



# Introduction





# Tujuan Materi Hari Ini

1. Memahami langkah-langkah dasar dalam melakukan *data preprocessing* yang dilakukan diantara proses *importing data* dan *data cleansing*, serta pentingnya *preprocess* dalam proyek *machine learning*
2. Menangani *missing values*, *data encoding*, *scaling*, dan *splitting*
3. Mendapatkan pengalaman praktis melalui contoh dan latihan.



# Apa itu *Data Preprocessing*?

- Dilakukan setelah *Exploratory Data Analysis* (EDA) dan *data cleaning*
- Mempersiapkan data untuk pemodelan
- **Contoh:** mentransformasi fitur kategorik menjadi fitur numerik (*dummy variable*)



# Mengapa *Preprocess* Dilakukan?

- Mengubah data menjadi sesuai untuk dimodelkan
- Meningkatkan performa model
- Memberikan hasil yang *reliable*





# Missing Values Handling



# Mengapa Ada Missing Values ?

- Faktanya, data di dunia nyata seringkali ditemukan dalam bentuk yang tidak rapi/kacau/berantakan/tidak teratur
  - **Did you know that 72% of organizations believe that data quality issues hinder customer trust and perception? (\*)**

\*[Top 9 Benefits of Data Cleansing for Businesses](<https://bit.ly/2QwMrab>)



# Mengapa Ada Missing Values ?

- **Values yang hilang dalam proses akuisisi data**
  - Sensor cuaca rusak selama analisis cuaca
  - Informasi pasien yang tidak lengkap untuk diagnosis medis dll.
- **Values terhapus secara tidak sengaja**
  - Data yang hilang
  - Tidak sengaja terhapus akibat *human error*



# Workflow Penanganan Missing Values

1. Ubah semua *missing values* menjadi “*null*” values.
2. Menganalisis jumlah dan jenis data dari *missing values*.
3. *Delete* atau *impute missing values* dengan cara yang tepat.
4. Evaluasi & bandingkan performa dari data yang telah ditangani/imputasi tadi saat di-inputkan pada model





# Missing Values



- Biasanya berupa dengan nilai seperti berikut:

'NA', '-' atau '.', dll



# Detect and Replace Missing Values

- Dataset Cases:
  - college.csv
  - pima-indian-diabetes.csv

Hands-on Coding in Google Colab



# Apa itu Data Imputation?

- proses mengisi atau menggantikan nilai yang hilang atau tidak lengkap dalam sebuah dataset mean
- Bertujuan untuk mempertahankan integritas dan kualitas dataset
- Menghindari kehilangan informasi yang berpotensi penting



# Imputasi: Teknik Dasar

- constant (contoh: '0')
- mean
- median
- mode or most frequent



# Implementasi & Visualisasi

- Hands-on Coding in Google Colab



# Label Encoder



# Implementasi



- Hands-on Coding in Google Colab



# One-Hot Encoder





# One-hot encoding

fav_color
blue
green
orange
green

fav_color_enc
[1, 0, 0]
[0, 1, 0]
[0, 0, 1]
[0, 1, 0]

Values: [blue, green, orange]

- blue : [1, 0, 0]
- green : [0, 1, 0]
- orange : [0, 0, 1]



# Implementasi



- Hands-on Coding in Google Colab



# Standardization



# Apa itu Standardization?

**Standardization:** mengubah data kontinu agar menyerupai distribusi normal

- Model-model *scikit-learn* mengasumsikan data yang diinputkan terdistribusi secara normal
- Menggunakan *data training* yang tidak terdistribusi secara normal dapat mengakibatkan error (*bias*)
- ***Log normalization*** dan ***feature scaling*** yang akan kita terapkan
- Diterapkan pada data yang bersifat numerik kontinu



# Kapan harus Standardization?

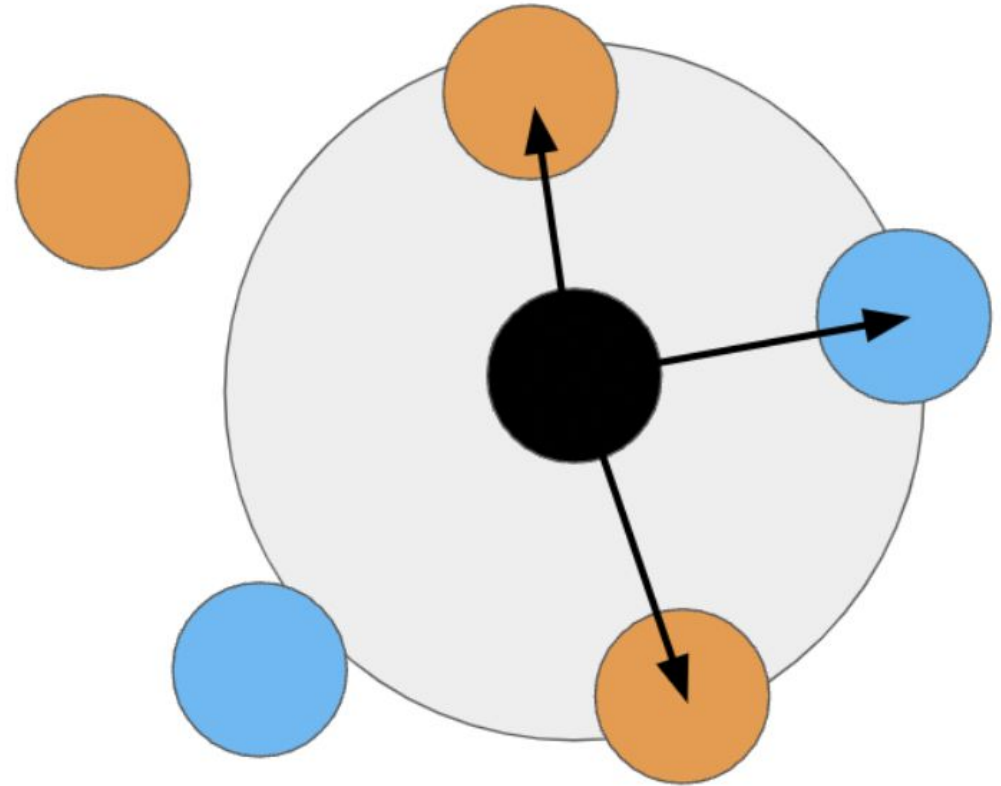
## 1. Model pada ruang linier (*linear space*)

### Contoh:

- k-Nearest Neighbors (kNN)
- Linear regression
- K-Means Clustering

## 2. Dataset yang memiliki fitur-fitur bersifat *high variance*

Hal ini dapat membiaskan model yang mengasumsikan data terdistribusi secara normal.

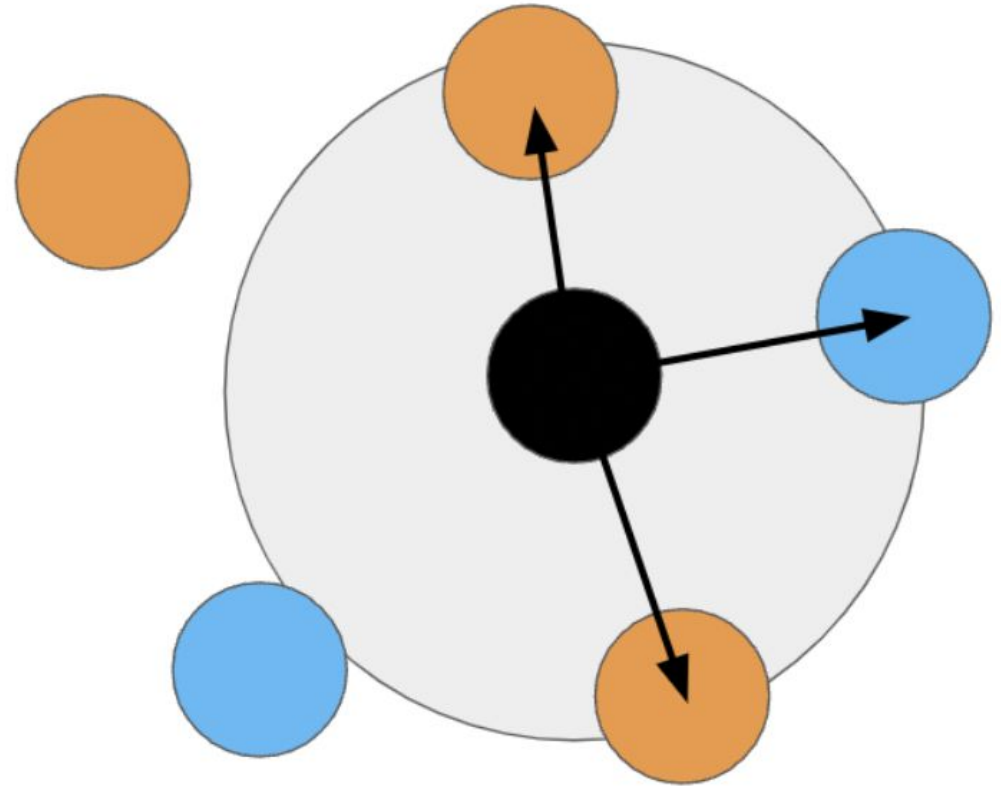




# Kapan harus Standardization?

3. **Fitur-fitur yang memiliki skala yang berbeda satu sama lainnya**

4. **Asumsi Linearitas**





# Apa itu *log normalization*?

- Berguna untuk fitur dengan variasi tinggi
- Menerapkan transformasi logaritma
- Log natural menggunakan konstanta  $e$  ( $\approx 2.718$ )
- $e^{3.4} = 30$

Number	Log
30	3.4
300	5.7
3000	8

- Menangkap perubahan relatif, besarnya perubahan, dan menjaga semuanya tetap positif



# Log normalization in Python

```
print(df)
```

	col1	col2
0	1.00	3.0
1	1.20	45.5
2	0.75	28.0
3	1.60	100.0

```
print(df.var())
```

col1	0.128958
col2	1691.729167
dtype:	float64

```
import numpy as np  
df["log_2"] = np.log(df["col2"])  
print(df)
```

	col1	col2	log_2
0	1.00	3.0	1.098612
1	1.20	45.5	3.817712
2	0.75	28.0	3.332205
3	1.60	100.0	4.605170

```
print(df[["col1", "log_2"]].var())
```

col1	0.128958
log_2	2.262886
dtype:	float64





# Apa itu *feature scaling*?

- Fitur-fitur pada skala yang berbeda
- Model dengan karakteristik linier
- Memusatkan nilai fitur di sekitar 0 dan mengubah menjadi ber-variants 1
- Mentransformasi agar mendekati distribusi normal



# How to scale data?

```
print(df)
```

```
   col1  col2  col3
0  1.00  48.0  100.0
1  1.20  45.5  101.3
2  0.75  46.2  103.5
3  1.60  50.0  104.0
```

```
print(df.var())
```

```
col1    0.128958
col2    4.055833
col3    3.526667
dtype: float64
```



# How to scale data?

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df),
                        columns=df.columns)
```

```
print(df_scaled)
```

	col1	col2	col3
0	-0.442127	0.329683	-1.352726
1	0.200967	-1.103723	-0.553388
2	-1.245995	-0.702369	0.799338
3	1.487156	1.476409	1.106776

```
print(df_scaled.var())
```

col1	1.333333
col2	1.333333
col3	1.333333
dtype:	float64



# Train-Test Split Dataset



# Apa itu Train-Test Splitting?

- Train-Test Splitting adalah salah satu teknik yang umum digunakan dalam *machine learning* untuk menguji kinerja model pada data yang belum pernah dilihat sebelumnya
- Bertujuan untuk memisahkan dataset menjadi train set dan subset pengujian test set.



# Manfaat Train-Test Splitting

- Mengukur kinerja model secara objektif
- Mencegah overfitting
- Generalisasi ML Model
- Mendeteksi masalah dan debugging
- Pemilihan model yang lebih baik



# Implementasi



- Hands-on Coding in Google Colab

**Thank  
YOU**

