

Learning Progress Review Week 10

By
Omicron



Anggota Kelompok



**Anugrah Yazid
Ghani**

[https://www.linkedin.com/
in/anugrah-yazid-7253bb221/](https://www.linkedin.com/in/anugrah-yazid-7253bb221/)



Fajar Achmad

[https://www.linkedin.com/
in/fajar-achmad-755945111/](https://www.linkedin.com/in/fajar-achmad-755945111/)



**Edo Mohammad
Hadad Gibran**

[https://www.linkedin.com/
in/edo-gibran-38505a142/](https://www.linkedin.com/in/edo-gibran-38505a142/)



Muhammad Fikri Fadila

[https://www.linkedin.com/
in/muhammad-fikri-fadila-a551161a6/](https://www.linkedin.com/in/muhammad-fikri-fadila-a551161a6/)




Daftar Isi




1. Advanced Visualization



2. Introduction to Machine Learning



3. Data Preprocessing Machine Learning



Advanced Visualization





Seaborn

- *Library* visualisasi data Python berdasarkan matplotlib.
- Seaborn menyediakan *interface* tingkat tinggi untuk menggambar grafik statistik yang menarik dan informatif.

Syntax import Seaborn library :

```
import seaborn as sns
```

Seaborn VS Matplotlib

Seaborn

Pros :

- Sederhana sehingga mudah dipelajari.
- *Handling* pandas dataframe yang lebih nyaman.

Cons :

- Menggunakan memori lebih banyak.
- Menghindari *overlapping* plot dengan bantuan *default theme*.
- Tidak ada *pie chart*.

Matplotlib

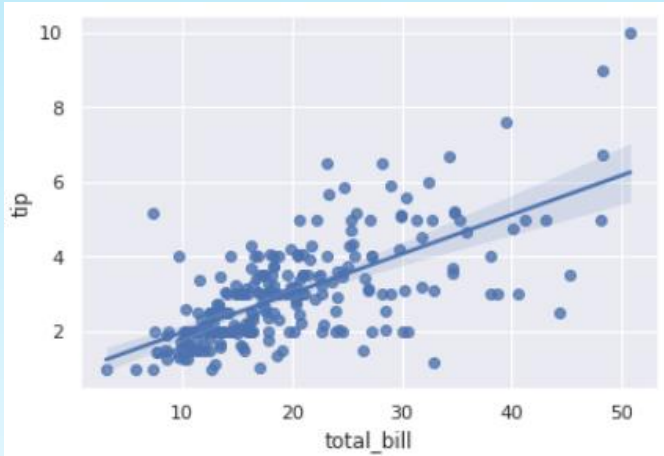
Pros :

- Dapat membuka dan menggunakan banyak gambar secara bersamaan.
- Dapat dimodifikasi dan *robust*.

Cons :

- Kompleks dan *syntax* yang panjang.

Using Seaborn to Plotting Regression Line Chart



Contoh :

Membuat grafik garis regresi untuk melihat hubungan diantara kolom total bill dan tip.

Syntax :

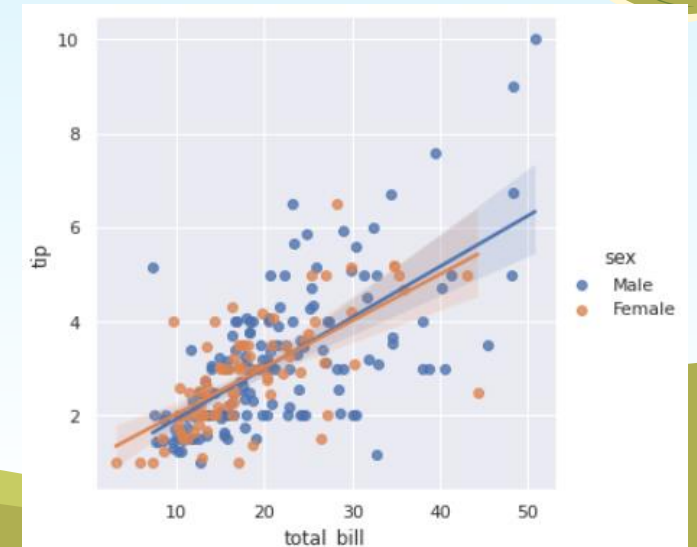
```
sns.regplot(x="total_bill", y="tip", data=tips)
```

Contoh :

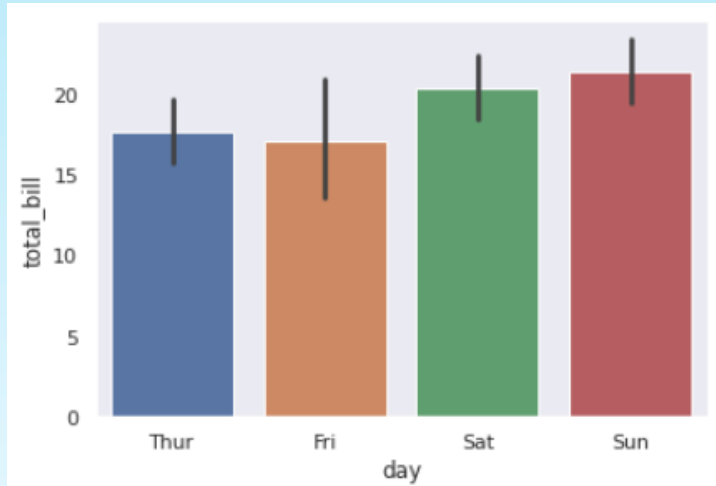
Membuat grafik garis regresi untuk melihat hubungan diantara kolom total bill dan tip dan mengklasifikasikannya berdasarkan jenis kelamin.

Syntax :

```
sns.lmplot(x="total_bill", y="tip",  
hue="sex", data=tips)
```



Using Seaborn to Plotting Bar Chart



Contoh :

Membuat grafik batang untuk melihat tren perolehan total bill dari hari Kamis – Minggu.

Syntax :

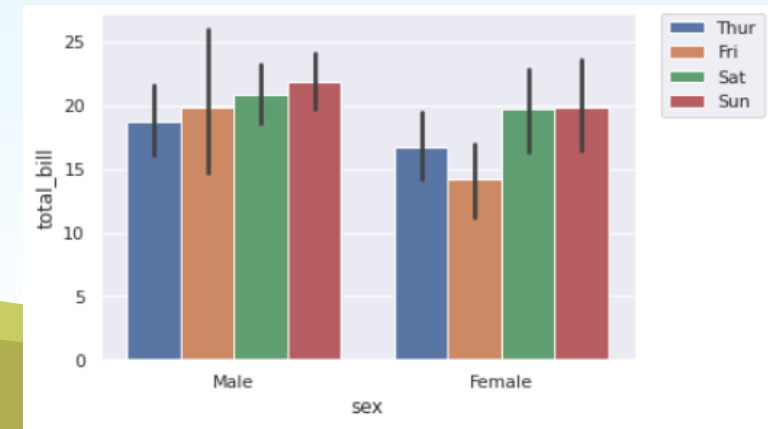
```
sns.barplot(x="day", y="total_bill", data=tips)
```

Contoh :

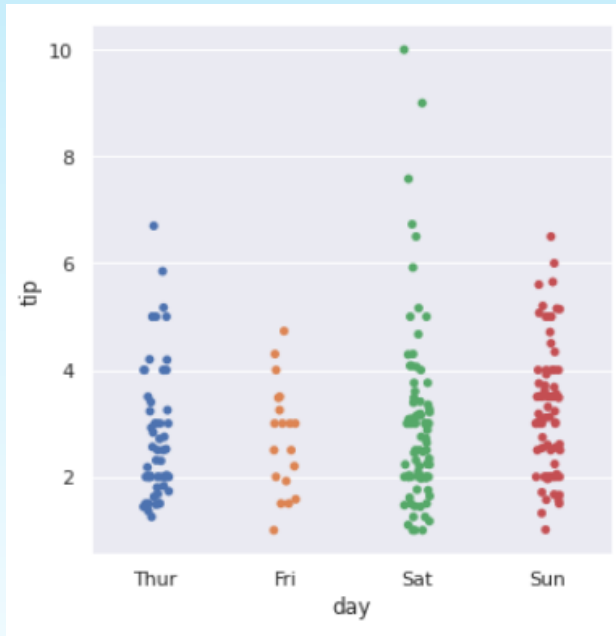
Membuat grafik batang untuk melihat tren perolehan total bill antara pria dan wanita, dan mengklasifikasikannya berdasarkan hari.

Syntax :

```
sns.barplot(x="sex", y="total_bill",  
hue="day", data=tips)  
plt.legend(bbox_to_anchor=(1.05,1),  
loc=2, borderaxespad=0.)
```



Using Seaborn to Plotting Catplot



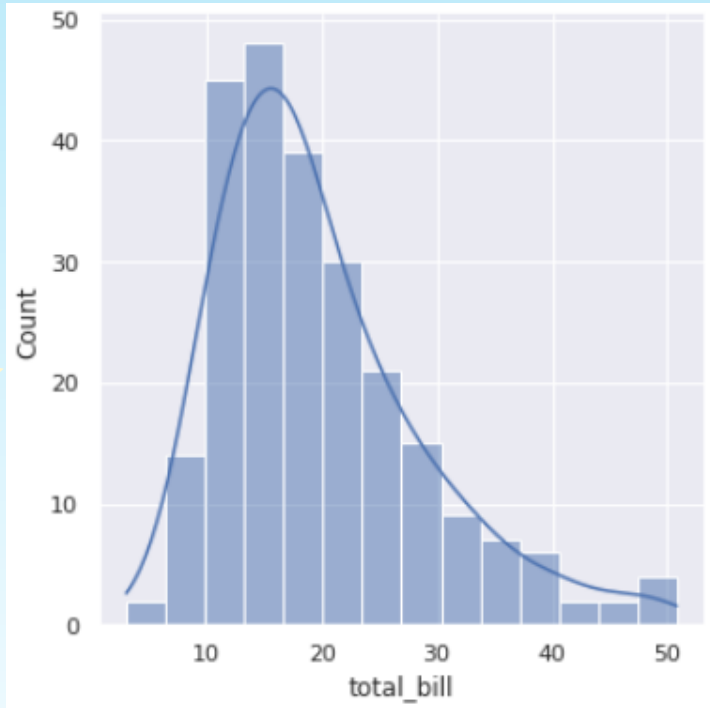
Contoh :

Membuat catplot untuk melihat persebaran data perolehan tip dari hari Kamis – Minggu.

Syntax :

```
sns.catplot(x="day", y="tip", data=tips)
```

Using Seaborn to Plotting Histogram



Contoh :

Membuat histogram untuk mengetahui persebaran data pada kolom “total_bill”.

Syntax :

```
sns.displot(tips.total_bill, kde=True)
```

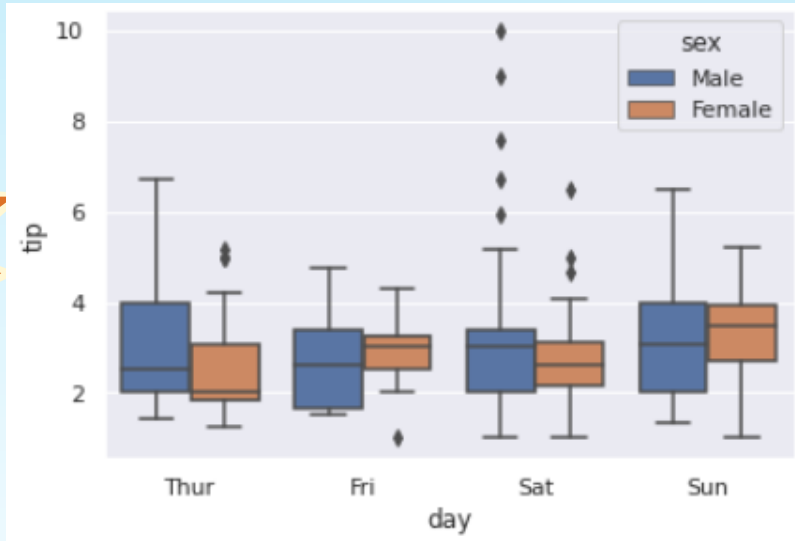
Using Seaborn to Plotting Boxplot

Contoh :

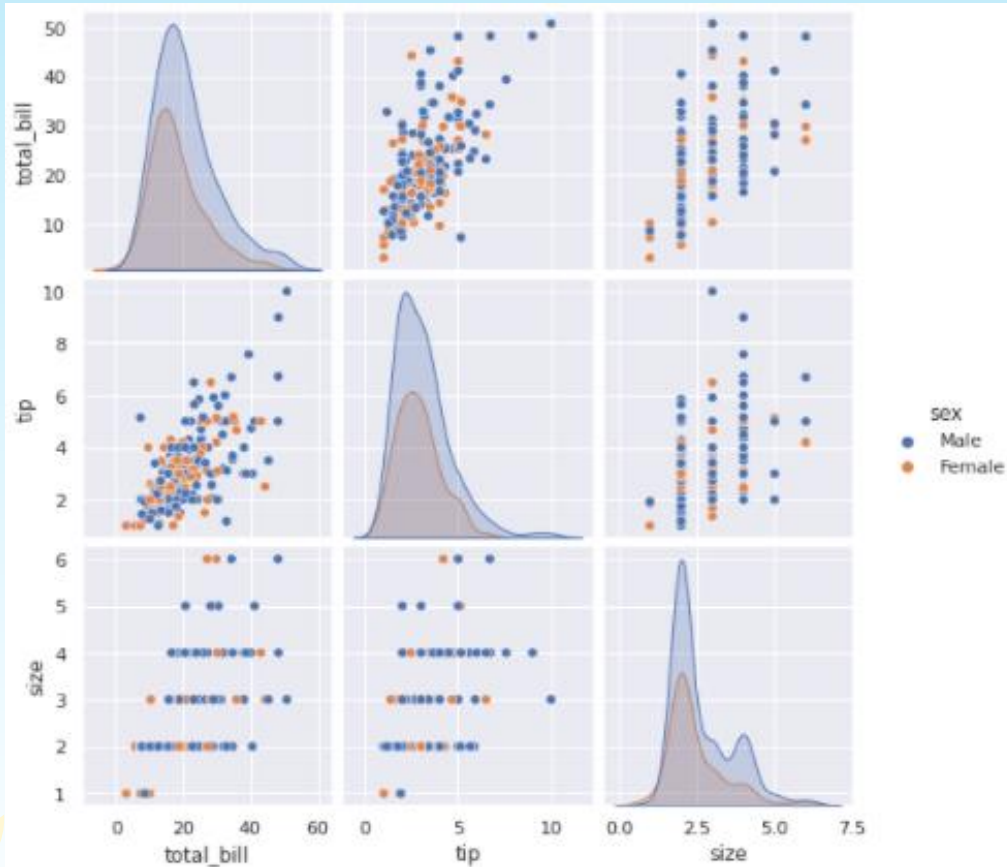
Membuat boxplot untuk mengetahui kecenderungan distribusi data Perolehan tip hari Kamis – Minggu berdasarkan jenis kelamin pelanggan.

Syntax :

```
sns.boxplot(x="day", y="tip", hue='sex', data=tips)
```



Using Seaborn to Plotting Scatterplot Matrix



Contoh :

Membuat matriks scatterplot untuk melihat hubungan persebaran data perolehan tip, total bill, dan size berdasarkan jenis kelamin pelanggan.

Syntax :

```
sns.pairplot(tips, hue="sex")
```

Ket :

“tips” adalah nama data yang digunakan.

Using Seaborn to Plotting Correlation Heatmap

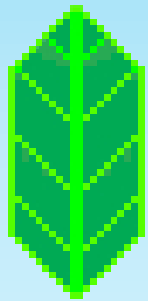


Contoh :

Membuat heatmap untuk melihat hubungan diantara kolom – kolom pada data “tips”.

Syntax :

```
sns.heatmap(tips.corr(), annot=True)
```



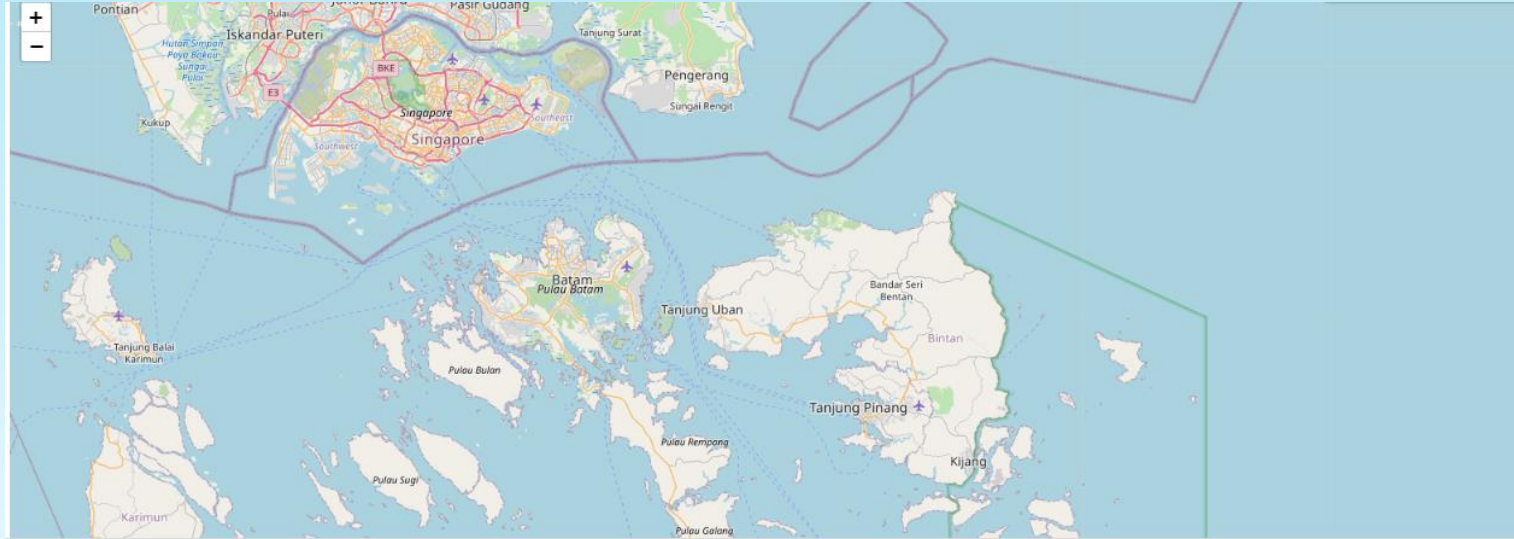
Folium

- Folium memudahkan untuk memvisualisasikan data yang telah dimanipulasi dengan Python pada peta interaktif.
- Folium memungkinkan untuk mengaplikasikan data ke peta untuk visualisasi *choropleth* serta meneruskan visualisasi vektor/raster/HTML yang kaya sebagai penanda pada peta.

Syntax import Folium library :

```
import folium
```

Contoh Hasil Penggunaan Folium



Membuat peta dengan *library* Folium.

Syntax :

```
map = folium.Map(location=[1.3396704621219861, 103.98355242289945],  
zoom_start=13)  
map
```



- 

```
import wordcloud
```


Contoh Hasil Penggunaan Wordcloud



Membuat wordcloud dengan *library* Wordcloud.

Syntax :

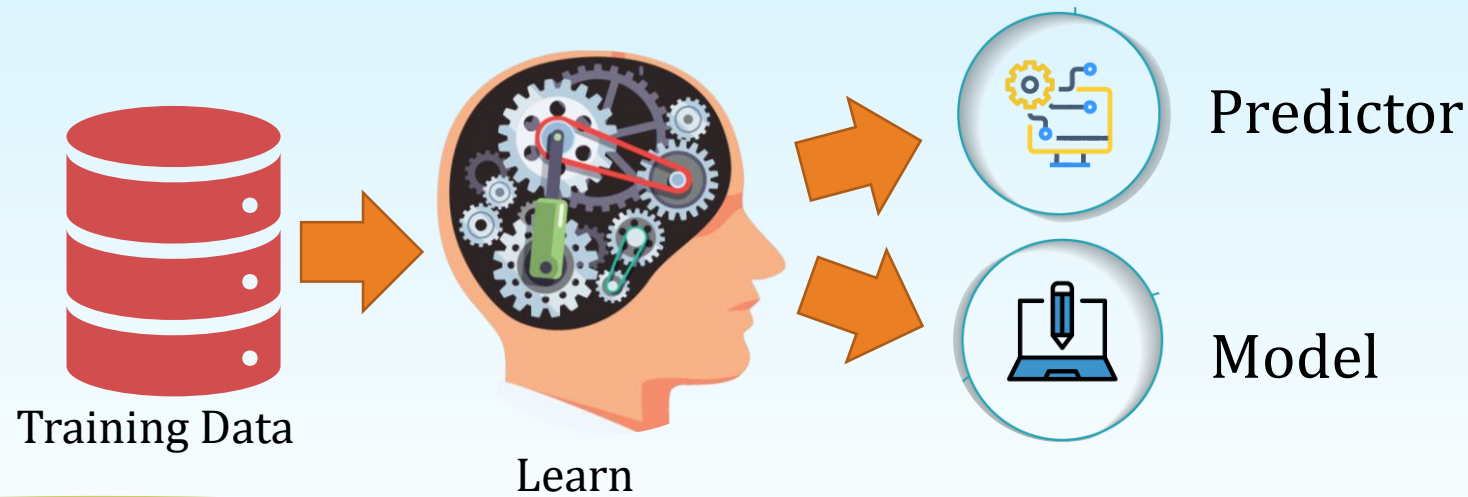
```
from wordcloud import WordCloud, STOPWORDS
wordcloud = WordCloud(width = 3000, height = 2000, random_state=1,
background_color='white', colormap='Pastell1', collocations=False,
stopwords = STOPWORDS).generate(text)
wordcloud
```

Introduction to Machine Learning



Apa itu Machine Learning ?

- ❏ Cabang Kecerdasan Buatan (AI)
- ❏ Pengembangan algoritma berdasarkan data empiris
- ❏ Digunakan untuk melakukan klasifikasi dan prediksi



Traditional Programming

vs

Machine Learning



Rules dibuat oleh seorang *programmer* menggunakan programnya



Rules dari data dirumuskan secara otomatis oleh algoritma

Data Mining

vs

Machine Learning

- ❏ Mengekstrak informasi yang berguna dari sejumlah besar data
- ❏ Digunakan untuk memahami aliran data
- ❏ Banyak campur tangan manusia
- ❏ Cenderung ke penelitian menggunakan metode seperti ML

- ❏ Menggunakan algoritma dari data serta pengalaman masa lalu
- ❏ Komputer belajar dan memahami aliran data
- ❏ Tidak ada upaya manusia yang diperlukan setelah desain
- ❏ Sistem belajar mandiri dan melatih untuk melakukan tugas cerdas

Types of Machine Learning

1. Supervised Learning

- Training data memiliki target class
- Klasifikasi, regresi/prediksi

2. Unsupervised Learning

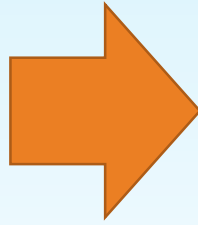
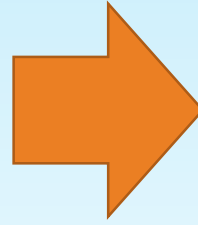
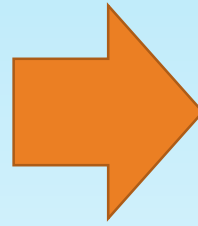
- Training data tidak memiliki target class
- *Clustering, association rules*

3. Semi-supervised Learning

- Sebagian training data memiliki outputs

4. Reinforcement Learning

- Rewards diberikan Ketika agent sukses dalam tugas tertentu
- Dan punishment Ketika salah

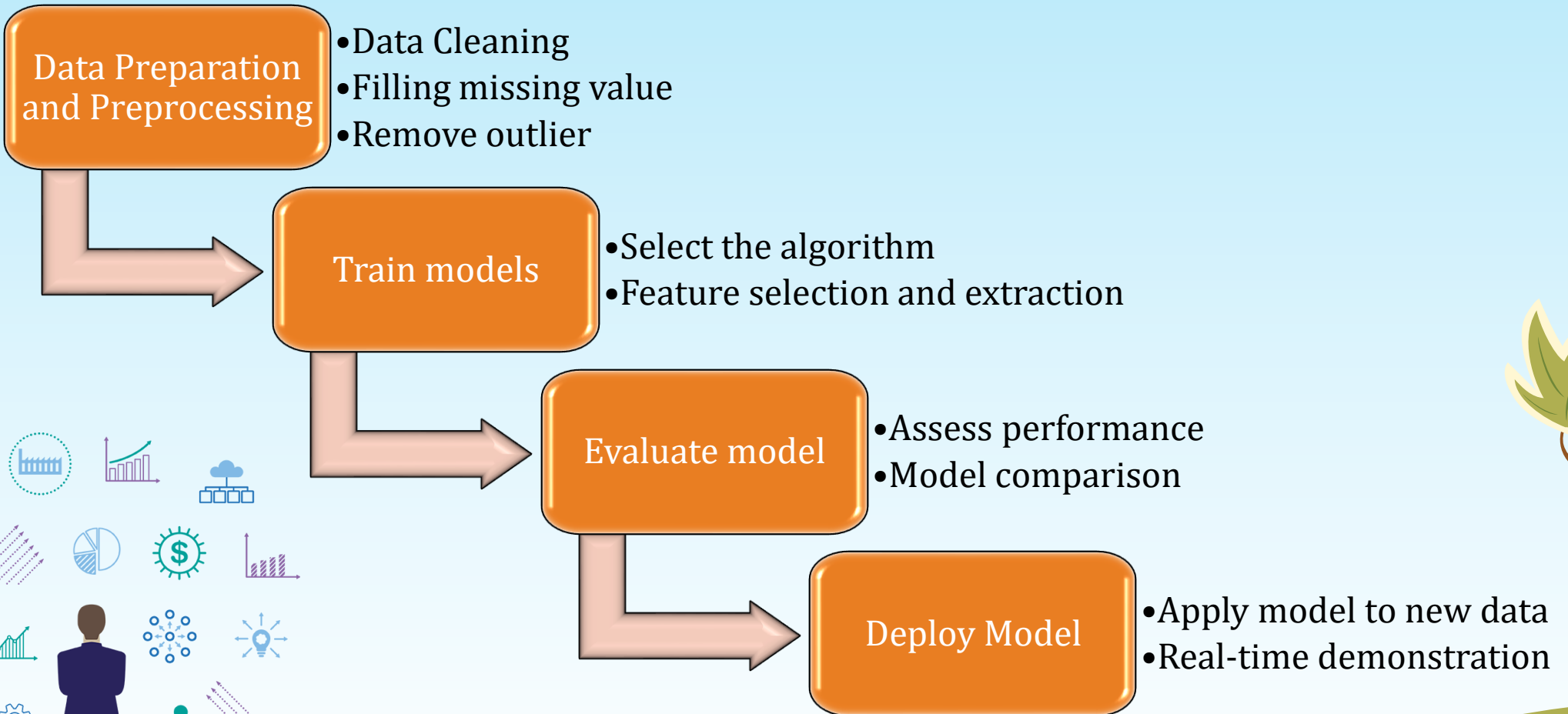


Index	Attrb 1	Attrb 2	Class
1	Aa bb	1.01	Yes
2	Aa cc	2.01	No

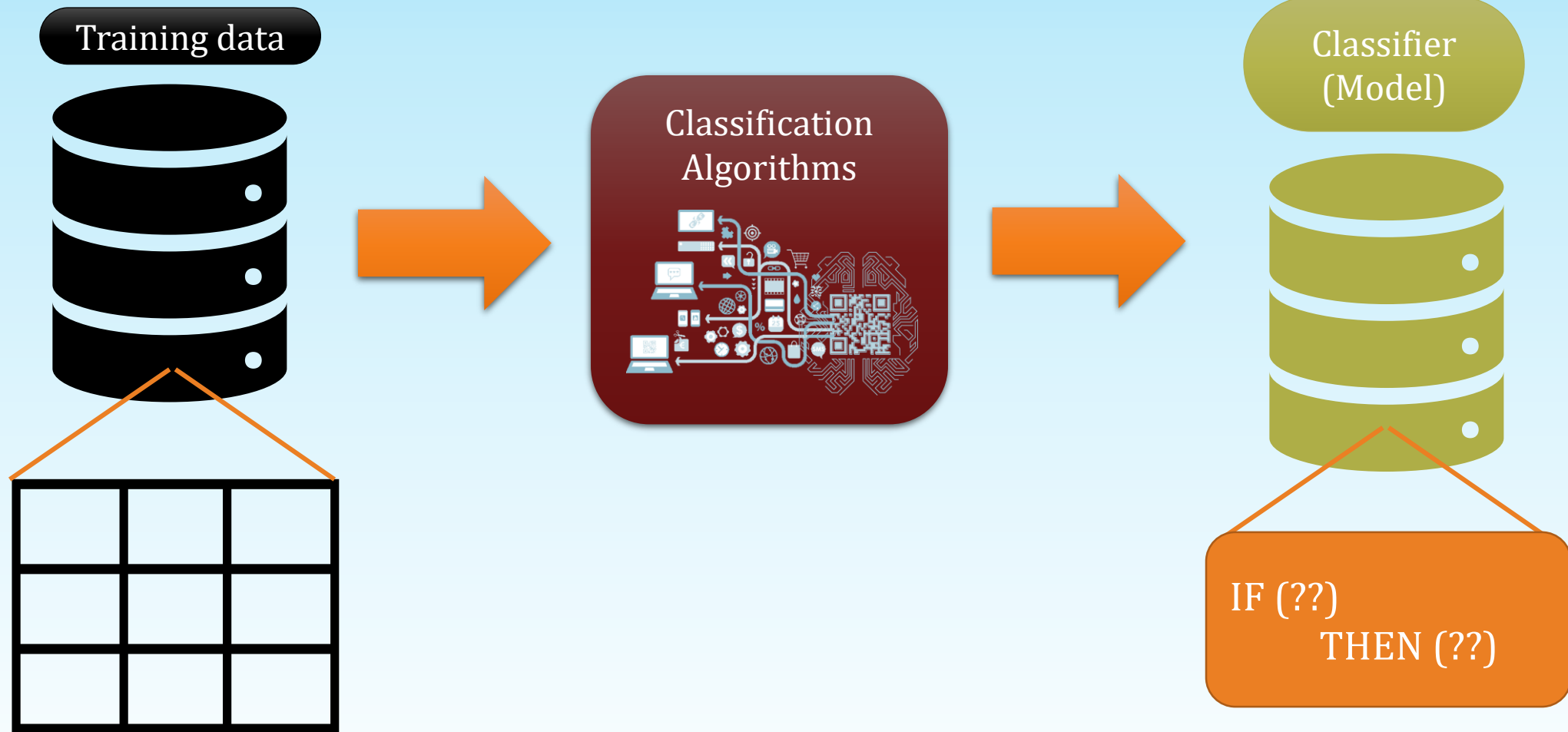
Index	Attrb 1	Attrb 2
1	Aa bb	1.01
2	Aa cc	2.01

Index	Attrb 1	Attrb 2	Class
1	Aa bb	1.01	Yes
2	Aa cc	2.01	No
3	Aa dd	1.02	
4	Aa ee	2.02	Yes

Stage in Machine Learning



Konstruksi model



Bias and variance

Bias

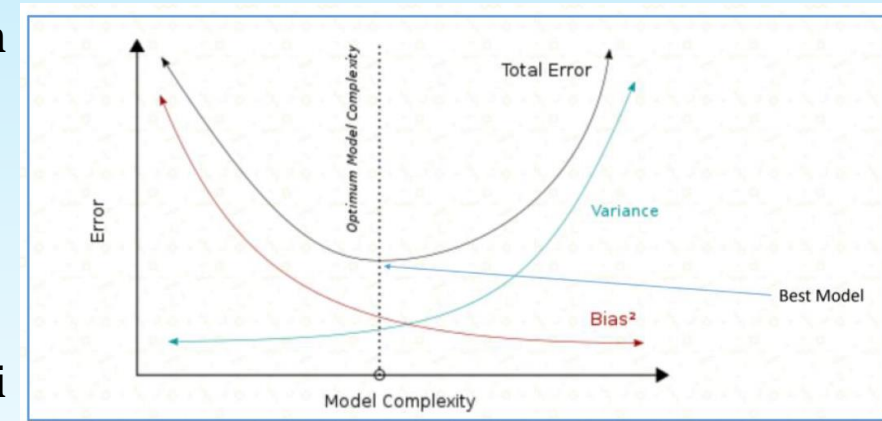
Bias adalah perbedaan antara rata rata hasil prediksi dari model ML yang kita kembangkan dengan data nilai yang sebenarnya.

Bias yang tinggi dikarenakan dalam pembangunan model ML dilakukan terlalu sederhana (*oversimplified*).

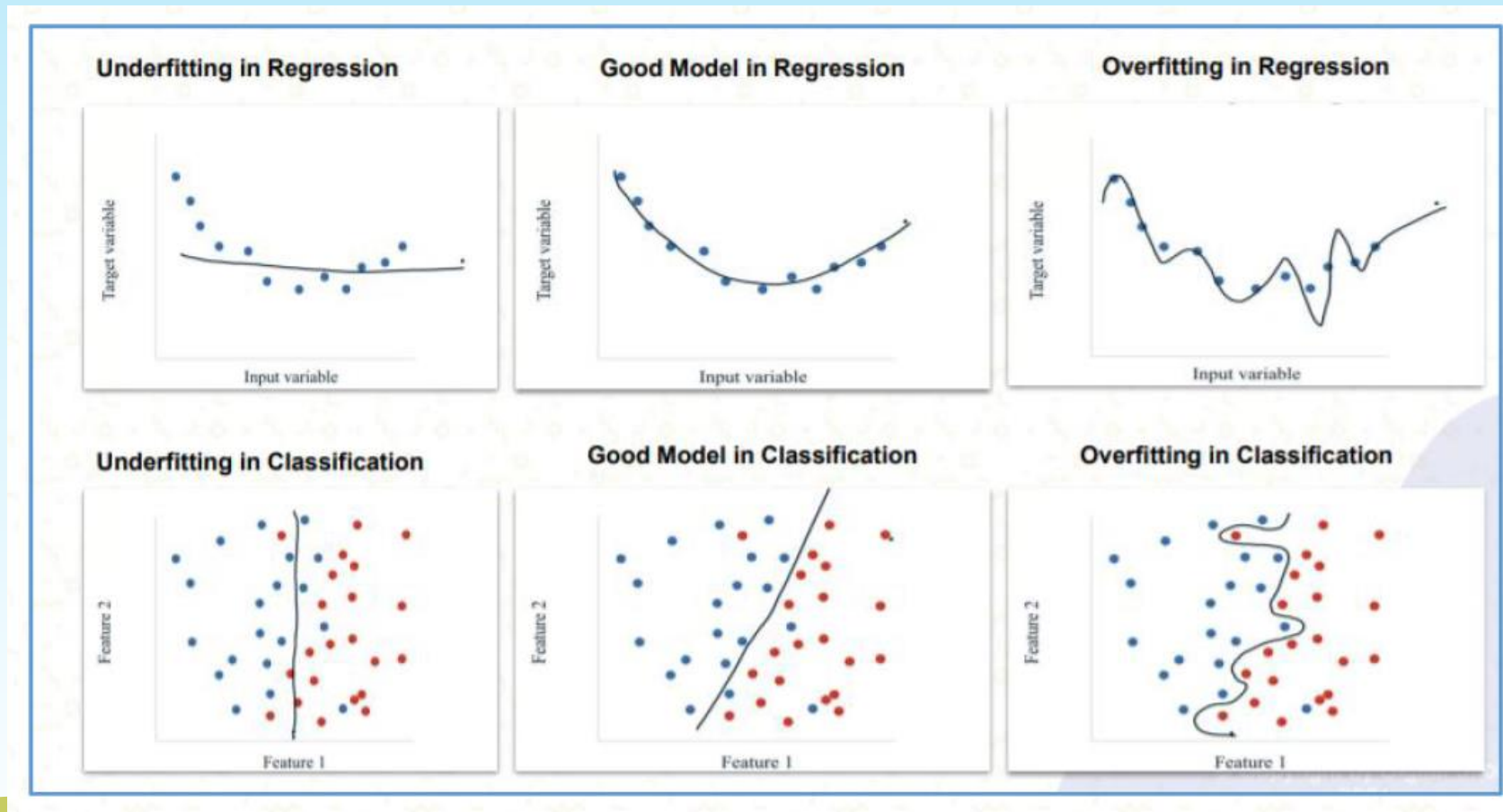
Variance

Variance adalah variabel dari prediksi yang memberikan kita informasi perserbaran data hasil prediksi.

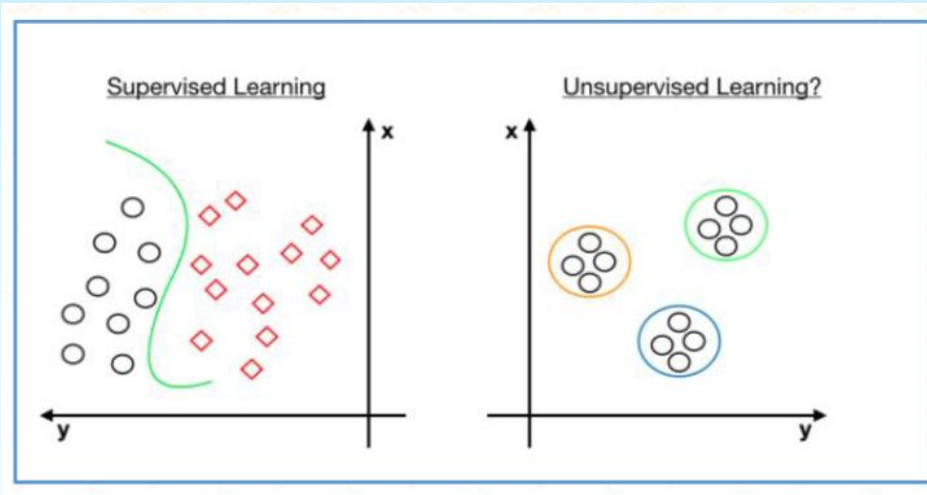
Model yang memiliki variance tinggi memiliki korelasi kuat hanya pada *training set*, sehingga akan berkinerja baik pada *training data* saja.



Underfitting and overfitting



Supervised vs Unsupervised Learning



Supervised = Mempelajari data untuk memprediksi output.

- Kita tahu target label, sehingga kita membuat model untuk memprediksi label.

Unsupervised = Menemukan pattern/ characteristic dari data.

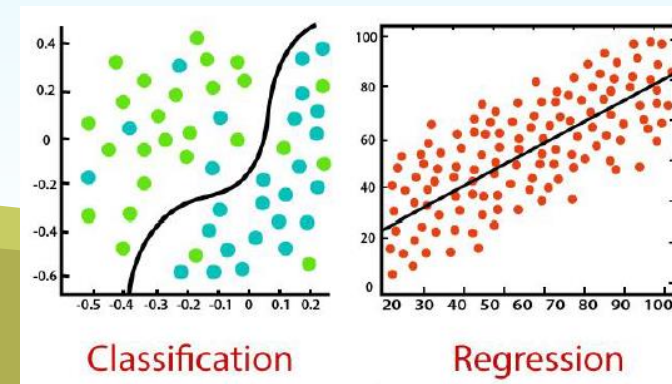
- Kita tidak mengetahui target label, sehingga kita membuat model yang mencoba mengelompokkan data.

Supervised Learning

Classification (klasifikasi) = metode yang menarik beberapa kesimpulan dari nilai input yang

diberikan pada saat training dan kemudian akan memprediksi label/kelas untuk data baru.

- *Regression* (regresi) = metode yang mencoba untuk menentukan kekuatan dan karakter hubungan antara satu variabel dependen dan serangkaian variabel lainnya (variabel independen).
- Algoritma regresi = nilai kontinu (seperti harga, gaji, usia, dll).
- Algoritma klasifikasi = nilai diskrit (seperti stroke atau normal, spam atau bukan spam, dll)
- Keduanya adalah *supervised learning*



Classification, regression, clustering

price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built
221900.0	3	1.00	1180	5650	1.0	0	0	3	7	1180	0	1955
538000.0	3	2.25	2570	7242	2.0	0	0	3	7	2170	400	1951
180000.0	2	1.00	770	10000	1.0	0	0	3	6	770	0	1933
604000.0	4	3.00	1960	5000	1.0	0	0	5	7	1050	910	1965
510000.0	3	2.00	1680	8080	1.0	0	0	3	8	1680	0	1987

Regression (house price dataset)

	ID	Sex	Marital status	Age	Education	Income	Occupation	
0	100000001	0	0	67	2	124670		1
1	100000002	1	1	22	1	150773		1
2	100000003	0	0	49	1	89210		0
3	100000004	0	0	45	1	171565		1
4	100000005	0	0	53	1	149031		1

Clustering (customer dataset)

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60162	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
...
5105	18234	Female	80.0	1	0	Yes	Private	Urban	83.75	NaN	never smoked	0
5106	44673	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked	0
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked	0
5108	37544	Male	51.0	0	0	Yes	Private	Rural	166.29	25.6	formerly smoked	0
5109	44579	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.2	Unknown	0

Classification (stroke dataset)

Linear Regression

- Membentuk hubungan antara dua variabel menggunakan garis lurus.

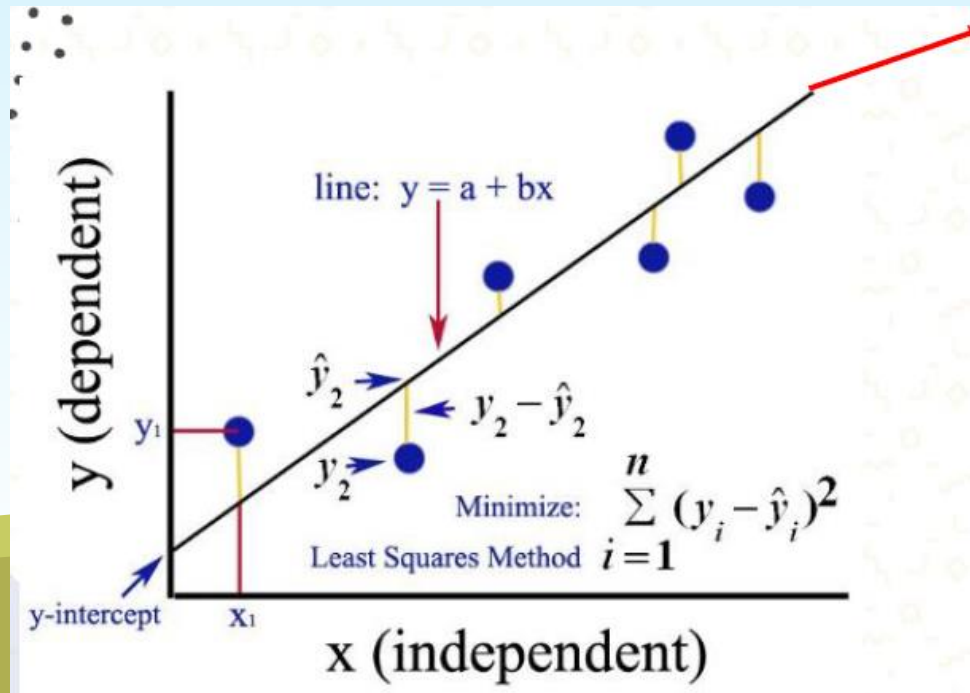
- Simple linear regression: $Y = a + bX + u$
- Multiple linear regression: $Y = a + b_1X_1 + b_2X_2 + b_3X_3 + \dots + b_tX_t + u$

Where:

- Y = the variable that you are trying to predict (dependent variable).
- X = the variable that you are using to predict Y (independent variable).
- a = the intercept.
- b = the slope.
- u = the regression residual.

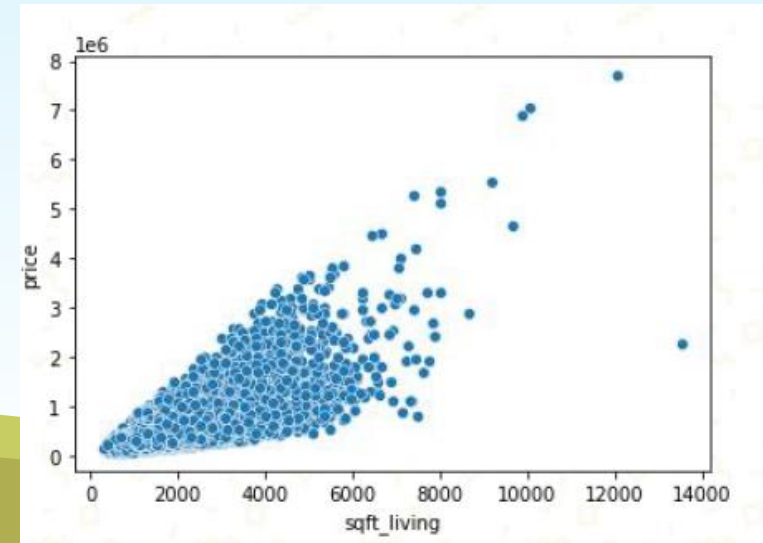
Linear Regression

- Regresi linier mencoba menggambar garis yang paling dekat dengan data dengan menemukan *slope* dan *intercept* dan meminimalkan *regression errors*.
- Ordinary Least Squares (OLS) adalah metode estimasi yang paling umum untuk model linier.



the best line would have the lowest sum of squared errors (SSE)

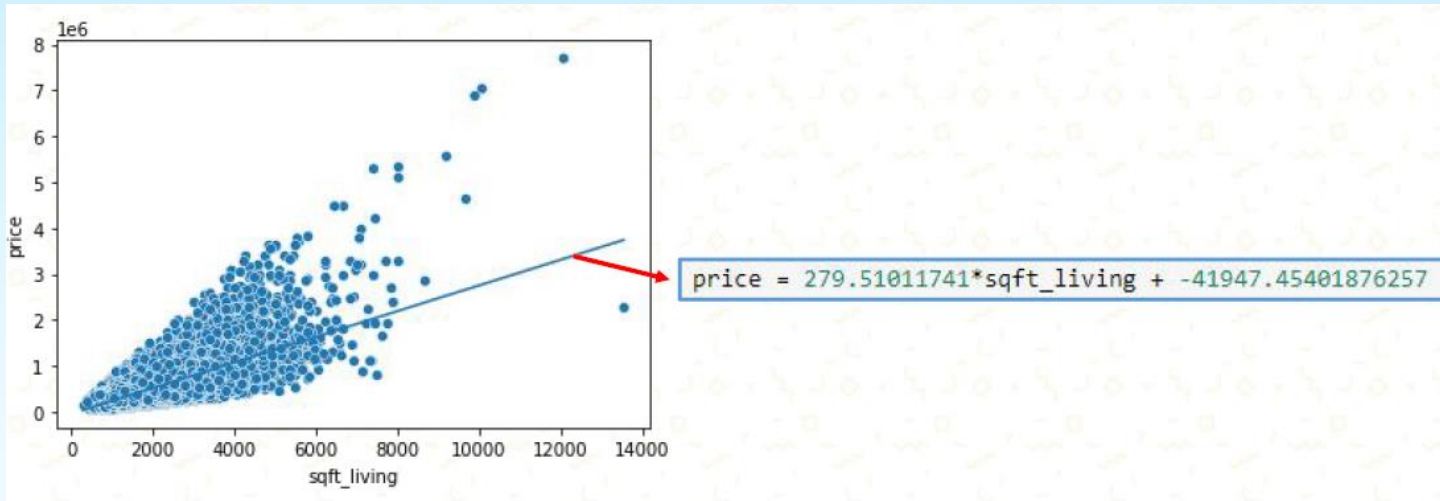
$$\sum_{i=1}^n (Y_i - \sum_{j=1}^p X_{ij}\beta_j)^2$$



Linear Regression

Example

- y (dependent variable) = *price* (house price)
- x (independent variable) = *sqft_living* (square feet)

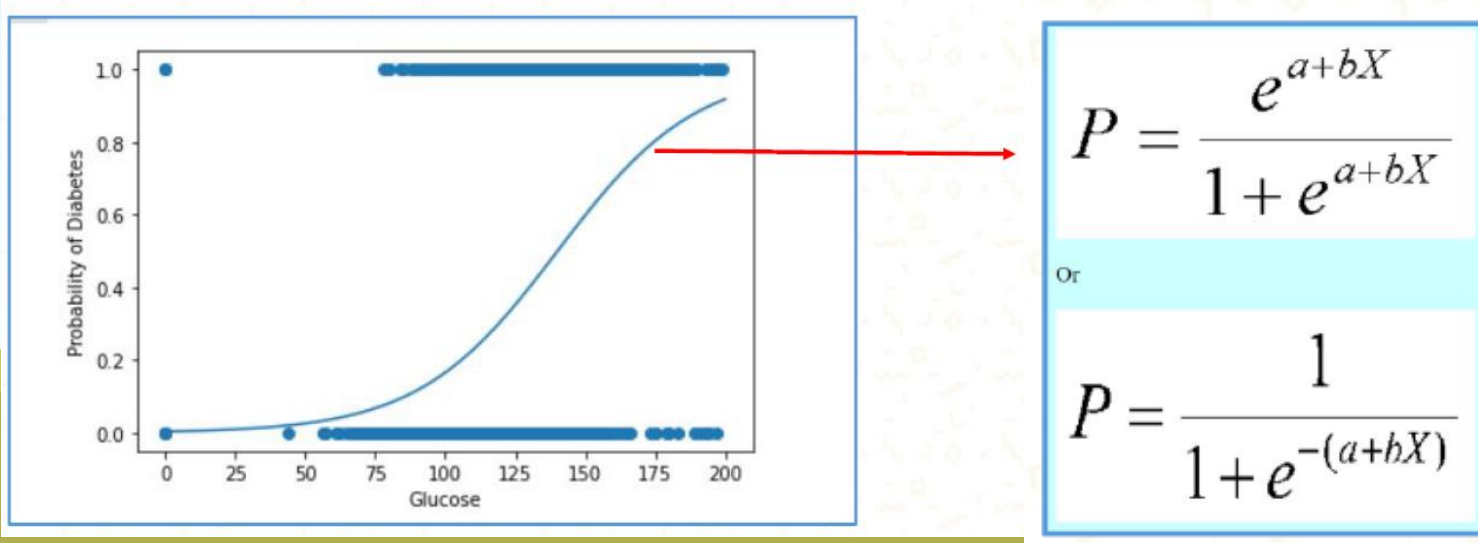


Q = House with 1000 square feet, approximate price?

A = USD 237562.663

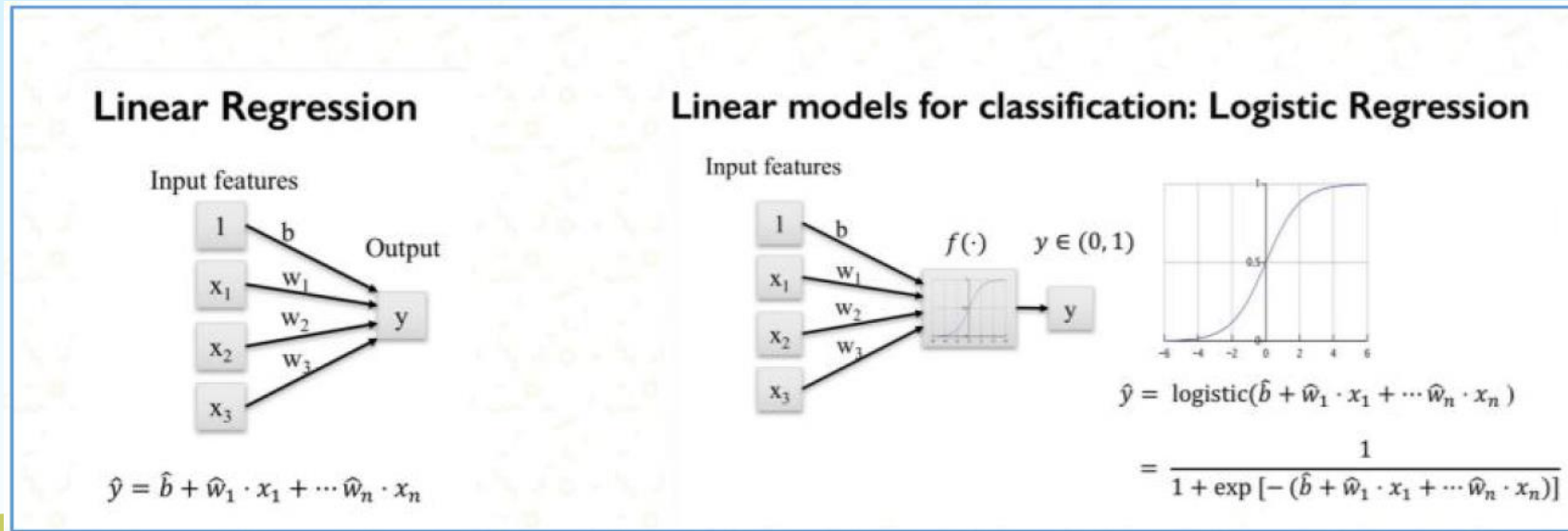
Logistic Regression

- Logistic Regression adalah algoritma klasifikasi Machine Learning yang digunakan untuk memprediksi ketika variabel dependen (target) adalah bersifat kategorik.
- Target adalah variabel biner yang berisi kelas 1 (untuk kasus benar/ya) atau 0 (untuk kasus salah/tidak).

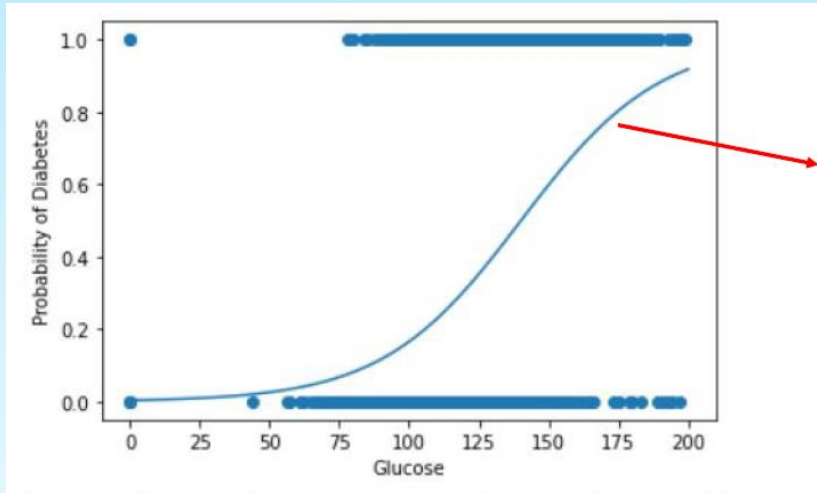


Logistic Regression

- Merupakan sebuah kasus khusus regresi linier di mana responsnya adalah '*log of odds*'.
- Model Regresi Logistik memprediksi $P(Y=1)$ dengan memasukkan data ke fungsi logit.



Logistic Regression



```
p = 1/(1 + np.exp
      (-(0.04033676*x - 5.6523997)))
```

Q = Patient with BG 190 mg/dL, is it diagnosed as diabetes?

A = Probability diabetes is 0.882

Logistic Regression

```
#hold out, dibagi menjadi training dan testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

#scaling
scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

# data preprocessing selesai

#mulai melakukan modelling. model ML learning dari training set
model=LogisticRegression()
model.fit(X_train, y_train)

# membuat prediksi
y_pred = model.predict(X_test)

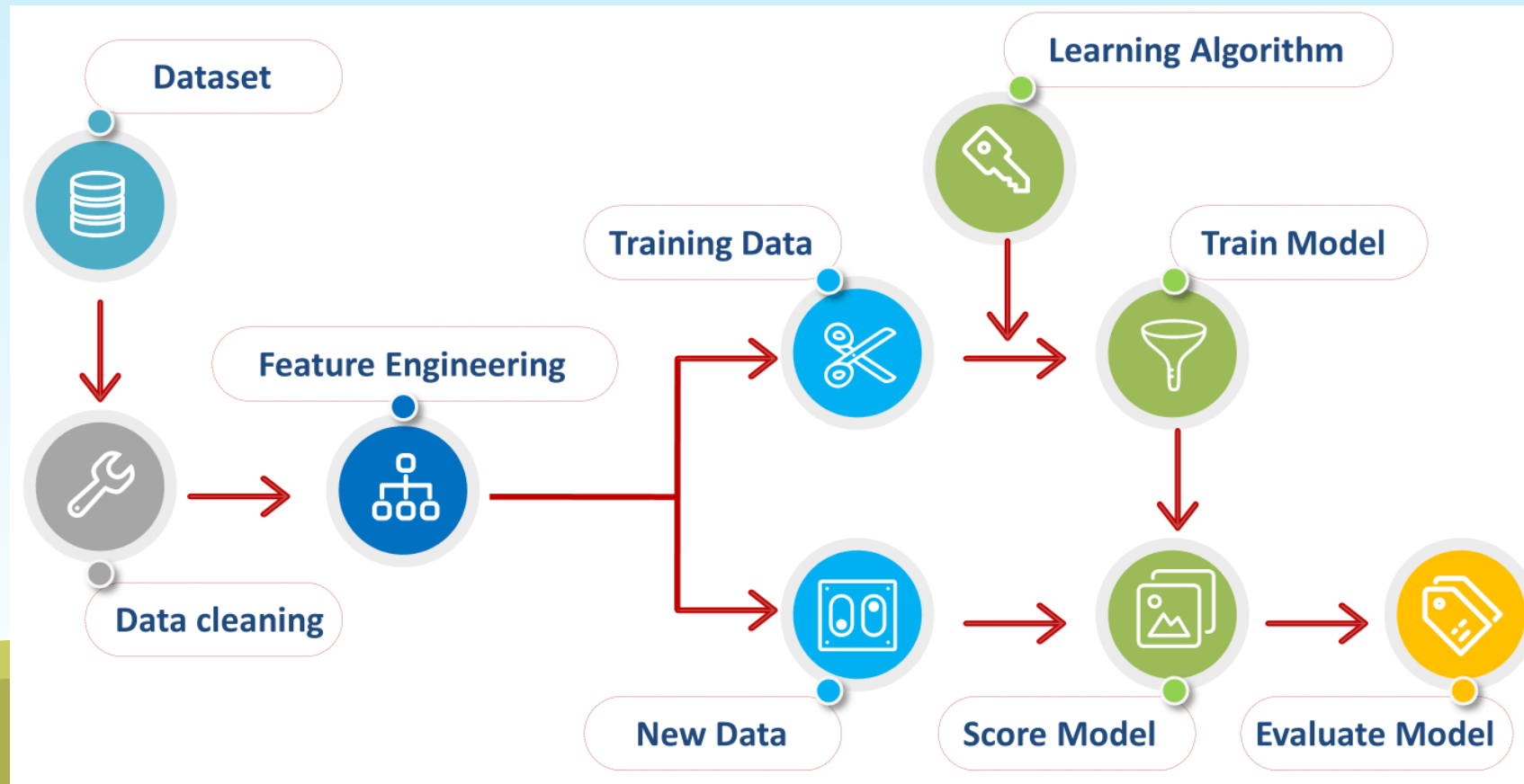
#menghitung performa model, dengan accuracy dll
print('Accuracy ',accuracy_score(y_test, y_pred))
print('Precision ',precision_score(y_test, y_pred, average='macro'))
print('Recall ',recall_score(y_test, y_pred, average='macro'))
print('Confusion matrix ', confusion_matrix(y_test, y_pred))
plot_confusion_matrix(model, X_test, y_test, cmap=plt.cm.Blues)
plt.show()
```

Data Preprocessing Machine Learning



DATA PREPROCESSING

Data preprocessing merupakan sekumpulan teknik yang diterapkan pada dataset untuk menghapus noise, meng-handle missing value, dan data yang tidak konsisten.



APA ITU *FEATURES* & *FEATURES ENGINEERING*?

FEATURES

- Fitur → Properti terukur dari objek yang kita coba analisis.
- Dalam dataset, fitur muncul sebagai **KOLOM**:

	A	B	C	D	E	F	G	H	I	J	K	L
1	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, Mr. male		22	1	0	A/5 21171	7.25		S
3	2	1	1	Cumings, Mrs female		38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikinen, M female		26	0	0	STON/O2. 31	7.925		S
5	4	1	1	Futrelle, Mrs female		35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr. W male		35	0	0	373450	8.05		S
7	6	0	3	Moran, Mr. J male			0	0	330877	8.4583		Q

- Kualitas fitur dalam kumpulan data memiliki dampak besar pada kualitas wawasan yang akan diperoleh saat pemodelan machine learning.

FEATURES ENGINEERING

- Proses mengubah data mentah menjadi *feature* yang siap dipakai oleh model *Machine Learning*.

JENIS-JENIS FITUR

FITUR

KATEGORIS

NUMERIK

NOMINAL

- Pengelompokkan **tanpa tingkatan**
- Gender
 - Warna Rambut
 - Warna Mata
 - Dll

ORDINAL

- Pengelompokkan **dengan tingkatan**
- Jenjang Pendidikan
 - Kepuasan Pelanggan
 - Dll

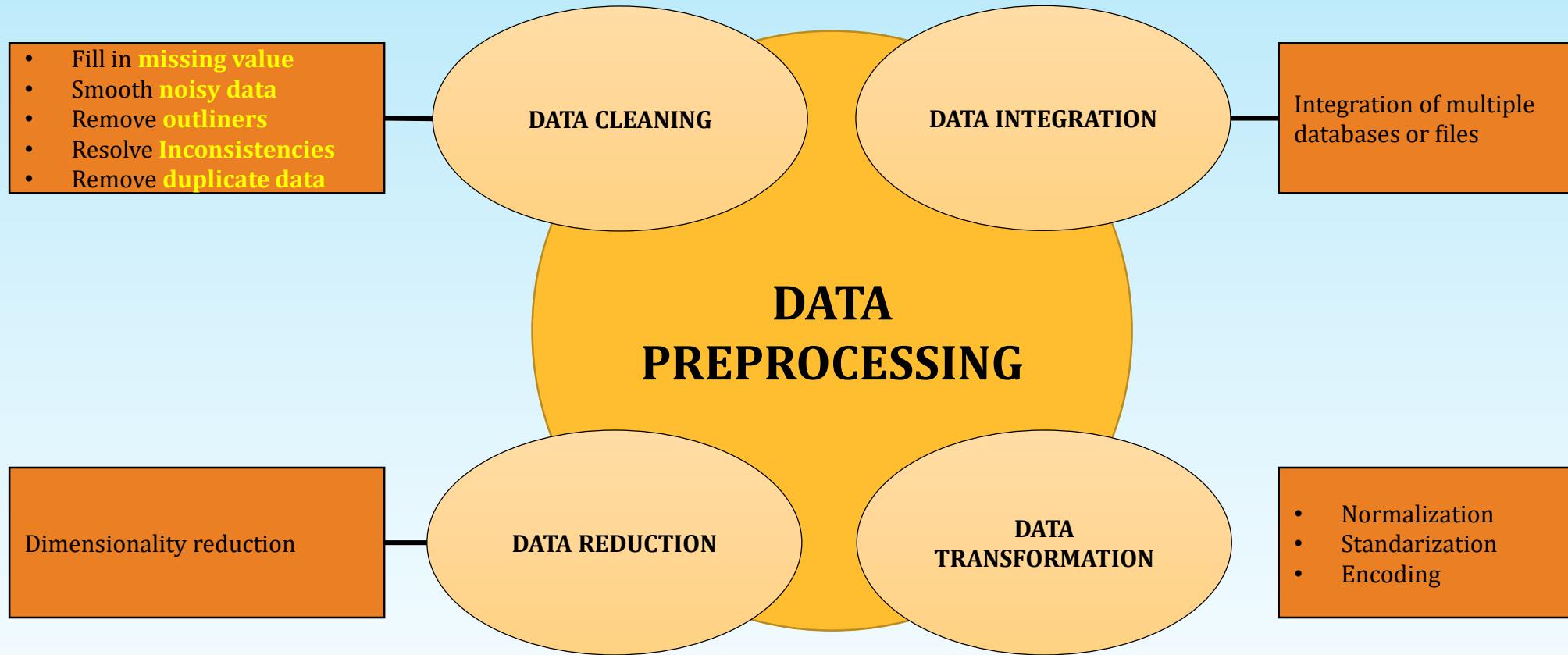
DICRETE

- Item yang dapat **dihitung**
- Jumlah Mahasiswa
 - Jumlah Kendaraan
 - Dll

CONTINUOUS

- Item yang dapat **diukur**
- Tinggi Badan
 - Suhu
 - Kecepatan
 - Dll

DATA PREPROCESSING MAIN JOB



DATA CLEANING - MISSING DATA

- Check Null Data

```
df.isnull()
# Returns a boolean matrix, if the value is NaN then True otherwise
False

df.isnull().sum()
# Returns the column names along with the number of NaN values in that
particular column
```

- Use imputer to handle missing value

df

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	0.0	11.0	12.0	NaN

```
from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values='NaN', strategy='mean', axis=0)
imputer = imputer.fit(df)
imputed_data = imputer.transform(df.values)
imputed_data
```

```
array([[ 1. ,  2. ,  3. ,  4. ],
       [ 5. ,  6. ,  7.5,  8. ],
       [ 0. , 11. , 12. ,  6. ]])
```

- Data memang tidak selalu tersedia
- Data yang hilang mungkin karena:
 - Kerusakan alat
 - Data tidak masuk karena keteledoran manusia
 - Data mungkin tidak dianggap penting saat input

Handle Missing Data

- Abaikan
- Isi Manual
- Isi Otomatis
 - Konstanta global (misal: "unknown")
 - Numerik: mean, median
 - Kategoris: modus

DATA CLEANING - NOISY DATA

- Noise adalah data yang berisi nilai-nilai yang salah atau anomali, yang biasanya disebut juga **outlier**.
- Nilai atribut yang salah mungkin karena:
 - Instrumen pengumpulan data yang salah
 - Terjadi masalah pada saat entri data
 - Terjadi masalah pada transmisi data

Handling Noisy Data

Binning

1. Urutkan Data
2. Partisi data ke dalam beberapa 'bin'
3. Lakukan smoothing:
 - bin-means
 - bin-medians
 - bin-boundaries

Regression → Mengganti outlier dengan fungsi regresi

Clustering → Mendeteksi & menghapus outlier

Combined computer and human inspection

Mendeteksi nilai yang mencurigakan dan diperiksa oleh manusia (misalnya, menangani kemungkinan outlier)

FEATURE ENCODING

One-Hot Encoding

- Mengubah setiap kategori sehingga memiliki nilai angka 1 atau angka 0

Color		Red	Yellow	Green
Red		1	0	0
Red		1	0	0
Yellow		0	1	0
Green		0	0	1
Yellow		0	0	1

Label Encoding

- Mengubah setiap kategori menjadi angka 1,2,3, ... dst

State (Nominal Scale)		State (Label Encoding)
Maharashtra		3
Tamil Nadu		4
Delhi		0
Karnataka		2
Gujarat		1
Uttar Pradesh		5

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X = LabelEncoder()
X[:, 0] = labelencoder_X.fit_transform(X[:, 0])

onehotencoder = OneHotEncoder(categorical_features = [0])
X = onehotencoder.fit_transform(X).toarray()

labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)
```

NORMALIZATION & STANDARDIZATION

Normalization

- Proses mengubah nilai-nilai suatu *feature* menjadi skala tertentu [0,1].

```
#scaling
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

X_train

```
array([[0.      , 1.      , 0.04498618, ..., 0.33333333, 0.15977676,
        1.      , ],
       [1.      , 1.      , 0.34656949, ..., 0.      , 0.01541158,
        1.      , ],
       [1.      , 0.      , 0.00728826, ..., 0.16666667, 0.02173075,
        1.      , ],
       ...,
       [1.      , 1.      , 0.50992712, ..., 0.      , 0.02753757,
        1.      , ],
       [0.      , 0.      , 0.17064589, ..., 0.33333333, 0.2342244 ,
        1.      , ],
       [0.      , 1.      , 0.25860769, ..., 0.16666667, 0.15085515,
        1.      , ]])
```

Standardization

- Proses mengubah nilai-nilai *feature* sehingga *mean* = 0 dan *standard deviation* = 1

```
#scaling
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

X_train

```
array([[ -1.63788124,  0.72077194, -1.91971935, ...,  1.99885349,
         0.98099823,  0.57000481],
       [ 0.80326712,  0.72077194, -0.0772525 , ..., -0.47932706,
        -0.46963364,  0.57000481],
       [ 0.80326712, -1.38740139, -2.15002771, ...,  0.75976322,
        -0.40613632,  0.57000481],
       ...,
       [ 0.80326712,  0.72077194,  0.92075038, ..., -0.47932706,
        -0.34778742,  0.57000481],
       [-1.63788124, -1.38740139, -1.15202483, ...,  1.99885349,
         1.72907416,  0.57000481],
       [-1.63788124,  0.72077194, -0.61463866, ...,  0.75976322,
         0.8913508 ,  0.57000481]])
```

WHEN TO USE NORMALIZATION & STANDARDIZATION

Normalization

- Ketika tidak tahu distribusi data atau bukan Gaussian.

TUJUAN

- Data dengan skala yang sama akan menjamin algoritma pembelajaran memperlakukan semua *feature* dengan adil

Standardization

- Ketika data memiliki dimensi variabel dan teknik yang digunakan (seperti regresi logistik, regresi linier, analisis diskriminan linier).

- Data dengan skala yang sama dan *centered* akan mempercepat algoritma pembelajaran
- Data dengan skala yang sama akan mempermudah interpretasi beberapa model ML

• Min-Max Scaling

Uses MinMaxScaler

Transform to defined range

$$y = \frac{x - \min x_i}{\max x_i - \min x_i}$$

Where

\bar{x} = mean

s = Standard deviation

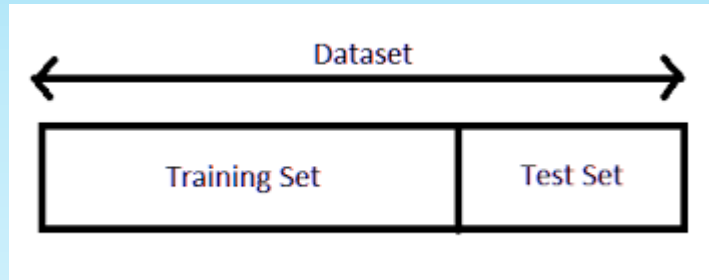
• Standardization

Uses StandardScaler

Transform to mean=0, sd=1

$$y = \frac{x - \bar{x}}{s}$$

TRAIN TEST SPLIT



- Training adalah proses ketika model mempelajari data
- Hasil dari training disebut model machine learning (trained model)
- Untuk membuktikan keakuratan model, diperlukan data uji (test data)
- Training set : subset untuk melatih model.
- Test set : subset untuk menguji model yang dilatih.
- Karena kurangnya data, kita bisa memisahkan dataset menjadi dua bagian yaitu training dan testing

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import MinMaxScaler

#hold out, dibagi menjadi training dan testing set
X_train, X_test, y_train, y_test = train_test_split(df_X, df_y, test_size=0.3, random_state=42)
```

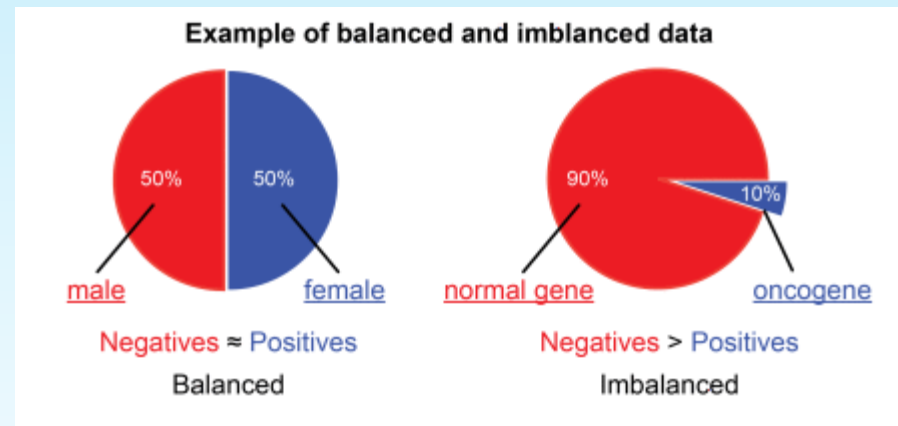
X_train

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
445	1	male	4.0	0	2	81.8583	S
650	3	male	28.0	0	0	7.8958	S
172	3	female	1.0	1	1	11.1333	S
450	2	male	36.0	1	2	27.7500	S
314	2	male	43.0	1	1	26.2500	S
...
106	3	female	21.0	0	0	7.6500	S
270	1	male	28.0	0	0	31.0000	S
860	3	male	41.0	2	0	14.1083	S
435	1	female	14.0	1	2	120.0000	S
102	1	male	21.0	0	1	77.2875	S

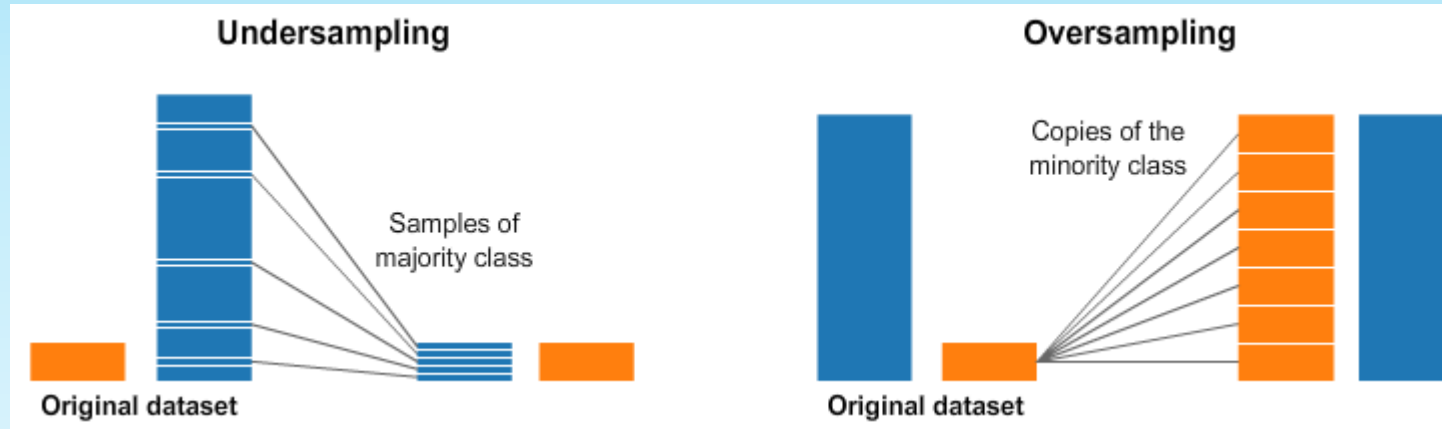
623 rows x 7 columns

IMBALANCED DATASET

Imbalanced data mengacu pada masalah klasifikasi di mana jumlah pengamatan per kelas tidak merata.



HANDLING IMBALANCED DATASET



Undersampling

- Menyeimbangkan distribusi kelas dengan menghilangkan contoh kelas mayoritas secara acak.

```
# import library
from imblearn.under_sampling import RandomUnderSampler

rus = RandomUnderSampler(random_state=42, replacement=True)# fit predictor and target variable
x_rus, y_rus = rus.fit_resample(x, y)

print('original dataset shape:', Counter(y))
print('Resample dataset shape', Counter(y_rus))
```

Oversampling

- Meningkatkan jumlah instance di kelas minoritas dengan mereplikasinya secara acak

```
# import library
from imblearn.over_sampling import RandomOverSampler

ros = RandomOverSampler(random_state=42)

# fit predictor and target variable
x_ros, y_ros = ros.fit_resample(x, y)

print('Original dataset shape', Counter(y))
print('Resample dataset shape', Counter(y_ros))
```



THANK YOU !

