
INSTITUT TEKNOLOGI BANDUNG
1920

Exploratory Data Analytics & Data Preparation

Informatics Research Group

School of Electrical Engineering and Informatics

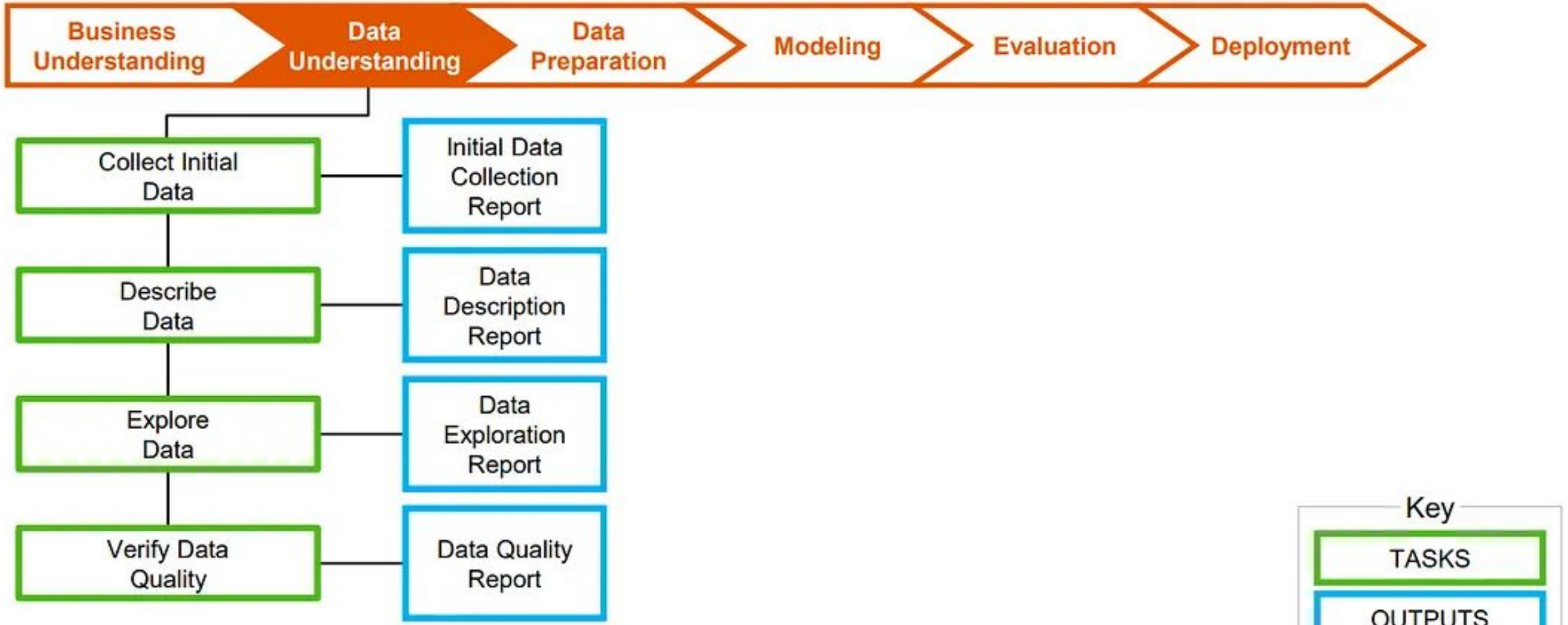
Institut Teknologi Bandung

Sources:

Materi Pelatihan Associate Data Scientist – Pusat Artificial Intelligence ITB

Data Understanding Phase – Overview

CRISP-DM – Phase 2: Data Understanding



Sumber: https://open.sap.com/courses/ds3

Collect Initial Data

- **Collect Initial Data** or acquire the data and its access to the data listed in the projects resources. Collecting initial data also means you need to have a checklist of the dataset you have acquired, the dataset location, the methods to acquire the datasets, and record any problems encountered and any solutions to the problems for the other users or project members to be aware of.

Describe Data

- **Describe Data** by examining the properties of the data acquired, provide a description report regarding the format of the data, quantity of data and even the records and fields in each table or datasets.

Explore Data

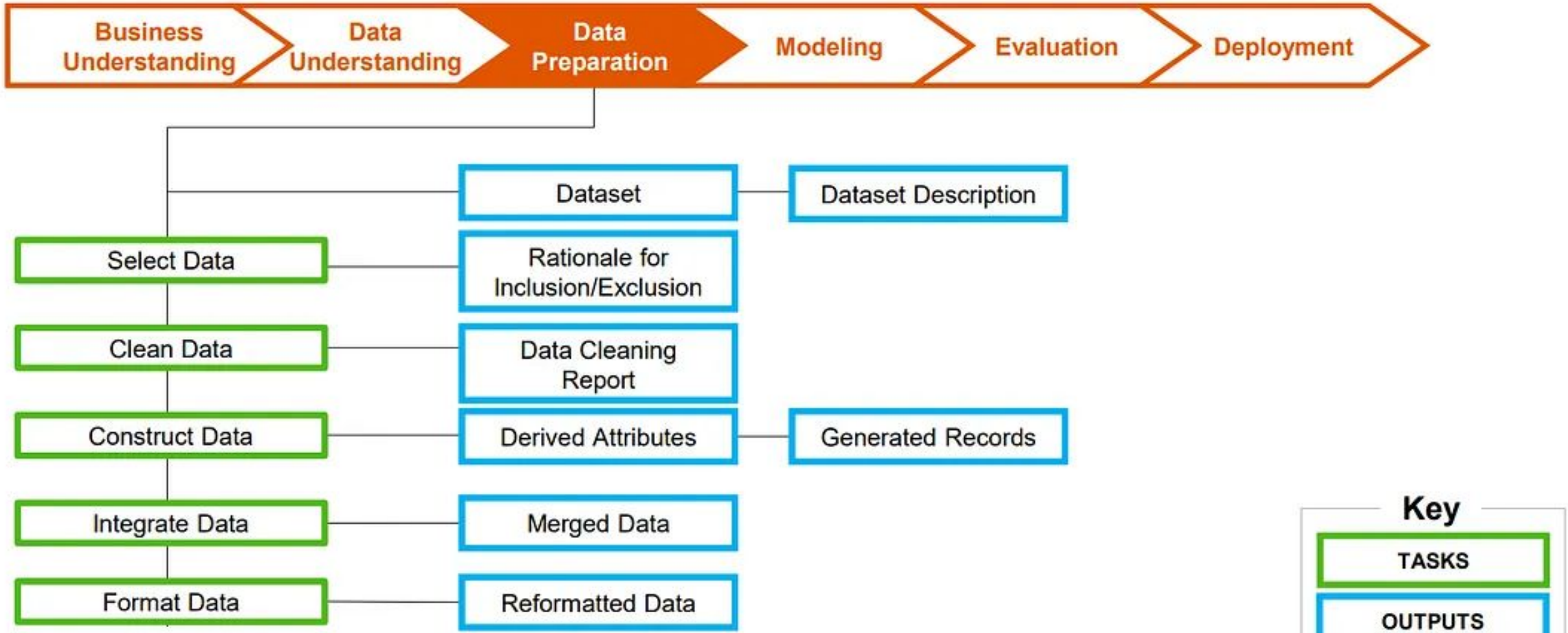
- **Explore Data** by using data science questions that can be quickly answered through querying, visualization, and reporting or summary report. In this stage, you will be able to find your first or initial hypothesis and their impact on the project.

Verify Data Quality

- **Verify Data Quality** by examining if the data is complete. If the data has errors or are there missing values and if there is, what is the percentage of the missing values versus the overall data obtained.

Data Preparation Phase – Overview

CRISP-DM – Phase 3: Data Preparation



5

Persiapan Data

- Proses dimana data yang sesuai dikumpulkan, dipilih, dibersihkan, dan diorganisir sesuai dengan kebutuhan bisnis untuk digunakan pada tahap pemodelan.

- Dilakukan setelah tahap pemahaman data.
  - Laporan hasil pemahaman data digunakan sebagai dasar untuk menentukan aksi apa yang harus dilakukan pada tahap ini.

Select Data	Select data or decide on the data to be used for analysis. One of the criteria in selecting the data is that it should be relevant to the data science goal that was identified in the business understanding phase. In selecting data, you also need to list the data to be excluded and included and the reasons for these decisions
Clean data	Clean data by raising the data quality to the level required by the selected analysis techniques. Here, you also need to describe what decisions and actions were taken to address the data quality problems reported during the Verify Data Quality task of the Data Understanding Phase
Construct data	Construct data by including derived attributes, entire new records, or transformed values for existing attributes. This may be conducting encoding methods especially for categorical variables or feature engineering
Integrate data	Integrate data by combining from multiple tables or records to create records or values. SQL knowledge and skill is very important and would come in handy in this part
Format data	Format data by transforming the data but not necessarily change its meaning but might be required by the modeling tool. An example would be transforming your data either by standardization or normalization

Tujuan Persiapan Data

- Meningkatkan kualitas data
- Memudahkan pemodelan

Berdasar Pengalaman dalam Persiapan Data

Data preparation is more than half of every data mining process

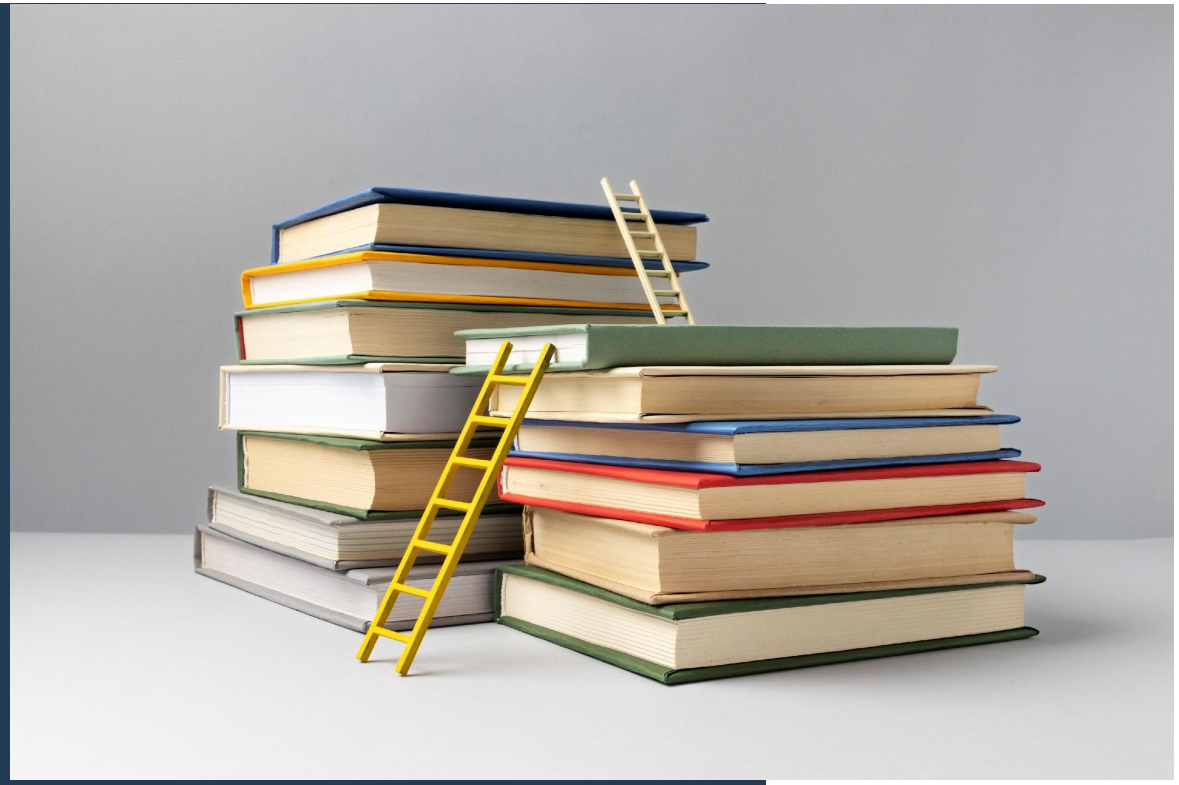
- Maxim of data mining: most of the effort in a data mining project is spent in data acquisition and preparation, and informal estimates vary from 50 to 80 percent

http://www.khabaza.com/9laws

Khabaza, T. (2014). From cognitive science to data mining: the first intelligence amplifier. In *From Animals to Robots and Back: Reflections on Hard Problems in the Study of Cognition: A Collection in Honour of Aaron Sloman* (pp. 191-204). Cham: Springer International Publishing.

Proses dalam Persiapan Data

1. **Pemilihan Data**
   a. *Record selection*
   b. *Feature selection*
2. **Perbaikan Data**
   a. Mengisi *missing values*
   b. Perbaikan error
   c. Penanganan *outlier*
   d. Penghapusan duplikasi
3. **Konstruksi Data**
   a. *Dimensionality reduction*
   b. Transformasi data
   c. *Encoding*
4. **Integrasi Data**
   a. *Data Join*
   b. *Append*

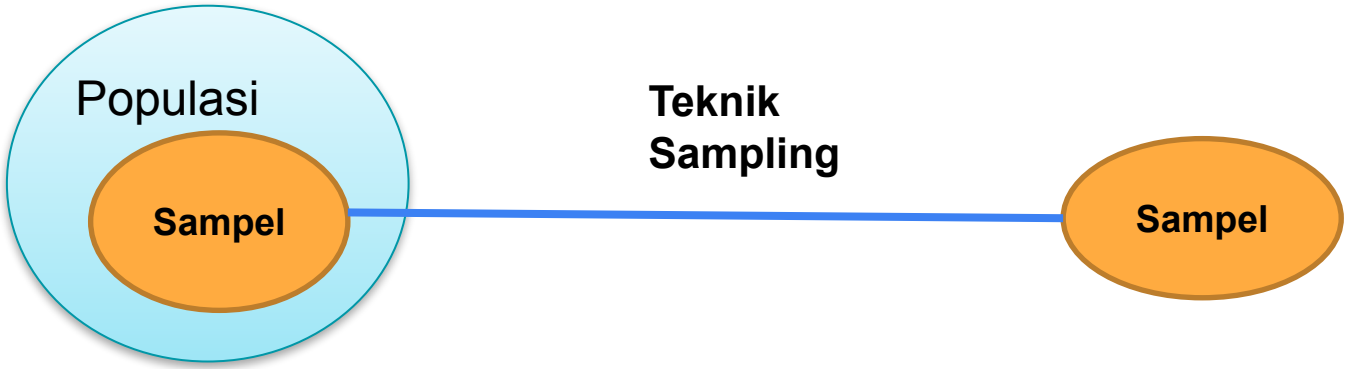


Pemilihan Data

a. Record selection
b. Feature selection

Record Selection (Sampling)

- **Sampling** adalah proses dalam analisis statistik dimana peneliti mengambil sejumlah **pengamatan** yang telah ditentukan sebelumnya dari **populasi** yang lebih **besar**.



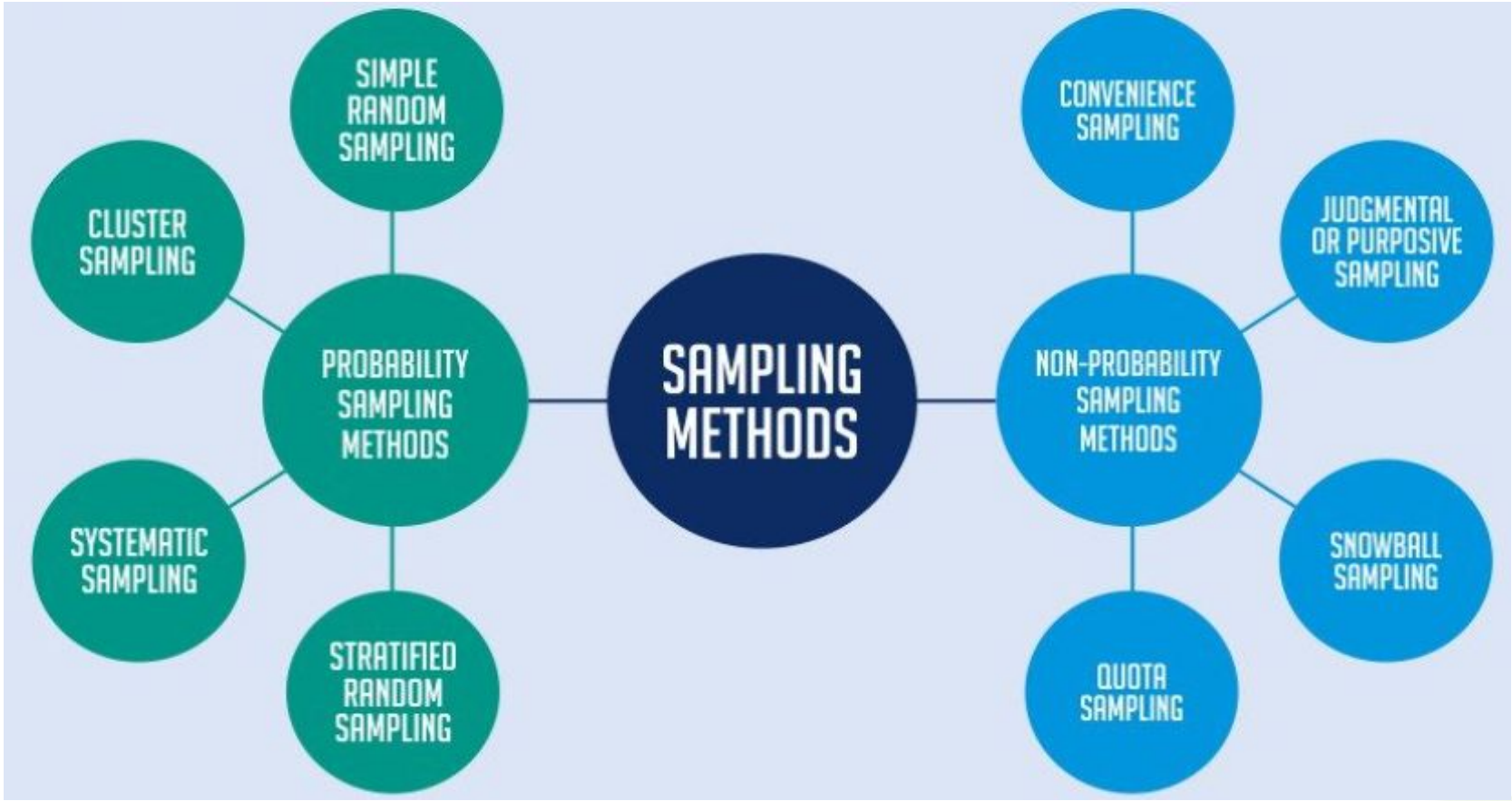
- Pengambilan sampel memungkinkan peneliti untuk melakukan studi tentang kelompok besar dengan menggunakan sebagian **kecil** dari populasi.

Kategori Metode Sampling

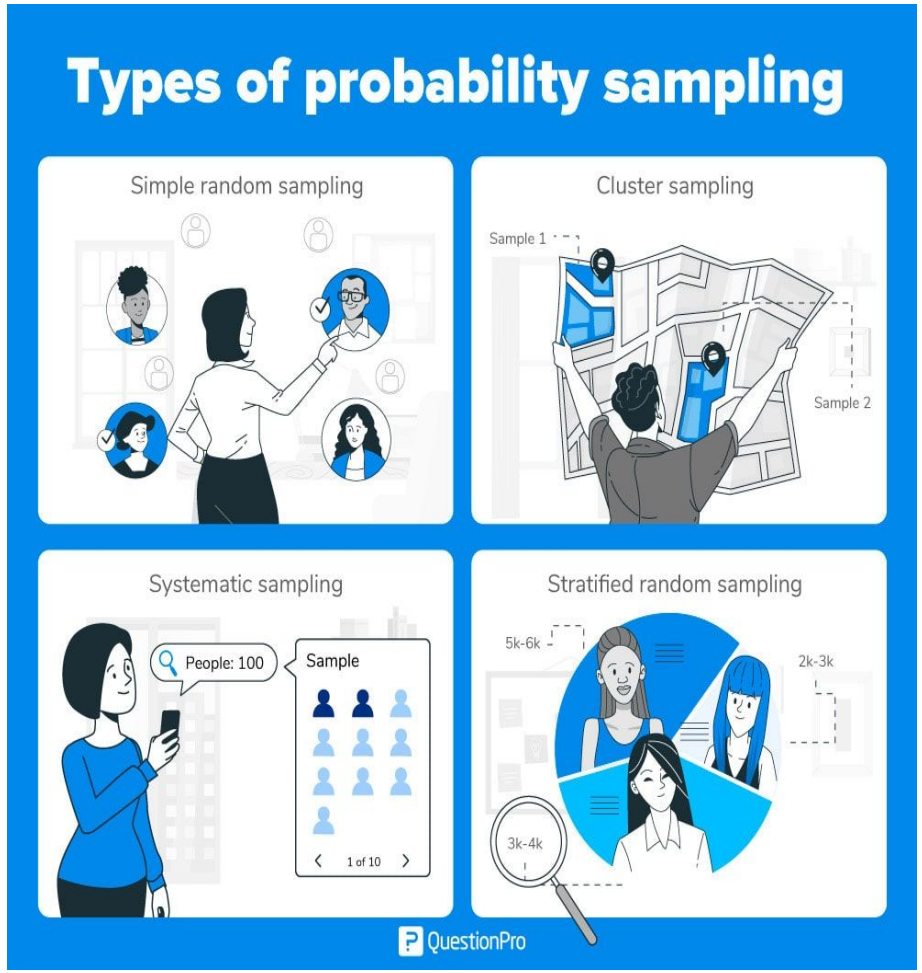
- **Probability Sampling:**
  Teknik pengambilan sampel dimana sampel dari populasi yang lebih besar dipilih dengan menggunakan metode berdasarkan teori probabilitas.

- **Non-Probability Sampling**
  Teknik pengambilan sampel dimana peneliti memilih sampel berdasarkan penilaian subjektif, bukan pemilihan acak.

Kategori Metode Sampling



Tipe Probability Sampling



1. **Simple Random Sampling**
   Pengambilan sampel dari anggota populasi dengan menggunakan acak tanpa memperhatikan apapun.
2. **Systematic Sampling**
   Pengambilan sampel secara sistematis dengan interval (jarak) tertentu antar sampel yang terpilih.
3. **Stratified Random Sampling**
   Pengambilan sampel dengan cara membagi populasi ke dalam kelompok-kelompok yang homogen (disebut strata), dan dari tiap stratum tersebut diambil sampel secara acak.
4. **Cluster Sampling**
   Pengambilan sampel dilakukan terhadap sampling unit, dimana sampling unitnya terdiri dari satu kelompok (cluster). Tiap item (individu) di dalam kelompok yang terpilih akan diambil sebagai sampel.

When to use probability sampling?

1

When you want to reduce the sampling bias
Probability sampling leads to higher quality findings because it provides an unbiased representation of the population.

2

When the population is usually diverse
This sampling method will help pick samples from various socio-economic strata, background, etc. to represent the broader population.

3

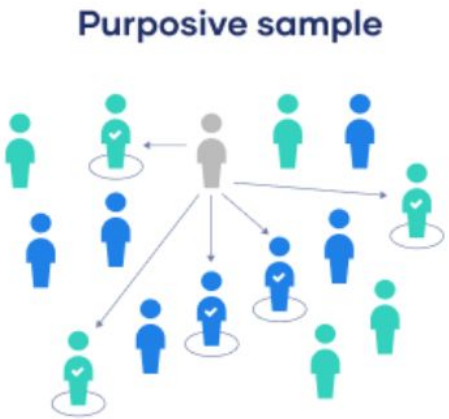
To create an accurate sample
Researchers use proven statistical methods to draw a precise sample size to obtain well-defined data.

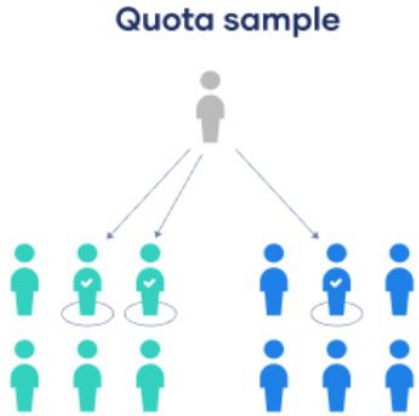
Learn more:
www.questionpro.com/blog/probability-sampling/

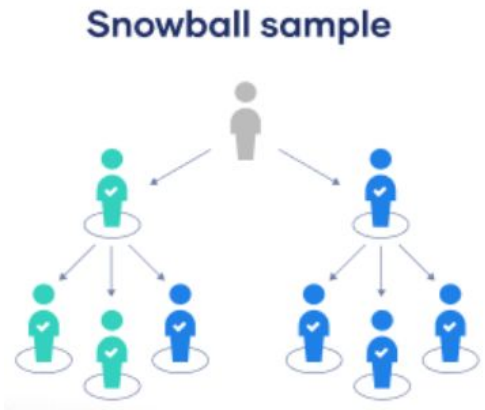
QuestionPro

Tipe Non-Probability Sampling

- **Purposive Sampling**
  Sample dipilih berdasarkan pertimbangan tertentu untuk memperoleh sampel dengan karakteristik yang dikehendaki.
- **Quota Sampling**
  Pengambilan sampel hanya berdasarkan pertimbangan peneliti dengan besar dan kriteria sampel ditentukan lebih dahulu.
- **Saturation Sampling**
  Semua anggota populasi digunakan sebagai sampel karena jumlah populasi tidak banyak, atau ingin membuat generalisasi dengan kesalahan sangat kecil.
- **Snowball Sampling**
  Sampel diambil secara berantai, mulai dari ukuran sampel yang kecil semakin menjadi besar.

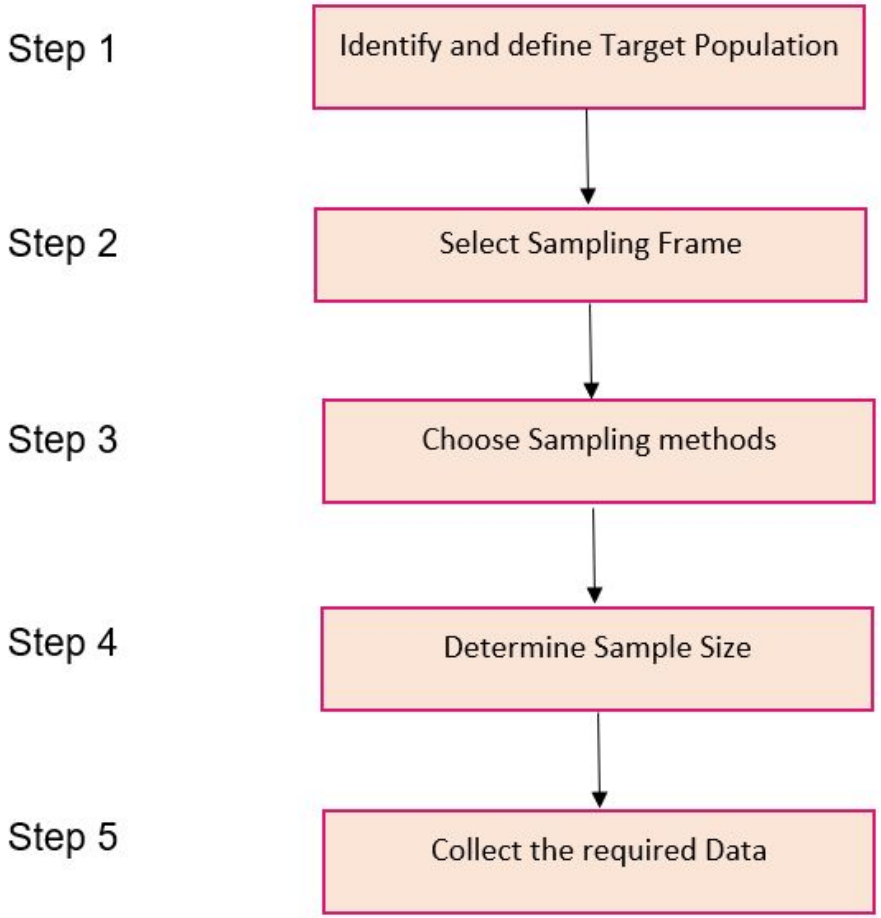






https://www.scribbr.com/methodology/sampling-methods/

Tahapan Melakukan Sampling

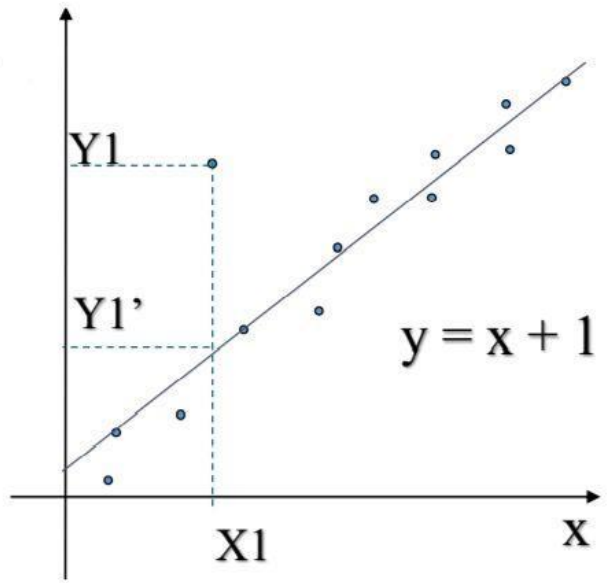


Teknik Pengurangan Data (Numerosity Reduction)

1. Parametrik:
   a. Regresi
   b. log-linear model, dll
2. Non-Parametrik
   a. Histogram
   b. Clustering
   c. Sampling

Numerosity Reduction dengan Pendekatan Parametrik

Regresi:



Regresi linier memodelkan hubungan antara dua atribut dengan memodelkan persamaan linier ke kumpulan data.

Misalkan kita perlu memodelkan fungsi linier antara dua atribut.

y = wx +b

y adalah atribut respons

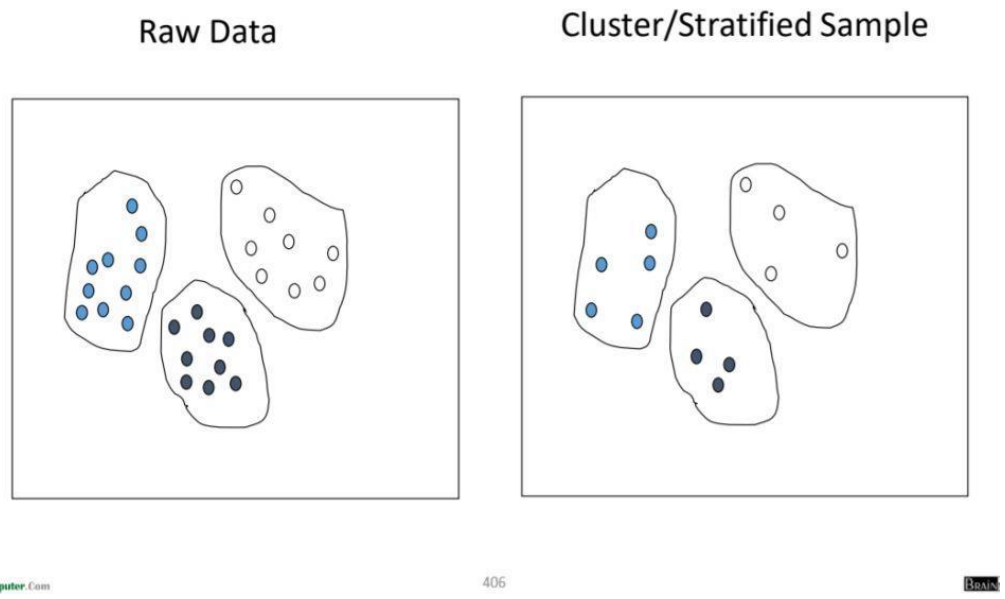
x adalah atribut prediktor.

Jika kita membahas dari segi data mining, atribut x dan atribut y adalah atribut numerik database, sedangkan w dan b adalah koefisien regresi.

Nilai w dan b dijadikan data baru untuk menggantikan n data pada kelompok data yang bisa diwakili oleh model regresi

Numerosity Reduction dengan Pendekatan Non-Parametrik

Clustering:



Teknik clustering mengelompokkan objek-objek yang mirip, sehingga objek-objek dalam satu cluster akan mirip satu sama lain, tetapi berbeda dengan objek-objek di cluster lain.
Seberapa mirip objek di dalam cluster dapat dihitung menggunakan fungsi jarak.

Centroid dari cluster digunakan sebagai data baru untuk mewakili data-data lain pada cluster yang sama

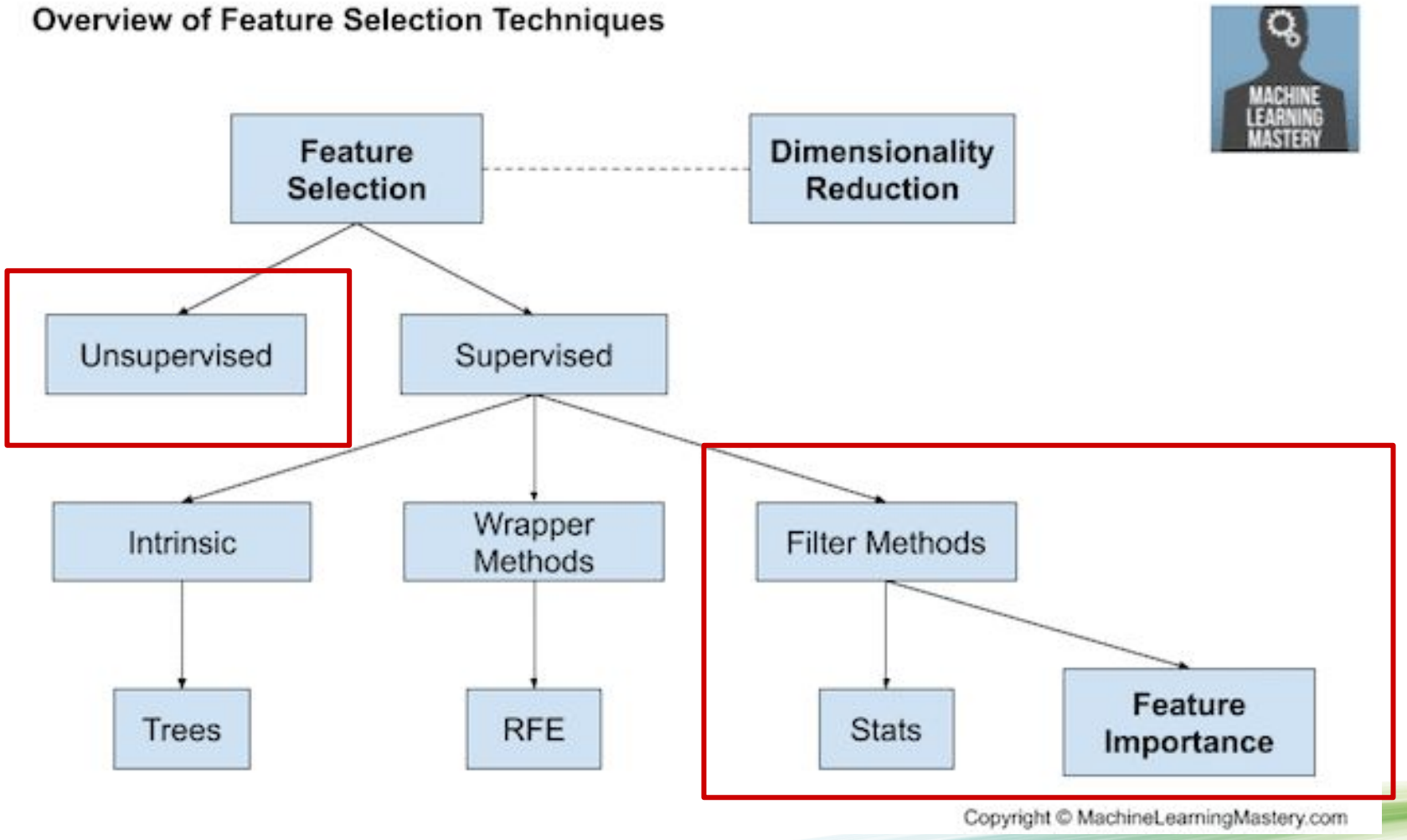
Feature Selection

- Pemilihan fitur adalah proses pengurangan jumlah variabel masukan saat mengembangkan model machine learning.

Tujuan Feature Selection

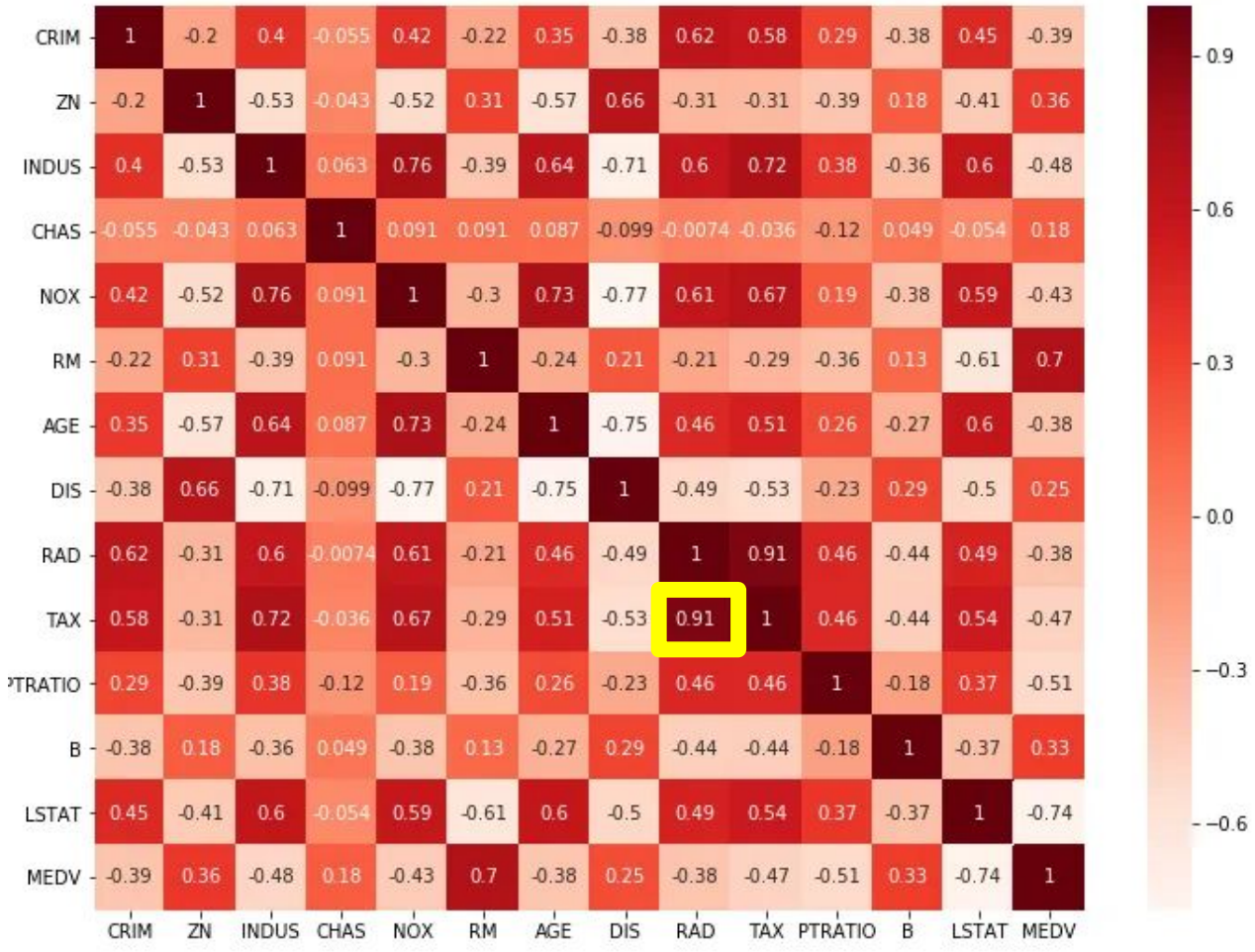
- Mengurangi jumlah variabel masukan untuk mengurangi biaya komputasi pemodelan
- Dalam beberapa kasus, ditujukan untuk meningkatkan kinerja model.

Teknik Feature Selection



Teknik Feature Selection berbasis Unsupervised

- Tidak melibatkan variable target
- Jika ada dua fitur yang memiliki korelasi yang kuat, hilangkan salah satu fitur



https://towardsdatascience.com/feature-selection-with-pandas-e3690ad8504b

Teknik Feature Selection berbasis Filter

- Mengevaluasi hubungan antara setiap variabel masukan dan variabel target (kelas) menggunakan:
  - statistik
  - memilih fitur yang penting (*feature importance*)

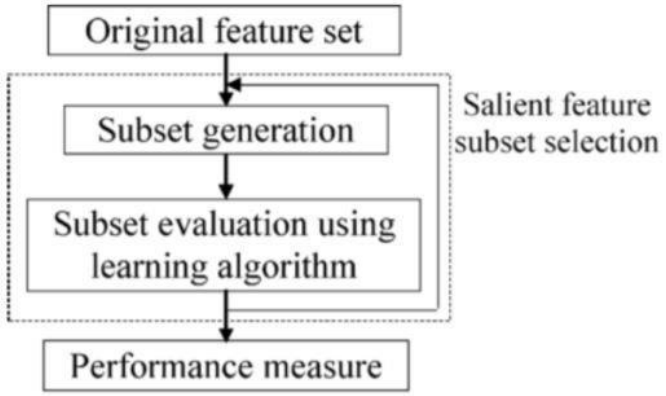
- Memilih variabel masukan yang memiliki hubungan paling kuat dengan variabel target.

Teknik Statistik untuk Feature Selection berbasis Filter

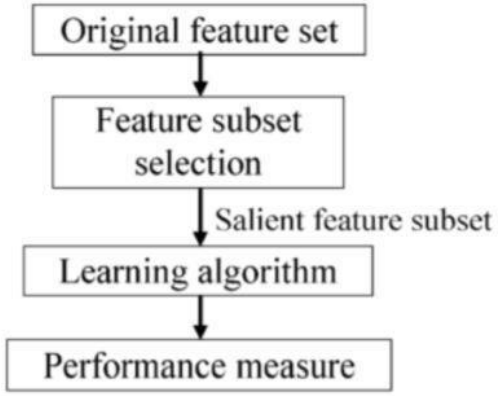
- Memilih variabel masukan yang memiliki hubungan yang paling kuat dengan variabel target secara statistik.
- Pilihan ukuran statistik bergantung pada tipe data variabel masukan dan keluaran.

Tipe Variable Masukan	Tipe Variable Target	Teknik Statistik
Numerik	Numerik	• Pearson's correlation coefficient (linear) • Spearman's rank coefficient (nonlinear)
Numerik	Categorical	• ANOVA correlation coefficient (linear) • Kendall's rank coefficient (nonlinear)
Categorical	Numerik	• ANOVA correlation coefficient (nonlinear) • Kendall's rank coefficient (linear)
Categorical	Categorical	• Chi-Squared test (contingency tables) • Mutual Information

Seleksi Fitur

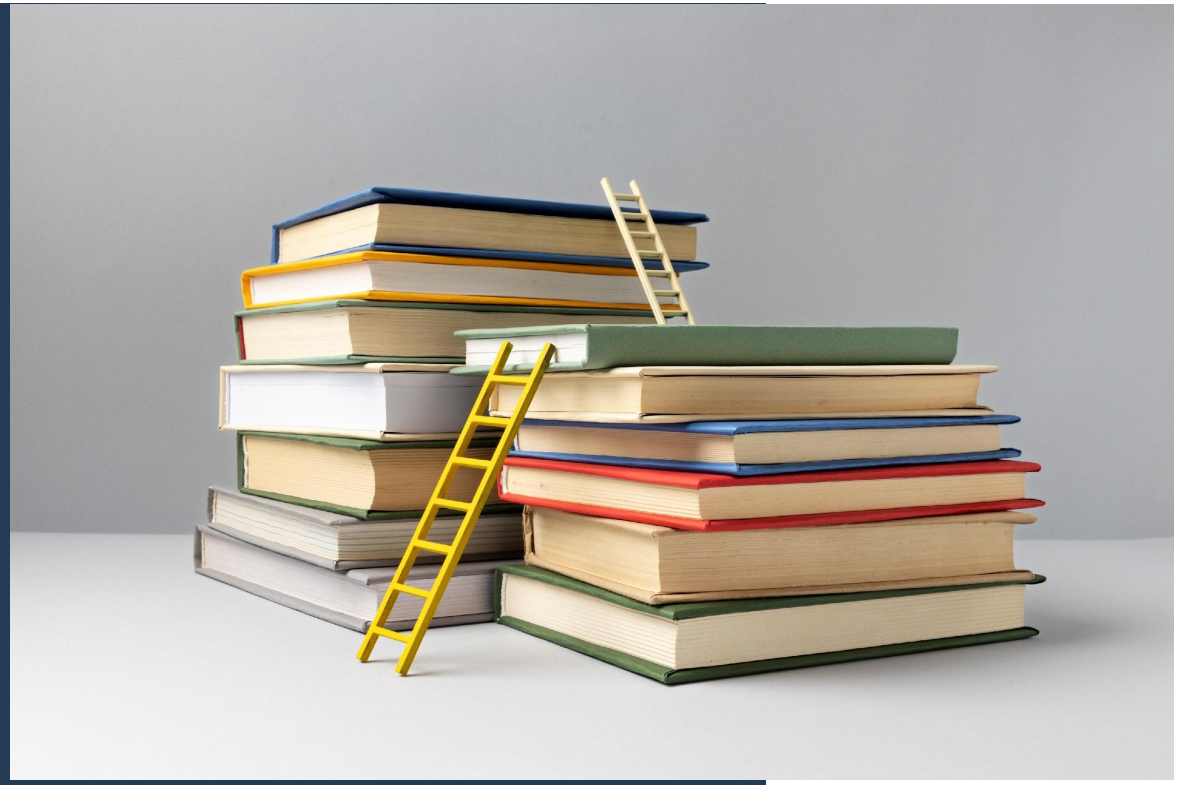


Wrapper Approach



Filter Approach

1. Dalam pendekatan wrapper, fitur-fitur digunakan untuk melatih model pembelajaran yang telah ditentukan. Fitur dikurangi secara bertahap dengan melihat kinerja model menaik atau menurun ketika fitur tersebut dihilangkan. Dapat menggunakan forward selection, backward elimination, randomized hill climbing, dll.
2. Dalam pendekatan filter, analisis statistik dari set fitur diperlukan, tanpa menggunakan model pembelajaran apapun. Dapat menggunakan information gain, chi square, log likelihood ratio, dll. (sudah dibahas sebelumnya)
3. Pendekatan yang embedded memanfaatkan kekuatan pelengkap pendekatan wrapper dan filter. Dapat menggunakan decision tree, weighted naïve bayes, dll.



Perbaikan Data

Proses dalam Persiapan Data (review)

1. **Pemilihan Data**
   a. *Record selection*
   b. *Feature selection*
2. **Perbaikan Data**
   a. Mengisi *missing values*
   b. Penghapusan duplikasi
   c. Penanganan *outlier*
   d. Perbaikan error
3. **Konstruksi Data**
   a. Dimensionality reduction
   b. Transformasi data
   c. Encoding
4. **Integrasi Data**
   a. *Data Join*
   b. *Append*

Contoh Data Kotor: Data Mengandung Missing Values

Row No.	age	education	balance	duration	campaign	y
1	58	tertiary	2143	261	1	no
2	44	secondary	29	151	1	no
3	33	secondary	?	76	1	no
4	47	unknown	1506	92	1	no
5	33	unknown	1	198	1	no
6	35	tertiary	231	139	1	no
7	28	tertiary	?	217	1	no
8	42	tertiary	2	380	1	no
9	58	primary	121	50	1	no
10	43	secondary	593	55	1	no
11	41	secondary	270	222	1	no
12	29	secondary	?	137	1	no
13	53	secondary	?	517	1	no
14	58	unknown	71	71	1	no
15	57	secondary	162	174	1	no
16	51	primary	229	353	1	no
17	45	unknown	13	98	1	no
18	57	primary	52	38	1	no

ExampleSet (45,211 examples, 0 special attributes, 6 regular attributes)

Contoh Data Kotor:
Data Tidak Konsisten

Row No.	nama_nasa...	jenis_kelamin	umur	jml_pinjaman	jkw
1	x1	P	40	345000	1
2	x2	L	31	350000	7
3	x3	L	29	649926	6
4	x4	P	2	459168	19
5	x5	WANITA	34	3055499	8
6	x6	L	49	2000000	19
7	x7	L	29	8333334	10
8	x8	L	27	4435001	8
9	x9	L	29	560000	19
10	x10	LAKI-LAKI	49	1443750	15
11	x11	LAKI-LAKI	42	3066000	10
12	x12	PRIA	26	4071669	20
13	x13	L	29	228655000	19
14	x14	L	55	840000	4
15	x15	L	38	3000000	24
16	x16	WANITA	29	1640000	19
17	x17	L	41	930000.010	4

A. Mengatasi Missing Values

1. Menghapus baris atau kolom dengan missing values
2. Mengisi dengan mean, median, atau modus
3. Mengisi dengan nilai tetap yang ditentukan sendiri
4. Mengisi dengan interpolasi yang memperkirakan nilai yang hilang berdasarkan tren data di sekitar
5. Menggunakan forward fill atau backward fill yaitu mengisi missing values dengan nilai sebelumnya atau setelahnya dalam kolom yang sama
6. Mengisi missing values dengan prediksi (imputasi lanjutan), yaitu menggunakan model untuk Prediksi nilai yang hilang berdasarkan nilai lain dalam dataset.

Implementasi Python:
Menghapus baris atau kolom dengan missing values

```
# dictionary of lists
dict = {'First':[100, 90, np.nan, 95],
        'Second': [30, 45, 56, np.nan],
        'Third':[np.nan, 40, 80, 98],
        'Forth':[50, 60, 70, 80]}

# creating a dataframe from list
df = pd.DataFrame(dict)
print(df)

# Menghapus baris yang memiliki missing values (axis=0)
df_dropped_rows = df.dropna()
print(df_dropped_rows)

# Menghapus kolom yang memiliki missing values (axis=1)
df_dropped_columns = df.dropna(axis= 1)
print(df_dropped_columns)
```

```
   First  Second  Third  Forth
0  100.0    30.0    NaN     50
1   90.0    45.0   40.0     60
2    NaN    56.0   80.0     70
3   95.0     NaN   98.0     80
   First  Second  Third  Forth
1   90.0    45.0   40.0     60
   Forth
0     50
1     60
2     70
3     80
```

Implementasi Python:
Mengisi dengan Mean, Median, Modus

```
# Mengisi missing values dengan mean
df_filled_mean = df.fillna(df.mean())
print(df_filled_mean)

# Mengisi missing values dengan median
df_filled_median = df.fillna(df.median())
print(df_filled_median)

# Mengisi missing values dengan modus
df_filled_mode = df.fillna(df.mode().iloc[0])
print(df_filled_mode)
```

```
   First     Second      Third  Forth
0  100.0  30.000000  72.666667     50
1   90.0  45.000000  40.000000     60
2   95.0  56.000000  80.000000     70
3   95.0  43.666667  98.000000     80
   First  Second  Third  Forth
0  100.0    30.0   80.0     50
1   90.0    45.0   40.0     60
2   95.0    56.0   80.0     70
3   95.0    45.0   98.0     80
   First  Second  Third  Forth
0  100.0    30.0   40.0     50
1   90.0    45.0   40.0     60
2   90.0    56.0   80.0     70
3   95.0    30.0   98.0     80
```

```
   First  Second  Third  Forth
0  100.0    30.0    NaN     50
1   90.0    45.0   40.0     60
2    NaN    56.0   80.0     70
3   95.0     NaN   98.0     80
```

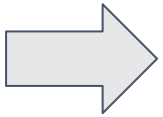
Implementasi Python:

Mengisi dengan Nilai Tetap (ditentukan sendiri)

```
# Mengisi missing values dengan nilai tetap, misalnya 0 atau
'Unknown'
df_filled_fixed = df.fillna(0)
print(df_filled_fixed)

# Untuk kolom tertentu
df_filled_fixed = df.fillna({'First': 0, 'Third': 'Unknown'})
print(df_filled_fixed)
```

	First	Second	Third	Forth
0	100.0	30.0	NaN	50
1	90.0	45.0	40.0	60
2	NaN	56.0	80.0	70
3	95.0	NaN	98.0	80



	First	Second	Third	Forth
0	100.0	30.0	0.0	50
1	90.0	45.0	40.0	60
2	0.0	56.0	80.0	70
3	95.0	0.0	98.0	80

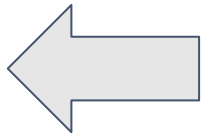
	First	Second	Third	Forth
0	100.0	30.0	Unknown	50
1	90.0	45.0	40.0	60
2	0.0	56.0	80.0	70
3	95.0	NaN	98.0	80

Implementasi Python:
Mengisi dengan Interpolasi

Interpolasi adalah metode untuk memperkirakan nilai antara dua titik data yang sudah diketahui.

```
# Interpolasi missing values
df_interpolated = df.interpolate()
print(df_interpolated)
```

	First	Second	Third	Forth
0	100.0	30.0	NaN	50
1	90.0	45.0	40.0	60
2	92.5	56.0	80.0	70
3	95.0	56.0	98.0	80



	First	Second	Third	Forth
0	100.0	30.0	NaN	50
1	90.0	45.0	40.0	60
2	NaN	56.0	80.0	70
3	95.0	NaN	98.0	80

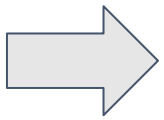
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.interpolate.html

Implementasi Python:
Mengisi dengan Forward/Backward

```
# Forward fill missing values
df_ffill = df.ffill()
print(df_ffill)

# Backward fill missing values
df_bfill = df.bfill()
print(df_bfill)
```

	First	Second	Third	Forth
0	100.0	30.0	NaN	50
1	90.0	45.0	40.0	60
2	NaN	56.0	80.0	70
3	95.0	NaN	98.0	80



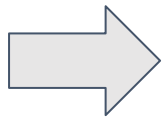
	First	Second	Third	Forth
0	100.0	30.0	NaN	50
1	90.0	45.0	40.0	60
2	90.0	56.0	80.0	70
3	95.0	56.0	98.0	80

	First	Second	Third	Forth
0	100.0	30.0	40.0	50
1	90.0	45.0	40.0	60
2	95.0	56.0	80.0	70
3	95.0	NaN	98.0	80

Implementasi Python:
Mengisi dengan Prediksi

```
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=2)
df_imputed = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)
print(df_imputed)
```

	First	Second	Third	Forth
0	100.0	30.0	NaN	50
1	90.0	45.0	40.0	60
2	NaN	56.0	80.0	70
3	95.0	NaN	98.0	80



	First	Second	Third	Forth
0	100.0	30.0	69.0	50.0
1	90.0	45.0	40.0	60.0
2	97.5	56.0	80.0	70.0
3	95.0	43.0	98.0	80.0

```
1 from sklearn.metrics.pairwise import nan_euclidean_distances
2 nan_euclidean_distances(df,df)
```

```
array([[ 0.        , 23.80476143, 46.38965402, 43.01162634],
       [23.80476143,  0.        , 49.27473998, 71.07742258],
       [46.38965402, 49.27473998,  0.        , 29.12043956],
       [43.01162634, 71.07742258, 29.12043956,  0.        ]])
```

https://scikit-learn.org/1.5/modules/generated/sklearn.impute.KNNImputer.html#sklearn.impute.KNNImputer

B. Menghilangkan Duplikasi

- Dalam beberapa kasus, duplikasi data dapat mempengaruhi hasil analisis, sehingga baris yang duplikat perlu dihapus.
- **Mengapa duplikasi data perlu dihilangkan:**
  - Menghindari distorsi statistik (tidak akurat)
  - Mengurangi overfitting dalam model
  - Efisiensi pengolahan data
  - Menghindari bias dalam hasil analisis

Implementasi Python:
Menghilangkan Duplikasi

```
df = pd.DataFrame({
    'brand': ['Yum Yum', 'Yum Yum', 'Indomie',
'Indomie', 'Indomie'],
    'style': ['cup', 'cup', 'cup', 'pack', 'pack'],
    'rating': [4, 4, 3.5, 15, 5]})

print(df)
# Menghapus baris yang duplikat
df_no_duplicates = df.drop_duplicates(subset=['brand',
'style'], keep='last')
df_no_duplicates
```

```
    brand style  rating
0  Yum Yum   cup     4.0
1  Yum Yum   cup     4.0
2  Indomie   cup     3.5
3  Indomie  pack    15.0
4  Indomie  pack     5.0
```

	brand	style	rating
1	Yum Yum	cup	4.0
2	Indomie	cup	3.5
4	Indomie	pack	5.0

C. Mengatasi Outliers

- Outliers biasanya dihapus dari data atau dilakukan imputasi seperti halnya data yang hilang

Implementasi Python:
Mengatasi Outliers

```
data = {'penjualan': [1200, 1300, 700, 1400, 1700,
3000, 1500, 1600, 1750, 1900, 2000, 2100],
        'pengeluaran_iklan': [200, 250, 180, 230, 300,
350, 220, 270, 310, 330, 380, 400]}
df = pd.DataFrame(data)

# Menghapus outliers berdasarkan IQR
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1

df_no_outliers = df[~((df < (Q1 - 1.5 * IQR)) | (df >
(Q3 + 1.5 * IQR))).any(axis=1)]
df_no_outliers
```

	penjualan	pengeluaran_iklan
0	1200	200
1	1300	250
2	700	180
3	1400	230
4	1700	300
6	1500	220
7	1600	270
8	1750	310
9	1900	330
10	2000	380
11	2100	400

Penjualan (sorted): [700,1200,1300,1400,1500,1600,1700,1750,1900,2000,2100,3000]
Q1=1375; Q3=1925; IQR=550 → outliers < 1375-1.5*550 = 550 or outliers > 1375+1.5*550=2200

D. Memperbaiki Kesalahan Data

- Data yang salah dapat:
  - diperbaiki
  - dihapus
  - dilakukan imputasi seperti halnya data yang hilang

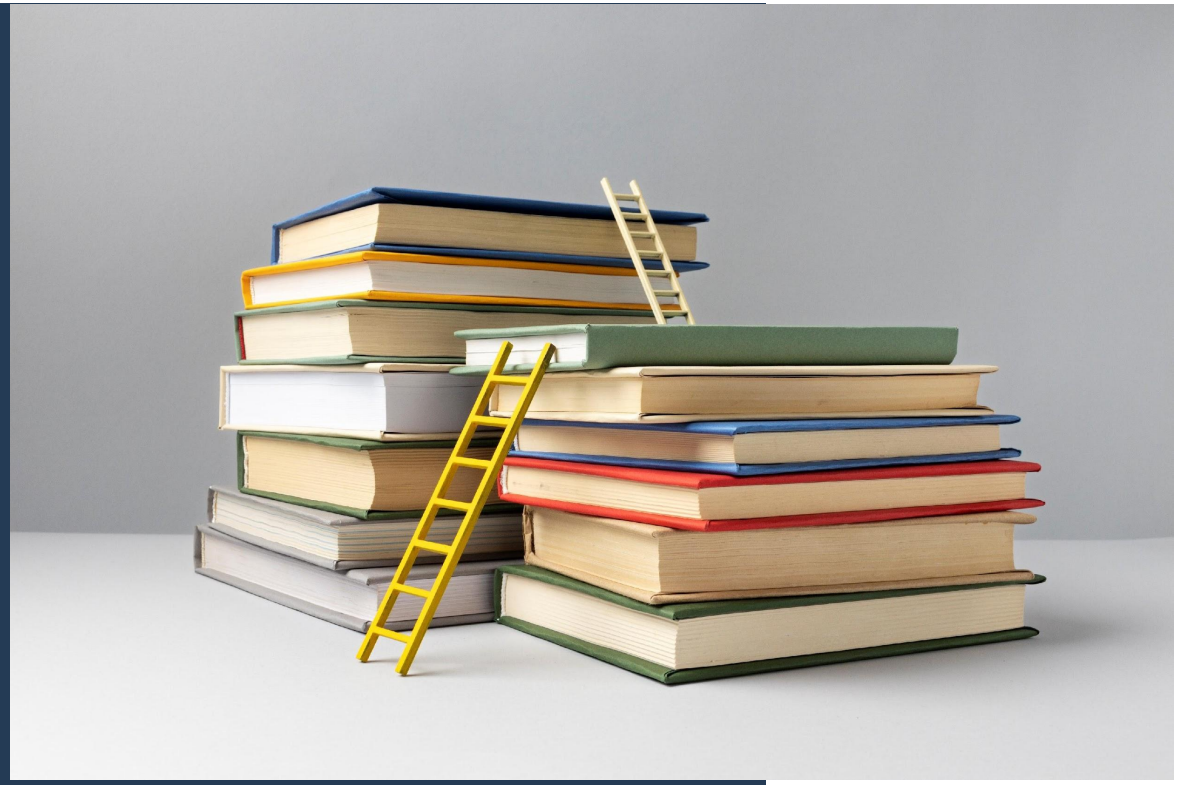
Implementasi Python:
Memperbaiki Kesalahan Data

```
# Memperbaiki tipe data
data = {'tanggal': ['2023-21-15', '2023-15-17', '2023-03-19',
'2023-04-21']}
df = pd.DataFrame(data)
df['tanggal'] = pd.to_datetime(df['tanggal'], errors='coerce')
print(df)
```

```
      tanggal
0         NaT
1         NaT
2  2023-03-19
3  2023-04-21
```

```
# Mengganti nilai yang salah
df = pd.DataFrame({
    'brand': ['Yum Yum', 'Yum Yum', 'Indomie', 'Indomie', 'Indomie'],
    'style': ['cup', 'cup', 'cup', 'pack', 'pack'],
    'rating': [4, 4, 3.5, 15, 5]})
df['style'] = df['style'].replace({'pack': 'sachet'})
print(df)
```

```
     brand   style  rating
0  Yum Yum     cup     4.0
1  Yum Yum     cup     4.0
2  Indomie     cup     3.5
3  Indomie  sachet    15.0
4  Indomie  sachet     5.0
```

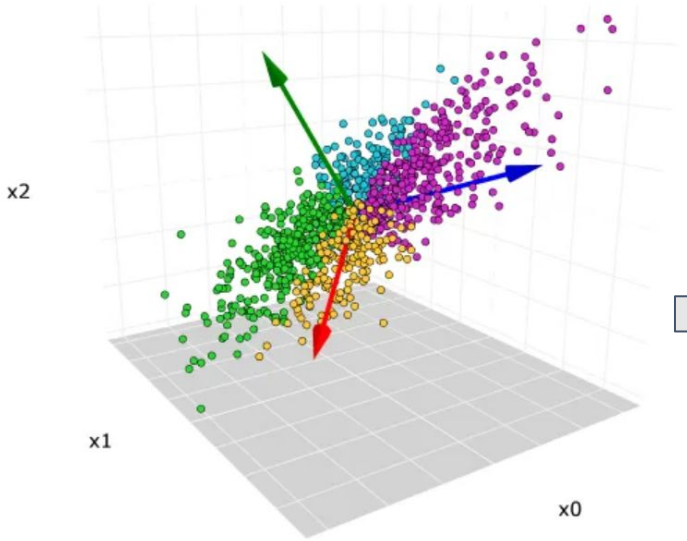


Konstruksi Data

Proses dalam Persiapan Data

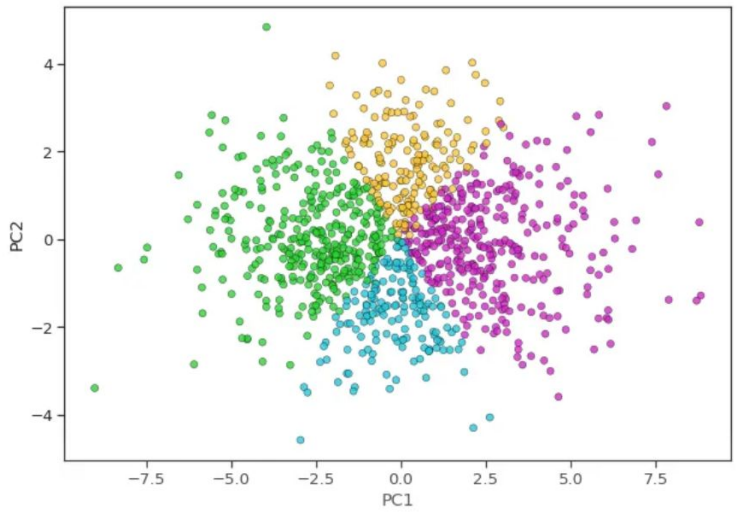
1. **Pemilihan Data**
   a. *Record selection*
   b. *Feature selection*
2. **Perbaikan Data**
   a. *Mengisi missing values*
   b. *Perbaikan error*
   c. *Penanganan outlier*
   d. *Penghapusan duplikasi*
3. **Konstruksi Data**
   a. Dimensionality reduction
   b. Transformasi data
   c. Encoding
4. **Integrasi Data**
   a. *Data Join*
   b. *Append*

Dimensionality Reduction



3 dimensi

sumber gambar: towardsdatascience.com



2 dimensi

1. Mengurangi dimensi data namun tetap mempertahankan “semantik” data
2. Berbeda dengan “Feature Selection” dimana dimensi dikurangi dengan cara menghapus atribut sehingga “semantik” data berubah
3. Contoh teknik:
   - PCA (Principal Component Analysis)
   - t-Distributed Stochastic Neighbor Embedding (t-SNE)
   - Uniform Manifold Approximation and Projection (UMAP)

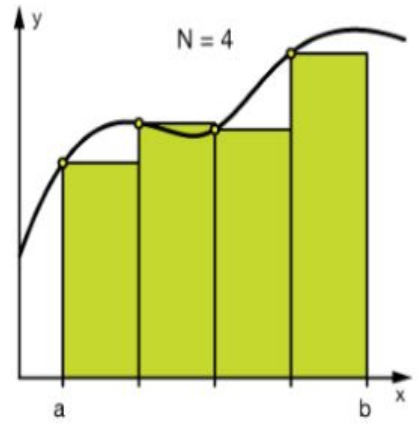
Principal Component Analysis (PCA)

1. Normalisasi data input: Setiap atribut berada dalam kisaran yang sama
2. Hitung vektor ortonormal (unit), yaitu Komponen utama
3. Setiap data input (vektor) adalah kombinasi linear dari nilai k vektor komponen utama.
4. Komponen utama diurutkan dalam urutan decreasing (menurun) “Signifikansi” atau kekuatan
5. Karena komponen diurutkan, ukuran data dapat dikurangi dengan menghilangkan komponen yang lemah, yaitu komponen-komponen dengan varian rendah.

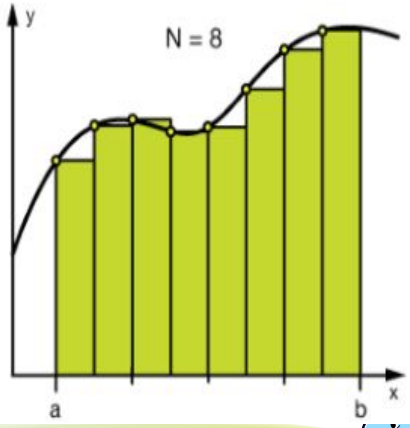
Transformasi Data

1. Data Smoothing (Binning)

Data binning adalah proses mengelompokkan nilai-nilai data kontinu menjadi interval atau "bin" yang lebih besar, yang mewakili rentang nilai tertentu.



- **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 redudansi 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”.



https://pandas.pydata.org/docs/reference/api/pandas.cut.html

2. Attribute Construction

- Atribut baru dibuat untuk membantu proses data mining dari atribut yang sudah ada.
- Contoh: Membuat atribut baru 'area' dari atribut 'tinggi' dan 'lebar'.

atribut lama di drop kah ?
→ kalau ada hypotesis kalau atribut baru pasti lebih bagus, boleh di drop, kalau engga ya boleh di keep

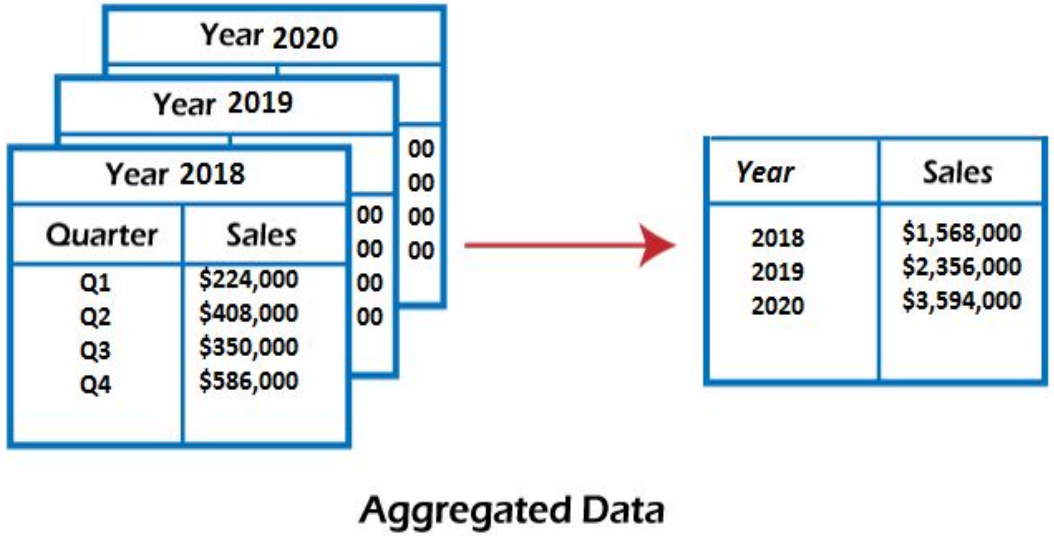
3. Data Generalization

- Mengubah atribut data tingkat rendah menjadi atribut data tingkat tinggi menggunakan hierarki konsep.
- Kegunaan:
  - Mendapatkan gambaran data yang lebih jelas.
  - Mengurangi kedetailan data
- Contoh:
  - Data umur dapat berupa (10, 20, 30, 40) dalam sebuah dataset dapat ditransformasikan ke tingkat konseptual yang lebih tinggi menjadi nilai kategoris anak-anak, remaja, dewasa, tua.
  - Menggantikan alamat rumah individu dengan nama kota atau negara bagian.

4. Data Aggregation

- **Metode menyimpan dan menyajikan data dalam format ringkasan.**
- Contoh:

  Kumpulan data laporan penjualan suatu perusahaan yang memiliki data penjualan triwulanan setiap tahun dapat di-agregasi untuk mendapatkan laporan penjualan tahunan perusahaan.



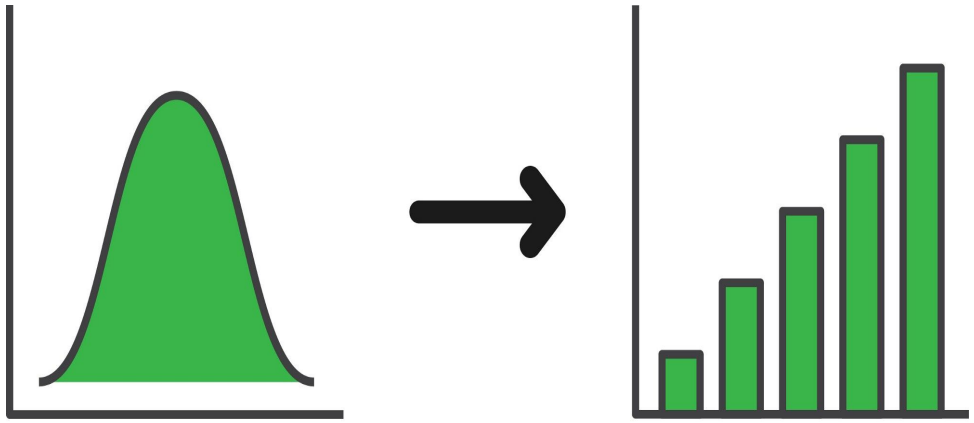
Aggregated Data

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.groupby.html

5. Data Discretization

- Proses mengubah fungsi, model, dan variabel kontinu menjadi diskrit.

- Contoh:
  - misal berat badan < 65 kg (ringan); 65 – 80 kg (mid); > 80 kg (berat).
  - pembulatan sebuah nilai riil ke nilai terdekat



Discretization Process

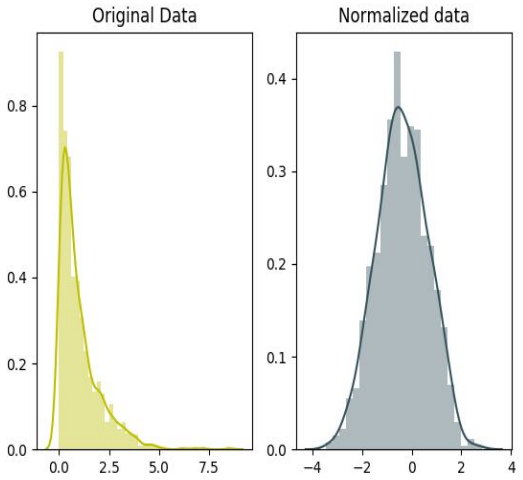
6. Data Normalization

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

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

Misal Xmax dan Xmin 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.



Encoding

	boro	salary	vegan
0	Manhattan	103	0
1	Queens	89	0
2	Manhattan	142	0
3	Brooklyn	54	1
4	Brooklyn	63	1
5	Bronx	219	0

- Ada algoritma Machine Learning yang tidak dapat menerima input berupa data kategorikal, harus berupa data numerik
- Contoh Teknik:
  - ordinal encoding
  - one-hot encoding

memberi index di nilai unik, soalnya data kategorikal gabisa di access..

→ misal ada 10 data unik, ya indexnya 10 gitu

	boro	boro_ordinal	salary
0	Manhattan	2	103
1	Queens	3	89

python: label encoding

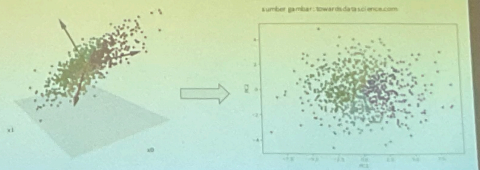
, dipakai kalau data-nya butuh urutan, contoh hari. senin lbh dekat ke selasa drpd senin ke rabu

nilainya hanya boleh 0 dan 1.

	boro	boro_Bronx	boro_Brooklyn	boro_Manhattan	boro_Queens	salary
0	Manhattan	0	0	1	0	103
1	Queens	0	0	0	1	89
2	Manhattan	0	0	1	0	142

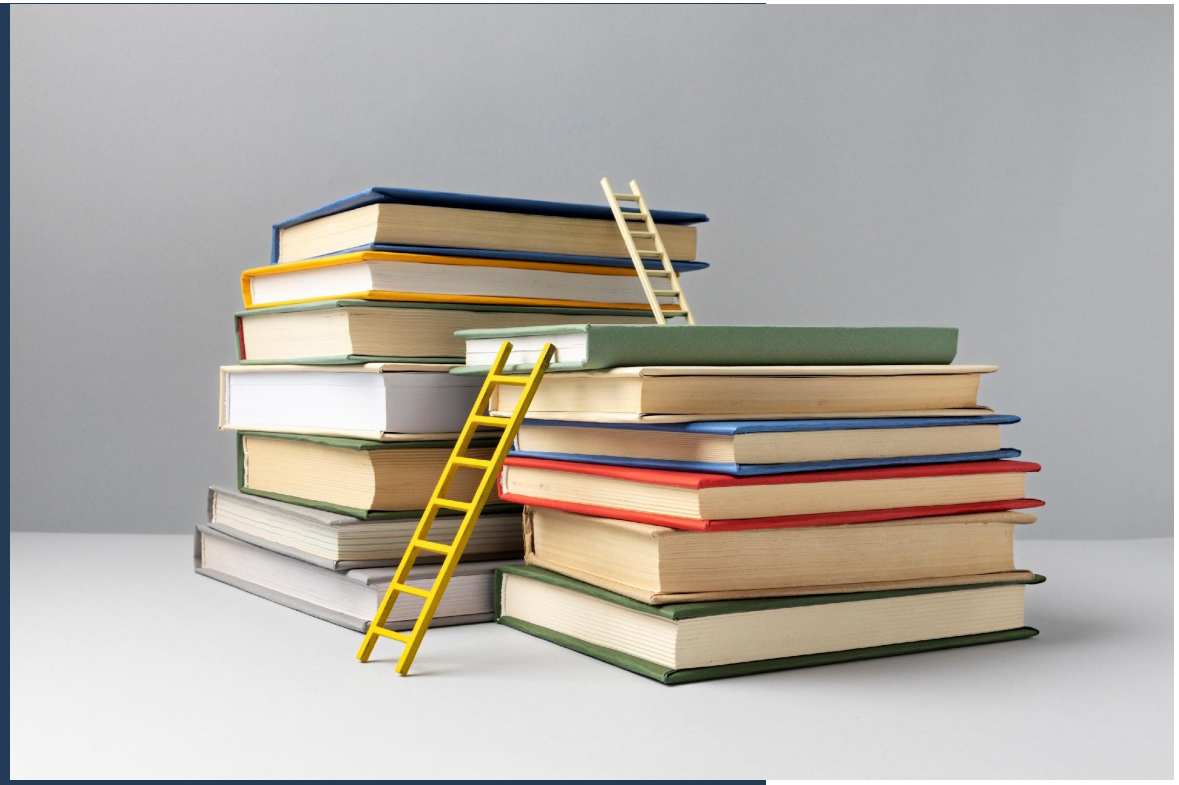
Data Preparation

Dimensionality Reduction



1. Mengurangi dimensi data namun tetap mempertahankan “semantik” data
2. Berbeda dengan “Feature Selection” dimana dimensi dikurangi dengan cara menghapus atribut sehingga “semantik” data berubah
3. Contoh teknik:
   - PCA (Principal Component Analysis)
   - t-Distributed Stochastic Neighbor Embedding (t-SNE)
   - Uniform Manifold Approximation and Projection (UMAP)

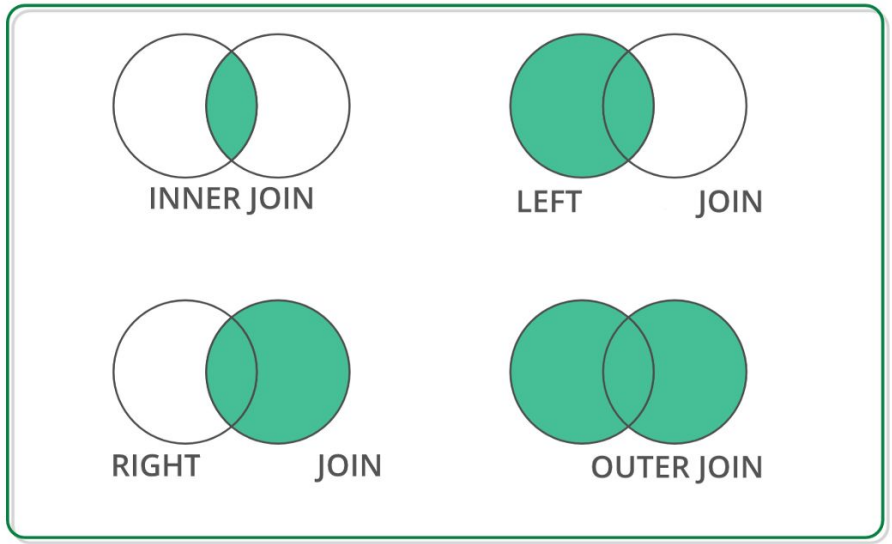
EDUNEX ITB

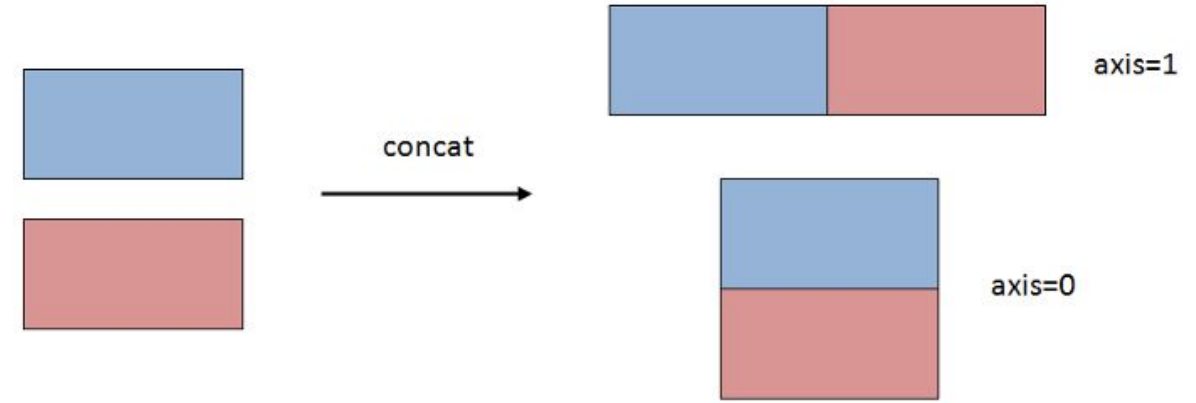


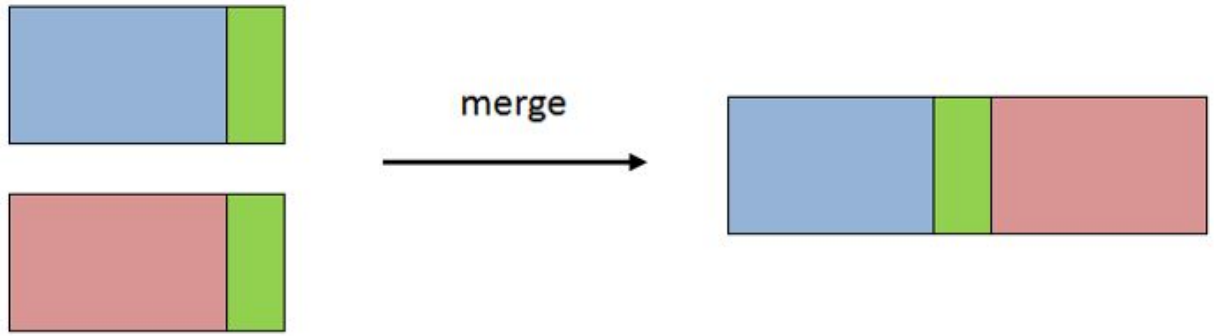
Integrasi Data

Integrasi Data

- **Concat**
- **Join**
- **Merge**







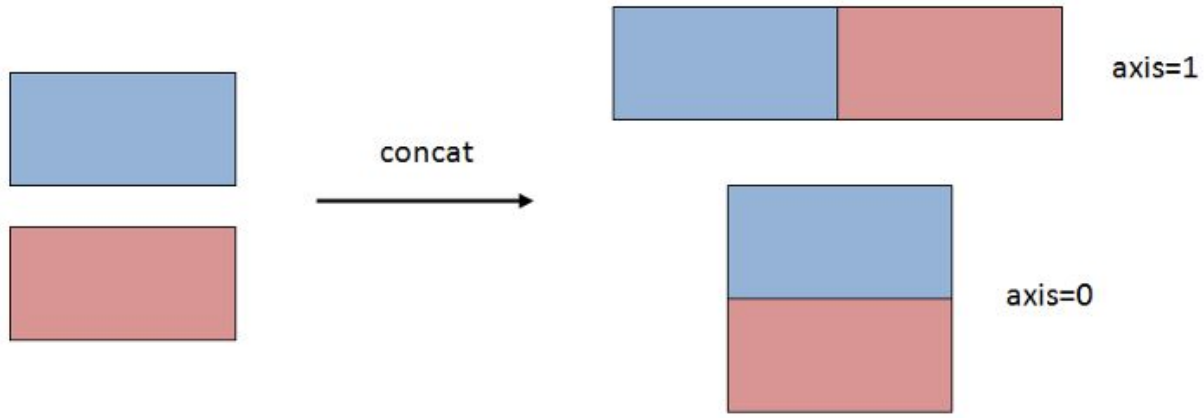
Concat

```
[ ]  df1 = pd.DataFrame({
         "A": ["A0", "A1", "A2"],
         "B": ["B0", "B1", "B2"],},
       index = [0,1,2],
     )
     df1
```

	A	B
0	A0	B0
1	A1	B1
2	A2	B2

```
[ ]  df2 = pd.DataFrame({
         "A": ["C0", "C1", "C2"],
         "B": ["D0", "D1", "D2"],},
       index = [3,4,5],
     )
     df2
```

	A	B
3	C0	D0
4	C1	D1
5	C2	D2



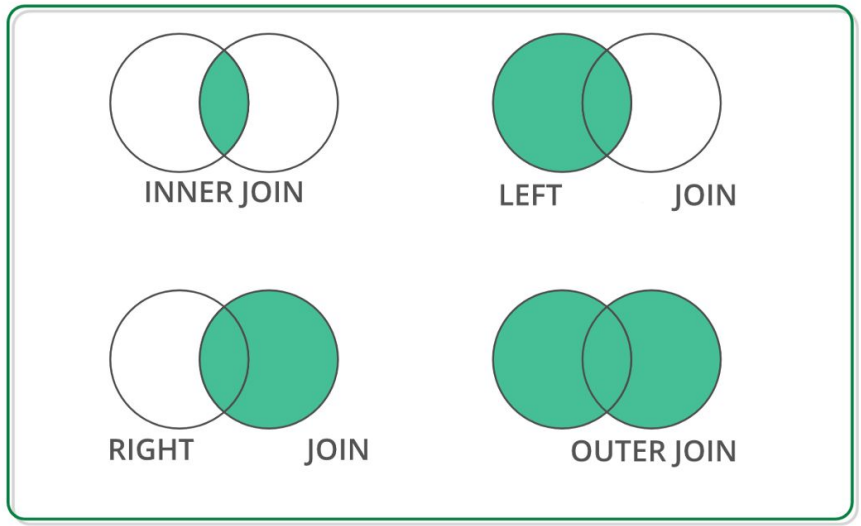

```
frames = [df1, df2]
result1 = pd.concat(frames)
result1
```

	A	B
0	A0	B0
1	A1	B1
2	A2	B2
3	C0	D0
4	C1	D1
5	C2	D2

```
result2 = pd.concat(frames, axis=1)
result2
```

	A	B	A	B
0	A0	B0	NaN	NaN
1	A1	B1	NaN	NaN
2	A2	B2	NaN	NaN
3	NaN	NaN	C0	D0
4	NaN	NaN	C1	D1
5	NaN	NaN	C2	D2

Join




```
[ ]  df1 = pd.DataFrame({
         "A": ["A0", "A1", "A2"],
         "B": ["B0", "B1", "B2"],},
       index = [0,1,2],
     )
     df1
```

	A	B
0	A0	B0
1	A1	B1
2	A2	B2

```
[ ]  df3 = pd.DataFrame({
         "B": ["B0", "B1", "B3"],
         "C": ["C0", "C1", "C3"],
       },
       index = [0,1,3],
     )
     df3
```

	B	C
0	B0	C0
1	B1	C1
3	B3	C3

```
result3 = pd.concat([df1, df3], axis=1)
result3
```

	A	B	B	C
0	A0	B0	B0	C0
1	A1	B1	B1	C1
2	A2	B2	NaN	NaN
3	NaN	NaN	B3	C3

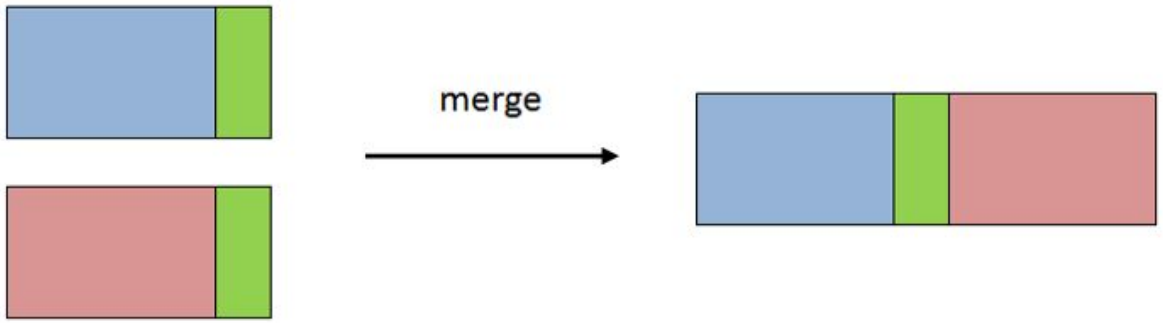
```
result4 = pd.concat([df1, df3], axis=1, join='inner')
result4
```

	A	B	B	C
0	A0	B0	B0	C0
1	A1	B1	B1	C1

```
result5 = pd.concat([df1, df3], join='inner')
result5
```

	B
0	B0
1	B1
2	B2
0	B0
1	B1
3	B3

Merge




```
[ ]  df1 = pd.DataFrame({
         "A": ["A0", "A1", "A2"],
         "B": ["B0", "B1", "B2"],},
       index = [0,1,2],
     )
     df1
```

	A	B
0	A0	B0
1	A1	B1
2	A2	B2

```
[ ]  df3 = pd.DataFrame({
         "B": ["B0", "B1", "B3"],
         "C": ["C0", "C1", "C3"],
       },
       index = [0,1,3],
     )
     df3
```

	B	C
0	B0	C0
1	B1	C1
3	B3	C3

```
result6 = pd.merge(df1, df3, on='B')
result6
```

	A	B	C
0	A0	B0	C0
1	A1	B1	C1

kalau mau lebih aman, pakai merge dibanding concat?

Salam
Semoga Bermanfaat

supervised : ada target labelnya
unsupervised : ga ada