

Data Preprocessing and Cleaning

Sesi 2- Bootcamp Machine Learning
and AI for Beginner



Outline

- Introduction to Data Preprocessing
- Basic Cleaning
- Outlier Handling
- Data Transformation
- Feature Encoding

Introduction to Data Preprocessing

**Menurutmu,
Kenapa data perlu dilakukan
preprocessing sebelum dianalisa
lebih lanjut?**

Data Preprocessing

Pentingnya melakukan preprocessing:

- Membersihkan data kotor
- Menyederhanakan data yang kompleks
- Mengidentifikasi dan mengatasi outlier
- Mengatasi ketidakseimbangan data
- Menormalisasi data
- Mengurangi dimensi

Data Cleaning

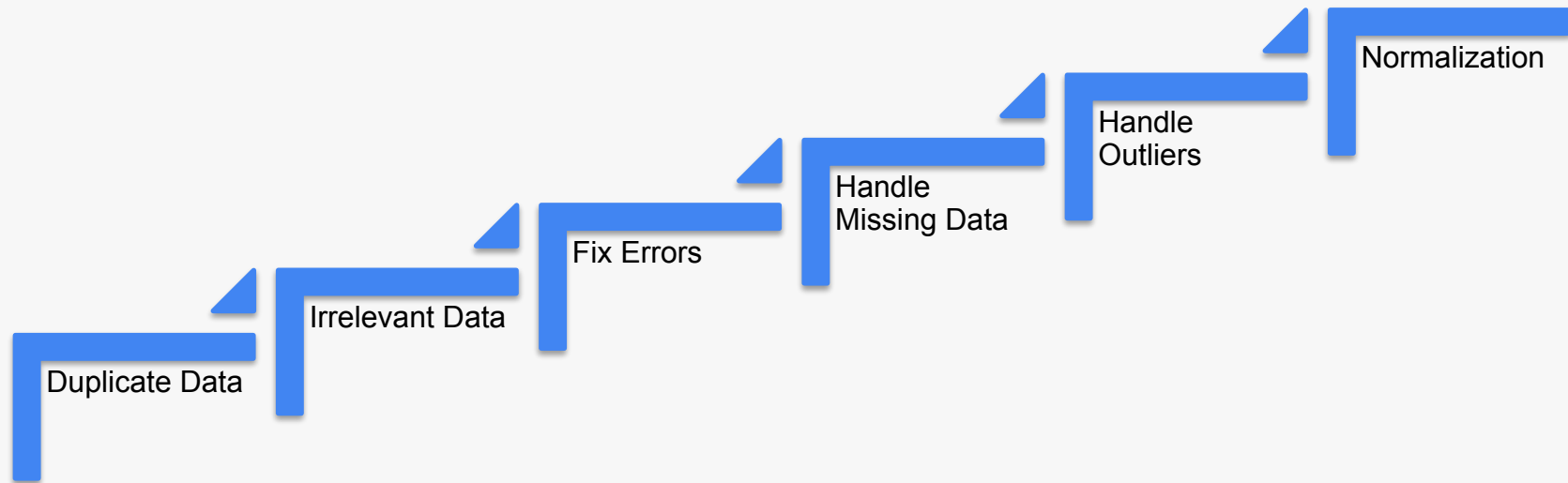
Data cleaning adalah suatu prosedur untuk memastikan kebenaran, konsistensi, dan kegunaan suatu data yang ada dalam dataset agar analisis didasarkan pada data yang akurat dan terpercaya.

Karena data di dunia nyata sangat kotor!

Beberapa penyebab ketidakbersihan pada data:

- User asal mengisi data
- Jenis input yang tidak wajib
- Ketidaksesuaian format data
- Tidak konsisten dalam pengumpulan data
- Kesalahan implementasi *tracker* data/ *engineering* mistakes
- *Scammer, abuser*
- Keterbatasan responden/sumber data

General Steps



Basic Cleaning

Duplicate Values

Keberadaan beberapa data yang identik atau mirip dalam dataset. Hal ini bisa terjadi karena kesalahan manusia dalam pengumpulan atau pemrosesan data.

Beberapa metode umum untuk mencari data yang duplikat:

- Berdasarkan atribut tertentu. Misalnya mengecek berdasarkan “ID Number” yang seharusnya unik.
- Berdasarkan semua atribut. Misalnya tidak ada atribut yang secara eksplisit harus unik.
- Berdasarkan subset atribut. Misalnya mengecek berdasarkan kombinasi “Nama”, “Alamat” dan “Nomor Telepon”.

Sebelum menghapus atau menggabungkan duplikat, penting untuk memahami konteks data.

Memeriksa Duplicate Data

Untuk melihat total data yang duplicate secara keseluruhan:

```
□ df.duplicated().sum()
```

Untuk melihat total data yang duplicate berdasarkan atribut tertentu:

```
□ df.duplicated(["kolom1", "kolom2"]).sum()
```

Jika ingin drop duplicate:

```
□ df.drop_duplicates()
```

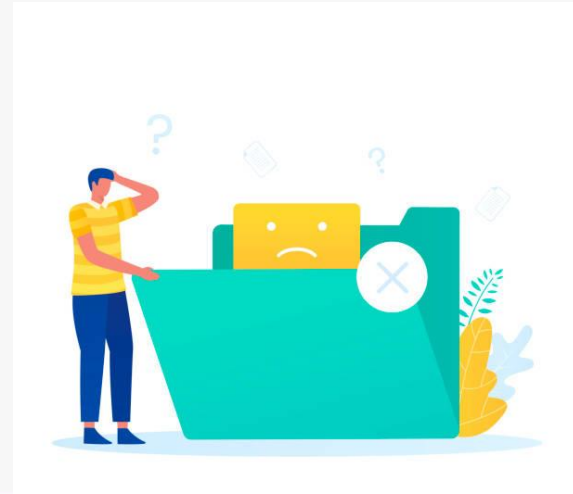
Missing Value

Pengertian : *Missing value* adalah peristiwa **hilang atau tidak terbacanya data**. Biasanya, data-data yang tidak dapat terdeteksi akan disimbolkan dengan “NaN”, NULL atau bisa juga cell kosong

Tujuan : *Missing value* yang tidak diatasi dapat menyebabkan kesalahan hasil analisis dan kesimpulan yang tidak sesuai dengan yang diharapkan. Selain itu, ada **beberapa mesin algoritma yang tidak memperbolehkan adanya *missing value*** dalam sebuah data.

Cara menangani *missing value* yang biasa dilakukan adalah sebagai berikut :

- *Leave as it is* (biarkan saja)
- *Drop Row* yang mengandung *missing value*
- *Imputation Process*



Memeriksa Missing Value

Untuk memeriksa missing value pada sebuah data dapat menggunakan method `.isnull()`

```
# Import library yang dibutuhkan
import pandas as pd

# Proses ekstraksi data dari sumber berformat excel
data_transaction = pd.read_excel('/content/Transaction Customer Telco.xlsx')

# Periksa banyaknya missing value pada tiap kolom
check_null_sum = data_transaction.isnull().sum()

# Tampilkan hasilnya
display(check_null_sum)
```

output :

Customer ID	5
Offer	0
Phone Service	0
Multiple Lines	0
Internet Service	0
Internet Type	0
Online Security	0
Online Backup	0
Device Protection Plan	0
Premium Tech Support	0
Streaming TV	0
Streaming Movies	0
Streaming Music	0
Unlimited Data	0
Contract	0
Paperless Billing	0
Payment Method	0
Monthly Charge	6
Total Charges	0
Total Refunds	0
Total Extra Data Charges	0
Total Long Distance Charges	0
Total Revenue	0
Updated At	0

dtype: int64

```
# Import library yang dibutuhkan
import pandas as pd

# Proses ekstraksi data dari sumber berformat excel
data_transaction = pd.read_excel('/content/Transaction Customer Telco.xlsx')

# Periksa kondisi missing value pada data
check_null = data_transaction.isnull().any()

# Tampilkan hasil
display(check_null)
```

output :

Customer ID	True
Offer	False
Phone Service	False
Multiple Lines	False
Internet Service	False
Internet Type	False
Online Security	False
Online Backup	False
Device Protection Plan	False
Premium Tech Support	False
Streaming TV	False
Streaming Movies	False
Streaming Music	False
Unlimited Data	False
Contract	False
Paperless Billing	False
Payment Method	False
Monthly Charge	True
Total Charges	False
Total Refunds	False
Total Extra Data Charges	False
Total Long Distance Charges	False
Total Revenue	False
Updated At	False

dtype: bool

#1 Handling Missing Value

1. Leave As It Is

Membiarkan saja missing value pada data juga mungkin dilakukan. Biasanya data yang kosong **pada kolom yang tidak begitu dibutuhkan analisa lebih lanjut** akan dibiarkan kosong begitu saja.

Contoh : Missing value pada kolom alamat customer, deskripsi produk atau kolom kolom lain yang sifatnya kolom tambahan. Ignore saja kolomnya altogether.

2. Drop Missing Value

Menghapus satu atau lebih row jika terdapat missing value juga mungkin dilakukan. Biasanya saat kolom yang sebagai *mandatory* proses bisnis bernilai NULL atau kosong.

Contoh : Kolom transaction_id pada tabel transaksi merupakan kolom wajib yang biasanya di generate otomatis oleh system. Namun jika ada data pada kolom ini hilang maka biasanya satu atau lebih row perlu dihapus untuk mempertahankan kevalidan data (dianggap *data corrupt*)

2. Drop Missing Value

Pada pandas drop missing value dapat menggunakan sintaks berikut :

DataFrame.dropna(subset, how)

```
# Import library yang dibutuhkan
import pandas as pd

# Proses ekstraksi data dari sumber berformat excel
data_transaction = pd.read_excel('/content/Transaction Customer Telco.xlsx')

# Proses drop row yang memiliki missing value pada subset tertentu
dropna_data_trx = data_transaction.dropna(subset = ['Customer ID'], how = 'any')

# Periksa banyak row sebelum dan sesudah di drop
banyakrow_sebelum_drop = data_transaction.shape[0]
banyakrow_setelah_drop = dropna_data_trx.shape[0]

# Tampilkan hasilnya
print(f'Banyak row sebelum di drop = {banyakrow_sebelum_drop} dan
      banyak row setelah di drop = {banyakrow_setelah_drop}')

output : Banyak row sebelum di drop = 7050 dan banyak row setelah di drop = 7045
```

#2 Handling Missing Value

3. Imputation

Imputation atau imputasi adalah proses **mengisi missing value dengan nilai tertentu**.

Biasanya diisi dengan rataan populasi data atau median (kolom numerik), dan dengan modus (kolom kategorik)

Perhatikan contoh berikut

X
4
4
6
7
9

Data dengan *missing value*

Hitung rataan data

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n x_i = \frac{1}{5} (4 + 4 + 6 + 7 + 9)$$

$$\bar{X} = \frac{1}{5} (30) = 6$$

X
4
4
6
6
7
9
6

Data setelah proses imputasi

Hitung Kembali rataan data

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n x_i = \frac{1}{7} (4 + 4 + 6 + 6 + 7 + 9 + 6)$$

$$\bar{X} = \frac{1}{7} (42) = 6$$

Setelah dilakukan proses imputasi maka metrics rataan data tidak berubah

3. Imputation

Pada pandas imputasi missing value dapat menggunakan sintaks berikut :

`Series.fillna(value)` atau `DataFrame.fillna(value)`

```
# Import library yang dibutuhkan
import pandas as pd

# Proses ekstraksi data dari sumber berformat excel
data_transaction = pd.read_excel('/content/Transaction Customer Telco.xlsx')

# Proses menghitung rata - rata

# Hitung rata-rata data pada kolom Monthly Charge
rataan_monthlycharge = data_transaction['Monthly Charge'].mean()

# Proses imputasi dengan .fillna()
data_transaction['Monthly Charge'] = data_transaction['Monthly harge'].fillna(rataan_monthlycharge)
```


3. Imputation

Pada proses ini, missing value dapat diisikan oleh perhitungan berikut

- Rataan : Gunakan sintaks `value = Series.mean()`
- Median : Gunakan sintaks `value = Series.median()`
- Modus : Gunakan sintaks `value = Series.mode()`

Setelah dilakukan perhitungan value barulah proses imputasi dilakukan dengan method **`.fillna(value)`**

Jika dilakukan pada DataFrame atau **`DataFrame.fillna()`** maka semua cell yang kosong akan diisi dengan value tersebut namun jika pada Series atau **`Series.fillna()`** maka hanya kolom tertentu yang akan dilakukan proses imputasi.

Rename Columns

Terkadang nama dari kolom tidak sesuai dan ingin diubah penulisannya agar lebih mudah dipahami.

	iD_	Name	G
0	1	Ani	P
1	2	Budi	L
2	3	Cahyo	L
3	4	Dodi	L
4	5	Ema	P
5	6	Firman	L

Rename Columns

Rename selective columns

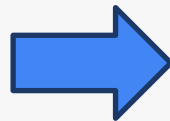
Format :

```
df.rename(columns={'old_name', 'new_name'})
```

Example :

```
df.rename(columns={'iD_', 'ID'})
```

	iD_	Name	G
0	1	Ani	P
1	2	Budi	L
2	3	Cahyo	L
3	4	Dodi	L
4	5	Ema	P
5	6	Firman	L



	ID	Name	G
0	1	Ani	P
1	2	Budi	L
2	3	Cahyo	L
3	4	Dodi	L
4	5	Ema	P
5	6	Firman	L

Rename Columns

Rename all columns

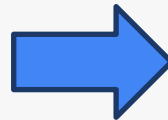
Format :

```
df.columns = ['a', 'b', 'c']
```

Example :

```
df.columns = ['ID', 'Nama', 'Gender']
```

	iD_	Name	G
0	1	Ani	P
1	2	Budi	L
2	3	Cahyo	L
3	4	Dodi	L
4	5	Ema	P
5	6	Firman	L



	ID	Nama	Gender
0	1	Ani	P
1	2	Budi	L
2	3	Cahyo	L
3	4	Dodi	L
4	5	Ema	P
5	6	Firman	L

Drop Columns

Terkadang kita tidak butuh suatu kolom untuk analisa lebih lanjut

Format :

```
df.drop('kolom', axis=1)
```

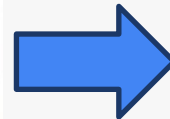
```
df.drop(columns='kolom name')
```

Axis 1 berarti drop berdasarkan kolom

Example :

```
df.drop('ID', axis=1)
```

	ID	Nama	Gender
0	1	Ani	P
1	2	Budi	L
2	3	Cahyo	L
3	4	Dodi	L
4	5	Ema	P
5	6	Firman	L



	Nama	Gender
0	Ani	P
1	Budi	L
2	Cahyo	L
3	Dodi	L
4	Ema	P
5	Firman	L

Hands-On

Outlier Handling

Outliers

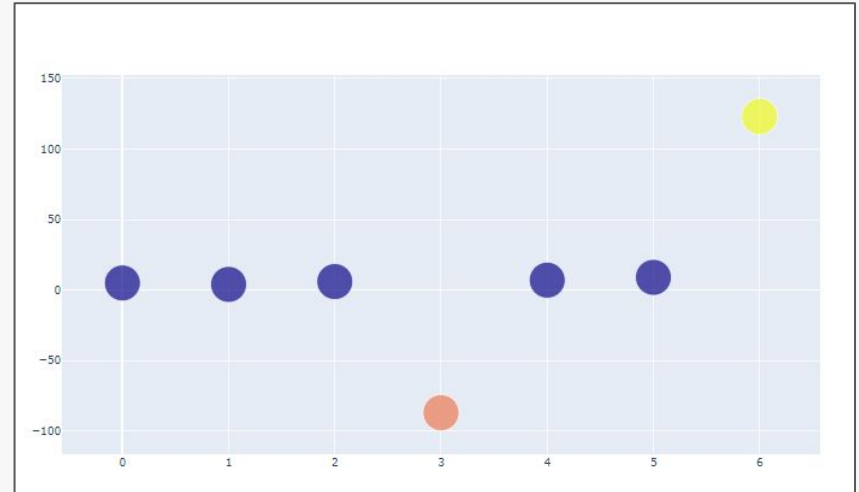
Pengertian : Pencilan (outliers) adalah titik data yang jauh dari kebanyakan data lainnya
Beberapa jenis pemeriksaan outlier adalah sebagai berikut

- Menggunakan Box Plot
- Menggunakan Distribusi Normal

-87 dapat dikatakan ekstrim bawah karena sebagian besar nilai (normal) terletak pada rentang 4-9 dan nilai tersebut jauh dibawah dari persekitaran nilai-nilai yang ada

X
5
4
6
-87
7
9
123

Pun juga nilai 123 dikatakan ekstrim atas karena nilai tersebut jauh diatas nilai – nilai di persekitarannya



Masalah Outlier

Untuk memahami pengaruh outlier terhadap ukuran pemusatan data antara mean atau median perhatikan contoh berikut

Perusahaan DQTech ingin mengetahui rata-rata dan juga nilai tengah dari data gaji karyawannya. Data dan perhitungan adalah sebagai berikut :

Posisi	Gaji
Data Analyst	5.000.000
Quality Assurance Eng.	5.750.000
Business Intelligence Dev.	6.000.000
Data Engineer	6.500.000
Data Scientist	7.000.000

$$\text{Rataan} = \frac{1}{n} \cdot \sum_{i=1}^n x_i = \frac{1}{5} (5jt + 5,75jt + \dots + 7jt) = 6.050.000$$
$$\text{Median} = 6.000.000$$

Kemudian ditambahkan data Gaji Manager IT pada sampel data diatas sehingga menjadi

Posisi	Gaji
IT Manager	21.200.000

$$\text{Rataan} = \frac{1}{n} \cdot \sum_{i=1}^n x_i = \frac{1}{6} (5jt + 5,75jt + \dots + 7jt + 21,2jt) = 8.575.000$$
$$\text{Median} = 6.500.000$$

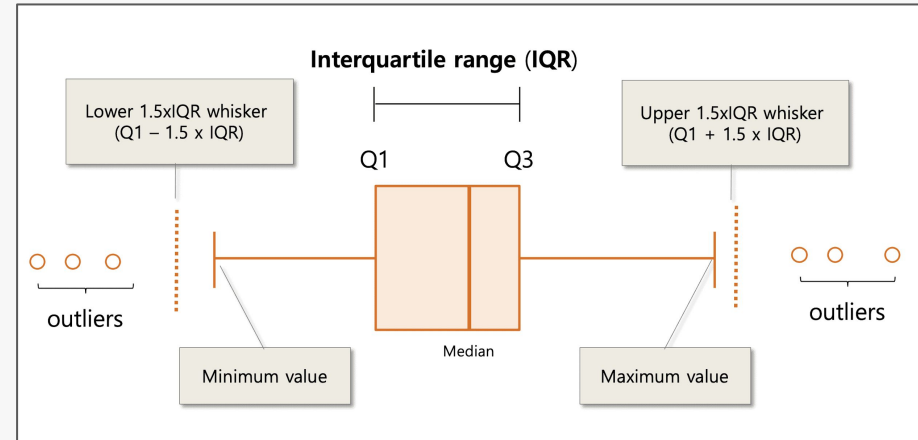
Dapatkah disimpulkan bahwa rata-rata gaji karyawan DQTech adalah 8.575.000?

Deteksi Outlier dengan Box Plot

Pengertian : Boxplot merupakan ringkasan distribusi sampel yang disajikan secara grafis yang bisa menggambarkan bentuk distribusi data (*skewness*), ukuran tendensi sentral dan ukuran penyebaran (keragaman) data pengamatan

Langkah perhitungan batas boxplot adalah sebagai berikut

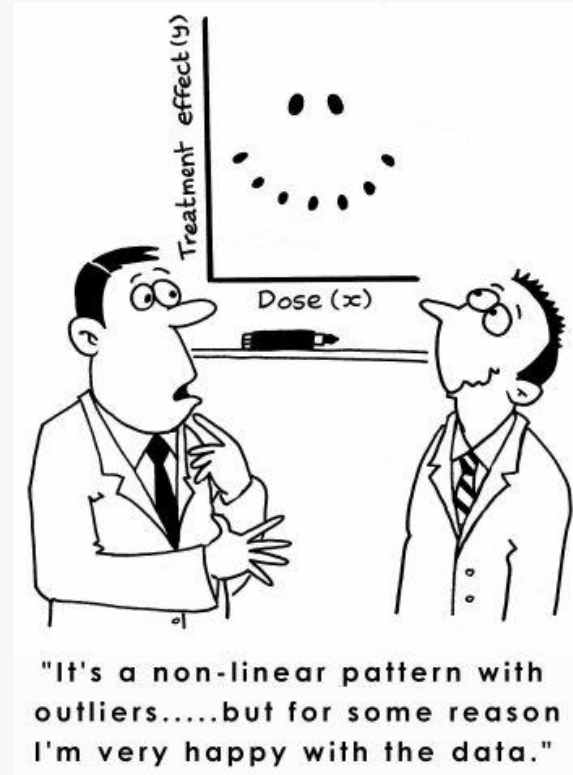
1. Hitung nilai median (nilai tengah)
2. Hitung nilai minimum dan maksimum
3. Hitung kuartil bawah (Q1) dan kuartil atas (Q3)
4. Hitung panjang langkah dengan rumus $1.5 \times (Q3 - Q1)$
5. Hitung pagar dalam = $Q1 - \text{langkah}$
6. Hitung pagar luar = $Q3 + \text{langkah}$



Mengatasi Outlier

Outlier dapat diatasi dengan 3 cara yang sama seperti penanganan *missing value* yakni

- *Leave As It Is* (biarkan saja)
- *Drop Row* yang mengandung *outliers*
- *Replace with Others Value*



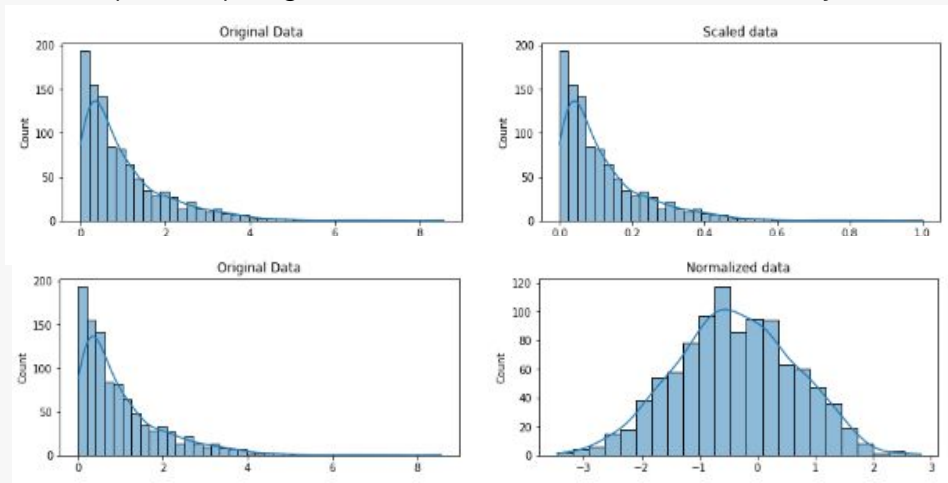
Data Transformation

Transformasi Data

Transformasi data adalah proses mengubah data mentah (*raw data*) menjadi format atau struktur yang lebih cocok untuk model atau algoritma dan juga penemuan data secara umum.

Transformasi Data yang umum digunakan adalah proses **Scaling** dan **Normalisasi**

- **scaling**: pengubahan data bertipe numerik menjadi rentang yang diinginkan (biasanya menjadi rentang 0 s.d. 1)
- **normalisasi** adalah proses pengubahan bentuk distribusi data menjadi data berdistribusi normal



Scaling

Salah satu proses scaling adalah dengan aturan Min-Max, dirumuskan sebagai

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Contoh :

Lakukan proses scaling dengan aturan min-max pada data 70, 80, 85, 35, 50, 67, 80, 82

Jawab :

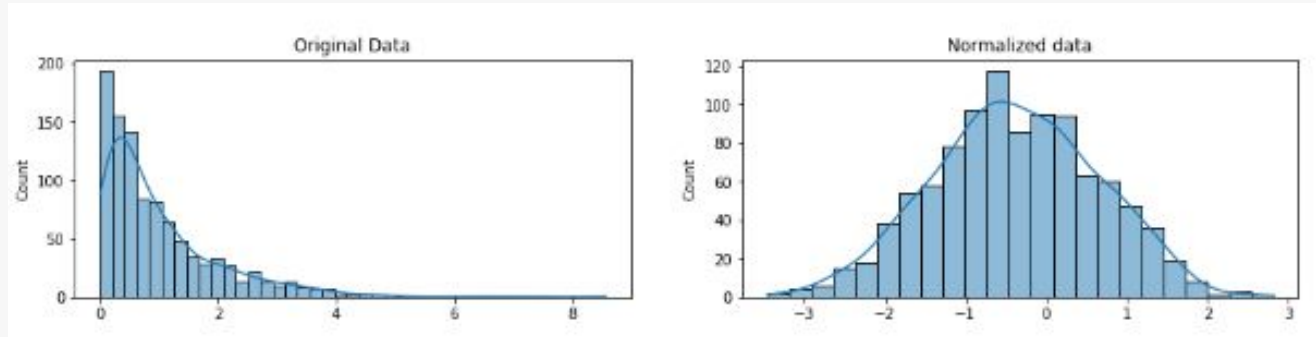
- $x_{\min} = 35$ dan $x_{\max} = 85$
- $x'_1 = \frac{x_1 - x_{\min}}{x_{\max} - x_{\min}} = \frac{70 - 35}{85 - 35} = 0.70$
- $x'_2 = \frac{x_2 - x_{\min}}{x_{\max} - x_{\min}} = \frac{80 - 35}{85 - 35} = 0.90$
- ...
- $x'_8 = \frac{x_8 - x_{\min}}{x_{\max} - x_{\min}} = \frac{82 - 35}{85 - 35} = 0.94$

x	$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$	x'
70		0.70
80		0.90
85		1.00
35		0.00
50		0.30
67		0.64
80		0.90
82		0.94

Dapat diperhatikan bahwa secara nilai berubah namun nilai yang di scaling akan tetap mempertahankan kedudukannya (nilai yang tinggi tetap ditransformasikan menjadi nilai yang tinggi begitu juga sebaliknya) namun pada range nilai yang lebih rendah

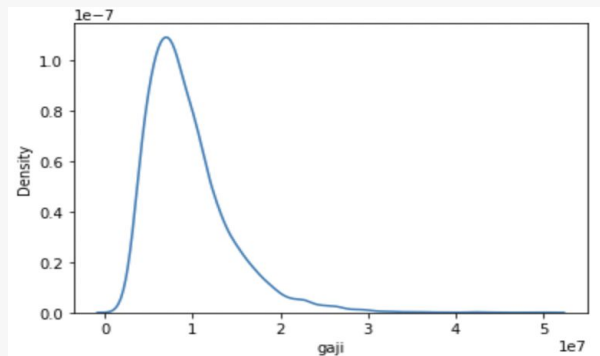
Normalisasi (log transformasi)

Proses pengubahan bentuk distribusi data menjadi data berdistribusi normal. Salah satu caranya adalah dengan menerapkan log transformasi pada kolom yang “punya buntut di kanan” (positively skewed)



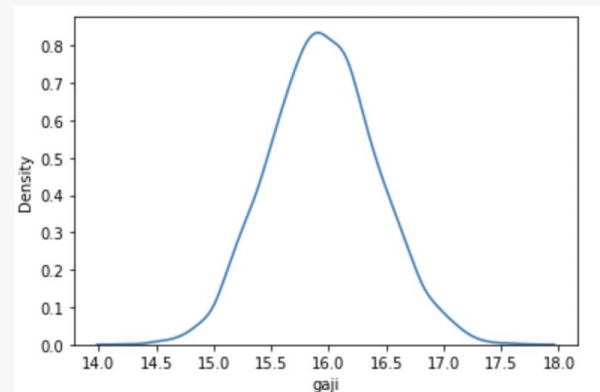
Normalisasi (log transformasi)

```
# distribusi gaji (nilai asli)  
sns.kdeplot(df['gaji'])
```



```
# distribusi gaji (setelah log transformation)  
sns.kdeplot(np.log(df['gaji']))  
# lebih mendekati distribusi normal!
```

```
# kita transformasi  
df['log_gaji'] = np.log(df['gaji'])  
  
# drop kolom gaji (nilai asli)  
df = df.drop(columns='gaji')
```



Scaling dengan sklearn.preprocessing

Scaling data menjadi rentang [0,1] dapat dilakukan dengan syntax berikut

```
from sklearn.preprocessing import MinMaxScaler
```

```
df['scaled_kolom'] = MinMaxScaler().fit_transform(df['kolom'].values.reshape(len(df), 1))
```

Kode di atas menunjukkan cara melakukan *normalization* dan *standardization* data dikolom gaji (hanya berbeda jenis scalernya). Berikut penjelasan tiap bagiannya:

1. `df[kolom].values` mengeluarkan nilai-nilai dari kolom
2. `.reshape(len(df), 1)` mengubah bentuk array menjadi format yang diperlukan
3. `MinMaxScaler().fit_transform()` melakukan transformasi scaling nilai fitur ke [0,1]
4. Hasilnya disimpan di kolom `scaled_kolom`

Hands-On

Quiz :
**Apa saja metode untuk
melakukan data cleaning?**

Feature Encoding

Apa itu **Feature Encoding**?

Feature Encoding adalah proses mengubah feature categorical menjadi feature numeric.

Mengapa kita perlu *feature encoding*?
Tak semua model/algorithm ML dapat menggunakan feature categorical.

Data Besaran Penghasilan dari Survei Abhal²

Gender	Pendidikan	Pekerjaan	Penghasilan (juta)
Laki-laki	S1	SWASTA	7
Laki-laki	SMA	PNS	13
Perempuan	S1	PNS	15
Laki-laki	S2	FREELANCE	24
Perempuan	S3	PNS	17
Perempuan	S1	SWASTA	23
Perempuan	SMA	FREELANCE	12


Pertanyaan :

Bagaimana cara menulis rumus dari Penghasilan **secara matematis**?

Teknik #1: Label Encoding

Label Encoding adalah perubahan feature categorical menjadi numeric dengan memberikan angka yang berbeda bagi masing-masing nilai unik

```
mapping_gender = {  
    'Laki-laki': 0,  
    'Perempuan': 1  
}  
df['gender'] = df['gender'].map(mapping_gender)  
  
mapping_pendidikan = {  
    'SMA': 0,  
    'S1': 1,  
    'S2': 2,  
    'S3': 3  
}  
df['pendidikan'] = df['pendidikan'].map(mapping_pendidikan)
```



	gender	pendidikan	pekerjaan	penghasilan
0	0	1	SWASTA	7
1	0	0	PNS	13
2	1	1	PNS	15
3	0	2	FREELANCE	24
4	1	3	PNS	17
5	1	1	SWASTA	23
6	1	0	FREELANCE	12

**Lalu bagaimana dengan kolom
'pekerjaan'?**

Teknik #2: One-hot Encoding

One-hot encoding adalah perubahan feature categorical menjadi numeric dengan menjadikan masing-masing nilai unik feature tersendiri

```
1 pd.get_dummies(df['pekerjaan'], prefix='kerja')
```

Kode di atas menunjukkan cara melakukan one-hot encoding pada kolom pekerjaan menggunakan `get_dummies()`. Berikut penjelasannya:

1. Parameter pertama adalah kolom yang ingin di one-hot encoding (pekerjaan)
2. Parameter prefix diisi dengan nama awalan dari kolom-kolom baru yang akan dihasilkan
3. Fungsi ini akan mengembalikan dataframe baru yang berisi feature-feature numerik

Teknik #2: One-hot Encoding (lanjutan)

	kerja_FREELANCE	kerja_PNS	kerja_SWASTA
0	0	0	1
1	0	1	0
2	0	1	0
3	1	0	0
4	0	1	0
5	0	0	1
6	1	0	0

Ketika kita tampilkan dataframe yang dihasilkan terlihat bahwa setiap nilai unik berubah menjadi kolom baru. Awalan nama kolom-kolom baru ini sesuai dengan isi parameter prefix.

Data Besaran Penghasilan dari Survei Abhal² (encoded)

Gender	Pendidikan	Freelance	PNS	Swasta	Penghasilan
0	1	0	0	1	7
0	0	0	1	0	13
1	1	0	1	0	15
0	2	1	0	0	24
1	3	0	1	0	17
1	1	0	0	1	23
1	0	1	0	0	12

Rumus nilai penghasilan:

- $A * \text{gender} + B * \text{pendidikan} + C * \text{freelance} + D * \text{PNS} + E * \text{swasta}$
- Dimana A, B, C, D, E didapat dari melatih model machine learning (regresi)

Recap :

Label Encoding
atau OHE?

- **Gunakan Label Encoding pada:**
 - Kolom kategorikal dengan jumlah distinct values = 2. E.g. Gender, respon ya/tidak, etc
 - Kolom kategorikal dengan tipe ordinal (punya urutan). E.g. tingkat pendidikan, intensitas (rendah/medium/tinggi), socio-economic status (A/B/C/D), etc
- **Selainnya, gunakan One Hot Encoding (OHE)**

Terima Kasih

Thanks!

