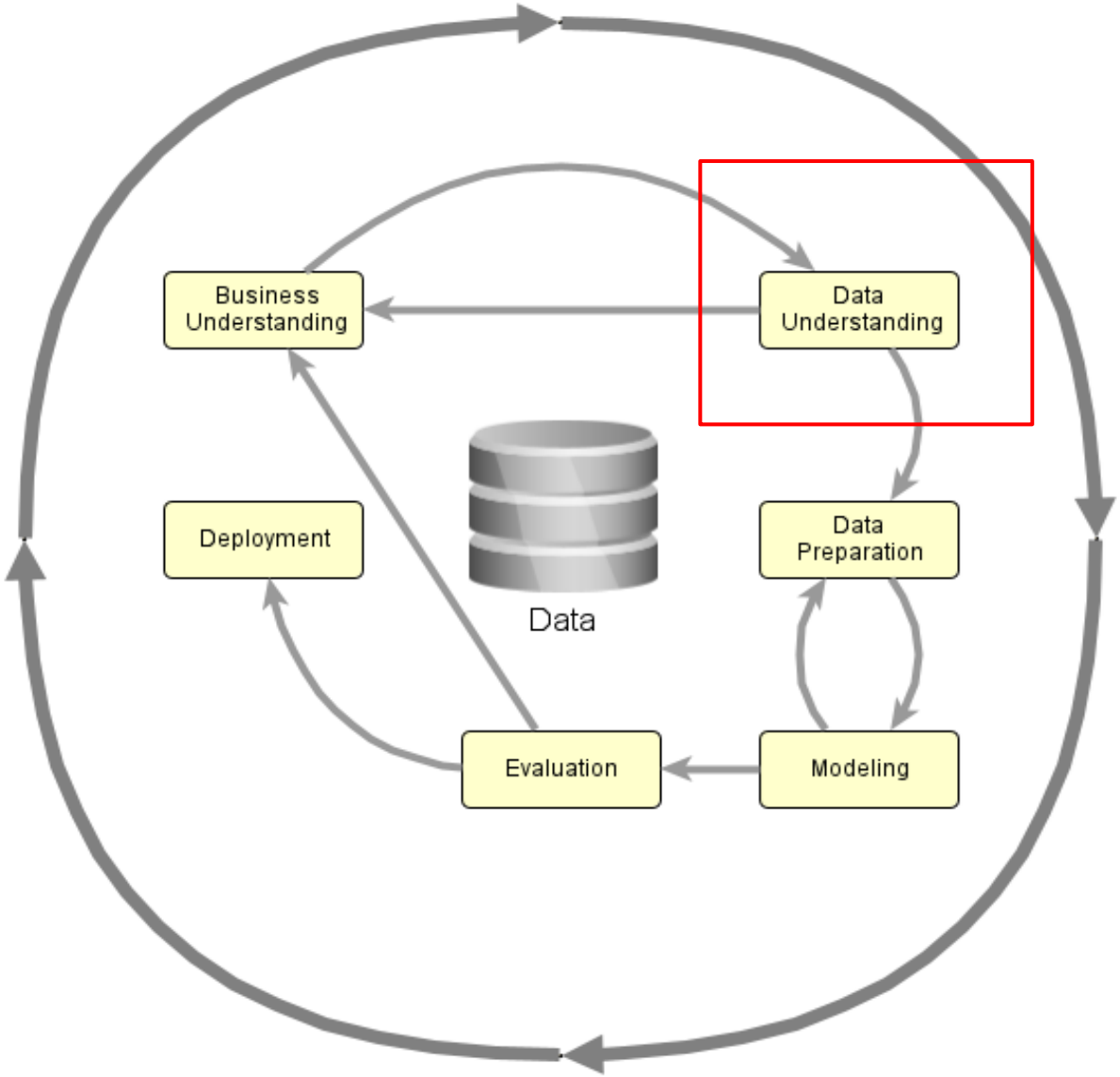
Pemahaman Data
(Data Understanding)

IF3170 Inteligensi Artifisial

Materi disiapkan oleh: Dessi Puji Lestari

KK Informatika / Pusat AI ITB

Metodologi Data Science



Data Understanding

Mengumpulkan dan **memahami** data yang akan digunakan

Mengapa Dibutuhkan Data Understanding?

- Data = bahan mentah solusi AI
- **Data sangat menentukan kualitas model**
- Data dari berbagai sumber belum tentu dapat langsung dipakai:
  - tujuan data berbeda-beda
  - tingkat kekayaan (*richness*) berbeda-beda
  - tingkat kepercayaan data dari berbagai sumber bisa jadi berbeda

Mengapa Dibutuhkan Data Understanding?

Data understanding memberikan gambaran awal tentang:

- tingkat kesesuaian data dengan masalah bisnis yang akan dipecahkan
- kelebihan data
- kekurangan dan batasan penggunaan data
- ketersediaan data (terbuka/tertutup, biaya akses, dsb.)
- langkah-langkah yang perlu dilakukan untuk meningkatkan kualitas dan kuantitas data (kualitas model)

Tujuan Data Understanding

- Mendapatkan wawasan awal dari data
- Memahami konten data
- Menilai kualitas data

Aktivitas *Data Understanding*

1. Pengumpulan Data Awal
2. Deskripsi Data
3. Eksplorasi Data
4. Verifikasi Kualitas Data

1.Pengumpulan Data

Aktivitas Pengumpulan Data

1. Identifikasi sumber data
2. Mengakses dan mengumpulkan data
3. Menentukan format data
4. Mengecek kelengkapan data
5. Mencatat metadata dan dokumentasi
6. Pembersihan awal data
7. Menyimpan data dengan aman

Identifikasi Sumber Data

Menentukan sumber data yang relevan

Sumber Internal	Spreadsheets (Excel, CSV, JSON, etc.)
	Databases: can be queried via SQL, etc.
	Text documents
	Multimedia documents (audio, video)
	Sensor IoT
Sumber Eksternal	Open data repositories
	Public domain web pages

Sumber Data Eksternal

- Portal Satu Data Indonesia (https://data.go.id)
- Portal Data Jakarta (https://satudata.jakarta.go.id)
- Portal Data Bandung (http://data.bandung.go.id)
- Badan Pusat Statistik (https://www.bps.go.id)
- Badan Informasi Geospasial (https://tanahair.indonesia.go.id/)
- UCI Machine Learning repository (https://archive.ics.uci.edu/ml/index.php)
- Kaggle (https://www.kaggle.com/datasets)
- World Bank Open Data (https://data.worldbank.org)
- UNICEF Data (https://data.unicef.org)
- WHO Open Data (https://www.who.int/data)
- IBM Data Asset eXchange (https://developer.ibm.com/exchanges/data/)
- DBPedia (https://www.dbpedia.org/resources/)
- Wikidata (https://www.wikidata.org/)
- Google Dataset Search: https://datasetsearch.research.google.com

Mengakses dan Mengumpulkan Data

Bisa dilakukan dengan berbagai cara, seperti:

- mengekstrak data dari *database*
- mengunduh file data
- menggunakan skrip otomatisasi untuk mengumpulkan data dari web atau API
- mengumpulkan sendiri, misal melakukan perekaman gambar, suara

Pastikan legalitas dari data dan data dikumpulkan secara etis (copyright, confidentiality, security).

Menentukan Format Data

- Memastikan data yang dikumpulkan dalam format yang sesuai, misal:
  - Format tabel (csv)
  - JSON
  - XML
  - wav, jpg, dll

Mengecek Kelengkapan Data

Memastikan bahwa semua data yang dibutuhkan telah dikumpulkan.

- Jika ada data yang hilang atau tidak lengkap, langkah-langkah tambahan mungkin perlu diambil, seperti melakukan survei tambahan, pengumpulan manual tambahan, atau menghubungi sumber data lain.

Proses pembangunan model seringkali dilakukan secara iteratif, pengumpulan data tambahan bisa saja dilakukan setelah pembangunan model pertama selesai dan evaluasi model mengindikasikan data yang digunakan masih kurang.

Mencatat Metadata dan Dokumentasi

- Selama pengumpulan data, penting untuk mencatat metadata, seperti:
  - Kapan data dikumpulkan
  - Dari mana data berasal
  - Parameter atau kondisi apa yang digunakan saat pengumpulan.
- Dokumentasi ini penting untuk referensi di masa depan dan untuk memastikan integritas data.

Menyimpan Data dengan Aman

- Data yang dikumpulkan perlu disimpan dengan aman dan diatur dengan baik untuk memfasilitasi akses dan penggunaan di tahap berikutnya.
- Buat SOP penyimpanan dan pengaksesan data

2. Deskripsi Data

Deskripsi Data

Memberikan gambaran umum tentang data, sehingga analis dapat mulai mengenali pola, anomali, dan struktur data yang mungkin memengaruhi analisis lebih lanjut.

1. Memahami Struktur Data
2. Memeriksa Nilai-Nilai Ringkasan Statistik (*Summary Statistics*)

1.Memahami Struktur Data

1. **Identifikasi Atribut dan Tipe Data:**
    a. Identifikasi setiap atribut dalam dataset
    b. Identifikasi tipe data masing-masing atribut
        - Numerik (integer, float)
        - kategorikal (nominal, ordinal)
        - teks, gambar, suara, atau jenis lainnya.
2. **Memeriksa Dimensi Data:**
    a. Menentukan jumlah baris (record) dalam dataset
    b. Menentukan jumlah kolom (atribut) dalam dataset.

Contoh Deskripsi Data

Automobile Data Set

Download: Data Folder, Data Set Description

Abstract: From 1985 Ward's Automotive Yearbook

Data Set Characteristics:	Multivariate	**Number of Instances:**	205
Attribute Characteristics:	Categorical, Integer, Real	**Number of Attributes:**	26
Associated Tasks:	Regression	**Missing Values?**	Yes

205 data objects = 205 rows

26 columns

Predict price based on 25 attributes of automobile data

http://archive.ics.uci.edu/ml/datasets/Automobile

Automobile Dataset: Attributes (categorical, integer, real)

Attribute: Attribute Range:
1. symboling: -3, -2, -1, 0, 1, 2, 3.
2. normalized-losses:continuous from 65 to 256.
3. make:alfa-romero, audi, bmw, chevrolet, dodge, honda, isuzu, jaguar, mazda, mercedes-benz, mercury, mitsubishi, nissan, peugot, plymouth, porsche, renault, saab, subaru, toyota, volkswagen, volvo
4. fuel-type: diesel, gas.
5. aspiration: std, turbo.
6. num-of-doors: four, two.
7. body-style: hardtop, wagon, sedan, hatchback, convertible.
8. drive-wheels: 4wd, fwd, rwd.
9. engine-location: front, rear.
10. wheel-base: continuous from 86.6 120.9.
11. length: continuous from 141.1 to 208.1.
12. width: continuous from 60.3 to 72.3.
13. height:continuous from 47.8 to 59.8.
14. curb-weight:continuous from 1488 to 4066.
15. engine-type:dohc, dohcv, l, ohc, ohcf, ohcv, rotor.
16. num-of-cylinders:eight, five, four, six, three, twelve, two.
17. engine-size:continuous from 61 to 326.
18. fuel-system: 1bbl, 2bbl, 4bbl, idi, mfi, mpfi, spdi, spfi.
19. bore:continuous from 2.54 to 3.94.
20. stroke:continuous from 2.07 to 4.17.
21. compression-ratio: continuous from 7 to 23.
22. horsepower:continuous from 48 to 288.
23. Peak-rpm: continuous from 4150 to 6600.
24. city-mpg: continuous from 13 to 49.
25. highway-mpg:continuous from 16 to 54.
26. Price: continuous from 5118 to 45400.

Implementasi Python: Persiapan

```
# Day 2 - Sesi 1 - Data Understanding Slide 23
#Mengimpor pustaka yang relevan
#Sebelum memuat dataset, kita perlu terlebih dahulu mengimpor semua pustaka yang relevan.
#numpy (Numerical Python) adalah library fundamental yang digunakan untuk komputasi.
#pandas adalah library untuk analisis dan manipulasi data, terutama untuk data berbentuk tabel (data terstruktur).
#matplotlib adalah library yang digunakan untuk membuat visualisasi data dalam berbagai bentuk grafik dan plot.
#seaborn adalah library visualisasi data yang dibangun di atas Matplotlib dan dirancang untuk membuat grafik statistik yang lebih menarik
dan informatif.
#io adalah library input/output yang berhubungan dengan membaca (input) dan menulis (output) data.
#scypi.stats adalah modul dalam library SciPy yang menyediakan fungsi-fungsi untuk statistik, distribusi probabilitas, dan uji hipotesis.

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
import io
import scipy.stats as stats

from google.colab import files, data_table
from scipy.stats import shapiro, kstest, normaltest, chi2_contingency, skew, kurtosis
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import MinMaxScaler, StandardScaler, OneHotEncoder
from sklearn.model_selection import StratifiedShuffleSplit
```

Implementasi Python:
Identifikasi Struktur Data

```
# Membaca dataset - data lengkap
df = pd.read_csv('ai4i2020.csv')
# Menampilan dataset dalam bentuk tabel
data_table.DataTable(df)
```

1 to 25 of 10000 entries

index	UDI	Product ID	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Machine failure	TWF	HDF	PWF
0	1	M14860	M	298.1	308.6	1551	42.8	0	0	0	0	0
1	2	L47181	L	298.2	308.7	1408	46.3	3	0	0	0	0
2	3	L47182	L	298.1	308.5	1498	49.4	5	0	0	0	0
3	4	L47183	L	298.2	308.6	1433	39.5	7	0	0	0	0
4	5	L47184	L	298.2	308.7	1408	40.0	9	0	0	0	0
5	6	M14865	M	298.1	308.6	1425	41.9	11	0	0	0	0
6	7	L47186	L	298.1	308.6	1558	42.4	14	0	0	0	0
7	8	L47187	L	298.1	308.6	1527	40.2	16	0	0	0	0
8	9	M14868	M	298.3	308.7	1667	28.6	18	0	0	0	0
9	10	M14869	M	298.5	309.0	1741	28.0	21	0	0	0	0
10	11	H29424	H	298.4	308.9	1782	23.9	24	0	0	0	0
11	12	H29425	H	298.6	309.1	1423	44.3	29	0	0	0	0

```
# Memeriksa dimensi data
print(f"Jumlah baris: {df.shape[0]}, Jumlah kolom: {df.shape[1]}")
# Menampilkan nama kolom dan tipe data
print(df.dtypes)
```

```
Jumlah baris: 10000, Jumlah kolom: 14
UDI                        int64
Product ID                object
Type                      object
Air temperature [K]      float64
Process temperature [K]  float64
Rotational speed [rpm]     int64
Torque [Nm]              float64
Tool wear [min]            int64
Machine failure            int64
TWF                        int64
HDF                        int64
PWF                        int64
OSF                        int64
RNF                        int64
dtype: object
```

2. Memeriksa Nilai Ringkasan Statistik

- **Statistik Deskriptif:**
  - Menghitung statistik dasar seperti rata-rata (mean), median, modus, minimum, maksimum, kuartil, dan standar deviasi untuk atribut numerik.
  - Untuk memberikan wawasan awal tentang distribusi dan skala data.
- **Distribusi Frekuensi:**
  - Untuk data kategorikal
  - Distribusi frekuensi dihitung untuk mengetahui seberapa sering kategori tertentu muncul dalam dataset.

Implementasi Python: Statistik Deskriptif

```
# Statistik deskriptif untuk variabel numerik
print(df.describe())
```

```
                UDI  Air temperature [K]  Process temperature [K]  \
count  10000.00000         10000.000000             10000.000000
mean    5000.50000           300.004930               310.005560
std     2886.89568             2.000259                 1.483734
min        1.00000           295.300000               305.700000
25%     2500.75000           298.300000               308.800000
50%     5000.50000           300.100000               310.100000
75%     7500.25000           301.500000               311.100000
max    10000.00000           304.500000               313.800000

       Rotational speed [rpm]   Torque [Nm]  Tool wear [min]  Machine failure  \
count            10000.000000  10000.000000     10000.000000     10000.000000
mean              1538.776100     39.986910       107.951000         0.033900
std                179.284096      9.968934        63.654147         0.180981
min               1168.000000      3.800000         0.000000         0.000000
25%               1423.000000     33.200000        53.000000         0.000000
50%               1503.000000     40.100000       108.000000         0.000000
75%               1612.000000     46.800000       162.000000         0.000000
max               2886.000000     76.600000       253.000000         1.000000

                TWF           HDF           PWF           OSF           RNF
count  10000.000000  10000.000000  10000.000000  10000.000000  10000.00000
mean       0.004600      0.011500      0.009500      0.009800      0.00190
std        0.067671      0.106625      0.097009      0.098514      0.04355
min        0.000000      0.000000      0.000000      0.000000      0.00000
25%        0.000000      0.000000      0.000000      0.000000      0.00000
50%        0.000000      0.000000      0.000000      0.000000      0.00000
75%        0.000000      0.000000      0.000000      0.000000      0.00000
max        1.000000      1.000000      1.000000      1.000000      1.00000
```

Implementasi Python:
Distribusi Frekuensi

```
# Distribusi frekuensi untuk
variabel kategorikal
for col in
df.select_dtypes(include='object').
columns:
 print(df[col].value_counts())
```

```
Product ID
M14860    1
L53850    1
L53843    1
L53844    1
L53845    1
         ..
M18193    1
M18194    1
L50515    1
L50516    1
M24859    1
Name: count, Length: 10000, dtype: int64
Type
L    6000
M    2997
H    1003
Name: count, dtype: int64
```

3. Eksplorasi Data

Tujuan Eksplorasi Data

- Untuk memahami struktur, mengidentifikasi pola, tren, korelasi dalam data
- Digunakan untuk memberikan wawasan tentang bentuk data dan permasalahan data serta solusi yang perlu dilakukan di tahap berikutnya (Persiapan Data)

Aktivitas dalam Eksplorasi Data

1. Analisis Statistik Deskriptif Lanjutan (Skewness, Kurtosis, dll)
2. Identifikasi Pola dan Tren
3. Analisis Hubungan antar Variabel
4. Analisis Kategorisasi dan Grup
5. Eksplorasi Waktu dan Sekuensial
6. Dokumentasi dan Interpretasi

1. Analisis Statistik Deskriptif Lanjutan

- Menghitung skewness (kemencengan)
- Menghitung Kurtosis (keruncingan)
- Melakukan uji normalitas
- Menganalisis distribusi data untuk setiap variabel, seperti distribusi normal, distribusi uniform, atau distribusi skewed.

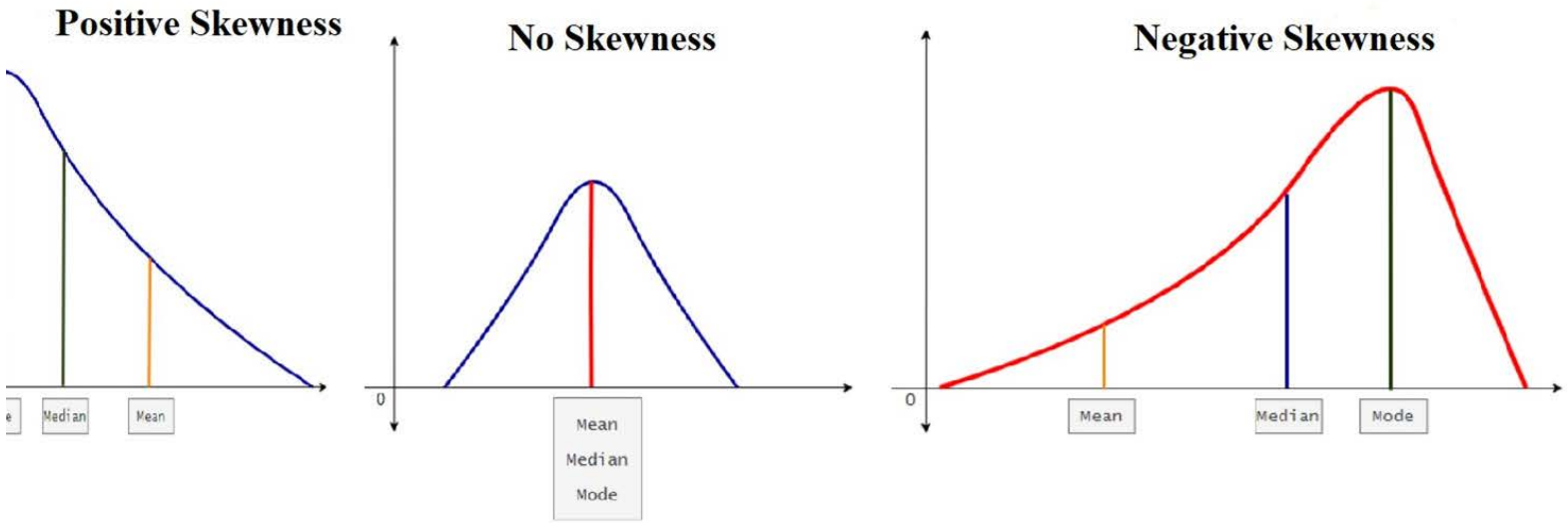
Skewness

- **Mengukur kemencengan data.**

  Nilai skewness positif menunjukkan distribusi yang skew ke kanan (tail lebih panjang ke kanan)

  Nilai skewness negatif menunjukkan skew ke kiri

  Nilai skewness ≈ 0: Distribusi data simetris atau mendekati simetris



Kurtosis

- **Kurtosis mengukur keruncingan atau "puncak" distribusi.**
  - Nilai kurtosis menunjukkan seberapa besar data memiliki puncak yang tinggi dan tajam atau datar dibandingkan dengan distribusi normal.
    - Jika kurtosis > 3 (leptokurtik): Distribusi memiliki puncak yang lebih tajam dan lebih tinggi dari distribusi normal. Hal ini menunjukkan adanya outliers yang signifikan.
    - Jika kurtosis < 3 (platykurtik): Distribusi lebih datar dan lebih tersebar dibandingkan distribusi normal.
    - Jika kurtosis ≈ 3 (mesokurtik): Distribusi mendekati distribusi normal.

Implementasi Python:
Menghitung Kemencengan dan Keruncingan

```
# Day 2 - Sesi 1 - Data Understanding Slide 34-35
# Data PredMain
# Menghitung skewness (kemencengan)
skewness = df['Air temperature [K]'].skew()
print(f"Skewness (Kemencengan) Air temperature [K]: {skewness}")

# Menghitung kurtosis (keruncingan)
kurt = df['Air temperature [K]'].kurtosis()
print(f"Kurtosis (Keruncingan) Air temperature [K]: {kurt}")
```

```
Skewness (Kemencengan) Air temperature [K]: 0.11427392052155042
Kurtosis (Keruncingan) Air temperature [K]: -0.8359616725774668
```

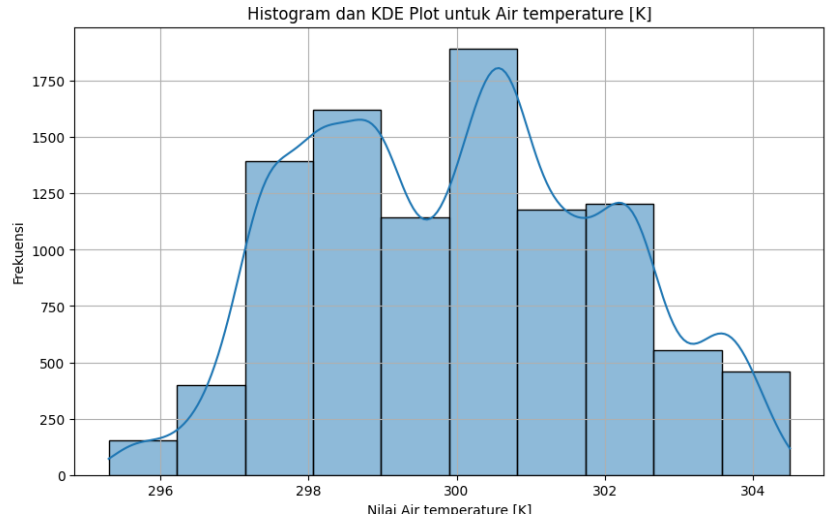
```
# Contoh DataFrame
data = {'var1': [12, 15, 14, 10, 50, 16, 14,
20, 13, 75, 18, 13, 14, 19, 13, 16, 14, 20,
13,12, 15, 14, 10]}
df = pd.DataFrame(data)
# Menghitung skewness (kemencengan)
skewness = df['var1'].skew()
print(f"Skewness (Kemencengan) var1:
{skewness}")
# Menghitung kurtosis (keruncingan)
kurt = df['var1'].kurtosis()
print(f"Kurtosis (Keruncingan) var1: {kurt}")
```

```
Skewness (Kemencengan) var1: 3.3064887615223464
Kurtosis (Keruncingan) var1: 11.12741883840204
```

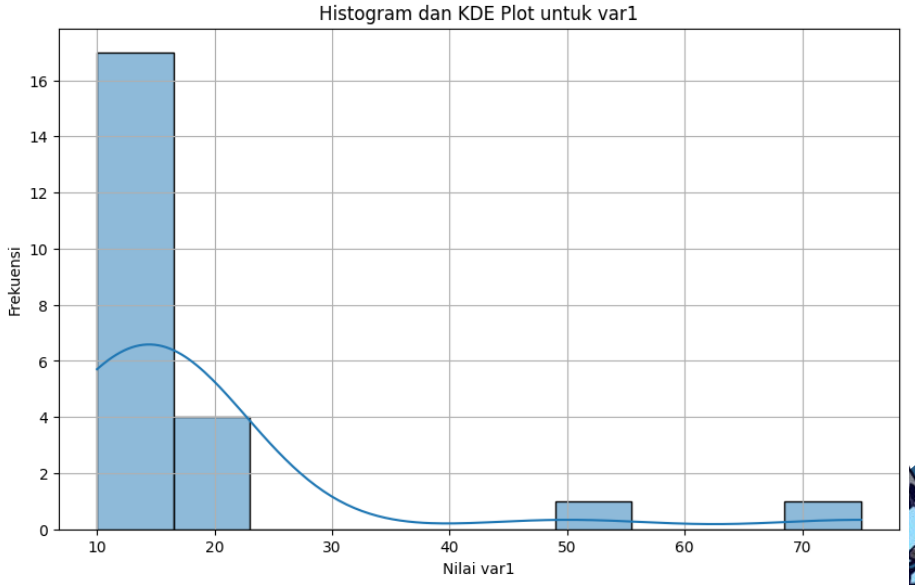
Center for
Artificial Intelligence
ITB

Implementasi Python:
Visualisasi Kemencengan dan Keruncingan

```
# Visualisasi Histogram dan KDE (Kernel Density Estimation)
plt.figure(figsize=(10, 6))
sn.histplot(df['Air temperature [K]'], kde=True, bins=10)
plt.title('Histogram dan KDE Plot untuk Air temperature [K]')
plt.xlabel('Nilai Air temperature [K]')
plt.ylabel('Frekuensi')
plt.grid(True)
plt.show()
```




```
# Visualisasi Histogram dan KDE (Kernel Density Estimation)
plt.figure(figsize=(10, 6))
sn.histplot(df['var1'], kde=True, bins=10)
plt.title('Histogram dan KDE Plot untuk var1')
plt.xlabel('Nilai var1')
plt.ylabel('Frekuensi')
plt.grid(True)
plt.show()
```



Uji Normalitas

- Langkah krusial untuk memastikan analisis data akurat, model valid, dan hasil dapat diandalkan dalam pengambilan keputusan.
  - Banyak metode statistik klasik (misal: regresi linier, ANOVA) mengasumsikan data berdistribusi normal

    ⇨ Uji normalitas memastikan validitas hasil dengan memeriksa asumsi ini.
  - Jika data tidak normal, transformasi data mungkin diperlukan.

Cara Menguji Normalitas

- **Shapiro-Wilk Test:**

  Nilai *p-value* di bawah 0.05 menandakan bahwa distribusi tidak normal.

- **Kolmogorov-Smirnov Test:**
  - Nilai statistik sebesar 1.0 menunjukkan perbedaan maksimal antara distribusi data dan distribusi normal yang diharapkan.
  - Nilai sangat mendekati nol menunjukkan data tidak berdistribusi normal.

- **D'Agostino and Pearson's Test:**
  - Uji gabungan untuk skewness dan kurtosis yang menguji normalitas.
  - Nilai statistik yang tinggi menunjukkan adanya deviasi yang signifikan dari distribusi normal baik dalam hal skewness maupun kurtosis.
  - p-value yang sangat kecil (di bawah 0.05) mengindikasikan data tidak normal.

Implementasi Python:
Uji Normalitas

```
# Uji normalitas menggunakan Shapiro-Wilk Test
shapiro_stat, shapiro_p_value = shapiro(df1['var1'])
print(f"Shapiro-Wilk Test: Statistic={shapiro_stat}, p-value={shapiro_p_value}")
# Uji normalitas menggunakan Kolmogorov-Smirnov Test
ks_stat, ks_p_value = kstest(df1['var1'], 'norm')
print(f"Kolmogorov-Smirnov Test: Statistic={ks_stat}, p-value={ks_p_value}")
# Uji normalitas menggunakan D'Agostino and Pearson's Test
dagostino_stat, dagostino_p_value = normaltest(df1['var1'])
print(f"D'Agostino and Pearson's Test: Statistic={dagostino_stat}, p-
value={dagostino_p_value}")
```

```
Shapiro-Wilk Test: Statistic=0.5009031103608638, p-value=8.743038680400318e-08
Kolmogorov-Smirnov Test: Statistic=1.0, p-value=0.0
D'Agostino and Pearson's Test: Statistic=38.862947231913644, p-value=3.639303149912843e-09
```

Data tidak berdistribusi normal

Eksplorasi Distribusi untuk Setiap Variabel

- **Histogram dan Density Plot:**

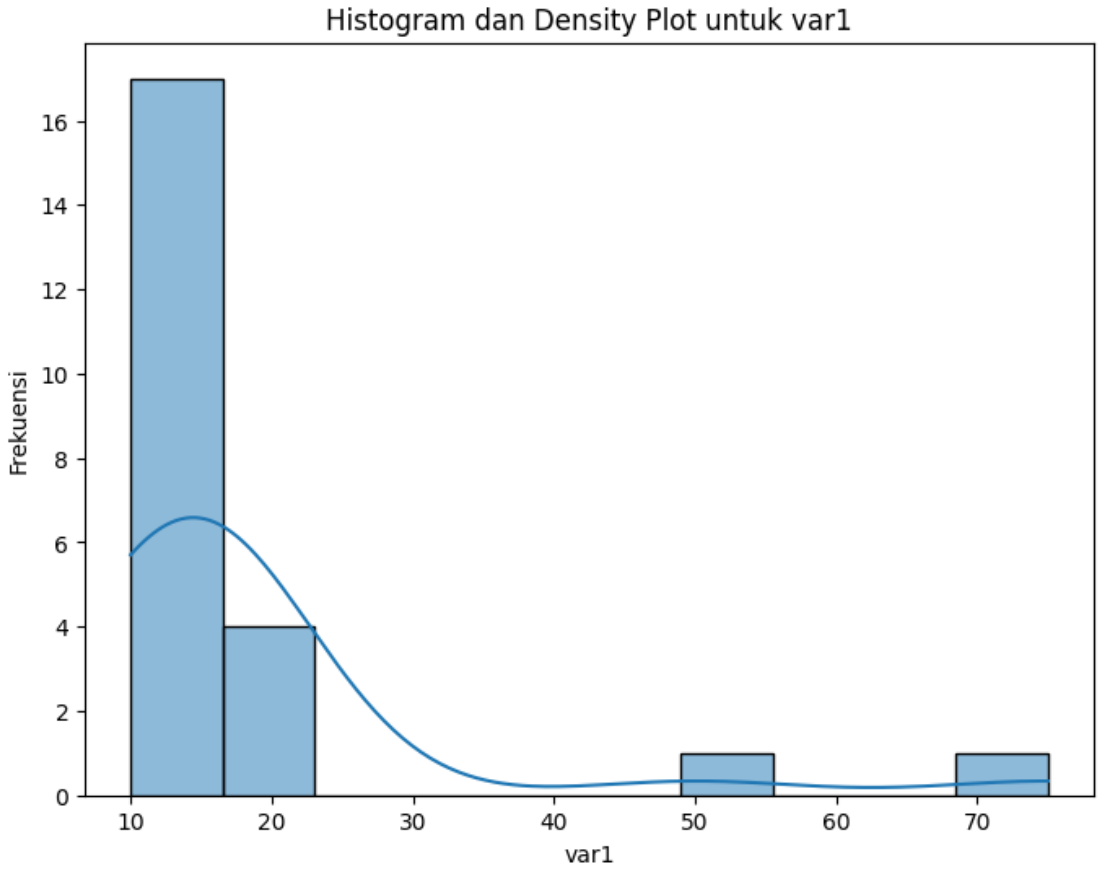
  Menunjukkan distribusi data secara visual dan overlay dengan *density plot* untuk melihat bentuk distribusi.

- **Q-Q Plot:**

  - Memeriksa kesesuaian distribusi data dengan distribusi normal.
  - Jika titik-titik mengikuti garis diagonal, data mengikuti distribusi normal.

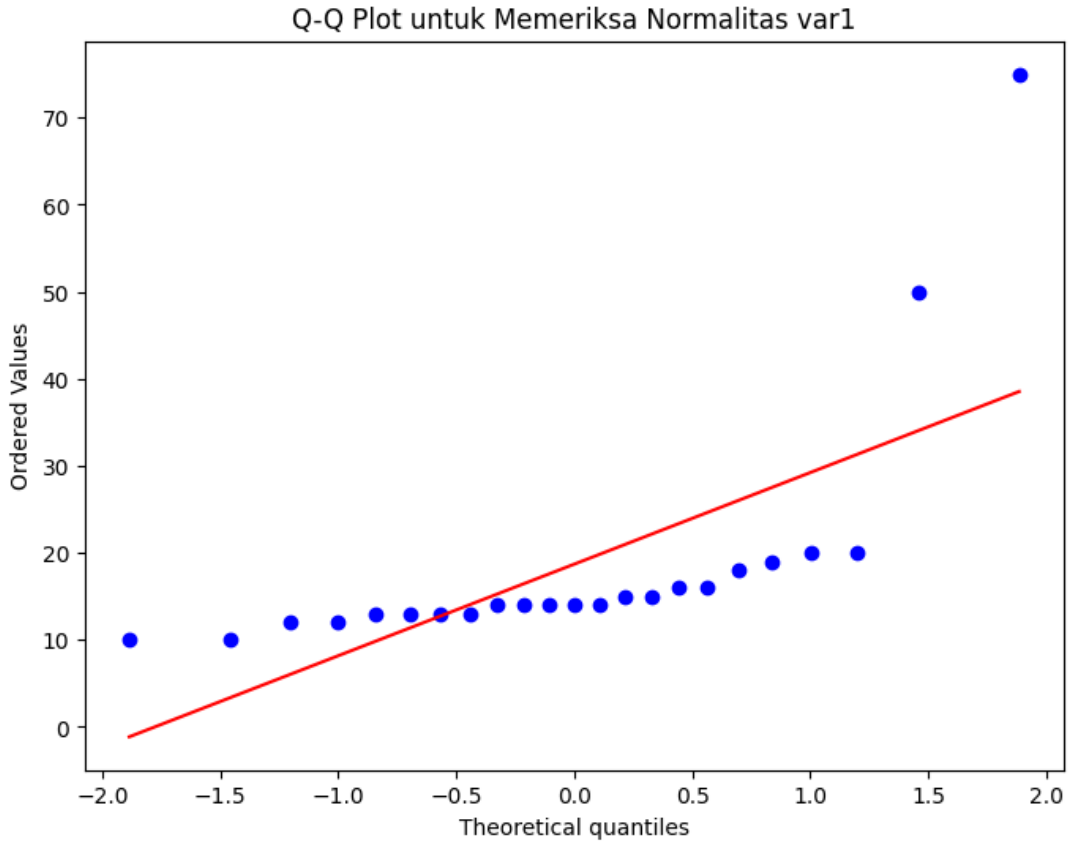
Implementasi Python:
Visualisasi Distribusi Data dengan Histogram

```
# Histogram untuk melihat
distribusi data
plt.figure(figsize=(8, 6))
sn.histplot(df1['var1'], kde=True,
bins=10)
plt.title('Histogram dan Density
Plot untuk var1')
plt.xlabel('var1')
plt.ylabel('Frekuensi')
plt.show()
```



Implementasi Python:
Visualisasi Distribusi Data dengan Q-Q Plot

```
# Q-Q plot untuk memeriksa normalitas
plt.figure(figsize=(8, 6))
stats.probplot(df1['var1'], dist="norm", plot=plt)
plt.title('Q-Q Plot untuk Memeriksa Normalitas var1')
plt.show()
```

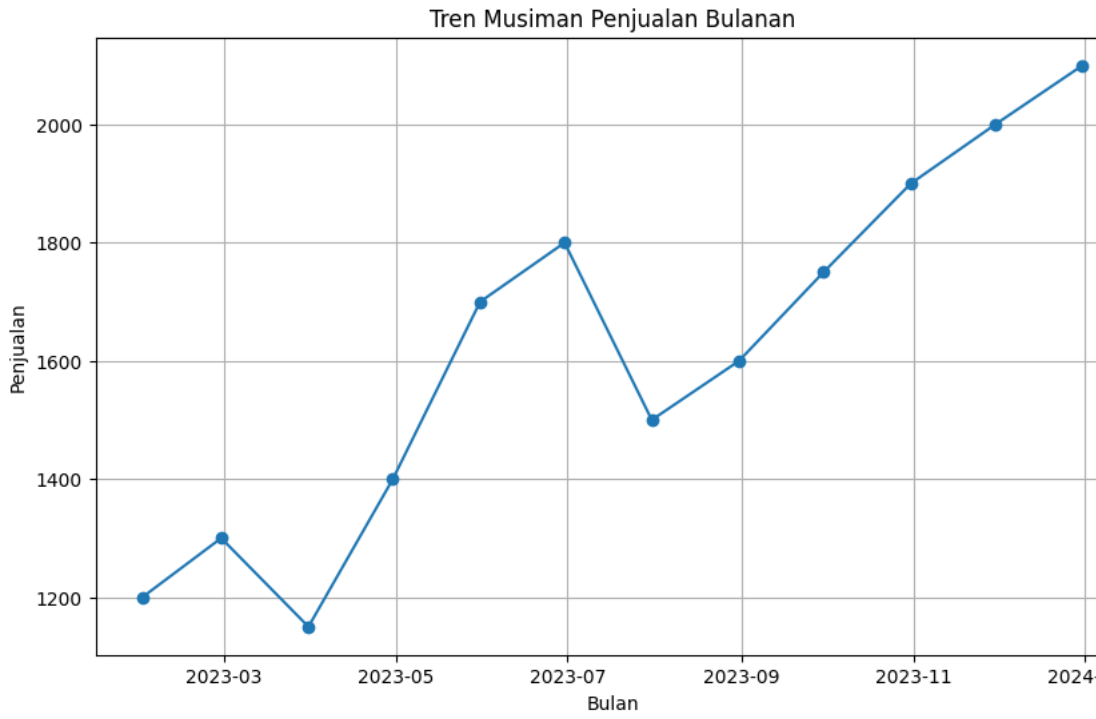


2. Identifikasi Pola dan Tren

- Menganalisis data untuk menemukan pola atau tren umum.
  - Misal:
    - mencari tren musiman dalam data penjualan
    - pola pembelian berdasarkan demografi pelanggan, dll
- Dapat menggunakan visualisasi misal line plot, bar plot, dll

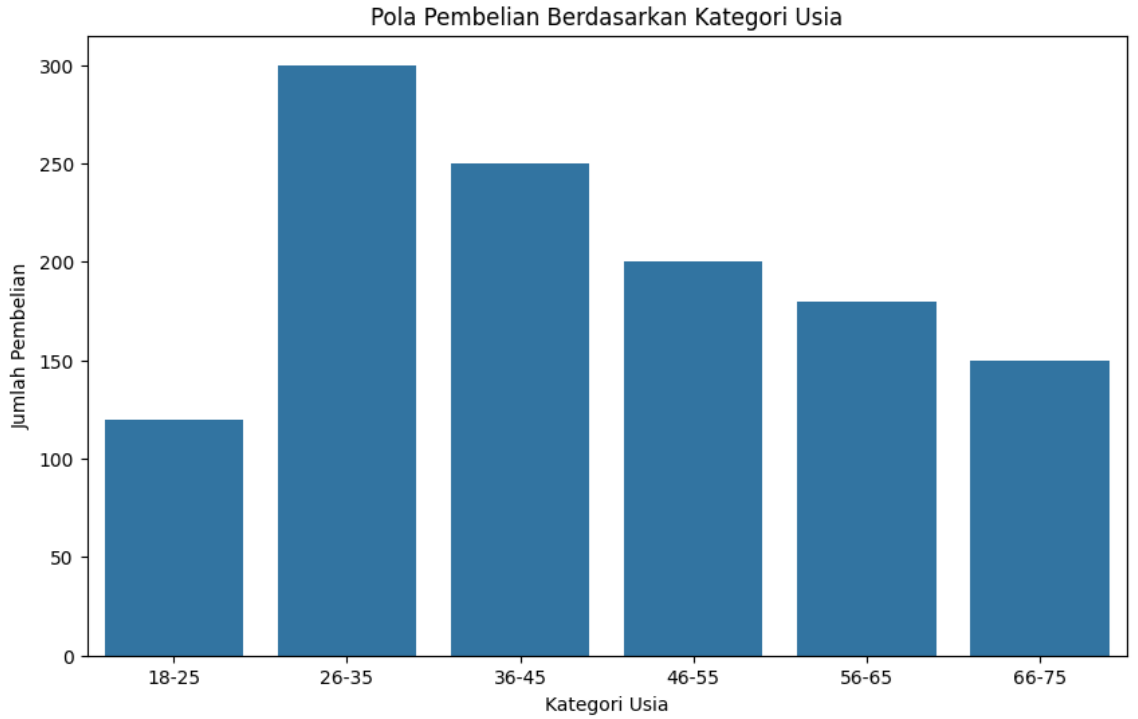
Implementasi Python:
Visualisasi Trend dengan Line plot

```
# Contoh data penjualan bulanan
data = {'bulan': pd.date_range(start='2023-01-
01', periods=12, freq='M'), 'penjualan': [1200,
1300, 1150, 1400, 1700, 1800, 1500, 1600, 1750,
1900, 2000, 2100]}
df = pd.DataFrame(data)
# Mengatur kolom 'bulan' sebagai index
df.set_index('bulan', inplace=True)
# Visualisasi tren penjualan
plt.figure(figsize=(10, 6))
plt.plot(df.index, df['penjualan'], marker='o')
plt.title('Tren Musiman Penjualan Bulanan')
plt.xlabel('Bulan')
plt.ylabel('Penjualan')
plt.grid(True)
plt.show()
```



Implementasi Python:
Visualisasi Pola dengan Bar chart

```
# Contoh data pembelian berdasarkan demografi pelanggan
data = {'usia': ['18-25', '26-35', '36-45', '46-55', '56-65', '66-75'], 'pembelian': [120, 300, 250, 200, 180, 150]}
df = pd.DataFrame(data)

# Visualisasi pola pembelian berdasarkan kategori usia
plt.figure(figsize=(10, 6))
sn.barplot(x='usia', y='pembelian', data=df)
plt.title('Pola Pembelian Berdasarkan Kategori Usia')
plt.xlabel('Kategori Usia')
plt.ylabel('Jumlah Pembelian')
plt.show()
```



3. Analisis Hubungan Antar Variabel

Untuk membantu memahami variabel mana yang mungkin saling memengaruhi atau berperan penting dalam model:

- **Korelasi antar Atribut Numerik:**
  - Menghitung matriks korelasi untuk memahami hubungan antara atribut numerik dalam dataset.
- **Relasi Atribut Kategorikal:**
  - Menganalisis hubungan antara atribut kategorikal
  - Menggunakan tabel kontingensi (*contingency tables*) untuk memahami distribusi gabungan.

3. Analisis Hubungan Antar Variabel (2)

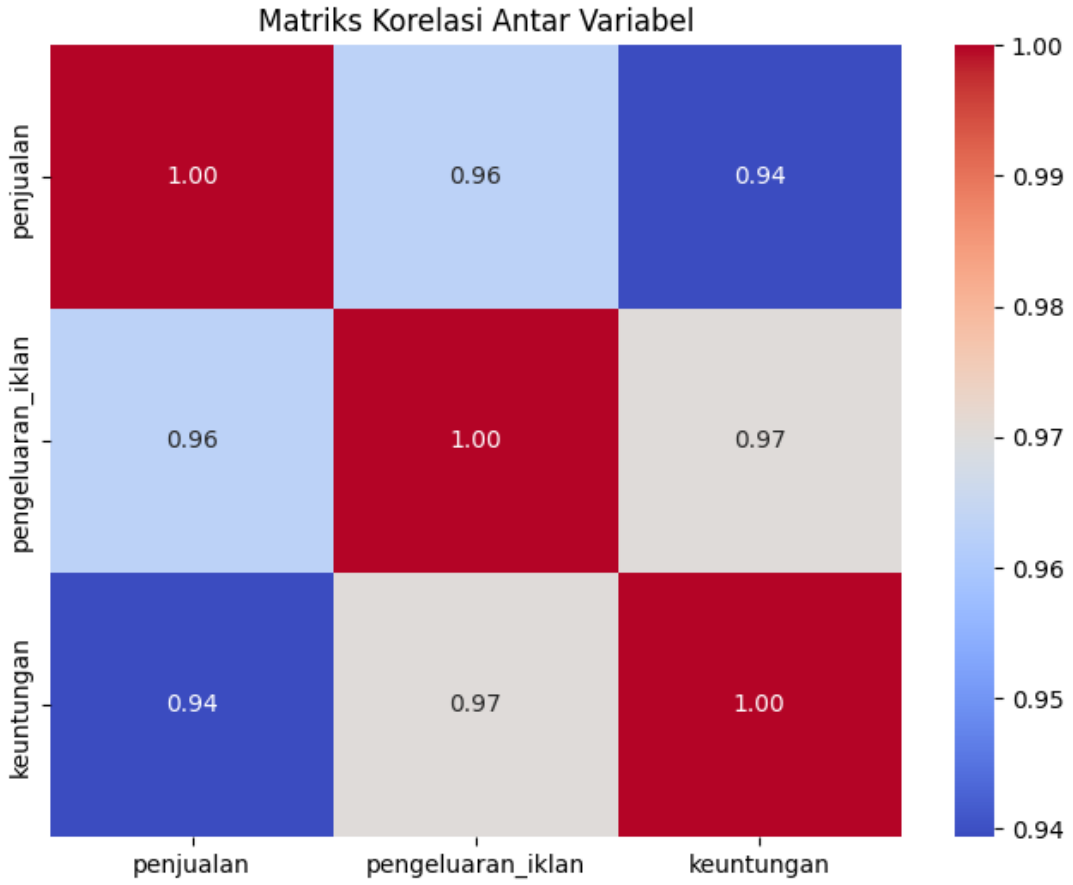
Cara yang dapat dilakukan:

- **Visualisasi Data**
  - Misal, menggunakan matriks korelasi, heatmaps, dll
- **Regresi Sederhana:**
  - Melakukan regresi linier sederhana antara dua variabel untuk melihat seberapa kuat hubungan antara keduanya.
  - Ini bisa membantu dalam mengidentifikasi variabel prediktor yang potensial.
- **Analisis Faktor:**

  Menggunakan analisis faktor atau PCA (Principal Component Analysis) untuk mengurangi dimensi data dan mengidentifikasi variabel-variabel utama yang menjelaskan sebagian besar variasi dalam data.

Implementasi Python:
Identifikasi Korelasi Antar Atribut

```
# Contoh data numerik untuk analisis korelasi
data = {'penjualan': [1200, 1300, 1150, 1400,
1700, 1800, 1500, 1600, 1750, 1900, 2000,
2100], 'pengeluaran_iklan': [200, 250, 180,
230, 300, 350, 220, 270, 310, 330, 380, 400],
'keuntungan': [300, 350, 320, 370, 400, 450,
340, 360, 390, 420, 460, 480]}
df = pd.DataFrame(data)
# Menghitung matriks korelasi
corr_matrix = df.corr()
# Visualisasi matriks korelasi menggunakan
heatmap
plt.figure(figsize=(8, 6))
sn.heatmap(corr_matrix, annot=True,
cmap='coolwarm', fmt='.2f')
plt.title('Matriks Korelasi Antar Variabel')
plt.show()
```



Implementasi Python:
Identifikasi Hubungan Atribut Kategorikal

```
#Menganalisis hubungan antara jenis kelamin (gender) dan preferensi produk (product preference).
# Contoh data kategorikal
data = {'gender': ['Male', 'Female', 'Male', 'Female', 'Male', 'Female', 'Female', 'Male', 'Female', 'Male'],
        'product_preference': ['Product A', 'Product B', 'Product A', 'Product A', 'Product B', 'Product B', 'Product A', 'Product B', 'Product A', 'Product B']}
df = pd.DataFrame(data)
# Membuat tabel kontingensi
contingency_table = pd.crosstab(df['gender'], df['product_preference'])
print(contingency_table)
```

```
product_preference  Product A  Product B
gender
Female                      3          2
Male                        2          3
```

Center for
Artificial Intelligence
ITB

4. Analisis Kategorisasi dan Grup

- **Analisis Segmentasi:**
  - Mengelompokkan data berdasarkan karakteristik tertentu untuk melihat perbedaan dalam kelompok yang berbeda.
  - Misalnya, menggunakan teknik clustering seperti K-Means atau hierarki clustering untuk membagi data ke dalam kelompok-kelompok berdasarkan kemiripan fitur.

(akan dibahas lebih lanjut di materi unsupervised modelling/clustering)

5. Eksplorasi Waktu dan Sekuensial

- **Analisis Time Series:**

  Untuk data yang berkaitan dengan waktu, melakukan analisis deret waktu untuk memahami pola temporal, seperti tren, siklus, dan musiman.

- **Analisis Sekuensial:**

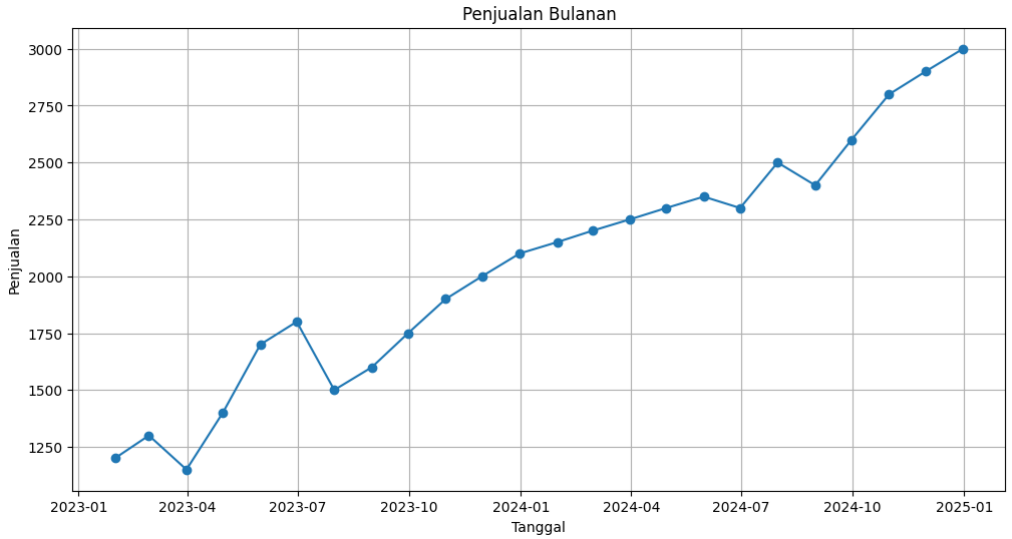
  Mengidentifikasi pola atau aturan dalam urutan data, seperti analisis urutan pembelian produk dalam data transaksi pelanggan.

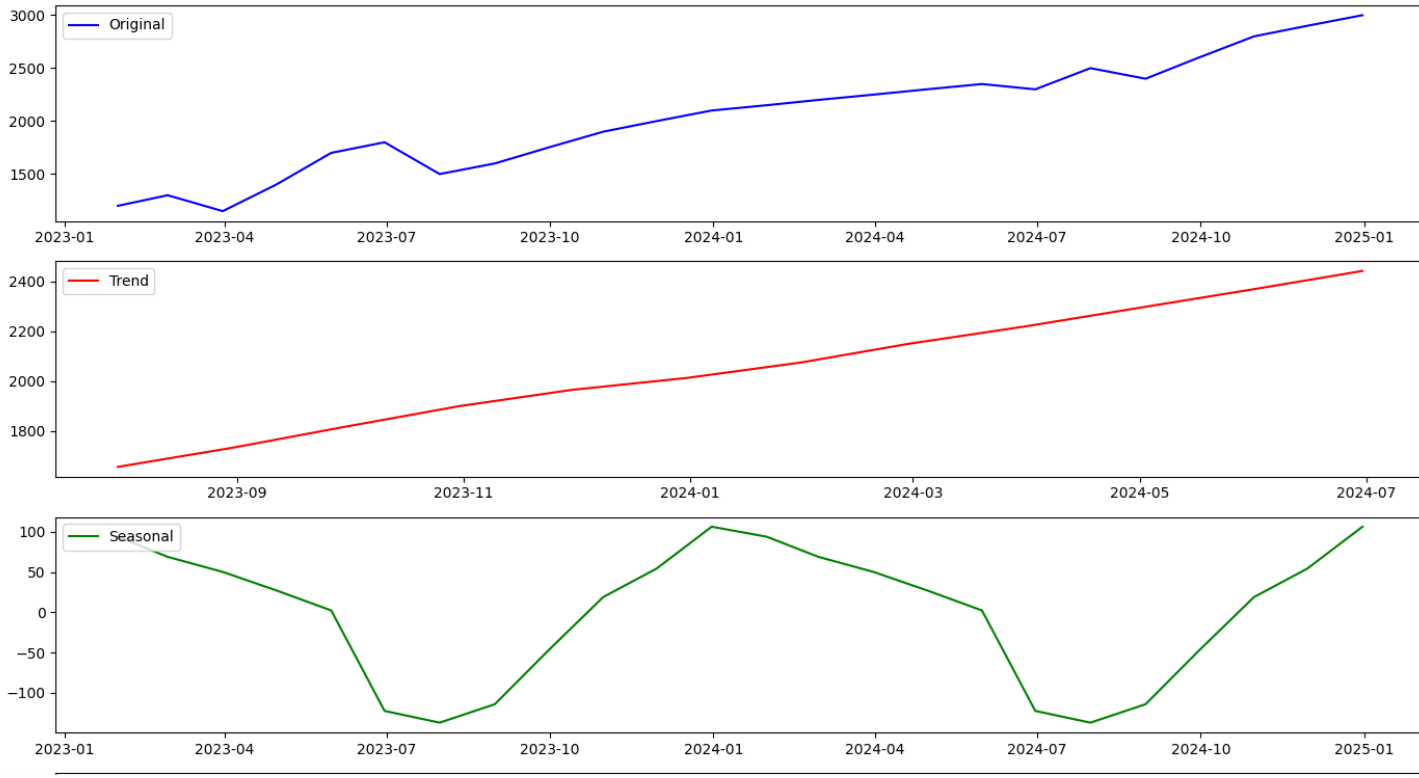
Implementasi Python:
Analisis Time Series

```
from statsmodels.tsa.seasonal import seasonal_decompose

# Contoh data penjualan bulanan
data = {'tanggal': pd.date_range(start='2023-01-01',
periods=24, freq='M'),
       'penjualan': [1200, 1300, 1150, 1400, 1700, 1800,
1500, 1600, 1750, 1900, 2000, 2100, 2150, 2200, 2250,
2300, 2350, 2300, 2500, 2400, 2600, 2800, 2900, 3000]}
df = pd.DataFrame(data)

# Mengatur 'tanggal' sebagai index
df.set_index('tanggal', inplace=True)

# Plot data time series
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['penjualan'], marker='o',
linestyle='-')
plt.title('Penjualan Bulanan')
plt.xlabel('Tanggal')
plt.ylabel('Penjualan')
plt.grid(True)
plt.show()
```

```
# Decompose time series
decomposition = seasonal_decompose(df['penjualan'],
model='additive')

# Plot decomposed components
plt.figure(figsize=(14, 10))
plt.subplot(4, 1, 1)
plt.plot(df.index, df['penjualan'], label='Original',
color='blue')
plt.legend(loc='upper left')
plt.subplot(4, 1, 2)
plt.plot(decomposition.trend, label='Trend',
color='red')
plt.legend(loc='upper left')
plt.subplot(4, 1, 3)
plt.plot(decomposition.seasonal, label='Seasonal',
color='green')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```

Implementasi Python:
Analisis Time Series





Center for
Artificial Intelligence
ITB

Implementasi Python:
Analisis Sekuensial

```
from mlxtend.frequent_patterns import apriori, association_rules

# Contoh data transaksi pembelian pelanggan
data = {'transaksi_id': [1, 1, 1, 2, 2, 3, 3, 3, 3, 4, 4],
        'produk': ['A', 'B', 'C', 'A', 'D', 'B', 'C', 'E', 'F', 'A', 'C']}
df = pd.DataFrame(data)
# Membuat format one-hot encoding untuk data transaksi
basket = df.groupby(['transaksi_id',
'produk'])['produk'].count().unstack().reset_index().fillna(0).set_index('transaksi_id')
basket = basket.map(lambda x: 1 if x > 0 else 0)
# Menerapkan algoritma Apriori untuk menemukan itemset yang sering muncul
frequent_itemsets = apriori(basket, min_support=0.1, use_colnames=True)
# Menghitung aturan asosiasi
rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1)
# Menampilkan itemset yang sering muncul dan aturan asosiasi
print("Frequent Itemsets:")
print(frequent_itemsets)
print("\n Association Rules:")
data_table.DataTable(rules)
```

Implementasi Python:
Analisis Sekuensial

```
Frequent Itemsets:
    support      itemsets
0      0.75           (A)
1      0.50           (B)
2      0.75           (C)
3      0.25           (D)
4      0.25           (E)
5      0.25           (F)
6      0.25        (A, B)
7      0.50        (C, A)
8      0.25        (A, D)
9      0.50        (C, B)
10     0.25        (E, B)
11     0.25        (B, F)
12     0.25        (E, C)
13     0.25        (C, F)
14     0.25        (E, F)
15     0.25     (C, A, B)
16     0.25     (E, C, B)
17     0.25     (C, B, F)
18     0.25     (E, B, F)
19     0.25     (E, C, F)
20     0.25  (E, C, B, F)
```

Support: proporsi atau persentase dari total transaksi yang mengandung item set tersebut.

Implementasi Python:
Analisis Sekuensial

index	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
0	frozenset({'A'})	frozenset({'D'})	0.75	0.25	0.25	0.3333333333333333	1.3333333333333333
1	frozenset({'D'})	frozenset({'A'})	0.25	0.75	0.25	1.0	1.3333333333333333
2	frozenset({'C'})	frozenset({'B'})	0.75	0.5	0.5	0.6666666666666666	1.3333333333333333
3	frozenset({'B'})	frozenset({'C'})	0.5	0.75	0.5	1.0	1.3333333333333333
4	frozenset({'E'})	frozenset({'B'})	0.25	0.5	0.25	1.0	2.0
5	frozenset({'B'})	frozenset({'E'})	0.5	0.25	0.25	0.5	2.0
6	frozenset({'B'})	frozenset({'F'})	0.5	0.25	0.25	0.5	2.0
7	frozenset({'F'})	frozenset({'B'})	0.25	0.5	0.25	1.0	2.0
8	frozenset({'E'})	frozenset({'C'})	0.25	0.75	0.25	1.0	1.3333333333333333
9	frozenset({'C'})	frozenset({'E'})	0.75	0.25	0.25	0.3333333333333333	1.3333333333333333
10	frozenset({'C'})	frozenset({'F'})	0.75	0.25	0.25	0.3333333333333333	1.3333333333333333
11	frozenset({'F'})	frozenset({'C'})	0.25	0.75	0.25	1.0	1.3333333333333333
12	frozenset({'E'})	frozenset({'F'})	0.25	0.25	0.25	1.0	4.0
13	frozenset({'F'})	frozenset({'E'})	0.25	0.25	0.25	1.0	4.0
14	frozenset({'C', 'A'})	frozenset({'B'})	0.5	0.5	0.25	0.5	1.0
15	frozenset({'A', 'B'})	frozenset({'C'})	0.25	0.75	0.25	1.0	1.3333333333333333
16	frozenset({'C'})	frozenset({'A', 'B'})	0.75	0.25	0.25	0.3333333333333333	1.3333333333333333
17	frozenset({'B'})	frozenset({'C', 'A'})	0.5	0.5	0.25	0.5	1.0
18	frozenset({'E', 'C'})	frozenset({'B'})	0.25	0.5	0.25	1.0	2.0
19	frozenset({'E', 'B'})	frozenset({'C'})	0.25	0.75	0.25	1.0	1.3333333333333333
20	frozenset({'C', 'B'})	frozenset({'E'})	0.5	0.25	0.25	0.5	2.0
21	frozenset({'E'})	frozenset({'C', 'B'})	0.25	0.5	0.25	1.0	2.0
22	frozenset({'C'})	frozenset({'E', 'B'})	0.75	0.25	0.25	0.3333333333333333	1.3333333333333333
23	frozenset({'B'})	frozenset({'E', 'C'})	0.5	0.25	0.25	0.5	2.0
24	frozenset({'C', 'B'})	frozenset({'F'})	0.5	0.25	0.25	0.5	2.0

Support menghitung proporsi atau frekuensi item set tersebut dalam semua transaksi.

Confidence dihitung dengan membagi jumlah transaksi yang mengandung kedua item (antecedent dan consequent) dengan jumlah transaksi yang mengandung antecedent.

Lift mengukur sejauh mana antecedent dan consequent lebih sering terjadi bersama-sama dibandingkan jika mereka terjadi secara independen. (confidence (antecedent - consequent) dibagi support (consequent)

6. Dokumentasi dan Interpretasi

- **Mencatat Temuan:**
  - Semua temuan penting selama eksplorasi data harus didokumentasikan dengan baik.
    - Variabel-variabel kunci
    - Pola yang ditemukan
    - Hipotesis yang muncul
- **Penyesuaian Hipotesis:**
  - Berdasarkan hasil eksplorasi, mungkin perlu dilakukan penyesuaian pada hipotesis awal atau rencana analisis.
  - Eksplorasi data sering kali membuka wawasan baru yang memengaruhi arah analisis selanjutnya.

4. Verifikasi Kualitas Data

Verifikasi Kualitas Data

- Untuk memastikan bahwa data yang digunakan **bersih, konsisten, dan dapat diandalkan**.
- Aktivitas ini bertujuan untuk mendeteksi berbagai masalah data yang dapat mempengaruhi hasil pemodelan, seperti:
  - Data yang hilang
  - Duplikasi
  - Inkonsistensi
  - Kesalahan format
  - Bias
  - Imbalanced (tidak berimbang)

Aktivitas dalam Verifikasi Kualitas Data

- Identifikasi *Missing Values*
- Identifikasi Duplikasi Data
- Verifikasi Konsistensi Data
- Verifikasi Format Data
- Identifikasi Keterkaitan Data
- Identifikasi Anomali Data
- Identifikasi Keterwakilan Data (Sampling Bias)
- Verifikasi Integritas Data
- Dokumentasi Kualitas Data

1.Identifikasi Missing Values

- Mengidentifikasi nilai-nilai yang hilang (*missing values*) dalam dataset dan menentukan seberapa banyak data yang hilang di setiap kolom dan baris.
- Ini penting karena data yang hilang bisa mempengaruhi analisis dan hasil yang diperoleh.

Contoh Implementasi Python:
Identifikasi Missing Values

```
dfnull = pd.read_csv('ai4i2020-null.csv')

# Identifikasi missing values
missing_values = dfnull.isnull().sum()
print(missing_values)
```

Akan menampilkan berapa jumlah NULL value pada setiap kolom

	0
UDI	0
Product ID	2
Type	0
Air temperature [K]	0
Process temperature [K]	0
Rotational speed [rpm]	0
Torque [Nm]	0
Tool wear [min]	0
Machine failure	0
TWF	0
HDF	0
PWF	0
OSF	0
RNF	0

dtype: int64

2. Identifikasi Duplikasi Data

- Memeriksa apakah ada baris data yang terduplikasi dalam dataset.
- Bisa terjadi karena kesalahan input atau penggabungan data yang tidak benar.

Contoh Implementasi Python:
Identifikasi Duplikat Data

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

print("Data Indomie:")
print(dfIndomie)

duplicated_rows = dfIndomie[dfIndomie.duplicated()]

print("\nDuplikat Data:")
print(duplicated_rows)
```

```
Data Indomie:
     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

Duplikat Data:
     brand style  rating
1  Yum Yum   cup     4.0
```

3. Identifikasi Konsistensi Data

- **Validasi Nilai Kategorikal:**
  - Memastikan konsistensi dalam data kategorikal
  - Seperti memastikan tidak ada nilai yang salah ejaan atau penggunaan yang tidak konsisten dari kategori yang sama, misal:
    - "Pria" dan "Laki-laki"
- **Validasi Rentang Nilai:**
  - Memeriksa apakah nilai numerik berada dalam rentang yang logis dan sesuai dengan domain data.
  - Misal: tidak ada usia negatif atau harga produk yang terlalu tinggi dibandingkan dengan harga normal

Contoh Implementasi Python:
Identifikasi Konsistensi Data

```
# Validasi kategori
print(dfnull['Type'].value_counts())
```

```
Type
L    6000
M    2997
H    1003
Name: count, dtype: int64
```

```
# Validasi rentang nilai
invalid_values = dfnull[(dfnull['Air temperature [K]'] < 0) | (dfnull['Air temperature [K]'] > 304.2)]
data_table.DataTable(invalid_values)
```

index	UDI	Product ID	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Machine failure	TWF	HDF	PWF	OSF	RNF
5068	5069.0	M19928	M	304.3	313.3	1540	39.0	14	0	0	0	0	0	0
5077	5078.0	L52257	L	304.3	313.3	1775	24.9	39	0	0	0	0	0	0
5078	5079.0	M19938	M	304.3	313.4	1482	43.8	41	0	0	0	0	0	0
5125	5126.0	M19985	M	304.3	313.5	1416	48.1	167	0	0	0	0	0	0
5126	5127.0	L52306	L	304.3	313.5	1802	25.4	170	0	0	0	0	0	0
5127	5128.0	M19987	M	304.3	313.5	2245	15.1	172	0	0	0	0	0	0

4. Identifikasi Format Data

- **Mengidentifikasi format data yang salah**
- Misal:
  - Validasi Format Tanggal dan Waktu:
    - Memeriksa apakah kolom tanggal dan waktu memiliki format yang benar
  - Data lain, seperti kode pos, nomor telepon, atau alamat email, sesuai dengan format yang diharapkan.

Implementasi Python:
Identifikasi Format Data

```
# Validasi format tanggal
# Contoh data dengan variabel waktu
data = {'tanggal': ['2023-01-15', '2023-15-17', '2023-03-
19', '2023-04-21', '2023-05-23', '2023-06-25', '2023-07-27',
'2023-08-29', '2023-09-30', '2023-10-01']}
df = pd.DataFrame(data)
df['tanggal'] = pd.to_datetime(df['tanggal'],
errors='coerce')
print(df)
```

```
     tanggal
0 2023-01-15
1        NaT
2 2023-03-19
3 2023-04-21
4 2023-05-23
5 2023-06-25
6 2023-07-27
7 2023-08-29
8 2023-09-30
9 2023-10-01
```

Implementasi Python:
Identifikasi Format Data

```
# Validasi format data lainnya
data = {'email': ['not valid email com',
'maria_garcia5678@randommail.com',
'linda.brown9876@mailservice.org',
'robert.jones5432@webmail.net',
'emily.davis2109@fakeemail.com',
'william.miller3456@demoemail.org',
'sophia.taylor6789@testingmail.com',
'james.wilson4321@samplemail.net',
'olivia.anderson9876@randommail.org',
'michael.thomas6543@fakemail.com'
]}
df = pd.DataFrame(data)
df['email_valid'] =
df['email'].str.contains(r'^[\w\.-]+@[\w\.-]+\.\w+$')
print(df)
```

	email	email_valid
0	not valid email com	False
1	maria_garcia5678@randommail.com	True
2	linda.brown9876@mailservice.org	True
3	robert.jones5432@webmail.net	True
4	emily.davis2109@fakeemail.com	True
5	william.miller3456@demoemail.org	True
6	sophia.taylor6789@testingmail.com	True
7	james.wilson4321@samplemail.net	True
8	olivia.anderson9876@randommail.org	True
9	michael.thomas6543@fakemail.com	True

5. Pemeriksaan Keterkaitan Data

- **Memeriksa hubungan antar variabel yang seharusnya memiliki hubungan logis.**
- Misalnya:
  - memastikan bahwa tanggal mulai selalu sebelum tanggal berakhir
  - kategori subtipe selalu sesuai dengan tipe utama.

Implementasi Python:
Pemeriksaan Keterkaitan Data

```
# Validasi tanggal mulai dan berakhir
data = {'tanggal_mulai': ['2023-01-15', '2023-06-22', '2023-08-
04', '2023-11-19', '2024-01-10'],'tanggal_akhir': ['2024-03-25',
'2024-05-17', '2023-07-09', '2024-09-30', '2023-12-14']}
df = pd.DataFrame(data)
invalid_dates = df[df['tanggal_mulai'] > df['tanggal_akhir']]
print(invalid_dates)
```

```
  tanggal_mulai tanggal_akhir
2    2023-08-04    2023-07-09
4    2024-01-10    2023-12-14
```

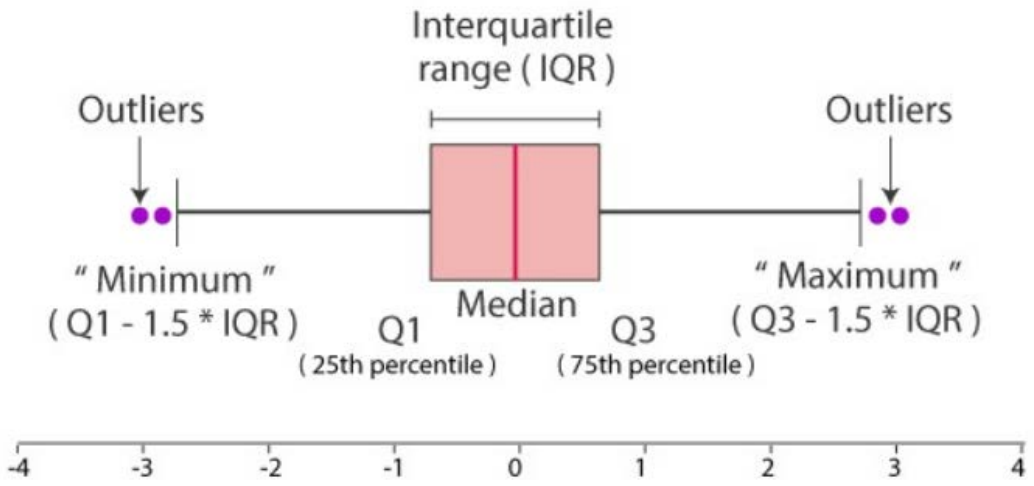
Implementasi Python:
Pemeriksaan Keterkaitan Data

```
# Validasi keterkaitan antar variabel
inconsistent_data = df[df['subtipe'] == 'A'] &
df['tipe'] != 'Tipe Utama A']
```

6. Identifikasi Anomali Data (Outliers)

- Memeriksa apakah ada outliers atau nilai ekstrim yang tidak sesuai dengan pola umum dalam data
- Diperlukan untuk menentukan apakah diperlukan pembersihan data pada tahap berikutnya atau pemahaman lebih lanjut tentang alasan di balik keberadaan outlier tersebut.
- **Dapat menggunakan analisis Distribusi Data:**
  - Menggunakan perhitungan IQR
  - Dapat juga Menggunakan visualisasi untuk mendeteksi anomali, seperti melihat distribusi data dengan histogram atau box plot untuk mengidentifikasi nilai yang tidak biasa.

Identifikasi Anomali Data (2)



Q1 = (n+1)*1/4

Q2 = (n+1)*3/4

IQR = Q3 - Q1

Lower Outlier = Q1 – (1.5 * IQR)

Higher Outlier= Q3 + (1.5 * IQR)

Menggunakan **Z-score** atau **Mahalanobis Distance** untuk mendeteksi outliers yang lebih kompleks.

Implementasi Python:
Identifikasi Outliers dengan IQR

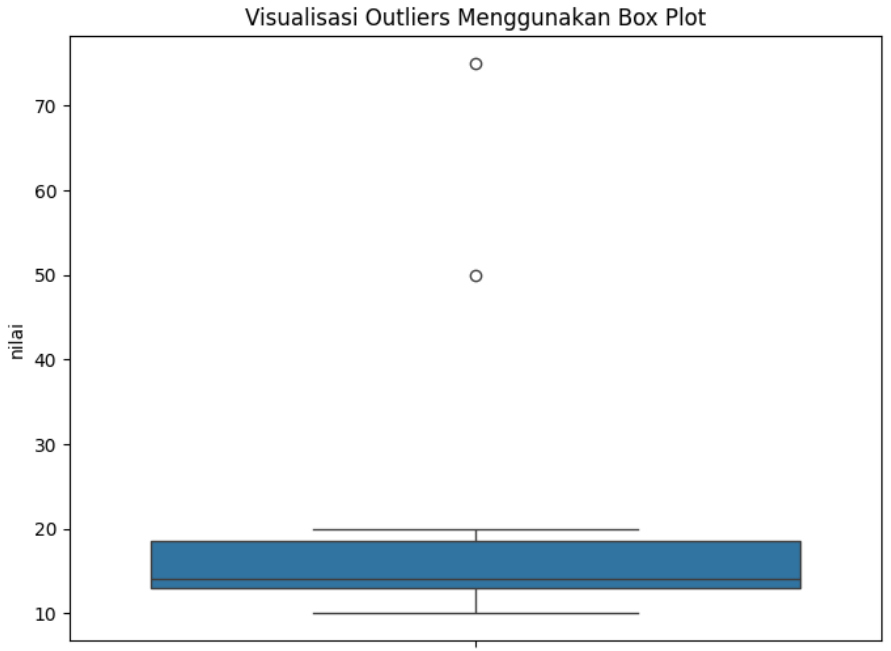
```
data = {'nilai': [12, 15, 14, 10, 50, 16, 14, 20, 13, 75, 18, 13, 14, 19, 13,
16, 14, 20, 13,12, 15, 14, 10]}
df = pd.DataFrame(data)

# Deteksi outliers menggunakan IQR
Q1 = df['nilai'].quantile(0.25)
Q3 = df['nilai'].quantile(0.75)
IQR = Q3 - Q1

outliers = df[(df['nilai'] < Q1 - 1.5 * IQR) | (df['nilai'] > Q3 + 1.5 * IQR)]
print(outliers)
```

```
   nilai
4     50
9     75
```

Center for
Artificial Intelligence
ITB

Implementasi Python:
Identifikasi Outliers dengan Visualisasi Boxplot

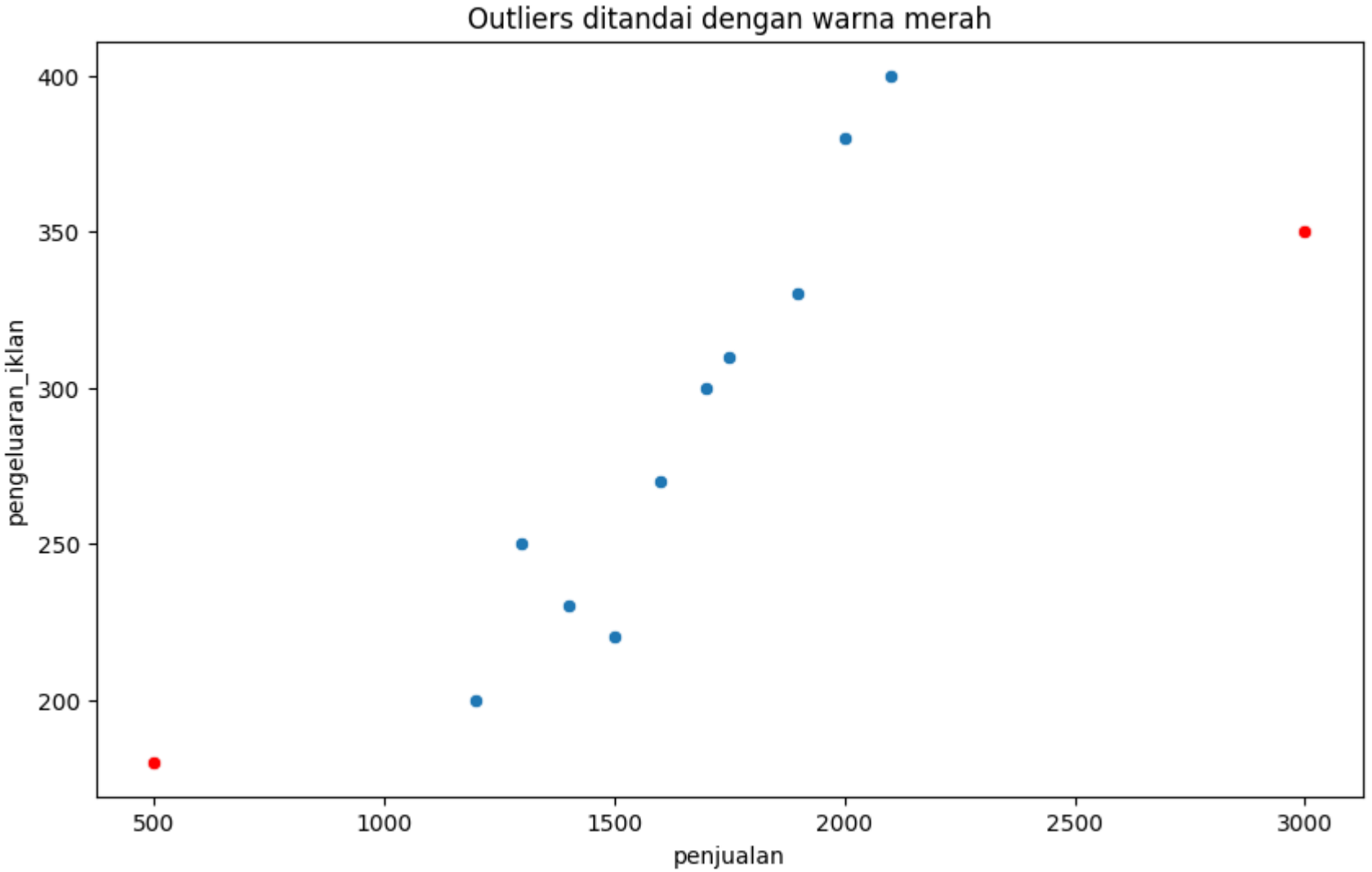
```
# Visualisasi outliers menggunakan box plot
plt.figure(figsize=(8, 6))
sn.boxplot(y=df['nilai'])
plt.title('Visualisasi Outliers Menggunakan Box Plot')
plt.show()
```



Implementasi Python:
Identifikasi Outliers dengan Visualisasi Scatter plot

```
data = {'penjualan': [1200, 1300, 500, 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)
# Visualisasi outliers menggunakan scatter plot
plt.figure(figsize=(10, 6))
sn.scatterplot(x='penjualan', y='pengeluaran_iklan', data=df)
# Deteksi outliers menggunakan IQR
Q1 = df['penjualan'].quantile(0.25)
Q3 = df['penjualan'].quantile(0.75)
IQR = Q3 - Q1
outliers = df[(df['penjualan'] < Q1 - 1.5 * IQR) | (df['penjualan'] > Q3 + 1.5 *
IQR)]
sn.scatterplot(x='penjualan', y='pengeluaran_iklan', data=outliers, color='red')
plt.title('Outliers ditandai dengan warna merah')
plt.show()
```

Implementasi Python:
Identifikasi Outliers dengan Visualisasi Scatter plot



Implementasi Python:
Identifikasi Outliers dengan Z Score

```
from scipy.stats import zscore
# Menghitung Z-Score
df['zscore'] = zscore(df['penjualan'])
print(df)
# Mengidentifikasi outliers (nilai dengan Z-
Score lebih besar dari 2 atau kurang dari -2)
# threshold nilai zscore yang dianggap outliers
perlu ditentukan berdasarkan data yang
dianalisis
outliers = df[np.abs(df['zscore']) > 2]
print(f"Jumlah outliers: {len(outliers)}")
```

```
    penjualan  pengeluaran_iklan    zscore
0        1500                200 -0.446417
1        1600                250 -0.251617
2         700                180 -2.004820
3        1400                230 -0.641218
4        1700                300 -0.056817
5        3000                350  2.475587
6        1500                220 -0.446417
7        1600                270 -0.251617
8        1750                310  0.040583
9        1900                330  0.332784
10       2000                380  0.527584
11       2100                400  0.722385
Jumlah outliers: 2
```

7. Identifikasi Keterwakilan Data

- **Analisis Distribusi Data:**

  Memastikan bahwa distribusi data dalam sampel mewakili populasi yang lebih besar, sehingga tidak ada bias yang akan mempengaruhi hasil analisis.

- **Pemeriksaan Imbalance Data**:

  Memeriksa apakah ada ketidakseimbangan kelas dalam data kategorikal, terutama untuk masalah klasifikasi.

Implementasi Python:
Identifikasi Keterwakilan Data

```
# Memeriksa distribusi kelas
print(dfnull['Type'].value_counts())
```

```
Type
L    6000
M    2997
H    1003
Name: count, dtype: int64
```

8. Identifikasi Integritas Data

- **Memastikan Konsistensi Data:**
  - Memeriksa apakah data di seluruh dataset konsisten
  - misalnya, apakah total penjualan di laporan harian sesuai dengan total penjualan di laporan bulanan.
- **Validasi Hubungan Antar Tabel:**

  Jika menggunakan database relasional, pastikan hubungan antar tabel, seperti foreign key constraints, dipatuhi.

Implementasi Python:
Identifikasi Integritas Data

```
# Memastikan konsistensi total penjualan
total_penjualan_harian = df_harian['penjualan'].sum()
total_penjualan_bulanan = df_bulanan['penjualan'].sum()
assert total_penjualan_harian == total_penjualan_bulanan,
"Penjualan tidak konsisten!"

# Validasi foreign key
valid_foreign_key = df_foreign_key.isin(df_primary_key).all()
```

9. Dokumentasi Kualitas Data

- Laporan Kualitas Data:
  - Menyusun laporan kualitas data yang mencakup semua masalah yang ditemukan, bagaimana masalah tersebut memengaruhi data,
  - Ini penting untuk memastikan transparansi dan memungkinkan anggota tim lain untuk memahami kondisi data yang ada.

Implementasi Python:
Laporan Kualitas Data

```
# Menyimpan laporan kualitas data ke dalam file teks
with open('data_quality_report.txt', 'w') as f:
    f.write(f"Missing values:\n{missing_values}\n")
    f.write(f"Duplicated rows:\n{len(duplicated_rows)}\n")
    f.write(f"Outliers detected:\n{len(outliers)}\n")
    f.write(f"Inconsistent data:\n{len(invalid_dates)}\n")
```

Kesimpulan

Aktivitas *Data Understanding*

1. **Pengumpulan Data Awal :**

   Penentuan sumber data, pengumpulan, format, penyimpanan, dokumentasi

2. **Deskripsi Data :** Struktur data, Ringkasan statistik, dan Dokumentasi

3. **Eksplorasi Data :**: Statistik Deskriptif Lanjutan, Pola dan Tren, Analisis Hubungan Antar Variabel, Analisis Kategorisasi dan Grup, Eksplorasi Waktu dan Sekuensial, Dokumentasi

4. **Verifikasi Kualitas Data:** Identifikasi *Missing Values,* Duplikasi Data, Konsistensi Data, Format Data, Keterkaitan Data, Anomali Data, Sampling Bias, Integritas Data, Dokumentasi

Terima Kasih