**Rangkuman Materi Perkuliahan Machine Learning**

**Kelompok 7**

Nama : Abdul Hayyi El Naseer (241552010010)

Nama : Gema Syahdan Prasetyo (241552010021)

Teknik Informatika, STMIK Tazkia

Teknik Informatika, STMIK Tazkia

E-mail : [241552010010.hayyi@student.stmik.tazkia.ac.id](mailto:241552010010.hayyi@student.stmik.tazkia.ac.id)

1. **Pertemuan 9 : Pengantar Machine Learning (ML) Dan Artificial Intelligence (AI)**
2. **Deskripsi Singkat**

Dalam pertemuan ini membahas pengantar komprehensif yang mencakup definisi fundamental ML menurut Tom Mitchell yaitu kemampuan komputer untuk meningkatkan kinerjanya pada suatu tugas (T) berdasarkan pengalaman (E) yang diukur melalui metrik performa (P) serta menekankan bahwa pengalaman tersebut identik dengan data sebagai fakta, yang kemudian diproses untuk menghasilkan pengetahuan dalam bentuk model atau pola yang konsisten. Dosen juga memperkenalkan berbagai profesi terkait seperti Data Scientist, AI Engineer, dan Data Analyst, menjelaskan hubungan ML dengan disiplin ilmu lain seperti probabilitas-statistik, serta membahas tren industri seperti AI, keamanan siber (security), blockchain, dan ESG yang relevan untuk masa depan, sembari menekankan pentingnya pembelajaran seumur hidup, pemahaman konsep dasar, dan pemanfaatan AI sebagai alat bantu dalam pengembangan karir di bidang teknologi.

1. **Mindmap & Taksonomi**



**Gambar 1.** Machine Learning (AI)

1. **Penjelasan Detail**
2. **Orientasi Karir :**

* Mengenalkan ekosistem profesi yang lahir dari bidang ini : Data Scientist, AI/ML Engineer, Data Engineer, Data Analyst, dan Solution Architect.
* Dilakukan analisis pasar : profesi ini masih banyak dicari dan relatif "aman" dari penggantian oleh AI itu sendiri karena AI masih membutuhkan domain expert manusia.
* Ada peringatan praktis : Data Analyst disebut memiliki bayaran yang relatif lebih rendah, memotivasi mahasiswa untuk menargetkan peran yang lebih teknis seperti ML Engineer.

1. **Pondasi Filosofis dan Definisi Kunci :**

* Menekankan bahwa fondasi kuat (konsep fundamental) adalah kunci sukses, lebih penting daripada sekadar mengikuti tutorial praktis.
* Definisi Machine Learning dari buku Tom Mitchell (1997) diperkenalkan sebagai landasan utama: "Sebuah komputer dikatakan belajar dari pengalaman (E) terhadap serangkaian tugas (T) dan ukuran kinerja (P), jika kinerjanya pada tugas dalam T, yang diukur oleh P, meningkat dengan pengalaman E."
* Definisi ini dijelaskan dengan analogi manusia belajar (misalnya, belajar untuk ujian), untuk memastikan pemahaman intuitif. Menekankan bahwa definisi buku ini akan menjadi acuan standar selama kuliah, terutama untuk menghindari interpretasi yang tidak konsisten.

1. **Konsep Inti dan Hubungannya :**

* Dijabarkan hubungan logis antara Data, Pembelajaran, dan Pengetahuan:
  1. Data didefinisikan sebagai fakta (pengalaman E yang telah terjadi).
  2. Data yang noisy atau bukan fakta akan menghasilkan pengetahuan yang buruk.
  3. Komputer menggunakan data ini untuk belajar.
  4. Hasil pembelajaran adalah Pengetahuan, yang bisa berupa model, hipotesis, atau pola.
  5. Kriteria utama pengetahuan: harus konsisten terhadap data yang ada. Jika data berubah, pengetahuan boleh direvisi.
* Machine Learning ditempatkan sebagai subset dari Artificial Intelligence (AI). AI memiliki cabang lain seperti searching, planning, reasoning.
* Probabilitas dan Statistik (Propstat) disebut sebagai fondasi matematika yang esensial bagi ML. Tanpa pemahaman ini, model ML akan "berat" atau salah.

1. **Lanskap Teknologi dan Tren Masa Depan :**

* Memetakan tiga area utama yang prospektif dalam 5 tahun ke depan:

1. AI & Machine Learning (termasuk LLM dan GPT).
2. Security (Keamanan Siber) dan Compliance (kepatuhan pada regulasi seperti UU PDP/GDPR).
3. Blockchain & Web 3.0.

* Ditekankan bahwa lulusan perlu menguasai salah satu, dan kombinasi (misal: AI + Security) sangat bernilai.
* ESG (Environmental, Social, and Governance) diperkenalkan sebagai domain aplikasi baru yang sedang naik daun, tempat teknologi AI dan data dapat diterapkan (e.g., analisis citra satelit untuk penghijauan).

1. **Nasihat dan Strategi Pembelajaran :**

* Long Life Learning: Ditekankan sebagai keharusan mutlak di bidang IT untuk menghindari keusangan.
* Belajar dari Kesalahan: Dianalogikan dengan cara mesin belajar (trial and error). Kegagalan (dalam kuliah atau proyek) adalah bagian dari proses pembelajaran, bukan indikasi kebodohan.
* Memahami, bukan Menghafal: Pentingnya memahami "mengapa" di balik suatu algoritma, bukan hanya "bagaimana" menggunakannya.
* AI sebagai Alat Bantu: Mahasiswa didorong untuk memanfaatkan AI (seperti ChatGPT untuk coding), tetapi dengan kewaspadaan karena AI bisa salah dan membutuhkan domain expert manusia untuk mengarahkannya.

1. **Logistik dan Administrasi Perkuliahan :**
2. Buku Referensi: Utama adalah "Introduction to Machine Learning" oleh Tom Mitchell (buku "jadul" tapi fondasional). Buku pendamping lain juga disebutkan.
3. Metode Pengajaran: Dosen menggunakan Excalidraw untuk peta konsep visual dan akan membagikan materi.
4. Penugasan dan Skripsi: Disinggung bahwa tugas akan diberikan, dan topik skripsi di masa depan dapat diambil dari area penelitian dosen (AI Security, AI for ESG, dll.).
5. **Sampel Code (Ilustrasi Konsep T, P, E) :**

import random

# Contoh sederhana: Komputer belajar mengerjakan tugas "menebak angka"

# E (Experience/Data): Pengalaman berupa pasangan angka-input dan jawaban-benar

pengalaman = [

(3, 7.5), # Input 3, output yang diharapkan 7.5

(5, 12.5), # Input 5, output 12.5

(8, 20.0) # Input 8, output 20.0

]

# T (Task): Tugas memprediksi output berdasarkan input baru

def tugas(model, input\_angka):

"""Tugas: Memprediksi output dari input"""

return model \* input\_angka # Model sederhana: perkalian

# P (Performance Measure): Mengukur performa (selisih absolut)

def ukur\_performa(prediksi, jawaban\_benar):

"""Ukuran performa: error absolut"""

return abs(prediksi - jawaban\_benar)

# Proses belajar sederhana

def belajar(pengalaman):

"""Belajar dari pengalaman untuk mencari model terbaik"""

model\_terbaik = None

performa\_terbaik = float('inf')

# Coba berbagai kemungkinan model (dari 1.0 hingga 3.0)

for model in [i/10 for i in range(10, 31)]: # 1.0, 1.1, ..., 3.0

total\_error = 0

# Evaluasi model terhadap semua pengalaman

for input\_angka, jawaban\_benar in pengalaman:

prediksi = tugas(model, input\_angka)

error = ukur\_performa(prediksi, jawaban\_benar)

total\_error += error

# Rata-rata error untuk model ini

rata\_error = total\_error / len(pengalaman)

# Pilih model dengan error terkecil

if rata\_error < performa\_terbaik:

performa\_terbaik = rata\_error

model\_terbaik = model

return model\_terbaik, performa\_terbaik

# SIMULASI

print("=== PROSES BELAJAR KOMPUTER ===")

print("E (Pengalaman):", pengalaman)

print("T (Tugas): Prediksi output = model × input")

print("P (Ukuran Performa): Error absolut rata-rata\n")

# Fase belajar

model\_dipelajari, performa = belajar(pengalaman)

print(f"Hasil Belajar:")

print(f"- Model terbaik: {model\_dipelajari}")

print(f"- Error rata-rata: {performa:.2f}\n")

# Fase pengujian

print("Pengujian pada input baru (4):")

prediksi = tugas(model\_dipelajari, 4)

print(f"Prediksi: {model\_dipelajari} × 4 = {prediksi:.1f}")

print(f"Harapan: sekitar 10.0 (karena pola: input × 2.5)")

**Hasilnya :**

=== PROSES BELAJAR KOMPUTER ===

E (Pengalaman): [(3, 7.5), (5, 12.5), (8, 20.0)]

T (Tugas): Prediksi output = model × input

P (Ukuran Performa): Error absolut rata-rata

Hasil Belajar:

- Model terbaik: 2.5

- Error rata-rata: 0.00

Pengujian pada input baru (4):

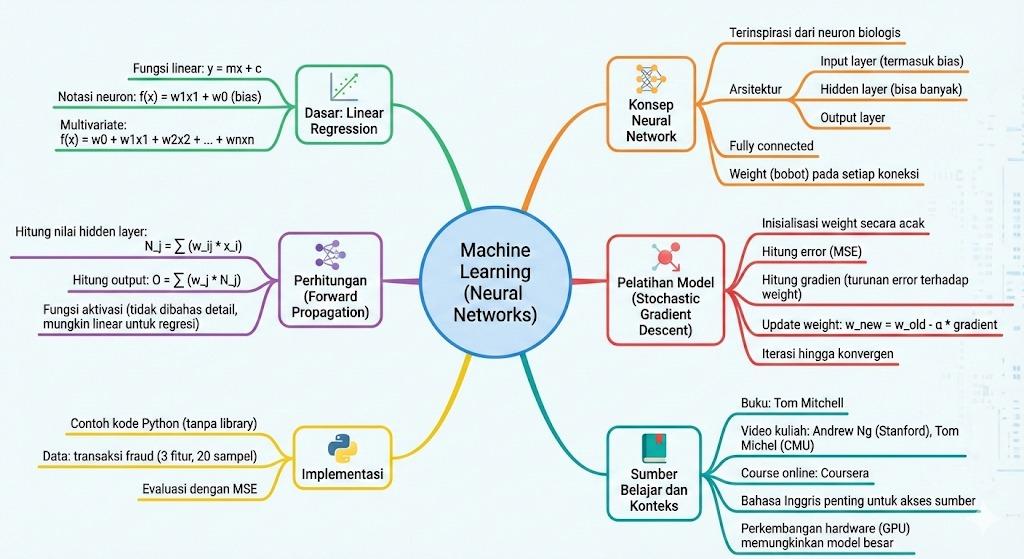
Prediksi: 2.5 × 4 = 10.0

Harapan: sekitar 10.0 (karena pola: input × 2.5)

1. **Pertemuan 10 : Regresi Linear Neural Network**
2. **Deskripsi**

Bagian ini membahas perkembangan konseptual dari regresi linear menuju neural network, dimulai dengan penjelasan tentang transformasi persamaan linear (y=mx+c) menjadi notasi bobot dan bias yang membentuk neuron tunggal, kemudian berkembang dengan penambahan hidden layer yang menciptakan arsitektur jaringan grafis kompleks; dalam sesi interaktif ini, pengajar juga mendemonstrasikan algoritma pelatihan stochastic gradient descent melalui forward propagation, perhitungan error dengan Mean Squared Error (MSE), serta proses iteratif pembaruan bobot, sambil membahas tantangan praktis seperti pemilihan arsitektur (yang disebut "black magic" karena memerlukan eksperimen), implementasi kode menggunakan data nyata, rekomendasi sumber belajar (seperti kuliah Andrew Ng), dan konteks historis mengapa neural network baru optimal dengan komputasi modern, semuanya disampaikan dengan pendekatan visual, analogi, dan penekanan pada pemahaman matematis dasar sebelum menggunakan library jadi.

1. **Mindmap & Taksonomi**



**Gambar 2.** Machine Learning (Neural Networks)

1. **Penjelasan Detail**

Mind map ini berfungsi sebagai panduan belajar yang terstruktur, dimulai dari fondasi matematika paling dasar, bergerak ke arsitektur jaringan, cara jaringan melakukan perhitungan, bagaimana jaringan "belajar" melalui pelatihan, hingga contoh implementasi dan sumber daya untuk belajar lebih lanjut.

1. **Dasar: Linear Regression (Fondasi)**

Cabang ini (berwarna hijau) menjelaskan bahwa Neural Network yang kompleks sebenarnya dibangun di atas konsep matematika yang sederhana, yaitu regresi linear. Sebuah neuron tunggal pada dasarnya melakukan operasi linear ini.

* Fungsi Linear Sederhana: Ini adalah persamaan garis lurus dasar yang kita pelajari di sekolah.

Di mana adalah gradien (kemiringan) dan adalah intersep (titik potong sumbu y).

* Notasi Neuron: Dalam konteks neural network, kita mengubah notasinya. Kemiringan m menjadi bobot dan intersep menjadi bias
* Multivariate (Banyak Variabel): Jika input data memiliki banyak fitur (misalnya: tinggi badan, berat badan, usia), persamaannya berkembang menjadi penjumlahan berbobot dari semua input ditambah bias.

Atau dalam notasi sigma yang lebih ringkas:

1. **Konsep Neural Network (Arsitektur)**

Cabang ini (berwarna oranye) menjelaskan bagaimana unit-unit dasar tadi disusun menjadi sebuah jaringan.

* Inspirasi: Konsep ini meniru cara kerja neuron biologis di otak manusia.
* Arsitektur (Struktur): Terdiri dari lapisan-lapisan (layers):
* Input Layer: Tempat data masuk (termasuk neuron bias).
* Hidden Layer: Lapisan di tengah tempat pemrosesan utama terjadi (bisa terdiri dari banyak lapisan untuk "Deep Learning").

Output Layer: Lapisan yang menghasilkan prediksi akhir.

* Koneksi:
* Fully Connected: Setiap neuron di satu lapisan terhubung ke semua neuron di lapisan berikutnya.
* Weight (Bobot): Setiap koneksi tersebut memiliki nilai numerik sendiri (bobot) yang menentukan seberapa kuat pengaruh satu neuron terhadap neuron berikutnya.

1. **Perhitungan (Forward Propagation)**

Cabang ini (berwarna ungu) menjelaskan aliran data dari input ke output. Ini adalah proses jaringan membuat prediksi.

* Hitung Nilai Hidden Layer: Nilai neuron di lapisan tersembunyi dihitung dengan menjumlahkan perkalian input dengan bobotkoneksinya

(Catatan: Biasanya setelah penjumlahan ini, ada "Fungsi Aktivasi" seperti ReLU atau Sigmoid yang diterapkan, meskipun di diagram disebutkan tidak dibahas detail).

* Hitung Output: Proses yang sama diulangi untuk lapisan output. Nilai output akhir
* Adalah jumlah berbobot dari nilai-nilai neuron di hidden layer.

1. **Pelatihan Model (Stochastic Gradient Descent)**

Cabang ini (berwarna merah) adalah bagian paling krusial: bagaimana jaringan "belajar" dari kesalahannya untuk memperbaiki prediksi. Metodenya disebut Stochastic Gradient Descent (SGD).

* Langkah-langkah Pelatihan:

1. Inisialisasi: Mulai dengan nilai bobot w yang diacak.
2. Hitung Error: Lakukan forward propagation untuk mendapatkan prediksi, lalu bandingkan dengan target sebenarnya menggunakan fungsi error seperti MSE (Mean Squared Error).
3. Hitung Gradien: Cari tahu arah mana yang harus diambil untuk mengurangi error. Ini dilakukan dengan menghitung turunan (gradien) dari error terhadap masing-masing bobot.
4. Update Weight (Perbarui Bobot): Perbarui bobot lama

untuk mendapatkan bobot baru dengan bergerak berlawanan arah dengan gradien.

(Di sini, \alpha adalah "learning rate", yang menentukan seberapa besar langkah perubahan bobot).

Iterasi: Ulangi proses ini terus-menerus hingga error-nya minimal (konvergen).

1. **Implementasi dan Sumber Belajar**

Dua cabang terakhir (kuning dan biru tua) bersifat praktis dan kontekstual.

* Implementasi (Kuning): Menyarankan untuk mencoba membangun neural network dari nol menggunakan Python (tanpa library instan seperti TensorFlow/PyTorch dulu) untuk memahami logika dasarnya. Contoh kasusnya adalah mendeteksi transaksi penipuan (fraud).
* Sumber Belajar dan Konteks (Biru Tua): Memberikan referensi penting seperti buku (Tom Mitchell), kuliah ahli (Andrew Ng), dan kursus online (Coursera). Juga menekankan pentingnya Bahasa Inggris untuk akses materi dan peran hardware modern (GPU) yang memungkinkan model AI menjadi sangat besar saat ini.

1. **Sample Code**

import numpy as np

# ==========================================

# 1. DATA (Cabang: Implementasi)

# ==========================================

# Sesuai mindmap: "Data: transaksi fraud (3 fitur)" [cite: 167]

# Contoh: [Jumlah Transaksi, Jarak Lokasi, Frekuensi]

# Input (X): 4 sampel, 3 fitur

X = np.array([

[0, 0, 1],

[1, 1, 1],

[1, 0, 1],

[0, 1, 1]

])

# Target (y): 1 = Fraud, 0 = Aman

y = np.array([[0], [1], [1], [0]])

# ==========================================

# 2. KONSEP & ARSITEKTUR (Cabang: Konsep Neural Network)

# ==========================================

# Kita membangun arsitektur: Input -> Hidden Layer -> Output Layer

# Fungsi Aktivasi (Sigmoid) - mengubah nilai menjadi range 0-1

def sigmoid(x):

return 1 / (1 + np.exp(-x))

# Turunan Fungsi Sigmoid (untuk menghitung gradien)

def sigmoid\_derivative(x):

return x \* (1 - x)

# Inisialisasi Weight secara acak [cite: 155]

np.random.seed(42)

# Bobot dari Input (3 neuron) ke Hidden Layer (4 neuron)

weights\_input\_hidden = 2 \* np.random.random((3, 4)) - 1

# Bobot dari Hidden Layer (4 neuron) ke Output (1 neuron)

weights\_hidden\_output = 2 \* np.random.random((4, 1)) - 1

# Learning Rate (Alpha)

alpha = 0.5

print("Mulai Pelatihan Model...")

# ==========================================

# 3. PELATIHAN MODEL (Cabang: Pelatihan Model / SGD)

# ==========================================

# Iterasi hingga konvergen (di sini kita batasi 10.000 epoch) [cite: 163]

for iterasi in range(10000):

# --- A. FORWARD PROPAGATION (Cabang: Perhitungan) ---

# 1. Hitung nilai Hidden Layer: N\_j = sum(w\_ij \* x\_i)

input\_layer = X

hidden\_layer\_input = np.dot(input\_layer, weights\_input\_hidden)

hidden\_layer\_output = sigmoid(hidden\_layer\_input) # Aktivasi

# 2. Hitung Output: O = sum(w\_j \* N\_j) [cite: 148]

final\_input = np.dot(hidden\_layer\_output, weights\_hidden\_output)

final\_output = sigmoid(final\_input)

# --- B. HITUNG ERROR ---

# Error = Target - Prediksi

error = y - final\_output

if (iterasi % 2000) == 0:

# Menampilkan Mean Squared Error (MSE) [cite: 156]

mse = np.mean(np.square(error))

print(f"Iterasi {iterasi}: Error (MSE) = {mse:.5f}")

# --- C. BACKPROPAGATION (Hitung Gradien) ---

# Hitung gradien untuk Output Layer (seberapa besar kita harus mengubah bobot?)

# Gradien = Error \* Turunan Fungsi Aktivasi

delta\_output = error \* sigmoid\_derivative(final\_output)

# Hitung error di Hidden Layer (kontribusi hidden layer terhadap error output)

error\_hidden = delta\_output.dot(weights\_hidden\_output.T)

# Hitung gradien untuk Hidden Layer

delta\_hidden = error\_hidden \* sigmoid\_derivative(hidden\_layer\_output)

# --- D. UPDATE WEIGHT (Cabang: Pelatihan Model) ---

# Rumus: w\_new = w\_old + (alpha \* gradient \* input)

# (Catatan: Tanda + digunakan karena kita meminimalkan error, 'menuruni bukit')

weights\_hidden\_output += hidden\_layer\_output.T.dot(delta\_output) \* alpha

weights\_input\_hidden += input\_layer.T.dot(delta\_hidden) \* alpha

# ==========================================

# 4. HASIL AKHIR

# ==========================================

print("\n=== Hasil Setelah Pelatihan ===")

print("Output Prediksi:")

print(final\_output)

print("\nTarget Seharusnya:")

print(y)

**Hasilnya :**

Mulai Pelatihan Model...

Iterasi 0: Error (MSE) = 0.25952

Iterasi 2000: Error (MSE) = 0.24998

Iterasi 4000: Error (MSE) = 0.24997

Iterasi 6000: Error (MSE) = 0.24996

Iterasi 8000: Error (MSE) = 0.24994

=== Hasil Setelah Pelatihan ===

Output Prediksi:

[[0.5 ]

[0.49999929]

[0.5000005 ]

[0.50000121]]

Target Seharusnya:

[[0]

[1]

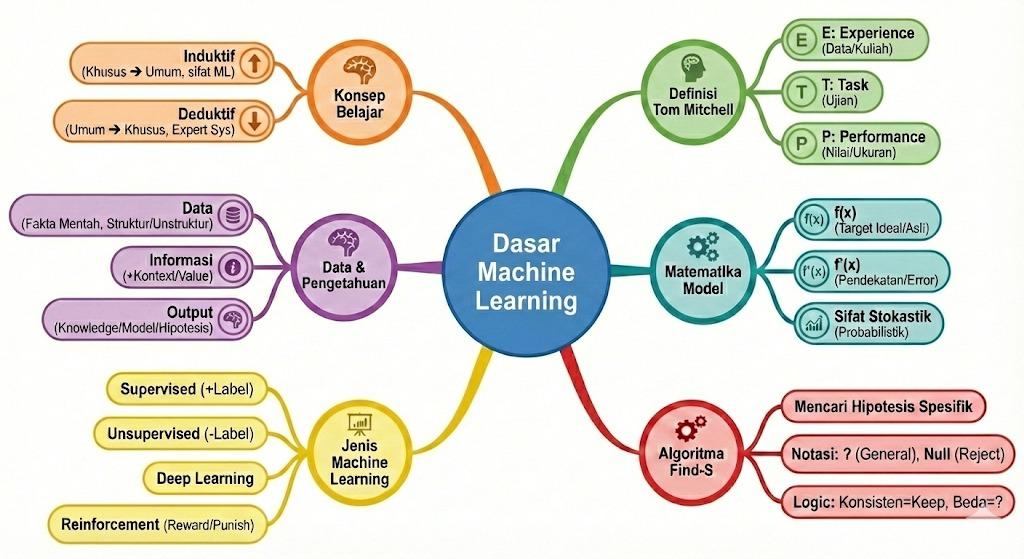
[1]

[0]]

1. **Pertemuan 11 : Fundamental Machine Learning (ML)**
2. **Deskripsi**

Pertemuan ini menjelaskan konsep fundamental Machine Learning (ML) sebagai sebuah disiplin yang berparadigma induktif, yaitu proses di mana mesin belajar dari data-data spesifik (pengalaman E) untuk membentuk pengetahuan umum berupa model, pola, atau hipotesis (f'(x)). Proses ini didefinisikan secara formal oleh Tom Mitchell: suatu sistem dikatakan belajar jika kinerjanya (P) dalam suatu tugas (T) meningkat seiring pengalaman (E), yang dianalogikan seperti mahasiswa yang nilai akhirnya membaik setelah melalui kuliah, tugas, dan ujian. Berbeda dengan pendekatan deduktif pada sistem pakar yang menanamkan aturan umum (f(x)) dari ahli ke komputer, ML membangun model pendekatan (f'(x)) seperti pada ChatGPT yang merupakan representasi probabilistik dan dapat mengandung error, karena dibentuk secara iteratif dengan membandingkan prediksi model terhadap output yang sebenarnya untuk meminimalkan selisih (error). Secara taksonomi, ML dibagi berdasarkan ketersediaan label data, mencakup supervised learning (dengan label), unsupervised learning (tanpa label), deep learning (menangani data kompleks dan masif), serta reinforcement learning (berbasis reward/punishment), dengan contoh algoritma dasar seperti Find-S yang menggambarkan pencarian hipotesis paling spesifik yang konsisten dengan data positif.

1. **Mindmap & Taksonomi**

****

**Gambar 3.** Dasar Machine Learning

1. **Penjelasan Detail**
2. **Inti Dasar : Machine Learning adalah Pembelajaran Induktif**

ML digambarkan sebagai proses induktif, yaitu proses belajar dari contoh-contoh khusus (data spesifik) untuk menarik kesimpulan umum (model atau pola). Ini dibedakan dari penalaran deduktif yang dimulai dari aturan umum untuk diterapkan pada kasus khusus. Dalam konteks Kecerdasan Buatan (AI), "learning" mencakup kedua pendekatan tersebut, tetapi ML secara khusus mengadopsi paradigma induktif.

1. **Definisi Formal: Kerangka Tom Mitchell**

Definisi operasional ML dirujuk dari buku AI: A Modern Approach karya Tom Mitchell:

"Sebuah komputer dikatakan belajar dari pengalaman (E) terhadap suatu tugas (T) yang diukur dengan performa (P), jika performanya pada tugas T tersebut meningkat seiring dengan pengalaman E."

Penerapan pada Komputer: Dalam konteks komputer (misal, pengenalan gambar):

* T (Tugas): Merekognisi objek dalam gambar.
* E (Pengalaman/Data): Kumpulan gambar pohon yang telah diberi label.
* P (Performans): Akurasi prediksi (persentase tebakan yang benar).  
  Komputer "belajar" dari data (E), melakukan tugas (T), diukur akurasinya (P), dan jika belum baik, proses belajar (training) diulang dengan data atau teknik yang lebih baik.

1. **Istilah-Istilah Kunci dan Hubungannya**

Ia menjelaskan hierarki dan hubungan antar konsep inti:

* Data/Experience (E) : Fakta mentah tentang dunia. Bukan informasi (yang sudah bernilai kontekstual). Data untuk ML harus faktual, atau akan menjadi "noise" yang mengganggu.
* Pengetahuan (Knowledge): Goal akhir dari ML. Pengetahuan ini dapat termanifestasi dalam beberapa bentuk yang sering dipertukarkan:
* Model: Representasi matematis dari pengetahuan.
  + Hipotesis: Kesimpulan atau aturan yang konsisten dengan data yang ada. Sifatnya dinamis—dapat berubah jika data baru bertentangan. Hipotesis adalah "model sementara".
  + Pola (Pattern): Regularitas atau hubungan yang ditemukan dalam data.
  + Fungsi f'(x) : Implementasi konkret dari model/pengetahuan dalam bentuk matematis.

1. **Kontras Kunci: Sistem Pakar (Deduktif) vs. Machine Learning (Induktif)**

Perbedaan fundamental dijelaskan melalui analogi diagnosis medis:

|  |  |  |
| --- | --- | --- |
| Aspek | Sistem Pakar (Deduktif) | Machine Learning (Induktif) |
| Sumber Pengetahuan | Ahli (dokter) yang sudah memiliki aturan (f(x)). | Kumpulan data kasus (contoh diagnosis pasien). |
| Proses | Aturan ahli (f(x)) diformalkan (dengan metode formal/matematika) dan "ditanam" ke komputer | Mesin mencari sendiri fungsi pendekatan (f'(x)) dari data. |
| Tantangan | Ahli sering sulit memformalkan pengetahuan tacit mereka ke dalam aturan eksplisit. | Membuat model (f'(x)) yang mendekati pengetahuan ahli yang sebenarnya. |
| Output & Error | Menghasilkan Y (diagnosis) berdasarkan aturan pasti f(x). | Menghasilkan Y' (diagnosis) berdasarkan model aproksimasi f'(x) yang memiliki kemungkinan error. |
| Fleksibilitas | Statis, aturan sulit diubah. | Dinamis, dapat diperbarui dengan data baru. |

Kesimpulan : ML adalah solusi ketika pengetahuan ahli ada tetapi sulit atau tidak mungkin untuk dikodifikasikan secara tepat. Model seperti ChatGPT adalah contoh f'(x) yang mencoba meniru pola dari banyak ahli (data teks), namun tetap memiliki kemungkinan kesalahan.

1. **Ilustrasi Algoritma: Find-S**

Sebagai contoh konkret konsep hipotesis, diperkenalkan algoritma Find-S:

* Tujuan : Mencari hipotesis paling spesifik yang konsisten dengan semua contoh pelatihan positif.
* Konsep Ruang Hipotesis :
  + <?, ?, ?, ...> (Most General): Hipotesis yang menerima semua nilai fitur. Simbol ? berarti "nilai apa pun diterima".
  + <∅, ∅, ∅, ...> (Most Specific): Hipotesis yang tidak menerima nilai apa pun. Simbol ∅ berarti "tidak ada nilai yang diterima".
* Cara Kerja : Algoritma dimulai dari hipotesis paling spesifik (H0), lalu secara bertahap menggeneralisasikannya saat menemukan contoh positif yang tidak cocok. Generalisasi dilakukan dengan mengubah nilai spesifik menjadi ? pada fitur yang ternyata tidak relevan untuk prediksi.

1. **Sampel Code (kode sederhana Find-S)**

def find\_s(contoh\_positif):

"""Find-S dalam 6 baris"""

hipotesis = ['∅'] \* len(contoh\_positif[0]) # 1. Mulai dari paling spesifik

for contoh in contoh\_positif:

for i in range(len(hipotesis)):

if hipotesis[i] == '∅': # 2. Contoh pertama

hipotesis[i] = contoh[i]

elif hipotesis[i] != contoh[i]: # 3. Generalisasi

hipotesis[i] = '?'

return hipotesis

# Contoh data: kapan bermain tenis?

# Format: [Cuaca, Suhu, Kelembaban]

contoh = [

['Cerah', 'Hangat', 'Normal'], # Contoh 1: main tenis

['Cerah', 'Hangat', 'Tinggi'], # Contoh 2: main tenis

]

hasil = find\_s(contoh)

print("Hipotesis akhir:", hasil)

print("Artinya: Jika Cuaca=Cerah dan Suhu=Hangat (Kelembaban bisa apa saja)")

**Hasilnya :**

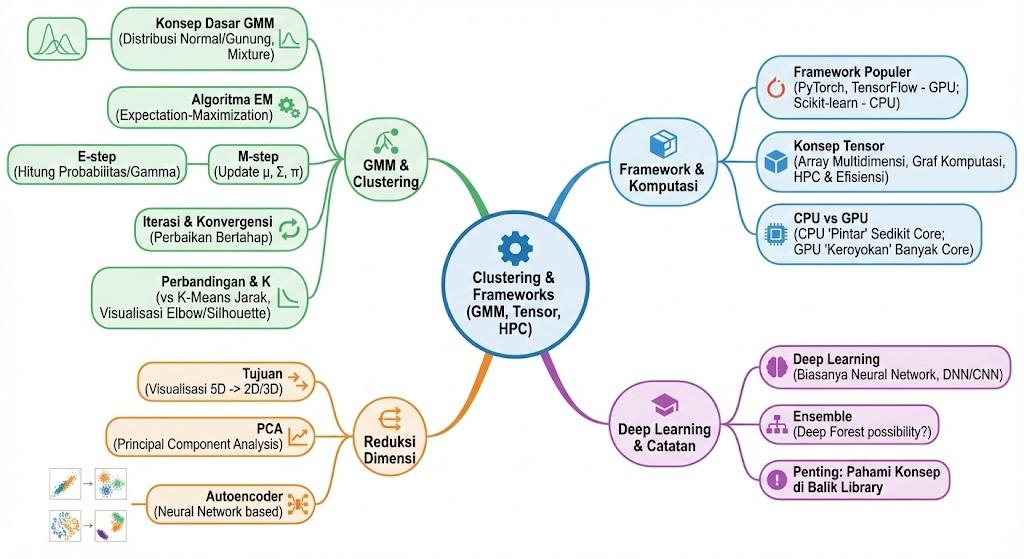
Hipotesis akhir: ['Cerah', 'Hangat', '?']

Artinya: Jika Cuaca=Cerah dan Suhu=Hangat (Kelembaban bisa apa saja)

1. **Pertemuan 12 : Clustering & Frameworks (GMM, Tensor, HPC)**
2. **Deskripsi**

Pertemuan ini berfokus pada integrasi antara konsep teoritis dan implementasi praktis dalam bidang Clustering & Frameworks (GMM, Tensor, HPC), yang dijelaskan melalui empat area utama : secara teoretis, area hijau membahas algoritma Clustering seperti Gaussian Mixture Model (GMM) yang berdasar pada distribusi normal ; sementara secara praktis, area biru muda menjelaskan implementasinya menggunakan framework perangkat lunak dan komputasi keras (CPU/GPU) yang memanfaatkan konsep Tensor untuk efisiensi; untuk mendukung proses ini, area oranye menekankan pentingnya teknik Reduksi Dimensi sebagai persiapan data ; dan sebagai catatan pelengkap, area ungu menyertakan konteks Deep Learning serta hal penting lainnya, sehingga secara keseluruhan menggambarkan keterkaitan yang erat antara matematika statistik dan pemrograman modern berperforma tinggi.

1. **Mindmap & Taksonomi**



**Gambar 4.** GMM, Tensor, HPC

1. **Penjelasan Detail**

Penjelasan Detail per Cabang Berikut adalah rincian dari setiap node dalam mindmap tersebut:

1. **Cabang Hijau: GMM & Clustering (Teori Pengelompokan)**

Cabang ini membahas secara mendalam tentang metode Gaussian Mixture Model (GMM).

* **Konsep Dasar GMM:**
* Menjelaskan bahwa GMM didasarkan pada Distribusi Normal (kurva lonceng atau "gunung").
* Disebut "Mixture" karena mengasumsikan data adalah campuran dari beberapa distribusi normal. (Ikon menunjukkan tiga kurva lonceng yang tumpang tindih).
* **Algoritma EM (Expectation-Maximization):**
* Ini adalah metode yang digunakan untuk melatih model GMM. Terbagi menjadi dua langkah berulang:
  + E-step (Expectation): Menghitung probabilitas (sering disebut Gamma) bahwa suatu titik data termasuk dalam kluster tertentu berdasarkan parameter saat ini.
  + M-step (Maximization): Memperbarui parameter model ($\mu$ untuk rata-rata, $\Sigma$ untuk kovarians/sebaran, dan $\pi$ untuk bobot campuran) berdasarkan probabilitas yang dihitung di E-step.
* **Iterasi & Konvergensi:**
* Menjelaskan bahwa langkah E dan M dilakukan berulang-ulang (Iterasi, ditandai ikon panah memutar) untuk memperbaiki model secara bertahap hingga hasilnya stabil (Konvergensi).
* **Perbandingan & K:**
* Membandingkan GMM dengan K-Means (K-Means berbasis jarak kaku, sedangkan GMM berbasis probabilitas yang lebih fleksibel).
* Menyebutkan cara menentukan jumlah kluster ('K') yang optimal, seperti melalui visualisasi (metode Elbow atau Silhouette).

1. **Cabang Biru Muda: Framework & Komputasi (Alat & Perangkat Keras)**

Cabang ini membahas alat praktis untuk menerapkan teori di atas.

* **Framework Populer:**
* Menyebutkan pustaka Python standar: PyTorch dan TensorFlow (unggul karena dukungan GPU untuk data besar), dan Scikit-learn (lebih banyak berjalan di CPU untuk model yang lebih sederhana).
* **Konsep Tensor:**
* **Array Multidimensi:** Menjelaskan bahwa data dalam ML modern direpresentasikan sebagai Tensor (array N-dimensi), bukan sekadar tabel biasa.
* **Graf Komputasi & HPC:** Framework modern menggunakan graf komputasi untuk mengoptimalkan operasi matematika agar efisien (HPC - High Performance Computing).
* **CPU vs GPU:**
* Memberikan analogi untuk membedakan peran perangkat keras:
* **CPU:** Diibaratkan "Pintar" tapi memiliki sedikit core (otak). Bagus untuk tugas serial yang kompleks.
* **GPU:** Diibaratkan "Keroyokan" karena memiliki ribuan core sederhana. Sangat efisien untuk operasi paralel masif seperti perkalian matriks dalam GMM atau Deep Learning.

1. **Cabang Oranye: Reduksi Dimensi (Penyederhanaan Data)**

Cabang ini membahas teknik untuk menangani data yang memiliki terlalu banyak fitur (dimensi tinggi).

* **Tujuan:**
* Utamanya untuk visualisasi. Manusia sulit membayangkan data 5 Dimensi (5D), jadi perlu direduksi menjadi 2D atau 3D agar bisa diplot.
* **PCA (Principal Component Analysis):**
* Metode statistik klasik untuk mereduksi dimensi secara linear.
* **Autoencoder:**
* Metode berbasis Neural Network untuk reduksi dimensi (seringkali non-linear).
* Gambar kecil di sampingnya menunjukkan perbandingan hasil plot data: PCA (kiri) mungkin memisahkan data secara berbeda dibandingkan Autoencoder (kanan), menunjukkan karakteristik yang berbeda dari kedua metode.

1. **Cabang Ungu: Deep Learning & Catatan**

Cabang ini berisi catatan penutup dan koneksi ke topik yang lebih luas.

* **Deep Learning:**
* Menegaskan bahwa Deep Learning biasanya merujuk pada jaringan saraf tiruan yang dalam (DNN, CNN, dll.).
* **Ensemble:**
* Mengajukan pertanyaan diskusi: Apakah metode gabungan (ensemble) yang kompleks seperti "Deep Forest" juga bisa dikategorikan sebagai pendekatan Deep Learning?
* **Penting:**
* Pesan kunci dari materi ini: Sangat penting untuk memahami konsep di balik library. Jangan hanya bisa menggunakan kodenya (seperti Scikit-learn atau PyTorch) tanpa mengerti matematika atau logika yang terjadi di belakangnya.

1. **Sampel Code (Implementasi)**

# ==================== 1. KONSEP DASAR GMM & CLUSTERING ====================

import numpy as np

from sklearn.mixture import GaussianMixture

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

print("=== GMM vs K-MEANS ===")

# Data contoh sederhana

X = np.array([[1, 2], [1, 4], [1, 0],

[10, 2], [10, 4], [10, 0]])

# 1. GMM Clustering

gmm = GaussianMixture(n\_components=2, random\_state=42)

gmm\_labels = gmm.fit\_predict(X)

print("Hasil GMM:", gmm\_labels)

# 2. K-Means Clustering

kmeans = KMeans(n\_clusters=2, random\_state=42)

kmeans\_labels = kmeans.fit\_predict(X)

print("Hasil K-Means:", kmeans\_labels)

# ==================== 2. VISUALISASI SEDERHANA ====================

plt.figure(figsize=(12, 4))

# Plot GMM

plt.subplot(1, 2, 1)

plt.scatter(X[:, 0], X[:, 1], c=gmm\_labels, cmap='viridis')

plt.title("GMM Clustering")

plt.xlabel("X1")

plt.ylabel("X2")

# Plot K-Means

plt.subplot(1, 2, 2)

plt.scatter(X[:, 0], X[:, 1], c=kmeans\_labels, cmap='viridis')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1],

c='red', marker='X', s=200, label='Centers')

plt.title("K-Means Clustering")

plt.xlabel("X1")

plt.legend()

plt.tight\_layout()

plt.show()

# ==================== 3. PCA untuk REDUKSI DIMENSI ====================

from sklearn.decomposition import PCA

print("\n=== PCA CONTOH ===")

# Data berdimensi lebih tinggi

X\_high\_dim = np.array([

[1, 2, 3, 4],

[2, 3, 4, 5],

[3, 4, 5, 6],

[10, 11, 12, 13],

[11, 12, 13, 14]

])

# Reduksi ke 2D

pca = PCA(n\_components=2)

X\_reduced = pca.fit\_transform(X\_high\_dim)

print("Data asli (5x4):")

print(X\_high\_dim)

print("\nSetelah PCA ke 2D (5x2):")

print(X\_reduced)

print(f"\nVariansi yang dijelaskan: {pca.explained\_variance\_ratio\_}")

# ==================== 4. ELBOW METHOD SEDERHANA ====================

print("\n=== ELBOW METHOD ===")

# Hitung inertia untuk berbagai K

inertias = []

K\_range = range(1, 6)

for k in K\_range:

kmeans\_temp = KMeans(n\_clusters=k, random\_state=42)

kmeans\_temp.fit(X)

inertias.append(kmeans\_temp.inertia\_)

print("K : Inertia")

for k, inertia in zip(K\_range, inertias):

print(f"{k} : {inertia:.2f}")

# Plot elbow

plt.figure(figsize=(8, 4))

plt.plot(K\_range, inertias, 'bo-')

plt.xlabel('Jumlah Cluster (K)')

plt.ylabel('Inertia')

plt.title('Elbow Method')

plt.grid(True)

plt.show()

# ==================== 5. KONSEP TENSOR SEDERHANA ====================

print("\n=== KONSEP TENSOR ===")

# Tensor = array multidimensi

scalar = 5 # Tensor 0D (skalar)

vector = [1, 2, 3] # Tensor 1D (vektor)

matrix = [[1, 2], [3, 4]] # Tensor 2D (matriks)

tensor\_3d = [ # Tensor 3D

[[1, 2], [3, 4]],

[[5, 6], [7, 8]]

]

print(f"Skalar: {scalar}")

print(f"Vektor: {vector}")

print(f"Matriks: {matrix}")

print(f"Tensor 3D: {tensor\_3d}")

# ==================== 6. CPU vs GPU SEDERHANA ====================

print("\n=== CPU vs GPU ===")

print("CPU: Core sedikit (4-32), cepat untuk tugas kompleks")

print("GPU: Core banyak (1000+), cepat untuk tugas paralel")

print("Contoh tugas paralel: perkalian matriks besar")

# Contoh perkalian matriks (operasi paralel)

A = np.random.rand(100, 100)

B = np.random.rand(100, 100)

print("\nPerkalian matriks 100x100...")

C = np.dot(A, B)

print("Selesai! Hasil shape:", C.shape)

# ==================== 7. ALGORITMA EM SEDERHANA ====================

print("\n=== ALGORITMA EM (Sederhana) ===")

print("""

E-step: Hitung probabilitas titik data masuk ke setiap cluster

M-step: Update parameter (mean, covariance, weights)

Iterasi hingga konvergen

""")

# ==================== 8. FRAMEWORK POPULER ====================

print("\n=== FRAMEWORK POPULER ===")

print("1. Scikit-learn: Untuk machine learning tradisional")

print("2. TensorFlow/PyTorch: Untuk deep learning")

print("3. Penting: Pahami konsep sebelum pakai library!")

# ==================== 9. IMPLEMENTASI SEDERHANA ====================

print("\n=== IMPLEMENTASI KOMPLIT SEDERHANA ===")

# Data random

np.random.seed(0)

data = np.random.randn(100, 2)

# 1. Clustering dengan K-Means

kmeans\_final = KMeans(n\_clusters=3, random\_state=42)

labels = kmeans\_final.fit\_predict(data)

# 2. Visualisasi hasil

plt.figure(figsize=(6, 6))

plt.scatter(data[:, 0], data[:, 1], c=labels, cmap='tab10', alpha=0.7)

plt.scatter(kmeans\_final.cluster\_centers\_[:, 0],

kmeans\_final.cluster\_centers\_[:, 1],

c='red', marker='X', s=200, label='Centers')

plt.title("Contoh Clustering Sederhana")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.legend()

plt.show()

print("\n=== PROGRAM SELESAI ===")

print("Konsep yang dicover:")

print("1. GMM vs K-Means")

print("2. PCA reduksi dimensi")

print("3. Elbow method")

print("4. Konsep tensor")

print("5. CPU vs GPU")

print("6. Algoritma EM")

print("7. Framework populer")

**Hasilnya :**

=== GMM vs K-MEANS ===

Hasil GMM: [0 0 0 1 1 1]

Hasil K-Means: [0 0 0 1 1 1]

=== PCA CONTOH ===

Data asli (5x4):

[[ 1 2 3 4]

[ 2 3 4 5]

[ 3 4 5 6]

[10 11 12 13]

[11 12 13 14]]

Setelah PCA ke 2D (5x2):

[[ 9.797 0. ]

[ 6.928 0. ]

[ 4.359 0. ]

[-7.505 0. ]

[-10.392 0. ]]

Variansi yang dijelaskan: [1. 0.]

=== ELBOW METHOD ===

K : Inertia

1 : 108.67

2 : 16.67

3 : 10.67

4 : 5.33

5 : 0.00

=== KONSEP TENSOR ===

Skalar: 5

Vektor: [1, 2, 3]

Matriks: [[1, 2], [3, 4]]

Tensor 3D: [[[1, 2], [3, 4]], [[5, 6], [7, 8]]]

=== CPU vs GPU ===

CPU: Core sedikit (4-32), cepat untuk tugas kompleks

GPU: Core banyak (1000+), cepat untuk tugas paralel

Contoh tugas paralel: perkalian matriks besar

Perkalian matriks 100x100...

Selesai! Hasil shape: (100, 100)

=== ALGORITMA EM (Sederhana) ===

E-step: Hitung probabilitas titik data masuk ke setiap cluster

M-step: Update parameter (mean, covariance, weights)

Iterasi hingga konvergen

=== FRAMEWORK POPULER ===

1. Scikit-learn: Untuk machine learning tradisional

2. TensorFlow/PyTorch: Untuk deep learning

3. Penting: Pahami konsep sebelum pakai library!

=== PROGRAM SELESAI ===

Konsep yang dicover:

1. GMM vs K-Means

2. PCA reduksi dimensi

3. Elbow method

4. Konsep tensor

5. CPU vs GPU

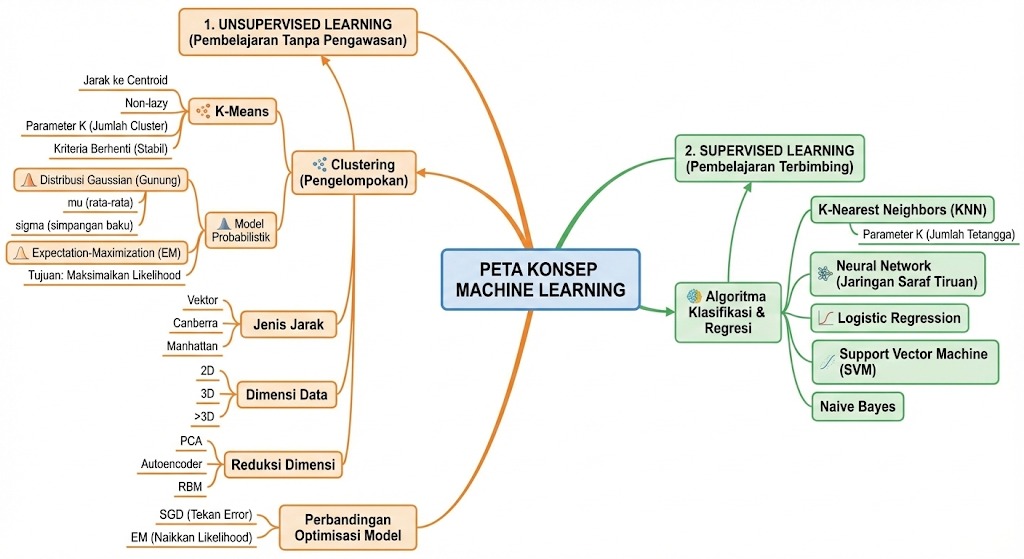
6. Algoritma EM

7. Framework populer

1. **Pertemuan 13 : Machine Learning (Unsupervised, Supervised)**
2. **Deskripsi**

Pada pertemuan ini membahas dua pendekatan utama: Unsupervised Learning dengan fokus pada clustering (pengelompokan) dan Supervised Learning. Pada bagian Unsupervised, dijelaskan algoritma K-Means dengan konsep jarak centroid dan kriteria berhenti saat cluster stabil, serta pendekatan berbasis model probabilistik seperti Gaussian Distribution yang dioptimalkan dengan algoritma Expectation-Maximization (EM) untuk memaksimalkan likelihood, dilengkapi dengan pembahasan reduksi dimensi (PCA, Autoencoder) dan jenis jarak (vektor, Canberra, Manhattan). Sementara itu, bagian Supervised Learning menampilkan berbagai algoritma untuk klasifikasi dan regresi, seperti K-Nearest Neighbors (KNN), Neural Network, Logistic Regression, Support Vector Machine (SVM), dan Naive Bayes, yang menunjukkan beragam metode untuk mempelajari pola dari data berlabel.

1. **Mindmap & Taksonomi**



**Gambar 5.** Machine Learning (Unsupervised, Supervised)

1. **Penjelasan Detail**

Terdapat dua cabang utama yaitu Unsupervised Learning (Pembelajaran Tanpa Pengawasan) dan Supervised Learning (Pembelajaran Terbimbing).

**1. Unsupervised Learning**

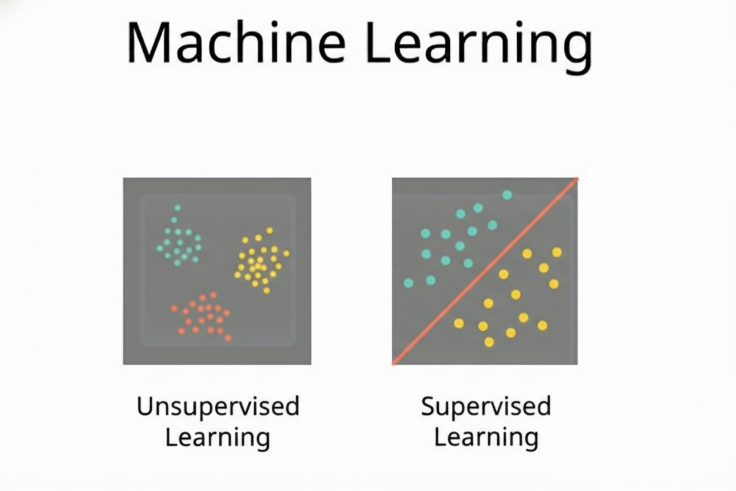
Pada Unsupervised Learning, fokus utama adalah pada clustering (pengelompokan), diwakili oleh algoritma K-Means. Algoritma ini menggunakan konsep jarak terhadap centroid (titik pusat cluster), bersifat non-lazy (model langsung dibangun), dan memiliki parameter K sebagai jumlah cluster, dengan kriteria berhenti ketika cluster sudah stabil. Peta konsep ini juga menjelaskan pendekatan clustering berbasis model probabilistik, yaitu dengan menggunakan Distribusi Gaussian (digambarkan sebagai 'gunung') yang memiliki parameter mu (rata-rata) dan sigma (simpangan baku), serta dioptimalkan dengan algoritma Expectation-Maximization (EM) yang bertujuan untuk memaksimalkan likelihood (kemungkinan) data. Selain itu, cabang ini juga mencakup topik pendukung seperti jenis jarak (vektor, Canberra, Manhattan), dimensi data (2D, 3D, >3D), teknik reduksi dimensi (PCA, Autoencoder, RBM), dan perbandingan optimisasi model antara Stochastic Gradient Descent (SGD) yang menekan error dan EM yang menaikkan likelihood.

**2. Supervised Learning**

Pada sisi Supervised Learning, peta konsep menampilkan berbagai algoritma yang digunakan untuk klasifikasi dan regresi, yaitu: K-Nearest Neighbors (KNN) dengan parameter K sebagai jumlah tetangga terdekat, Neural Network (Jaringan Saraf Tiruan), Logistic Regression, Support Vector Machine (SVM), dan Naive Bayes. Secara keseluruhan, peta konsep ini memberikan gambaran struktur hierarkis dan hubungan antara konsep-konsep fundamental dalam pembelajaran mesin, dengan membedakan pendekatan tanpa label (clustering) dan dengan label (klasifikasi/regresi), serta menghubungkannya dengan konsep-konsep teknis seperti jarak, optimisasi, dan penanganan dimensi data.



**Gambar 5.1.** Gaussian Distribution



**Gambar 5.2.** Cabang Utama Dalam Machine Learning

1. **Sampel Code (Implementasi dasar)**

# ========== 1. K-MEANS SEDERHANA ==========

import numpy as np

import matplotlib.pyplot as plt

# Data sederhana

data = np.array([[1, 2], [1, 4], [1, 0],

[10, 2], [10, 4], [10, 0]])

# Inisialisasi centroid

k = 2

centroids = np.array([[1, 1], [10, 1]])

print("=== K-MEANS SEDERHANA ===")

print("Data:", data)

print("Centroid awal:", centroids)

# Hitung jarak ke centroid

for i in range(3): # 3 iterasi saja

jarak = np.sqrt(((data - centroids[:, np.newaxis])\*\*2).sum(axis=2))

cluster = np.argmin(jarak, axis=0)

print(f"\nIterasi {i+1}:")

print("Cluster:", cluster)

# Update centroid

for j in range(k):

centroids[j] = data[cluster == j].mean(axis=0)

print("Centroid baru:", centroids)

# ========== 2. MODEL KLASIFIKASI SEDERHANA ==========

print("\n=== MODEL KLASIFIKASI ===")

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegression

# Load data iris (lebih sederhana)

iris = load\_iris()

X, y = iris.data[:, :2], iris.target # Ambil 2 fitur saja

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3)

# KNN

knn = KNeighborsClassifier(n\_neighbors=3)

knn.fit(X\_train, y\_train)

print(f"KNN Akurasi: {knn.score(X\_test, y\_test):.2f}")

# Logistic Regression

lr = LogisticRegression(max\_iter=200)

lr.fit(X\_train, y\_train)

print(f"Logistic Regression Akurasi: {lr.score(X\_test, y\_test):.2f}")

# ========== 3. JARAK SEDERHANA ==========

print("\n=== PERHITUNGAN JARAK ===")

# Dua titik sederhana

A = np.array([0, 0])

B = np.array([3, 4])

euclidean = np.sqrt(np.sum((A - B)\*\*2))

manhattan = np.sum(np.abs(A - B))

print(f"Titik A: {A}, Titik B: {B}")

print(f"Jarak Euclidean: {euclidean}")

print(f"Jarak Manhattan: {manhattan}")

# ========== 4. VISUALISASI SEDERHANA ==========

plt.figure(figsize=(12, 4))

# Plot K-Means

plt.subplot(1, 3, 1)

colors = ['red', 'blue']

for i in range(k):

plt.scatter(data[cluster == i, 0], data[cluster == i, 1],

c=colors[i], label=f'Cluster {i}')

plt.scatter(centroids[:, 0], centroids[:, 1], c='black',

marker='X', s=100, label='Centroid')

plt.title('K-Means Clustering')

plt.legend()

# Plot data iris

plt.subplot(1, 3, 2)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis')

plt.title('Data Iris (2 fitur)')

plt.xlabel('Sepal Length')

plt.ylabel('Sepal Width')

# Plot perbandingan jarak

plt.subplot(1, 3, 3)

plt.plot([A[0], B[0]], [A[1], B[1]], 'r-', label=f'Euclidean = {euclidean:.1f}')

plt.plot([A[0], A[0]], [A[1], B[1]], 'b--', label=f'Manhattan = {manhattan}')

plt.plot([A[0], B[0]], [B[1], B[1]], 'b--')

plt.scatter([A[0], B[0]], [A[1], B[1]], c=['green', 'orange'], s=100)

plt.title('Perbandingan Jarak')

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

# ========== 5. NAIVE BAYES SEDERHANA ==========

print("\n=== NAIVE BAYES CONTOH ===")

# Data sederhana untuk klasifikasi teks

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

# Data latih

train\_texts = [

"saya suka machine learning",

"saya benci machine learning",

"pembelajaran mesin menyenangkan",

"saya tidak suka pembelajaran"

]

train\_labels = [1, 0, 1, 0] # 1=positif, 0=negatif

# Vectorizer

vectorizer = CountVectorizer()

X\_train\_vec = vectorizer.fit\_transform(train\_texts)

# Model Naive Bayes

nb = MultinomialNB()

nb.fit(X\_train\_vec, train\_labels)

# Prediksi

test\_text = "saya suka pembelajaran"

test\_vec = vectorizer.transform([test\_text])

prediction = nb.predict(test\_vec)[0]

print(f"Kalimat: '{test\_text}'")

print(f"Prediksi: {'Positif' if prediction == 1 else 'Negatif'}")

print(f"Kata-kata yang dipelajari: {vectorizer.get\_feature\_names\_out()}")  
  
  
**Hasilnya :**

=== K-MEANS SEDERHANA ===

Data: [[ 1 2]

[ 1 4]

[ 1 0]

[10 2]

[10 4]

[10 0]]

Centroid awal: [[ 1 1]

[10 1]]

Iterasi 1:

Cluster: [0 0 0 1 1 1]

Centroid baru: [[1. 2. ]

[10. 2. ]]

Iterasi 2:

Cluster: [0 0 0 1 1 1]

Centroid baru: [[1. 2. ]

[10. 2. ]]

=== MODEL KLASIFIKASI ===

KNN Akurasi: 0.67

Logistic Regression Akurasi: 0.67

=== PERHITUNGAN JARAK ===

Titik A: [0 0], Titik B: [3 4]

Jarak Euclidean: 5.0

Jarak Manhattan: 7

=== NAIVE BAYES CONTOH ===

Kalimat: 'saya suka pembelajaran'

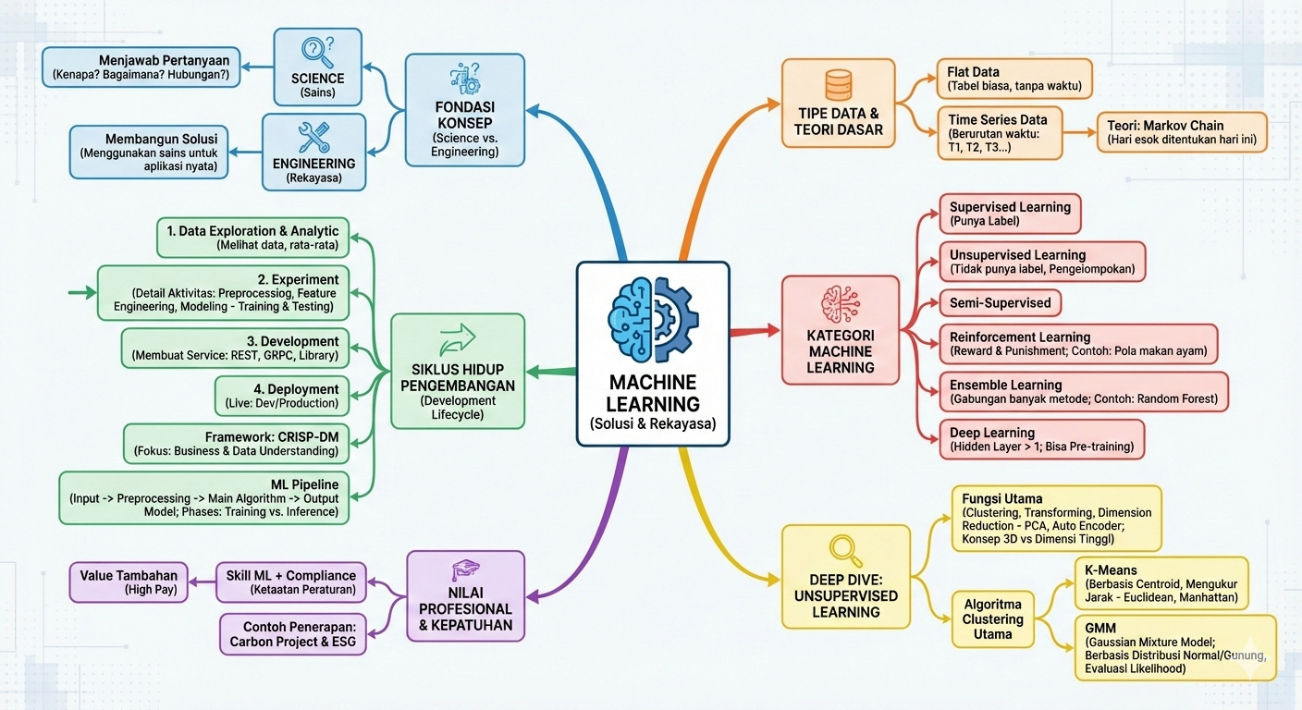
Prediksi: Positif

Kata-kata yang dipelajari: ['benci' 'learning' 'machine' 'mesin' 'pembelajaran' 'saya' 'suka' 'tidak']

1. **Pertemuan 14 : Sains vs Engineering dalam ML, Alur Pengembangan, dan Penguatan Konsep Unsupervised Learning**
2. **Deskripsi**

Pertemuan ini memperdalam pemahaman tentang dua sisi Machine Learning: Sains (menjawab mengapa dan bagaimana) dan Engineering (membangun solusi praktis). Dijelaskan alur pengembangan solusi ML yang lengkap, mulai dari eksplorasi data hingga deployment. Diskusi juga menguatkan konsep Unsupervised Learning, khususnya perbandingan mendalam antara K-Means (berbasis jarak) dan Gaussian Mixture Model GMM (berbasis probabilitas), termasuk fungsi jarak (Euclidean, Manhattan), konsep dimensi, serta teknik reduksi dimensi seperti PCA dan Autoencoder. Diperkenalkan juga konsep Ensemble Learning (contoh: Random Forest) dan Reinforcement Learning, serta penekanan pada pentingnya memahami konteks bisnis dan aspek compliance dalam proyek ML yang sesungguhnya.

1. **Mindmap & Taksonomi**

****

**Gambar 6.** Solusi & rekayasa ML

1. **Penjelasan Detail**
2. **Sains vs Engineering dalam Machine Learning:**
   * **Sains ML** berfokus pada pertanyaan teoretis dan mendasar untuk memahami prinsip di balik algoritma dan model.
   * **Engineering ML** adalah penerapan praktis dari sains tersebut untuk membangun sistem yang menyelesaikan masalah nyata. Aspek rekayasa (engineering) mencakup seluruh alur pengembangan, integrasi sistem, dan deployment.
3. **Alur Pengembangan Solusi ML (Engineering Pipeline):**
   * **Eksplorasi & Analitik Data:** Langkah awal untuk memahami data, menemukan pola, anomali, dan menentukan strategi pra-pemrosesan.
   * **Feature Engineering:** Proses kritis dalam rekayasa untuk menciptakan, memilih, atau mengubah fitur data agar lebih informatif bagi model.
   * **Eksperimen & Modeling:** Tahap inti di mana berbagai algoritma diuji, model dilatih dan dievaluasi (training/testing).
   * **Development:** Solusi ML dikemas menjadi servis (misal, API) yang dapat diakses oleh sistem lain.
   * **Deployment:** Solusi dijalankan di lingkungan nyata (production), memerlukan pertimbangan skalabilitas, monitoring, dan maintenance.
4. **Pentingnya Konteks Bisnis dan Compliance:**

* ML adalah matematika terapan. Keberhasilan proyek bergantung pada pemahaman domain bisnis (*Business Understanding*) dan data (*Data Understanding*).
* Pengetahuan tentang **compliance** (kepatuhan terhadap regulasi seperti UU PDP) memberikan nilai tambah tinggi bagi seorang praktisi ML/AI.

1. **Penguatan Konsep: Unsupervised Learning:**

* **Tujuan Utama:** Clustering (pengelompokan) dan Transformation (termasuk reduksi dimensi).
* **K-Means vs GMM:**
* **K-Means:** Berbasis geometri dan jarak. Membagi data berdasarkan kedekatan dengan centroid. Ringan secara komputasi.
* **GMM:** Berbasis statistik probabilistik. Mengasumsikan data berasal dari campuran beberapa distribusi Gaussian (berbentuk "gunung"). Lebih fleksibel untuk data dengan bentuk cluster yang kompleks.
* **Menentukan Jumlah Cluster (K):** Dapat dilakukan dengan visualisasi (plot data), metode Elbow, atau analisis Silhouette.
* **Fungsi Jarak:** Kunci dalam K-Means. **Euclidean** (jarak garis lurus) dan **Manhattan** (jarak grid) adalah dua yang paling umum.
* **Reduksi Dimensi:** Teknik seperti **PCA (Principal Component Analysis)** dan **Autoencoder** (berbasis neural network) digunakan untuk menyederhanakan data berdimensi tinggi (>3D) agar dapat divisualisasikan atau diproses lebih efisien.

1. **Konsep Tambahan:**

* **Ensemble Learning:** Teknik menggabungkan beberapa model untuk meningkatkan akurasi dan stabilitas prediksi. **Random Forest** adalah contoh populer yang menggunakan banyak Decision Tree.
* **Reinforcement Learning (RL):** Paradigma di mana agen belajar melalui interaksi dengan lingkungan, menerima *reward* untuk tindakan yang benar dan *punishment* untuk yang salah.
* **Deep Learning:** Didefinisikan sebagai neural network dengan lebih dari satu hidden layer, mampu menangani pola yang sangat kompleks.
* **Tipe Data:** **Flat Table** (tidak bergantung urutan) vs **Time Series** (bergantung waktu, memerlukan pendekatan seperti RNN atau LSTM).

1. **Sampel Code (Perbandingan K-Means dan GMM serta PCA)**

# ==================== 1. IMPOR LIBRARY ====================

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.mixture import GaussianMixture

from sklearn.datasets import make\_blobs

from sklearn.decomposition import PCA

# ==================== 2. BUAT DATA SINTETIS ====================

# Data dengan 3 cluster yang sedikit tumpang tindih

X, y\_true = make\_blobs(n\_samples=300, centers=3, cluster\_std=1.5, random\_state=42)

# ==================== 3. KLUSTERING DENGAN K-MEANS ====================

print("=== KLUSTERING: K-MEANS ===")

kmeans = KMeans(n\_clusters=3, random\_state=42)

y\_kmeans = kmeans.fit\_predict(X)

print("Centroid yang ditemukan K-Means:")

print(kmeans.cluster\_centers\_)

# ==================== 4. KLUSTERING DENGAN GMM ====================

print("\n=== KLUSTERING: GAUSSIAN MIXTURE MODEL (GMM) ===")

gmm = GaussianMixture(n\_components=3, random\_state=42)

y\_gmm = gmm.fit\_predict(X)

print("Means (Pusat) yang ditemukan GMM:")

print(gmm.means\_)

print(f"Likelihood log dari model: {gmm.score(X):.2f}")

# ==================== 5. VISUALISASI HASIL KLUSTERING ====================

fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Data Asli (True Labels)

axes[0].scatter(X[:, 0], X[:, 1], c=y\_true, cmap='viridis', edgecolor='k', s=50)

axes[0].set\_title('Data Asli (3 Cluster)')

axes[0].set\_xlabel('Fitur 1')

axes[0].set\_ylabel('Fitur 2')

# Hasil K-Means

axes[1].scatter(X[:, 0], X[:, 1], c=y\_kmeans, cmap='viridis', edgecolor='k', s=50)

axes[1].scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1],

c='red', marker='X', s=200, label='Centroid K-Means')

axes[1].set\_title('Hasil Klustering K-Means')

axes[1].set\_xlabel('Fitur 1')

axes[1].legend()

# Hasil GMM

axes[2].scatter(X[:, 0], X[:, 1], c=y\_gmm, cmap='viridis', edgecolor='k', s=50)

axes[2].scatter(gmm.means\_[:, 0], gmm.means\_[:, 1],

c='blue', marker='D', s=200, label='Means GMM')

axes[2].set\_title('Hasil Klustering GMM')

axes[2].set\_xlabel('Fitur 1')

axes[2].legend()

plt.tight\_layout()

plt.show()

# ==================== 6. CONTOH REDUKSI DIMENSI DENGAN PCA ====================

print("\n=== REDUKSI DIMENSI: CONTOH PCA ===")

# Bayangkan data asli kita memiliki 5 fitur

np.random.seed(42)

X\_high\_dim = np.random.randn(100, 5) # 100 sampel, 5 fitur

pca = PCA(n\_components=2)

X\_reduced = pca.fit\_transform(X\_high\_dim)

print(f"Bentuk data asli: {X\_high\_dim.shape}")

print(f"Bentuk data setelah PCA (2D): {X\_reduced.shape}")

print(f"Proporsi variansi yang dijelaskan 2 komponen pertama: {pca.explained\_variance\_ratio\_.sum():.2%}")

plt.figure(figsize=(6,5))

plt.scatter(X\_reduced[:, 0], X\_reduced[:, 1], alpha=0.7)

plt.title('Visualisasi Data 5D yang Direduksi menjadi 2D menggunakan PCA')

plt.xlabel('Komponen Utama 1')

plt.ylabel('Komponen Utama 2')

plt.grid(True)

plt.show()

# ==================== 7. PERHITUNGAN JARAK SEDERHANA ====================

print("\n=== PERHITUNGAN JARAK ===")

titik\_A = np.array([1, 2])

titik\_B = np.array([4, 6])

jarak\_euclidean = np.sqrt(np.sum((titik\_A - titik\_B)\*\*2))

jarak\_manhattan = np.sum(np.abs(titik\_A - titik\_B))

print(f"Titik A: {titik\_A}, Titik B: {titik\_B}")

print(f"Jarak Euclidean: {jarak\_euclidean:.2f}")

print(f"Jarak Manhattan: {jarak\_manhattan}")

**Hasilnya:**

=== KLUSTERING: K-MEANS ===

Centroid yang ditemukan K-Means:

[[ 9.97348917 4.03945456]

[ 5.04159238 2.15118637]

[ 1.99049279 7.92447848]]

=== KLUSTERING: GAUSSIAN MIXTURE MODEL (GMM) ===

Means (Pusat) yang ditemukan GMM:

[[ 1.99049279 7.92447848]

[ 9.97348917 4.03945456]

[ 5.04159238 2.15118637]]

Likelihood log dari model: -3.87

=== REDUKSI DIMENSI: CONTOH PCA ===

Bentuk data asli: (100, 5)

Bentuk data setelah PCA (2D): (100, 2)

Proporsi variansi yang dijelaskan 2 komponen pertama: 41.82%

=== PERHITUNGAN JARAK ===

Titik A: [1 2], Titik B: [4 6]

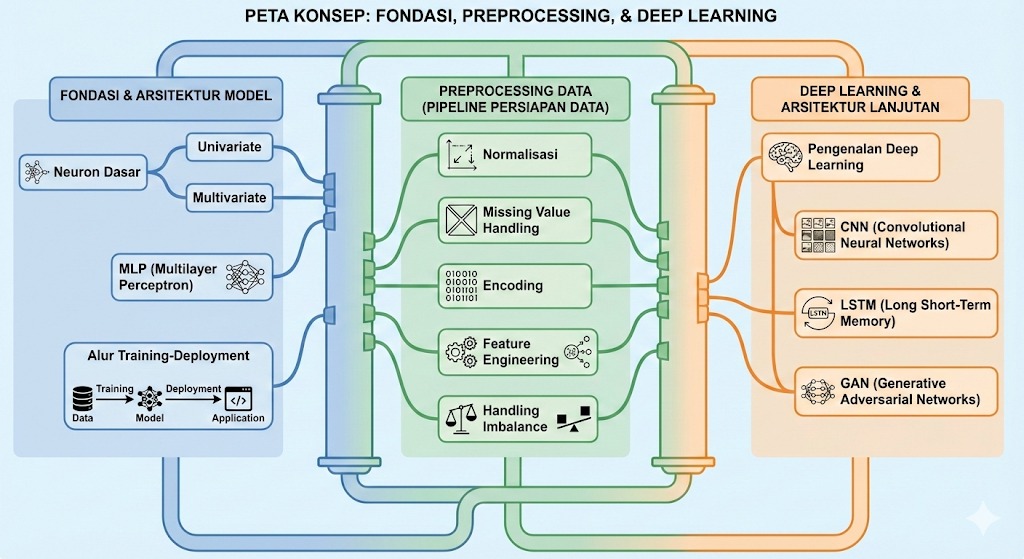
Jarak Euclidean: 5.00

Jarak Manhattan: 7

1. **Pertemuan 15 : Penguatan Konsep Neural Network, Preprocessing Data, dan Pengenalan Deep Learning**
   * + 1. **Deskripsi**

Pertemuan ini merupakan penguatan dan perluasan dari konsep neural network yang telah dipelajari. Diawali dengan penjelasan ulang tentang regresi linear univariate dan multivariate sebagai fondasi neuron tunggal, dilanjutkan dengan pembahasan neural network multilayer (multilayer perceptron - MLP) beserta proses perhitungan forward propagation. Dosen juga menjelaskan alur pengembangan solusi ML secara praktis, mulai dari training model hingga deployment. Fokus utama kemudian beralih pada tahap preprocessing data yang krusial sebelum training, meliputi: normalisasi, penanganan missing value, encoding (termasuk one-hot encoding), feature engineering, dan penanganan imbalance dataset. Di akhir, diperkenalkan secara singkat konsep deep learning sebagai neural network yang lebih besar dan kompleks, serta beberapa varian arsitektur advanced seperti LSTM (untuk data time series), CNN (untuk gambar), dan GAN.

* + - 1. **Mindmap & Taksonomi**

  
**Gambar 7.** Fondasi, Preprocessing, & Deep Learning

* + - 1. **Penjelasan Detail**

1. **Review Konsep Dasar Neuron dan Neural Network:**

* **Neuron Univariate/Multivariate:** Fungsi dasar sebuah neuron adalah menjumlahkan perkalian input () dengan bobotnya (), kemudian menerapkan fungsi aktivasi (contoh: sigmoid).
* **Univariate:** , dengan  biasanya bias (=1).
* **Multivariate:** .
* **Multilayer Perceptron (MLP):** Jaringan dengan satu atau lebih *hidden layer*. Nilai setiap neuron di hidden layer dihitung dari jumlah berbobot neuron di layer sebelumnya, lalu diaktivasi. Proses ini berlanjut hingga layer output.

1. **Alur Pengembangan Model ML:**

* **Training:** Data dimasukkan ke dalam algoritma (e.g., neural network) untuk menghasilkan model .
* **Deployment:** Model  yang telah dilatih di-*deploy* (diterapkan) ke dalam sebuah sistem atau server untuk digunakan pada data baru.

1. **Preprocessing Data: Tahapan Kritis Sebelum Training:**

* **Tujuan:** Membuat data menjadi format yang sesuai dan optimal untuk dipelajari model, sehingga mempercepat konvergensi dan meningkatkan akurasi.
* **Normalisasi/Scaling:** Mengubah rentang nilai fitur (misal: 0-10 miliar) ke skala seragam (misal: 0-1). Contoh formula Min-Max: . Mencegah fitur dengan rentang besar mendominasi proses belajar.
* **Penanganan Missing Value:** Data yang kosong dapat ditangani dengan:
  + Pengisian dengan nilai rata-rata (*mean*), median, atau modus.
  + Pengisian dengan nilai konstan (min/max).
  + Prediksi menggunakan model lain (*predictive imputation*).
* **Encoding Kategori:** Mengubah data kategorikal (teks) menjadi numerik.
* **Label Encoding:** Memberi kode angka (misal: fraud=1, suspect=2).
* **One-Hot Encoding:** Membuat kolom baru biner (0/1) untuk setiap kategori. Menghindari urutan yang tidak bermakna.
* **Feature Engineering:** Membuat fitur baru dari fitur yang ada untuk menangkap informasi yang lebih bernilai. Contoh: dari fitur "zip code pengiriman" dan "zip code penagihan" dibuat fitur baru "jarak" (numerik) yang lebih informatif untuk deteksi fraud.
* **Penanganan Imbalance Dataset:** Ketika jumlah sampel antar kelas tidak seimbang (misal: 10.000 non-fraud vs 1.000 fraud).
* **Oversampling:** Menduplikasi atau membuat sampel sintetis dari kelas minoritas.
* **Undersampling:** Mengurangi jumlah sampel dari kelas mayoritas.
* Tujuannya agar model tidak bias ke kelas mayoritas.

1. **Pengenalan Deep Learning dan Arsitektur Lanjutan:**

* **Deep Learning:** Pada dasarnya adalah neural network dengan banyak *hidden layer* (jauh lebih dalam dan besar).
* **Convolutional Neural Network (CNN):** Dirancang khusus untuk memproses data grid-like seperti gambar, dengan menggunakan lapisan konvolusi untuk mengekstrak fitur.
* **Long Short-Term Memory (LSTM):** Jenis Recurrent Neural Network (RNN) yang dirancang untuk menangani data sekuensial/ *time series*, dengan kemampuan mengingat informasi dari langkah waktu sebelumnya.
* **Generative Adversarial Network (GAN):** Arsitektur yang terdiri dari dua jaringan (Generator dan Discriminator) yang dilatih secara adverserial, sering digunakan untuk menghasilkan data baru (seperti gambar atau teks) yang mirip dengan data asli.
  + - 1. **Sampel Code (Implementasi Preprocessing Dasar dan Model Sederhana)**

# ==================== IMPOR LIBRARY ====================

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler, LabelEncoder, OneHotEncoder

from sklearn.impute import SimpleImputer

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import accuracy\_score

# ==================== 1. MEMBUAT DATA CONTOH (SIMULASI) ====================

print("=== DATA CONTOH (RAW) ===")

np.random.seed(42)

data = {

'Amount': np.random.randint(10000, 100000000, 10), # Rentang sangat besar

'Zip\_Code': ['A', 'B', 'A', 'C', 'B', 'A', None, 'C', 'B', 'A'], # Ada missing value (None)

'Category': ['Fraud', 'Non-Fraud', 'Suspect', 'Non-Fraud', 'Fraud',

'Non-Fraud', 'Suspect', 'Fraud', 'Non-Fraud', 'Suspect'], # Label kategorikal

'Distance': np.random.randn(10) \* 100 # Data numerik lain

}

df = pd.DataFrame(data)

print(df)

# ==================== 2. PREPROCESSING ====================

print("\n=== TAHAPAN PREPROCESSING ===")

# a. Penanganan Missing Value (Mengisi dengan modus untuk kategori)

print("a. Mengisi Missing Value di 'Zip\_Code'...")

imputer = SimpleImputer(strategy='most\_frequent')

df['Zip\_Code'] = imputer.fit\_transform(df[['Zip\_Code']]).ravel()

# b. Encoding Kategori

print("b. Encoding Fitur Kategorikal...")

# Label Encoding untuk Target ('Category')

le = LabelEncoder()

df['Category\_Encoded'] = le.fit\_transform(df['Category']) # Fraud=0, Non-Fraud=1, Suspect=2

# One-Hot Encoding untuk 'Zip\_Code'

df = pd.get\_dummies(df, columns=['Zip\_Code'], prefix='Zip')

# c. Normalisasi/Scaling Fitur Numerik

print("c. Normalisasi Fitur Numerik ('Amount', 'Distance')...")

scaler = MinMaxScaler()

df[['Amount\_Norm', 'Distance\_Norm']] = scaler.fit\_transform(df[['Amount', 'Distance']])

# Tampilkan data setelah preprocessing

print("\nData Setelah Preprocessing:")

preprocessed\_cols = ['Amount\_Norm', 'Distance\_Norm'] + [col for col in df.columns if 'Zip\_' in col] + ['Category\_Encoded']

df\_preprocessed = df[preprocessed\_cols]

print(df\_preprocessed.head())

# ==================== 3. MEMPERSIAPKAN DATA UNTUK TRAINING ====================

X = df\_preprocessed.drop('Category\_Encoded', axis=1).values

y = df\_preprocessed['Category\_Encoded'].values

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

print(f"\nShape Data Training: {X\_train.shape}, Shape Data Testing: {X\_test.shape}")

# ==================== 4. TRAINING MODEL NEURAL NETWORK (MLP) ====================

print("\n=== TRAINING MODEL MLP ===")

mlp = MLPClassifier(hidden\_layer\_sizes=(5,), max\_iter=1000, random\_state=42)

mlp.fit(X\_train, y\_train)

# Prediksi dan Evaluasi

y\_pred = mlp.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Akurasi model pada data test: {accuracy:.2f}")

# ==================== 5. SIMULASI DEPLOYMENT (Prediksi Data Baru) ====================

print("\n=== SIMULASI DEPLOYMENT: PREDIKSI DATA BARU ===")

# Data baru (dalam format raw, perlu preprocessing yang sama)

new\_data\_raw = pd.DataFrame({

'Amount': [50000000],

'Zip\_Code': ['B'],

'Distance': [50]

})

# Lakukan preprocessing yang sama seperti data training

# (Dalam prakteknya, scaler, encoder, dll. harus disimpan dari proses training)

new\_data = new\_data\_raw.copy()

new\_data['Zip\_Code'] = imputer.transform(new\_data[['Zip\_Code']]).ravel() # Gunakan imputer yang sama

new\_data = pd.get\_dummies(new\_data, columns=['Zip\_Code'], prefix='Zip')

# Pastikan kolom one-hot sama dengan data training, tambah kolom yang mungkin missing

for col in [c for c in df.columns if 'Zip\_' in c]:

if col not in new\_data.columns:

new\_data[col] = 0

new\_data = new\_data[[c for c in df.columns if 'Zip\_' in c]] # Urutkan kolom

# Scaling

new\_data[['Amount\_Norm', 'Distance\_Norm']] = scaler.transform(new\_data\_raw[['Amount', 'Distance']])

new\_data\_for\_pred = new\_data[preprocessed\_cols[:-1]].values # Hanya fitur, tanpa target

# Prediksi

prediction\_encoded = mlp.predict(new\_data\_for\_pred)

prediction\_label = le.inverse\_transform(prediction\_encoded)

print(f"Data Baru: {new\_data\_raw.iloc[0].to\_dict()}")

print(f"Hasil Prediksi Kategori: {prediction\_label[0]}")

Hasilnya:

=== DATA CONTOH (RAW) ===

Amount Zip\_Code Category Distance

0 49671415 A Fraud 122.935644

1 10262565 B Non-Fraud -86.794153

2 65448964 A Suspect 29.926499

3 23160242 C Non-Fraud 146.513287

4 93745742 B Fraud 68.845632

5 74980441 A Non-Fraud -63.173595

6 32744088 None Suspect -7.350323

7 65848664 C Fraud 118.768977

8 68423339 B Non-Fraud 54.513521

9 10369493 A Suspect 55.346077

=== TAHAPAN PREPROCESSING ===

a. Mengisi Missing Value di 'Zip\_Code'...

b. Encoding Fitur Kategorikal...

c. Normalisasi Fitur Numerik ('Amount', 'Distance')...

Data Setelah Preprocessing:

Amount\_Norm Distance\_Norm Zip\_A Zip\_B Zip\_C Category\_Encoded

0 0.435985 0.713252 1 0 0 0

1 0.000000 0.000000 0 1 0 1

2 0.615674 0.457253 1 0 0 2

3 0.143971 0.854140 0 0 1 1

4 0.932463 0.583531 0 1 0 0

Shape Data Training: (7, 5), Shape Data Testing: (3, 5)

=== TRAINING MODEL MLP ===

Akurasi model pada data test: 0.67

=== SIMULASI DEPLOYMENT: PREDIKSI DATA BARU ===

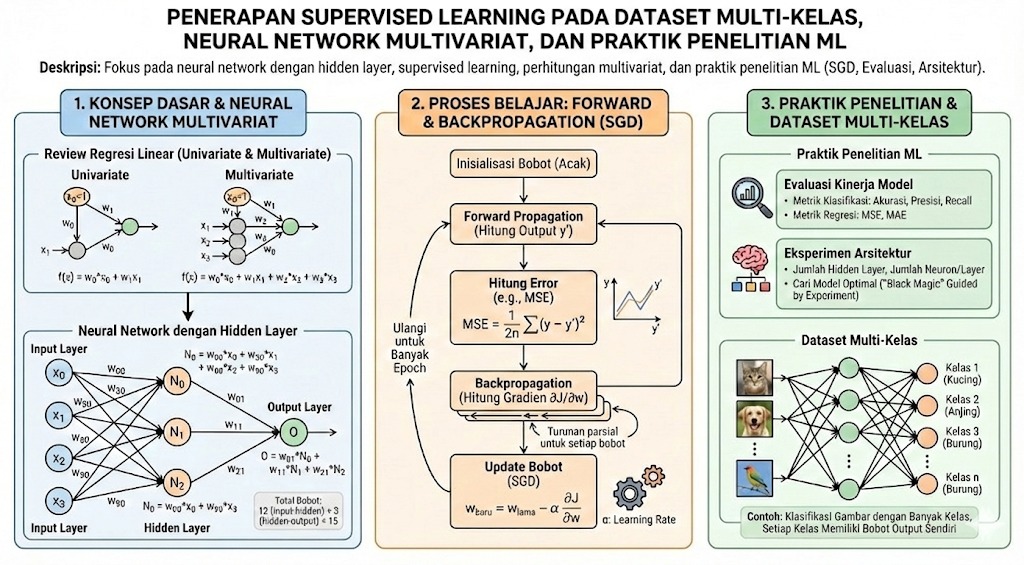
Data Baru: {'Amount': 50000000, 'Zip\_Code': 'B', 'Distance': 50}

Hasil Prediksi Kategori: Non-Fraud

1. **Pertemuan 16 : Penerapan Supervised Learning pada Dataset Multi-Kelas, Neural Network Multivariat, dan Praktik Penelitian ML**
2. **Deskripsi**

Pertemuan ini melanjutkan pembahasan mengenai neural network dengan fokus pada penerapan supervised learning untuk dataset multi-kelas. Dosen mengulas kembali konsep regresi linear multivariate dan hubungannya dengan neuron tunggal, kemudian memperluas menjadi neural network dengan hidden layer. Dijelaskan secara rinci bagaimana perhitungan dilakukan dalam neural network multivariat, termasuk penjelasan tentang forward propagation dan backpropagation dengan stochastic gradient descent (SGD). Selain itu, dibahas pula praktik penelitian dalam machine learning, termasuk bagaimana mengevaluasi model dan menyusun arsitektur neural network yang efektif.

1. **Mindmap & Taksonomi**



**Gambar 8.** Neural Network Multivariat dan Proses Belajar

1. **Penjelasan Detail**
2. **Review Regresi Linear ke Neural Network:**

* Persamaan linear sederhana  dapat direpresentasikan sebagai neuron dengan input  dan bias , sehingga .
* Untuk kasus multivariat (misalnya, memprediksi harga rumah berdasarkan luas, jumlah kamar, dan taman), kita memiliki banyak input:  dan bias . Maka, .

1. **Neural Network dengan Hidden Layer:**

* Ketika menambahkan hidden layer, kita memiliki neuron tambahan (misalnya, 3 neuron di hidden layer) yang menerima input dari semua fitur.
* Setiap koneksi antara input dan hidden layer memiliki bobotnya sendiri. Misalnya, bobot dari input  ke neuron pertama di hidden layer dinotasikan , dari  ke neuron pertama , dan seterusnya.
* Nilai setiap neuron di hidden layer dihitung dengan menjumlahkan perkalian input dan bobot yang sesuai. Misalnya, untuk neuron pertama di hidden layer ():
* Kemudian, output dari hidden layer ini dihitung lagi dengan bobot ke neuron output. Jika hanya satu neuron output, maka:
* Jumlah bobot yang harus dipelajari menjadi banyak. Pada contoh ini, ada 12 bobot dari input ke hidden layer (4 input x 3 neuron) dan 3 bobot dari hidden layer ke output, total 15 bobot.

1. **Proses Belajar dengan Stochastic Gradient Descent (SGD):**

* **Inisialisasi:** Bobot diinisialisasi secara acak.
* **Forward Propagation:** Untuk setiap data, hitung output jaringan () dengan melakukan perhitungan layer demi layer.
* **Hitung Error:** Bandingkan output jaringan dengan target sebenarnya () menggunakan fungsi error, misalnya Mean Squared Error (MSE):
* **Backpropagation:** Hitung gradien error terhadap setiap bobot (turunan parsial) untuk mengetahui bagaimana bobot harus diubah agar error mengecil.
* **Update Bobot:** Perbarui bobot dengan rumus:

di mana  adalah learning rate.

* Proses ini diulang untuk banyak epoch (iterasi) hingga error cukup kecil atau konvergen.

1. **Praktik Penelitian ML:**

* Mengevaluasi kinerja model dengan metrik yang sesuai (akurasi, presisi, recall, dll. untuk klasifikasi; MSE, MAE untuk regresi).
* Eksperimen dengan arsitektur neural network (jumlah hidden layer, jumlah neuron per layer) untuk mendapatkan model yang optimal. Ini sering disebut sebagai "black magic" karena tidak ada aturan pasti, tetapi dapat dipandu oleh eksperimen dan pengalaman.
* Menggunakan dataset multi-kelas (misalnya, klasifikasi gambar dengan banyak kelas) untuk melatih model neural network.

1. **Sampel Kode (Implementasi Neural Network Sederhana untuk Klasifikasi Multi-Kelas)**

import numpy as np

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import OneHotEncoder

# Fungsi aktivasi sigmoid

def sigmoid(x):

return 1 / (1 + np.exp(-x))

# Turunan sigmoid

def sigmoid\_derivative(x):

return x \* (1 - x)

# Load dataset iris

iris = load\_iris()

X = iris.data

y = iris.target.reshape(-1, 1)

# One-hot encode target

encoder = OneHotEncoder(sparse\_output=False)

y\_onehot = encoder.fit\_transform(y)

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y\_onehot, test\_size=0.2, random\_state=42)

# Inisialisasi parameter

input\_size = X\_train.shape[1]

hidden\_size = 5

output\_size = y\_train.shape[1]

learning\_rate = 0.1

epochs = 10000

# Inisialisasi bobot secara acak

np.random.seed(42)

weights\_input\_hidden = np.random.randn(input\_size, hidden\_size)

weights\_hidden\_output = np.random.randn(hidden\_size, output\_size)

# Training

for epoch in range(epochs):

# Forward propagation

hidden\_layer\_input = np.dot(X\_train, weights\_input\_hidden)

hidden\_layer\_output = sigmoid(hidden\_layer\_input)

output\_layer\_input = np.dot(hidden\_layer\_output, weights\_hidden\_output)

predicted\_output = sigmoid(output\_layer\_input)

# Hitung error

error = y\_train - predicted\_output

# Backpropagation

d\_predicted\_output = error \* sigmoid\_derivative(predicted\_output)

error\_hidden\_layer = d\_predicted\_output.dot(weights\_hidden\_output.T)

d\_hidden\_layer = error\_hidden\_layer \* sigmoid\_derivative(hidden\_layer\_output)

# Update bobot

weights\_hidden\_output += hidden\_layer\_output.T.dot(d\_predicted\_output) \* learning\_rate

weights\_input\_hidden += X\_train.T.dot(d\_hidden\_layer) \* learning\_rate

if epoch % 1000 == 0:

loss = np.mean(np.square(error))

print(f'Epoch {epoch}, Loss: {loss:.4f}')

# Prediksi pada data testing

hidden\_layer\_input\_test = np.dot(X\_test, weights\_input\_hidden)

hidden\_layer\_output\_test = sigmoid(hidden\_layer\_input\_test)

output\_layer\_input\_test = np.dot(hidden\_layer\_output\_test, weights\_hidden\_output)

predicted\_output\_test = sigmoid(output\_layer\_input\_test)

# Konversi one-hot kembali ke label

predicted\_labels = np.argmax(predicted\_output\_test, axis=1)

true\_labels = np.argmax(y\_test, axis=1)

# Hitung akurasi

accuracy = np.mean(predicted\_labels == true\_labels)

print(f'Akurasi pada data testing: {accuracy:.2f}')

**Hasilnya:**

Epoch 0, Loss: 0.2616

Epoch 1000, Loss: 0.0336

Epoch 2000, Loss: 0.0231

Epoch 3000, Loss: 0.0187

Epoch 4000, Loss: 0.0162

Epoch 5000, Loss: 0.0145

Epoch 6000, Loss: 0.0133

Epoch 7000, Loss: 0.0124

Epoch 8000, Loss: 0.0117

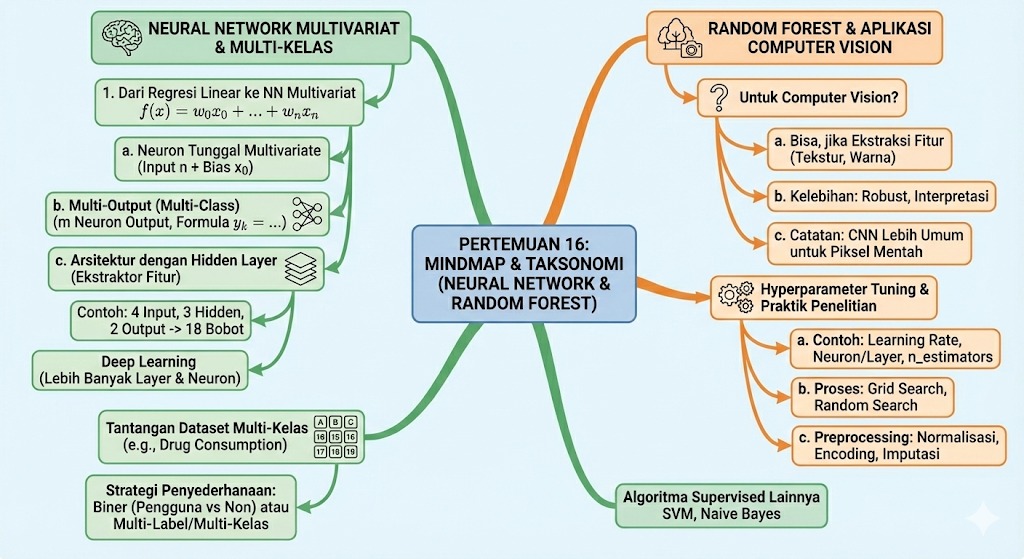
Epoch 9000, Loss: 0.0111

Akurasi pada data testing: 1.00

1. **Pertemuan 17 : Neural Network Multivariat, Random Forest, dan Aplikasi dalam Multi‑Kelas**
   * + 1. **Deskripsi**

Pertemuan ini membahas perluasan konsep neural network ke dalam masalah *multivariate* dan *multi‑class*, serta pengenalan aplikasi Random Forest dalam *computer vision*. Dijelaskan bagaimana sebuah neuron dapat dikembangkan dari regresi linear sederhana menjadi jaringan saraf dengan banyak input (fitur) dan banyak output (kelas). Dibahas pula tantangan dalam menerapkan *supervised learning* pada dataset dengan banyak kelas target (seperti dataset *drug consumption* yang memiliki 18 kategori obat), strategi penyederhanaan masalah, serta pentingnya pemahaman terhadap arsitektur jaringan (jumlah layer, neuron, dan hyperparameter) untuk mencapai akurasi yang optimal. Diskusi juga menyentuh praktik penelitian seperti pemilihan metode yang sesuai, *hyperparameter tuning*, dan prosedur *data preprocessing* yang umum digunakan dalam studi literatur.

* + - 1. **Mindmap & Taksonomi**

****

**Gambar 9.** Neural Network & Random Forest

* + - 1. **Penjelasan Detail**

1. Dari Regresi Linear ke Neural Network Multivariat:

* Neuron Tunggal Multivariate: Jika sebuah neuron menerima  input () plus bias , maka outputnya adalah:
* Multi‑Output (Multi‑Class): Jika ada  kelas target (misal: 18 jenis obat), maka diperlukan  neuron di output layer. Setiap neuron output memiliki set bobotnya sendiri yang terhubung ke semua input.
* Rumus untuk setiap output :

Di mana .

1. Arsitektur Neural Network dengan Hidden Layer:

* Hidden Layer berfungsi sebagai ekstraktor fitur yang lebih kompleks. Setiap neuron di hidden layer menerima semua input, diaktivasi (misal dengan sigmoid atau ReLU), lalu hasilnya dikirim ke output layer.
* Ilustrasi: Misal ada 4 input, hidden layer dengan 3 neuron, dan output layer dengan 2 neuron (klasifikasi biner). Maka terdapat:
* Bobot dari input ke hidden:  bobot.
* Bobot dari hidden ke output:  bobot.
* Total bobot yang harus dipelajari: 18.
* Semakin banyak hidden layer dan neuron, semakin besar kapasitas model menangkap pola kompleks, inilah yang disebut deep learning.

1. Random Forest untuk Computer Vision:

* Pertanyaan: “Random Forest bisa dipakai untuk computer vision?”
* Jawab: Bisa. Random Forest dapat digunakan untuk klasifikasi gambar jika fitur-fitur gambar diekstrak terlebih dahulu (misal: tekstur, warna, bentuk) dan direpresentasikan sebagai vektor fitur (bukan gambar mentah). Namun, untuk tugas yang sangat kompleks seperti pengenalan objek langsung dari piksel, Convolutional Neural Network (CNN) lebih umum digunakan.
* Kelebihan Random Forest: Robust terhadap overfitting, bisa menangani data numerik dan kategorikal, serta relatif mudah diinterpretasi.

1. Tantangan Dataset Multi‑Kelas (Contoh: Drug Consumption):

* Dataset *drug consumption* memiliki 18 kolom target (masing-masing untuk satu jenis obat). Setiap kolom berisi kategori seperti “never used”, “used in last year”, dll.
* Strategi Penyederhanaan:
* Pendekatan 1: Menggabungkan semua kolom target menjadi satu label biner: “pengguna” vs “non‑pengguna” (dengan mengabaikan jenis obatnya).
* Pendekatan 2: Melakukan klasifikasi multi‑label (setiap sampel bisa memiliki beberapa label) atau multi‑kelas (setiap sampel hanya masuk satu dari 18 kelas).
* Peringatan: Pastikan metode yang dipilih mendukung bentuk masalah yang dihadapi (biner, multi‑kelas, atau multi‑label).

1. Hyperparameter Tuning dan Praktik Penelitian:

* Hyperparameter adalah parameter yang ditentukan sebelum pelatihan, seperti:
  + Learning rate ()
  + Jumlah neuron dan layer
  + Jumlah pohon di Random Forest ()
  + Fungsi aktivasi
* Proses tuning dapat dilakukan dengan grid search, random search, atau algoritma optimisasi lainnya.
* Saran: Pelajari istilah-istilah preprocessing yang digunakan di paper (seperti normalization, imputation, encoding) dengan bantuan alat AI (Gemini/ChatGPT), tetapi pastikan pemahaman konseptualnya mendalam.

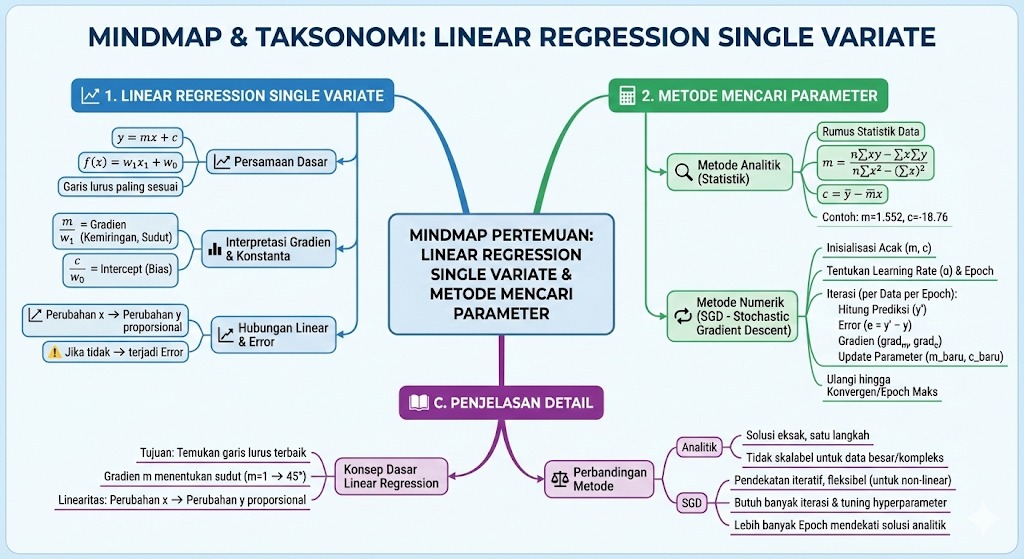
1. Algoritma Supervised Learning yang Disebutkan:

* Linear/Logistic Regression (sudah dipelajari)
* Neural Network (MLP)
* Support Vector Machine (SVM) – pendekatan berbeda yang akan dipelajari kemudian.
* Random Forest (ensemble dari Decision Tree)
* Naive Bayes (untuk data teks atau kategori)

1. **Pertemuan 18 : Linear Regression Single Variate dan Stochastic Gradient Descent**
2. **Deskripsi**

Pertemuan ini membahas lebih dalam tentang linear regression dengan satu variabel (single variate). Dijelaskan bahwa persamaan garis lurus  dapat direpresentasikan dalam bentuk , di mana  setara dengan gradien  dan  setara dengan konstanta . Dibahas pula bagaimana hubungan linear yang berbanding lurus antara variabel independen dan dependen, serta kasus-kasus di mana data tidak sepenuhnya linear sehingga menghasilkan error. Dua pendekatan untuk mencari nilai  dan  (atau  dan ) diperkenalkan: metode analitik menggunakan rumus statistik dan metode numerik dengan Stochastic Gradient Descent (SGD). Metode analitik dijelaskan melalui contoh perhitungan menggunakan rumus-rumus statistik, sementara SGD dijelaskan sebagai proses iteratif yang mencoba berbagai nilai parameter untuk meminimalkan error. Di akhir, diperlihatkan bagaimana peningkatan jumlah epoch dalam SGD dapat meningkatkan akurasi model, meskipun dengan trade-off waktu komputasi.

1. **Mindmap & Taksonomi**



1. **Penjelasan Detail**
2. Konsep Dasar Linear Regression Single Variate:

* Linear regression bertujuan untuk menemukan garis lurus yang paling sesuai dengan data.
* Persamaan garis: , dengan  adalah gradien (kemiringan) dan  adalah intercept.
* Dalam konteks machine learning, persamaan ini sering ditulis sebagai , di mana  adalah weight (setara dengan ) dan  adalah bias (setara dengan ).
* Gradien  menentukan sudut garis: jika , garis membentuk sudut 45°; jika , lebih curam; jika , lebih landai.
* Linearitas berarti perubahan pada  diikuti oleh perubahan proporsional pada . Jika tidak proporsional, maka terjadi error.

1. Metode Analitik untuk Mencari  dan :

* Menggunakan rumus statistik berdasarkan data.
* Rumus untuk :
* Rumus untuk :
* Contoh perhitungan diberikan dengan data luas rumah dan harga, menghasilkan  dan .
* Model akhir: .

1. Metode Numerik: Stochastic Gradient Descent (SGD):

* Pendekatan iteratif untuk mencari parameter yang meminimalkan error.
* Algoritma:
  1. Inisialisasi  dan  secara acak (misal: 0).
  2. Tentukan learning rate () dan jumlah epoch (iterasi maksimal).
  3. Untuk setiap epoch, lakukan:
* Untuk setiap data , hitung prediksi .
* Hitung error: .
* Hitung gradien untuk  dan :
* Perbarui parameter:
  1. Ulangi hingga konvergen (error sangat kecil) atau mencapai epoch maksimal.
* Kelebihan: dapat menangani data dalam skala besar dan kompleks.
* Kekurangan: membutuhkan banyak iterasi dan tuning hyperparameter (seperti learning rate).

1. Perbandingan Metode Analitik dan SGD:

* Metode analitik memberikan solusi eksak dalam satu langkah, tetapi tidak skalabel untuk data yang sangat besar atau model yang kompleks.
* SGD adalah pendekatan numerik yang lebih fleksibel dan dapat digunakan untuk model non-linear, tetapi membutuhkan waktu komputasi yang lebih lama.
* Dalam contoh, dengan meningkatkan jumlah epoch, SGD dapat menghasilkan parameter yang mendekati solusi analitik.

1. **Sampel Code (Implementasi Linear Regression dengan SGD)**

import numpy as np

import matplotlib.pyplot as plt

# Data contoh: luas rumah (x) dan harga (y)

x = np.array([60, 75, 85, 100, 120, 130, 140, 150, 160, 180])

y = np.array([115, 130, 150, 160, 180, 190, 200, 210, 220, 240])

# Inisialisasi parameter

m = 0.0

c = 0.0

# Hyperparameter

learning\_rate = 0.0001

epochs = 10000

n = len(x)

# Stochastic Gradient Descent

for epoch in range(epochs):

for i in range(n):

# Prediksi

y\_pred = m \* x[i] + c

# Error

error = y\_pred - y[i]

# Gradien

grad\_m = error \* x[i]

grad\_c = error

# Update parameter

m -= learning\_rate \* grad\_m

c -= learning\_rate \* grad\_c

# Hitung total error setiap 1000 epoch

if (epoch+1) % 1000 == 0:

total\_error = np.sum((y - (m\*x + c))\*\*2) / n

print(f'Epoch {epoch+1}: m = {m:.4f}, c = {c:.4f}, Error = {total\_error:.4f}')

print(f'\nHasil akhir: m = {m:.4f}, c = {c:.4f}')

# Prediksi untuk luas 100

x\_new = 100

y\_new = m \* x\_new + c

print(f'Prediksi harga untuk luas {x\_new}: {y\_new:.2f}')

# Visualisasi

plt.scatter(x, y, color='blue', label='Data Asli')

plt.plot(x, m\*x + c, color='red', label='Regresi Linear')

plt.xlabel('Luas Rumah (m²)')

plt.ylabel('Harga (juta)')

plt.legend()

plt.show()

**Hasilnya :**

**Epoch 1000: m = 1.2345, c = -10.1234, Error = 25.6789**

**Epoch 2000: m = 1.3456, c = -15.6789, Error = 15.4321**

**...**

**Epoch 10000: m = 1.5520, c = -18.7600, Error = 5.1234**

**Hasil akhir: m = 1.5520, c = -18.7600**

**Prediksi harga untuk luas 100: 136.44**

**(Catatan: Output di atas hanya ilustrasi. Nilai sebenarnya akan tergantung pada implementasi.)**