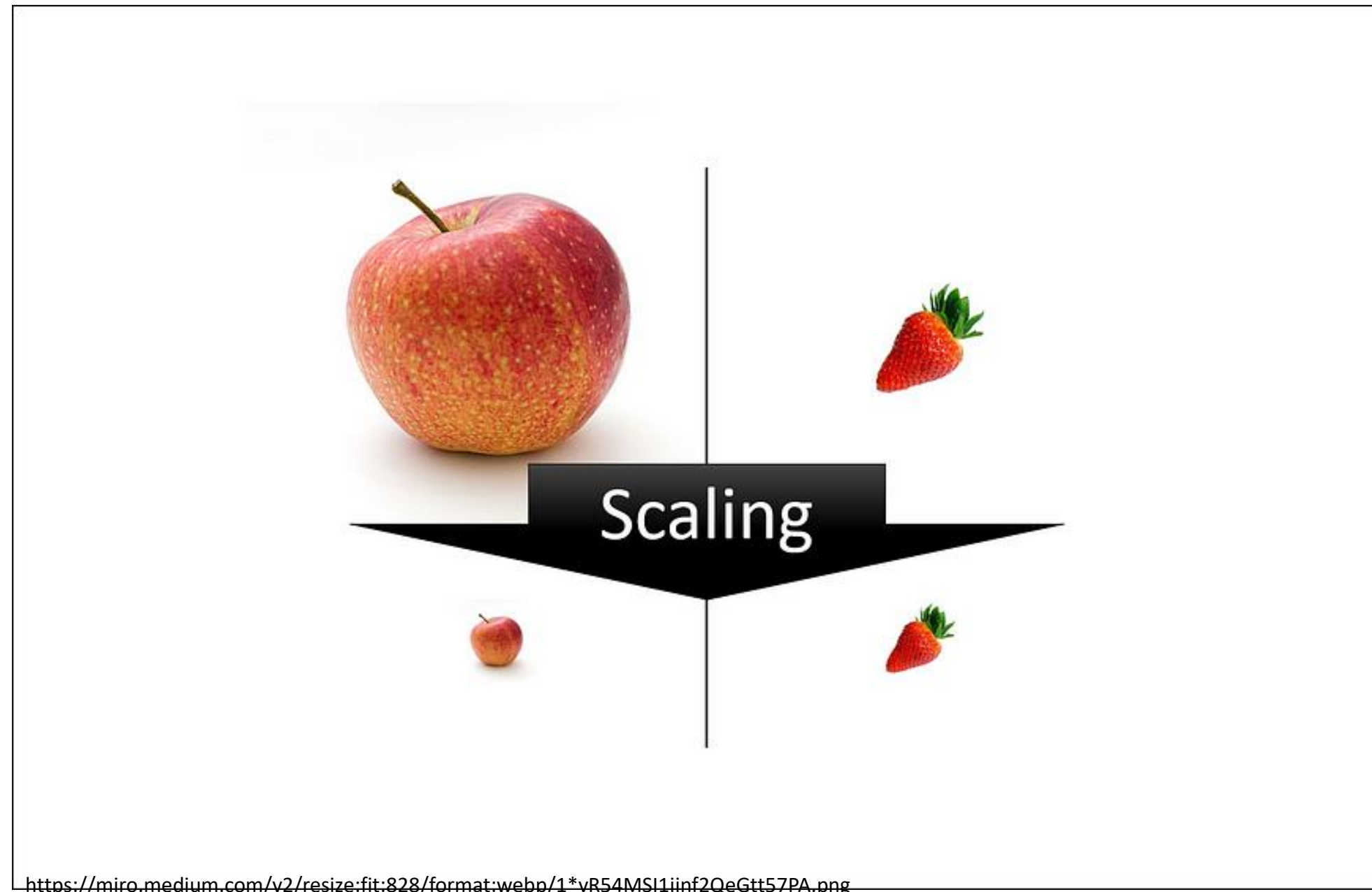
Smoothing : menghapus noise dari data

Aggregation : ringkasan, konstruksi data cube

Normalization : min-max, Z-Score, decimal scaling

Data Transformation

Normalization atau dikenal juga penskalaan fitur (*feature scaling*) dan *standardisasi*



Data Transformation

Normalization

a. Min-max normalization: menghasilkan [new_min,new_max]

Tujuan: mengubah data kedalam rentang nilai 0 hingga 1.

Rumus :

$$x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}}$$

Dengan:

x: fitur yang akan dihitung

x_{new} : ukuran fitur baru yang akan dihasilkan

x_{old} : ukuran fitur lama yang dimiliki

x_{min} : ukuran minimal yang dimiliki oleh fitur

x_{max} : ukuran maksimal yang dimiliki oleh fitur

Data Transformation

Contoh:

Tabel 1.1 fitur gaji dan pengalaman	
Gaji	Pengalaman
5.000.000	1
7.000.000	3
9.000.000	5
12.000.000	7
15.000.000	10

Lakukan normalisasi data di tabel 1.1 dengan menggunakan teknik minmax.

Data Transformation

Jawaban:

Dengan menggunakan rumus min-max normalisasi

$$\begin{aligned} x_1 &= \frac{1 - 1}{10 - 1} = 0 \\ x_2 &= \frac{3 - 1}{10 - 1} = 0.22 \\ x_3 &= \frac{5 - 1}{10 - 1} = 0.44 \\ x_4 &= \frac{7 - 1}{10 - 1} = 0.67 \\ x_5 &= \frac{10 - 1}{10 - 1} = 1 \end{aligned}$$

Tabel 1.2 fitur pengalaman yang sudah dinormalisasikan

Pengalaman	Pengalaman yang dinormalisasikan
1	0
3	0.22
5	0.44
7	0.67
10	1

Data Transformation

Program:

Dengan menggunakan rumus min-max normalisasi pada Python

```
In [24]: #input library
import pandas as pd #mengubah dimensi data, membuat tabel, memeriksa data, membaca data, dsb
from sklearn.preprocessing import MinMaxScaler #melakukan normalisasi data menggunakan MinMaxScaler
```

```
In [27]: # Upload Data in a DataFrame
data = pd.read_csv("data latihan pertemuan 3.csv")
print(data)
```

	Gaji	Pengalaman
0	5000000	1
1	7000000	3
2	9000000	5
3	12000000	7
4	15000000	10

```
In [29]: scaler = MinMaxScaler()
scaled = scaler.fit_transform(data[['Pengalaman']])
print(scaled)
```

```
[[0.      ]
 [0.22222222]
 [0.44444444]
 [0.66666667]
 [1.      ]]
```

Data Transformation

Standardization (Z-score normalization)

Teknik normalisasi data yang menggunakan nilai rata-rata dan standar deviasi untuk menghitung nilai baru dari setiap titik data. Tujuan: untuk membuat distribusi data menjadi memiliki mean 0 dan standar deviasi 1.

$$z = \frac{x - \mu}{\sigma}$$

Dengan:

z : ukuran fitur yang sudah distandarisasi

x : fitur yang akan dihitung

μ : rata – rata

σ : standar deviasi

Data Transformation

Jawaban:

Dengan menggunakan rumus *Z-score normalization*

$$\mu = \frac{\sum x_i}{N} = \frac{1 + 3 + 5 + 7 + 10}{5} = \frac{26}{5} = 5.2$$
$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$
$$= \sqrt{\frac{(1 - 5.2)^2 + (3 - 5.2)^2 + (5 - 5.2)^2 + (7 - 5.2)^2 + (10 - 5.2)^2}{5}}$$
$$= 3.12$$

$$z_1 = \frac{1 - 5.2}{3.12} = -1,34439$$
$$z_2 = \frac{3 - 5.2}{3.12} = -0.7042$$
$$z_3 = \frac{5 - 5.2}{3.12} = -0.06402$$
$$z_4 = \frac{7 - 5.2}{3.12} = 0.576166$$
$$z_5 = \frac{10 - 5.2}{3.12} = 1,536443$$

Tabel 1.3 fitur pengalaman yang sudah dinormalisasikan

Pengalaman	Pengalaman yang dinormalisasikan
1	-1,34439
3	-0,7042
5	-0,06402
7	0,576166
10	1,536443

Data Transformation

Program:

Dengan menggunakan rumus StandardScaler pada Python

```
In [ ]: #Standardisasi Scaler
```


```
In [30]: #input library
import pandas as pd #mengubah dimensi data, membuat tabel, memeriksa data, membaca data, dsb
from sklearn.preprocessing import StandardScaler #melakukan normalisasi data menggunakan StandardScaler
```

```
In [31]: scaler = StandardScaler()
scaled = scaler.fit_transform(data[['Pengalaman']])
print(scaled)
```

```
[[ -1.34438724]
 [ -0.70420284]
 [ -0.06401844]
 [  0.57616596]
 [  1.53644256]]
```


Latihan

1. Download data train.csv pada <https://www.kaggle.com/competitions/titanic/data> kemudian lakukan langkah-langkah pra-pemrosesan (data cleaning dan data transformation) dengan menggunakan python.



TERIMA KASIH