PERTEMUAN 13

Pengolahan File dan Data Eksternal

Membaca log produksi dan membuat laporan otomatis dalam format TXT, CSV, dan JSON

open · with · read · write · csv · json · os



Data Today, Better Tomorrow.

Capaian Pembelajaran Pertemuan Ini

Setelah mengikuti pertemuan ini, mahasiswa diharapkan mampu mengolah data dari file eksternal untuk kebutuhan pelaporan industri.

SUB-CPMK

Mahasiswa mampu membaca, menulis, dan memperbarui data dari file eksternal untuk kebutuhan pelaporan industri.

Level kognitif: C3
(Menerapkan)

Indikator Ketercapaian

1. Membuka file dengan mode **r**, **w**, **a**, dan **x** menggunakan pernyataan **with**.
2. Membaca dan menulis file **TXT** serta **CSV**.
3. Mengonversi data Python ke dan dari format **JSON**.
4. Membuat laporan otomatis ke file dan menangani file yang tidak ditemukan.

KODE LEBIH RAPI
DATA LEBIH BERMAKNA
MASA DEPAN LEBIH NYATA

Peta Materi

Materi pada pertemuan ini disusun secara sistematis agar mudah dipahami dan langsung diterapkan.

1 Konsep File



Alasan memakai file, mode, pernyataan with, path

Slide 4 – 6

2 Membaca TXT



read, readline, loop baris, parsing, error

Slide 7 – 9

3 Menulis & Kelola



Mode w, a, x, dan modul os

Slide 10 – 12

4 CSV



Format, reader, DictReader, writer

Slide 13 – 16

5 JSON



Pemetaan tipe, dump/load, update data

Slide 17 – 19

6 Pipeline & Promax



Pareto, downtime, studi kasus laporan

Slide 20 – 25



Saran Alokasi Waktu

30 menit konsep dan TXT, 35 menit CSV dan JSON, 35 menit pipeline, promax, dan studi kasus.



Persiapan

Siapkan file data praktik di Colab sebelum kelas dimulai (unggah lewat panel Files, atau buat dengan %%writefile).

Mengapa Program Perlu File?

File memungkinkan data disimpan secara permanen, dapat dibuka kembali, serta dapat digunakan oleh program lain.

Variabel di RAM

Cepat, tetapi hanya berlaku selama program berjalan. Data akan hilang saat program selesai atau Colab di-restart.

File di Disk

Permanen, bisa dibuka oleh program lain, dan dapat dikirim ke orang lain. Cocok untuk menyimpan log, data produksi, laporan, dan konfigurasi.

Analogi:
Variabel seperti tulisan di papan tulis ruang produksi, file seperti buku arsip. Papan tulis dihapus setiap ganti shift, arsip tetap tersimpan.

Contoh Isi File

TXT — log & laporan

`catatan_shift.txt`

```
Shift pagi: 480 unit
Mesin CNC-02 ganti tool
Terjadi sedikit getaran
pada spindle
Shift siang: 470 unit
Tidak ada kendala
signifikan
```

CSV — data tabel

`output_harian.csv`

```
tanggal,shift,output
2026-11-10,Pagi,245
2026-11-10,Siang,230
2026-11-10,Malam,210
```

JSON — konfigurasi

`config.json`

```
{
  "target_harian": 500,
  "batas_cacat_persen": 2.5,
  "mesin": "CNC-02",
  "aktif": true
}
```

Intinya: file membuat data bertahan lama dan menghubungkan program dengan mesin, Excel, dan sistem lain.

SATU LANGKAH KODE
UNTUK HASIL
YANG LEBIH BESAR

Mode Membuka File

Mode menentukan apa yang boleh dilakukan terhadap file serta apa yang terjadi jika file ada atau tidak ada.

Mode	Arti	Jika file tidak ada	Jika file sudah ada	Contoh di industri
"r"	Read (baca)	FileNotFoundError	Dibaca dari awal	Membaca log mesin
"w"	Write (tulis)	Dibuat baru	**Isi lama dihapus**	Membuat laporan harian baru
"a"	Append (tambah)	Dibuat baru	**Ditambah di akhir**	Mencatat log kejadian
"x"	Create (buat)	Dibuat baru	FileExistsError	Mencegah laporan tertimpa

Contoh Penggunaan

```python
# Membaca file
with open("log.txt", "r") as f:
    data = f.read()

# Menulis file (membuat / menimpa)
with open("laporan.txt", "w") as f:
    f.write("Laporan harian\n")

# Menumbah data di akhir file
with open("log.txt", "a") as f:
    f.write("Mesin berjalan normal\n")

# Membuat file baru (jika belum ada)
with open("data_baru.txt", "x") as f:
    f.write("File ini dibuat pertama kali")
```

Hati-hati dengan mode "w"

Membuka file dengan mode w langsung mengosongkan isinya, bahkan sebelum satu baris pun ditulis. Data yang penting bisa hilang dalam sekejap.

Intinya

Gunakan r untuk membaca, w untuk membuat ulang, a untuk menambah, dan x untuk membuat tanpa menimpa. Tambahan b (misal "rb") untuk file biner seperti gambar.

KELOLA FILE
KELOLA DATA
KELOLA MASA DEPAN

Pernyataan with dan Path File

Pernyataan with membuat pengelolaan file lebih aman dan rapi, sedangkan path menentukan lokasi file.

Pernyataan with

- Secara otomatis menutup file setelah selesai digunakan.
- Lebih aman, rapi, dan direkomendasikan.
- Tidak perlu memanggil `f.close()`.
- Cocok untuk membaca maupun menulis file.

Contoh Menggunakan with

```python
# Membaca file
with open("data.txt", "r") as f:
    for baris in f:
        print(baris.strip())
```

Tanpa menggunakan with (tidak disarankan)

```python
f = open("data.txt", "r")
for baris in f:
    print(baris.strip())
f.close()  # harus ditutup manual

```

Path File

- Path menunjukkan lokasi file pada sistem.
- Bisa berupa **path relatif** (dari folder kerja saat ini) atau **path absolut** (lokasi lengkap).
- Gunakan modul os untuk bekerja dengan **path** agar lebih fleksibel.
- Di Colab, file yang diunggah biasanya berada di folder `/content/`.

Jenis Path

Path relatif

Lokasi file relatif terhadap folder kerja saat ini.

```
data.txt
folder/data.csv
```

Path absolut

Lokasi lengkap dari root direktori.

```
/content/data.txt
C:\Users\Nama\data.txt
```

Struktur Folder (Contoh di Colab)



Gunakan `os.path` untuk menggabungkan path secara aman, misalnya:

```python
import os
path = os.path.join(
 "data", "produksi.csv")
```

Intinya

Gunakan **with** untuk pengelolaan file yang aman dan gunakan **path** yang tepat agar program mudah dijalankan di berbagai lingkungan.

Membaca File Teks: read, readline, dan Loop Baris

Metode dasar untuk membaca isi file teks, baik seluruh isi, per baris, maupun sebagai daftar baris.

catatan_shift.txt

```
Shift pagi: 480 unit
Mesin CNC-02 ganti tool 08:40
Target tercapai
```

Contoh Membaca dengan Loop Baris

python

```python
with open("catatan_shift.txt", encoding="utf-8") as f:
    judul = f.readline().strip()      # baris 1
    print(judul)
    for no, baris in enumerate(f, start=2):
        print(no, baris.strip())      # sisa baris
```

OUTPUT

```
Shift pagi: 480 unit
Mesin CNC-02 ganti tool 08:40
Target tercapai
```

`read()`	Seluruh isi jadi satu string
`readline()`	Satu baris berikutnya
`readlines()`	Semua baris jadi list
`for baris in f`	Baris demi baris, hemat memori

Intinya

setiap baris diakhiri karakter \n.
Gunakan strip() untuk membuangnya.

8 Mengurai Isi File: Menghitung Kejadian STOP Mesin

log_mesin.txt

```
2026-11-10 08:15 CNC-01 STOP
2026-11-10 08:40 CNC-01 RUN
2026-11-10 09:05 CNC-02 STOP
2026-11-10 10:35 CNC-02 RUN
2026-11-10 13:20 CNC-01 STOP
2026-11-10 13:50 CNC-01 RUN
2026-11-10 14:10 CNC-03 STOP
2026-11-10 14:25 CNC-03 RUN
```

split() memecah baris

```
"2026-11-10 08:15 CNC-01 STOP".split()
→ ['2026-11-10', '08:15', 'CNC-01', 'STOP']
```

Kode Program

python

```python
stop = {}
with open("log_mesin.txt", encoding="utf-8") as f:
    for baris in f:
        tgl, jam, mesin, status = baris.split()
        if status == "STOP":
            stop[mesin] = stop.get(mesin, 0) + 1
print(stop)
```

OUTPUT

```
{'CNC-01': 2, 'CNC-02': 1, 'CNC-03': 1}
```

Intinya: baca per baris → pecah dengan split() → olah dengan pola dictionary (Pertemuan 11).

9 Membaca File dengan Aman

Kode Program

```python
import os

nama = "log_minggu_lalu.txt"
try:
    with open(nama, encoding="utf-8") as f:
        data = f.read()
except FileNotFoundError:
    print(f"File {nama} tidak ditemukan.")

# alternatif: cek dulu sebelum membuka
if os.path.exists("log_mesin.txt"):
    print("File siap diproses")
```

OUTPUT

```
File log_minggu_lalu.txt tidak ditemukan.
File siap diproses
```

Penyebab umum

- Salah ketik nama file atau ekstensi
- File belum diunggah ke Colab
- Folder kerja berbeda
- Huruf besar/kecil berbeda (Linux)

Intinya:
program yang membaca file wajib siap menghadapi file yang tidak ada: pakai try-except FileNotFoundError atau cek os.path.exists().

10 Menulis File: Membuat Laporan Harian

Kode Program

python

```python
hasil = {"CNC-01": 520, "CNC-02": 470}

with open("laporan.txt", "w", encoding="utf-8") as f:
    f.write("LAPORAN PRODUKSI 10-11-2026\n")
    for m, o in hasil.items():
        f.write(f"{m}: {o} unit\n")
    print("Total:", sum(hasil.values()), file=f)
```

`laporan.txt (hasil)`

```
LAPORAN PRODUKSI 10-11-2026
CNC-01: 520 unit
CNC-02: 470 unit
Total: 990
```

`f.write(teks)`	Tulis teks; \n tidak otomatis
`writelines(list)`	Tulis banyak teks
`print(..., file=f)`	Seperti print, \n otomatis

Intinya:

write() hanya menerima string dan tidak menambah baris baru. Ubah angka ke teks dengan f-string dan akhiri dengan \n.

11

Mode a dan x: Log Kejadian dan Mencegah Tertimpa

Mode **a** dipakai untuk menambah isi di akhir file, sedangkan mode **x** dipakai untuk membuat file baru tanpa menimpa file lama.

Mode a / Kode Program

python

```python
from datetime import datetime

def catat(pesan):
    waktu = datetime.now().strftime("%Y-%m-%d %H:%M")
    with open("kejadian.log", "a", encoding="utf-8") as f:
        f.write(f"[{waktu}] {pesan}\n")

catat("CNC-02 STOP: bearing panas")
catat("CNC-02 RUN")
```

kejadian.log

```
[2026-11-10 09:05] CNC-02 STOP: bearing panas
[2026-11-10 10:35] CNC-02 RUN
```

Mengapa mode a?

1. Menambah data di akhir file
2. Isi lama tetap aman
3. Cocok untuk log kejadian

Mode x: aman dari penimpaan

python

```python
try:
    with open("laporan.txt", "x") as f:
        f.write("...")
except FileExistsError:
    print("Sudah ada, tidak ditimpa")
```

Perbedaan singkat

Mode a (append)	Tambah isi di akhir file; jika file belum ada, file akan dibuat.
Mode x (exclusive create)	Buat file baru; error jika file sudah ada.

Kapan dipakai?

1. Gunakan mode **a** untuk log mesin, histori, catatan proses.
2. Gunakan mode **x** untuk laporan final, template resmi, atau file yang tidak boleh tertimpa.

Intinya:

mode **a** menambah isi tanpa menghapus data lama, sehingga cocok untuk log. mode **x** hanya membuat file baru dan akan gagal jika nama file sudah ada, sehingga aman dari penimpaan.

12 Mengelola File dan Folder dengan Modul os

Kode Program

python

```python
import os

os.makedirs("arsip", exist_ok=True)          # buat folder
os.replace("laporan.txt",
           "arsip/laporan_2026-11-10.txt") # pindah + ganti nama
print(os.listdir("arsip"))                   # isi folder

if os.path.exists("kejadian.log"):
    os.remove("kejadian.log")                # hapus file
print(os.path.exists("kejadian.log"))        # False
```

OUTPUT

```
['laporan_2026-11-10.txt']
False
```

`os.path.exists(p)`	Cek file/folder ada
`os.listdir(p)`	Daftar isi folder
`os.makedirs(p)`	Buat folder
`os.replace(a, b)`	Pindah / ganti nama
`os.remove(p)`	Hapus file permanen

Urutan kerja yang aman

1. Buat folder tujuan
2. Pindahkan / ganti nama file
3. Lihat hasilnya dengan listdir()
4. Hapus file hanya jika yakin

Intinya:

os.remove() tidak masuk Recycle Bin. Selalu cek dengan exists() lebih dulu agar lebih aman.

13 Mengenal CSV: Format Tabel Paling Universal

`output_harian.csv`

```
tanggal,shift,output
2026-11-10,Pagi,245
2026-11-10,Siang,238
2026-11-11,Pagi,251
2026-11-11,Siang,229
```

tanggal	shift	output
2026-11-10	Pagi	245
2026-11-10	Siang	238
2026-11-11	Pagi	251
2026-11-11	Siang	229

Dibuka di Excel atau spreadsheet: satu baris = satu record, koma = batas kolom

Perhatian: Excel berbahasa Indonesia

- Di beberapa komputer Indonesia, Excel memakai titik koma (;) karena koma dipakai sebagai tanda desimal.
- Akibatnya, file CSV kadang terbuka menjadi satu kolom saja.
- **Solusi:** impor file sebagai CSV lalu pilih pemisah yang sesuai (, atau ;).
- Di Python, jika perlu, gunakan `delimiter=';'` saat membaca file.

Intinya:

CSV (Comma-Separated Values) adalah file teks sederhana berbentuk tabel. Baris pertama biasanya berisi header atau nama kolom. Setiap baris berikutnya adalah satu data/record. Nilai dipisahkan dengan koma, dan saat dibaca di Python biasanya masuk sebagai teks (string) terlebih dahulu.

14

csv.reader: Membaca Baris sebagai List

csv.reader() membaca setiap baris pada file CSV dan mengubahnya menjadi list (daftar nilai).

Kode Program

python

```python
import csv

total = 0
with open("output_harian.csv", newline="", encoding="utf-8") as f:
    pembaca = csv.reader(f)
    header = next(pembaca)          # lewati baris judul
    for baris in pembaca:
        total += int(baris[2])      # kolom ke-3: output

print(header)
print(f"Total output: {total} unit")
```

OUTPUT

```
['tanggal', 'shift', 'output']
Total output: 963 unit
```

Setiap baris = list

```
['2026-11-10', 'Pagi', '245']
baris[2] → '245' (string)
```

next(pembaca)

Mengambil baris pertama sebagai header sehingga loop dimulai dari data.

Intinya

- Selalu konversi dengan `int()` atau `float()` sebelum menghitung.
- `newline=""` wajib saat membuka file CSV.

Alur Pembacaan CSV



KELOLA FILE
KELOLA DATA
KELOLA MASA DEPAN

csv.DictReader: Mengakses Kolom dengan Nama

csv.DictReader() membaca setiap baris pada file CSV sebagai dictionary, sehingga kita dapat mengakses data menggunakan nama kolom (header), bukan posisi kolom.

Kode Program

python

```python
import csv

per_shift = {}
with open("output_harian.csv", newline="", encoding="utf-8") as f:
    for row in csv.DictReader(f):
        s = row["shift"]
        per_shift[s] = per_shift.get(s, 0) + int(row["output"])

print(per_shift)
```

OUTPUT

```
{'Pagi': 496, 'Siang': 467}
```

Setiap baris = dictionary

```
{'tanggal': '2026-11-10',
 'shift': 'Pagi',
 'output': '245'}
```

Nama kolom menjadi **key**, **isi** data menjadi **value**.

Kenapa lebih mudah?

- `row["output"]` lebih jelas dibaca.
- Tidak tergantung urutan kolom seperti baris[2].
- Aman jika posisi kolom berubah tetapi header tetap.

Kapan dipakai?

1. Saat file CSV memiliki header.
2. Saat ingin kode mudah dipahami.
3. Saat data diproses berdasarkan nama kolom.

Intinya: Intinya:

DictReader = CSV + dictionary. Metode ini menjadi pilihan utama untuk file CSV yang memiliki header karana akses data lebih jelas, mudah dipelihara, dan tahan terhadap perubahan struktur kolom.

Alur Menggunakan csv.DictReader



KELOLA FILE
KELOLA DATA
KELOLA MASA DEPAN

16

csv.writer: Menyimpan Hasil Olahan ke CSV

csv.writer() digunakan untuk menulis data (list/dictionary) ke dalam file CSV.

Kode Program

python

```python
import csv
# hasil dari slide 15
# per_shift = {'Pagi': 496, 'Siang': 467}

with open("rekap_shift.csv", "w", newline="", encoding="utf-8") as f:
    penulis = csv.writer(f)
    penulis.writerow(["shift", "total", "rata_rata"])
    for s, t in per_shift.items():
        penulis.writerow([s, t, t / 2])
```

rekap_shift.csv (hasil)

rekap_shift.csv

```
shift,total,rata_rata
Pagi,496,248.0
Siang,467,233.5
```

Cara Kerja csv.writer

1. *csv.writer()* membuat objek penulis CSV.
2. *writerow()* menulis satu baris data.
3. Data Python otomatis dikonversi menjadi format CSV.
4. *newline=""* mencegah munculnya baris kosong pada Windows.

Alternatif: DictWriter

python

```python
writer = csv.DictWriter(f, fieldnames=["shift", "total"])
writer.writeheader()
writer.writerow({"shift": "Pagi", "total": 496})
```

Kapan digunakan?

- Menyimpan hasil analisis data.
- Membuat laporan produksi.
- Ekspor data Python ke Excel/spreadsheet.

Intinya:
csv.writer digunakan untuk mengubah data list/dictionary Python menjadi file CSV yang mudah dibaca Excel. Gunakan writerow() untuk setiap baris dan selalu perhatikan *newline* saat membuat file.

Alur Menyimpan Data ke CSV



KELOLA FILE
KELOLA DATA
KELOLA MASA DEPAN

17

Mengenal JSON: Data Terstruktur yang Mirip Dictionary

JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan dan mudah dibaca.

pabrik_config.json — JSON

```
{
    "pabrik": "PT Presisi Nusantara",
    "target_harian": 500,
    "shift": ["Pagi", "Siang"],
    "lembur": false,
    "catatan": null
}
```

JSON menggunakan pasangan **key-value**, mirip seperti dictionary pada Python.

Python	JSON
`dict`	`object { }`
`list, tuple`	`array [ ]`
`str`	`string " "`
`int, float`	`number`
`True / False`	`true / false`
`None`	`null`

Struktur JSON

1. Object menggunakan kurung kurawal `{ }`
2. Data disimpan dalam pasangan `key` dan `value`
3. Key harus menggunakan tanda kutip ganda `" "`
4. Array menggunakan tanda kurung siku `[ ]`

Intinya:

JSON (JavaScript Object Notation) adalah format standar pertukaran data antar sistem. Struktur JSON sangat mirip dictionary Python sehingga mudah dibaca dan diolah.

Kapan digunakan?

- File konfigurasi aplikasi
- Pertukaran data API/web
- Penyimpanan data sensor IoT
- Integrasi antar aplikasi



18

json.dump dan json.load: Simpan & Baca Konfigurasi

JSON digunakan untuk menyimpan struktur data Python ke file dan mengembalikannya saat program dijalankan kembali.

Kode Program

pythom

```
import json

config = ["pabrik": "PT Presisi Nusantara", "target_harian": 500,
          "shift": ["Pagi", "Siang"], "lembur": False, "catatan": None}

with open("pabrik_config.json", "w", encoding="utf-8") as f:
    json.dump(config, f, indent=4)

with open("pabrik_config.json", encoding="utf-8") as f:
    cfg = json.load(f)

print(cfg["target_harian"], type(cfg["shift"]), cfg["lembur"])
```

OUTPUT

```
500 <class 'list"> False
```

Fungsi	Arah
`json.dump(obj, f)`	Python → file
`json.load(f)`	file → Python
`json.dumps(obj)`	Python → string
`json.loads(teks)`	string → Python

Perbedaan dump dan load

- **`json.dump()`**: mengubah dictionary/list Python menjadi file JSON.
- **`json.load()`**: membaca file JSON dan mengubahnya kembali menjadi object Python.

Kapan digunakan?

1. Menyimpan konfigurasi aplikasi.
2. Menyimpan pengaturan mesin atau sistem produksi.
3. Memindahkan data antar program.

Intinya:

dump/load adalah pasangan utama untuk pertukaran data Python dengan file JSON. Gunakan `indent=4` agar file mudah dibaca manusia. Kola.

Alur Simpan & Baca JSON



KELOLA FILE
KELOLA DATA
KELOLA MASA DEPAN

19

Pola Baca–Ubah–Simpan: Memperbarui Data Mesin

Data JSON tidak diubah langsung sebagian, tetapi dibaca menjadi object Python, dimodifikasi, lalu disimpan kembali.

mesin.json (sebelum) — JSON

```
{
  "CNC-01": {"lokasi": "Line A", "jam": 996, "status": "OK"},
  "CNC-02": {"lokasi": "Line B", "jam": 480, "status": "OK"}
}
```

mesin.json (sesudah) — JSON

```
{
  "CNC-01": {"lokasi": "Line A", "jam": 1004, "status": "PM"},
  "CNC-02": {"lokasi": "Line B", "jam": 488, "status": "OK"}
}
```

Kode Program Python — python

```python
import json
with open("mesin.json", encoding="utf-8") as f:
    mesin = json.load(f)

for kode, info in mesin.items():
    info["jam"] += 8
    if info["jam"] >= 1000:
        info["status"] = "PM"

with open("mesin.json", "w", encoding="utf-8") as f:
    json.dump(mesin, f, indent=2)
```

1. **Baca** — load file JSON menjadi dictionary Python
   `mesin = json.load(f)`
2. **Ubah** — proses data menggunakan loop dan kondisi if
   ```
   info["jam"] += 8
   if info["jam"] >= 1000:
       hifo["status"] = "PM"
   ```
3. **Simpan** — dump kembali dictionary ke file JSON
   `json.dump(mesin, f, indent=2)`

Intinya:
Pola baca-ubah-simpan adalah pola umum pengolahan file: ambil data → manipulasi → simpan hasil. Metode ini digunakan pada konfigurasi aplikasi, log mesin, dan data produksi.

Mengapa tidak edit langsung file?

- File JSON adalah teks, bukan database aktif.
- Cara aman adalah membaca seluruh struktur data ke memori.
- Setelah perubahan selesai, tulis ulang file agar struktur tetap valid.

Alur Baca – Ubah – Simpan JSON



KELOLA FILE
KELOLA DATA
KELOLA MASA DEPAN

20

Pola Pipeline Pengolahan Data Industri

Pengolahan data industri umumnya mengikuti alur berurutan: membaca data, memvalidasi, mengolah, lalu menghasiikan laporan.

BACA
open(), csv, json
Pertemuan 13

→

VALIDASI
try-except, raise
P12

→

OLAH
loop, if, dict, modul
P9-P12

→

TULIS
write, csv.writer, json.dump
P13

Data Mentah

Data awal dari sensor, mesin, dan sistem monitoring yang belum diproses.

Contoh sumber data:
- log_produksi.csv
- log_mesin.txt
- config.json

Pemeriksaan

Membersihkan data, mengecek kesalahan format, dan memastikan data siap digunakan.

Hal yang diperiksa:
- Cek format data
- Deteksi file kosong
- Tangani kesalahan input
- Pastikan tipe data sesuai

Perhitungan

Menghasilkan informasi seperti total, rata-rata, downtime, dan klasifikasi kualitas.

Contoh proses:
- Hitung total produksi
- Hitung rata-rata
- Klasifikasi kondisi mesin
- Analisis tren / statistik

Hasil

Output berupa laporan TXT, CSV untuk Excel, atau JSON untuk aplikasi lain.

Contoh output:
- laporan.txt (untuk supervisor)
- ringkasan.csv (untuk Excel)
- hasil.json (untuk aplikasi lain)
- dashboard / visualisasi

Intinya:

Hampir semua sistem pengolahan data industri mengikuti empat tahap utama: input → validasi → proses → output. Memahami pipeline membuat program lebih terstruktur, mudah diuji, dan mudah dikembangkan.

Alur Pipeline Pengolahan Data Industri



21

Contoh Analisis Pareto Cacat Produk dari Data CSV

Studi kasus: Data inspeksi kualitas produk pada lini produksi pabrik manufaktur.
Tujuan: mengidentifikasi jenis cacat utama menggunakan analisis Pareto dan memberikan prioritas perbaikan.



Kode Program Python

python

```
import csv
from collections import dedfodliict

# 1. Baca data dari file CSV
data = dedfodliict(int)
with open("inspeksi_produk.csv", newline="", encoding"utf-8") as f:
     reader = csv.DictReader(f)
     for row in reader:
         jenis = row["jenis_cacat"]
         jumlah = int(row["jumlah"])
         data(jenis) += jumlah
     )

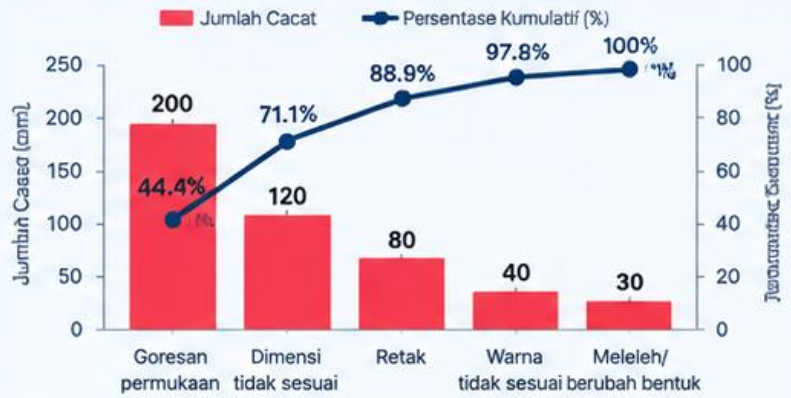
# 2. Urutkan berdaserkan jumlah (terbesar ke terkecil)
hasil = sorted(data.items(), key=lamba x: x[1], reverse"True)
total = sum(data.valles())
kum = 0

# 3. Tulis hasil ke CSV (opsional)
with open("hasil_pareto.csv", "w", newline="", encoding'"utf-8") as f:
    writer = csv.writer(f)
    writer.writerow(["Jenis Cacat", "Jumlah", "Persen", "Kumulatif %"])
    for jenis, jumlah in hasil:
      persen = jumlah / total * 100
      kum += persen
      writer.writerow([jenis, jumlah, round(persen,1), round(kum,1)]])

# 4. Tampilkan di layar (opsional)
for jenis, jumlah in hasil:
   persen = jumlah / total * 100
print(f"{jenis:22s}      jumlah:>6d    {persen:>6.1f%")
```

Diagram Pareto Cacat Produk

Gabungan grafik jumlah dan persentase kumulatif



Hasil Analisis Pareto (hasil_pareto.csv)

No	Jenis Cacat	Jumlah	Persen (%)	Kumulatif (%)
1	Goresan permukaan	200	44.4	44.4
2	Dimensi tidak sesuai	120	26.7	71.1
3	Retak	80	17.8	88.9
4	Warna tidak sesuai	40	8.9	97.8
5	Meleleh/berubah bentuk	30	6.7	100.0
	Total Total	450	100.0	-

Contoh Data CSV

inspeksi_produk.csv

id	tanggal	produk	jenis_cacat	jumlah
P001	2026-11-10	Gear Housing	Goresan permukaan	50
P002	2026-11-10	Gear Housing	Dimensi tidak sesuai	20
P003	2026-11-10	Gear Housing	Retak	10
P004	2026-11-11	Gear Housing	Goresan permukaan	30
P005	2026-11-11	Gear Housing	Warna tidak sesuai	10
P006	2026-11-12	Gear Housing	Dimensi tidak sesuai	40
P007	2026-11-12	Gear Housing	Retak	25
P008	2026-11-12	Gear Housing	Meleleh/berubah bentuk	15
...	...	...	... ...	...

Insight dari Hasil Analisis

44.4% cacat disebabkan oleh goresan permukaan.

- 71.1% cacat berasal dari dua jenis utama (goresan permukaan dan dimensi tidak sesuai).
- Fokus perbaikan dimulai dari dua jenis cacat utama (prinsip Pareto 80/20).
- Analisis ini membantu tim quality control dalam menentukan prioritas tindakan perbaikan.

Alur Proses Analisis Pareto

Data Inspeksi (inspeksi_produk.csv) → **Python Processing** → **Analisis Pareto** → **Hasil Analisis** → **Tindak Lanjut**

- Data Inspeksi: Data hasil inspeksi kualitas dari lini produksi
- Python Processing: Baca data, kelompokkan jenis cacat, hitung jumlah dan persentase
- Analisis Pareto: Urutkan data, hitung persentase kumulatif, buat diagram Pareto
- Hasil Analisis: Tabel hasil, diagram Pareto, dan insight untuk keputusan
- Tindak Lanjut: Fokus perbaikan pada jenis cacat utama

22

Contoh Analisis Downtime Mesin dari Data CSV

Studi kasus: Data log downtime mesin pada lini produksi pabrik manufaktur.
Tujuan: mengidentifikasi mesin dengan downtime tertinggi untuk menentukan prioritas perbaikan dan maintenance.



Kode Program Python

python

```python
import csv
from collections import dedfodiiict

# 1. Baca data dari file CSV
downtime = dedfodiiict(int)
with open("log_downtime_mesin.csv", newline="", encoding"utf-8") as f:
    reader = csv.DictReader(f)
    for row in reader:
        mesin = row["mesin"]
        durasi = int(row["durasi_menit"])
        downtime[mesin] += durasi
    )

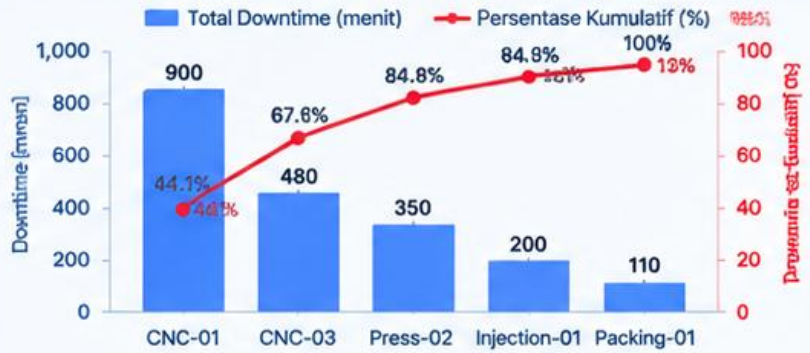
# 2. Urutkan berdaserkan total downtime (terbesar ke terkecil)
hasil = sorted(downtime.items(), key=lamba x: x[1], reverse"Tree)
total = sum(downtime.values())

# 3. Hitung persentase dan kumulatif
data_hasil = []
kumulatif = 0
for mesin, menit in hasil:
    persen = menit / total * 100
    kumulatif += persen
    data_hasil.append(mesin, menit, round(persen,1), round(kumulatif,1)))

# 4. Simpan hasil ke file CSV
with open("hasil_downtime.csv", "w", newline="", encoding"utf-8") as f:
    writer = csv.writer(f)
    writer.writerow(["Mesin", "Total Downtime (menit)", "Persen (%)", "Kumulatif (%)"])
    writer.writerows(data_hasit)
print("Analisis selesai. Hasil disimpan ke hasil_downtime.csv")
```

Diagram Pareto Downtime Mesin

Gabungan grafik total downtime dan persentase kumulatif



Hasil Analisis (hasil_downtime.csv)

No	Mesin	Total Downtime (menit)	Persen (%)	Kumulatif (%)
1	CNC-01	900	44.1	44.1
2	CNC-03	480	23.5	67.6
3	Press-02	350	17.2	84.8
4	Injection-01	200	9.8	94.6
5	Packing-01	110	5.4	100.0
	Total	2,040	100.0	-

Contoh Data CSV

log_downtime_mesin.csv

tanggal	mesin	durasi_menit	penyebab
2026-11-01	CNC-01	120	Overheat
2026-11-01	CNC-03	60	Sensor Error
2026-11-01	Press-02	45	Hidrolik Bocor
2026-11-02	CNC-01	180	Tool Change
2026-11-02	Injection-01	30	Listrik Padam
2026-11-03	CNC-03	120	Motor Fault
...	...	...	...

Insight dari Hasil Analisis

- 44.1% downtime terjadi pada mesin CNC-01.
- 67.6% downtime berasal dari dua mesin utama (CNC-01 dan CNC-03).
- Fokus perbaikan dimulai dari dua mesin tersebut (prinsip Pareto 80/20).
- Data dapat digunakan untuk penjadwalan preventive maintenance.

Manfaat untuk Industri

- Mengurangi waktu henti mesin.
- Meningkatkan produktivitas produksi.
- Menentukan prioritas maintenance berbasis data.
- Mendukung keputusan investasi peralatan.

Alur Proses Analisis Downtime Mesin

File Data Log Downtime (log_downtime_mesin.csv)
Data waktu henti mesin dari sistem monitoring

→ **Python Processing**
Baca data, kelompokkan per mesin, hitung total downtime dan persentase

→ **Analisis Pareto**
Urutkan data, hitung persentase kumulatif, buat diagram Pareto

→ **Hasil Laporan**
Tabel hasil + diagram (insight & rekomendasi)

→ **Tindak Lanjut**
Perbaikan mesin prioritas dan jadwal maintenance

24

Penanganan Error dan Validasi Data

Data dari file eksternal sering tidak sempurna. Kita perlu menangani error dan melakukan validasi agar program tidak berhenti ketika ada data yang salah, kosong, atau format tidak sesuai.

Kode Program Python

python

```python
import csv

def valid_int(value, default=0):
    try:
        return int(value)
    except (ValueError, TypeError):
        return default

def proses_data(file_name):
    data_bersih = []
    baris_salah = []

    with open(file_name, "r", encoding="utf-8") as f:
        reader = csv.DictReader(f)
        for i, row in enumerate(reader, start=2):
            try:
                tanggal = row.get("tanggal", "").strip()
                mesin = row.get("mesin", "").strip()
                durasi = valid_int(row.get("durasi_menit", 0))

                if not tanggal or not mesin:
                    raise ValueError("Kolom wajib kosong")

                data_bersih.append({
                    "tanggal": tanggal,
                    "mesin": mesin,
                })  "durasi_menit": durasi
            except Exception as e:
                baris_salah.append ({
                    "baris": i,
                    "data": str(row),
                    "error": str(e)
    return data_bersih, baris_salah

data_ok, data_error = proses_data("log_mesin.csv")
print(f"Data valid: {len(data_ok)} baris"
print(f"Data error: {len(data_erro} baris"
```

Jenis Error yang Umum Terjadi

FileNotFoundError	File tidak ditemukan atau nama file salah.
ValueError	Data tidak sesuai format (misalnya teks saat diharapkan angka).
KeyError	Nama kolom tidak ada di file.
TypeError	Nilai kosong / tipe data tidak sesuai.
UnicodeDecodeError	Masalah encoding file.

Teknik Validasi Data

1. **Cek kolom wajib**
   Pastikan kolom penting tidak kosong (tanggal, mesin, dll).
2. **Validasi tipe data**
   Ubah ke int, float, atau tanggal yang sesuai.
3. **Tangani nilai kosong**
   Beri default atau lewati baris tersebut.
4. **Bersihkan data**
   Gunakan strip() untuk menghapus spasi berlebih.
5. **Catat data bermasalah**
   Simpan baris yang error untuk diperbaiki nanti.

Intinya:

Gunakan try-except untuk menangani error, dan lakukan validasi data sebelum diolah. Dengan ini, program tetap berjalan meskipun ada data yang bermasalah.

Contoh Hasil

Data Valid (diproses)

tanggal	mesin	durasi_menit
2026-11-01	CNC-01	120
2026-11-01	CNC-03	60
2026-11-02	Press-02	45

Data Bermasalah (dicatat)

baris	data	error
5	{'mesin': '', ...}	Kolom wajib kosong
8	{'durasi_menit': 'abc', ...}	invalid literal for int()
12	{'mesin': None, ...}	Kolom wajib kosong

24

Penanganan Error dan Validasi Data

Data dari file eksternal sering tidak sempurna. Kita perlu menangani error dan melakukan validasi agar program tidak berhenti ketika ada data yang salah, kosong, atau format tidak sesuai.

Kode Program Python — python

```python
import csv
def to_int(value, default=0):
    """Mengonversi nilai ke integer dengan penanganan error."""
    try:
        return int(value)
    except (ValueError, TypeError):
        return default:

def proses_data(file_name):
    data_valid = []
    data_error = []
    with open(file_name, "r", encoding="utf-8") as f:
      reader = csv.DictReader(f)
      for i, row in enumerate(reader, start=2):  # nomor baris di file
          try:
              tanggal = row.get("tanggal", "").strip()
              mesin = row.get("mesin", "").strip()
              durasi = to_int(row.get("durasi_menit", ""))
              if not tanggal or not mesin:
                  raise ValueError("Kolom wajib kosong")
              data_valid.append({"tanggal": tanggal,
                                 "mesin": mesin,
                                 "durasi_menit": durasi})
          except Exception as e:
              data_error.append({"baris": i,
                                 "data": row,
                                 "error": str(e)})
      return data_valied, data_error
```

Intinya:

Gunakan **try–except** untuk menangani error, dan lakukan validasi data sebelum diolah. Dengan ini, program tetap berjalan meskipun ada data yang bermasalah.

Jenis Error yang Umum Terjadi

FileNotFoundError	File tidak ditenukan atau nama file salah.
ValueError	Data tidak sesuai format (contoh: teks saat diharapkan angka).
KeyError	Nama kolom tidak ada di file (header berbeda).
TypeError	Tipe data tidak sesuai (contoh: *None* atau nilai kosong).
UnicodeDecodeError	Masalah encoding file (baca dengan *encoding='utf-8'*).

Teknik Validasi Data

1. **Cek kolom wajib**
   Pastikan kolom penting tidak kosong (tanggal, mesin, durasi, dll).
2. **Validasi tipe data**
   Konversi data ke tipe yang sesuai (int, float, tanggal, dll).
3. **Validasi rentang nilai**
   Pastikan nilai dalam batas wajar (contoh: durasi > 0 dan < 1440).
4. **Tangani nilai kosong**
   Beri nilai default atau lewati baris tersebut.
5. **Bersihkan data**
   Gunakan ***.strip()*** untuk menghapus spasi dan karakter yang tidak dinginkan.

Contoh Data Valid (diproses)

tanggal	mesin	durasi_menit
2026-11-01	CNC-01	120
2026-11-01	CNC-03	60
2026-11-02	Press-02	45

Contoh Data Bermasalah (ditangani error)

baris	tanggal	mesin	durasi_menit	error
5		CNC-01	120	Kolom wajib kosong
8	2026-11-01		abc	invalid literal for int()
12	2026/11/03	CNC-03		Kolom wajib kosong

Alur Penanganan Error dan Validasi Data



KELOLA FILE
KELOLA DATA
KELOLA MASA DEPAN

Terima Kasih

Sampai jumpa di pertemuan berikutnya!



Analisis Data Industri dengan NumPy dan Pandas

Dari Data Mentah Produksi Menjadi Informasi dan Keputusan Berbasis Data



2 Capaian Pembelajaran Pertemuan Ini

SUB-CPMK

Mahasiswa mampu mengolah dan menganalisis dataset industri menggunakan library NumPy dan Pandas.

Level kognitif: C4 (Menganalisis)

Indikator Ketercapaian

- Membuat dan mengoperasikan array NumPy, termasuk array 2 dimensi.
- Memuat data ke DataFrame dan membersihkannya: nilai kosong, duplikat, tipe data.
- Melakukan filter, pengelompokan (groupby), dan agregasi.
- Menghitung statistik deskriptif dan menginterpretasikan hasilnya, misalnya OEE.

3 Peta Materi

1 NumPy

Array, operasi vektor, 2D, statistik, filter

Slide 4–8

2 Dasar Pandas

Series, DataFrame, membaca CSV

Slide 9–11

3 Mengenal Data

Eksplorasi dan diagnosis kualitas data

Slide 12–13

4 Membersihkan Data

Teks, tipe, kosong, duplikat, kolom baru

Slide 14–16

5 Menganalisis

Seleksi, urut, groupby, statistik deskriptif

Slide 17–20

6 Promax & Studi Kasus

Takt time, Pareto cacat, OEE

Slide 21–25

4 Mengapa NumPy dan Pandas?



PERTEMUAN 13: MANUAL

```python
rekap = {}
with open("log.csv", newline="") as f:
    for row in csv.DictReader(f):
        m = row["mesin"]
        rekap[m] = rekap.get(m, 0) + int(row("output"])
```

PERTEMUAN 14: PANDAS

```python
import pandas as pd
df = pd.read_csv("log.csv")
rekap = df.groupby("mesin")["output"].sum()
```

Cepat
Operasi pada jutaan data tanpa loop manual.

Ringkas
Satu baris menggantikan belasan baris kode.

Standar industri
Dipakai data analyst dan engineer di seluruh dunia.

Intinya: NumPy = mesin hitung angka, Pandas = spreadsheet yang bisa diprogram. Keduanya menjadi dasar analisis data industri modern.

5 NumPy Array: Berbeda dengan List

python

```python
import numpy as np

ct = [12.4, 11.8, 12.9, 12.1, 13.5]    # list biasa
print(ct * 2)            # list DIGANDAKAN

arr = np.array(ct)                      # array NumPy
print(arr * 2)           # setiap nilai DIKALI 2
print(arr.dtype, arr.shape)
```

OUTPUT

```
[12.4, 11.8, 12.9, 12.1, 13.5, 12.4, 11.8, 12.9, 12.1, 13.5]
[24.8 23.6 25.8 24.2 27. ]
float64 (5,)
```

Aspek	list	np.array
Tipe isi	Boleh campur	Satu tipe (dtype)
Operasi * 2	Menggandakan list	Mengalikan tiap nilai
Kecepatan	Lambat untuk data besar	Sangat cepat
Dimensi	List bersarang	1D, 2D, nD (shape)

Intinya: array NumPy melakukan operasi pada seluruh elemen sekaligus, tanpa loop.

6 Operasi Vektor: Menghitung Banyak Mesin Sekaligus

Kode Python

```python
import numpy as np

# CNC-01, CNC-02, CNC-03
A = np.array([0.95, 0.88, 0.97])   # availability
P = np.array([0.92, 0.85, 0.96])   # performance
Q = np.array([0.98, 0.96, 0.99])   # quality

oee = A * P * Q                    # tanpa loop!
print(oee.round(3))                # [0.857 0.718 0.922]
print(f"Rata-rata OEE: {oee.mean():.1%}")   # 83.2%

downtime_menit = np.array([90, 45, 120])
print(downtime_menit / 60)         # [1.5  0.75 2.  ] jam
```

Cara kerja A × P × Q

0.95	×	0.92	×	0.98	=	0.857
0.88	×	0.85	×	0.96	=	0.718
0.97	×	0.96	×	0.99	=	0.922

Materi Penjelasan

1. NumPy menghitung seluruh data sekaligus tanpa for.
2. A, P, dan Q berisi data 3 mesin pada posisi yang sejajar.
3. Perkalian array dilakukan per elemen: mesin ke-1 dengan ke-1, dst.
4. Rata-rata OEE dari tiga mesin adalah 83.2%.
5. Pembagian dengan satu angka berlaku ke semua elemen (broadcasting).

Intinya:
Operasi antar-array dilakukan per posisi (elemen ke-1 dengan ke-1, dst.). Operasi dengan satu angka berlaku ke semua elemen. Inilah kekuatan utama **vectorization** di NumPy.

CNC-02
OEE terendah

7 Array 2 Dimensi: Matriks Produksi Mesin × Hari

Kode Python

```python
prod = np.array([[520, 510, 530, 505, 515],    # CNC-01
                 [470, 455, 488, 462, 470],    # CNC-02
                 [505, 498, 512, 520, 501]])   # CNC-03

print(prod.shape)                  # (3, 5): 3 mesin, 5 hari
print(prod[1, 2])                  # 488: CNC-02 hari ke-3
print(prod[:, 0])                  # [520 470 505]: hari ke-1
print(prod.sum(axis=1))            # [2580 2345 2536]: per mesin
print(prod.sum(axis=0))            # [1495 1463 1530 1487 1486]
print(prod.mean(axis=1).round(1)) # [516.  469.  507.2]
```

Peta indeks [baris, kolom]

CNC-01	520	510	530	505	515
CNC-02	470	455	488	462	470
CNC-03	505	498	512	520	501

axis=1 → ↓ axis=0

Kuning = prod[1, 2] Abu = prod[:, 0] (semua baris, kolom 0)

Indeks dimulai dari 0, sama seperti list.

Materi Penjelasan

1. Array 2D berbentuk matriks: baris = mesin, kolom = hari produksi.
2. shape = (3, 5) artinya ada 3 mesin dan 5 hari pengamatan.
3. prod[1, 2] mengambil baris ke-2, kolom ke-3, yaitu 488.
4. axis=1 menghitung sepanjang baris (hasil per mesin), axis=0 sepanjang kolom (hasil per hari).
5. prod[:, 0] berarti semua baris pada kolom 0, yaitu output hari pertama dari semua mesin.

Intinya:

Array 2 dimensi sangat cocok untuk data industri yang berbentuk mesin × waktu.
Memahami indeks dan axis memudahkan kita menghitung total dan rata-rata tanpa loop manual.

8 Statistik dan Filter Boolean pada NumPy

Kode Python

```python
import numpy as np

ct = np.array([12.4, 11.8, 12.9, 12.1, 13.5,
               12.4, 11.9, 12.6, 12.2, 12.7])  # detik

print(ct.mean(), np.median(ct))        # 12.45 12.4
print(ct.std(ddof=1).round(3))         # 0.506 (sampel)
print(np.percentile(ct, 90).round(2))  # 12.96

lambat = ct[ct > 12.5]                 # filter boolean
print(lambat, (ct > 12.5).sum())       # [12.9 13.5 12.6 12.7] 4
print(np.where(ct > 12.5, "LAMBAT", "OK"))
```

OUTPUT

```
['OK' 'OK' 'LAMBAT' 'OK' 'LAMBAT' 'OK' 'OK' 'LAMBAT' 'OK' 'LAMBAT']
```

`mean()`	Rata-rata
`np.median()`	Nilai tengah
`std(ddof=1)`	Simpangan baku sampel
`np.percentile(x, 90)`	90% data di bawah nilai ini
`arr[kondisi]`	Ambil elemen yang True
`np.where(k, a, b)`	Ternary untuk array

Default std NumPy: ddof=0 (populasi).

Materi Penjelasan

1. Fungsi statistik NumPy bekerja sangat cepat untuk data numerik dalam array.
2. mean dan median membantu melihat pusat data, tetapi median lebih tahan terhadap nilai ekstrem.
3. std(ddof=1) dipakai untuk simpangan baku sampel agar sesuai konteks pengukuran data praktik.
4. Filter boolean ct > 12.5 menghasilkan array True/False, lalu dipakai untuk menyaring data lambat.
5. np.where memberi label otomatis untuk setiap elemen, misalnya OK atau LAMBAT.

Intinya: NumPy dapat menghitung statistik dan melakukan filter data tanpa loop manual. Dengan boolean filtering, kita bisa menemukan data yang memenuhi kondisi tertentu dengan cepat.

9 Pandas: Series dan DataFrame

- **DataFrame** = tabel dua dimensi, seperti satu sheet Excel.
- **Series** = satu kolom dari DataFrame (array 1D berlabel).
- **Index** = label baris, biasanya dimulai dari 0, 1, 2, ...
- Setiap kolom DataFrame adalah Series (array NumPy).
- Cocok untuk data industri: bisa menyimpan data produksi, mesin, waktu, cacat, dll.

Contoh membuat Series dan DataFrame

python

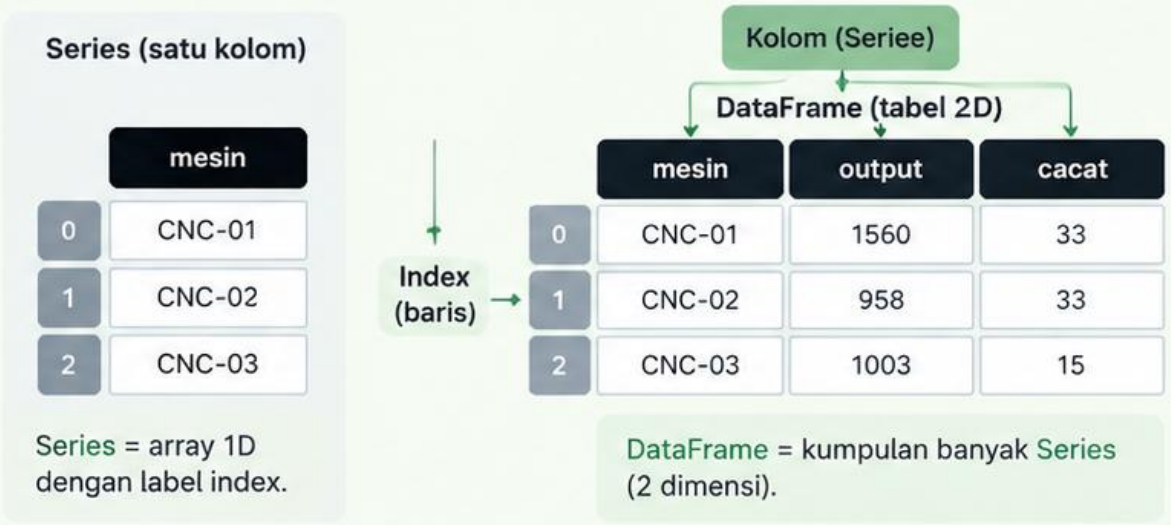
```python
import pandas as pd

# Membuat Series (satu kolom data)
mesin = pd.Series(["CNC-01", "CNC-02", "CNC-03"],
                  name="mesin")

# Membuat DataFrame dari beberapa Series(list
df = pd.DataFrame({
    "mesin":  ["CNC-01", "CNC-02", "CNC-03"],
    "output": [1560, 958, 1003],
    "cacat":  [33, 33, 15]
})

print(mesin)
print()
print(df)
```

Ilustrasi Series dan DataFrame



Output

```
0   CNC-01
1   CNC-02
2   CNC-03
Name: mesin, dtype: object

    mesin  output  cacat
0  CNC-01    1560     33
1  CNC-02     958     33
2  CNC-03    1003     15
```

Dari Mana Datanya?

DataFrame bisa dibuat manual (seperti contoh di atas) atau dibaca dari file CSV.

```python
df = pd.read_csv("produksi.csv")
```

File CSV akan dibahas lebih detail di slide berikutnya.

Intinya: Series untuk satu kolom data, DataFrame untuk tabel lengkap. Struktur ini memudahkan analisis, filter, aggregasi, dan visualisasi data industri.

10 Membuat DataFrame dan Kolom Baru

Kode Python

```python
import pandas as pd

data = {"mesin": ["CNC-01", "CNC-02", "CNC-03"],
        "output": [1560, 958, 1003],
        "cacat": [33, 33, 15]}

df = pd.DataFrame(data)

df["cacat_persen"] = df["cacat"] / df["output"] * 100
print(df.round(2))

s = df["output"]
print(s.max(), s.idxmax())
```

Output

```
    mesin   output   cacat   cacat_persen
0  CNC-01     1560      33           2.12
1  CNC-02      958      33           3.44
2  CNC-03     1003      15           1.50
   1560   0
```

Sumber DataFrame

dict of list (kolom demi kolom)

```
data = {"mesin": [...],
        "output": [...]}
```

list of dict (baris demi baris)

```
[{"mesin": "CNC-01", ...},
 {"mesin": "CNC-02", ...}]
```

pd.read_csv / read_excel (dari file data)

```
df = pd.read_csv("data.csv")
df = pd.read_excel("data.xlsx")
```

Ilustrasi DataFrame

Index	mesin	output	cacat	cacat_persen
0	CNC-01	1560	33	2.12
1	CNC-02	958	33	3.44
2	CNC-03	1003	15	1.50

Kolom baru hasil operasi

Materi Penjelasan

1. DataFrame menyimpan data dalam bentuk tabel dua dimensi, seperti satu sheet Excel.
2. Kolom baru dibuat dengan memberikan hasil operasi ke df["nama_kolom"].
3. Operasi otomatis diterapkan ke seluruh baris tanpa loop manual.
4. Series adalah satu kolom DataFrame yang dapat dianalisis dengan fungsi statistik.
5. Kolom turunan membantu membuat indikator industri seperti persentase cacat.

Intinya: Pandas memungkinkan kita mengubah data mentah menjadi informasi baru dengan operasi kolom yang sederhana dan efisien.

11 Membaca Dataset Produksi dari CSV



Kode Python

python

```python
import pandas as pd

df = pd.read_csv("produksi_see.csv")
print(df.shape)          # (31, 8) = 31 baris, 8 kolom
df.head(3)               # 3 baris pertama
```

Output (Contoh DataFrame)

	tanggal	mesin	shift	waktu_rencana	downtime	ct_ideal	output	cacat
0	2026-11-10	CNC-01	Pagi	450	31.0	50	457	3
1	2026-11-10	CNC-02	Pagi	450	25.0	55	385	16
2	2026-11-10	CNC-03	Pagi	450	15.0	45	556	10

Penjelasan Kolom Utama

Kolom	Arti
waktu_rencana	Waktu produksi terencana per shift (menit)
downtime	Waktu mesin berhenti (menit)
ct_ideal	Cycle time ideal (detik/unit)

Poin Penting

1. **read_csv()** mengubah file eksternal menjadi tabel DataFrame.
2. **shape** menunjukkan ukuran data (jumlah baris dan kolom).
3. **head()** digunakan untuk melihat contoh data sebelum analisis.
4. DataFrame menjadi dasar proses cleaning, filtering, dan analisis statistik.

Intinya: `read_csv()` membaca seluruh dataset CSV menjadi DataFrame.
Parameter `sep=';'` digunakan jika CSV Excel Indonesia memakai titik koma; `pd.read_excel()` digunakan untuk file .xlsx.

12 Eksplorasi Awal: Mengenali Isi Data

Kode Python python

```python
import pandas as pd

df.describe().round(1)          # statistik kolom angka
df["mesin"].value_counts()      # frekuensi setiap nilai
df["shift"].unique()            # daftar nilai unik
df.columns                      # nama semua kolom
```

Output: df.describe().round(1)

	downtime	ct_ideal	cacat
count	30.0	30.0	31.0
mean	32.0	50.2	9.2
std	20.3	4.2	3.8
min	3.0	45.0	1.0
50%	24.0	50.0	10.0
max	71.0	55.0	16.0

Output: df["mesin"].value_counts()

mesin	
CNC-01	10
CNC-03	10
CNC-02	10
cnc-02	1

Temuan mencurigakan

1. **count downtime 30, bukan 31:** ada nilai kosong (missing value).
2. **Kolom output tidak muncul di describe:** tipenya bukan angka (misalnya string).
3. **“cnc-02” tercatat sebagai mesin keempat:** ada perbedaan penulisan (huruf kecil).
4. **Total 31 baris: ada baris lebih** (mungkin duplikat atau data tambahan).

Materi Penjelasan

1. **describe()** melihat statistik dasar data numerik (*count, mean, std, min, 50%, max*).
2. **value_counts()** menemukan jumlah setiap kategori (misalnya jumlah data per mesin).
3. **unique()** digunakan untuk mencari variasi nilai (misalnya daftar shift yang ada).
4. Eksplorasi awal membantu menemukan data kosong, salah format, dan inkonsistensi.

Intinya: jangan langsung menghitung. Lakukan eksplorasi dulu untuk menemukan masalah data.



13 Diagnosis Kualitas Data: info; inna, duplicated

Kode Python python

```python
import pandas as pd

df.info()                                  # tipe & jumlah non-null
df.isna().sum()                            # nilai kosong per kolom
df.duplicated().sum()                      # jumlah baris duplikat
df[df["output"].str.contains("[a-z]")]  # teks aneh
```

Output: df.info() (ringkas)

```
#   Column          Non-Null   Dtype
3   waktu_rencana   31         int64
4   downtime        30         float64
5   ct_ideal        31         int64
6   output          31         object
7   cacat           31         int64
```

Output: Diagnosa Masalah

```
downtime  1
(isna)

duplikat  1

output "510"
baris 21
```

Masalah Data

No	Masalah	Bukti
1	Nama mesin tidak seragam	`"cnc-02"`
2	Output berupa teks	`Dtype object`
3	Nilai kosong	`downtime: 1`
4	Baris duplikat	`duplicated: 1`

Materi Penjelasan

1. `df.info()` digunakan untuk melihat struktur dataset, jumlah data, dan tipe setiap kolom.
2. `isna()` membantu menemukan nilai kosong sebelum proses analisis.
3. `duplicated()` mendeteksi baris yang tercatat lebih dari satu kali.
4. Tipe `object` pada kolom angka dapat menunjukkan data tercampur teks dan angka.
5. Diagnosis kualitas data dilakukan sebelum cleaning dan perhitungan statisttk.

Intinya: kualitas analisis bergantung pada kualitas data. Periksa struktur, missing value, tipe data, dan duplikasi sebelum mengolah dataset.

14 Membersihkan Data (1): Teks dan Tipe Data

Kode Python python

```python
# 1. Seragamkan nama mesin: buang spasi, huruf kapital
df["mesin"] = df["mesin"].str.strip().str.upper()
print(df["mesin"].value_counts().to_dict())

# 2. Ubah teks ke angka; yang gagal menjadi NaN
df["output"] = pd.to_numeric(df["output"], errors="coerce")
print(df["output"].isna().sum())

# 3. Ubah teks tanggal ke tipe tanggal
df["tanggal"] = pd.to_datetime(df["tanggal"])
```

Fungsi Utama yang Digunakan

`.str.strip()`	Buang spasi di tepi
`.str.upper()`	Huruf kapital
`.str.replace(a, b)`	Ganti teks
`pd.to_numeric()`	Teks → angka
`pd.to_datetime()`	Teks → tanggal
`.astype(int)`	Ubah tipe kolom

Materi Penjelasan

1. Data industri sering memiliki penulisan tidak konsisten sehingva perlu distandarkan.
2. Fungsi .str digunakan untuk melakukan operasi pada kolom teks secara langsung.
3. errors="coerce" mengubah data angka yang gagal dibaca menjadi NaN agar dapat diperiksa.
4. Konversi tipe data penting agar operasi matematika dan statistik berjalan benar.
5. Data bersih menjadi dasar untuk filtering, agregasi, dan analisis lanjutan.

Intinya: sebelum melakukan analisis, pastikan teks seragam dan tipe data sesuai. Cleaning data mencegah kesalahan hasii perhitungan.



15 Membersihkan Data (2): Duplikat dan Nilai Kosong

Kode Python

python

```python
df = df.drop_duplicates()              # 31 -> 30 baris

# isi downtime kosong dengan median MESIN YANG SAMA
med = df.grouby("mesin")["downtime"].transform("median")
df["downtime"] = df["downtime"].fillna(med)   # CNC-01 -> 24.0

df = df.dropna(subset=["output"])      # buang baris output rusak
df["output"] = df["output"].astype(int)
print(len(df), df.isna).sum).sum())    # 29 0
```

Kenapa median per mesin?

Median seluruh data dapat tercampur antar mesin. Median per mesin lebih mewakili kondisi mesin itu sendiri dan lebih tahan terhadap nilai ekstrem.

Intinya

Setiap keputusan pembersihan harus dapat dipertanggungjawabkan dan dicatat di laporan.

Strategi Nilai Kosong

Strategi	Kapan dipakai
`dropna()`	Nilai inti hilang dan tidak bisa ditebak (output)
`fillna(0)`	Kosong memang berarti nol
`fillna(median)`	Perkiraan wajar untuk data pendukung

Materi Penjelasan

1. drop_duplicates() menghapus data yang tercatat lebih dari satu kali.
2. fillna() mengganti nilai kosong dengan aturan yang dipilih.
3. Median per kelompok menjaga karakteristik tiap mesin.
4. dropna(subset) digunakan hanya pada kolom penting.
5. Cleaning harus memiliki alasan bisnis dan teknis.



16 Menambah Kolom Hasil Perhitungan

Kode Python

```python
import numpy as np

df["waktu_operasi"] = df["waktu_rencana"] - df["downtime"]
df["cacat_persen"] = df["cacat"] / df["output"] * 100
df["status"] = np.where(df["cacat_persen"] > 2.5, "WASPADA", "OK")

df[["mesin", "shift", "output", "cacat_persen", "status"]].head(4)
df["status"].value_counts()
```

Output (Contoh DataFrame)

	mesin	shift	output	cacat_persen	status
0	CNC-01	Pagi	457	0.66	OK
1	CNC-02	Pagi	386	4.15	WASPADA
2	CNC-03	Pagi	556	1.80	OK
3	CNC-01	Siang	511	0.98	OK

Bandingkan dengan P10

Dulu: per baris (if ... else ...)

```
if df["cacat_persen"] > 2.5:
    status = "WASPADA"
else:
    status = "OK"
```

Sekarang: satu baris dengan np.where()

```
df["status"] = np.where(
    df["cacat_persen"] > 2.5,
    "WASPADA", "OK"
)
```

Materi Penjelasan

1. Kolom baru dibuat dari hasil operasi kolom yang sudah ada.
2. Pandas melakukan perhitungan secara vectorized tanpa loop manual.
3. **np.where()** memberi label kondisi secara otomatis.
4. Kolom turunan membantu analisis performa mesin.
5. DataFrame menjadi lebih informatif setelah ditambahkan indikator baru.

Intinya: Kolom baru adalah hasil transformasi data. Dengan operasi antar kolom dan **np.where()**, analisis kondisi dapat dilakukan secara cepat untuk seluruh dataset.

17 Memilih Data: Kolom, Baris, dan Filter



Kode Python

python

```python
# filter satu kondisi + pilih kolom
df.loc[df["mesin"] == "CNC-02", ["tanggal", "shift", "output"]]

# beberapa kondisi: & (dan), | (atau), wajjb pakai kurung
f = df[(df["mesin"] == "CNC-02") & (df["downtime"] > 60)]
print(f[["tanggal", "shift", "downtime"]])

df.iloc[0:2, 1:4]   # berdasarkan posisi
```

Output

	tanggal	shift	downtime
10	2026-11-11	Siang	62.0
13	2026-12-12	Pagi	71.0
17	2026-12-12	Siang	61.0

Cara Memilih Data

`df["kolom"]`	Satu kolom (Series)
`df[["a", "b"]]`	Beberapa kolom
`df[kondisi]`	Baris yang memenuhi syarat
`df.loc[baris, kolom]`	Berdasarkan label
`df.iloc[i, j]`	Berdasarkan posisi indeks

Materi Penjelasan

1. Pemilihan kolom digunakan untuk mengambil variabel tertentu dalam DataFrame.
2. Filter kondisi digunakan untuk mencari data sesuai kriteria analisis.
3. Operator & dan | digunakan untuk menggabungkan beberapa kondisi.
4. Setiap kondisi harus menggunakan tanda kurung agar terbaca benar oleh Python.
5. loc cocok untuk label, sedangkan iloc cocok untuk posisi indeks.

Intinya: Pandas memungkindan pemilihan data secara fleksibel. Dengan filter dan indexing yang tepat, analisis dapat fokus pada bagian data yang relevan.



18 Mengurutkan Data dan Mencari Top-N

Kode Python

python

```python
# 3 shift dengan downtime terlama
df.sort_values("downtime", ascending=False).head(3)

# 3 shift dengan persentase cacat tertinggi
df.nlargest(3, "cacat_persen")[["mesin", "shift", "cacat_persen"]]

# urut berdasarkan beberapa kolom
df.sort_values(["mesin", "tanggal"])
```

Output: 3 downtime terlama

	tanggal	mesin	shift	downtime
12	2026-11-12	CNC-02	Pagi	71.0
10	2026-11-11	CNC-02	Siang	62.0
17	2026-11-12	CNC-02	Siang	61.0

Output: 3 cacat persen tertinggi

	mesin	shift	cacat_persen
13	CNC-02	Pagi	4.36
1	CNC-02	Pagi	4.15
20	CNC-02	Pagi	3.81

Pola yang muncul

Top-3 downtime dan top-3 cacat semuanya CNC-02. Cacat tertinggi juga selalu di shift Pagi. Ini menjadi petunjuk awal untuk investigasi akar masalah.

Materi Penjelasan

1. **sort_values()** digunakan untuk mengurutkan data berdasarkan satu atau beberapa kolom.
2. **ascending=False** membuat urutan dari nilai terbesar ke terkecil.
3. **head(n)** mengambil n data teratas setelah sorting.
4. **nlargest()** langsung mengambil nilai terbesar tanpa sorting manual.
5. Analisis Top-N membantu menemukan mesin atau kondisi yang paling membutuhkan perhatian.

Intinya: sort_values() untuk mengurutkan, **nlargest/nsmallest** untuk mengambil N data teratas atau terbawah.
Metode ini membantu menemukan prioritas masalah dalam dataset industri.



19 groupby dan agg: Rekap per Kelompok

Kode Python

python

```python
df.groupby("mesin")["output"].sum()

rekap = df.groupby("mesin").agg(
    total_output=("output", "sum"),
    rata_output=("output", "mean"),
    total_cacat=("cacat", "sum"),
    downtime_max=("downtime", "max")
).round(1)

df.groupby(["mesin", "shift"])["output"].mean()
rekap.to_csv("rekap_mesin.csv")
```

Output (DataFrame hasil rekap)

	mesin	total_output	rata_output	total_cacat	downtime_max
0	CNC-01	4814	481.4	61	37.0
1	CNC-02	3638	363.8	124	71.0
2	CNC-03	4932	548.0	75	23.0

Pola split-apply-combine



Materi Penjelasan

1. **groupby()** membagi data menjadi kelompok berdasarkan kategori tertentu.
2. **agg()** memungkinkan banyak perhitungan sekaligus dalam satu proses.
3. Fungsi **sum**, **mean**, dan **max** menghaslikan indikator ringkasan berbeda.
4. Hasil agregasi lebih mudah digunakan untuk evaluasi performa mesin.
5. **groupby** menggantikan loop manual untuk analisis dataset besar.

Intinya: **groupby** dan **agg** mengubah data mentah menjadi ringkasan informasi yang siap dianalisis untuk pengambilan keputusan industri.

20

Terima Kasih

TELAH MENGIKUTI PEMBAHASAN

Semoga materi ini membantu memahami pengolahan dan analisis data menggunakan **NumPy & Pandas**, serta dapat langsung diterapkan dalam studi maupun pekerjaan nyata.



Data
Menjadi Informasi

Informasi
Menjadi Pemahaman

Pemahaman
Menjadi Keputusan

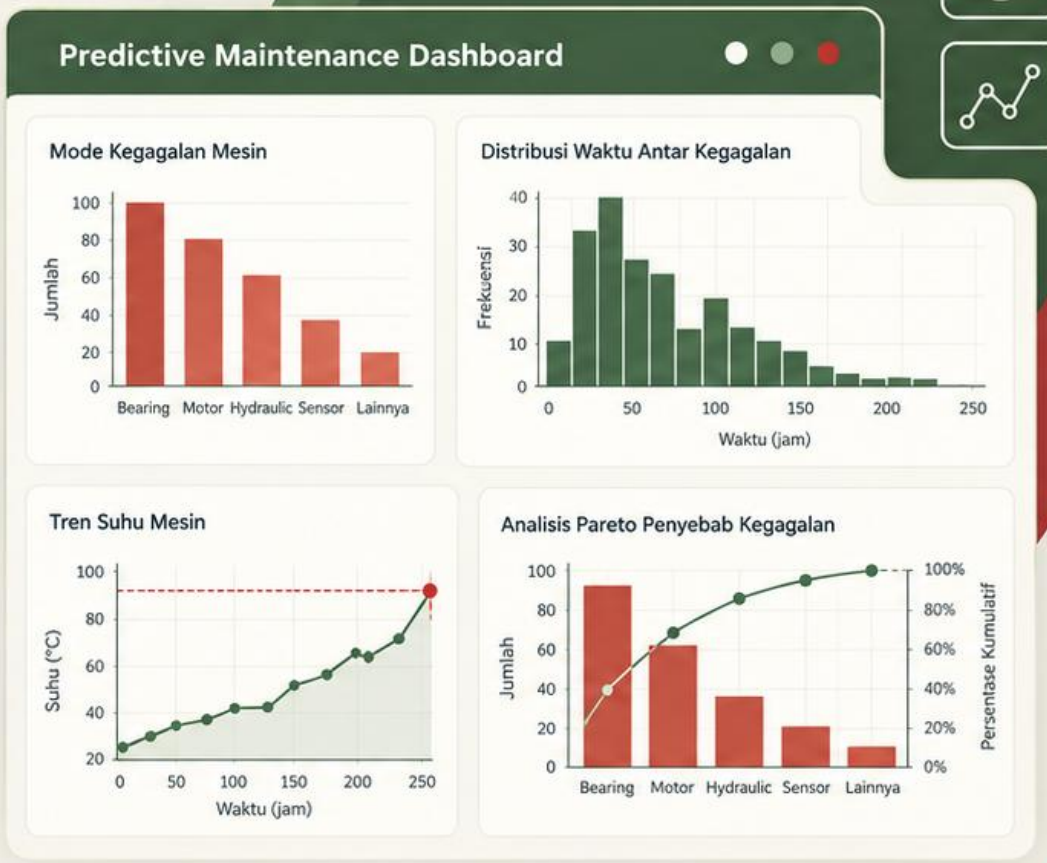
Keputusan
Memberi Dampak Nyata

PERTEMUAN 15 • PEMROGRAMAN KOMPUTER

Analisis dan Visualisasi Data Teknik Industri dengan Dataset Kaggle

Dari dataset nyata ke grafik, wawasan, dan rekomendasi perawatan mesin

Kaggle • pandas • matplotlib • EDA • Pareto • dashboard



Dashboard yang akan dibuat hari ini

Rujukan utama: W3Schools Matplotlib Tutorial & Pandas Tutorial; kaggle.com/datasets

2 Capaian Pembelajaran Pertemuan Ini

SUB-CPMK

Mahasiswa mampu memperoleh, mengeksplorasi, menganalisis, dan memvisualisasikan dataset industri nyata dari Kaggle untuk menghasilkan rekomendasi berbasis data.

Level kognitif: C4–C6 (Menganalisis, Mengevaluasi, Mencipta)

Indikator Ketercapaian

- Mencari, memilih, dan mengunduh dataset Kaggle serta memahami metadata dan lisensinya.
- Memuat dan membersihkan dataset dengan Pandas.
- Melakukan EDA dengan statistik deskriptif dan groupby.
- Membuat line, bar, histogram, scatter, pie, subplot, dan diagram Pareto.
- Menarik kesimpulan dan rekomendasi perbaikan dari hasil analisis.

3 Peta Materi

1 Kaggle

Platform, memilih dataset, cara unduh, metadata

2 Dasar Matplotlib

Anatomi grafik, plot, label, subplot

3 Jenis Grafik

Bar, histogram, scatter, pie, Pareto

4 EDA Dataset Kaggle

Muat, statistik, boxplot, menarik kesimpulan

5 Studi Kasus

Rekayasa fitur, dashboard perawatan

6 Proyek Kelompok

Tugas EDA, proposal & rubrik UAS

4

Mengenal Kaggle: Gudang Dataset dan Komunitas Data

Kaggle adalah platform komunitas data science milik Google. Di sini kita bisa mencari dataset, belajar dari notebook orang lain, dan mencoba tantangan analisis data.

Bagi mahasiswa Teknik Industri, Kaggle bermanfaat untuk menemukan data nyata tentang manufaktur, kualitas, perawatan mesin, rantai pasok, energi, dan topik industri lainnya sebagai bahan latihan analisis.

Jadi, Kaggle bukan hanya tempat download data, tetapi juga tempat belajar bagaimana data diolah menjadi informasi dan keputusan.

Intinya: Kaggle adalah tempat mencari data nyata, melihat contoh analisis, lalu mencoba mengolah data sendiri. Cocok untuk belajar dari dasar sampai membuat mini project.

Datasets
Ratusan ribu dataset publik yang siap dipakai untuk latihan dan proyek.

Code (Notebooks)
Contoh analisis dari pengguna lain yang bisa dibaca, dipelajari, dan dimodifikasi.

Competitions
Ajang latihan pemecahan masalah data dengan studi kasus nyata.

Learn
Materi singkat gratis untuk Python, Pandas, visualisasi, dan machine learning dasar.

5 Mencari dan Memilih Dataset yang Tepat

Kata kunci untuk Teknik Industri

- manufacturing
- predictive maintenance
- quality control
- supply chain
- inventory
- production line

Filter di halaman Datasets: jenis file (CSV), ukuran, lisensi, dan urutkan berdasarkan Hotness, Most Votes, atau Usability.

Kriteria	Yang diperiksa
Relevan	Sesuai masalah Teknik Industri yang ingin dianalisis
Dokumentasi	Ada penjelasan kolom, satuan, dan sumber data
Lisensi	Boleh dipakai untuk pendidikan (CC0, CC BY, dll.)
Ukuran	Cukup besar untuk dianalisis, tetapi muat di Colab
Usability	Skor kelengkapan dari Kaggle; makin tinggi makin baik

Intinya: dataset yang baik bukan yang terbesar, tetapi yang relevan, terdokumentasi, dan sah dipakai.

6 Tiga Cara Membawa Dataset Kaggle ke Notebook

1 Unduh manual

Buka halaman dataset → klik Download → ekstrak file ZIP → unggah CSV ke panel Files di Google Colab.

Paling mudah untuk pemula

2 kagglehub (kode)

Library resmi Kaggle; dataset diunduh langsung dari notebook.

Praktis & bisa diulang

3 Kaggle Notebook

Klik New Notebook di halaman dataset; data otomatis tersedia di folder /kaggle/input/.

Tanpa unduh sama sekali

Contoh kode mengambil dataset dengan kagglehub — python

```python
import kagglehub, os
path = kagglehub.dataset_download(
    "stephanmatzka/predictive-maintenance-dataset-ai4i-2020"
)
print(os.listdir(path))        # lihat isi folder
df = pd.read_csv(os.path.join(path, "ai4i2020.csv"))
```

Intinya:

handle dataset (pemilik/nama) terlihat di URL halaman Kaggle. Beberapa dataset memerlukan login atau API token.

7 Memahami Metadata Dataset AI4I 2020

Kolom	Arti
`Type`	Varian kualitas produk: L, M, H
`Air temperature [K]`	Suhu udara lingkungan (Kelvin)
`Process temperature [K]`	Suhu proses (Kelvin)
`Rotational speed [rpm]`	Kecepatan putar spindle
`Torque [Nm]`	Torsi
`Tool wear [min]`	Lama pemakaian tool (menit)
`Machine failure`	**1 = mesin gagal, 0 = normal**

10.000 baris • 14 kolom • data sintetis yang meniru kondisi industri nyata

5 MODE KEGAGALAN

- **TWF** — Tool wear failure: tool aus
- **HDF** — Heat dissipation: panas tak terbuang
- **PWF** — Power failure: daya terlalu rendah/tinggi
- **OSF** — Overstrain: beban berlebih
- **RNF** — Random failure: acak

Intinya: baca metadata sebelum menganalisis: arti kolom, satuan, dan cara data dibuat.

Kalau metadata dipahami, kita tidak salah menafsirkan angka.

8 Mengenal Matplotlib: Anatomi Sebuah Grafik

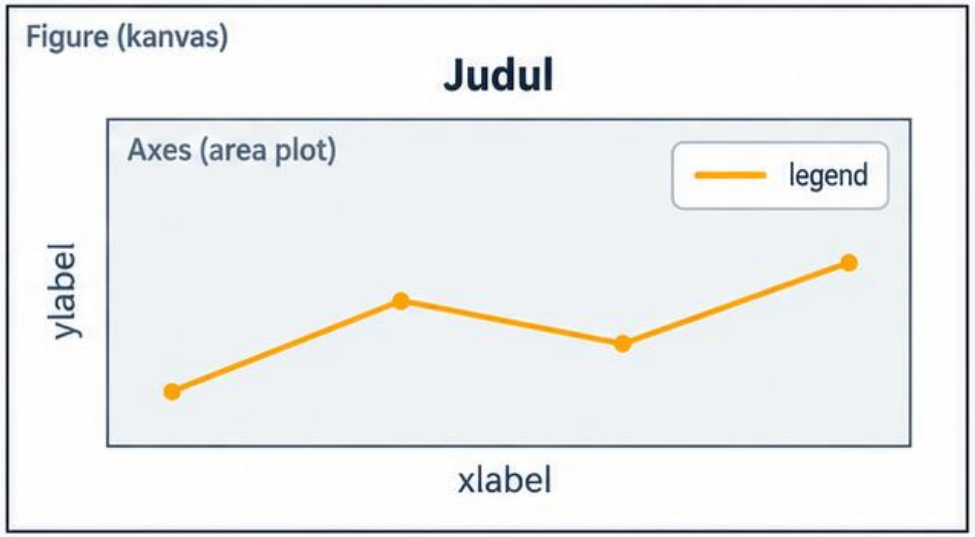
```python
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y = [10, 15, 13, 18]

plt.plot(x, y, label='Produksi')   # 1. gambar data
plt.title('Contoh Grafik')         # 2. judul
plt.xlabel('Hari')                 # 3. label sumbu X
plt.ylabel('Jumlah')               # 4. label sumbu Y
plt.legend()                       # 5. keterangan
plt.show()                         # 6. tampilkan grafik
```

Intinya: Matplotlib bekerja seperti melukis bertahap. Kita mullai dari data, lalu menambahkan judul, label, dan legenda. Perintah show() dipakai untuk menampilkan grafik akhirnya.

Bagian-bagian grafik



Konvensi: `import matplotlib.pyplot as plt`

Kalau memahami bagian grafik, kita akan lebih mudah membaca dan membuat visualisasi yang rapi.

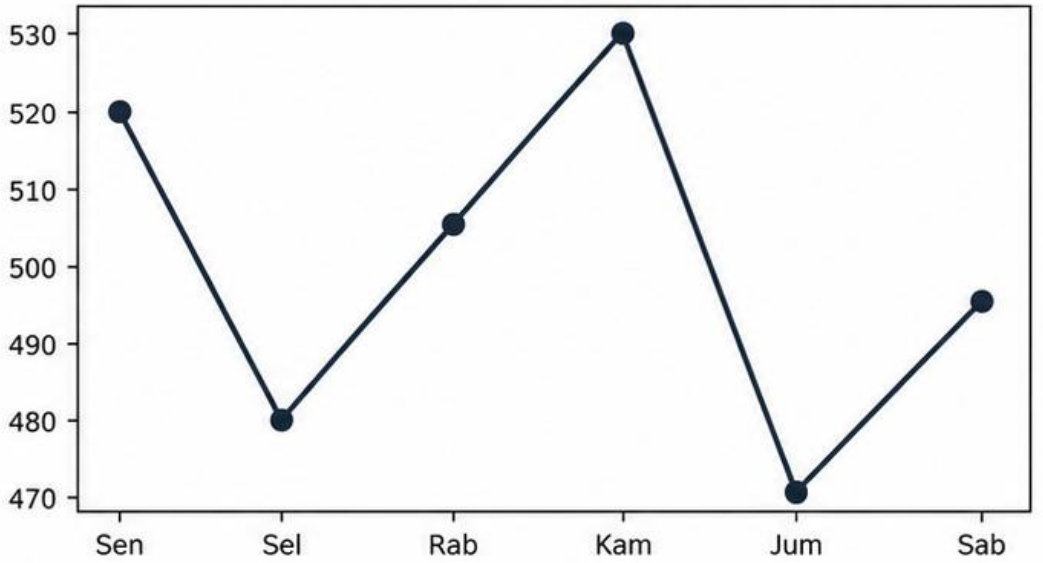
9 Grafik Garis dengan plot(): Marker, Garis, dan Warna

```python
import matplotlib.pyplot as plt

hari = ["Sen", "Sel", "Rab", "Kam", "Jum", "Sab"]
output = [520, 480, 505, 530, 470, 495]

plt.plot(hari, output, marker='o', linestyle='-',
         color="#1F2A36', linewidth=2)
plt.show()
```

Parameter	Contoh nilai
marker	"o" lingkaran, "s" kotak, "^" segitiga
linestyle	"-" utuh, "--" putus, ":" titik
color	"red", "#1F2A36"
linewidth	2



Intinya: grafik garis cocok untuk data yang berurutan, terutama waktu atau tren. Marker membantu menandai tiap titik data, garis menghubungkan perubahan, dan warna membantu membedakan grafik.

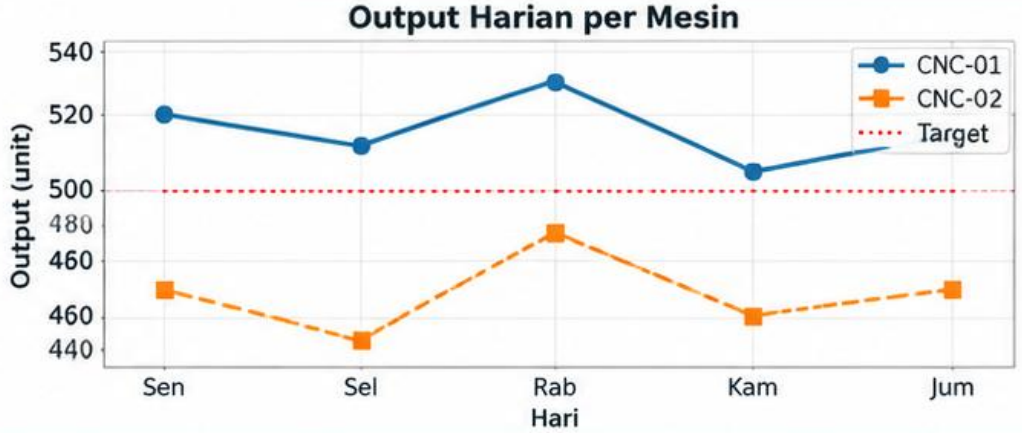
Gunakan grafik garis saat ingin melihat naik-turun nilai dari hari ke hari, bulan ke bulan, atau urutan proses.

10 Grafik yang Komunikatif: Judul, Label, Grid, Legend

```python
import matplotılib.pyplot as plt

hari = ["Sen", "Sel", "Rab", "Kam", "Jum"]
cnc01 = [520, 510, 530, 505, 515]
cnc02 = [470, 455, 488, 462, 470]

plt.figure(figsize=(7,4))
plt.plot(hari, cnc01, marker="o", label="CNC-01")
plt.plot(hari, cnc02, marker="s", linstyle="--",
   label="CNC-02")
plt.axhline(500, color="red", linestyle=":", label="Target")
plt.title("Output Harian per Mesin")
plt.xlabel("Hari")
plt.ylabel("Output (unit)")
plt.grid(alpha=0.3)
plt.legend()
plt.show()
```



Elemen	Fungsi / Penjelasan
title()	Memberikan judul grafik (menjelaskan isi grafik).
xlabel(), ylabel()	Label sumbu X dan Y, sertakan satuan data.
legend() + label	Membedakan beberapa garis/seri data.
axhline()	Menambahkan garis acuan (misal target/batas).
savefig()	Menyimpan grafik ke file untuk laporan.

Intinya: grafik yang baik bukan hanya menampilkan data, tetapi membantu pembaca memahami pesan utama. **Judul, label, grid,** dan legenda membuat grafik lebih mudah dibaca dan tidak menimbulkan salah interpretasi.

Catatan pembelajaran: Gunakan elemen tambahan seperlunya: terlalu banyak warna dan dekorasi dapat mengurangi fokus analisis.

11

Subplot Matplotlib: Menampilkan Beberapa Grafik Sekaligus

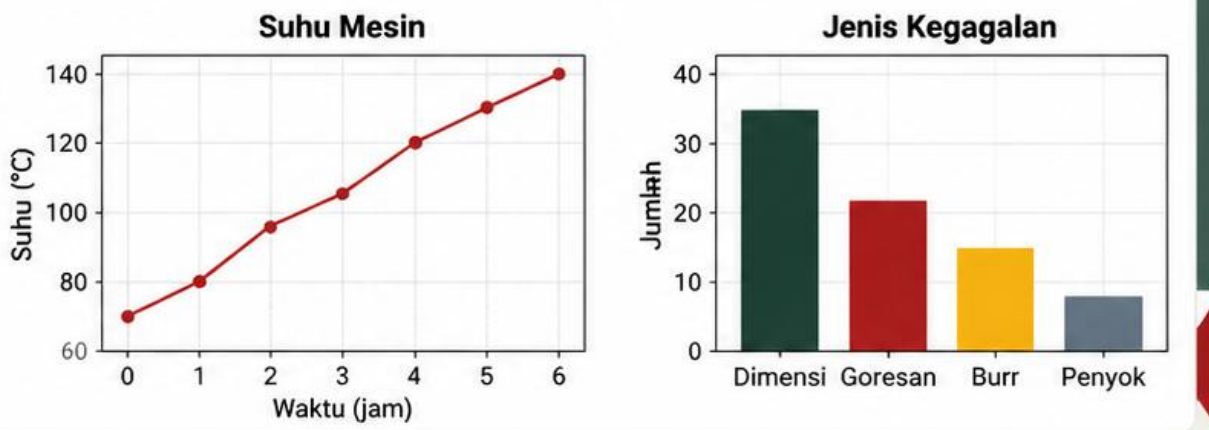
```python
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 2, figsize=(8,3))

ax[0].plot(waktu, suhu)
ax[0].set_title('Suhu Mesin')

ax[1].bar(jenis, jumlah)
ax[1].set_title('Jenis Kegagalan')

plt.tight_layout()
plt.show()
```



Elemen	Fungsi
plt.subplots()	Membuat beberapa area grafik
ax[0], ax[1]	Mengakses masing-masing grafik
set_title()	Memberi judul grafik
tight_layout()	Merapikan jarak antar grafik

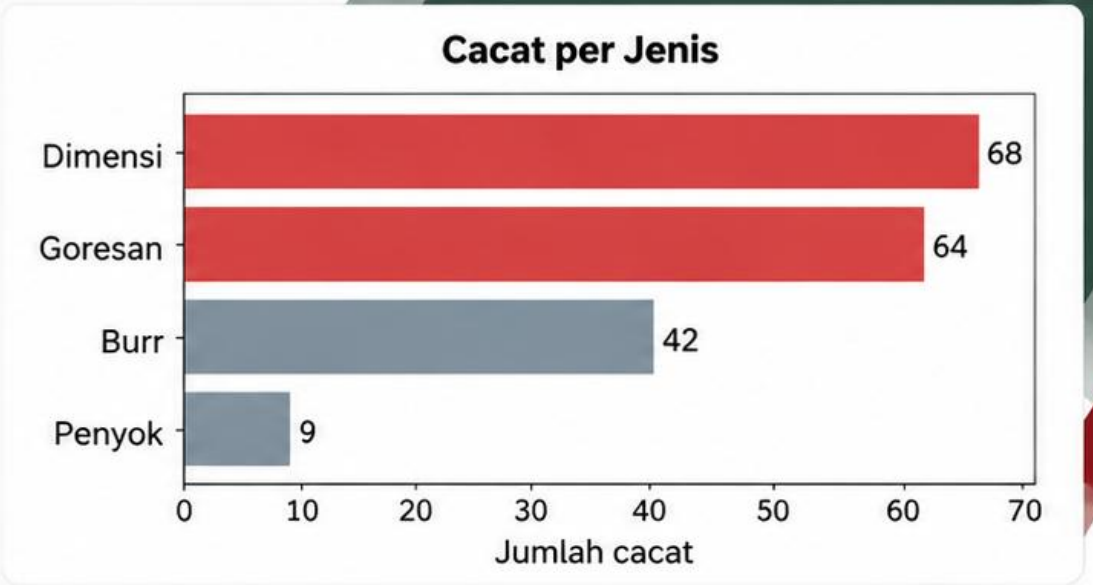
Intinya: subplot digunakan ketika kita ingin membandingkan beberapa informasi dalam satu tampilan. Dalam analisis industri, misalnya membandingkan tren suhu mesin dan jenis kegagalan secara bersamaan.

12 Grafik Batang: Membandingkan Kategori

```python
import matplotlib.pyplot as plt

jenis = ["Dimensi", "Goresan", "Burr", "Penyok"]
jumlah = [68, 64, 42, 9]
warna = ["#D64545" if n > 50 else "#7C8B99" for n in jumlah]

fig, ax = plt.subplots(figsize=(6, 3.5))
batang = ax.barh(jenis, jumlah, color=warna)
ax.bar_label(batang)
ax.invert_yaxis()
ax.set_xlabel("Jumlah cacat")
ax.set_title("Cacat per Jenis")
plt.show()
```



bar() digunakan untuk membuat batang vertikal, sedangkan **barh()** membuat batang horizontal. Grafik batang cocok untuk membandingkan nilai antar kategori.

Parameter	Fungsi
bar() / barh()	Membuat grafik batang
bar_label()	Menampilkan nilai pada batang
invert_yaxis()	Mengurutkan kategori dari atas
set_xlabel()	Memberi nama sumbu

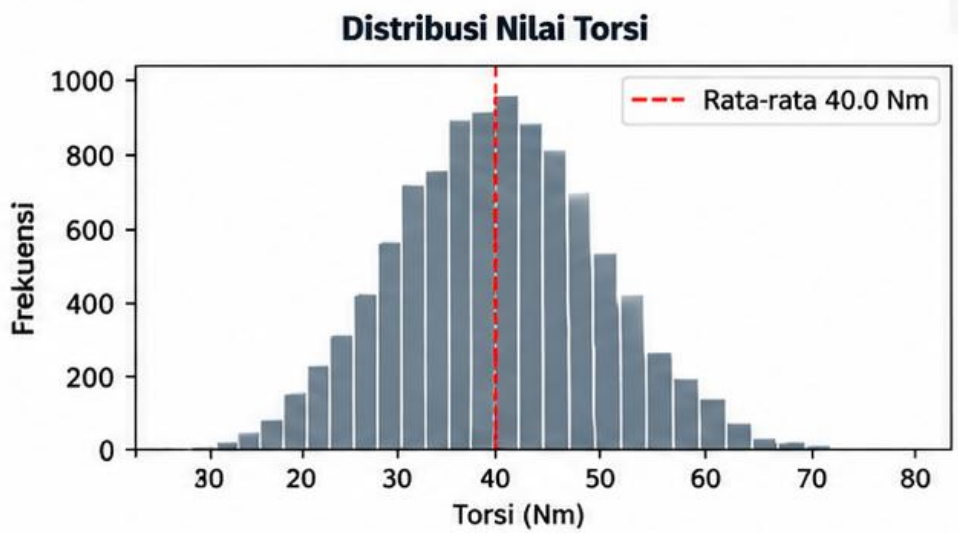
Intinya: urutkan kategori agar mudah dibandingkan, gunakan warna hanya untuk menekankan informasi penting, dan tampilkan nilai agar pembaca cepat memahami data.

13 Histogram: Melihat Distribusi Data

```python
import matplotlib.pyplot as plt

torsi = df["Torque [Nm]"]

plt.hist(torsi, bins=30, color="#7C8B99", edgeclor="white")
plt.axvline(torsi.mean(), color="red", linestyle="--",
            label=f"Rata-rata {torsi.mean():.1f} Nm")
plt.xlabel("Torsi (Nm)")
plt.ylabel("Frekuensi")
plt.legend()
plt.show()
```



Membaca histogram

Histogram digunakan untuk melihat bagaimana data tersebar. Dari bentuk grafik kita dapat mengetahui apakah data terkonsentrasi, menyebar, atau memiliki nilai yang tidak biasa.

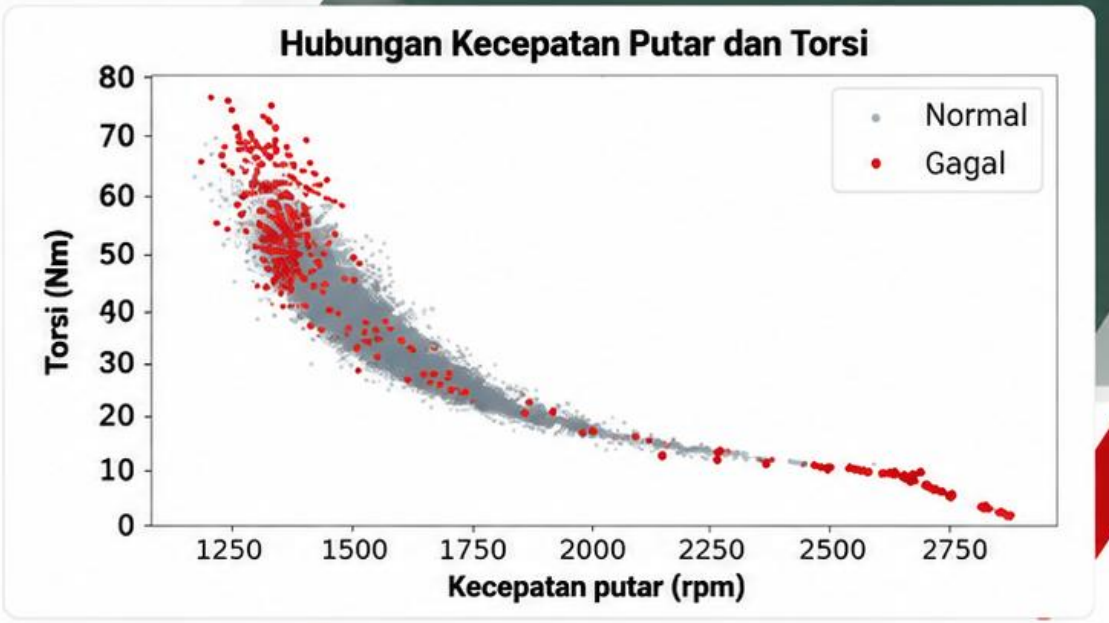
Aspek	Penjelasan
Sumbu X	Menunjukkan rentang nilai data (bin)
Sumbu Y	Menunjukkan jumlah data pada setiap rentang
Bentuk distribusi	Membantu melihat pola data

Intinya: histogram menjawab pertanyaan: di mana data terkonsentrasi dan seberapa menyebar? Jumlah bin menentukan tingkat detail distribusi.

14 Scatter Plot: Mencari Hubungan Dua Variabel

```python
normal = df[df['Machine failure'] == 0]
gagal = df[df['Machine failure'] == 1]

plt.scatter(normal['Rotational speed [rpm]''],
            normal['Torque [Nm]'],
            s=4, alpha=0.3, color='#7C8B99', label='Normal')
plt.scatter(gagal['Rotational speed [rpm]'],
            gagal['Torque (Nm)'],
            s=10, color='#D64545'', label='Gagal')
plt.xlabel('Kecepatan putar (rpm)')
plt.ylabel('Torsi (Nm)')
plt.legend()
plt.show()
```



Membaca scatter plot

Setiap titik mewakili satu data (satu mesin). Posisi titik menunjukkan nilai kecepatan putar (rpm) pada sumbu X dan torsi (Nm) pada sumbu Y. Pola titik yang menurun menunjukkan hubungan negatif: semakin tinggi kecepatan putar, semakin rendah torsi. Warna membedakan kategori normal dan gagal.

Parameter	Fungsi
plt.scatter(x, y)	Membuat scatter plot (titik).
s	Ukuran titik (marker size).
alpha	Transparansi titik (0 = transparan, 1 = solid).
color	Warna titik (misal: **"#7C8B99"**, **"#D64545"**).
label	Nama kategori untuk legenda.

Intinya: scatter plot digunakan untuk melihat hubungan antara dua variabel dan membandingkan kelompok (misalnya mesin normal vs gagal).

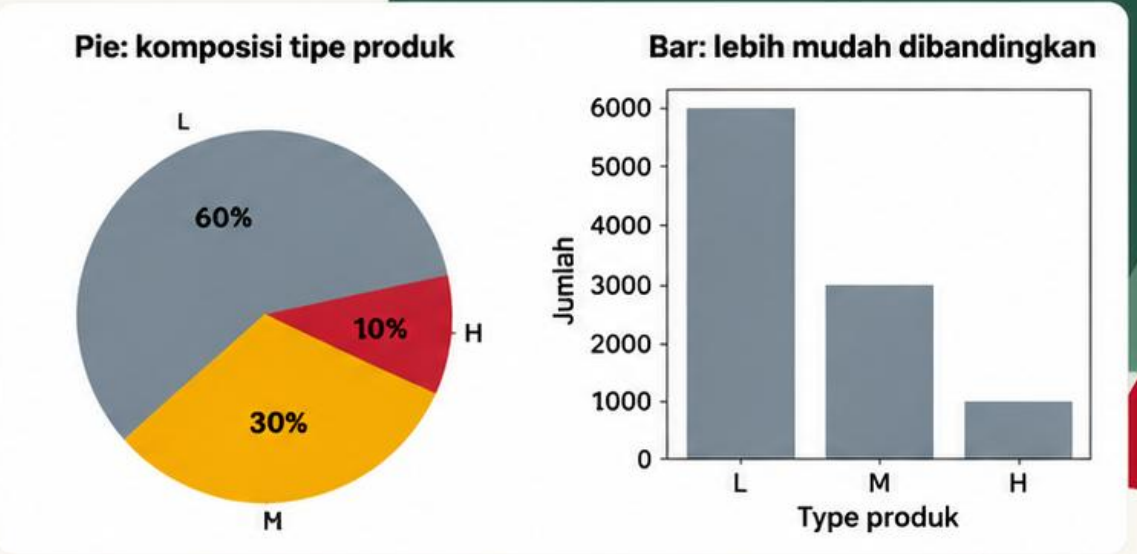
Catatan pembelajaran: perhatikan pola sebaran titik, arah hubungan (positif/negatif), dan apakah ada kelompok yang terpisah dengan jelas.

15 Pie Chart: Komposisi, dan Kapan Sebaiknya Dihindari

```python
import matplotlib.pyplot as plt

jumlah = df["Type"].value_counts()

fig, ax = plt.subplots(1, 2, figsize=(8, 3.5))
ax[0].pie(jumlah, labels=jumlah.index, autopct="%1.0f%%",
          colors=["#7C8B99", "#F2A900", "#D64545"])
ax[0].set_title("Pie: komposisi tipe produk")
ax[1].bar(jumlah.index, jumlah.values, color="#7C8B99")
ax[1].set_title("Bar: lebih mudah dibandingkan")
plt.show()
```



Intinya: pie hanya untuk melihat porsi dari keseluruhan. Untuk membandingkan besar nilai, grafik batang hampir selalu lebih jelas.

Element	Fungsi
pie()	Membuat grafik proporsi
bar()	Membandingkan nilai kategori
value_counts()	Menghitung jumlah setiap kategori

Aturan pemakaian pie

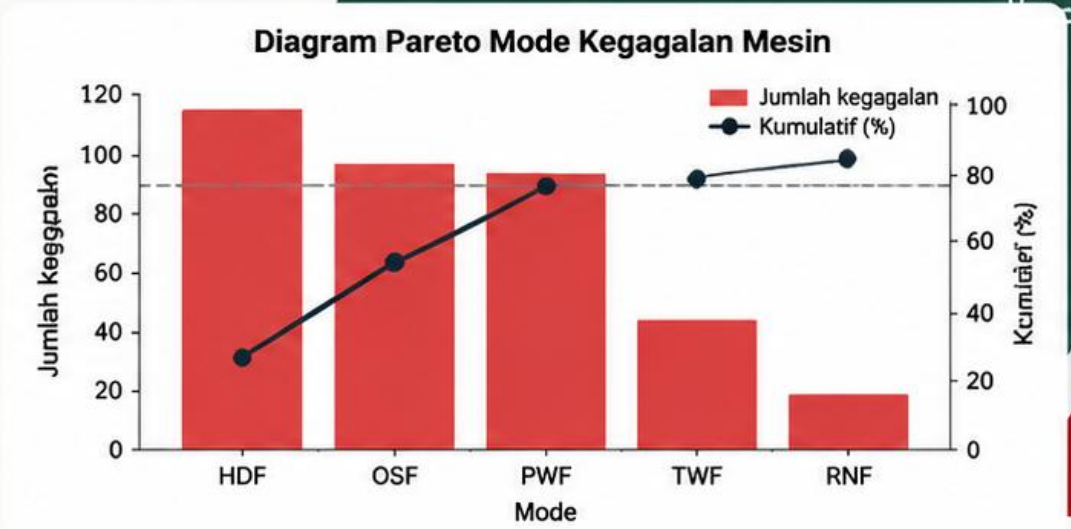
Pakai	2–4 bagian, menunjukkan proporsi dari total 100%.
Hindari	Terlalu banyak kategori, nilai yang mirip, atau perbandingan antar-waktu.

16 Diagram Pareto dengan Matplotlib

```python
import matplotlib.pyplot as plt

mode = df[["HDF", "OSF", "PWF", "TWF", "RNF"]].sum()
mode = mode.sort_values(ascending=False)
kum = mode.cumsum() / mode.sum() * 100

fig, ax1 = plt.subplots(figsize=(7,4))
ax1.bar(mode.index, mode.values, color="#D64545")
ax2 = ax1.twinx()
ax2.plot(mode.index, kum, color="#1F2A36", marker="o")
ax2.axhline(80, color="gray", iinestyle="--")
plt.show()
```



Membaca Pareto

1. Batang menunjukkan jumlah kejadian.
2. Garis menunjukkan persentase kumulatif.
3. Fokus pada penyebab terbesar terlebih dahulu.

Fungsi	Kegunaan
bar()	Membuat jumlah kejadian (batang)
plot()	Membuat garis kumulatif
twinx()	Menambahkan sumbu Y kedua (untuk persentase)
sort_values()	Mengurutkan penyebab dari terbesar ke terkecil

Mode	n	Kum %
HDF	115	30.8
OSF	98	57.1
PWF	95	82.6

Total mode 373 > 339 mesin gagal

Intinya: prinsip Pareto membantu menenukan sedikit penyebab utama yang memberikan dampak terbesar. Pada industri, analisis ini membantu menentukan prioritas perbaikan mesin.

17 EDA Langkah 1: Memuat dan Memeriksa Dataset Kaggle

```python
import pandas as pd

df = pd.read_csv("ai4i2020.csv")
print(df.shape)
print(df.isna().sum().sum(), df.duplicated().sum())

df.columns = ["udi", "product_id", "tipe", "suhu_udara",
              "suhu_proses", "rpm", "torsi", "keausan_tool",
              "gagal", "TWF", "HDF", "PWF", "OSF", "RNF"]

print(df["gagal"].value_counts())
print(f"Laju kegagalan: {df['gagal'].mean():.2%}")
```

OUTPUT

```
gagal
0    9661
1     339
Laju kegagalan: 3.39%
```

Kabar baik

Tanpa nilai kosong dan duplikat. Dataset Kaggle yang dikurasi umumnya lebih bersih, tetapi tetap wajib diperiksa.

Nama kolom diganti

Nama asli panjang dan berisi spasi serta kurung. Nama pendek memudahkan penulisan kode.

Intinya

Data sangat tidak seimbang: hanya 3,39% gagal. Rata-rata keseluruhan bisa menyembunyikan pola kegagalan.

Elemen	Fungsi
shape	Melihat ukuran data (jumah baris, kolom)
isna()	Mengecek nilai kosong (missing value)
duplicated()	Mengecek data ganda (duplikat)
value_counts()	Melihat distribusi kategori (frekuensi tiap nilai)

18 EDA Langkah 2: Statistik dan Perbandingan Kelompok

```python
import pandas as pd

kolom = ["suhu_udara", "suhu_proses", "rpm", "torsi", "keausan_tool"]
df.groupby("gagal")[kolom].mean().round(1)

df.groupby("tipe")["gagal"].agg(["count", "sum", "mean"])
```

OUTPUT 1: Rata-rata variabel berdasarkan kondisi mesin

gagal	suhu_udara	suhu_proses	rpm	torsi	keausan_tool
0 (Normal)	300.0	310.0	1540.0	39.6	106.7
1 (Gagal)	300.9	310.3	1496.5	50.2	143.8

OUTPUT 2: Statistik kegagalan berdasarkan tipe produk

tipe	count	sum (gagal)	mean (gagal)
H	1003	21	0.021
L	6000	235	0.039
M	2997	83	0.028

Apa yang terlihat?

- **Torsi saat gagal** lebih tinggi (50,2 Nm vs 39,6 Nm).
- **Keausan tool** lebih besar (143,8 menit vs 106,7 menit).
- **Suhu udara** dan suhu proses relatif sama.
- **Tipe L** memiliki tingkat **gagal** tertinggi (3,9%).

Cara membaca hasil

Statistik kelompok membantu menemukan variabel yang berbeda antara mesin normal dan gagal. Variabel yang berbeda dapat menjadi kandidat penyebab kegagalan.

Intinya

Bandingkan kelompok gagal vs normal untuk menemukan faktor yang paling membedakan kondisi mesin.

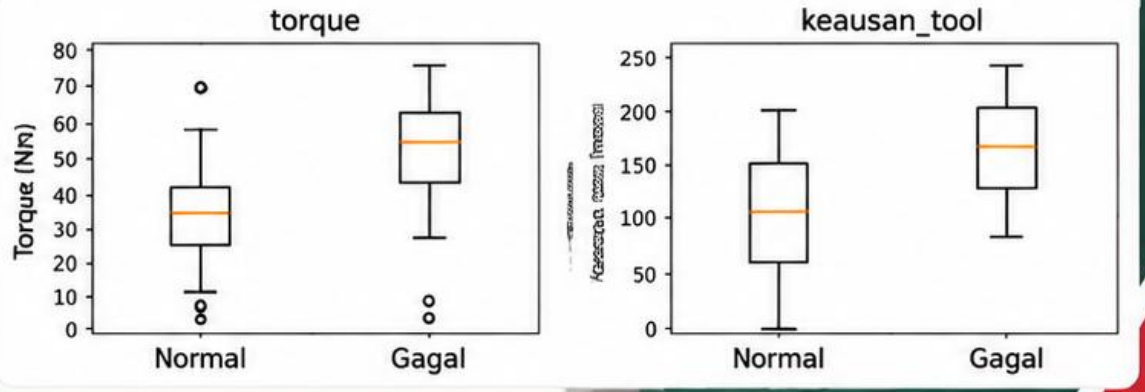
Fungsi	Kegunaan
groupby()	Membagi data berdasarkan kelompok (misal: gagal, tipe).
mean()	Menghitung rata-rata nilai.
agg()	Membuat beberapa ringkasan statistik (misal: count, sum, mean).

19

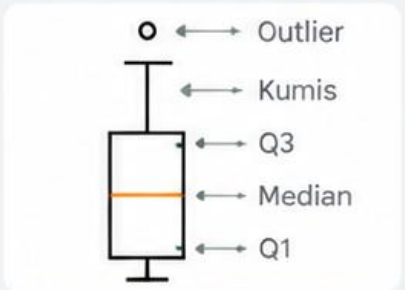
EDA Langkah 3: Boxplot Membandingkan Sebaran

```python
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 2, figsize=(8, 3.5))
for i, kol in enumerate(["torque", "keausan_tool"]):
    ax[i].boxplot([df.loc[df["gagal"] == 0, kol],
                   df.loc[df["gagal"] == 1, kol]])
    ax[i].set_xticks([1,2], ["Normal", "Gagal"])
    ax[i].set_title(kol)
plt.tight_layout()
plt.show()
```



Cara membaca boxplot



1. Kotak = 50% data tengah (Q1–Q3)
2. Garis tengah = median
3. Kumis = rentang data wajar
4. Titik di luar = outlier (pencilan)

Elemen	Makna
Median	Nilai tengah data
Q1 – Q3	Sebaran 50% data utama
Whisker	Rentang nilai normal (tidak termasuk outlier)
Outlier	Data yang menyimpang dari pola umum

Intinya: boxplot membantu melihat perbedaan distribusi antar kelompok. Jangan hanya melihat rata-rata, karena bentuk sebaran dan pencilan juga penting dalam analisis kegagalan mesin.

Catatan pembelajaran: Gunakan boxplot untuk membandingkan kelompok dan menemukan pola yang tidak terlihat dari nilai rata-rata saja.

TERIMA KASIH

Semoga materi ini bermanfaat dan
dapat membantu dalam memahami
analisis dan visualisasi data menggunakan Python.

