



Direktorat Jenderal Pendidikan Tinggi, Riset, dan Teknologi
Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi
Republik Indonesia



MICROCREDENTIAL: ASSOCIATE DATA SCIENTIST

01 November – 10 Desember 2021

Pertemuan ke-9

Data Preparation 3: Mengkonstruksi Data



[ditjen.dikti](https://www.facebook.com/ditjen.dikti)



@ditjendik
ti



[ditjen.dikti](https://www.instagram.com/ditjen.dikti/)



Ditjen
Diktiristek



<https://dikti.kemdikbud.go.id/>

Profil Pengajar: Erwin Eko Wahyudi, S.Kom., M.Cs.



Jabatan Akademik: Tenaga Pengajar

Latar Belakang Pendidikan:

- S1: Ilmu Komputer UGM, 2012-2017
- S2: Ilmu Komputer UGM, 2017-2019

Riwayat/Pengalaman Pekerjaan:

- Dosen, UGM, 2021-sekarang
- AI Engineer Recommender System, Bukalapak, 2019-2021

Contact Pengajar:

Ponsel:

0812 1195 4011

Email:

erwin.eko.w@ugm.ac.id

Course Definition

- Pelatihan ini adalah bagian ketiga dari Data Preparation.
- Data Preparation yang dibahas adalah transformasi data yaitu:
 - Representasi Fitur
 - Rekayasa Fitur
 - Pelabelan Data
- Ada beberapa teknik transformasi data yang digunakan sesuai kebutuhan dan ketersediaan/jenis data baik numerik maupun kategorik



Learning Objective

Dalam pelatihan ini diharapkan:

- A. Peserta mampu menganalisis teknik transformasi data
- B. Peserta mampu melakukan transformasi data
- C. Peserta mampu membuat dokumentasi konstruksi data
- D. Peserta mampu melakukan pelabelan data
- E. Peserta mampu membuat dokumentasi pelabelan data

Referensi: SKKNI Data Science

KODE UNIT : J.62DMI00.009.1

JUDUL UNIT : Mengkonstruksi Data

DESKRIPSI UNIT: Unit kompetensi ini berhubungan dengan pengetahuan, keterampilan, dan sikap kerja yang dibutuhkan dalam mengkonstruksi data untuk proyek *data science*

| ELEMEN KOMPETENSI | KRITERIA UNJUK KERJA |
|--|--|
| 1. Menganalisis teknik transformasi data | 1.1 Analisis data untuk menentukan representasi fitur data awal . 1.2 Analisis representasi fitur data awal untuk menentukan teknik rekayasa fitur yang diperlukan untuk pembangunan model <i>data science</i> . |
| 2. Melakukan transformasi data | 2.1 Transformasi dilakukan untuk mendapatkan fitur data awal. 2.2 Rekayasa fitur data dilakukan untuk mendapatkan fitur baru yang diperlukan untuk pembangunan model <i>data science</i> . |
| 3. Membuat dokumentasi konstruksi data | 2.3 Teknis transformasi data dijabarkan dalam bentuk tertulis. 2.4 Hasil transformasi data dan rekomendasi hasil transformasi dituangkan dalam bentuk tertulis. |

1. Konteks variabel

- 1.1 Representasi fitur data awal dapat berupa kolom data atau fitur tipe data yang dapat digunakan untuk algoritme *machine learning* sesuai dengan tipe data.
- 1.2 Rekayasa fitur data dapat berupa normalisasi, pemilihan fitur tipe data baru, menambahkan kolom data baru.
- 1.3 Kolom data baru adalah kolom data yang merupakan turunan nilai dari satu atau lebih data yang ada.
- 1.4 Fitur tipe data adalah fitur dari data yang akan digunakan dalam algoritme *machine learning* untuk data tidak terstruktur. Tipe data tidak terstruktur seperti *free text*, suara, gambar, dan video. Contoh fitur adalah seperti TF-IDF, frekuensi suara, warna, lokasi piksel, dan lainnya.
- 1.5 Normalisasi adalah cara yang diterapkan pada data berstruktur yang memiliki nilai berjenjang. Teknik normalisasi diantaranya, *binning*, *minimum-maximum*, *scaling*.



Referensi: SKKNI Data Science

KODE UNIT : J.62DMI00.010.1

JUDUL UNIT : Menentukan Label Data

DESKRIPSI UNIT: Unit kompetensi ini berhubungan dengan pengetahuan, keterampilan, dan sikap kerja yang dibutuhkan untuk menentukan label data untuk pembangunan model *data science*

| ELEMEN KOMPETENSI | KRITERIA UNJUK KERJA |
|---|--|
| 1. Melakukan pelabelan data | 1.1 Analisis hasil pelabelan data sejenis yang sudah ada diuraikan kesesuaianya dengan Standard Operating Procedure (SOP) pelabelan . 1.2 Pelabelan data dilakukan sesuai dengan SOP pelabelan. |
| 2. Membuat laporan hasil pelabelan data | 2.1. Statistik hasil pelabelan diuraikan pada laporan. 2.2. Evaluasi proses pelabelan diuraikan pada laporan. |

BATASAN VARIABEL

1. Konteks variabel
 - 1.1 Pelabelan data adalah proses memberikan label pada data yang akan digunakan pada pemodelan *machine learning*.
 - 1.2 *Standard Operating Procedure (SOP)* pelabelan adalah panduan langkah-langkah dan aturan dalam melakukan proses pelabelan data sesuai dengan domain data.

Transformasi Data



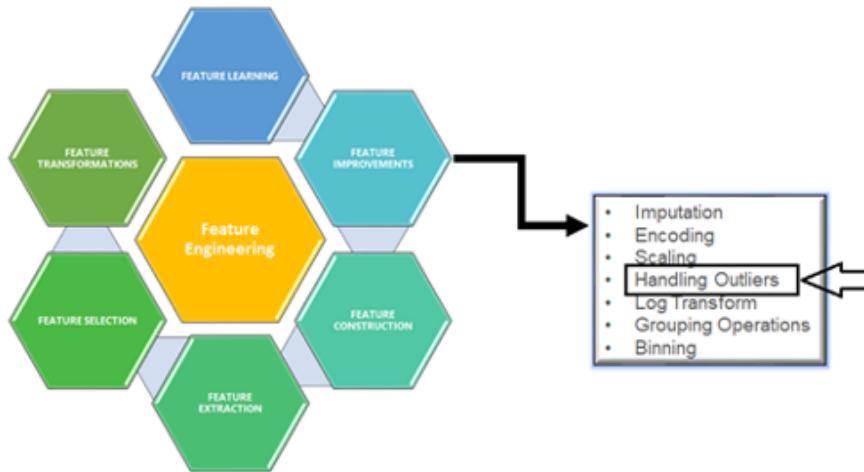


Data Transformation

- Representasi Fitur atau Pembelajaran Fitur:
 - Teknik-Teknik yang memungkinkan sistem bekerja otomatis menemukan representasi yang diperlukan (untuk deteksi fitur atau klasifikasi dari dataset),
 - menggantikan rekayasa fitur manual, dan
 - memungkinkan mesin mempelajari fitur dan menggunakan untuk melakukan tugas tertentu.
- Rekayasa Fitur:
 - Proses mengubah data mentah menjadi fitur yang:
 - Mewakili masalah mendasar ke model prediktif,
 - menghasilkan akurasi model yang lebih baik pada data yang tidak terlihat.



Rekayasa Fitur



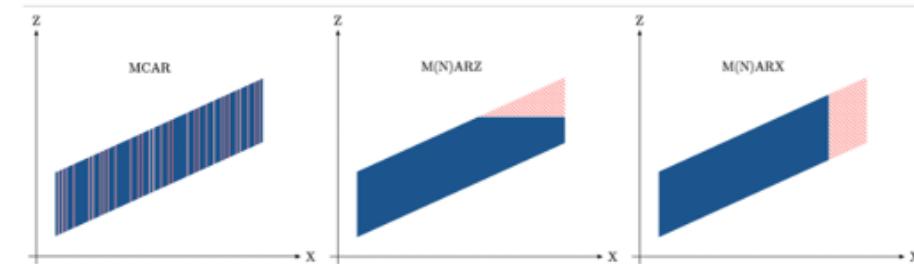
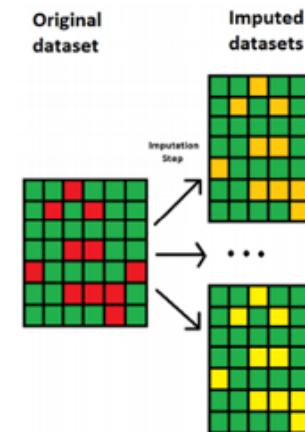
Outlier





Imputasi

- Pengertian: Mengganti nilai/data yang hilang (missing value; NaN; blank) dengan nilai pengganti.
- Jenis missing value:
 1. Missing Completely At Random (MCAR).
 2. Missing At Random (MAR).
 3. Missing Not At Random (MNAR).



sum.gbr:
https://www.ucl.ac.uk/population-health-sciences/sites/population-health-sciences/files/quartagno_1.pdf
https://rianneschouten.github.io/missing_data_science/assets/blogpost.html



Missing Completely At Random (MCAR)

- Definisi: Probabilitas sebuah instance yang hilang tidak bergantung pada nilai yang diketahui atau nilai yang hilang itu sendiri.
- Contoh: 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.

Missing At Random (MAR)

- Definisi: Probabilitas sebuah instance yang hilang mungkin bergantung pada nilai yang diketahui tetapi tidak pada nilai yang hilang itu sendiri.
- Contoh Sensor: 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.
- Contoh survei: Apakah seseorang menjawab pertanyaan atau tidak - mis. tentang usia- dalam survei tidak tergantung pada jawaban itu sendiri, tetapi mungkin tergantung pada jawaban untuk pertanyaan lain, yaitu jenis kelamin perempuan.



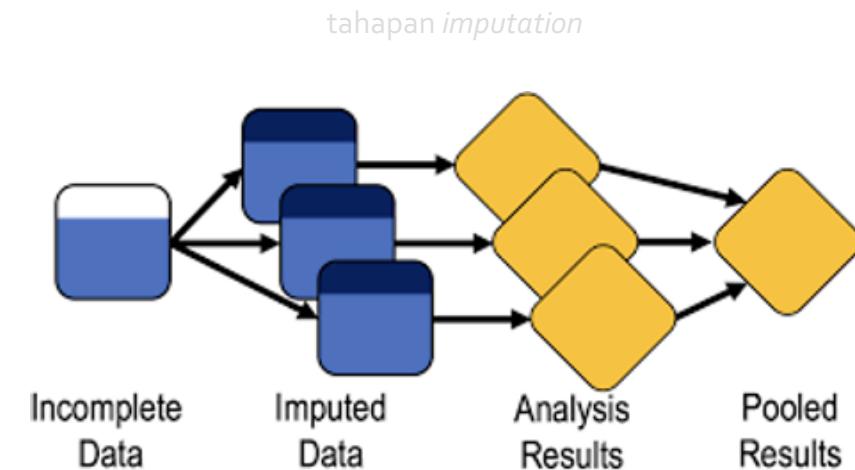
Missing Not At Random (MNAR)

- Definisi: probabilitas sebuah instance hilang dapat bergantung pada nilai variabel itu sendiri.
- Contoh sensor: Dalam kasus sensor suhu, sensor tidak berfungsi dengan baik saat suhu lebih dingin dari 5°C.
- Contoh survei: Apakah seseorang menjawab pertanyaan atau tidak - mis. jumlah hari sakit - dalam survei memang tergantung pada jawabannya sendiri - seperti yang bisa terjadi pada beberapa orang yang kelebihan berat badan.



Tahapan dan Teknik Imputasi

- Jika tipe data Variabel Numerik
 - Imputasi mean atau median.
 - Imputasi nilai suka-suka (arbitrary).
 - Imputasi nilai/data ujung (end of tail).
- Jika tipe data Variabel Kategorik
 - Imputasi kategori yang sering muncul.
 - Tambah kategori yang hilang.



sumber gbr:
https://www.researchgate.net/publication/334213038_Missing_data_and_bias_in_physics_education_research_A_case_for_using_multiple_imputation

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.

The diagram illustrates the process of data imputation. On the left, a dataset with five columns (col1 to col5) and three rows contains several NaN values. A large arrow points from this state to the right, where the dataset has been transformed. Above this transformation, the text "mean()" indicates the operation performed. The resulting dataset on the right shows that all NaN values have been replaced by the mean of their respective columns. Below this, another diagram shows the imputation of missing age values. On the left, a list of ages (29, 43, NA, 25, 34, NA, 50) is shown with a "mean" calculation: $(29 + 43 + 25 + 34 + 50)/5 = 36,2$. A large arrow points to the right, where the original list is now replaced by the imputed value 36,2. The original NA value is also highlighted in red.

| | 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 |

mean()

| | col1 | col2 | col3 | col4 | col5 |
|---|------|------|------|------|------|
| 0 | 2.0 | 5.0 | 3.0 | 6.0 | 7.0 |
| 1 | 9.0 | 11.0 | 9.0 | 0.0 | 7.0 |
| 2 | 19.0 | 17.0 | 6.0 | 9.0 | 7.0 |

| | Age |
|---|-----|
| 1 | 29 |
| 2 | 43 |
| 3 | NA |
| 4 | 25 |
| 5 | 34 |
| 6 | NA |
| 7 | 50 |

mean

$= (29 + 43 + 25 + 34 + 50)/5$

$= 36,2$

| | Age |
|---|------|
| 1 | 29 |
| 2 | 43 |
| 3 | 36,2 |
| 4 | 25 |
| 5 | 34 |
| 6 | 36,2 |
| 7 | 50 |

sumber gbr:
<https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779>
<https://heartbeat.fritz.ai/hands-on-with-feature-engineering-techniques-imputing-missing-values-6c22b49d4060>



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

Hands On (Lanjutan)

```
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



Imputasi Nilai Suka-suka

- Pro:
 - Asumsi data tidak *missing at random*.
 - Mudah dan cepat diterapkan untuk melengkapi dataset.
- Kontra:
 - Mendistorsi variansi dan distribusi variabel asal/orijinal serta covariant variabel sisa data.
 - Membentuk outlier (jika nilai arbitrer berada di akhir distribusi).
 - Semakin besar NA maka semakin besar distorsi.
 - Hindari memilih nilai arbitrer yg mendekati nilai mean atau median.

| Age |
|-----|
| 29 |
| 43 |
| NA |
| 25 |
| 34 |
| NA |
| 50 |



| Age |
|-----|
| 29 |
| 43 |
| 99 |
| 25 |
| 34 |
| 99 |
| 50 |



Hands On

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

Import pandas dan numpy

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

Masukan Data

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

Ubah ke dataframe

umur

0 29.0

1 43.0

2 NaN

3 25.0

4 34.0

5 NaN

6 50.0

Output

```
data.fillna(99)
```

Mengatasi nilai missing value dengan imputasi suka-suka

umur

0 29.0

1 43.0

2 99.0

3 25.0

4 34.0

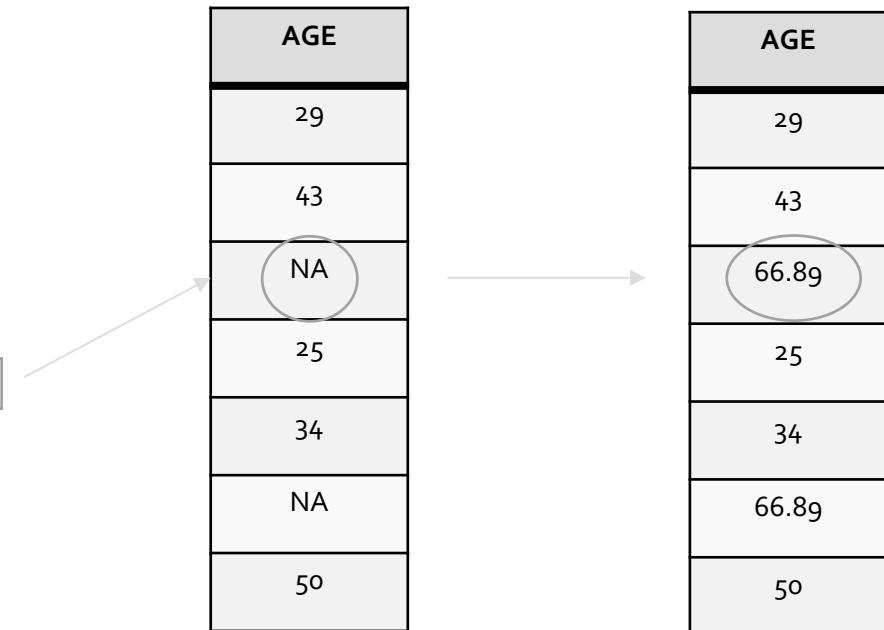
5 99.0

6 50.0

Output

Imputasi End of Tail

- Pro:
 - Mirip dengan imputasi suka-suka.
 - Cocok untuk variabel numerik.
- Ketentuan khusus:
 - Dalam memilih nilai arbitrer:
 - Jika variabel berdistribusi normal, maka nilai arbiter = mean + 3 * std deviasi.
 - Jika variabelnya *skew*, maka gunakan aturan IQR proximity ($IQR = 75\text{th Quantile} - 25\text{th Quantile}$; Upper limit = $75\text{th Quantile} + IQR \times 3$; Lower limit = $25\text{th Quantile} - IQR \times 3$).
 - Hanya digunakan pada data latih (training set).



untuk data distribusi normal



Hands On

```
import pandas as pd
import numpy as np
.
.
.
umur = {'umur' : [29, 43, np.nan, 25, 34, np.nan, 50]}
data = pd.DataFrame(umur)
data
```

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

→ import pandas dan numpy

masukkan data

→ Output

```
from feature_engine.imputation import EndTailImputer
.
.
.
imputer = EndTailImputer(imputation_method='gaussian', tail='right')
imputer.fit(data) → cocokan imputer ke set
test_t = imputer.transform(data) → mengubah data
```

test_t

| | umur |
|---|-----------|
| 0 | 29.000000 |
| 1 | 43.000000 |
| 2 | 66.896905 |
| 3 | 25.000000 |
| 4 | 34.000000 |
| 5 | 66.896905 |
| 6 | 50.000000 |

→ Output

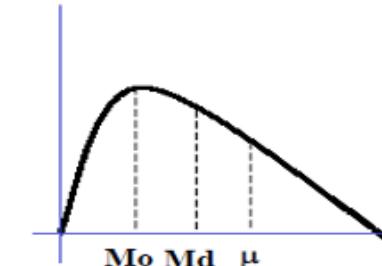
Skewness (kecenderungan/kemiringan/kemencengan)

Miring ke kanan (positif skew)

Data yang miring ke kanan memiliki ekor panjang yang memanjang ke kanan.

Cara alternatif untuk membicarakan kumpulan data yang miring ke kanan adalah dengan mengatakan bahwa itu secara positif.

Jika rata-rata (mean) $>$ median $>$ modus, maka pada kurva distribusi frekuensi, nilai mean akan terletak di sebelah kanan, sedangkan median terletak di tengahnya dan modus di sebelah kiri

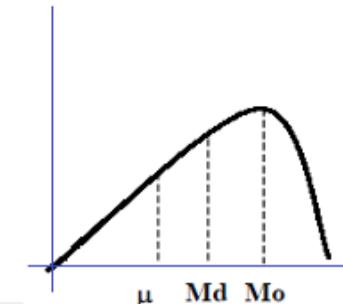


Skewness (kecenderungan/kemiringan/kemencengan)

Miring ke kiri (negatif skew)

Data yang miring ke kiri memiliki ekor panjang yang memanjang ke kiri. Cara alternatif untuk membicarakan kumpulan data yang miring ke kiri adalah dengan mengatakan bahwa itu miring secara negatif.

Jika rata-rata (mean) $<$ median $<$ modus, maka pada kurva distribusi frekuensi, nilai mean akan terletak di sebelah kiri, sedangkan median terletak di tengahnya dan modus di sebelah kanan



Imputasi Frequent Category/Modus

- Pro:
 - Cocok untuk data dengan missing at random.
 - Mudah dan cepat diterapkan.
 - Cocok utk data yang memiliki *skew*
 - Dapat digunakan dalam produksi (mis. dalam model deployment).
- Kontra:
 - Mendistorsi relasi label dengan frekuensi tertinggi vs variabel lain.
 - Menghasilkan over-representation jika banyak data yang missing.

| Make | Price |
|------|-------|
| Ford | Ford |
| Ford | Ford |
| Fiat | Fiat |
| BMW | BMW |
| Ford | Ford |
| Kia | Kia |
| | |
| Fiat | Fiat |
| Ford | Ford |
| | |
| Kia | Kia |

Mode = Ford





Hands On

```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
```

Import pandas, SimpleImputer dari
sklearn.impute, numpy

```
make = {'make' : ['Ford', 'Ford', 'Fiat', 'BMW', 'Ford', 'Kia', np.nan, 'Fiat', 'Ford', np.nan, 'Kia']}
```

Masukan
Data

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

```
data
```

Ubah menjadi data frame

| | make |
|----|------|
| 0 | Ford |
| 1 | Ford |
| 2 | Fiat |
| 3 | BMW |
| 4 | Ford |
| 5 | Kia |
| 6 | NaN |
| 7 | Fiat |
| 8 | Ford |
| 9 | NaN |
| 10 | Kia |

Output



Hands On (Lanjutan)

```
imp = SimpleImputer(strategy='most_frequent')  
  
imp.fit_transform(data)  
  
array([['Ford'],  
       ['Ford'],  
       ['Fiat'],  
       ['BMW'],  
       ['Ford'],  
       ['Kia'],  
       ['Ford'],  
       ['Fiat'],  
       ['Ford'],  
       ['Ford'],  
       ['Kia']], dtype=object)
```

Mengatasi missing value
dengan frequent category /
modus

Output

Imputasi Random Sample

- Pro:
 - Cocok untuk data *missing at random*.
 - Ganti missing value dengan nilai lain dalam distribusi yang sama dari variabel asli.
 - Mudah dan cepat.
 - Mempertahankan varians dari variabel.
- Kontra:
 - Randomness.
 - Membutuhkan memori besar untuk deployment karena perlu untuk menyimpan data latih yg asli untuk ekstraksi nilai.

| Gender | Age |
|--------|-----|
| Male | 29 |
| Male | NA |
| NA | 43 |
| Female | 25 |
| Male | 34 |
| NA | 50 |
| Female | NA |



| Gender | Age |
|--------|-----|
| Male | 29 |
| Male | 34 |
| Female | 43 |
| Female | 25 |
| Male | 34 |
| Male | 50 |
| Female | 25 |



Hands On

```
from feature_engine.imputation import RandomSampleImputer  
import pandas as pd  
import numpy as np  
  
data = {'Gender' : ['Male', 'Male', np.nan, 'Female', 'Male', np.nan, 'Female'],  
        'Age' : [29, np.nan, 43, 25, 34, 50, np.nan]}  
  
df = pd.DataFrame(data)  
  
df
```

import RandomSampleImputer,
pandas, numpy

| | Gender | Age |
|---|--------|------|
| 0 | Male | 29.0 |
| 1 | Male | NaN |
| 2 | NaN | 43.0 |
| 3 | Female | 25.0 |
| 4 | Male | 34.0 |
| 5 | NaN | 50.0 |
| 6 | Female | NaN |

masukkan data dan
ubah menjadi
dataframe

Output



Hands On (Lanjutan)

```
imputer = RandomSampleImputer(random_state = 29)
```

→ Buat imputernya

```
imputer.fit(df)
```

→ Cocokan imputer ke set

```
test_t = imputer.transform(df)
```

→ Mengubah data

```
test_t
```

| | Gender | Age |
|---|--------|------|
| 0 | Male | 29.0 |
| 1 | Male | 34.0 |
| 2 | Male | 43.0 |
| 3 | Female | 25.0 |
| 4 | Male | 34.0 |
| 5 | Female | 50.0 |
| 6 | Female | 50.0 |

→ Output



Imputasi Nilai Nol/Konstanta

- Pro:
 - Cocok untuk variabel kategorik.
- Kontra:
 - Tidak memperhitungkan korelasi antar fitur.
 - Memunculkan bias dalam data.

The screenshot shows a Jupyter Notebook cell with two data frames. The first data frame, labeled 'df', has columns 'col1' through 'col5'. The second data frame, labeled 'df.fillna(0)', shows the result of replacing NaN values with 0. A blue arrow points from the original data frame to the result.

| | 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 |

df.fillna(0)

| | col1 | col2 | col3 | col4 | col5 |
|---|------|------|------|------|------|
| 0 | 2 | 5.0 | 3.0 | 6 | 0.0 |
| 1 | 9 | 0.0 | 9.0 | 0 | 7.0 |
| 2 | 19 | 17.0 | 0.0 | 9 | 0.0 |

sumber gbr:
<https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779>



Hands On

```
: #Contoh 1 - Imputasi Nilai Nol/Konstanta
#Masukkan data

umur = {'umur' : [29, 43, np.nan, 25, 34, np.nan, 50]} → masukkan data dan ubah
#Ubah ke dataframe
menjadi dataframe

data = pd.DataFrame(umur)

#Tampilkan nilai yang ada pada variabel 'umur' dalam bentuk dataframe 'data'

data
```

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

Contoh nilai hilang

masukkan data dan ubah
menjadi dataframe

```
: #Contoh 1 - Imputasi Nilai Nol/Konstanta
#Mengatasi nilai missing value dengan imputasi suka-suka
#Nilai suka-suka yang diberikan pada contoh adalah 99
```

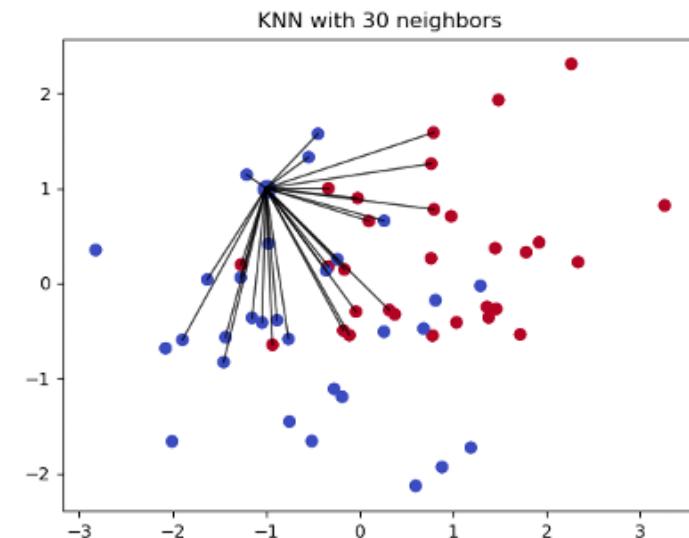
```
data.fillna(0) → Imputasi nilai nol
```

| | umur |
|---|------|
| 0 | 29.0 |
| 1 | 43.0 |
| 2 | 0.0 |
| 3 | 25.0 |
| 4 | 34.0 |
| 5 | 0.0 |
| 6 | 50.0 |

Output

Imputasi dengan K-NN

- Pro:
 - Lebih akurat vs *mean/median/most frequent.*
- Kontra:
 - Biaya komputasi mahal (karena KNN bekerja dengan menyimpan seluruh dataset pelatihan dalam memori).
 - Sensitif terhadap outlier dalam data (tidak seperti SVM).

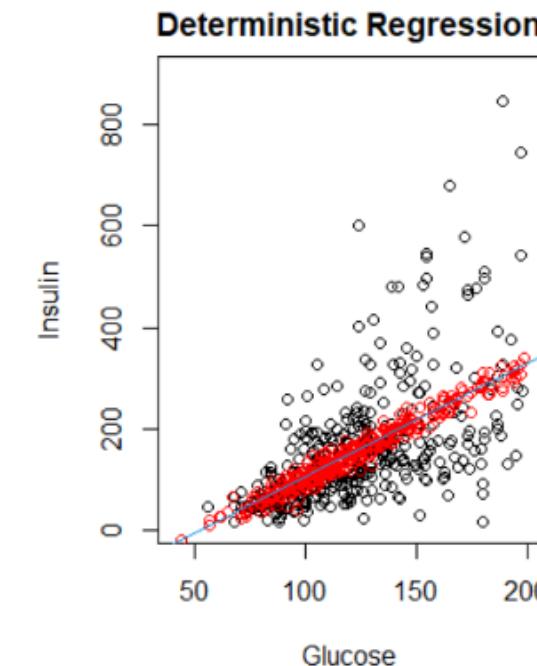


sumber gbr:
<https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779>



Imputasi Regresi: Deterministik

- Deterministik
 - Mengganti missing value dengan prediksi yang tepat dari model regresi.
 - Tidak mempertimbangkan variasi acak di sekitar kemiringan (slope regresi).
 - Nilai imputasi seringkali terlalu tepat (precise) dan overestimasi dari korelasi X-Y.



sumber gbr:
<https://statisticsglobe.com/regression-imputation-stochastic-vs-deterministic/>

Hands On

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as mno
from sklearn import linear_model
%matplotlib inline
```

```
df = pd.read_csv("diabetes.csv")
df.head(3)
```

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|-------------|---------|---------------|---------------|---------|------|--------------------------|-----|---------|
| 0 | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| 1 | 1 | 85 | 68 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| 2 | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   Pregnancies      768 non-null    int64  
 1   Glucose          768 non-null    int64  
 2   BloodPressure    768 non-null    int64  
 3   SkinThickness    768 non-null    int64  
 4   Insulin          768 non-null    int64  
 5   BMI              768 non-null    float64 
 6   DiabetesPedigreeFunction 768 non-null    float64 
 7   Age              768 non-null    int64  
 8   Outcome          768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Import Library

Menentukan nilai yang hilang

Output

melihat info dari dataframe

Output

```
df.describe() → menghitung nilai statistik
```

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|-------|-------------|------------|---------------|---------------|------------|------------|--------------------------|------------|------------|
| count | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 |
| mean | 3.845052 | 120.894531 | 69.106469 | 20.536458 | 79.798479 | 31.992578 | 0.471876 | 33.240885 | 0.348958 |
| std | 3.389578 | 31.972618 | 19.356807 | 15.952218 | 115.244002 | 7.884160 | 0.331329 | 11.780232 | 0.476951 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.078000 | 21.000000 | 0.000000 |
| 25% | 1.000000 | 99.000000 | 62.000000 | 0.000000 | 0.000000 | 27.300000 | 0.243750 | 24.000000 | 0.000000 |
| 50% | 3.000000 | 117.000000 | 72.000000 | 23.000000 | 30.500000 | 32.000000 | 0.372500 | 28.000000 | 0.300000 |
| 75% | 6.000000 | 140.250000 | 80.000000 | 32.000000 | 127.250000 | 36.600000 | 0.626250 | 41.000000 | 1.000000 |
| max | 17.000000 | 199.000000 | 122.000000 | 99.000000 | 846.000000 | 67.100000 | 2.420000 | 81.000000 | 1.000000 |

```
df.loc[df["Glucose"] == 0.0, "Glucose"] = np.NAN
df.loc[df["BloodPressure"] == 0.0, "BloodPressure"] = np.NAN
df.loc[df["SkinThickness"] == 0.0, "SkinThickness"] = np.NAN
df.loc[df["Insulin"] == 0.0, "Insulin"] = np.NAN
df.loc[df["BMI"] == 0.0, "BMI"] = np.NAN
```

```
df.isnull().sum()[1:6]
```

| | |
|---------------------|-----|
| Glucose | 5 |
| BloodPressure | 35 |
| SkinThickness | 227 |
| Insulin | 374 |
| BMI | 11 |
| dtype: int64 | |

Output

Output

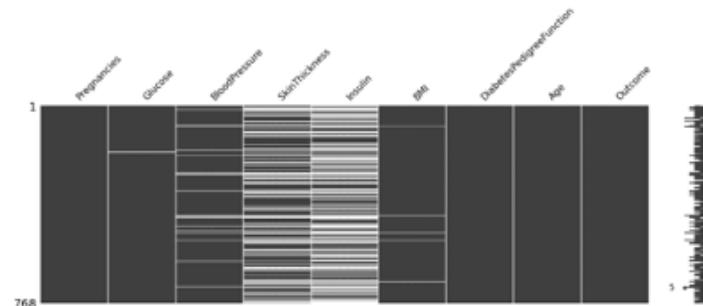
df.loc[] --> memilih table berdasarkan lokasi

Hands On (Lanjutan)

```
mno.matrix(df, figsize = (20, 6))
```

```
<FigureSubplot>
```

→ Membuat Matriks,
figsize --> ukuran figure



→ Output

```
missing_columns = ["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]
```

```
def random_imputation(df, feature):  
  
    number_missing = df[feature].isnull().sum()  
    observed_values = df.loc[df[feature].notnull(), feature]  
    df.loc[df[feature].isnull(), feature + '_imp'] = np.random.choice(observed_values, number_missing, replace = True)  
  
    return df
```

```
for feature in missing_columns:  
    df[feature + '_imp'] = df[feature]  
    df = random_imputation(df, feature)
```

→ Menggunakan Regresi untuk
memperhitungkan data yang
hilang



Hands On (Lanjutan)

```
deter_data = pd.DataFrame(columns = ["Det" + name for name in missing_columns])  
  
for feature in missing_columns:  
  
    deter_data["Det" + feature] = df[feature + "_imp"]  
    parameters = list(set(df.columns) - set(missing_columns) - {feature + '_imp'})
```

Deterministic Regression
Imputation

```
model = linear_model.LinearRegression()  
model.fit(X = df[parameters], y = df[feature + '_imp'])
```

Buat model Regresi Linier untuk memperkirakan
data yang hilang

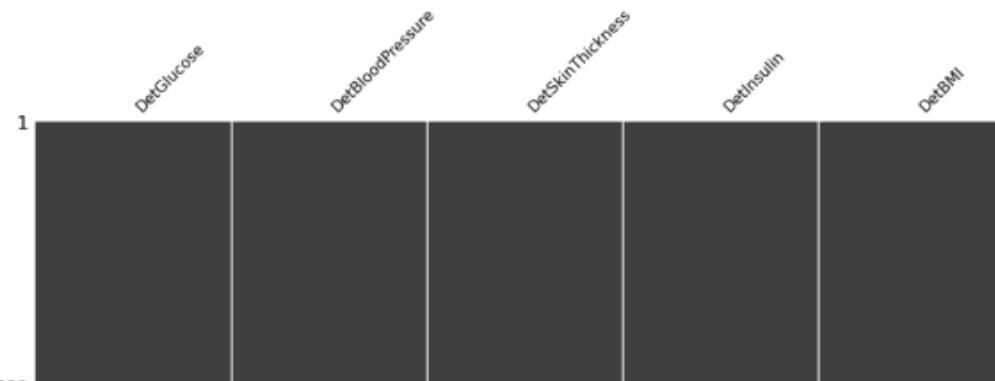
```
deter_data.loc[df[feature].isnull(), "Det" + feature] = model.predict(df[parameters])[df[feature].isnull()]
```

amatil bahwa kita menyimpan
indeks data yang hilang dari
kerangka data asli

```
mno.matrix(deter_data, figsize = (20,5))
```

membuat matriks

```
<AxesSubplot:>
```

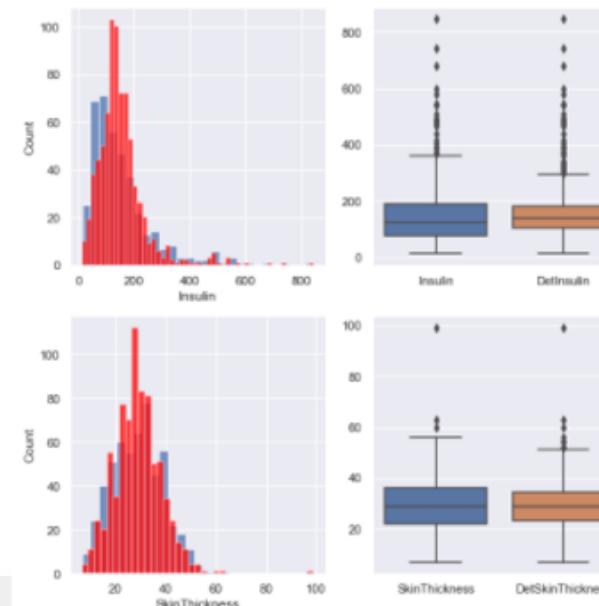


Output



Hands On (Lanjutan)

```
sns.set()  
fig, axes = plt.subplots(nrows = 2, ncols = 2)  
fig.set_size_inches(8, 8)  
  
for index, variable in enumerate(["Insulin", "SkinThickness"]):  
    sns.histplot(df[variable].dropna(), kde = False, ax = axes[index, 0])  
    sns.histplot(deter_data["Det" + variable], kde = False, ax = axes[index, 0], color = 'red')  
  
    sns.boxplot(data = pd.concat([df[variable], deter_data["Det" + variable]]), axis = 1,  
                ax = axes[index, 1])  
  
plt.tight_layout()
```



membuat
chart

```
pd.concat([df[["Insulin", "SkinThickness"]], deter_data[["DetInsulin", "DetSkinThickness"]]], axis = 1).describe().T
```

| | count | mean | std | min | 25% | 50% | 75% | max |
|------------------|-------|------------|------------|------|------------|------------|------------|-------|
| Insulin | 394.0 | 155.548223 | 118.775855 | 14.0 | 76.250000 | 125.000000 | 190.000000 | 846.0 |
| SkinThickness | 541.0 | 29.153420 | 10.476982 | 7.0 | 22.000000 | 29.000000 | 36.000000 | 99.0 |
| DetInsulin | 768.0 | 154.559090 | 89.394698 | 14.0 | 105.029991 | 140.217157 | 182.000000 | 846.0 |
| DetSkinThickness | 768.0 | 29.097247 | 9.189058 | 7.0 | 23.000000 | 29.000000 | 34.461758 | 99.0 |

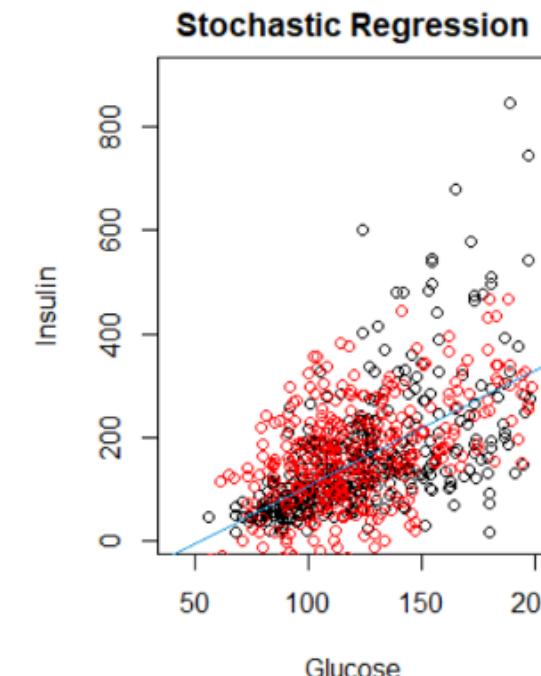
Output

Output



Imputasi Regresi: Stokastik

- Stokastik
 - Mengatasi masalah dalam imputasi regresi deterministik.
 - Menambahkan fitur “random error” sehingga reproduksi korelasi X-Y lebih tepat.



sumber gbr:
<https://statisticsglobe.com/regression-imputation-stochastic-vs-deterministic/>

Hands On

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as mno
from sklearn import linear_model
%matplotlib inline
```

import library

```
df = pd.read_csv("diabetes.csv")
df.head(3)
```

Menentukan nilai yang hilang

| Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|-------------|---------|---------------|---------------|---------|-----|--------------------------|-------|---------|
| 0 | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 |
| 1 | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 |
| 2 | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 |

Output

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   Pregnancies      768 non-null    int64  
 1   Glucose          768 non-null    int64  
 2   BloodPressure    768 non-null    int64  
 3   SkinThickness    768 non-null    int64  
 4   Insulin          768 non-null    int64  
 5   BMI              768 non-null    float64 
 6   DiabetesPedigreeFunction 768 non-null    float64 
 7   Age              768 non-null    int64  
 8   Outcome          768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

melihat info dari data frame

Output

| df.describe() | | | | | | | | |
|---------------|------------|---------------|---------------|------------|------------|--------------------------|------------|------------|
| Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| count | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 |
| mean | 3.84562 | 120.894531 | 69.105488 | 20.598458 | 39.799479 | 31.987578 | 0.471876 | 33.240685 |
| std | 3.368576 | 31.972616 | 19.335007 | 15.852210 | 110.244002 | 7.804160 | 0.371329 | 11.760232 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.078000 | 21.000000 |
| 25% | 1.000000 | 89.000000 | 62.000000 | 0.000000 | 0.000000 | 27.300000 | 0.243759 | 24.000000 |
| 50% | 3.000000 | 117.000000 | 72.000000 | 23.000000 | 30.500000 | 32.000000 | 0.372500 | 29.000000 |
| 75% | 6.000000 | 140.250000 | 80.000000 | 32.000000 | 127.250000 | 38.800000 | 0.626250 | 41.000000 |
| max | 17.000000 | 199.000000 | 122.000000 | 98.000000 | 848.000000 | 67.100000 | 2.420000 | 81.000000 |

menghitung data statistik

Output

```
df.loc[df["Glucose"] == 0.0, "Glucose"] = np.NAN
df.loc[df["BloodPressure"] == 0.0, "BloodPressure"] = np.NAN
df.loc[df["SkinThickness"] == 0.0, "SkinThickness"] = np.NAN
df.loc[df["Insulin"] == 0.0, "Insulin"] = np.NAN
df.loc[df["BMI"] == 0.0, "BMI"] = np.NAN
df.isnull().sum()[1:6]
```

df.loc[] --> memilih tabel berdasarkan lokasi

| | |
|---------------|-------|
| Glucose | 5 |
| BloodPressure | 35 |
| SkinThickness | 227 |
| Insulin | 374 |
| BMI | 11 |
| dtype: | int64 |

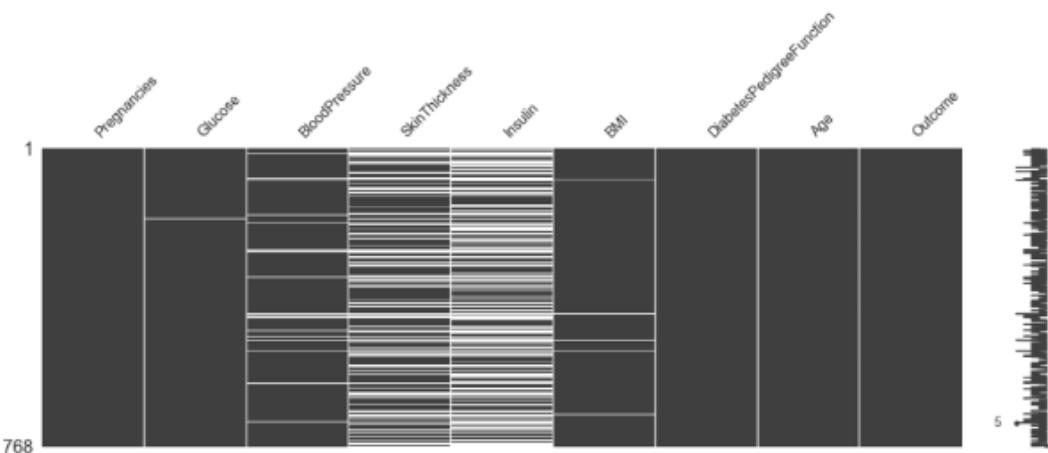
Output



Hands On (Lanjutan)

```
mno.matrix(df, figsize = (20, 6))
```

```
<AxesSubplot:>
```



→ membuat matriks

```
missing_columns = ["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]
```

```
def random_imputation(df, feature):
```

```
    number_missing = df[feature].isnull().sum()
    observed_values = df.loc[df[feature].notnull(), feature]
    df.loc[df[feature].isnull(), feature + '_imp'] = np.random.choice(observed_values, number_missing, replace = True)
    return df
```

```
for feature in missing_columns:
    df[feature + '_imp'] = df[feature]
    df = random_imputation(df, feature)
```

→ Output

Menggunakan Regresi
untuk memperhitungkan
data yang hilang



Hands On (Lanjutan)

```
random_data = pd.DataFrame(columns = ["Ran" + name for name in missing_columns])

for feature in missing_columns:

    random_data["Ran" + feature] = df[feature + '_imp']
    parameters = list(set(df.columns) - set(missing_columns) - {feature + '_imp'})

    model = linear_model.LinearRegression()
    model.fit(X = df[parameters], y = df[feature + '_imp'])
```

Stochastic Regression
Imputation

```
predict = model.predict(df[parameters])
std_error = (predict[df[feature].notnull()] - df.loc[df[feature].notnull(), feature + '_imp']).std()
```

amati bahwa kita menyimpan
indeks data yang hilang dari
kerangka data asli

```
random_predict = np.random.normal(size = df[feature].shape[0],
                                    loc = predict,
                                    scale = std_error)
random_data.loc[(df[feature].isnull()) & (random_predict > 0), "Ran" + feature] = random_predict[(df[feature].isnull()) &
                                                                 (random_predict > 0)]
```

Kesalahan Standar dari perkiraan regresi
sama dengan std() dari kesalahan setiap
perkiraan

Hands On (Lanjutan)

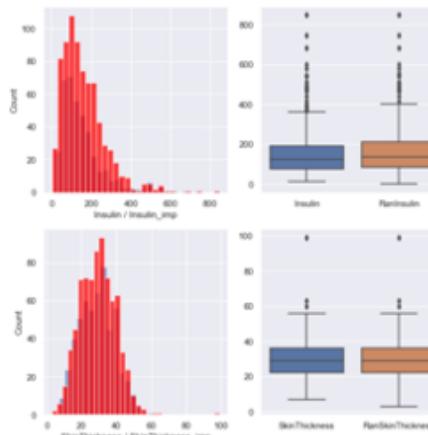
```
sns.set()
fig, axes = plt.subplots(nrows = 2, ncols = 2)
fig.set_size_inches(8, 8)

for index, variable in enumerate(["Insulin", "SkinThickness"]):
    sns.histplot(df[variable].dropna(), kde = False, ax = axes[index, 0])
    sns.histplot(random_data["Ran" + variable], kde = False, ax = axes[index, 0], color = 'red')
    axes[index, 0].set(xlabel = variable + " / " + variable + '_imp')

    sns.boxplot(data = pd.concat([df[variable], random_data["Ran" + variable]]), axis = 1,
                ax = axes[index, 1])

plt.tight_layout()
```

→ membuat chart



→ Output



Hands On (Lanjutan)

```
pd.concat([df[["Insulin", "SkinThickness"]], random_data[["RanInsulin", "RanSkinThickness"]]], axis = 1).describe().T
```

| | count | mean | std | min | 25% | 50% | 75% | max |
|------------------|-------|------------|------------|-----------|-----------|------------|------------|-------|
| Insulin | 394.0 | 155.548223 | 118.775855 | 14.000000 | 76.250000 | 125.000000 | 190.000000 | 846.0 |
| SkinThickness | 541.0 | 29.153420 | 10.476982 | 7.000000 | 22.000000 | 29.000000 | 36.000000 | 99.0 |
| RanInsulin | 768.0 | 159.817877 | 107.962192 | 1.350895 | 82.852612 | 137.134053 | 212.448998 | 846.0 |
| RanSkinThickness | 768.0 | 29.056917 | 10.211228 | 2.984951 | 22.000000 | 29.000000 | 36.000000 | 99.0 |

→ Output

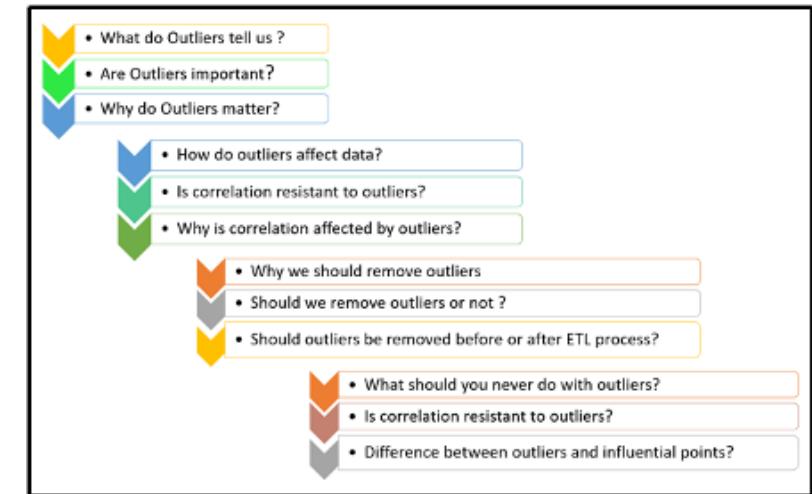


Kesimpulan Teknik Imputasi

- Dua kategori imputasi:
 - Berdasarkan layer/tahapan: single imputation dan multiple imputation
- Tidak ada cara sempurna untuk mengkompensasi missing value (nilai yang hilang) dalam kumpulan data.
- Setiap strategi memiliki kinerja lebih baik untuk kumpulan data tertentu dan tipe data yang hilang tetapi dapat berkinerja jauh lebih buruk pada jenis kumpulan data lainnya.

Mengatasi (Handling) Outlier (HO)

- 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).



sumber gbr:
<https://heartbeat.fritz.ai/hands-on-with-feature-engineering-techniques-dealing-with-outliers-fcc9f57cb63b>
<https://www.analyticsvidhya.com/blog/2021/03/zooming-out-a-look-at-outlier-and-how-to-deal-with-them-in-data-science/>

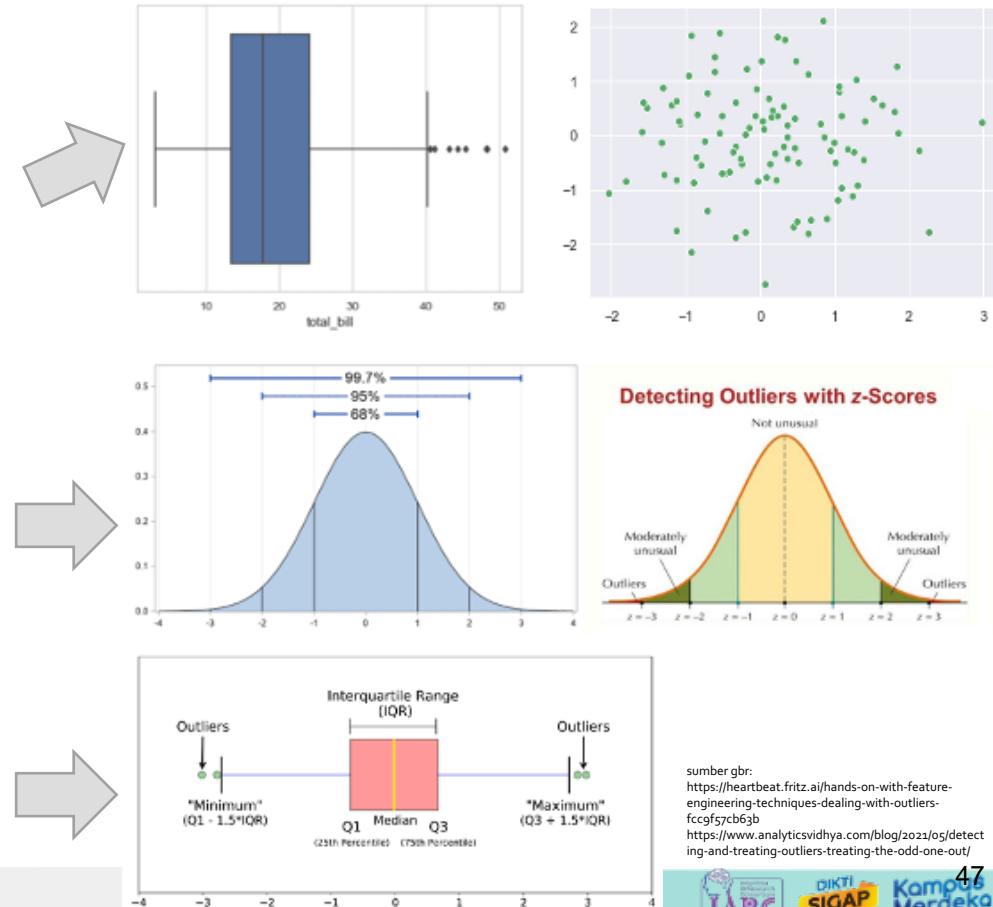


Mengatasi (*Handling*) Outlier (HO)

- Jenis/kategori:
 - Univariate vs multivariate
 - Parametrik vs non-parametrik
 - Deteksi dan Cari Outlier dengan:
 - Visualisasi
 - Distribusi normal
- Teknik Mengatasi Outlier:
- Trimming
 - Winsorizing
 - Imputing
 - Discretization
 - Censoring
 - Z-score
 - Linear Regression Model

Deteksi Outlier

- 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)





Teknik HO: Trimming (Pangkas) vs Winsorizing

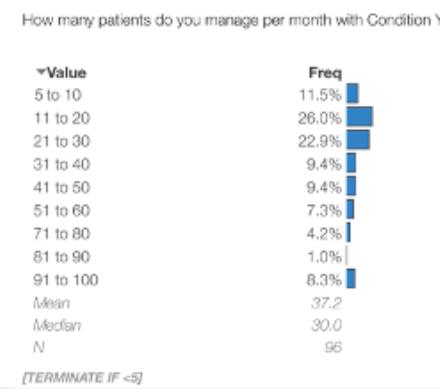
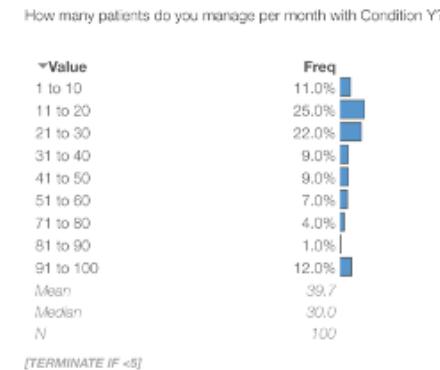
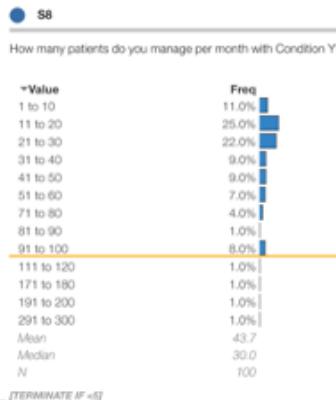
- Nama lain: Truncation (Potong)
- Definisi: Menghapus outlier dari dataset
- Perlu memutuskan metrik untuk menentukan outlier.

Definisi:

Mengganti outlier dari dataset dengan nilai persentil setiap ujung/batas atas dan bawah.

Trimming vs Winsorizing

- Contoh kasus: laporan jumlah pasien yang ditangani tiap dokter/bulan (di bawah 100 pasien), dengan 4% dilaporkan lebih dari 100 pasien.



range [5,100].

Outlier tidak dibuang, namun dimasukan ke dalam range terdekat. Sehingga nilai mean mengecil. Median dan N tidak berubah.

Membuang data yang berada di luar range. Nilai N berubah, mean mengecil, median masih dinilai yg sama.



Hands On

```
import numpy as np
from scipy.stats.mstats import winsorize
from scipy.stats.mstats import trima
```

```
a = np.array([10, 4, 9, 8, 5, 3, 7, 2, 1, 6])
```

```
wins = winsorize(a, limits=[0.1, 0.2])
wins
```

```
masked_array(data=[8, 4, 8, 8, 5, 3, 7, 2, 2, 6],
              mask=False,
              fill_value=999999)
```

```
trims = trima(a, limits=(2,8))
print(trimas)
```

```
[-- 4 -- 8 5 3 7 2 -- 6] → Output
```

→ import numpy dan scipy

→ Masukkan array berisi 1 - 10

→ Winsorize akan mengganti 10% nilai terendah dan 20% nilai tinggi

→ Output

→ Trims akan memotong nilai dengan batas tertentu

Discretization

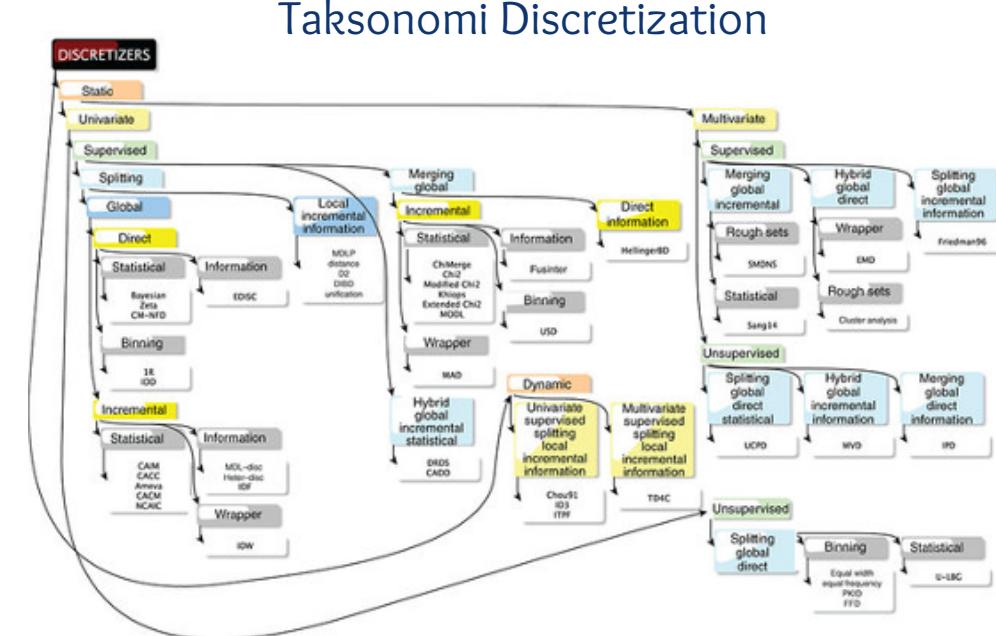
- Definisi:
 - Proses mengubah fungsi, model dan variabel kontinu menjadi diskret (data kontinu di *Ukur* (measured) vs data kontinu di *Hitung* (counted)).
- Nama lain: Binning.
- Dasar Pertimbangan:
 - Data kontinu memiliki derajat kebebasan (DoF) yang tak hingga.
 - Data kontinu lebih mudah dipahami dan disimpan dalam bentuk kategori/grup
 - misal berat badan < 65 kg (ringan); 65 – 80 kg (mid); > 80 kg (berat).



Discretization Process

Discretization

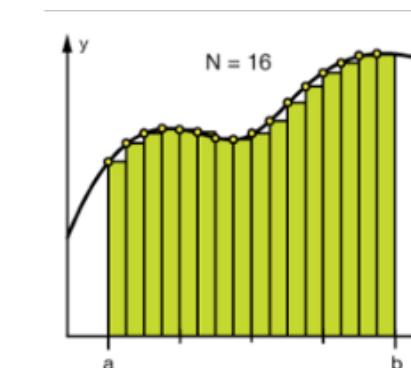
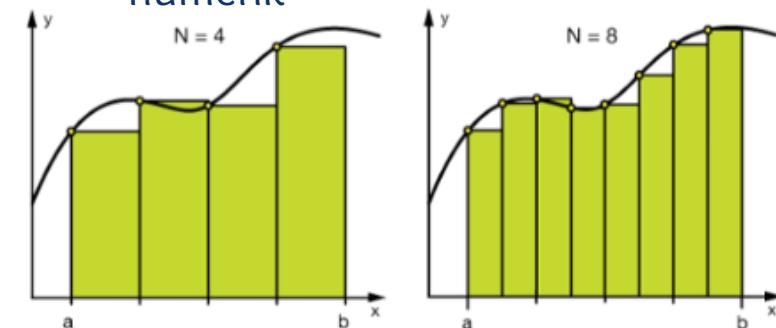
- Jenis:
 - Supervised
 - Decision Tree.
 - Unsupervised
 - Equal-width discretization.
 - Equal-frequency discretization.
 - K-means discretization.
 - Lainnya
 - Custom discretization.



Binning

- Pro:
 - Dapat diterapkan pada data kategorik dan numerik.
 - Model lebih robust dan mencegah overfitting.
- Kontra:
 - Meningkatnya biaya kinerja perhitungan.
 - Mengorbankan informasi.
 - Untuk kolom data numerik, dapat menyebabkan redundansi untuk beberapa algoritma.
 - Untuk kolom data kategorik, label dengan frekuensi rendah berdampak negatif pada robustness model statistik.
 - Untuk ukuran data dengan 100 ribu baris, disarankan menggabungkan label/kolom dengan record yang < 100 menjadi kategori baru, misal “Lain-lain”.

Ilustrasi binning untuk data numerik



Hands On

```
import pandas as pd
import numpy as np

df = pd.read_csv('bins.csv')
df
```

| | Item | Harga |
|----|---------|-------|
| 0 | Item_1 | 1500 |
| 1 | Item_2 | 3300 |
| 2 | Item_3 | 11000 |
| 3 | Item_4 | 87500 |
| 4 | Item_5 | 45000 |
| 5 | Item_6 | 28600 |
| 6 | Item_7 | 39900 |
| 7 | Item_8 | 91000 |
| 8 | Item_9 | 64700 |
| 9 | Item_10 | 6000 |
| 10 | Item_11 | 19000 |
| 11 | Item_12 | 2700 |

→ import pandas, numpy

```
bins = np.linspace(min(df['Harga']), max(df['Harga']), 4)
bins
```

```
array([ 1500. , 31333.3333333, 61166.6666667, 91000. ]) → Output
```

membuat array yang berisi sejumlah angka dengan jarak yang sama dengan linspace(), linspace --> numpy.linspace(nilai minimum, nilai maksimum, jumlah elemen)

→ Output

```
df = pd.read_csv('bins.csv')
kategori = ['Murah', 'Standar', 'Mahal']
df['Harga Binned'] = pd.cut(df['Harga'], bins, labels=kategori, include_lowest=True)
df
```

| | Item | Harga | Harga Binned |
|----|---------|-------|--------------|
| 0 | Item_1 | 1500 | Murah |
| 1 | Item_2 | 3300 | Murah |
| 2 | Item_3 | 11000 | Murah |
| 3 | Item_4 | 87500 | Mahal |
| 4 | Item_5 | 45000 | Standar |
| 5 | Item_6 | 28600 | Murah |
| 6 | Item_7 | 39900 | Standar |
| 7 | Item_8 | 91000 | Mahal |
| 8 | Item_9 | 64700 | Mahal |
| 9 | Item_10 | 6000 | Murah |
| 10 | Item_11 | 19000 | Murah |
| 11 | Item_12 | 2700 | Murah |

- bins --> mengelompokkan data
 - binning dengan cut()
 - membuat kategori harga

→ Output

```
df = pd.read_csv('bins.csv')
interval_range = pd.interval_range(start=0, freq=10000, end=100000)
df['Harga Binned 2'] = pd.cut(df['Harga'], bins = interval_range)
df
```

| | Item | Harga | Harga Binned 2 |
|----|---------|-------|-----------------|
| 0 | Item_1 | 1500 | (0, 10000] |
| 1 | Item_2 | 3300 | (0, 10000] |
| 2 | Item_3 | 11000 | (10000, 20000] |
| 3 | Item_4 | 87500 | (80000, 90000] |
| 4 | Item_5 | 45000 | (40000, 50000] |
| 5 | Item_6 | 28600 | (20000, 30000] |
| 6 | Item_7 | 39900 | (30000, 40000] |
| 7 | Item_8 | 91000 | (90000, 100000] |
| 8 | Item_9 | 64700 | (50000, 70000] |
| 9 | Item_10 | 6000 | (0, 10000] |
| 10 | Item_11 | 19000 | (10000, 20000] |
| 11 | Item_12 | 2700 | (0, 10000] |

menggunakan interval range untuk binning data dengan cut()

→ Output



Hands On (Lanjutan)

```
df = pd.read_csv('bins.csv')
df['Harga Binned 3'] = pd.qcut(df['Harga'], 3)
df
```

→ binning dengan qcut()

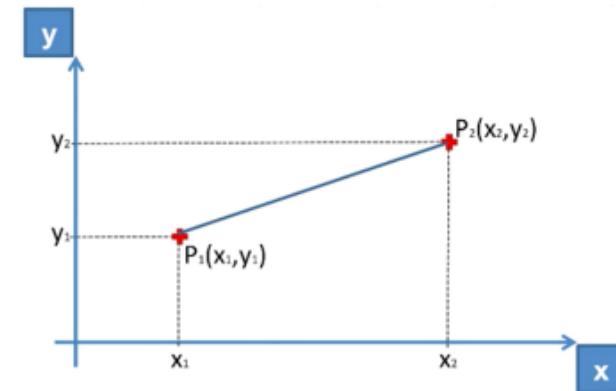
| | Item | Harga | Harga Binned 3 |
|----|---------|-------|----------------------|
| 0 | Item_1 | 1500 | (1499.999, 9333.333] |
| 1 | Item_2 | 3300 | (1499.999, 9333.333] |
| 2 | Item_3 | 11000 | (9333.333, 41600.0] |
| 3 | Item_4 | 87500 | (41600.0, 91000.0] |
| 4 | Item_5 | 45000 | (41600.0, 91000.0] |
| 5 | Item_6 | 28600 | (9333.333, 41600.0] |
| 6 | Item_7 | 39900 | (9333.333, 41600.0] |
| 7 | Item_8 | 91000 | (41600.0, 91000.0] |
| 8 | Item_9 | 64700 | (41600.0, 91000.0] |
| 9 | Item_10 | 6000 | (1499.999, 9333.333] |
| 10 | Item_11 | 19000 | (9333.333, 41600.0] |
| 11 | Item_12 | 2700 | (1499.999, 9333.333] |

→ Output

Scaling (Penskalaan)

- Dasar:

- Sering diabaikan oleh pemula di Data Science.
- Data numerik (biasanya) tidak memiliki range. range “Usia” vs range “Gaji” tidak sama (karakteristik berbeda). *Usia* memiliki rentang dari 1 sampai 150 (dalam tahun), sedangkan *Gaji* memiliki rentang dari 10 ribu sampai 100 ribu (dalam dolar). Untuk itu membandingkan perlu *scaling*.
- Beberapa algoritma Machine Learning (regresi linear dan logistik dan Neural Network; SVM, KNN, K-means; LDA; PCA) yang menggunakan teknik optimasi Euclidian Distance 2 titik.



$$\text{Euclidean Distance between } P_1 \text{ and } P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- Dengan menggunakan rumus Euclidean Distance diatas, maka jelas bahwa hasil perhitungan pada kolom *Usia* dan *Gaji* akan memiliki jarak (distance) yang sangat jauh. Disinilah proses Feature Scaling dibutuhkan.
- Feature Scaling adalah suatu cara untuk membuat numerical data pada dataset memiliki rentang nilai (scale) yang sama. Tidak ada lagi satu variabel data yang mendominasi variabel data lainnya.



Hands On

```
import pandas as pd
from sklearn.preprocessing import Normalizer
```

```
data = pd.read_csv("scalling.csv")
```

```
max_abs = Normalizer(norm = 'l2')
```

```
max_abs.fit(data)
```

```
train_scaled = max_abs.transform(data)
```

```
train_scaled
```

```
array([[6.11110997e-04, 9.99999813e-01],
       [5.62499911e-04, 9.99999842e-01],
       [5.55555470e-04, 9.99999846e-01],
       [6.22950699e-04, 9.99999806e-01],
       [7.99999744e-04, 9.99999680e-01],
       [6.03448166e-04, 9.99999818e-01],
       [5.19230699e-04, 9.99999865e-01],
       [6.07594825e-04, 9.99999815e-01],
       [6.02409529e-04, 9.99999819e-01],
       [5.52238722e-04, 9.99999848e-01]])
```

→ import library

→ membaca data

→ buat objek skalar dengan normalizer

→ sesuaikan scalar dengan data

→ mengubah data

→ Output

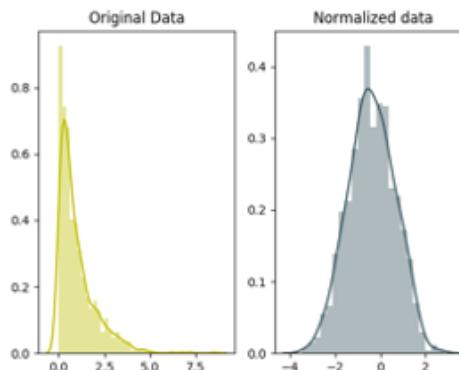
Scaling: Jenis

| Standardisation | Normalisation |
|--|---|
| $x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$ | $x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$ |



Scaling: Normalisasi

- Nama Lain: Min-Max Scaling.
- Definisi: Teknik penskalaan di mana nilai-nilai digeser dan diubah skalanya sehingga nilainya berkisar antara 0 dan 1 (rentang [0,1]).



$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Di sini, X_{max} dan X_{min} masing-masing adalah nilai maksimum dan minimum dari fitur.

- Ketika nilai X adalah nilai minimum dalam kolom, pembilangnya adalah 0, dan karenanya X' adalah 0.
- Sebaliknya, ketika nilai X adalah nilai maksimum dalam kolom, pembilangnya sama dengan penyebutnya sehingga nilai X' adalah 1.
- Jika nilai X berada di antara nilai minimum dan maksimum, maka nilai X' berada di antara 0 dan 1.



Hands On

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

import pandas
dan numpy

```
data = pd.read_csv("scalling.csv")  
data
```

membaca data

| | Age | Income |
|---|-----|--------|
| 0 | 44 | 72000 |
| 1 | 27 | 48000 |
| 2 | 30 | 54000 |
| 3 | 38 | 61000 |
| 4 | 40 | 50000 |
| 5 | 35 | 58000 |
| 6 | 27 | 52000 |
| 7 | 48 | 79000 |
| 8 | 50 | 83000 |
| 9 | 37 | 67000 |

Output

```
means = data.mean(axis = 0)
```

menghitung mean

```
max_min = data.max(axis = 0) - data.min(axis = 0)
```

menghitung
max - min

```
train_scaled = (data - means) / max_min
```

menerapkan
transformasi ke
data

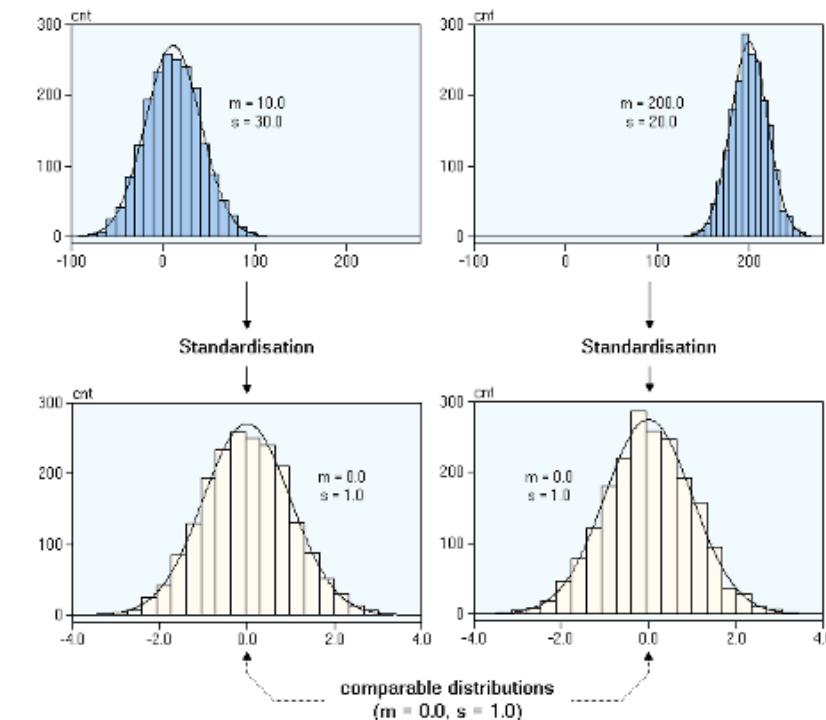
```
train_scaled
```

| | Age | Income |
|---|-----------|-----------|
| 0 | 0.278261 | 0.274286 |
| 1 | -0.460870 | -0.411429 |
| 2 | -0.330435 | -0.240000 |
| 3 | 0.017391 | -0.040000 |
| 4 | 0.104348 | -0.354286 |
| 5 | -0.113043 | -0.125714 |
| 6 | -0.460870 | -0.297143 |
| 7 | 0.452174 | 0.474286 |
| 8 | 0.539130 | 0.588571 |
| 9 | -0.026087 | 0.131429 |

Output

Scaling: Standardisasi

- Tujuan: Berfokus pada mengubah data mentah menjadi informasi yang dapat digunakan sebelum dianalisis.
- Definisi: Teknik yang menskalakan data sehingga memiliki mean = 0 dan standar deviasi = 1
- Kontra:
 - Menambah langkah dalam data preparation
 - Waktu bertambah



Hands On

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

→ import library

```
data = pd.read_csv("scaling.csv") → membaca data
data
```

| | Age | Income |
|---|-----|--------|
| 0 | 44 | 72000 |
| 1 | 27 | 48000 |
| 2 | 30 | 54000 |
| 3 | 38 | 61000 |
| 4 | 40 | 50000 |
| 5 | 35 | 58000 |
| 6 | 27 | 52000 |
| 7 | 48 | 79000 |
| 8 | 50 | 83000 |
| 9 | 37 | 67000 |

→ Output

```
scaler = StandardScaler() → buat objek scaler
```

```
scaler.fit(data) → sesuaikan scaler dengan data
```

```
train_scaled = scaler.transform(data) → mengubah data
kereta dan uji
```

```
train_scaled
```

```
array([[ 0.8273403 ,  0.81886943],
       [-1.37028238, -1.22830415],
       [-0.98246661, -0.71651075],
       [ 0.05170877, -0.11941846],
       [ 0.31025261, -1.05770635],
       [-0.336107 , -0.37531516],
       [-1.37028238, -0.88710855],
       [ 1.34442799,  1.41596173],
       [ 1.60297184,  1.75715732],
       [-0.07756315,  0.39237494]]) → Output
```

Contoh Kasus Scaling

| | Country | Age | Salary | Purchased |
|----|---------|-----|--------|-----------|
| 1 | France | 44 | 72000 | No |
| 2 | Spain | 27 | 48000 | Yes |
| 3 | Germany | 30 | 54000 | No |
| 4 | Spain | 38 | 61000 | No |
| 5 | Germany | 40 | | Yes |
| 6 | France | 35 | 58000 | Yes |
| 7 | Spain | | 52000 | No |
| 8 | France | 48 | 79000 | Yes |
| 9 | Germany | 50 | 83000 | No |
| 10 | France | 37 | 67000 | Yes |

The range of Age: 27 - 50

The range of Salary: 48,000 - 83,000

```
dataset['Age'].min()
```

27.0

```
dataset['Salary'].min()
```

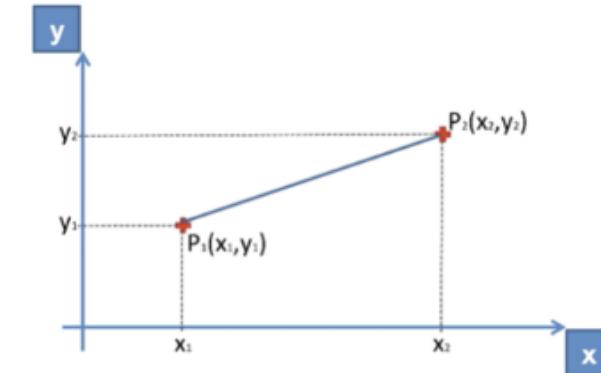
48000.0

```
dataset['Age'].max()
```

50.0

```
dataset['Salary'].max()
```

83000.0



$$\text{Euclidean Distance between } P_1 \text{ and } P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Let x be the no. of Salary and y be the no. of Age

Example: x1&y1 are in row 2, x2&y2 are in row 9

$$(x_2 - x_1)^2 = (83000 - 48000)^2$$

$$= 1225000000$$

$$(y_2 - y_1)^2 = (50 - 27)^2$$

$$= 529$$



Contoh Kasus Scaling

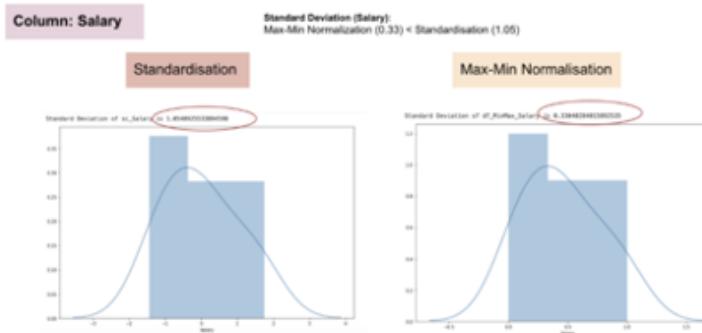
- Ketika kita menghitung persamaan jarak (distance) Euclidean, jumlah $(x_2-x_1)^2$ jauh lebih besar daripada jumlah $(y_2-y_1)^2$ yang berarti jarak Euclidean akan didominasi oleh Gaji jika kita tidak menerapkan penskalaan. Perbedaan Usia berkontribusi lebih sedikit terhadap perbedaan keseluruhan.
- Oleh karena itu, kita harus menggunakan penskalaan untuk membawa semua nilai ke besaran yang sama dan dengan demikian, menyelesaikan masalah ini.

| Standardisation | | Max-Min Normalization | |
|-----------------|-----------|-----------------------|-------------------|
| | | Age | Salary |
| 0 | 0.758874 | 7.494733e-01 | 0.739130 0.685714 |
| 1 | -1.711504 | -1.438178e+00 | 0.000000 0.000000 |
| 2 | -1.275555 | -8.912655e-01 | 0.130435 0.171429 |
| 3 | -0.113024 | -2.532004e-01 | 0.478261 0.371429 |
| 4 | 0.177609 | 6.632192e-16 | 0.565217 0.450794 |
| 5 | -0.548973 | -5.266569e-01 | 0.347826 0.285714 |
| 6 | 0.000000 | -1.073570e+00 | 0.512077 0.114286 |
| 7 | 1.340140 | 1.387538e+00 | 0.913043 0.885714 |
| 8 | 1.630773 | 1.752147e+00 | 1.000000 1.000000 |
| 9 | -0.258340 | 2.937125e-01 | 0.434783 0.542857 |



Contoh Kasus Scaling

After Feature scaling.

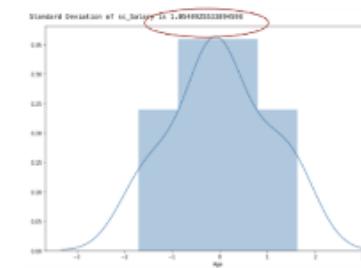


Normal distribution and Standard Deviation of Salary.

Column:Age

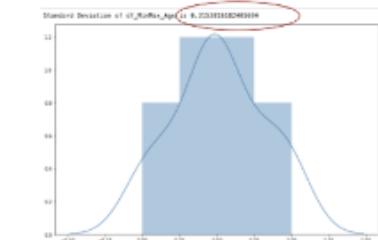
Standard Deviation (Age):
Max-Min Normalization (0.315) < Standardisation (1.06)

Standardisation



Normal distribution and Standard Deviation of Age.

Max-Min Normalisation



sumber gbr:
<https://www.kdnuggets.com/2020/04/data-transformation-standardization-normalization.html>

Dokumentasi Fitur





Perlunya Dokumentasi Data/Fitur

Dokumentasi data

dapat **menjembatani kesenjangan** antara **transaksi** (pembuatan data) dan **analisis** (konsumsi data). Dokumentasi data yang baik memungkinkan pengguna, ataupun rekan tim untuk memahami siapa/apa/kapan/di mana/bagaimana/mengapa data tersebut dibentuk ataupun dikonsumsi.



Parameter/Daftar Isi Dokumentasi Data Transformation

Laporan dokumentasi data transformation, setidaknya memiliki parameter berikut:

- Fitur awal dan rekayasa fitur yang digunakan
- Teknik transformasi data yang diterapkan
 - Apakah algoritma pemodelan mengharapkan jenis data tertentu, seperti numerik? Jika demikian, lakukan transformasi yang diperlukan
 - Apakah data perlu dinormalisasi sebelum pemodelan?
 - Bisakah atribut yang hilang dibangun menggunakan agregasi, rata-rata, atau induksi?
- Hasil transformasi
- Rekomendasi transformasi

Pelabelan Data



Referensi: SKKNI Data Science

KODE UNIT : **J.62DMI00.010.1**

JUDUL UNIT : **Menentukan Label Data**

DESKRIPSI UNIT: Unit kompetensi ini berhubungan dengan pengetahuan, keterampilan, dan sikap kerja yang dibutuhkan untuk menentukan label data untuk pembangunan model *data science*

| ELEMEN KOMPETENSI | KRITERIA UNJUK KERJA |
|---|--|
| 1. Melakukan pelabelan data | 1.1 Analisis hasil pelabelan data sejenis yang sudah ada diuraikan kesesuaianya dengan Standard Operating Procedure (SOP) pelabelan . 1.2 Pelabelan data dilakukan sesuai dengan SOP pelabelan. |
| 2. Membuat laporan hasil pelabelan data | 2.1. Statistik hasil pelabelan diuraikan pada laporan. 2.2. Evaluasi proses pelabelan diuraikan pada laporan. |

BATASAN VARIABEL

1. Konteks variabel
 - 1.1 Pelabelan data adalah proses memberikan label pada data yang akan digunakan pada pemodelan *machine learning*.
 - 1.2 *Standard Operating Procedure (SOP)* pelabelan adalah panduan langkah-langkah dan aturan dalam melakukan proses pelabelan data sesuai dengan domain data.



Pelabelan Data - Intro

- **Kuantitas & kualitas data pelatihan** yang secara langsung **menentukan keberhasilan** suatu **algoritma AI** sehingga tidak mengherankan jika rata-rata 80% waktu yang dihabiskan untuk proyek AI membahas data pelatihan yang mencakup proses **pelabelan data**.
- **Keakuratan model AI** Anda berkorelasi langsung dengan kualitas data yang digunakan untuk melatihnya.
- Hal ini menjadi satu alasan mengapa **proses pelabelan data merupakan bagian integral dari alur kerja persiapan data dalam membangun model AI yang andal**.

Pelabelan Data - Intro

Pelabelan data dalam konteks pembelajaran mesin adalah proses mendeteksi serta menandai sampel data.

- Tahapan proses ini menjadi sangat penting dalam hal pembangunan model dengan pendekatan pembelajaran mesin berbasiskan supervised-learning.
- Pelabelan Data mengacu pada proses menambahkan tag atau label pada data masukan yang berbentuk *gambar, video, teks, dan audio*.
- Tag ini membentuk representasi dari kelas objek apa yang dimiliki data dan membantu model pembelajaran mesin untuk mengidentifikasi kelas objek tertentu saat ditemui dalam data tanpa tag.
- Secara umum, pelabelan data dapat merujuk pada tugas yang mencakup penandaan data, anotasi, klasifikasi, moderasi, transkripsi atau pemrosesan.



Hands-on : OneHotEncoding (Mempersiapkan data)

```
## Import library
import random
import pandas as pd
import numpy as np

# Membaca data mentah (raw data)
data = pd.read_csv('crx.data', header=None)

# Membentuk list Fitur A1..A17
varnames = ['A'+str(s) for s in range(1,17)]

# Men-set masing-masing kolom dari data yang ada
# Hal ini dilakukan karena data mentah yang dipakai tidak memiliki judul fitur
data.columns = varnames

# Mengganti data dengan nilai ? dengan nilai Nan
data = data.replace('?', np.nan)
```

- Import library yang dibutuhkan
- memuat data
- membentuk list
- set masing-masing kolom
- mengganti "?" menjadi nan



Hands-on : OneHotEncoding (Mempersiapkan data)

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

→ Import library yang dibutuhkan

```
data = pd.read_csv('crx.data', header=None)
```

→ memuat data

```
varnames = ['A'+str(s) for s in range(1,17)]
```

→ membentuk list

```
data.columns = varnames
```

→ set masing-masing kolom

```
data = data.replace('?', np.nan)
```

→ mengganti "?" menjadi nan



Hands-on : OneHotEncoding (Mempersiapkan data)

```
data['A2'] = data['A2'].astype('float')
data['A14'] = data['A14'].astype('float')
```



merubah tipe data dari kolom A2 dan A14 menjadi tipe float

```
data['A16'] = data['A16'].map({'+':1, '-':0})
```



melakukan proses mapping data text menjadi menjadi bentuk binary

```
cat_cols = [c for c in data.columns if data[c].dtypes=='O']
num_cols = [c for c in data.columns if data[c].dtypes!='O']
```



membuat list dengan fitur data dengan jenis kategori dan jenis numerik

```
data[num_cols] = data[num_cols].fillna(0)
data[cat_cols] = data[cat_cols].fillna('Missing')
```



melakukan inputasi terhadap data yang kosong dengan nilai 0 untuk list data bernilai numerik Missing untuk data kolom fitur berjenis kategori

```
data.to_csv('creditApprovalUCI.csv', index=False)
```



menyimpan data yang telah di transformasi dalam bentuk csv



Hands-on : OneHotEncoding (Penggunaan Teknik)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
```



import library

```
data = pd.read_csv('creditApprovalUCI.csv')
data.head()
```



Membaca data yang telah di transformasi sebelumnya

| | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | A12 | A13 | A14 | A15 | A16 |
|---|----|-------|-------|----|----|----|----|------|----|-----|-----|-----|-----|-------|-----|-----|
| 0 | b | 30.83 | 0.000 | u | g | w | v | 1.25 | t | t | 1 | f | g | 202.0 | 0 | 1 |
| 1 | a | 58.67 | 4.460 | u | g | q | h | 3.04 | t | t | 6 | f | g | 43.0 | 560 | 1 |
| 2 | a | 24.50 | 0.500 | u | g | q | h | 1.50 | t | f | 0 | f | g | 280.0 | 824 | 1 |
| 3 | b | 27.83 | 1.540 | u | g | w | v | 3.75 | t | t | 5 | t | g | 100.0 | 3 | 1 |
| 4 | b | 20.17 | 5.625 | u | g | w | v | 1.71 | t | f | 0 | f | s | 120.0 | 0 | 1 |



Hands-on : OneHotEncoding (Penggunaan Teknik)

```
X_train, X_test, y_train, y_test = train_test_split( data.drop(labels=['A16'], axis=1), data['A16'], test_size=0.3,  
random_state=8)
```



Memisahkan data yang akan digunakan sebagai testing site

```
X_train['A4'].unique()  
array(['u', 'y', 'Missing', 'l'], dtype=object)
```



Melakukan pengecekan nilai apa saja yang ada pada kolom A4

```
tmp = pd.get_dummies(X_train['A4'], drop_first=True)  
tmp.head()
```



mengacuhkan nilai 'missing' dan menampilkan data

| | l | u | y |
|-----|---|---|---|
| 596 | 0 | 1 | 0 |
| 303 | 0 | 1 | 0 |
| 204 | 0 | 0 | 1 |
| 351 | 0 | 0 | 1 |
| 118 | 0 | 1 | 0 |



Hands-on : OneHotEncoding (Penggunaan Teknik)

```
vars_categorical = ['A1', 'A4', 'A5', 'A6', 'A7', 'A9', 'A10', 'A12', 'A13']
```

→ membentuk list dari masing-masing kolom

```
X_train_enc = pd.get_dummies(X_train[vars_categorical], drop_first=True)  
X_test_enc = pd.get_dummies(X_test[vars_categorical], drop_first=True)
```

→ menangkap nama kategori untuk mengidentifikasi variabel biner yang dihasilkan

```
X_train_enc.head()
```

| | A1_a | A1_b | A4_I | A4_u | A4_y | A5_g | A5_gg | A5_p | A6_aa | A6_c | ... | A7_j | A7_n | A7_o | A7_v | A7_z | A9_t | A10_t | A12_t | A13_p | A13_s |
|-----|------|------|------|------|------|------|-------|------|-------|------|-----|------|------|------|------|------|------|-------|-------|-------|-------|
| 596 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 303 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 204 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | ... | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 351 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 118 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |

5 rows x 36 columns

```
encoder = OneHotEncoder(categories='auto', drop='first', sparse=False)
```

→ Melakukan proses encoding

```
encoder.fit(X_train[vars_categorical])
```

```
OneHotEncoder(drop='first', sparse=False)
```



Pelabelan Data - Intro

Pelabelan data adalah bagian utama dari alur kerja pra pemrosesan data untuk machine learning. Data berlabel ini kemudian digunakan untuk melatih model pembelajaran mesin untuk menemukan "makna" dalam data baru yang serupa dan relevan.

Pelabelan data menyusun data untuk membuatnya bermakna.

Sepanjang proses ini, data scientist berusaha keras untuk memperoleh kualitas dan kuantitas yang baik.

Label yang lebih akurat ditambah dengan jumlah data berlabel yang lebih besar menciptakan model pembelajaran mendalam yang lebih berguna,

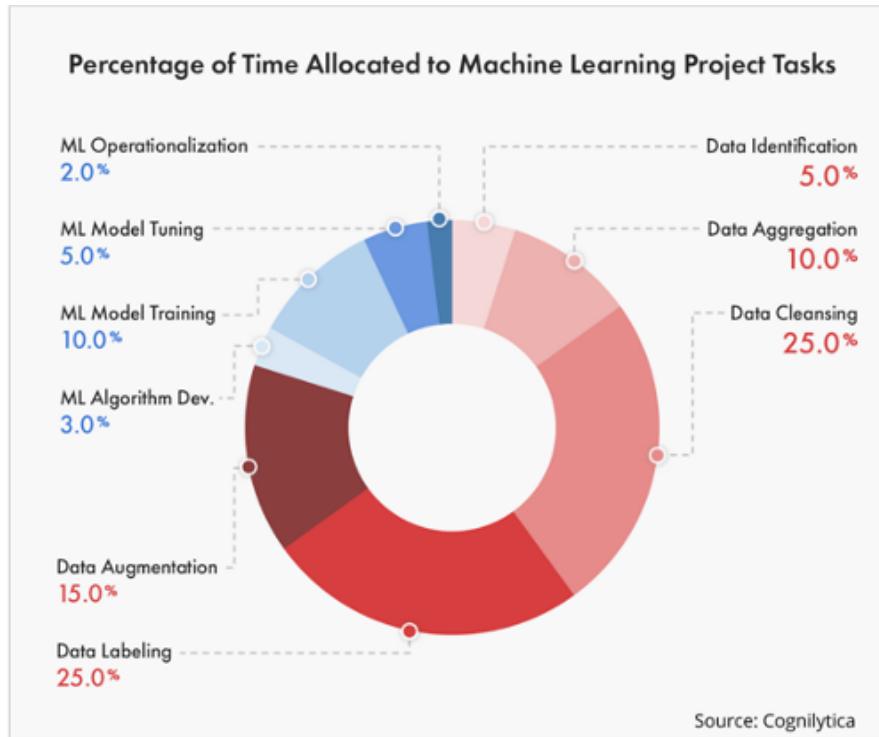
karena model pembelajaran mesin yang dihasilkan mendasarkan keputusan mereka pada semua data berlabel.

Pelabelan Data - Data Training

Data pelatihan mengacu pada data yang telah dikumpulkan untuk diumpulkan ke model pembelajaran mesin untuk membantu model mempelajari data lebih lanjut.

- Data pelatihan dapat berupa berbagai bentuk, termasuk gambar, suara, teks, atau fitur tergantung pada model pembelajaran mesin yang digunakan dan tugas ataupun business goal yang ingin dicapai.
- Data pelatihan bisa diberi anotasi maupun tidak diberi anotasi.
- Ketika data pelatihan dianotasi, label tersebut disebut sebagai dasar kebenaran atau ***Ground Truth*** - istilah digunakan untuk informasi yang telah diketahui sebelumnya bernilai benar.

Pelabelan Data - Mengapa Proses Pelabelan Diperlukan ?



- Anda memiliki banyak data yang tidak berlabel.
- Sebagian besar data tidak dalam bentuk berlabel, hal merupakan tantangan bagi sebagian besar tim proyek Data Science.
- Sepenuhnya 80% dari waktu proyek berbasis AI dihabiskan untuk mengumpulkan, mengatur, dan memberi label data,
- menurut firma analis Cognilytica, dan ini adalah waktu yang tidak dapat dihabiskan oleh tim karena mereka berlomba untuk mendapatkan data yang dapat digunakan, yaitu data yang terstruktur dan diberi label dengan benar untuk melatih dan menerapkan model.



Pelabelan Data - Unlabelled vs Labelled

- Dataset pelatihan bergantung pada jenis permasalahan ataupun tujuan model pembelajaran mesin yang ingin kita bentuk.
- Algoritma Machine/Deep Learning dapat secara luas diklasifikasikan berdasarkan jenis data yang mereka butuhkan dalam tiga tipe, yaitu :
 - Supervised Learning
 - Unsupervised Learning
 - Semi-supervised Learning

Pelabelan Data - Supervised Learning

- Pembelajaran terawasi, jenis yang paling umum, adalah jenis algoritma pembelajaran mesin yang memerlukan data dan label yang sesuai untuk dilatih.
- Pendekatan ini biasanya digunakan untuk menyelesaikan permasalahan **klasifikasi** dan **segmentasi**.
- Prosedur pelatihan tipikal terdiri dari memasukkan data yang telah beranotasi ke mesin untuk membantu model belajar, dan kemudian melakukan pengujian model yang terbentuk pada data yang tak beranotasi.
- Untuk menemukan keakuratan metode tersebut, data beranotasi (ground truth) dengan label tersembunyi biasanya digunakan dalam tahap pengujian algoritma.
- Dengan demikian, data beranotasi merupakan kebutuhan mutlak untuk melatih model pembelajaran mesin secara terawasi.



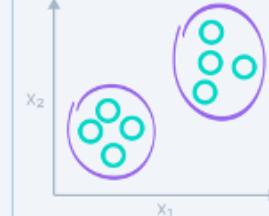
Pelabelan Data - Unsupervised Learning

- Dalam pembelajaran tanpa pengawasan, data input merupakan data tanpa anotasi dan model berlatih tanpa pengetahuan tentang label yang mungkin dimiliki data input.
- Algoritma unsupervised termasuk autoencoder yang memiliki output yang sama dengan inputnya.
- Metode pembelajaran tanpa pengawasan juga mencakup algoritma pengelompokan yang mengelompokkan data ke dalam cluster 'n', di mana 'n' adalah hyperparameter.



Pelabelan Data - Supervised vs Unsupervised

| Supervised learning | Unsupervised learning |
|--|--|
| Input data is labeled | Input data is unlabeled |
| Has a feedback mechanism | Has no feedback mechanism |
| Data is classified based on the training dataset | Assigns properties of given data to classify it |
| Divided into Regression & Classification | Divided into Clustering & Association |
| Used for prediction | Used for analysis |
| Algorithms include: decision trees, logistic regressions, support vector machine | Algorithms include: k-means clustering, hierarchical clustering, apriori algorithm |
| A known number of classes | A unknown number of classes |





Pelabelan Data - Semi-supervised Learning

- Dalam pembelajaran semi-diajari, kombinasi data beranotasi dan tidak beranotasi digunakan untuk melatih model.
- Meskipun hal ini mengurangi biaya anotasi data dengan menggunakan kedua jenis data tersebut, pada umumnya pendekatan ini menggunakan banyak asumsi pada data pelatihan yang digunakan untuk membangun model.
- Kasus penggunaan pembelajaran semi-diajari salah satunya klasifikasi urutan Protein dan analisis konten Internet.



Pelabelan Data - Semi-supervised Learning

Dataset awal :

Eclipse



Non- eclipse



Dataset yang telah
diperkaya :





Pelabelan Data - Semi-supervised Learning

Dataset yang telah dilabeli secara manual



Setelah menjalankan algoritma supervised learning pada studi kasus ini, hasil model kedua **pasti** mengungguli model pertama yang dibangun hanya berisi dua gambar sebagai data pelatihan.

Tetapi pendekatan ini hanya berlaku untuk tujuan kecil karena anotasi/pelabelan manual ke kumpulan data besar bisa sangatlah sulit dan mahal.

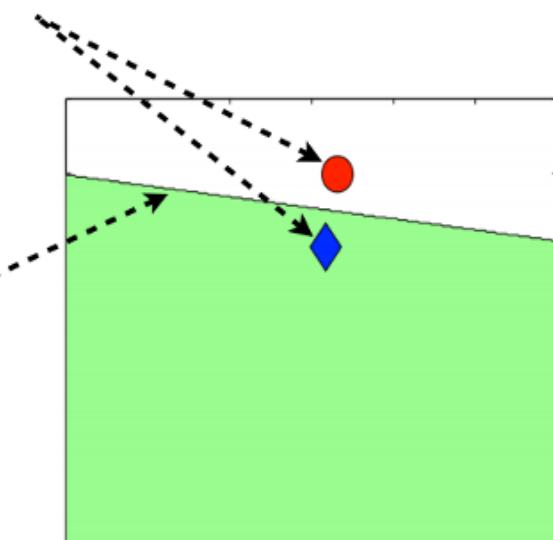
untuk memecahkan jenis masalah ini, terdapat jenis pembelajaran yang berbeda yang dikenal sebagai **semi-supervised learning**, yang menggunakan baik data berlabel (pembelajaran terbimbing) dan data tidak berlabel (pembelajaran tanpa pengawasan).



Pelabelan Data - Peran data tanpa label

Pada pembangunan model dengan data berlabel, Anda hanya memiliki dua titik data yang termasuk dalam dua kategori berbeda, dan garis yang ditarik adalah batas keputusan dari setiap model yang diawasi.

Labeled Instances
Decision Boundary

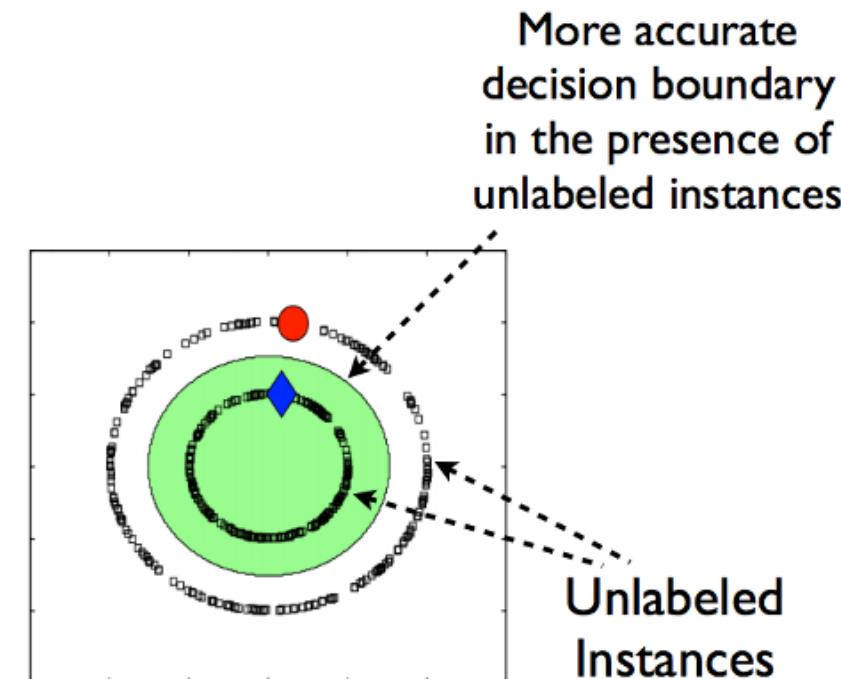




Pelabelan Data - Peran data tanpa label

Selanjutnya, katakanlah kita menambahkan beberapa data yang tidak berlabel ke data ini seperti yang ditunjukkan pada gambar samping.

Gambar ini (di kanan), garis pembatas antara wilayah hijau dan putih menjadi lebih presisi. Perbedaan garis pembatas tersebut menunjukkan bahwa dengan memberikan menambahkan data yang tidak berlabel, garis batas keputusan model menjadi lebih akurat.





Pelabelan Data - Peran data tanpa label

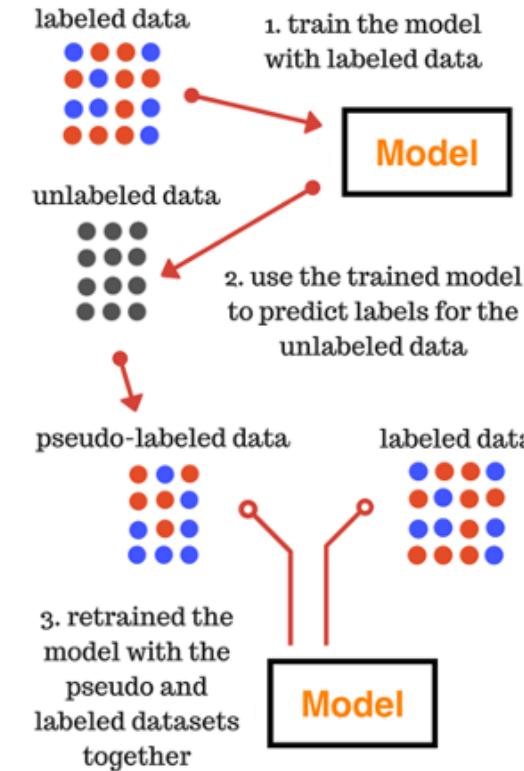
Contoh kasus

Keuntungan menggunakan data tidak berlabel adalah:

- Data berlabel mahal dan sulit didapat sedangkan data tidak berlabel berlimpah dan murah.
- Meningkatkan ketahanan model dengan batas keputusan yang lebih tepat.

Pseudo Labelling

- Manusia tidak hanya belajar dari informasi tetapi mampu memahami suatu berdasarkan kesamaan karakteristik yang dimiliki
- Bisakah kita membangun sistem yang membutuhkan pengawasan minimal yang dapat mempelajari sebagian besar tugas sendiri?
- Terdapat berbagai teknik penerapan semi-Supervised Learning, salah satu tekniknya adalah Teknik Pelabelan Pseudo atau pseudo-labelling.



Ref :

[4] Q. Xie, M.-T. Luong, E. Hovy, and Q.V. Le, “Self-training with noisy student improves ImageNet classification,” arXiv:1911.04252, 2020.

[5] I.Z. Yalniz, H. Jégou, K. Chen, M. Paluri, and D. Mahajan, “Billion-scale semi-supervised learning for image classification,” arXiv:1905.00546, 2019.

[6] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E.D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel, “FixMatch: Simplifying semi-supervised learning with consistency and confidence,” arXiv:2001.07685, 2020.



Human in The Loop (HITL)

Istilah Human-In-The-Loop paling sering mengacu pada pengawasan konstan dan validasi hasil model AI oleh manusia.

Ada dua cara utama di mana manusia menjadi bagian dari loop Machine Learning:

- Memberi label pada data pelatihan: Seorang anotator (*experts*) diwajibkan untuk memberi label pada data pelatihan yang diumpulkan ke model pembelajaran mesin (diawasi/semi-diawasi).
- Melatih model: Data scientist melatih model dengan terus-menerus mengawasi detail model seperti fungsi kerugian (*loss function*) dan hasil prediksi.
- Terkadang kinerja model dan prediksi divalidasi oleh manusia dan hasil validasi diumpulkan kembali ke model.



Pendekatan Pelabelan Data

- Pendekatan pelabelan bergantung pada **pernyataan masalah**, **kerangka waktu proyek**, dan **jumlah orang** yang terkait dengan pekerjaan.
- Pelabelan internal dan crowdsourcing sangat umum, terminologi ini juga dapat mencakup bentuk-bentuk baru pelabelan baru dan anotasi yang memanfaatkan AI dan pembelajaran aktif (*active learning*) untuk melakukan tugas pelabelan/anotasi tersebut.
- Pendekatan yang paling umum untuk anotasi data tercantum di bawah ini
 - In-house data labelling
 - Crowdsourcing
 - Outsourcing
 - Machine-based annotation

In-house Labelling

- Memiliki kualitas tertinggi dan umumnya dilakukan oleh data scientist dan insinyur data yang dipekerjaan di organisasi.
- Pelabelan berkualitas tinggi sangat penting untuk industri seperti asuransi atau perawatan kesehatan, dan seringkali memerlukan konsultasi dengan para ahli di bidang terkait untuk pelabelan data yang tepat.
- Seperti yang diharapkan untuk pelabelan internal, dengan **peningkatan kualitas anotasi, waktu yang dibutuhkan untuk membuat anotasi cukup tinggi**, sehingga seluruh proses pelabelan dan pembersihan data menjadi sangat lambat.

Crowdsourcing

- Crowdsourcing mengacu pada proses **memperoleh data beranotasi dengan bantuan sejumlah besar pekerja lepas yang terdaftar di platform crowdsourcing**.
- Kumpulan data yang dianotasi sebagian besar terdiri dari data sepele seperti gambar hewan, tumbuhan, dan lingkungan alam dan tidak memerlukan keahlian tambahan.
- Oleh karena itu, tugas membuat **anotasi pada kumpulan data sederhana** sering kali dilakukan secara crowdsource ke platform yang memiliki puluhan ribu annotator data terdaftar.

Outsourcing Labelling

- Outsourcing adalah jalan tengah antara crowdsourcing dan pelabelan data internal di mana tugas anotasi data dialihdayakan ke organisasi atau individu.
- Salah satu keuntungan outsourcing adalah kita dapat menilai topik tertentu sebelum pekerjaan diserahkan.
- Pendekatan membangun kumpulan data anotasi ini sangat cocok untuk proyek yang tidak memiliki banyak dana, namun membutuhkan kualitas anotasi data yang signifikan.

Machine-based Annotation

- Salah satu bentuk anotasi yang paling baru adalah anotasi berbasis mesin.
- Anotasi berbasis mesin mengacu pada penggunaan alat anotasi dan otomatisasi yang secara drastis dapat meningkatkan kecepatan anotasi data tanpa mengorbankan kualitas hasil pelabelan.
- Perkembangan otomatisasi baru-baru ini dalam alat anotasi mesin tradisional—menggunakan algoritma pembelajaran mesin yang tidak diawasi dan semi-diawasi—membantu secara signifikan mengurangi beban kerja pada pemberi label manusia.

Algoritma tanpa pengawasan (*unsupervised learning*) seperti pengelompokan dan algoritma *semi-supervised* yang baru banyak dikembangkan untuk melakukan proses pelabelan data AI—seperti pembelajaran aktif (*active learning*) adalah salah satu pendekatan yang dapat mengurangi waktu anotasi hingga batas tertentu.

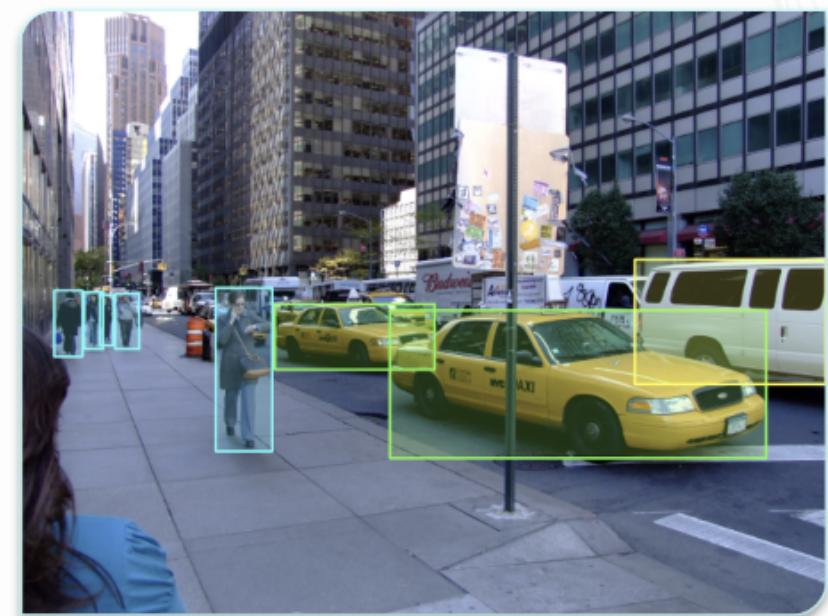


Kelebihan dan Kekurangan Masing-masing Metode

| Approach | Description | Pros | Cons |
|-----------------------------------|---|---|--|
| Internal labeling | Assignment of tasks to an in-house data science team | <ul style="list-style-type: none">✓ Predictable results✓ High accuracy of labeled data✓ The ability to track progress | <ul style="list-style-type: none">✗ It takes much time |
| Outsourcing | Recruitment of temporary employees on freelance platforms, posting vacancies on social media and job search sites | <ul style="list-style-type: none">✓ The ability to evaluate applicants' skills | <ul style="list-style-type: none">✗ The need to organize workflow |
| Crowdsourcing | Cooperation with freelancers from crowdsourcing platforms | <ul style="list-style-type: none">✓ Cost savings✓ Fast results | <ul style="list-style-type: none">✗ Quality of work can suffer |
| Specialized outsourcing companies | Hiring an external team for a specific project | <ul style="list-style-type: none">✓ Assured quality | <ul style="list-style-type: none">✗ Higher price compared to crowdsourcing |
| Synthetic labeling | Generating data with the same attributes of real data | <ul style="list-style-type: none">✓ Fewer constraints for using sensitive and regulated data✓ Training data without mismatches and gaps✓ Cost- and time-effectiveness | <ul style="list-style-type: none">✗ High computational power required |
| Data programming | Using scripts that programmatically label data to avoid manual work | <ul style="list-style-type: none">✓ Automation✓ Fast results | <ul style="list-style-type: none">✗ Lower quality dataset |



Pelabelan data : Pengolahan Citra



Pelabelan data : Pengolahan Citra

Permasalahan pengolahan citra memerlukan data visual beranotasi dalam bentuk gambar. Anotasi data dalam pengolahan citra bergantung pada tugas visual yang kita inginkan untuk dilakukan oleh model.

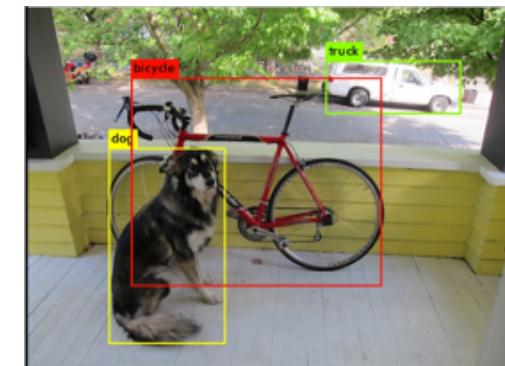
Jenis anotasi data umum pada kasus pengolahan citra antara lain :

- ***Image Classification*** - klasifikasi gambar
 - Anotasi data untuk klasifikasi gambar memerlukan penambahan tag ke gambar yang sedang dikerjakan.
 - Jumlah tag unik di seluruh database adalah jumlah kelas yang dapat diklasifikasi oleh model.
 - Masalah klasifikasi dapat dibagi lagi menjadi:
 - Klasifikasi kelas biner (yang hanya terdiri dari dua tag)
 - Klasifikasi multiclass (yang berisi beberapa tag)
 - Selain itu, klasifikasi multi-label juga dapat dilihat, terutama dalam hal deteksi penyakit, dan mengacu pada setiap gambar yang memiliki lebih dari satu tag.
- ***Image Segmentation*** - segmentasi gambar
 - Dalam Segmentasi Gambar, tugas algoritma pengolahan citra (*Computer Vision*) adalah memisahkan objek dalam gambar dari latar belakangnya dan objek lain dalam gambar yang sama.
 - Ini umumnya berarti peta piksel dengan ukuran yang sama dengan gambar yang berisi 1 di mana objek ada dan 0 di mana anotasi belum dibuat.
 - Untuk beberapa objek yang akan disegmentasi dalam gambar yang sama, peta piksel untuk setiap objek digabungkan berdasarkan saluran dan digunakan sebagai kebenaran dasar untuk model.

Pelabelan data : Pengolahan Citra

- **Object Detection - Deteksi obyek**

- Deteksi Objek mengacu pada deteksi objek dan lokasinya melalui pengolahan citra.
- Anotasi data dalam deteksi objek sangat berbeda dari yang ada di Klasifikasi Gambar, dengan setiap objek dianotasi menggunakan kotak pembatas (*bounding box*).
- Kotak pembatas adalah segmen persegi panjang terkecil yang berisi objek dalam gambar.
- Anotasi kotak pembatas biasanya disertai dengan tag di mana setiap kotak pembatas diberi label pada gambar.
- Umumnya, koordinat kotak pembatas ini dan tag yang sesuai untuknya disimpan dalam file JSON terpisah dalam format kamus dengan nomor gambar/ID gambar menjadi kunci kamus.



- **Pose Estimation - Estimasi pose**

- Estimasi pose mengacu pada penggunaan alat Computer Vision untuk memperkirakan pose seseorang dalam sebuah gambar.
- Estimasi pose berjalan dengan mendeteksi titik-titik kunci dalam tubuh dan menghubungkan titik-titik kunci ini untuk mendapatkan pose.
- **Ground Truth (GT)** yang sesuai untuk model estimasi pose, menjadi poin kunci dari sebuah gambar.
- **Ground Truth (GT)** dapat berupa data koordinat sederhana yang diberi label dengan bantuan tag, di mana setiap koordinat memberikan lokasi titik kunci tertentu, yang diidentifikasi oleh tag, pada gambar masing-masing.





Hands-on : Segmentasi

```
## import library

from skimage.color import rgb2gray
import numpy as np
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
from scipy import ndimage
from PIL import Image
from sklearn.cluster import KMeans
from skimage.filters import sobel
import skimage.segmentation
import skimage
import warnings
warnings.filterwarnings("ignore")
```



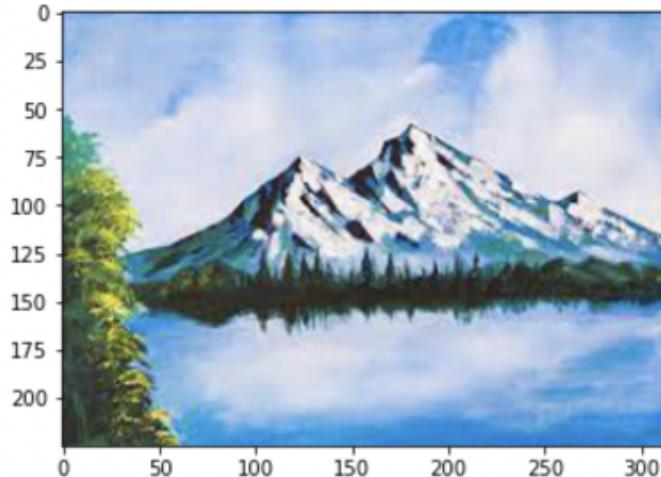
Import library yang dibutuhkan



Hands-on : Segmentasi

```
image=Image.open('mountain.jpeg')
image=image.resize((320,225))
image=np.array(image)
plt.imshow(image)

<matplotlib.image.AxesImage at 0x7fb53b1abf10>
```



memuat gambar



output



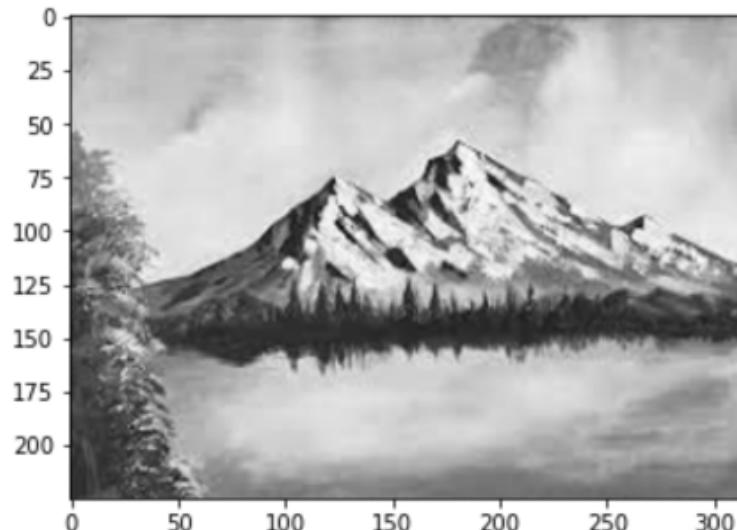
Hands-on : Segmentasi

```
# Making the gray scale of the image
```

```
gray = rgb2gray(image)  
plt.imshow(gray, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7fb518107190>
```

mengubah gambar
menjadi grayscale



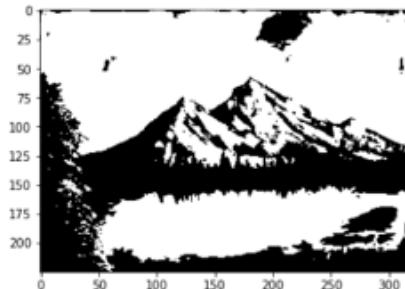
output



Hands-on : Segmentasi

```
# What if we use the mean of the pixels in the whole image as threshold and kinda use it for segmentation
arr=gray.flatten()
for i in range(len(arr)):
    if arr[i]>arr.mean():
        arr[i]=1
    else:
        arr[i]=0
gray_segmented=arr.reshape(gray.shape[0],gray.shape[1])

plt.imshow(gray_segmented,cmap='gray')
<matplotlib.image.AxesImage at 0x7fb53b1475e0>
```



Segmentasi obyek menjadi 2 bagian yang berbeda berdasarkan nilai threshold yang ditentukan



output





Hands-on : Segmentasi

```
# What if we tune the above function more??  
arr=gray.flatten()  
for i in range(len(arr)):  
    if arr[i]>=arr.mean():  
        arr[i]=4  
    elif arr[i]>=0.75:  
        arr[i]=3  
    elif arr[i]>0.5 :  
        arr[i]=2  
    elif arr[i]>0.25:  
        arr[i]=1  
    else:  
        arr[i]=0  
gray_segmented_2=arr.reshape(gray.shape[0],gray.shape[1])
```



Segmentasi obyek menjadi 5 bagian yang berbeda berdasarkan nilai threshold yang ditentukan

Hands-on : Segmentasi

```
# There are 5 segments in the below image :)
plt.figure(figsize=(18,8))
plt.imshow(gray_segmented_2,cmap='pink')
plt.axis("off")
plt.show()
```



Menampilkan gambar setelah melakukan segmentasi objek menjadi 5 bagian

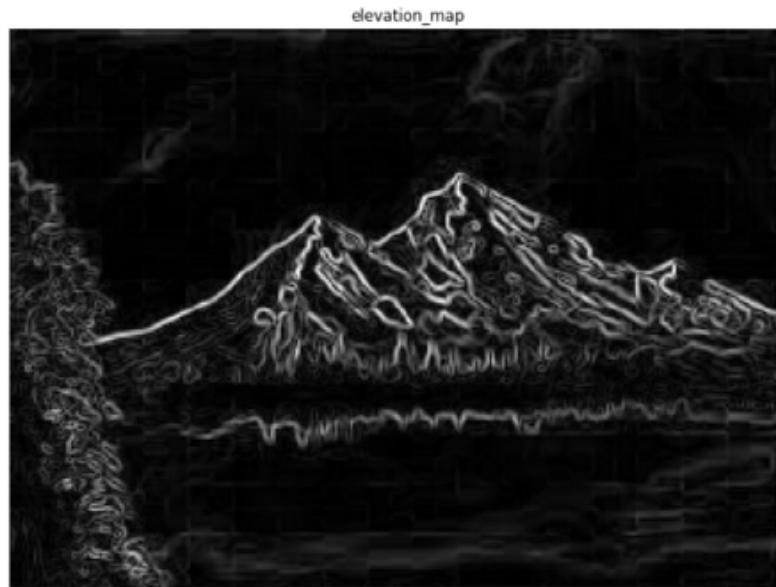
output



Hands-on : Segmentasi

```
imm=image[:, :, 0]
elevation_map = sobel(imm)

fig, ax = plt.subplots(figsize=(18,8))
ax.imshow(elevation_map, cmap='gray', interpolation='nearest')
ax.axis('off')
ax.set_title('elevation_map')
plt.show()
```



Menampilkan gambar tanpa mencoba melakukan interpolasi antar piksel

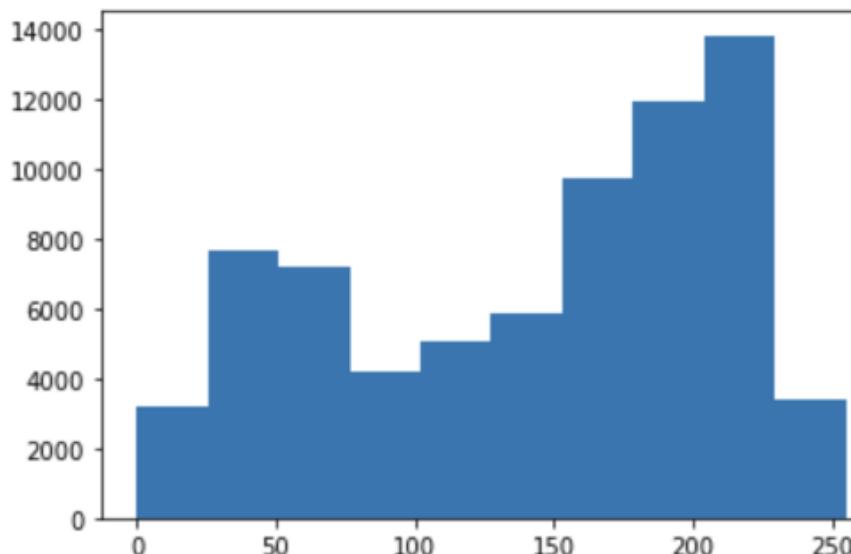
output

Hands-on : Segmentasi

```
plt.hist(imm.flatten())  
plt.show()
```



membuat histogram



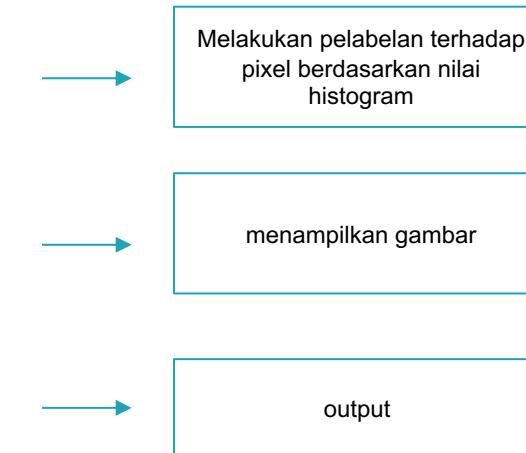
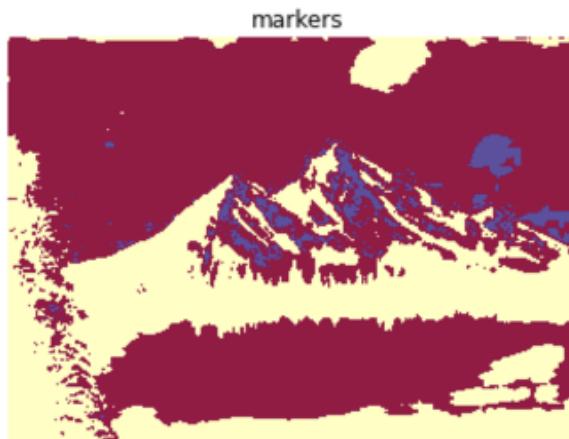
output

Hands-on : Segmentasi

```
markers = np.zeros_like(imm)
markers[imm < 117] = 1
markers[imm > 232] = 2

fig, ax = plt.subplots(figsize=(8,4))
ax.imshow(markers, cmap='Spectral', interpolation='nearest')
ax.axis('off')
ax.set_title('markers')

Text(0.5, 1.0, 'markers')
```



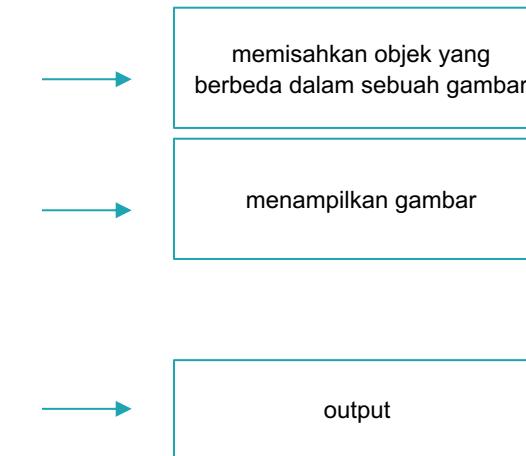
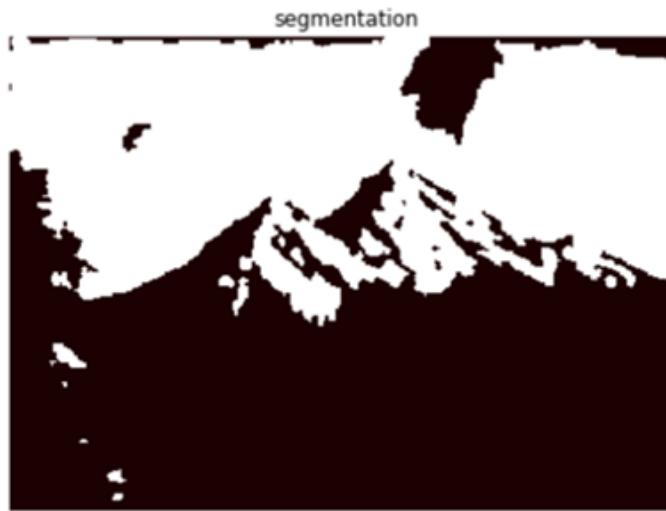


Hands-on : Segmentasi

```
segmentation = skimage.segmentation.watershed(elevation_map, markers)

fig, ax = plt.subplots(figsize=(10,5))
ax.imshow(segmentation, cmap='pink', interpolation='nearest')
ax.axis('off')
ax.set_title('segmentation')

Text(0.5, 1.0, 'segmentation')
```



Pelabelan data : Natural Language Processing (NLP)

Speaking to/with machines 'like with a friend'
Powered by machine learning and natural language processing



Pelabelan data : Natural Language Processing (NLP)

Pemrosesan bahasa alami (atau disingkat NLP) mengacu pada analisis bahasa manusia dan bentuknya selama interaksi baik dengan manusia lain maupun dengan mesin. Menjadi bagian dari linguistik komputasi awalnya, NLP telah berkembang lebih lanjut dengan bantuan Artificial Intelligence dan Deep Learning.

Berikut adalah beberapa pendekatan pelabelan data untuk pelabelan data NLP :

- Entity annotation and linking
 - Anotasi entitas mengacu pada anotasi entitas atau fitur tertentu dalam korpus data yang tidak berlabel.
 - Kata 'Entitas' dapat mengambil bentuk yang berbeda tergantung pada tugas yang dihadapi.
 - Untuk anotasi kata benda yang tepat, kami telah menamai anotasi entitas yang mengacu pada identifikasi dan penandaan nama dalam teks.
 - Untuk analisis frasa, kami mengacu pada proses sebagai penandaan frasa kunci di mana kata kunci atau frasa kunci dari teks dianotasi.
 - Untuk analisis dan anotasi elemen fungsional dari teks apapun seperti kata kerja, kata benda, preposisi, kami menggunakan penandaan Parts of Speech, disingkat sebagai penandaan POS.
 - Penandaan POS digunakan dalam penguraian, terjemahan mesin, dan pembuatan data linguistik.
 - Anotasi entitas diikuti dengan penautan entitas, di mana entitas beranotasi ditautkan ke repositori data di sekitarnya untuk menetapkan identitas unik ke masing-masing entitas ini. Hal ini sangat penting ketika teks berisi data yang dapat ambigu dan perlu disambiguasi.
 - Tautan entitas sering digunakan untuk anotasi semantik, di mana informasi semantik entitas ditambahkan sebagai anotasi.

Pelabelan data : Natural Language Processing (NLP)

- **Text Classification**
 - Mirip dengan klasifikasi gambar di mana kami menetapkan label ke data gambar, dalam klasifikasi teks, menetapkan satu atau beberapa label ke blok teks.
 - Sementara dalam anotasi dan penautan entitas, kita memisahkan entitas di dalam setiap baris teks, dalam klasifikasi teks, teks dianggap sebagai keseluruhan dan satu set tag ditetapkan ke dalamnya
 - Jenis klasifikasi teks meliputi klasifikasi berdasarkan sentimen (untuk analisis sentimen) dan klasifikasi berdasarkan topik yang ingin disampaikan teks (untuk kategorisasi topik).
- **Phonetic Annotation**
 - Anotasi fonetik mengacu pada pelabelan koma dan titik koma yang ada dalam teks dan sangat diperlukan dalam chatbot yang menghasilkan informasi tekstual berdasarkan input yang diberikan kepada mereka.
 - Koma dan berhenti di tempat yang tidak diinginkan dapat mengubah struktur kalimat, menambah pentingnya langkah ini.

Pelabelan Data : Best Practise

- Dengan pembelajaran yang diawasi menjadi bentuk pembelajaran mesin yang paling umum saat ini, pelabelan data ditemukan di hampir setiap tempat kerja yang membahas tentang AI.
- Berikut adalah beberapa praktik terbaik untuk pelabelan data untuk AI guna memastikan model Anda tidak rusak karena data yang buruk:

Proper dataset collection and cleaning

Data harus beragam tetapi spesifik untuk pernyataan masalah. Data yang beragam memungkinkan kami untuk menyimpulkan model ML dalam beberapa skenario dunia nyata sambil mempertahankan spesifisitas sehingga mengurangi kemungkinan kesalahan. Demikian pula, pemeriksaan bias yang tepat mencegah model dari overfitting ke skenario tertentu.

Proper Annotation Approach

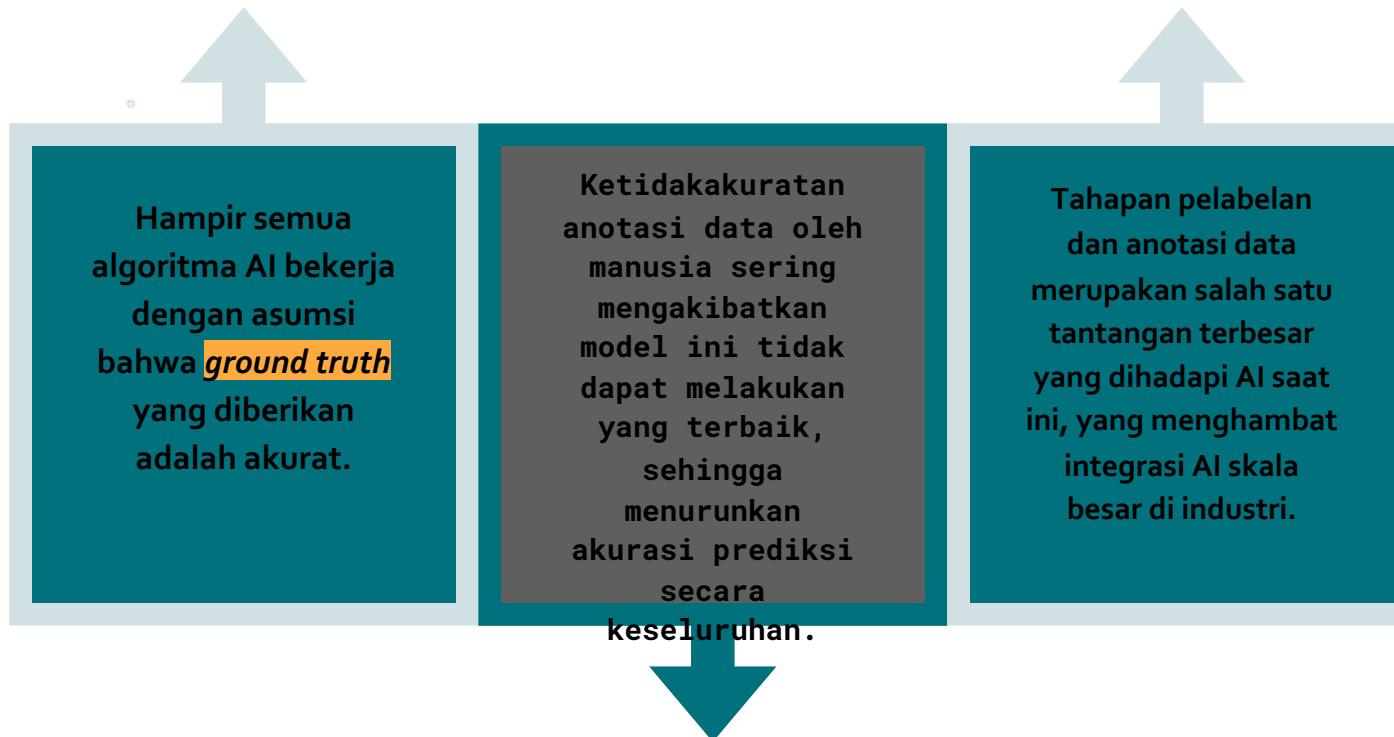
Data yang akan dianotasi harus diberi label melalui pelabelan internal, outsourcing, atau melalui cara crowdsourcing. Pilihan yang tepat dari pendekatan pelabelan data yang dilakukan membantu menjaga anggaran tetap terkendali tanpa mengurangi akurasi anotasi.

QA

Quality Assurance mencegah label palsu dan data yang tidak diberi label dengan benar diumpulkan ke algoritme ML. Anotasi yang tidak tepat dapat menjadi noise dan merusak model ML yang dapat dibangun.



Pelabelan Data : Best Practise



Dokumentasi Pelabelan Data



Kualitas dan Akurasi Data

- Akurasi dalam pelabelan data mengukur **seberapa dekat pelabelan dengan *ground truth***, atau seberapa baik fitur berlabel dalam data set konsisten dengan kondisi dunia nyata. Misal dalam *computer vision*, dalam meletakkan kotak pembatas di sekitar objek di satu jalanan) atau model pemrosesan bahasa alami (NLP) seperti mengklasifikasikan teks untuk sentimen sosial.
- **Kualitas** dalam pelabelan data adalah tentang **akurasi dataset secara keseluruhan**. Apakah pekerjaan semua pemberi label terlihat sama? Apakah pelabelan secara konsisten akurat di seluruh data set? Misalkan kita memiliki 29, 89, atau 999 pelabel data yang bekerja secara bersamaan.



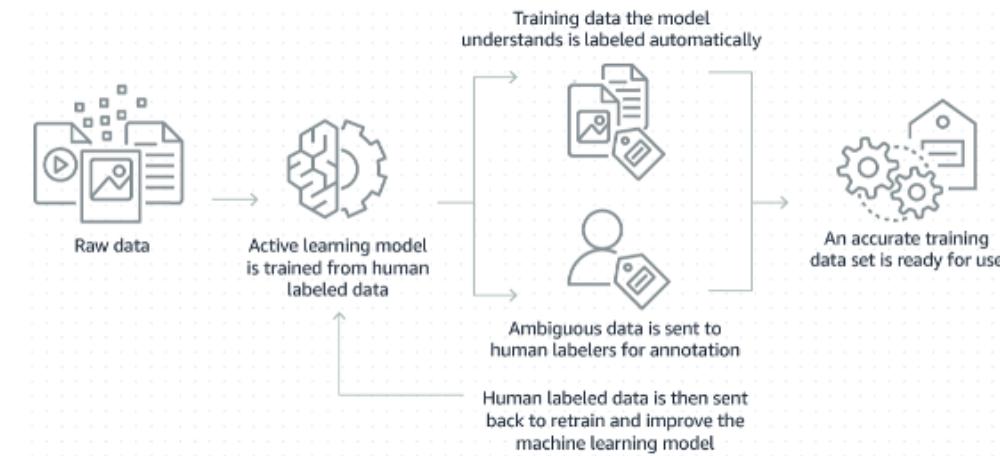
Menganalisis Akurasi Pelabelan Data

- Business Goals suatu AI yang berbeda memerlukan ukuran kualitas data yang berbeda.
- Keseimbangan dan variasi titik data di dalam dataset merupakan indikator seberapa baik algoritma dapat memprediksi suatu titik atau pola selanjutnya.
 - Misal tugas suatu AI adalah membedakan antara kendaraan yang bergerak dan tidak bergerak. Jika dataset memuat 90% gambar mobil bergerak tetapi hanya 10% yang diparkir, maka dapat dianggap tidak seimbang.
 - Untuk mengatasi masalah ini dapat digunakan teknik seperti oversampling, downsampling atau weight balancing.



Menganalisis Akurasi Pelabelan Data

- Kualitas data set untuk pelatihan model sering ditentukan oleh seberapa tepat label dan kategori ditempatkan pada setiap titik data.
- Namun, bukan hanya tentang keakuratan pelabelan data tetapi juga tentang seberapa konsisten keakuratannya.
 - Akurasi dan konsistensi data diukur selama proses penjaminan mutu, langkah-langkah terpisah yang dapat dilakukan secara manual atau otomatis.
 - Pendekatan yang berbeda dapat digabungkan untuk *cross check* dan memastikan kesempurnaan data set.



Apa yang mempengaruhi kualitas data dalam pelabelan?

- *Knowledge and context*
 - Pengetahuan dasar satu domain dan pemahaman kontekstual sangat penting seperti pemberi label untuk membuat set data terstruktur berkualitas tinggi.
- *Agility*
 - Pelabelan data berkembang saat dilakukan pengujian dan validasi model, sehingga harus disiapkan data set baru dan memperkaya data set yang ada untuk meningkatkan hasil algoritma Machine Learning.
- *Relationship*
 - Kita memerlukan pemberi label data yang dapat merespons dengan cepat dan mengikuti alur kerja tim, berdasarkan apa yang telah dipelajari dalam fase pengujian dan validasi model.
- *Communication*
 - Pendekatan umpan balik (feedback) adalah cara terbaik untuk membangun komunikasi dan kolaborasi yang andal antara tim dan pemberi label data.



Metode QA untuk Mengukur Kualitas Data

- **Consensus Algorithm**
 - Merupakan proses untuk mencapai reliabilitas data melalui kesepakatan pada satu titik data di antara beberapa individu pemberi label data atau suatu organisasi.
 - Konsensus dapat dilakukan dengan menetapkan sejumlah *reviewer* per titik data (umumnya untuk *data open source*) atau sepenuhnya otomatis.
- **Benchmarking and Gold Standard**
 - Benchmarking adalah pendekatan yang lebih kompleks dan andal untuk QA, karena menggunakan standar tertentu.
 - Menggunakan otomatisasi, pemberi label mendapatkan benchmark secara acak untuk memastikan bahwa label dan anotasi mematuhi referensi yang telah ditentukan
 - Ahli diperlukan hanya untuk membuat referensi dan meninjau kualitas secara keseluruhan dan potensi penyimpangan.



Metode QA untuk Mengukur Kualitas Data

- **Sample review**
 - Pilih sampel acak dari hasil pelabelan yang telah diselesaikan.
 - Pekerja yang lebih berpengalaman, seperti pemimpin tim atau manajer proyek, dapat meninjau sampel untuk mengukur akurasinya.
- **Cronbach's Alpha Test**
 - Digunakan sebagai ukuran korelasi rata-rata atau konsistensi item dalam dataset, yang tergantung pada karakteristik penelitian (misal homogenitas).
 - Dapat membantu dengan cepat melihat keandalan label secara keseluruhan.

Cronbach's Alpha Test

- Dikenal sebagai ukuran konsistensi internal yang digunakan dalam konteks instrumen pengukuran multi-item dan memiliki aplikasi yang luas.
- Cronbach's Alpha digunakan untuk mengestimasi item data dalam dataset termasuk label.

$$\alpha = \frac{k}{k-1} \left(1 - \frac{\sum s_i^2}{s_x^2} \right)$$

- Dimana α adalah koefisien reliabilitas, k adalah jumlah item set, s_i^2 adalah nilai *variance* setiap item i dimana $i = 1, 2, \dots, k$, and s_x^2 adalah $\sum s_i^2$ *variance* dari semua item. Semakin tinggi nilai koefisien α , maka setiap item memiliki nilai *covariance* dan dapat dihitung (memiliki kesamaan konsep).
- Kategori reliabilitas tinggi dengan nilai $\alpha > 0.05$.

Cronbach's Alpha Test

- Besarnya koefisien reliabilitas berhubungan langsung dengan skor standar deviasi yang diperoleh dari sampel data apa pun karena koefisien reliabilitas adalah koefisien korelasi.

| Series Number | N of Respondents | N of Items | Range | Variance | Standard Deviation | α | Mean |
|---------------|------------------|------------|--------|----------|--------------------|----------|-------|
| 1 | 200 | 25 | 27.00 | 27.05 | 5.20 | 0.10 | 69.54 |
| 2 | 200 | 25 | 30.00 | 24.36 | 4.93 | 0.01 | 74.29 |
| 3 | 200 | 25 | 35.00 | 42.18 | 6.49 | 0.47 | 80.13 |
| 4 | 200 | 25 | 40.00 | 61.79 | 7.86 | 0.67 | 81.10 |
| 5 | 200 | 25 | 45.00 | 65.50 | 8.09 | 0.67 | 80.42 |
| 6 | 200 | 25 | 51.00 | 68.48 | 8.27 | 0.68 | 79.83 |
| 7 | 200 | 25 | 60.00 | 79.55 | 8.91 | 0.74 | 80.83 |
| 8 | 200 | 25 | 72.00 | 76.62 | 8.75 | 0.72 | 80.93 |
| 9 | 200 | 25 | 89.00 | 103.97 | 10.19 | 0.80 | 80.57 |
| 10 | 200 | 25 | 100.00 | 108.52 | 10.41 | 0.81 | 80.63 |

- Dapat dilihat bahwa semakin tinggi nilai Range dan Variance membuat nilai reliabilitas juga naik.

Keamanan Pelabelan Data

- What are the security risks of outsourcing data labeling?
 - Mengakses data dari jaringan yang tidak aman atau menggunakan perangkat tanpa perlindungan malware
 - Mengunduh atau simpan sebagian data (mis., screen capture, flash drive)
 - Memberi label data saat berada di tempat umum
 - Tidak memiliki pelatihan, konteks, atau akuntabilitas terkait dengan aturan keamanan untuk pekerjaan labeling
 - Bekerja di lingkungan fisik atau digital yang tidak disertifikasi untuk mematuhi peraturan data (mis., HIPAA, SOC 2).
- Tiga area yang perlu menjadi perhatian untuk menjaga keamanan dokumen
 - **Orang dan Tenaga Kerja:** Ini dapat mencakup pemeriksaan latar belakang untuk pekerja dan mungkin mengharuskan pemberi label untuk menandatangani perjanjian kerahasiaan (NDA) atau dokumen serupa yang menguraikan persyaratan keamanan data.
 - **Teknologi dan Jaringan:** Pekerja mungkin diminta untuk menyerahkan perangkat yang mereka bawa ke tempat kerja, seperti ponsel atau tablet.
 - **Fasilitas dan Ruang Kerja:** Pekerja dapat duduk di tempat yang menghalangi orang lain untuk melihat pekerjaan mereka.



Summary

- Transformasi Data adalah bagian dari Data Preparation
- Membutuhkan pengetahuan dasar dan detail serta waktu yang mayoritas untuk menjamin data yang akan dianalisis sebersih mungkin
- Transformasi data dapat menggunakan beberapa teknik rekayasa fitur (feature engineering)
- Normalisasi, Standardisasi adalah bagian proses atau tahapan yang diperlukan untuk mentransformasi data
- Selain data terstruktur, transformasi data juga krusial dilakukan untuk data yang semi terstruktur dan tidak terstruktur (unstructured) seperti teks, image, audio dan video
- Dokumentasi juga dilakukan untuk proses transformasi data, seleksi fitur maupun pelabelan data
- Pelabelan bergantung pada pernyataan masalah, kerangka waktu proyek, dan jumlah orang yang terkait dengan pekerjaan

Referensi

- https://www.ucl.ac.uk/population-health-sciences/sites/population-health-sciences/files/quartagno_1.pdf
- https://rianneschouten.github.io/missing_data_science/assets/blogpost/blogpost.html
- <https://towardsdatascience.com/tf-term-frequency-idf-inverse-document-frequency-from-scratch-in-python-6c2b61b78558>
- <https://dataaspirant.com/nlp-text-preprocessing-techniques-implementation-python>
- https://www.oreilly.com/library/view/blueprints-for-text/9781492074076/assets/btap_0401.png
- <https://monkeylearn.com/unstructured-data>
- <https://medium.com/machine-learning-id/melakukan-feature-scaling-pada-dataset-229531bb08de>
- <https://protnobi.com/post/extreme-values-winsorize-trim-or-retain>
- <https://heartbeat.fritz.ai/hands-on-with-feature-engineering-techniques-dealing-with-outliers-fcc9f57cb63b>
- <https://www.analyticsvidhya.com/blog/2021/05/detecting-and-treating-outliers-treating-the-odd-one-out/>



Tools / Lab Online

- Jupyter Notebook
- Google Collabs



Quiz / Tugas

Quiz dapat diakses melalui <https://spadadikti.id/>

Terima kasih