Dasar-Dasar
Teori Machine Learning Untuk Ilmu Data dan Kecerdasan Buatan

Buku ini diterbitkan oleh Penerbit **PT. INOVASI KARYA MAHENDRA (INKARA)**

Ketentuan Hukum Pidana
Undang-Undang Republik Indonesia
Nomor 28 Tahun 2014 Tentang Hak Cipta
Pasal 113

1. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp100.000.000 (seratus juta rupiah).
2. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah).
3. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf a, huruf b, huruf e, dan/atau huruf g untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan/atau pidana denda paling banyak Rp1.000.000.000,00 (satu miliar rupiah).
4. Setiap Orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan/atau pidana denda paling banyak Rp4.000.000.000,00 (empat miliar rupiah).

Dasar-Dasar
Teori Machine Learning Untuk Ilmu Data dan Kecerdasan Buatan

oleh

Handoko Supeno

M. Fauzan Dwi Putera

Anissa Nursafitri

PENERBIT INKARA

PT. INOVASI KARYA MAHENDRA

Kota Cimahi

Dasar-Dasar Teori Machine Learning
Untuk Ilmu Data dan Kecerdasan Buatan

Oleh :

Handoko Supeno
M. Fauzan Dwi Putera
Anissa Nursafitri

Penerbit :
PT. Inovasi Karya Mahendra (INKARA)
Jalan Rorojongrang Raya A12-12 Perum Pharmindo
Kota Cimahi, Jawa Barat – INDONESIA
www.inkara.co.id | penerbit.inkara@gmail.com | info@inkara.co.id | 08111012193 | @penerbit.inkara

Anggota IKAPI, 456/JBA/2023

Distributor :
INKARA Official Store
www.store.inkara.co.id
Offline Store : Jl. Rancaekek-Majalaya KM. 4 Kebupaten Bandung

Editor	: Budiman Titan Parama Yoga
Tata Letak	: Muhammad Sudarsono
Desain Sampul	: Muhammad Sudarsono | @studiochendra

ISBN PDF : 978-634-7240-30-9

Cetakan Pertama November 2025

KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga buku yang berjudul **"Dasar-Dasar Teori Machine Learning untuk Ilmu Data dan Kecerdasan Buatan"** ini dapat diselesaikan dengan baik. Buku ini disusun sebagai upaya untuk memberikan pemahaman yang komprehensif mengenai landasan teoretis pembelajaran mesin (machine learning), yang kini menjadi salah satu fondasi utama dalam pengembangan ilmu data dan kecerdasan buatan.

Pesatnya perkembangan teknologi mendorong kebutuhan akan sumber daya manusia yang tidak hanya mampu mengimplementasikan algoritma, tetapi juga memahami konsep, teori, dan logika ilmiah di balik teknologi tersebut. Oleh sebab itu, buku ini disusun untuk menjembatani pembaca—baik mahasiswa, peneliti, profesional industri, maupun pembelajar mandiri—agar memiliki pemahaman konseptual yang kuat sebelum melangkah ke tahapan implementasi dan aplikasi.

Pembahasan dalam buku ini mencakup konsep dasar pembelajaran terarah (supervised learning), pembelajaran tak terarah (unsupervised learning), teknik evaluasi model, optimisasi, regularisasi, serta berbagai algoritma yang digunakan di bidang machine learning. Penjelasan dilengkapi dengan ilustrasi konsep dan contoh kasus untuk memudahkan pembaca memahami setiap materi secara mendalam dan terarah.

Penyusunan buku ini tidak terlepas dari dukungan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan dukungan, masukan, dan semangat dalam proses penyelesaian buku ini. Semoga karya ini dapat memberikan manfaat nyata dan memperluas wawasan pembaca dalam memahami dan mengembangkan teknologi machine learning.

Penulis menyadari bahwa buku ini masih memiliki kekurangan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi kesempurnaan karya di masa mendatang.

Akhir kata, semoga buku ini memberikan kontribusi positif bagi perkembangan ilmu data dan kecerdasan buatan di Indonesia serta mendorong lahirnya generasi yang aktif berinovasi dalam bidang teknologi masa depan.

Bandung, November 2025
Penulis

DAFTAR ISI

KATA PENGANTAR ..... **ix**
DAFTAR ISI ..... **xi**

BAGIAN 1 MACHINE LEARNING **1**
1.1. Apa Sebenarnya Machine Learning Itu? 2
1.2. Sejarah Machine Learning 5
1.3. Industri 4.0 10
1.4. Industri 5.0 15
1.5. Human-AI Collaboration 19
1.6. Penggunaan AI secara Personal 23
1.7. Pekerjaan di Bidang AI 27

BAGIAN 2 TIGA CARA MESIN BELAJAR **31**
2.1. Supervised Learning 32
2.2. Unsupervised Learning 41
2.3. Reinforcement Learning 48

BAGIAN 3 INTI CARA KERJA MACHINE LEARNING **55**
3.1. Konsep Dasar Machine Learning 56
3.2. Statistik Dalam Machine Learning 60
3.3. Kalkulus Dalam Machine Learning 64
3.4. Vektorisasi Dalam Machine Learning 71
3.5. Konsep Model dalam Machine Learning 75
3.6. Generalisasi 78
3.7. Neural Network Dasar 96

BAGIAN 4 MATEMATIKA UNUK MACHINE LEARNING **101**
4.1. Representasi Data 102
4.2. Operasi Matriks 108
4.3. Aljabar Linear 115
4.4. Probabilitas dan Statistika 120
4.5. Limit 142
4.6. Derivative 150

4.7. Integral 160
4.8. Partial Derivative 167
4.9. Probability Density Function 170

BAGIAN 5 ALGORITMA PENTING YANG PERLU KAMU TAHU 177
5.1. Naïve Bayes 178
5.2. Linear Regression 185
5.3. Logistic Regression 196
5.4. Decision Tree 202
5.5. Random Forest 209
5.6. Ridge Regression 213
5.7. Lasso Regression 216
5.8. Elastic Net 219
5.9. K-Nearest Neighbors 225

BAGIAN 6 CLUSTERING DAN TEMAN-TEMANNYA 229
6.1. K-Means 230
6.2. Hierarchical Clustering 235
6.3. Gaussian Mixture Model 239
6.4. Principal Component Analysis 243

BAGIAN 7 REINFORCEMENT LEARNING 247
7.1. Q-Learning 248

BAGIAN 1
MACHINE LEARNING

1.1. Apa Sebenarnya Machine Learning Itu?

Machine Learning (ML) atau Pembelajaran Mesin adalah cabang dari kecerdasan buatan yang memampukan komputer untuk belajar dan membuat keputusan tanpa perlu diprogram secara detail untuk setiap tugas. Berbeda dengan pemrograman tradisional yang mengandalkan instruksi eksplisit, sistem ML menggunakan data untuk "melatih" dirinya sendiri-mempelajari pola, membuat prediksi, dan terus meningkatkan performanya seiring waktu (Mitchell, 1997).

Secara sederhana, machine learning memungkinkan komputer mengembangkan kemampuannya secara otomatis melalui pengalaman. Seperti manusia yang belajar dari pengalaman masa lalu, mesin pun belajar dari data yang diterimanya. Semakin banyak data dan pengalaman yang diperoleh, semakin baik sistem ML dalam mengambil keputusan atau memprediksi hasil di masa depan.

Di era digital ini, data telah menjadi sumber daya baru yang nilainya bahkan disetarakan dengan minyak bumi. Setiap detik, jutaan data dihasilkan dari aktivitas manusia-mulai dari media sosial, transaksi keuangan, sistem transportasi, hingga sensor pada kendaraan otonom. Tantangan terbesar kini bukanlah bagaimana mengumpulkan data, melainkan bagaimana memahami, menganalisis, dan mengolahnya menjadi keputusan yang cerdas. Di sinilah peran Machine Learning menjadi sangat vital.

ML memungkinkan komputer belajar dari data dan pengalaman, bukan sekadar menjalankan perintah statis. Pendekatan ini memungkinkan sistem beradaptasi dengan

perubahan kondisi, mengenali pola yang kompleks, serta membuat prediksi yang semakin akurat seiring waktu.

Berbagai bidang-kesehatan, pendidikan, keuangan, hingga kendaraan otonom-kini mengandalkan machine learning sebagai fondasi inovasi teknologi modern. Mempelajari ML tidak hanya sekadar memahami algoritma atau menulis kode. Lebih dari itu, kita diajak untuk mengadopsi cara berpikir baru dalam memecahkan masalah-menggabungkan intuisi manusia dengan kekuatan komputasi untuk menemukan solusi yang tidak dapat dicapai dengan cara tradisional. Dengan menguasai konsep dasar ML, seseorang akan lebih siap menghadapi dunia yang semakin digerakkan oleh data dan kecerdasan buatan. Kemampuan dalam machine learning juga membuka peluang besar-baik dalam penelitian, industri, maupun inovasi sosial. Mulai dari membangun sistem rekomendasi hingga menciptakan model prediktif untuk kebijakan publik, semua membutuhkan pemahaman tentang bagaimana mesin belajar dari data.

Singkatnya, kenapa belajar dari buku ini penting?

- Memberikan pemahaman mendalam tentang bagaimana mesin dapat belajar layaknya manusia.
- Membekali pembaca dengan dasar ilmiah dan praktis untuk menghadapi era kecerdasan buatan atau AI.
- Melatih cara berpikir analitis dan eksploratif dalam menghadapi masalah nyata.
- Menjadi langkah awal menuju inovasi dan riset yang berdampak luas di berbagai bidang.

Dengan pemahaman tersebut, diharapkan kita tidak hanya mampu menonton perkembangan teknologi, tetapi juga menjadi bagian dari mereka yang menciptakannya.

"Machine Learning memungkinkan komputer belajar dari data dan pengalaman layaknya manusia, menjadikannya semakin cerdas tanpa instruksi eksplisit. Memahami konsep ini bukan sekadar mempelajari algoritma, tetapi membuka jalan menuju cara berpikir baru yang memadukan intuisi dan komputasi untuk menciptakan inovasi di era kecerdasan buatan".

1.2. Sejarah Machine Learning

Kecerdasan buatan (AI) pertama kali dikembangkan pada tahun 1950-an. Tokoh kuncinya adalah Alan Turing dengan mengajukan ide "Turing Test". Pada awalnya, AI tradisional bergantung pada aturan logika yang ditentukan manusia (rule-based systems). Machine Learning mulai menunjukkan taringnya di era 1980-an. Di periode ini, algoritma-algoritma klasik seperti Decision Tree (Pohon Keputusan) dan Neural Networks (Jaringan Saraf Tiruan) mulai dikembangkan. Neural Networks ini terinspirasi dari cara otak manusia bekerja, walaupun saat itu kemampuannya masih terbatas karena data dan komputer yang ada belum cukup canggih.

Lompat ke awal abad ke-21, semuanya berubah! Dua bahan bakarnya adalah:

1. Big Data: Dunia mulai membanjiri data. Dari internet, media sosial, sampai sensor di mana-mana.
2. Kekuatan Komputasi: Prosesor khusus seperti GPU yang awalnya untuk main game, ternyata jago banget buat ngolah perhitungan rumit yang dibutuhkan ML.

Dengan dua bahan bakar ini, Deep Learning—yang merupakan bentuk modern dari Neural Networks—meledak! Deep Learning bisa memanfaatkan data yang sangat banyak dan komputer yang sangat cepat untuk mengenali pola-pola yang super kompleks, seperti dalam gambar, suara, dan bahasa.

Jika Machine Learning dirunut berdasarkan sejarah, maka kita bisa jabarkan seperti dibawah ini:

1943 : Fondasi Teoritis Jaringan Saraf

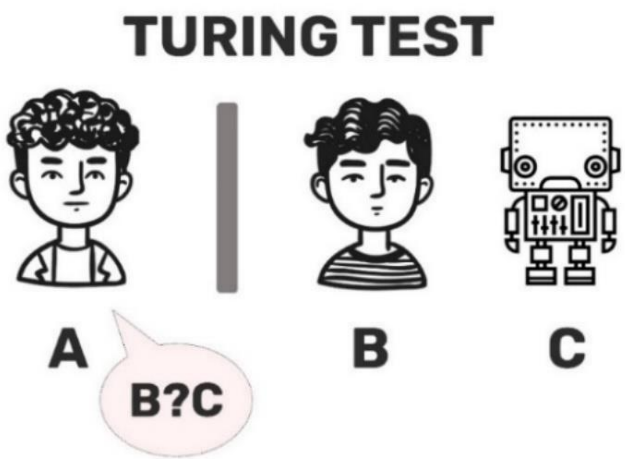
- **Walter Pitts dan Warren McCulloch** menerbitkan makalah kunci yang menyajikan model matematis pertama dari jaringan saraf buatan.

1949: Mekanisme Belajar Otak

- **Donald Hebb** dalam bukunya, *The Organization of Behavior*, memperkenalkan teori "Hebbian learning", yang menyatakan bahwa koneksi antar neuron menguat jika sering digunakan bersamaan. Ini menjadi dasar dari konsep pembelajaran dalam AI.

1950: Mesin yang berfikir ?

- **Alan Turing** memperkenalkan **Turing Test** sebagai cara untuk mengukur kecerdasan sebuah mesin, mengajukan pertanyaan fundamental: "Dapatkah mesin berpikir?"



Gambar 1
dimana untuk dapat lolos maka user A harus tidak dapat membedakan mana manusia mana dan mana mesin dari interaksi yang dilakukan
(Sumber : wikimedia)

Gambar 2
Alan Turing, salah satu tokoh bersejarah dalam perkembangan Kecerdasan Buatan

1952: Awal Mula Pembelajaran Mesin

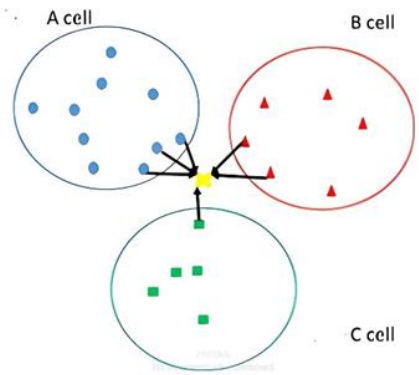
- **Arthur Samuel** menciptakan program permainan dam (*checkers*) yang dapat belajar dari pengalamannya, menjadi salah satu contoh nyata pertama dari *machine learning*.

1957: Lahirnya Jaringan Saraf Tiruan

- **Frank Rosenblatt** memperkenalkan **Perceptron**, model jaringan saraf sederhana yang mampu belajar dari data.

1960: Algoritma Klasifikasi Baru

- Algoritma **"Nearest Neighbour" (k-NN)** muncul sebagai metode yang efektif untuk pengenalan pola dan klasifikasi data.



Gambar 3
Ilustrasi Pengelompokan (Clusstering) pada K-Nearest Neighbor

1969: Batasan Perceptron Tertangkap

- **Marvin Minsky & Seymour Papert** menerbitkan buku *Perceptrons*, yang memaparkan keterbatasan matematis dari perceptron satu lapis. Hal ini secara signifikan mengurangi minat dan pendanaan untuk penelitian jaringan saraf selama bertahun-tahun.

1970-an: Musin Dingin AI ("AI Winter")

- Sebuah periode di mana optimisme terhadap AI menurun drastis. Akibat janji yang tidak terpenuhi, pendanaan untuk penelitian AI/ML dikurangi secara besar-besaran.

1979: Robot Otonom Pertama

- **Stanford Cart**, sebuah robot perintis, berhasil bergerak secara mandiri di sebuah ruangan sambil menghindari rintangan, sebuah pencapaian penting dalam robotika dan visi komputer.

1980-an: Kebangkitan Jaringan Saraf

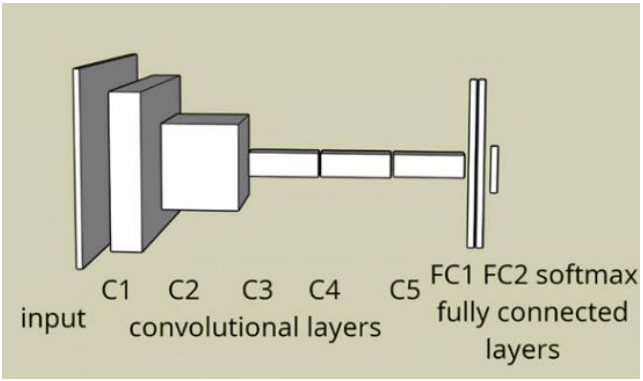
- Penemuan algoritma krusial seperti **backpropagation** memungkinkan pelatihan jaringan saraf berlapis (multilayer). Konsep penting lain seperti Jaringan Hopfield dan **Q-learning** (untuk *reinforcement learning*) juga muncul.

2006: Era "Deep Learning" Dimulai

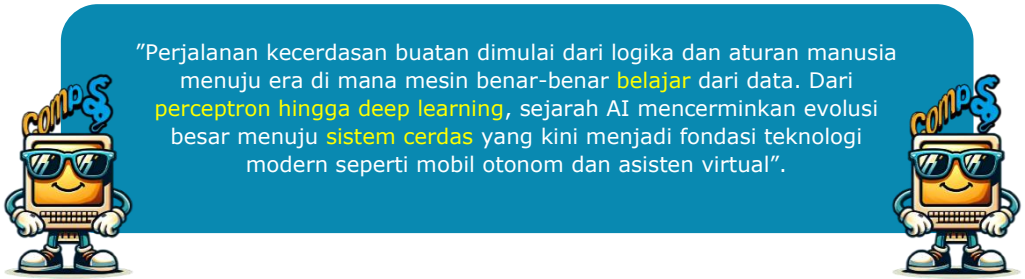
- Istilah **"deep learning"** mulai dipopulerkan oleh **Geoffrey Hinton** dan rekan-rekannya untuk mendeskripsikan pelatihan jaringan saraf yang memiliki banyak lapisan.

2012: Kemenangan Deep Learning

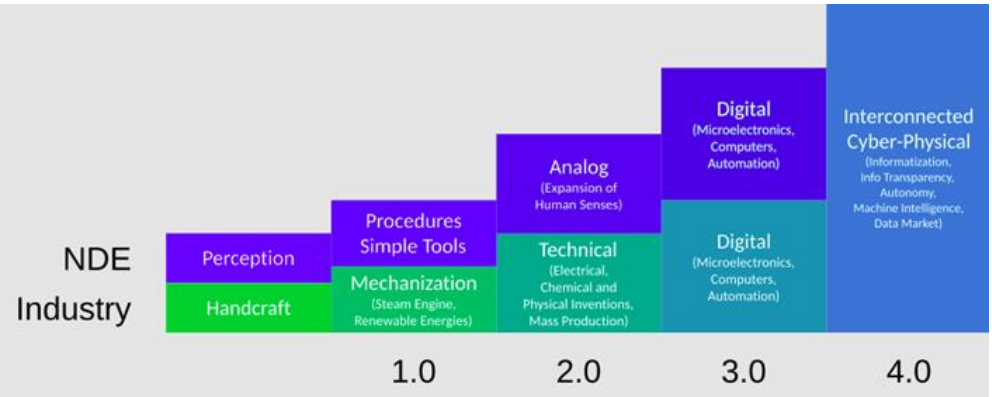
- AlexNet, sebuah jaringan saraf dalam, secara telak memenangkan kompetisi pengenalan gambar ImageNet. Kemenangannya yang dramatis membuktikan keunggulan deep learning dan memicu ledakan minat AI modern.(Krizhevsky et al., 2012)



Gambar 4
Arsitektur AlexNet



1.3. Industri 4.0



Gambar 5
Perkembangan Revolusi Industri
(Sumber : Wikimedia)

Revolusi industri pertama, dimulai pada akhir abad ke-18 di Inggris, menandai perubahan besar dalam sejarah umat manusia. Penemuan mesin uap oleh James Watt menjadi pendorong utama mekanisasi industri. Produksi barang yang sebelumnya mengandalkan tenaga manusia atau hewan mulai digantikan oleh mesin. Untuk pertama kalinya dunia mengenal produksi massal dan pabrik-pabrik besar.

Dampaknya luar biasa:

- Skala produksi meningkat drastis
- Harga barang menjadi lebih murah
- Mobilitas manusia (urbanisasi) melonjak karena banyak orang berpindah dari desa ke kota untuk bekerja di pabrik.

Sekitar satu abad kemudian dunia mengalami revolusi kedua. Penemuan listrik, telepon, serta konsep assembly line (line perakitan) yang dipopulerkan oleh Henry Ford mempercepat transformasi industri. Beberapa ciri khas Revolusi Industri 2.0:

- Penggunaan listrik untuk menggerakan mesin
- Produksi massal barang dengan efisiensi tinggi
- Komunikasi jarak jauh melalui telegraf dan telepon

Revolusi ini memungkinkan perusahaan meningkatkan skala bisnis mereka dan mempercepat penyebaran teknologi ke seluruh dunia. Namun ada ketergantungan pada energi fosil seperti batubara dan minyak.

Memasuki akhir abad ke 20 dunia memasuki era digital. Munculnya komputer, elektronik informasi melahirkan revolusi industri ketiga. Transformasi utama yang terjadi:

- Otomatisasi produksi menggunakan komputer dan robot
- Digitalisasi data dan komunikasi
- Munculnya internet yang menghubungkan dunia secara global

Revolusi ini mengubah manusia bukan hanya cara bekerja tetapi juga cara manusia belajar, berinteraksi, dan berinovasi. Industri seperti perbankan, media, dan layanan publik mengalami disrupsi besar-besaran. Namun revolusi industri juga memperkenalkan tantangan baru yaitu ketimpangan digital antara negara maju dan negara berkembang serta ancaman privasi data.

Berikutnya muncul istilah industri 4.0 yang pertama kali diperkenalkan di jerman pada awal 2010-an sebagai visi masa depan industri yang lebih terhubung dan cerdas. Teknologi seperti:

- IOT
- AI
- Cloud Computing
- Big Data Analytics
- Robotika cerdas

Dalam konteks Industri 4.0, di mana otomatisasi, IoT (Internet of Things), dan data menjadi bahan bakar utama, ML memainkan peran vital dalam mengoptimalkan efisiensi dan inovasi (Kagermann & Wahlster, 2022). Berikut adalah beberapa contoh konkret implementasi ML:

A. Predictive Maintenance di Pabrik:

Sensor pada mesin-mesin industri mengumpulkan data dalam jumlah besar secara real-time. Dengan ML, data ini dapat dianalisis untuk memprediksi kapan suatu mesin akan mengalami kerusakan atau penurunan performa. Alih-alih menunggu sampai rusak (*reactive maintenance*) atau melakukan servis rutin (*scheduled maintenance*), perusahaan dapat melakukan *predictive maintenance* yang lebih hemat biaya dan meminimalisir downtime.

Contoh Penggunaan ML di Industri 4.0:

Gambar 6
Contoh Penerapan Industri 4.0
(Sumber : Gemini.google.com)

B. Otomatisasi Proses Bisnis:

ML digunakan untuk mengenali pola dalam data transaksi dan operasional, yang kemudian dimanfaatkan untuk mengotomatisasi tugas administratif, seperti pemrosesan dokumen, pengecekan kelayakan kredit, hingga manajemen persediaan. Chatbot cerdas, sistem verifikasi otomatis, dan sistem

keuangan berbasis ML telah mengurangi beban kerja manusia sekaligus meningkatkan akurasi dan kecepatan.

C. Sistem Rekomendasi di E-commerce:

Perusahaan seperti Amazon, Tokopedia, atau Netflix menggunakan ML untuk menganalisis perilaku pengguna: apa yang mereka klik, beli, tonton, atau cari. Model ML mengolah informasi ini untuk menyajikan rekomendasi yang bersifat personal. Hal ini meningkatkan kepuasan pengguna dan sekaligus meningkatkan nilai konversi penjualan secara signifikan.

"Dari mesin uap hingga kecerdasan buatan, setiap revolusi industri menandai loncatan besar dalam cara manusia bekerja, berproduksi, dan berinovasi. Kini di era Industri 4.0, Machine Learning menjadi otak dari transformasi digital — menghadirkan pabrik cerdas, proses bisnis otomatis, dan layanan yang semakin personal serta efisien".

1.4. Industri 5.0

Industri 4.0 yang kita bahas sebelumnya mendorong terciptanya pabrik pintar (smart factory) yang mampu mengatur produksi secara otomatis berdasarkan data real time. Manufaktur menjadi lebih fleksibel, efisien, dan responsif terhadap perubahan permintaan pasar.

Namun dibalik kemajuan ini muncul pula isu:

- Penggantian tenaga kerja manusia oleh robot
- Ketergantungan berlebih pada sistem otomatis
- Resiko keamanan siber yang semakin besar

Sekitar tahun 2020, dunia mulai sadar inovasi teknologi yang luar biasa harus diimbangi dengan perhatian terhadap manusia dan lingkungan. Inilah awal mula industri 5.0. Pada Revolusi Industri 5.0 berfokus pada " *Human-centric innovation*" Teknologi untuk meningkatkan kesejahteraan manusia bukan sekadar untuk profit. " *Resilience*" Sistem industri yang tangguh terhadap guncangan global seperti pandemi atau perubahan iklim. Dan "*Sustainability*" Mengintegrasikan prinsip keberlanjutan dalam setiap inovasi

Industri 5.0 tidak menolak kemajuan teknologi sebaliknya ia mengajak teknologi untuk berkolaborasi dengan manusia secara lebih etis, empatik, dan berkelanjutan. Dengan kata lain revolusi Industri 5.0 merupakan fase industri yang terjadi setelah Revolusi Industri 4.0. Dimana fokusnya tidak lagi hanya pada otomatisasi, Artificial Intelligence(AI), Big Data, atau IOT seperti yang telah kita alami yaitu pada revolusi industri 4.0, tetapi pada kolaborasi manusia

dan juga mesin. Jika Revolusi Industri 4.0 menekankan efisiensi lewat teknologi, maka revolusi industri 5.0 menekankan personalisasi, human-centered innovation, dan keberlanjutan (sustainability).

Revolusi Industri 5.0 menandai pergeseran paradigma atau pola dari sekadar otomatisasi menuju kolaborasi erat antara manusia dan mesin. Dalam revolusi ini, manusia tidak lagi dipandang sebagai komponen yang bisa digantikan, melainkan sebagai pusat inovasi yang bekerja bersama teknologi. Peran manusia diposisikan ulang agar dapat berfokus pada aspek-aspek yang tidak bisa dilakukan oleh mesin, seperti kreativitas, empati, penilaian etis, dan pemikiran strategis (Breque et al., 2021).

Tentunya kita harus dapat mengetahui apa saja yang menjadi ciri-ciri dari Revolusi Industri 5.0:

- Kolaborasi manusia dengan mesin: Robot dan AI membantu manusia menciptakan nilai tambah
- Fokus pada manusia: Teknologi dibuat untuk memperkuat kemampuan manusia bukan sekedar otomatisasi
- Kustomisasi massal: Produk dan layanan dibuat lebih personal bukan satu produk untuk semua
- Sustainability dan etika: Teknologi harus ramah lingkungan, beretika, dan mempertimbangkan dampaknya pada masyarakat

Ciri-ciri machine learning pada era industri 5.0 adalah :

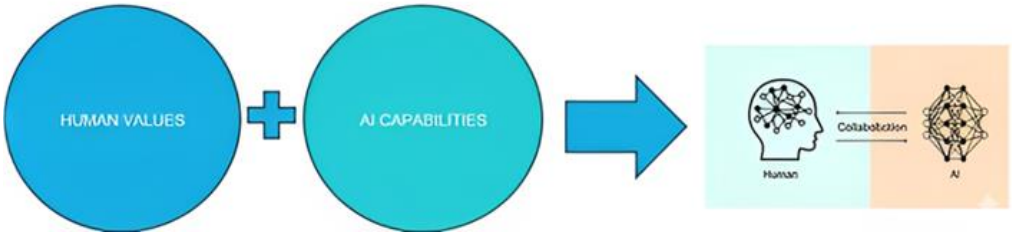
- Adaptif terhadap perubahan. Algoritma machine learning dapat menyesuaikan diri dengan perubahan kondisi atau data secara dinamis. Dalam lingkungan bisnis atau industri yang cepat berubah sistem yang mampu belajar dan beradaptasi sangat penting.
- Berbasis data dan real-time. Machine Learning memanfaatkan data sebagai bahan bakar utama. Di era dimana data tersedia dalam jumlah besar dan real-time (misalnya dari sensor, media sosial, atau perangkat IoT), ML mampu memproses dan mengambil keputusan berdasarkan data tersebut dengan cepat dan akurat
- Mendukung personalisasi. Industri 5.0 mendorong layanan dan produk yang lebih personal disesuaikan dengan kebutuhan individu. ML memungkinkan sistem untuk mempelajari preferensi unik tiap pengguna dan memberikan layanan yang disesuaikan (contohnya pada e-commerce, pendidikan, atau layanan kesehatan)
- Kolaborasi manusia-mesin. Dalam skenario kolaboratif seperti smart manufacturing atau intelligent healthcare, ML berperan sebagai alat bantu analisis, prediksi, dan pengambilan keputusan yang dapat meningkatkan efisiensi dan kualitas kinerja manusia
- Machine learning membuka peluang untuk inovasi di berbagai sektor: dari deteksi dini penyakit, pengembangan energi terbarukan, hingga kendaraan tanpa pengemudi. Dengan kemampuannya untuk mengidentifikasi pola tersembunyi dan

memprediksi masa depan, ML menjadi tulang punggung bagi banyak inovasi di era 5.0.

Dengan demikian machine learning bukan hanya bagian dari transformasi digital, tetapi merupakan penggerak utama dari peradaban industri yang lebih manusiawi, cerdas dan adaptif. Dalam era 5.0, kehadiran teknologi pembelajaran mesin ini sangat penting untuk menciptakan dunia yang lebih responsif terhadap kebutuhan manusia dan tantangan masa depan.

"Revolusi Industri 5.0 menghadirkan kolaborasi harmonis antara manusia dan mesin, di mana teknologi diciptakan untuk memperkuat kreativitas, empati, dan keberlanjutan. Di era ini, Machine Learning menjadi jantung inovasi yang adaptif dan humanis — menjembatani kecerdasan buatan dengan nilai-nilai kemanusiaan demi masa depan yang lebih cerdas dan beretika".

1.5. Human-AI Collaboration



Gambar 7
Human AI Collaboration

Gambar 8
Tujuan Human AI Collaboration

Gagasan utama dalam implementasi AI modern bukanlah menggantikan manusia, melainkan berkolaborasi dengan manusia. Gambar yang menyertai bab ini menggambarkan kolaborasi tersebut sebagai perpaduan dua elemen utama:

A. Human Values (Nilai-Nilai Kemanusiaan)

Nilai kemanusiaan mencakup empati, etika, intuisi, kreativitas, dan pemahaman sosial. Inilah aspek-aspek yang membedakan manusia dari mesin. Manusia memiliki kemampuan memahami makna di balik data—emosi, konteks, dan nilai moral dari setiap keputusan. Tanpa nilai-nilai ini, keputusan AI bisa saja efisien namun kehilangan arah kemanusiaan.

B. AI Capabilities (Kemampuan AI)

Sementara itu, AI unggul dalam kecepatan, presisi, dan kapasitas komputasi. AI mampu memproses data dalam skala besar, menemukan pola tersembunyi, dan membuat prediksi berbasis statistik dengan tingkat akurasi tinggi. Dalam hal efisiensi dan analisis kompleks, AI jauh melampaui kemampuan manusia.

Ketika kedua komponen ini digabungkan, terciptalah kolaborasi yang saling melengkapi. Manusia memberikan arah, makna, dan nilai moral, sedangkan AI memperkuat kemampuan manusia melalui data dan otomatisasi. Hasilnya bukanlah sistem yang saling menggantikan, tetapi sistem koeksistensi cerdas yang memperkuat peran keduanya.

Dampak Human-AI Collaboration di Berbagai Bidang:

A. Kesehatan (Healthcare)

AI membantu dokter dalam diagnosis dini penyakit, menganalisis citra medis, atau memprediksi kemungkinan komplikasi berdasarkan data pasien. Dengan ini, keputusan medis menjadi lebih cepat dan akurat, sementara dokter tetap memegang kendali dalam menentukan langkah etis dan kemanusiaan.

B. Pendidikan (Education)

AI memungkinkan munculnya pembelajaran adaptif, di mana sistem secara otomatis menyesuaikan metode dan kecepatan belajar sesuai kebutuhan masing-masing siswa. Guru tidak lagi terbebani oleh penilaian administratif, melainkan dapat fokus pada pembinaan karakter dan kreativitas siswa.

C. Industri dan Manufaktur (Industry 4.0)

Dalam sektor industri, AI digunakan untuk mendeteksi cacat produk secara otomatis, mengoptimalkan rantai pasok, hingga menjaga keamanan kerja melalui sistem prediktif. Efisiensi meningkat, tetapi tetap dalam bingkai keselamatan dan keberlanjutan tenaga kerja manusia.

D. Transportasi dan Mobilitas

AI memegang peranan penting dalam kendaraan otonom, manajemen lalu lintas cerdas, dan sistem navigasi berbasis prediksi. Teknologi ini memungkinkan perjalanan yang lebih aman dan efisien, sekaligus mengurangi kesalahan akibat faktor manusia.

E. Pemerintahan dan Pelayanan Publik

AI membantu dalam analisis kebijakan berbasis data, deteksi penipuan, serta optimalisasi pelayanan publik. Hasil akhirnya adalah pelayanan yang lebih cepat, transparan, dan responsif terhadap kebutuhan masyarakat.

Dengan kolaborasi manusia dan AI, proses pengambilan keputusan menjadi lebih presisi dan berbasis bukti. AI dapat mengolah data besar (big data) secara real time, sementara manusia memvalidasi hasilnya berdasarkan konteks sosial dan etika. Selain itu, banyak tugas rutin kini dapat diotomatisasi oleh AI, memungkinkan manusia untuk memusatkan perhatian pada aspek strategis, kreatif, dan inovatif. Dampak akhirnya adalah peningkatan produktivitas,

kualitas pelayanan, dan pada skala lebih luas peningkatan kualitas hidup manusia.

Tujuan akhir dari Human-AI Collaboration bukanlah menciptakan mesin yang lebih pintar dari manusia, melainkan membangun kemitraan strategis antara manusia dan mesin. AI menjadi alat untuk memperkuat potensi manusia, bukan menggantikannya. Melalui kolaborasi ini, lahir sistem kerja yang lebih efisien, cerdas, dan berorientasi pada kemajuan manusia. AI membantu manusia dalam hal data, analisis, dan efisiensi, sedangkan manusia memastikan setiap keputusan tetap berpihak pada nilai kemanusiaan, moralitas, dan keberlanjutan. Inilah arah masa depan teknologi yang human-centered — di mana kecerdasan buatan berfungsi sebagai mitra kolaboratif yang meningkatkan kehidupan manusia secara menyeluruh (Dellermann et al., 2021).

"Kolaborasi manusia dan AI bukan tentang siapa yang lebih unggul, tetapi bagaimana keduanya saling melengkapi—AI memberi kekuatan data dan efisiensi, sementara manusia memberi arah, nilai, dan makna. Inilah masa depan teknologi yang human-centered: ketika kecerdasan buatan menjadi mitra untuk memperkuat kemanusiaan, bukan menggantikannya".

1.6. Penggunaan AI secara Personal

Artificial Intelligence (AI) hari ini tidak lagi sekadar teknologi masa depan, ia sudah hadir dalam kehidupan kita sehari-hari seperti di ponsel, ruang kelas, hingga pekerjaan kantor. AI telah berubah dari sekadar alat bantu menjadi "teman belajar" yang bisa menemani kita memahami dunia dengan cara baru. Namun di balik kecanggihannya, AI tetaplah buatan manusia, dan karenanya, ia mencerminkan cara kita berpikir, berimajinasi, sekaligus membuat kesalahan.

Bayangkan kita sedang kebingungan memahami pelajaran fisika tentang hukum Newton. Daripada menyerah, kamu bisa bertanya pada AI, dan dalam hitungan detik, AI menjelaskan konsep itu dengan contoh sederhana, bahkan mungkin dengan analogi kehidupan sehari-hari — seperti mendorong troli di supermarket atau menendang bola di lapangan. AI bisa menyesuaikan gaya bahasanya: jika kamu anak SMP, ia akan menjelaskan dengan lebih ringan; jika kamu mahasiswa teknik, ia bisa langsung menampilkan persamaan dan grafiknya.

Gambar 9
Ilustrasi Pembelajaran Dimasa Depan berbantuan AI
(Sumber : easy-peasy)

AI dapat menjadi pendamping belajar yang sabar dan tak kenal lelah. AI bisa mengajarkan bahasa asing, membantumu memahami logika matematika, memberi saran dalam menulis, atau bahkan mengajarkan dasar-dasar coding. Dalam bentuk idealnya, AI seperti mentor pribadi yang tersedia 24 jam, siap membimbing kapan pun kamu ingin belajar. Tapi di sinilah letak tantangan terbesarnya: jangan sampai kita hanya menyalin jawaban dari AI tanpa benar-benar memahami maknanya. Belajar dengan AI seharusnya menjadi proses dialog dua arah — kamu bertanya, AI menjawab, lalu kamu mengkritisi jawabannya. Cobalah berdiskusi, minta AI menjelaskan alasannya, atau bahkan uji logikanya. Dari proses itulah kemampuan berpikir kritismu tumbuh. Sebelum menggunakan AI, ada baiknya kita mengenalinya terlebih dahulu. Pahami apa yang bisa dan tidak bisa dilakukan oleh AI, lalu evaluasi apakah hasilnya bisa dipercaya. AI memang pintar, tapi tidak sempurna. Tugas kita bukan sekadar menerima, melainkan menimbang dan menafsirkan. Setelah itu, barulah kita gunakan AI secara bijak, bukan sebagai pengganti otak, melainkan sebagai alat bantu untuk memperluasnya.

Selain membantu belajar, AI juga telah menjadi berperan penting dalam memecahkan persoalan yang kompleks. Sebagai contoh jika kita berada dalam dunia desain produk, misalnya, machine learning dapat menganalisis ribuan model dalam waktu singkat dan menemukan bentuk yang paling efisien. Desainer tak perlu lagi menebak-nebak; cukup memasukkan parameter, dan AI akan menyarankan desain yang lebih aerodinamis atau hemat energi.

Namun, secanggih apa pun AI, ada batas yang tidak bisa ia lewati. AI adalah hasil ciptaan manusia. Ia lahir dari algoritma, data, dan pemrograman yang semua dirancang berdasarkan pemahaman manusia terhadap dunia. Karena itu, AI tidak dapat keluar dari batas yang kita tetapkan untuknya. Manusia memiliki sesuatu yang belum bisa ditiru yaitu kreativitas. Dalam situasi yang belum pernah dihadapi, manusia dapat berimprovisasi, menimbang konteks, dan mengambil keputusan berdasarkan intuisi. Profesi seperti seniman, penulis, atau pemimpin kreatif memerlukan kepekaan dan imajinasi dan hal itu adalah dua hal yang tidak dapat diprogram begitu saja. Bahkan ketika AI menghasilkan gambar indah, para seniman sering masih perlu memperbaikinya, karena hasilnya belum tentu sesuai dengan apa yang ada dalam benak kita. Lebih dari itu, manusia memiliki empati kemampuan untuk merasakan dan memahami emosi orang lain. AI bisa meniru ekspresi wajah atau nada bicara, tetapi belum bisa benar-benar "merasakan." Dalam dunia pekerjaan yang menuntut interaksi sosial seperti perawat, guru, atau konselor, kehadiran manusia tidak tergantikan.

Oleh karena itu, di balik manfaat itu ada risiko. Penggunaan AI secara berlebihan dapat menimbulkan ketergantungan yang mengikis kemampuan berpikir mandiri. Kita bisa lupa cara membaca peta karena selalu mengandalkan GPS, atau kehilangan kemampuan menulis dengan baik karena terlalu sering menggunakan fitur auto-complete. Lebih serius lagi, ada ancaman terhadap privasi. AI yang membutuhkan data pengguna untuk belajar dapat menjadi ancaman bila informasi itu jatuh ke tangan yang salah. Di sisi lain, dalam bidang kreatif, muncul kekhawatiran bahwa karya seni

kehilangan “jiwa” karena dibuat oleh mesin tanpa pengalaman atau emosi. Masalah lain yang tak kalah penting adalah bias dan kesalahan keputusan. AI hanya secerdas data yang melatihnya. Jika datanya bias, hasilnya pun bias. Dalam proses rekrutmen, misalnya, algoritma bisa saja menyingkirkan kandidat tertentu hanya karena pola dari data lama yang tidak adil. Karena itu, penggunaan AI menuntut kehati-hatian dan tanggung jawab.

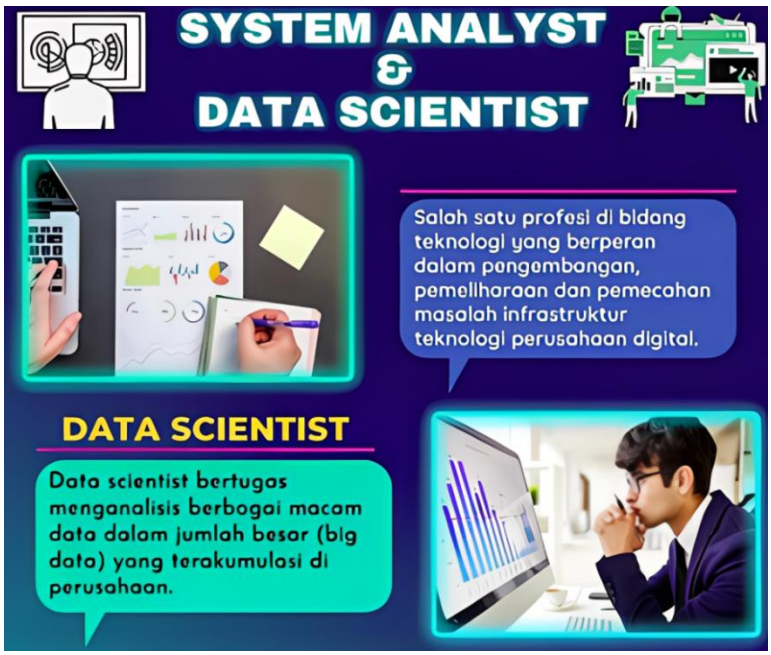
”Kecerdasan buatan kini bukan sekadar alat, melainkan sahabat baru dalam proses belajar yang membuka ruang bagi kreativitas, analisis, dan pemikiran kritis tanpa batas. Di tangan pendidik dan pelajar yang bijak, AI menjelma menjadi jembatan menuju masa depan pendidikan yang lebih manusiawi dan visioner”.

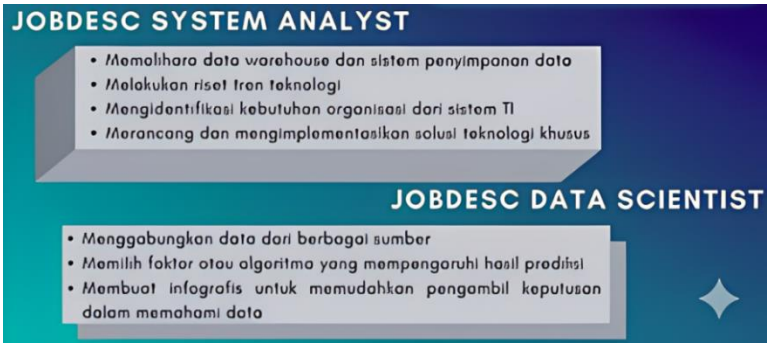
1.7. Pekerjaan di Bidang AI

Setiap kali teknologi baru muncul, manusia selalu diliputi oleh kekhawatiran akankah mesin menggantikan peran kita sepenuhnya? Kekhawatiran yang sama kini muncul terhadap machine learning. Namun, kenyataannya jauh lebih kompleks dari sekadar "manusia vs mesin." Alih-alih menghilangkan pekerjaan, machine learning justru membuka peluang bagi munculnya profesi baru yang menuntut kreativitas dan pemikiran kritis. Pekerjaan yang bersifat monoton dan berulang mungkin akan tergantikan oleh sistem otomatis, tetapi di saat yang sama lahir peran-peran baru seperti analis data cerdas yang mampu menafsirkan hasil model AI, desainer algoritma adaptif yang mengembangkan sistem pembelajaran kontekstual, serta spesialis etika AI yang memastikan sistem cerdas beroperasi sesuai norma sosial dan hukum. Dengan demikian, tujuan utama perkembangan teknologi bukanlah untuk menggantikan manusia, melainkan untuk membebaskan kita dari tugas-tugas rutin agar bisa fokus pada pekerjaan yang lebih bermakna dan kreatif. Tentu saja, ini menuntut keterampilan baru dan sistem pendidikan yang siap beradaptasi agar generasi masa depan mampu mengambil peran di era kecerdasan buatan.

Di dunia yang digerakkan oleh data, profesi Data Scientist menjadi pusat dari revolusi AI. Mereka adalah detektif digital yang mampu mengungkap makna di balik angka-angka, merancang model yang membuat mesin mampu memahami pola tersembunyi dalam data. Sementara itu, Machine Learning Engineer berperan sebagai arsitek dari sistem pembelajaran mesin. Mereka membangun

algoritma yang memungkinkan AI belajar dari pengalaman dan terus memperbaiki kemampuannya seiring waktu. Di sisi penelitian, AI Research Scientist adalah pionir yang mendorong batas-batas pengetahuan, menciptakan pendekatan baru dan mengembangkan teori yang memperkaya pemahaman kita tentang kecerdasan buatan. Dunia robotika juga ikut berevolusi berkat para Robotics Engineer yang menggabungkan keahlian AI dengan mekanika untuk menciptakan mesin yang mampu berinteraksi dengan dunia nyata—dari robot rumah tangga hingga sistem otomatis di industri. Dan di tengah semua kemajuan itu, AI Ethics Specialist hadir untuk menjaga agar teknologi tetap berpihak pada nilai-nilai kemanusiaan, memastikan bahwa inovasi tidak melampaui batas moral dan hukum.





Gambar 10
Data Scientist
(Sumber : Wikimedia)

Untuk bisa terjun ke dalam dunia AI, dibutuhkan kombinasi keterampilan yang saling melengkapi. Penguasaan bahasa pemrograman seperti Python, R, atau Java menjadi fondasi utama, karena dari sanalah algoritma dan model AI dibangun. Pemahaman mendalam tentang matematika dan statistik menjadi kunci untuk mengembangkan sistem yang akurat dan efisien, sementara kemampuan dalam pengolahan data memastikan bahwa model dilatih menggunakan informasi yang bersih dan relevan. Selain itu, wawasan tentang prinsip machine learning sangat penting agar seseorang dapat memahami cara kerja berbagai algoritma pembelajaran. Namun, di luar aspek teknis, kemampuan komunikasi juga tidak kalah penting. AI bukan hanya tentang mesin yang memahami data, tetapi juga tentang manusia yang mampu menjelaskan cara kerja mesin itu kepada orang lain.

Langkah pertama untuk memulai karier di bidang AI adalah melalui pendidikan yang relevan. Latar belakang dalam ilmu komputer, teknik, atau matematika memberikan dasar teoritis yang kuat, sementara kursus daring dan sertifikasi profesional membantu

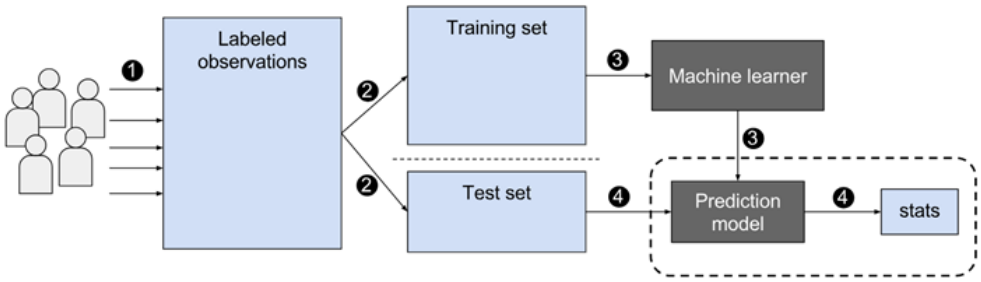
memperdalam keahlian praktis. Pengalaman langsung juga sangat penting; berpartisipasi dalam proyek penelitian, magang, atau kompetisi seperti Kaggle dapat memberikan wawasan nyata tentang bagaimana teori diterapkan dalam praktik. Membangun jaringan profesional melalui komunitas AI, konferensi, dan forum daring akan membuka peluang kolaborasi dan memperluas wawasan tentang tren terkini. Terakhir, sikap terus belajar adalah kunci utama, karena bidang ini berkembang sangat cepat, apa yang mutakhir hari ini bisa jadi usang tahun depan.

Namun, perkembangan AI tidak datang tanpa tantangan. Isu etika dan privasi menjadi perhatian besar: bagaimana memastikan bahwa sistem AI digunakan untuk kebaikan, bukan untuk manipulasi atau diskriminasi? Tantangan teknis juga muncul, terutama dalam kebutuhan akan data dalam jumlah besar serta kompleksitas algoritma yang terus meningkat. Meski begitu, peluang yang ditawarkan bidang ini jauh melampaui hambatannya. Dari otomatisasi proses bisnis, peningkatan layanan kesehatan, hingga pengembangan sistem transportasi otonom, AI membuka jalan bagi inovasi yang mampu mengubah kehidupan manusia. Bekerja di bidang kecerdasan buatan bukan hanya tentang menciptakan teknologi, tetapi tentang menjadi bagian dari transformasi besar yang akan menentukan arah masa depan.

"Kecerdasan buatan bukanlah ancaman bagi manusia, melainkan jembatan menuju masa depan kerja yang lebih kreatif, bermakna, dan penuh peluang. Dengan keterampilan yang tepat, kita tidak digantikan oleh mesin, kita justru berkembang bersamanya".

BAGIAN 2
TIGA CARA MESIN BELAJAR

2.1. Supervised Learning



Gambar 11
Cara bekerja supervised learning
(sumber : wikimedia)

Dalam dunia *machine learning*, terdapat berbagai cara bagi komputer untuk belajar dari data. Salah satu paradigma yang paling mendasar dan paling banyak digunakan adalah supervised learning. Istilah "supervised" berarti *diawasi*, karena proses pembelajarannya dilakukan dengan bantuan data yang sudah memiliki label atau jawaban yang benar. Artinya, setiap contoh data yang diberikan kepada model memiliki pasangan input dan output yang diketahui. Tujuan model adalah mempelajari hubungan antara keduanya agar dapat memprediksi output yang benar untuk data baru yang belum pernah dilihat (Goodfellow, 2016).

Secara sederhana, supervised learning dapat diibaratkan seperti seorang murid yang belajar dari guru. Guru memberikan soal beserta jawaban yang benar, lalu murid mencoba memahami pola yang menghubungkan keduanya. Jika murid menjawab salah, guru akan mengoreksi kesalahan tersebut, dan murid memperbaiki pemahamannya. Dalam konteks machine learning, data berperan sebagai soal, label sebagai jawaban guru, dan algoritma

pembelajaran berfungsi untuk menyesuaikan "pemahaman" model agar kesalahannya semakin kecil dari waktu ke waktu.

Supervised learning menjadi landasan utama banyak sistem kecerdasan buatan modern karena konsepnya sangat intuitif dan terbukti efektif. Dari pengenalan wajah di ponsel pintar hingga sistem rekomendasi film di platform streaming, hampir semuanya menggunakan prinsip supervised learning dalam tahap pelatihannya. Secara historis, pendekatan ini berakar dari konsep statistik klasik seperti regresi linear, yang kemudian berkembang pesat dengan dukungan komputasi modern dan algoritma canggih seperti neural network dan support vector machine.

- **Komponen Utama dalam Supervised Learning**

Agar pembelajaran terawasi dapat berjalan, ada beberapa komponen utama yang harus dipahami: dataset, model, fungsi kerugian, dan algoritma optimisasi.

a. Dataset

Dataset merupakan kumpulan data yang digunakan untuk melatih dan menguji model. Setiap data umumnya terdiri atas dua bagian:

- Fitur (features): variabel input yang berisi informasi penting dari data, misalnya luas rumah, jumlah kamar, lokasi, dan tahun pembangunan.
- Label (target): nilai yang ingin diprediksi oleh model, misalnya harga rumah.

Sebelum pelatihan dimulai, dataset biasanya dibagi menjadi dua atau tiga bagian:

- Training set: digunakan untuk melatih model agar belajar dari pola yang ada.
- Validation set (opsional): digunakan untuk menyesuaikan parameter dan menghindari overfitting.
- Test set: digunakan untuk mengukur kemampuan model dalam menghadapi data baru yang tidak pernah dilihat sebelumnya.

Perbandingan umum antara training dan test set biasanya sekitar 80:20 atau 70:30, tergantung pada ukuran dataset.

b. Model

Model dalam supervised learning berperan sebagai fungsi yang memetakan input ke output. Secara umum, model dapat ditulis sebagai:

$$\hat{y} = f(x; \theta)$$

dimana:

- x adalah input (fitur)
- $\hat{y}$ adalah prediksi model
- θ adalah parameter yang akan dipelajari (misalnya bobot pada jaringan syaraf)

Tugas utama algoritma pembelajaran adalah mencari nilai parameter θ terbaik sehingga hasil prediksi $\hat{y}$ sedekat mungkin dengan nilai sebenarnya y

Fungsi kerugian kemudian digunakan untuk mengukur seberapa jauh hasil prediksi model dari label yang benar. Nilai ini akan menjadi acuan bagi proses optimisasi.

Beberapa fungsi kerugian umum antara lain:

Untuk regresi:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

di mana MSE (Mean Squared Error) menghitung rata-rata kuadrat selisih antara prediksi dan nilai sebenarnya.

Untuk klasifikasi, sering digunakan cross-entropy loss, yang mengukur perbedaan antara distribusi probabilitas prediksi dan label sebenarnya.

c. Optimisasi dan Gradient Descent

Agar model semakin baik, parameter θ diperbaharui secara bertahap untuk meminimalkan fungsi kerugian. Salah satu metode paling populer untuk ini adalah gradient descent. Secara intuitif, gradient descent bekerja seperti seseorang yang sedang menuruni lembah yang berkabut: setiap langkah diambil ke arah paling curam yang menurunkan ketinggian. Dalam kasus ini, "ketinggian" mewakili nilai *loss*, dan "arah curam" ditentukan oleh gradien fungsi tersebut.

Proses pembaruan parameternya dapat ditulis sebagai:

$$\theta_{baru} = \theta_{lama} - \alpha\frac{\partial L}{\partial \theta}$$

dimana :

L adalah fungsi kerugian

$\frac{\partial L}{\partial \theta}$ adalah gradiennya

α adalah *learning rate*, yang menentukan seberapa besar langkah pembaruan dilakukan

Supervised learning umumnya dibagi menjadi dua kategori besar: regresi dan klasifikasi.

a. Regresi (Regression)

Regresi digunakan ketika output yang ingin diprediksi bersifat kontinu, artinya berupa angka yang dapat memiliki banyak kemungkinan nilai. Contohnya:

- Memprediksi harga rumah berdasarkan luas, lokasi, dan jumlah kamar.
- Memprediksi suhu udara berdasarkan data cuaca sebelumnya.
- Memprediksi pendapatan seseorang berdasarkan usia dan tingkat pendidikan.

Salah satu algoritma paling sederhana adalah Linear Regression, di mana hubungan antara input dan output diasumsikan linier. Persamaannya adalah:

$$\hat{y} = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

Meski sederhana, model ini menjadi fondasi bagi banyak metode lanjutan seperti Ridge Regression dan Lasso Regression.

b. Klasifikasi (Classification)

Klasifikasi digunakan ketika output bersifat diskrit, yaitu berupa kategori.

Contohnya:

- Mengklasifikasi email sebagai *spam* atau *bukan spam*.
- Menentukan apakah pasien *sehat* atau *sakit* berdasarkan hasil tes medis.
- Mengenali objek pada gambar seperti *kucing*, *anjing*, atau *mobil*.

Dalam klasifikasi, model sering kali menghasilkan probabilitas setiap kelas, lalu memilih kelas dengan probabilitas tertinggi sebagai prediksi akhir.

Algoritma yang umum digunakan antara lain:

- Logistic Regression
- Decision Tree
- K-Nearest Neighbors (KNN)
- Support Vector Machine (SVM)
- Neural Network

Setelah model dilatih, kita perlu mengukur seberapa baik kemampuannya dalam membuat prediksi. Metode evaluasi tergantung pada jenis tugas yang dilakukan.

Untuk regresi:

- Mean Squared Error (MSE): rata-rata kuadrat selisih antara nilai prediksi dan nilai sebenarnya.
- Root Mean Squared Error (RMSE): akar dari MSE, memiliki satuan yang sama dengan target.
- Mean Absolute Error (MAE): rata-rata selisih absolut antara nilai prediksi dan nilai sebenarnya.

- R^2 (R-squared): proporsi variasi dalam data yang bisa dijelaskan oleh model.

Untuk klasifikasi:

- Akurasi: proporsi prediksi yang benar.
- Presisi dan Recall: penting dalam kasus data tidak seimbang, seperti deteksi penyakit langka.
- F1-Score: kombinasi harmonis antara presisi dan recall.
- Confusion Matrix: tabel yang menunjukkan hubungan antara prediksi dan label sebenarnya.

Sebagai contoh, dalam deteksi spam, model dengan akurasi tinggi belum tentu baik jika hampir semua pesan bukan spam. Oleh karena itu, metrik seperti presisi dan recall menjadi sangat penting.

Dua masalah klasik yang sering muncul pada supervised learning adalah:

- Overfitting: model terlalu meniru data latih, bahkan termasuk noise, sehingga gagal mengenali pola umum.
- Analogi: murid menghafal soal dan jawaban tanpa memahami konsep.
- Underfitting: model terlalu sederhana sehingga tidak mampu menangkap pola dalam data. Analogi: murid belajar setengah hati sehingga tidak memahami materi.

Beberapa cara umum untuk mengatasinya:

- Menggunakan regularisasi (misalnya L1 atau L2).
- Melakukan cross-validation.
- Mengumpulkan lebih banyak data atau menambah variasi data.

- Menggunakan teknik early stopping untuk menghentikan pelatihan saat model mulai overfit.

Selain overfitting, beberapa tantangan lain adalah:

- Kualitas data: data yang kotor, hilang, atau salah label bisa menurunkan performa model.
- Imbalanced data: ketika satu kelas jauh lebih banyak dari yang lain, model cenderung bias.
- Generalization: kemampuan model bekerja dengan baik pada data baru, bukan hanya pada data latih.

Supervised learning telah menjadi tulang punggung berbagai teknologi yang kita gunakan setiap hari. Berikut beberapa penerapannya:

1. **Pengenalan Wajah dan Suara**

   Sistem keamanan biometrik menggunakan supervised learning untuk mengenali pola unik wajah atau suara seseorang. Model dilatih dari ribuan contoh wajah untuk mempelajari ciri khas setiap individu.

2. **Diagnosa Medis Otomatis**

   Dalam dunia kesehatan, algoritma supervised learning digunakan untuk membantu dokter mendiagnosis penyakit dari citra medis seperti rontgen atau hasil laboratorium. Misalnya, model klasifikasi dapat membedakan apakah sel darah menunjukkan tanda-tanda kanker.

3. **Sistem Rekomendasi**

   Platform seperti Netflix, Spotify, atau YouTube menggunakan supervised learning untuk memprediksi film atau musik yang

mungkin disukai pengguna berdasarkan riwayat tontonan mereka.

4. **Keuangan dan Bisnis**

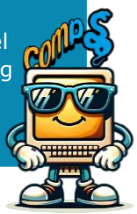
   Dalam dunia finansial, model digunakan untuk mendeteksi penipuan kartu kredit, memprediksi risiko pinjaman, hingga menganalisis tren pasar saham.

5. **Kendaraan Otonom**

   Mobil tanpa pengemudi memanfaatkan supervised learning untuk mengenali rambu lalu lintas, mendeteksi pejalan kaki, dan memperkirakan jarak antar kendaraan berdasarkan data kamera.

Supervised learning adalah paradigma dasar dalam machine learning yang memungkinkan komputer belajar dari contoh berlabel untuk memahami hubungan antara input dan output. Melalui proses pelatihan yang melibatkan dataset, fungsi kerugian, dan optimisasi, model dapat meniru pola yang ada dalam data dan menghasilkan prediksi yang akurat untuk situasi baru. Meskipun konsepnya sederhana, kekuatan supervised learning sangat besar. Ia menjadi fondasi bagi banyak algoritma canggih modern dan aplikasi dunia nyata yang kita gunakan setiap hari. Dengan terus berkembangnya jumlah data dan kemampuan komputasi, supervised learning akan tetap menjadi salah satu pilar utama dalam pengembangan kecerdasan buatan di masa depan.

" Supervised learning adalah metode machine learning di mana model dilatih menggunakan data berlabel untuk mempelajari hubungan antara input dan output yang diketahui. Tujuannya adalah agar model mampu memprediksi output baru secara akurat berdasarkan pola yang telah dipelajari dari data latih".



2.2. Unsupervised Learning

Berbeda dengan *supervised learning* yang membutuhkan data berlabel, unsupervised learning bekerja tanpa adanya label atau jawaban yang benar. Dalam paradigma ini, model hanya diberikan data mentah, tanpa informasi tambahan tentang kategori atau nilai target yang seharusnya. Tantangan model adalah menemukan pola, struktur, atau representasi tersembunyi di balik data itu sendiri (Goodfellow, 2016).

Analogi sederhananya adalah seorang penjelajah yang masuk ke sebuah kota baru tanpa peta dan tanpa petunjuk arah. Ia tidak tahu mana daerah pemukiman, mana kawasan industri, atau di mana pusat kota berada. Namun, dengan mengamati bentuk bangunan, kepadatan orang, dan pola jalan, sang penjelajah lambat laun bisa mengenali pola dan membentuk pemahamannya sendiri tentang struktur kota tersebut. Begitulah cara kerja unsupervised learning: model berusaha memahami data dengan menemukan keteraturan dan kemiripan antar sampel, tanpa bantuan label dari luar.

Pendekatan ini sangat penting dalam dunia *machine learning*, karena pada kenyataannya data berlabel sangat sulit dan mahal untuk diperoleh, sementara data mentah jumlahnya sangat melimpah. Misalnya, jutaan gambar di internet tidak memiliki label yang menjelaskan isinya, dan hanya sebagian kecil data medis memiliki anotasi dari dokter ahli. Dengan memanfaatkan *unsupervised learning*, komputer dapat mengekstraksi pola dan pengetahuan dari data tanpa perlu "diajari" terlebih dahulu.

Inti dari unsupervised learning adalah mencari struktur tersembunyi dalam data. Model mencoba memahami bagaimana data saling berhubungan atau berbeda satu sama lain berdasarkan karakteristiknya. Secara umum, ada dua pendekatan utama dalam unsupervised learning:

1. Clustering (pengelompokan), yaitu menemukan kelompok data yang mirip satu sama lain.
2. Dimensionality Reduction (reduksi dimensi) yaitu menyederhanakan data berdimensi tinggi menjadi representasi yang lebih ringkas.

Clustering adalah proses mengelompokkan data ke dalam beberapa klaster (kelompok) berdasarkan kemiripan karakteristiknya. Data yang berada dalam klaster yang sama diharapkan memiliki kesamaan fitur, sedangkan data antar klaster berbeda secara signifikan.

Salah satu algoritma paling populer adalah K-Means Clustering. Proses kerjanya secara intuitif adalah sebagai berikut:

1. Tentukan jumlah klaster K yang diinginkan.
2. Pilih secara acak K titik awal sebagai pusat klaster (centroid).
3. Setiap data kemudian ditempatkan pada klaster dengan pusat terdekat.
4. Setelah semua data diklasterkan, pusat baru dihitung ulang berdasarkan rata-rata data di klaster tersebut.
5. Langkah 3 dan 4 diulang hingga posisi pusat klaster stabil (tidak banyak berubah).

Dengan cara ini, data yang awalnya tersebar tanpa struktur akan terbagi menjadi beberapa kelompok alami.

Contoh aplikasinya:

- Mengelompokkan pelanggan e-commerce berdasarkan perilaku belanja.
- Mengelompokkan dokumen berdasarkan topik.
- Mengelompokkan gambar berdasarkan kesamaan visual.

Selain K-Means, ada juga algoritma lain seperti:

- Hierarchical Clustering, yang membentuk struktur mirip pohon untuk menggambarkan hubungan antar data.
- DBSCAN, yang mampu mengenali klaster dengan bentuk tak beraturan dan dapat memisahkan *noise* (data yang tidak termasuk klaster manapun).
- Gaussian Mixture Model (GMM), yang mengasumsikan bahwa setiap klaster mengikuti distribusi probabilitas Gaussian.

Agar lebih jelas, mari kita perinci elemen-elemen penting dalam proses unsupervised learning:

a. Representasi Data

Langkah pertama adalah menyiapkan data dalam bentuk yang dapat dipahami oleh algoritma. Biasanya, data dinyatakan dalam bentuk matriks fitur, di mana setiap baris mewakili satu sampel dan setiap kolom mewakili fitur atau karakteristik dari data tersebut.

b. Ukuran Kemiripan (Similarity Measure)

Karena tidak ada label, satu-satunya cara model mengenali pola adalah dengan mengukur kemiripan atau jarak antar data. Beberapa ukuran jarak yang umum digunakan:

- Euclidean Distance — untuk data numerik.
- Cosine Similarity — untuk data vektor seperti teks.
- Manhattan Distance — untuk data yang perbedaan absolutnya lebih bermakna.

Ukuran jarak ini menjadi dasar bagi algoritma untuk menentukan apakah dua data "dekat" atau "jauh" satu sama lain. Tidak seperti supervised learning yang bisa membandingkan hasil prediksi dengan label benar, evaluasi dalam unsupervised learning jauh lebih menantang karena tidak ada kebenaran absolut.

Namun, ada beberapa pendekatan umum:

- Silhouette Score: mengukur seberapa baik setiap data ditempatkan dalam klaster yang benar.
- Davies–Bouldin Index dan Calinski–Harabasz Index: menilai seberapa padat dan terpisah klaster yang dihasilkan.
- Untuk kasus tertentu, jika tersedia sebagian label, dapat dilakukan *external evaluation* seperti *Adjusted Rand Index*.

Walaupun bekerja tanpa pengawasan, unsupervised learning memiliki banyak aplikasi penting dalam berbagai bidang kehidupan:

1. Segmentasi Pelanggan dalam Bisnis

   Perusahaan e-commerce menggunakan clustering untuk mengelompokkan pelanggan berdasarkan pola pembelian, frekuensi transaksi, atau minat produk.

   Dengan informasi ini, strategi pemasaran dapat disesuaikan secara personal, misalnya menawarkan diskon berbeda untuk setiap segmen pelanggan.

2. Analisis Teks dan Dokumen

   Dalam bidang pemrosesan bahasa alami (NLP), topic modeling menggunakan pendekatan unsupervised untuk menemukan tema umum dalam kumpulan dokumen besar tanpa perlu label topik. Contohnya, algoritma seperti Latent Dirichlet Allocation (LDA) bisa mengelompokkan ribuan artikel berita ke dalam kategori seperti politik, olahraga, dan hiburan.

3. Deteksi Anomali (Anomaly Detection)

   Dengan mengenali pola normal pada data, model dapat mendeteksi kejadian yang tidak biasa, seperti penipuan kartu kredit, aktivitas jaringan mencurigakan, atau sensor mesin yang rusak.

4. Komputer Visi

   Dalam bidang pengolahan citra, *unsupervised feature learning* digunakan untuk menemukan representasi visual tanpa label. Misalnya, model dapat belajar mengenali tepi, bentuk, atau tekstur secara otomatis sebelum digunakan untuk tugas klasifikasi yang lebih kompleks.

5. Genomik dan Bioinformatika

   Peneliti menggunakan unsupervised learning untuk mengelompokkan ekspresi gen, menganalisis data DNA, atau menemukan hubungan biologis antar sampel.

6. Pra-pemrosesan Data untuk Deep Learning

   Banyak model modern menggunakan hasil *unsupervised pretraining* untuk mempercepat dan menstabilkan proses pelatihan *supervised* berikutnya.

❖ **Kelebihan Unsupervised Learning:**

- Tidak membutuhkan label, sehingga bisa memanfaatkan data dalam jumlah besar.
- Berguna untuk eksplorasi awal data dan menemukan pola tersembunyi.
- Dapat digunakan untuk pra-pemrosesan dan kompresi data.

❖ **Keterbatasan Unsupervised Learning:**

- Sulit dievaluasi karena tidak ada jawaban benar yang jelas.
- Hasil klaster bisa bervariasi tergantung parameter awal.
- Interpretasi hasil terkadang subjektif.

❖ **Tantangan Unsupervised Learning:**

- Menentukan jumlah klaster yang optimal.
- Menghadapi data berdimensi tinggi dan tidak seimbang.
- Menangani *noise* atau data outlier yang dapat mengganggu hasil.

Pendekatan Unsupervised Learning justru menjadi semakin penting di era *big data*, ketika jumlah data mentah meningkat pesat sementara label sulit didapat. Dengan kemampuan menemukan pola tersembunyi, *unsupervised learning* membuka jalan bagi berbagai inovasi — mulai dari segmentasi pelanggan, analisis teks, hingga penemuan ilmiah dalam bioteknologi. Walaupun bekerja "tanpa guru", unsupervised learning justru mengajarkan kita bahwa pengetahuan sejati sering kali muncul dari eksplorasi dan penemuan pola alami di balik kompleksitas data.

" Unsupervised learning adalah metode machine learning yang digunakan untuk menemukan pola, struktur, atau hubungan tersembunyi dalam data yang tidak memiliki label. Pendekatan ini membantu model memahami karakteristik alami data, seperti pengelompokan atau distribusi, tanpa arahan eksplisit dari target output".

2.3. Reinforcement Learning

Reinforcement Learning (RL) adalah salah satu paradigma pembelajaran mesin yang unik dan berbeda dari dua paradigma utama lainnya: supervised learning dan unsupervised learning. Jika supervised learning belajar dari data berlabel, dan unsupervised learning mencari pola tersembunyi tanpa label, maka reinforcement learning mengambil jalan yang berbeda, algoritma ini belajar melalui interaksi langsung dengan lingkungan (Sutton et al., 1998).

Dalam RL, model pembelajar disebut agen (agent), sedangkan dunia tempat agen berinteraksi disebut lingkungan (environment). Alih-alih diberi contoh input-output seperti dalam supervised learning, agen harus menemukan sendiri strategi terbaik dengan mencoba berbagai tindakan (actions) dan mengamati hasil yang diperoleh dalam bentuk imbalan (reward) atau hukuman (punishment). Tujuan utama agen adalah memaksimalkan reward total jangka panjang, bukan hanya reward sesaat.

Untuk memahami konsep RL secara intuitif, bayangkan Anda sedang melatih seekor anjing. Setiap kali anjing Anda duduk ketika diperintah, Anda memberinya makanan sebagai hadiah (reward). Namun, jika ia mengabaikan perintah, Anda tidak memberinya apa pun, atau bahkan menegurnya (punishment). Dengan pengulangan, anjing belajar bahwa "duduk ketika diperintah" maka akan "mendapat makanan" sebagai hadiah, sehingga ia mulai melakukannya secara konsisten tanpa harus diberi tahu terus-menerus.

Proses yang sama terjadi pada reinforcement learning: agen berinteraksi dengan lingkungan, mencoba berbagai aksi, dan belajar dari umpan balik (feedback) yang diberikan. Melalui pengalaman ini, agen membentuk kebijakan (policy), yaitu aturan atau strategi yang menentukan aksi apa yang harus diambil dalam setiap kondisi tertentu untuk memaksimalkan total reward di masa depan.

Secara matematis, reinforcement learning sering diformalkan menggunakan Markov Decision Process (MDP). MDP menggambarkan dunia RL dengan komponen-komponen berikut:

$$MDP = (S, A, P, R, \gamma)$$

dengan:

- S adalah himpunan state (keadaan lingkungan)
- A adalah himpunan action (tindakan yang dapat diambol oleh agen)
- $P(s'|s,a)$ adalah probabilitas transisi, yaitu peluang berpundah state s ke s' setelah mengambil aksi a
- $R(s,a)$ adalah fungsi reward yaitu imbalan yang diterima agen setelah mengambil aksi a pada state s
- γ faktor diskonto (discount factor) $0 \leq \gamma \leq 1$ yang menentukan seberapa penting reward dimasa depan dibanding reward saat ini.

Tujuan utama agen adalah menemukan policy optimal yaitu fungsi $\pi(a|s)$ yang memaksimalkan nilai harapan reward kumulatif di masa depan:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Agen berusaha memaksimalkan nilai ekspektasi dari G_t tersebut melalui eksplorasi dan pembelajaran dari pengalaman Reinforcement learning beroperasi berdasarkan prinsip trial and error yaitu mencoba berbagai tindakan, mengamati hasilnya, lalu memperbarui strategi berdasarkan hasil yang diperoleh.

Proses ini berlangsung berulang kali hingga agen menemukan perilaku yang paling efektif.

Tahapan umum RL meliputi:

1. Inisialisasi: Agen memulai dengan pengetahuan acak tentang lingkungan.
2. Eksplorasi: Agen mencoba berbagai aksi untuk memahami bagaimana lingkungan merespons.
3. Eksploitasi: Setelah cukup belajar, agen mulai memilih aksi yang diyakini menghasilkan reward tinggi.
4. Evaluasi dan Pembaruan: Agen memperbarui nilai atau kebijakan berdasarkan feedback yang diterima.

Dalam praktiknya, RL harus menyeimbangkan dua hal penting:

- Eksplorasi (exploration): mencoba aksi baru untuk mendapatkan informasi.
- Eksploitasi (exploitation): menggunakan aksi yang sudah terbukti memberikan reward tinggi.

Keseimbangan antara keduanya dikenal sebagai exploration-exploitation trade-off. Jika agen terlalu banyak mengeksploitasi, ia bisa terjebak pada strategi lokal yang suboptimal. Sebaliknya, jika terlalu banyak mengeksplorasi, ia tidak akan pernah memanfaatkan strategi terbaik yang telah ditemukan.

Ada dua pendekatan utama dalam RL, yaitu value-based dan policy-based, serta satu pendekatan gabungan yang dikenal sebagai actor-critic.

a. Value-Based Methods

Pendekatan ini berfokus pada memperkirakan nilai dari suatu state atau aksi, yaitu seberapa baik keputusan yang diambil pada kondisi tertentu.

Fungsi nilai yang umum digunakan:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t|S_t = s]$$

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t|S_t = s, A_t = a]$$

Disini $V^{\pi}(s)$ adalah nilai dari state s dibawah policy π, sedangkan $Q^{\pi}(s,a)$ adalah nilai dari kombinasi state-action.

Salah satu algoritma paling terkenal dari pendekatan ini adalah Q-Learning, dengan rumus pembaruan:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

dimana:

- α adalah learning rate
- R adalah reward yang diterima
- s' adalah state berikutnya
- a' adalah aksi berikutnya

b. Policy-Based Methods

Pendekatan ini langsung mempelajari policy optimal tanpa memperkirakan fungsi nilai secara eksplisit.

Model menggunakan parameter θ untuk mengontrol probabilitas memilih aksi tertentu:

$$\pi_\theta(a|s) = P(A_t = a|S_t = s;\theta)$$

Tujuan pembelajarannya adalah memaksimalkan expected reward melalui metode Policy Gradient, dengan memperbarui parameter:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

dimana $J(\theta)$ adalah fungsi objektif yang merepresentasikan reward rata-rata yang ingin dimaksimalkan

c. Actor-Critic Methods

Metode ini menggabungkan keunggulan dua pendekatan sebelumnya.

- Actor: mempelajari policy $\pi(a|s)$ (bagian yang memutuskan aksi).
- Critic: memperkirakan value function untuk menilai kualitas aksi yang diambil oleh actor.

Actor memperbarui strategi berdasarkan saran dari critic, sedangkan critic belajar mengevaluasi tindakan actor.

Pendekatan ini banyak digunakan dalam algoritma modern seperti A3C (Asynchronous Advantage Actor-Critic) dan PPO (Proximal Policy Optimization).

Walaupun sangat menjanjikan, RL juga menghadapi sejumlah tantangan besar:

- Sample inefficiency: RL membutuhkan banyak percobaan untuk belajar strategi yang baik.
- Sparse rewards: dalam banyak kasus, reward muncul jarang sehingga sulit dioptimalkan.
- Stabilitas pelatihan: proses pembelajaran sering tidak stabil, terutama pada lingkungan kompleks.
- Catastrophic forgetting: ketika agen belajar tugas baru, ia bisa melupakan tugas lama (masalah continual learning).

Untuk mengatasi tantangan ini, berbagai riset sedang dikembangkan, seperti Deep Reinforcement Learning (DRL) yang memadukan RL dengan deep neural network, hierarchical RL, meta-RL, dan multi-agent RL.

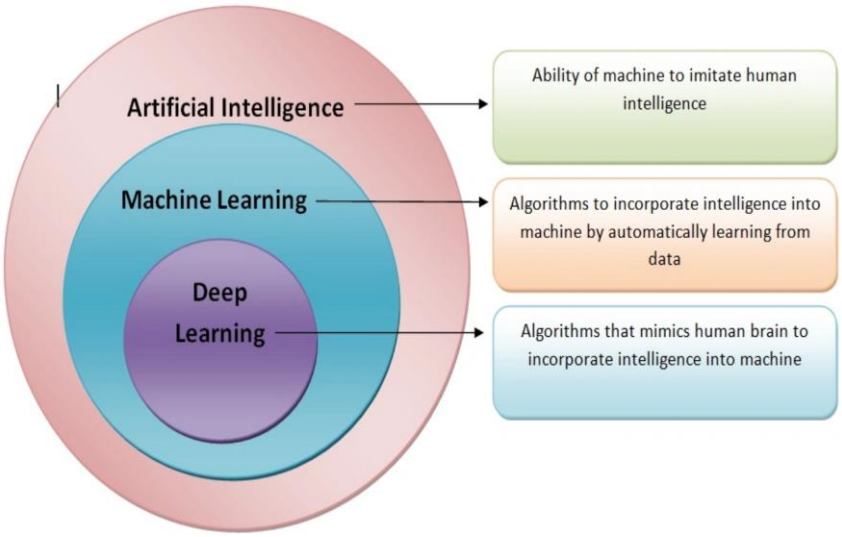
" Reinforcement learning adalah metode machine learning di mana agen belajar melalui interaksi dengan lingkungan untuk memaksimalkan total reward yang diperoleh dari serangkaian tindakan. Proses pembelajarannya berbasis trial-and-error, sehingga agen secara bertahap menemukan strategi optimal untuk mencapai tujuan tertentu".

BAGIAN 3
INTI CARA KERJA MACHINE LEARNING

3.1. Konsep Dasar Machine Learning

Mungkin Anda juga pernah mendengar istilah machine learning dan AI digunakan secara bergantian, seolah keduanya sama. Padahal, keduanya memiliki hubungan yang saling terkait, tetapi tidak identik. AI (Artificial Intelligence) adalah payung besar yang mencakup berbagai teknologi yang memungkinkan mesin meniru kecerdasan manusia, seperti penalaran, pengambilan keputusan, hingga pemahaman bahasa. Penting dipahami bahwa machine learning bukanlah keseluruhan AI melainkan ia hanyalah salah satu cara untuk mewujudkan AI.

Machine Learning (ML) adalah cabang dari kecerdasan buatan (Artificial Intelligence atau AI) yang berfokus pada pengembangan algoritma dan model matematis yang memungkinkan komputer untuk belajar dari data tanpa diprogram secara eksplisit untuk setiap tugas tertentu. Artinya sistem yang menggunakan machine learning dapat meningkatkan kinerjanya seiring waktu berdasarkan pengalaman atau data yang dikumpulkan. Secara sederhana machine learning bekerja dengan cara menganalisis pola-pola dalam data, lalu menggunakan pola tersebut untuk membuat prediksi atau mengambil keputusan di di masa depan. Ilustrasi posisi machine learning dengan AI dan Deep Learning dapat dilihat pada Gambar dibawah ini.



Gambar 12
Ilustrasi posisi keilmuan Artificial Intelligence, Machine Learning, dan Deep Learning

Sebelum melangkah lebih jauh dalam mempelajari machine learning, penting untuk memahami istilah-istilah dasar yang digunakan dalam bidang ini. Setiap algoritma machine learning pada dasarnya bekerja dengan menerima data masukan (input), mempelajari pola yang terdapat di dalamnya, dan kemudian menghasilkan keluaran (output) dalam bentuk angka.

Sebagai contoh, misalkan Anda ingin mendiagnosis suatu penyakit menggunakan algoritma machine learning. Anda dapat menyatakan hasil diagnosis sebagai berikut:

- 1 → orang tersebut terkena penyakit, dan
- 0 → orang tersebut tidak terkena penyakit.

Pendekatan ini dikenal sebagai respons biner (binary response). Namun, dalam banyak kasus, kita mungkin ingin memberikan jawaban yang tidak mutlak. Sebagai gantinya, kita dapat

menggunakan nilai antara 0 dan 1 untuk menunjukkan probabilitas seseorang menderita penyakit tertentu.

- Nilai 0 menunjukkan kemungkinan tidak sakit sama sekali,
- Nilai 1 menunjukkan pasti sakit, dan
- Nilai di antara keduanya (misalnya 0.7) menunjukkan tingkat keyakinan model bahwa seseorang sakit dengan probabilitas 70%.

Agar algoritma machine learning dapat memberikan prediksi yang akurat, ia membutuhkan dua hal penting:

1. Data contoh (sample data), yaitu informasi atau pengamatan yang dikumpulkan dari dunia nyata.
2. Respons yang diketahui (target/label), yaitu hasil sebenarnya yang ingin diprediksi oleh model.

Contohnya, jika Anda memiliki data medis pasien (seperti suhu tubuh, tekanan darah, dan hasil tes darah) serta informasi apakah pasien tersebut sakit atau tidak, maka data medis disebut fitur (features) dan status sakit/tidak sakit disebut label atau respons.

- **Mengenal Konsep Fitur (Feature)**

  Setiap informasi yang digunakan sebagai masukan (input) dalam machine learning disebut fitur (feature) — istilah ini diambil dari dunia statistik. Dalam konteks pemrograman, istilah *feature* digunakan untuk membedakannya dari variabel dalam kode program, agar tidak membingungkan antara *data feature* dan *variable name*.

  Fitur yang baik adalah fitur yang berkorelasi atau berhubungan dengan respons yang ingin diprediksi, sehingga model dapat

belajar fungsi yang memetakan input menjadi output secara akurat.

- **Jenis-Jenis Fitur**

Secara umum, terdapat dua jenis fitur dalam machine learning:

1. Fitur Kuantitatif (Quantitative Features)

   Fitur ini berbentuk angka dan dapat diukur secara numerik, misalnya:

   - ✓ Jumlah penjualan,
   - ✓ Suhu udara,
   - ✓ Umur,
   - ✓ Skor, atau
   - ✓ Ranking.

   Fitur kuantitatif ideal untuk machine learning, karena dapat langsung diolah secara matematis.

2. Fitur Kualitatif (Qualitative Features)

   Fitur ini bersifat simbolik atau kategorikal, misalnya berupa kata, label, atau konsep, seperti:

   - ✓ Warna: merah, hijau, biru
   - ✓ Cuaca: cerah, mendung, hujan
   - ✓ Status: benar atau salah

Fitur seperti ini tidak dapat langsung diproses oleh algoritma, karena komputer hanya dapat memahami angka. Oleh karena itu, fitur kualitatif harus diubah menjadi bentuk numerik sebelum digunakan.

Bayangkan punya asisten pribadi yang makin pintar sendiri setiap hari! Itulah ML - teknologi yang membuat komputer bisa belajar otomatis dari data, menganalisis pola tersembunyi, lalu membuat keputusan cerdas tanpa perlu diajari step-by-step. Ini bukan magic, ini matematika + data yang bikin mesin makin cerdas sendiri!

3.2. Statistik Dalam Machine Learning

Faktanya, statistik dan machine learning memiliki banyak kesamaan. Keduanya sama-sama berfokus pada bagaimana memahami pola dalam data serta membuat prediksi berdasarkan informasi yang tersedia. Statistik berperan penting karena menyediakan metode matematis untuk menganalisis data, menafsirkan hasil, dan memvalidasi kebenaran suatu model.

Dengan kata lain, tanpa kontribusi statistik, machine learning tidak akan memiliki pijakan kuat dalam hal analisis data, inferensi, dan prediksi yang terukur. Statistik adalah jembatan yang menghubungkan data mentah dengan algoritma cerdas, sehingga mesin mampu belajar layaknya manusia.

Statistik bukan hanya alat bantu matematis, namun ia adalah bahasa utama dalam machine learning (Bishop & Nasrabadi, 2006a). Kegunaan statistik dalam machine learning antara lain:

- Statistik digunakan untuk membersihkan, menganalisis, dan memahami data sebelum model dilatih.
- Konsep seperti sampling, distribusi, mean, variance, dan standard deviation menjadi dasar dari normalisasi fitur, estimasi parameter, dan evaluasi performa model.
- Algoritma Naïve Bayes mengandalkan distribusi probabilitas antar fitur.
- Linear Regression meminimalkan error yang didefinisikan secara statistik, dsb.

Statistik memberikan cara untuk:

1. Menggambarkan data secara ringkas (melalui mean dan median),
2. Memahami penyebaran data (melalui variance dan standard deviation),
3. Mengambil kesimpulan dari data terbatas (melalui sampling),
4. Mengantisipasi ketidakpastian (melalui distribusi probabilitas).

Dalam pembelajaran mesin, statistik adalah fondasi yang membuat algoritma "mengerti" dunia melalui data — meskipun data itu tidak lengkap, tidak sempurna, dan penuh variasi karena salam dunia nyata, kita jarang memiliki data yang lengkap atau sempurna. Ada dua alasan utama:

1. Keterbatasan pengamatan

   Tidak mungkin menghimpun semua kejadian di dunia. Misalnya, jika kita ingin membangun model yang mengenali penyakit langka, tentu kita tidak memiliki data dari seluruh pasien di dunia — hanya sebagian kecil (sampel) yang bisa dikumpulkan
2. Kesalahan pengukuran

   Setiap alat ukur memiliki tingkat ketidakakuratan. Bahkan ketika kita menimbang berat badan sendiri, hasilnya mungkin sedikit berbeda setiap kali kita naik ke timbangan. Hal ini disebabkan oleh error pengukuran, ketidakstabilan alat, atau gangguan acak (noise) selama proses pengamatan.

Dengan demikian, data yang digunakan dalam machine learning selalu bersifat parsial dan tidak sempurna. Namun, ini bukan hal yang buruk. Justru, dengan memahami prinsip statistik, kita bisa tetap memperoleh kesimpulan yang andal dari data yang terbatas.

Dalam statistik:

1. Populasi adalah keseluruhan himpunan data atau peristiwa yang ingin kita pelajari.
2. Sampel adalah sebagian dari populasi yang diambil untuk dianalisis.

Karena mengumpulkan seluruh populasi hampir mustahil, machine learning bekerja dengan sampel. Agar hasil dari sampel dapat mewakili populasi secara akurat, kita menggunakan teknik sampling (pengambilan sampel). Dua metode yang umum digunakan adalah:

1. Random Sampling (Sampel Acak)
   Pemilihan data dilakukan secara acak tanpa bias. Tujuannya adalah agar peluang setiap anggota populasi untuk terpilih sama besar. Metode ini efektif karena, secara teori, distribusi nilai pada sampel akan menyerupai distribusi nilai pada populasi.
2. Stratified Sampling (Sampel Bertingkat)
   Dalam praktiknya, pengambilan acak murni bisa menghasilkan ketidakseimbangan, misalnya terlalu banyak data dari satu kelompok dan terlalu sedikit dari kelompok lain.

Untuk mengatasi hal ini, stratified sampling membagi populasi ke dalam kelompok (strata) berdasarkan kategori tertentu (misalnya jenis kelamin, umur, atau wilayah), lalu melakukan sampling secara proporsional dari setiap kelompok. Teknik ini memastikan bahwa setiap kelompok terwakili secara adil dalam sampel.

Tanpa statistik, machine learning bagai kapal tanpa kompas — statistik memberikan fondasi matematis yang memungkinkan algoritma memahami pola dari data yang tidak sempurna, mengatasi keterbatasan sampel dan noise melalui konsep seperti distribusi probabilitas, sampling representatif, dan pengukuran variabilitas, sehingga mesin tetap bisa belajar efektif dari real-world data yang serba terbatas dan penuh ketidakpastian.

3.3. Kalkulus Dalam Machine Learning

Kalkulus adalah alat matematika untuk memahami perubahan (Goodfellow, 2016). Dalam machine learning, kalkulus digunakan untuk:

- Mengukur bagaimana error berubah terhadap parameter model
- Membantu algoritma optimasi (seperti Gradient Descent) menemukan parameter terbaik

Tanpa kalkulus neural network tidak bisa belajar. Konsep utama meliputi:

- Turunan (Derivative): Mengukur perubahan fungsi terhadap variabel, digunakan untuk menemukan gradien dalam optimasi seperti Gradient Descent.
- Gradien: Vektor turunan parsial yang menunjukkan arah perubahan tercepat dari suatu fungsi. Penting untuk memperbarui parameter model selama pelatihan.
- Integral: Digunakan dalam beberapa aplikasi ML, seperti menghitung probabilitas kumulatif dalam distribusi kontinu.
- Optimasi: Teknik seperti Gradient Descent menggunakan kalkulus untuk meminimalkan fungsi kerugian (loss function) guna menemukan parameter model yang optimal.

Contoh aplikasi: Dalam pelatihan neural network, kalkulus digunakan untuk menghitung gradien dari fungsi kerugian terhadap bobot model.

Jenis-jenis Matriks yang Digunakan pada Machine Learning

Dalam konteks matematika dan pembelajaran mesin (*machine learning*), terdapat beberapa jenis matriks yang memiliki sifat khusus dan fungsi yang berbeda-beda. Mengenal jenis-jenis matriks ini penting karena banyak algoritma yang memanfaatkan karakteristik khusus dari masing-masing bentuk matriks untuk mempercepat perhitungan atau menjaga kestabilan numerik.

1. **Matriks Nol (Zero Matrix)**

   Matriks nol adalah matriks yang semua elemennya bernilai 0. Matriks ini berfungsi seperti angka nil dalam operasi aljabar biasa. Sebagai contoh:

   $$O = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

   Jika A adalah matriks berukuran sama dengan O, maka berlaku:

   $$A + O = A$$

   matriks nol digunakan sebagai elemen identitas dalam penjumlahan matriks dan sering muncul dalam representasi kesalahan atau gradien awal yang belum terisi nilai.

2. **Matriks Identitas (Identity Matrix)**

   Matriks identitas adalah matriks bujur sangkar yang memiliki nilai 1 pada diagonal utama dan 0 pada posisi lainnya. Contoh matriks identitas berukuran 3 x 3:

   $$\mathrm{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

   Matriks identitas berfungsi sebagai elemen netral dalam perkalian matriks, karena:

   $$\mathrm{A \times I = I \times A = A}$$

Dalam machine learning, matriks identitas sering digunakan dalam regularisasi dan transformasi linier yang mempertahankan bentuk data tanpa perubahan.

3. Matriks Diagonal

Matriks diagonal adalah matriks bujur sangkar yang semua elemennya bernilai nol kecuali oada diagonal utama. Sebagai contoh:

$$\begin{bmatrix} 3 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 7 \end{bmatrix}$$

Operasi perkalian dengan matriks diagonal sangat efisien karena hanya mempengaruhi nilai pada posisi diagonal. Matriks diagonal sering digunakan untuk merepresentasikan bobot atau skala fitur tertentu model machine learning.

4. Matriks Simetris

Matriks simetris adalah matriks bujur sangkar yang sama dengan transposenya, yaitu:

$$A = A^T$$

Artinya setiap elemen pada posisi baris ke-i dan kolom ke-j sama dengan elemen baris ke-j kolom ke-i. Secara simbolik dituliskan sebagai:

$$a_{ij} = a_{ji} \text{ untuk semua } i, j$$

Contoh matriks simetris adalah:

$$\begin{bmatrix} 2 & 3 \\ 3 & 4 \end{bmatrix}$$

5. Matriks Ortogonal

Matriks ortogonal adalah matriks yang jika dikalikan dengan transposenya menghasilkan matriks identitas

$$A^T A = AA^T = I$$

Artinya setiap kolom (dan baris) pada matriks ortogonal merupakan vektor yang saling tegak lurus dan memiliki panjang 1. Contoh matriks ortogonal adalah:

$$\mathrm{Q} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Matriks ortogonal penting dalam komputasi numerik karena menjaga stabilitas perhitungan. Dalam machine learning, jenis matriks ini digunakan dalam dekomposisi ortogonal, rotasi data, dan reduksi dimensi seperti pada Principal Component Analysis (PCA)

6. Matriks Segitiga

Matriks segitiga adalah matriks bujur sangkar yang memiliki nilai nol di satu sisi diagonal utama. Jika elemen-elemen dibawah diagonal utama bernilai nol, disebut matriks segitiga atas (upper triangular). Jika elemen-elemen di atas diagonal utama bernilai nol disebut matriks segitiga bawah (lower triangular)

Contoh matriks segitiga atas:

$$U = \begin{bmatrix} 2 & 4 & 6 \\ 0 & 5 & 8 \\ 0 & 0 & 7 \end{bmatrix}$$

Jenis matriks ini sering muncul dalam dekomposisi LU (Lower-Upper Decomposition), yang digunakan untuk menyelesaikan sistem persamaan linear secara efisien.

Penggunaan Matriks Pada Machine Learning

Dalam machine learning, matriks merupakan struktur data yang sangat penting karena mampu merepresentasikan kumpulan data dan operasi matematis secara efisien (Murphy, 2012). Sebagian besar algoritma machine learning, mulai dari regresi linear hingga deep learning, didasarkan pada operasi aljabar linear, seperti perkalian matriks, transpos, invers, dan dekomposisi. Matriks memudahkan komputer untuk melakukan komputasi dalam jumlah besar secara paralel menggunakan pustaka numerik seperti NumPy, TensorFlow, atau PyTorch.

Setiap dataset biasanya disusun dalam bentuk matriks fitur, di mana:

$$\mathrm{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix}$$

Keterangan:

- X: matriks fitur berukuran $n \times m$
- n: jumlah sampel (data point)
- m: jumlah fitur (variabel input)
- x_{ij}: nilai fitur ke-j dari sampel ke-i

Contoh: jika kita memiliki dataset tinggi dan berat badan 100 orang, maka $n = 100$, dan $m = 2$.

Pada model linear sederhana, prediksi dapat ditulis dalam bentuk:

$$\mathrm{y} = X\mathrm{w} + \mathrm{b}$$

Keterangan:

- X: matriks input berukuran $n \times m$
- w: vektor bobot (parameter model) berukuran $m \times 1$
- b: bias (dapat berupa skalar atau vektor)
- y: hasil prediksi berukuran $n \times 1$

Model ini menjadi dasar bagi regresi linear, klasifikasi logistik, dan bahkan neural network. Setiap baris dari Xdikalikan dengan bobot $\mathrm{w}$untuk menghasilkan prediksi pada setiap sampel.

Dalam jaringan saraf tiruan (neural network), operasi perkalian matriks digunakan di setiap lapisan (layer) untuk mentransformasi representasi data.

Misalnya pada satu lapisan neuron:

$$\mathrm{z} = XW + \mathrm{b}$$
$$\mathrm{a} = f(\mathrm{z})$$

Keterangan:

- X: input dari lapisan sebelumnya $(n \times m)$
- W: matriks bobot $(m \times k)$ yang menghubungkan $m$neuron input ke $k$neuron output
- f: fungsi aktivasi non-linear (misalnya ReLU, sigmoid)
- a: output aktivasi dari lapisan tersebut

Di sini, perkalian matriks memungkinkan ratusan atau ribuan neuron beroperasi secara bersamaan dalam satu operasi vektor, mempercepat proses komputasi.

Selama proses pelatihan, pembaruan bobot dilakukan menggunakan gradien yang juga direpresentasikan sebagai matriks:

$$W_{\text{baru}} = W_{\text{lama}} - \eta \frac{\partial L}{\partial W}$$

Keterangan:

- L: fungsi loss (misalnya Mean Squared Error atau Cross Entropy)
- $\frac{\partial L}{\partial W}$: turunan parsial loss terhadap bobot (berbentuk matriks dengan ukuran sama dengan W)
- η: learning rate

Dengan demikian, proses *backpropagation* dalam deep learning sesungguhnya adalah serangkaian operasi matriks dan turunan matriks.

Kalkulus berperan sebagai kompas yang memandu proses pembelajaran mesin melalui turunan dan gradien, sementara matriks bertindak sebagai mesin yang menggerakkan komputasi data secara efisien. Tanp kalkulus, model tidak tahu harus belajar ke mana, tanpa matriks, komputasi akan terlalu lambat untuk data skala besar.

3.4. Vektorisasi Dalam Machine Learning

Dalam machine learning, operasi matematis sering kali melibatkan perhitungan dalam skala besar, seperti ribuan hingga jutaan data dan parameter. Jika setiap operasi dilakukan satu per satu menggunakan perulangan (loop), waktu komputasi akan menjadi sangat besar. Untuk mengatasi hal ini, digunakan konsep vektorisasi (vectorization) — yaitu proses mengekspresikan operasi matematika dalam bentuk operasi matriks dan vektor sehingga seluruh perhitungan dapat dilakukan secara paralel dan efisien. Vektorisasi memungkinkan komputer untuk memproses banyak elemen sekaligus menggunakan arsitektur perangkat keras seperti CPU modern dan GPU, yang dioptimalkan untuk operasi berbasis matriks (Goodfellow, 2016).

Secara matematis, vektorisasi berarti menuliskan operasi yang semula bersifat elemen demi elemen kedalam bentuk aljabar linear maka seluruh proses perkalian dan penjumlahan dilakukan sekaligus oleh komputer dalam satu langkah vektorisasi, tanpa memerlukan perulangan eksplisit.

Vektorisasi memberikan beberapa keuntungan utama:

1. Efisiensi Komputasi

   Operasi matriks dijalankan langsung oleh pustaka numerik tingkat rendah yang sangat teroptimasi, sehingga kecepatan komputasi meningkat drastis.

2. Kesederhanaan Formulasi Matematis
   Rumus yang kompleks dapat dinyatakan dengan lebih ringkas dalam bentuk matriks, memudahkan analisis teoritis.
3. Konsistensi Struktural
   Semua data dan parameter dinyatakan dalam bentuk vektor atau matriks, sehingga mudah dikombinasikan, dimodifikasi, atau dioptimalkan.

One Hot Encoding

One hot encoding digunakan untuk Mengubah Data Kualitatif Menjadi Numerik. Misalkan kita angkat kasus dimana data cuaca untuk membantu mesin menentukan apakah seseorang sebaiknya bermain tenis di luar ruangan. Dataset yang digunakan terdiri atas empat fitur:

- Outlook: cerah (*sunny*), mendung (*overcast*), hujan (*rain*)
- Temperature: sejuk (*cool*), sedang (*mild*), panas (*hot*)
- Humidity: tinggi (*high*), normal (*normal*)
- Windy: benar (*true*), salah (*false*)

Karena komputer tidak bisa langsung memahami simbol seperti *sunny* atau *rain*, maka data tersebut harus diubah menjadi angka. Salah satu teknik paling sederhana yang digunakan adalah One-Hot Encoding.

Dengan *one-hot encoding*, setiap nilai simbolik diubah menjadi fitur biner baru yang bernilai 0 atau 1.

Sebagai contoh, fitur Outlook akan diubah menjadi tiga fitur baru:

- Outlook:Sunny
- Outlook:Overcast
- Outlook:Rain

Jika kondisi hari ini adalah cerah (*sunny*), maka nilai ketiganya akan menjadi:

- Outlook:Sunny = 1
- Outlook:Overcast = 0
- Outlook:Rain = 0

Dengan cara ini, algoritma machine learning dapat memahami informasi kualitatif dalam bentuk numerik, tanpa kehilangan maknanya.

Fungsi Pemetaan

Setelah memahami apa itu fitur (feature) dan respons (response), langkah berikutnya adalah memahami hubungan matematis antara keduanya. Dalam machine learning, hubungan ini direpresentasikan sebagai fungsi pemetaan yang berusaha mengubah data masukan menjadi keluaran yang diinginkan. Dalam konteks matematika, fungsi pemetaan dapat digambarkan sebagai:

$$f: X \rightarrow Y$$

Artinya, fungsi f memetakan sekumpulan data masukan X (fitur) ke sekumpulan keluaran Y (respons). Dalam machine learning, fungsi ini biasanya tidak diketahui secara eksplisit. Oleh karena itu, tujuan utama algoritma machine learning adalah mempelajari

bentuk terbaik dari fungsi f berdasarkan contoh-contoh data yang sudah diketahui hasilnya.

Contoh sederhana, jika kita memiliki data tinggi dan berat badan seseorang, maka machine learning dapat mempelajari fungsi:

$$f(tinggi, berat) = Indikasi\ obesitas$$

Fungsi f disini tidak diberikan secara langsung, melainkan ditemukan learned melalui pross pelatihan (training).

Vektorisasi menggantikan loop tradisional dengan operasi matriks terpadu, memanfaatkan kemampuan paralelisasi CPU/GPU untuk mempercepat komputasi skala besar. Bersama one-hot encoding yang mengubah data kategorikal menjadi representasi biner, dan fungsi pemetaan yang memodelkan hubungan antara fitur dan target, ketiganya membentuk fondasi efisiensi dan efektivitas dalam transformasi dan pemrosesan data ML.

3.5. Konsep Model dalam Machine Learning

Fungsi f yang telah dipelajari dari data disebut model. Model adalah representasi matematis dari pola yang terdapat dalam data. Selama proses pelatihan, algoritma machine learning mencari bentuk fungsi f yang paling tepat dengan cara menyesuaikan parameter-parameter internal agar hasil prediksi mendekati nilai sebenarnya (label).

Sebagai ilustrasi:

- Input (fitur): luas rumah, jumlah kamar, lokasi
- Output (respons): harga rumah
- Tujuan model: menemukan hubungan antara fitur dan harga agar dapat memprediksi harga rumah baru yang belum pernah dilihat.

Training dalam Machine Learning

Bagi banyak orang, konsep awal tentang pemrograman cukup sederhana: kita memiliki sebuah fungsi, kemudian fungsi tersebut menerima data sebagai input, dan menghasilkan output tertentu. Contohnya, seorang programmer menulis fungsi Add() yang menerima dua angka sebagai input, misalnya 1 dan 2. Fungsi tersebut akan menghasilkan output 3.

$$Add(1,2) = 3$$

Dalam paradigma pemrograman tradisional, seorang programmer harus menentukan secara jelas fungsi apa yang digunakan untuk memanipulasi data agar menghasilkan output sesuai keinginan.

Machine learning membalikkan paradigma tersebut.

- Kita sudah tahu input (misalnya angka 1 dan 2).
- Kita juga sudah tahu output yang diinginkan (misalnya hasilnya 3).
- Tetapi, kita tidak tahu fungsi apa yang seharusnya dipakai untuk menghubungkan input ke output.

Disinilah fungsi dari training. Training adalah proses di mana sebuah algoritma pembelajar (*learner algorithm*) diberi banyak contoh pasangan input-output yang benar. Dari data ini, algoritma berusaha menemukan atau *membangun fungsi* yang dapat memetakan input menjadi output (Mitchell, 1997).

Dengan kata lain:

- Programmer tidak lagi menuliskan fungsi Add() secara eksplisit.
- Sebaliknya, algoritma *belajar sendiri* fungsi tersebut berdasarkan contoh-contoh yang diberikan.

Hasil dari Training adalah output dari proses training yang biasanya berupa:

- Probabilitas sebuah kelas → misalnya, apakah sebuah email termasuk kategori *spam* atau *bukan spam*.
- Nilai numerik → misalnya, prediksi harga rumah berdasarkan ukuran dan lokasinya.

Jadi, training bukan hanya sekadar memberi data, melainkan juga mengajari mesin untuk menyusun fungsi yang fleksibel sehingga mampu memprediksi hasil dengan benar, meskipun pada data baru yang belum pernah dilihat sebelumnya.

Pada machine learning, setiap algoritma mengubah data input dan hasil yang diharapkan menjadi sebuah fungsi. Fungsi inilah yang digunakan untuk memprediksi atau mengambil keputusan. Namun perlu dicatat bahwa fungsi yang terbentuk akan selalu spesifik terhadap jenis tugas yang dipelajari. Artinya, meskipun algoritma dasar yang digunakan sama, fungsi hasil belajar tidak bisa dipakai sembarangan.

- Fungsi yang terbentuk untuk *bermain catur* tidak bisa digunakan untuk *mendiagnosis kanker*.
- Fungsi yang dilatih untuk *mengenali wajah* tidak otomatis bisa dipakai untuk *menerjemahkan bahasa*.

Konsep model dalam Machine Learning merujuk pada representasi matematis atau komputasional yang mempelajari pola dari data untuk membuat prediksi atau keputusan, dan kualitas serta kemampuan generalisasinya sangat bergantung pada data pelatihan, algoritma yang digunakan, serta proses optimisasi yang dilakukan.

3.6. Generalisasi

Salah satu kunci utama dalam machine learning adalah generalisasi. Generalisasi berarti kemampuan algoritma untuk membangun fungsi yang tidak hanya bekerja pada data yang digunakan saat training, tetapi juga dapat bekerja dengan baik pada data baru yang belum pernah dilihat sebelumnya (Goodfellow, 2016).

Jika sebuah algoritma hanya mengingat data training tanpa bisa mengenali pola yang lebih umum, maka algoritma tersebut akan gagal saat dihadapkan pada data nyata (*real-world data*). Inilah yang disebut dengan overfitting yaitu ketika model terlalu menempel pada data training sehingga tidak bisa beradaptasi dengan variasi baru. Sebaliknya, jika model mampu menggeneralisasi dengan baik, maka meskipun data yang masuk baru model mampu mengenalinya secara baik.

Analoginya adalah seorang guru tidak bisa mengajarkan semua kemungkinan soal matematika kepada muridnya. Yang dilakukan guru adalah memberikan beberapa contoh soal, lalu mengajarkan pola atau aturan umumnya. Dengan demikian, ketika murid menghadapi soal baru yang berbeda, ia tetap bisa menyelesaikannya dengan benar. Begitu pula dengan machine learning: generalization adalah kemampuan mesin untuk "belajar aturan umum dari contoh terbatas" agar bisa menghadapi situasi nyata yang jauh lebih komplek

Pipeline Machine Learning

Pipeline Machine Learning adalah alur kerja yang terdiri dari langkah-langkah berurutan untuk mengembangkan, melatih, dan mendeploy model ML. Pipeline ini memastikan bahwa proses pengembangan model terstruktur, dapat direproduksi, dan efisien. Langkah-langkah utama meliputi:

1. Pengumpulan dan Praproses Data: Mengumpulkan data berkualitas tinggi dan mempersiapkannya untuk pelatihan model.
2. Rekayasa Fitur (Feature Engineering): Membuat atau memilih fitur yang relevan untuk meningkatkan performa model.
3. Pemilihan Model dan Penyetelan Hiperparameter: Memilih algoritma yang sesuai dan mengoptimalkan parameternya.
4. Evaluasi Model: Mengukur performa model menggunakan metrik yang sesuai dengan tujuan bisnis atau penelitian.
5. Deployment dan Monitoring: Mengintegrasikan model ke dalam sistem produksi dan memantau performanya.

Pipeline ini sering diulang secara iteratif untuk meningkatkan model berdasarkan hasil evaluasi atau data baru.

Data Collection & Preprocessing

Data adalah fondasi dari Machine Learning. Kualitas dan kuantitas data sangat memengaruhi performa model. Proses pengumpulan dan praproses data melibatkan langkah-langkah berikut:

1. Pengumpulan Data
    - Sumber Data: Data dapat berasal dari berbagai sumber, seperti database internal, API publik, web scraping, sensor

IoT, atau dataset open-source (misalnya, Kaggle, UCI Repository).

- Jenis Data: Data bisa terstruktur (tabel numerik atau kategorikal) atau tidak terstruktur (teks, gambar, audio).
- Pertimbangan: Pastikan data representatif, relevan dengan masalah, dan mematuhi regulasi privasi seperti GDPR atau UU Perlindungan Data Pribadi.
- Contoh: Untuk prediksi harga rumah, data dapat mencakup luas tanah, jumlah kamar, lokasi, dan harga historis dari database real estate.

2. Praproses Data

   Praproses data bertujuan untuk membersihkan dan menyiapkan data agar dapat digunakan oleh model ML. Langkah-langkah utama meliputi:

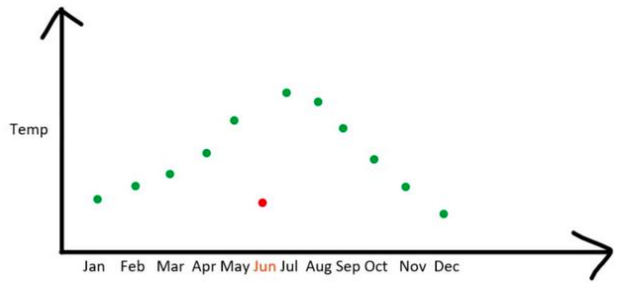
   a) Penanganan Data Hilang:

   - ✓ Identifikasi: Deteksi nilai yang hilang menggunakan analisis seperti isnull() pada Python.
   - ✓ Strategi:
     - Imputasi: Mengisi nilai hilang dengan mean, median, atau modus (untuk data numerik) atau kategori paling umum (untuk data kategorikal).
     - Penghapusan: Menghapus baris/kolom dengan data hilang jika proporsinya kecil.
     - Model-based: Menggunakan model ML sederhana untuk memprediksi nilai hilang.

- ✓ Contoh: Dalam dataset medis, jika kolom "tekanan darah" memiliki nilai hilang, imputasi dengan median dapat digunakan untuk data numerik.

b) Penanganan Outlier:

- ✓ Identifikasi: Gunakan metode statistik seperti IQR (Interquartile Range) atau Z-score untuk mendeteksi outlier.
- ✓ Strategi:
  - Penghapusan: Menghapus outlier jika tidak relevan.
  - Transformasi: Menggunakan log transform atau winsorization untuk mengurangi dampak outlier.
- ✓ Contoh: Dalam prediksi harga rumah, rumah dengan harga ekstrem (misalnya, miliaran di daerah murah) dapat dianggap outlier dan dihapus.



Gambar 13
Ilustrasi outlier (titik data merah di bulan juni)
(Sumber : Wikimedia)

c) Encoding Data Kategorikal:

- ✓ One-Hot Encoding: Mengubah variabel kategorikal menjadi vektor biner (misalnya, "merah", "biru" menjadi [1,0], [0,1]).

- ✓ Label Encoding: Mengubah kategori menjadi angka (misalnya, "merah"=0, "biru"=1), cocok untuk data ordinal.
- ✓ Contoh: Dalam dataset pelanggan, kolom "kota" (Jakarta, Bandung, Surabaya) dapat diubah menjadi one-hot encoding.

d) Normalisasi/Skala Data:

- ✓ Min-Max Scaling: Menyesuaikan data ke rentang [0,1].
- ✓ Standardization: Mengubah data agar memiliki mean=0 dan deviasi standar=1.
- ✓ Contoh: Dalam dataset dengan fitur "usia" (20-80 tahun) dan "pendapatan" (juta rupiah), normalisasi memastikan kedua fitur memiliki skala yang sebanding.

e) Penanganan Ketidakseimbangan Data:

- ✓ Oversampling: Menambah sampel kelas minoritas (misalnya, SMOTE).
- ✓ Undersampling: Mengurangi sampel kelas mayoritas.
- ✓ Contoh: Dalam deteksi penipuan kartu kredit, data penipuan (kelas minoritas) dapat di-oversampling untuk menyeimbangkan dataset.

f) Pembersihan Data: Menghapus duplikat, memperbaiki kesalahan penulisan, atau menstandardisasi format (misalnya, format tanggal).

g) Pemisahan Data: Membagi data menjadi set pelatihan (training), validasi, dan pengujian (testing) dengan rasio umum seperti 70:15:15 atau 80:10:10.

Feature Engineering

Feature engineering adalah proses menciptakan atau memilih fitur baru dari data mentah untuk meningkatkan performa model. Fitur yang baik dapat mempercepat pelatihan, meningkatkan akurasi, dan mengurangi overfitting.

Langkah-Langkah Feature Engineering

1. Ekstraksi Fitur:
    a) Membuat fitur baru dari data mentah, misalnya:
        ✓ Dari kolom tanggal: Ekstrak hari, bulan, tahun, atau apakah hari itu akhir pekan.
        ✓ Dari teks: Ekstrak panjang teks, jumlah kata unik, atau frekuensi kata kunci.
    b) Contoh: Dalam analisis sentimen, fitur seperti jumlah kata positif/negatif dapat diekstrak dari ulasan teks.
2. Transformasi Fitur:
    a) Log Transform: Mengurangi efek nilai ekstrem (misalnya, untuk pendapatan).
    b) Polynomial Features: Membuat kombinasi fitur (misalnya, (x_1 \cdot x_2)) untuk menangkap hubungan non-linier.
    c) Binning: Mengubah data numerik menjadi kategorikal (misalnya, usia menjadi kelompok "muda", "dewasa", "lansia").
    d) Contoh: Dalam prediksi harga rumah, fitur "luas tanah" dapat diubah menjadi log untuk mengurangi skewness.

3. Seleksi Fitur:
    a) Filter Methods: Memilih fitur berdasarkan statistik seperti korelasi atau chi-square test.
    b) Wrapper Methods: Menggunakan model (misalnya, Recursive Feature Elimination) untuk mengevaluasi kombinasi fitur.
    c) Embedded Methods: Memilih fitur selama pelatihan model (misalnya, regularisasi L1 pada regresi linier).
    d) Contoh: Dalam klasifikasi spam, fitur dengan korelasi rendah terhadap label dapat dihapus.
4. Domain Knowledge: Menggunakan pengetahuan spesifik domain untuk membuat fitur, misalnya, rasio utang terhadap pendapatan untuk prediksi kredit.

- **Model Selection & Hyperparameter Tuning**

Memilih model yang tepat dan menyetel hiperparameternya adalah langkah kunci untuk mencapai performa optimal.

Pemilihan Model

1. Jenis Model:
    a) Regresi: Linear Regression, Ridge, Lasso untuk prediksi nilai kontinu.
    b) Klasifikasi: Logistic Regression, Decision Tree, Random Forest, SVM, Neural Networks untuk prediksi kelas.
    c) Clustering: K-Means, Hierarchical Clustering untuk pengelompokan data tanpa label.
    d) Deep Learning: CNN untuk gambar, RNN/LSTM untuk data berurutan, Transformer untuk NLP.

2. Pertimbangan:
    a) Kompleksitas Data: Model sederhana seperti regresi linier cocok untuk data linier, sedangkan model kompleks seperti neural networks diperlukan untuk data non-linier.
    b) Ukuran Data: Model seperti SVM atau Random Forest lebih cocok untuk dataset kecil hingga sedang, sedangkan Deep Learning membutuhkan data besar.
    c) Interpretasi: Model seperti Decision Tree lebih mudah diinterpretasikan dibandingkan neural networks.
    d) Sumber Daya: Pertimbangkan ketersediaan GPU/TPU untuk model kompleks.
3. Contoh: Untuk klasifikasi teks pendek, Random Forest atau Logistic Regression dapat digunakan, tetapi untuk penerjemahan mesin, Transformer lebih cocok.

- **Penyetelan Hiperparameter**

Hiperparameter adalah parameter yang ditentukan sebelum pelatihan, seperti learning rate atau jumlah pohon di Random Forest.

1. Metode Penyetelan:
    a) Grid Search: Mencoba semua kombinasi hiperparameter dalam rentang tertentu. Lambat tetapi menyeluruh.
    b) Random Search: Mencoba kombinasi acak, lebih cepat untuk ruang hiperparameter besar.
    c) Bayesian Optimization: Menggunakan model probabilitas untuk memprediksi kombinasi hiperparameter terbaik, lebih efisien.

d) Contoh: Dalam Random Forest, hiperparameter seperti jumlah pohon (n_estimators) dan kedalaman pohon (max_depth) dapat disetel menggunakan Grid Search.

2. Validasi Silang (Cross-Validation):
   a) Membagi data pelatihan menjadi beberapa lipatan (folds) untuk menguji performa model secara robust.
   b) Contoh: K-Fold Cross-Validation (misalnya, 5-fold) memastikan model diuji pada subset data yang berbeda.
3. Alat: Scikit-learn (GridSearchCV, RandomizedSearchCV), Optuna, atau Hyperopt untuk penyetelan otomatis.

Tantangan:

- Penyetelan hiperparameter memakan waktu dan sumber daya.
- Overfitting pada data validasi jika penyetelan terlalu agresif.

Evaluasi Model

Dalam dunia machine learning, setelah kita melatih sebuah model, kita perlu mengetahui seberapa baik model tersebut bekerja. Proses ini disebut evaluasi model. Evaluasi sangat penting karena model yang terlihat bagus pada data latih belum tentu mampu memprediksi dengan baik pada data baru. Untuk itu, kita memerlukan metrik evaluasi, yaitu ukuran atau indikator yang digunakan untuk menilai kinerja model. kita akan fokus pada metrik evaluasi untuk masalah klasifikasi, yaitu jenis tugas di mana model berusaha mengelompokkan data ke dalam kelas tertentu — misalnya, mengklasifikasikan email menjadi spam atau bukan

spam, atau mendeteksi apakah gambar mengandung kucing atau tidak (Goodfellow, 2016).

Salah satu alat paling dasar dan penting dalam evaluasi klasifikasi adalah Confusion Matrix. Matriks ini menunjukkan bagaimana hasil prediksi model dibandingkan dengan data sebenarnya.Matriks berikut menggambarkan kemungkinan hasilnya:

Prediksi/Aktual	Positif (Aktual)	Negatif (Aktual)
Positif (Prediksi)	True Positive (TP)	False Positive (FP)
Negatif (Prediksi)	False Negative (FN)	True Negative (TN)

1. **True Positive (TP):** Model memprediksi positif, dan ternyata benar-benar positif. Contoh: Model memprediksi pasien sakit, dan pasien memang sakit.
2. **True Negative (TN):** Model memprediksi negatif, dan ternyata benar-benar negatif. Contoh: Model memprediksi pasien sehat, dan pasien memang sehat.
3. **False Positive (FP):** Model memprediksi positif, padahal sebenarnya negatif. Contoh: Model memprediksi pasien sakit, padahal sebenarnya sehat. (false alarm)
4. **False Negative (FN):** Model memprediksi negatif, padahal sebenarnya positif. Contoh: Model memprediksi pasien sehat, padaha *Hasil Prediksi* l sebenarnya sakit. (missed detection)

AKURASI

Accuracy (akurasi) adalah metrik paling umum yang digunakan untuk menilai kinerja model klasifikasi.

Rumusnya adalah:

$$Accuracy = \frac{TP + TN}{TP + TN + FP = FN}$$

Artinya, akurasi menghitung berapa proporsi prediksi yang benar dari seluruh prediksi yang dilakukan.

Contoh Sederhana

Misalkan dari 100 pasien, model memprediksi dengan hasil:

TP = 50

TN = 40

FP = 5

FN = 5

$$Accuracy = \frac{50 + 40}{50 + 40 + 5 + 5} = \frac{90}{100} = 0.9 = 90\%$$

Model ini memiliki akurasi 90%, yang berarti 9 dari 10 prediksi model benar.

Akurasi tidak selalu mencerminkan performa sebenarnya, terutama jika data tidak seimbang (imbalanced data). Misalnya, jika dari 100 data hanya 5 yang positif, model bisa mendapat akurasi 95% hanya dengan selalu menebak negatif. Namun model seperti itu tidak berguna untuk mendeteksi kasus positif yang penting.

- **PRESISI**

Precision (presisi) mengukur seberapa akurat model ketika memprediksi kelas positif.

Rumusnya adalah:

$$Precision = \frac{TP}{TP + FP}$$

Artinya, dari semua prediksi *positif* yang dibuat oleh model, berapa banyak yang benar-benar *positif*. Contohnya, jika model memprediksi 60 pasien sakit, dan 50 di antaranya benar-benar sakit, maka:

$$Precision = \frac{50}{60} = 0.83 = 83\%$$

Artinya, 83% dari prediksi positif model memang benar.

Makna Praktis

- Precision tinggi → sedikit *false positive*
- Precision penting ketika kesalahan prediksi positif berbahaya. Contoh: deteksi spam. Kita tidak ingin email penting ditandai sebagai spam (FP).

RECALL

Recall (atau disebut juga Sensitivity atau True Positive Rate) mengukur seberapa banyak kasus positif yang berhasil ditemukan oleh model.

$$Recall = \frac{TP}{TP + FN}$$

Artinya, dari semua data yang sebenarnya *positif*, berapa banyak yang berhasil diprediksi *positif* oleh model. Contohnya jika ada 100 pasien sakit, dan model hanya mendeteksi 80 di antaranya:

$$Recall = \frac{80}{100} = 0.8 = 80\%$$

Makna Praktis

- Recall tinggi → sedikit *false negative*
- Recall penting ketika kegagalan mendeteksi kasus positif berisiko tinggi, seperti deteksi penyakit atau sistem keamanan.

F1-SCORE

Terkadang, kita ingin menyeimbangkan antara precision dan recall, karena keduanya bisa saling bertolak belakang. Untuk itu digunakan F1-Score, yaitu rata-rata harmonis antara precision dan recall:

$$F1\_Score = 2\ x \frac{(Precision\ x\ Recall)}{(Precision + Recall)}$$

F1-Score bernilai antara 0 dan 1. Nilai mendekati 1 berarti model memiliki keseimbangan yang baik antara precision dan recall. Contohnya jika:

- Precision = 0.8
- Recall = 0.6

Maka:

$$F1 = 2\ x \frac{(0.8\ x\ 0.6)}{0.8 + \ 0.6} = 0.69$$

Artinya, model memiliki performa menengah, tidak terlalu presisi dan tidak terlalu sensitif.

Kesimpulan evaluasi model:

- Jika ingin *prediksi benar sebanyak mungkin*: gunakan Accuracy
- Jika ingin meminimalkan *false alarm*: gunakan Precision
- Jika ingin meminimalkan *kasus terlewat*: gunakan Recall
- Jika ingin keseimbangan: gunakan F1-Score

MEAN SQUARED ERROR (MSE)

Mean Squared Error (MSE) adalah ukuran yang menghitung rata-rata dari kuadrat selisih antara nilai sebenarnya (y_i) dan hasil prediksi model $\hat{y}_i$. Dengan kata lain, MSE memberi tahu kita seberapa besar kesalahan model secara rata-rata, namun karena dikalikan kuadrat, kesalahan besar akan dihukum lebih keras.

Rumus:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

dimana:

- n adalah jumlah data
- y_i adalah nilai sebenarnya
- $\hat{y}_i$ adalah nilai prediksi model

Contoh Sederhana misalkan kita ingin memprediksi tinggi badan seseorang berdasarkan umur, dan hasilnya sebagai berikut:

Tabel 1
Hasil Prediksi

Data	Nilai Sebenarnya (y_i)	Prediksi Model ($\hat{y}_i$)
1	160	165
2	170	175
3	180	178

Maka selisih kuadrat tiap data adalah:

$$(160 - 165)^2 = 25\ ;\ (170 - 175)^2 = 25\ ;\ (180 - 178)^2 = 4$$

Sehingga:

$$MSE = \frac{25 + 25 + 4}{3} = \frac{54}{3} = 18$$

Interpretasi

- MSE = 0 berarti model sempurna, tanpa kesalahan.
- Semakin besar MSE → semakin besar kesalahan rata-rata model.

ROOT MEAN SQUARED ERROR (RMSE)

RMSE adalah akar kuadrat dari MSE. Tujuannya sederhana: agar satuan hasilnya sama dengan data aslinya, sehingga lebih mudah dipahami.

Rumus:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Dari contoh sebelumnya, kita sudah menghitung MSE = 18.

Maka:

$$RMSE = \sqrt{18} \approx 4.24$$

Artinya, rata-rata kesalahan prediksi model sekitar 4.24 cm.

Interpretasi

- Semakin kecil nilai RMSE, semakin baik model dalam memprediksi.
- RMSE berguna jika kita ingin tahu seberapa jauh rata-rata prediksi menyimpang dari nilai sebenarnya dalam satuan asli data.

- **Mean Absolute Error (MAE)**

  Berbeda dari MSE yang mengkuadratkan selisih, MAE hanya mengambil nilai mutlak selisih antara prediksi dan nilai sebenarnya. MAE lebih "lembut" terhadap kesalahan besar karena tidak memberi hukuman berlebihan seperti MSE.

  Rumus:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

  Contohnya, jika menggunakan data sebelumnya:

  |160−165|=5 ; |170−175|=5 ; |180−178|=2

$$MAE = \frac{5+5+2}{3} = 4$$

  Artinya, rata-rata model salah sekitar 4 cm dari nilai sebenarnya.

  Interpretasi

  - MAE memiliki satuan yang sama dengan data asli.
  - Cocok digunakan ketika kita ingin mengetahui kesalahan rata-rata secara langsung tanpa menghukum kesalahan besar secara berlebihan.

  Perbandingan dengan MSE/RMSE

  - MSE/RMSE → sensitif terhadap kesalahan besar (outlier).
  - MAE → lebih stabil terhadap outlier. Jadi, pilihan metrik tergantung pada apakah kita ingin menghukum kesalahan besar atau tidak.

R^2 SCORE (KOEFISIEN DETERMINASI)

R^2 Score mengukur seberapa besar proporsi variasi data yang bisa dijelaskan oleh model. Secara intuitif, R^2 menjawab pertanyaan: "Seberapa baik model kita menjelaskan data yang ada?"

Nilainya berkisar dari 0 hingga 1, bahkan bisa negatif jika model sangat buruk.

Rumus

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

dimana :

- $\bar{y}$ adalah rata-rata dari semua nilai sebenarnya
- Bagian atas $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ adalah jumlah error model (disebut residual sum of squares)
- Bagian bawah $\sum_{i=1}^{n}(y_i - \bar{y})^2$ adalah total variasi data (disebut total sum odd squares). Jika model mampu menjelaskan semua variasi data (tanpa error), maka $R^2 = 1$. Sebaliknya, jika model tidak lebih baik dari sekadar menebak rata-rata, maka $R^2 = 0$.

Contoh Interpretasi

- $R^2 = 0.9$ → model menjelaskan 90% variasi data dengan baik.
- $R^2 = 0.5$ → model hanya menjelaskan 50% variasi data, masih banyak ruang untuk perbaikan.
- $R^2 < 0$ → model lebih buruk daripada tebakan rata-rata.

Tidak ada satu ukuran yang paling sempurna. Pemilihan metrik tergantung pada konteks dan tujuan:

- Gunakan MSE/RMSE jika Anda ingin menghukum kesalahan besar dengan keras.
- Gunakan MAE jika Anda ingin metrik yang lebih stabil terhadap *outlier*.

Gunakan R^2 Score untuk melihat *seberapa baik model menjelaskan data secara keseluruhan.*

"Generalisasi adalah kunci model ML yang sukses—bukan hanya unggul pada data latih, tetapi juga tangguh menghadapi data baru. Ini dicapai melalui pipeline yang solid: praproses data, rekayasa fitur, pemilihan model, penyetelan hiperparameter, dan evaluasi ketat. Tanpa generalisasi, model hanya menjadi penghafal data, tanpa pipeline yang terstruktur, pengembangan model menjadi tidak efisien dan sulit direproduksi".

3.7. Neural Network Dasar

Neural network dasar, sering disebut sebagai Multilayer Perceptron (MLP), adalah model sederhana yang terdiri dari lapisan input, satu atau lebih lapisan tersembunyi, dan lapisan output (Goodfellow, 2016). Berikut adalah penjelasan rinci tentang komponen dan cara kerjanya:

Struktur Neural Network

- Lapisan Input: Menerima data mentah (misalnya, piksel gambar atau fitur numerik). Setiap fitur direpresentasikan sebagai node.
- Lapisan Tersembunyi: Melakukan transformasi data melalui operasi linier (perkalian bobot dan penambahan bias) diikuti oleh fungsi aktivasi non-linier. Lapisan ini mengekstrak fitur yang semakin kompleks seiring bertambahnya kedalaman.
- Lapisan Output: Menghasilkan prediksi akhir, seperti skor probabilitas untuk klasifikasi atau nilai numerik untuk regresi.

Misalkan kita membangun neural network untuk memprediksi apakah seseorang akan membeli produk berdasarkan dua fitur: usia dan pendapatan. Inputnya adalah vektor ([usia, pendapatan]), yang melewati lapisan tersembunyi dengan 10 neuron (masing-masing dengan bobot dan bias), lalu ke lapisan output dengan fungsi Sigmoid untuk menghasilkan probabilitas (0 atau 1). Proses ini melibatkan perhitungan matriks dan fungsi aktivasi di setiap langkah.

Backpropagation & Optimisasi

Backpropagation (backward propagation of errors) adalah algoritma inti untuk melatih neural network. Ini memungkinkan model memperbarui bobot berdasarkan kesalahan prediksi dengan cara yang efisien.

Cara Kerja Backpropagation

1. Forward Pass: Hitung prediksi model dan fungsi kerugian berdasarkan data input dan label sebenarnya.
2. Backward Pass: Hitung gradien fungsi kerugian terhadap setiap bobot dan bias menggunakan aturan rantai (chain rule) dari kalkulus. Gradien ini menunjukkan seberapa besar setiap parameter berkontribusi pada kesalahan.
3. Perbarui Parameter: Gunakan gradien untuk memperbarui bobot dan bias dengan algoritma optimisasi seperti Gradient Descent

ALGORITMA OPTIMISASI

- Gradient Descent: Memperbarui bobot berdasarkan gradien dari seluruh dataset. Lambat untuk dataset besar.
- Stochastic Gradient Descent (SGD): Memperbarui bobot berdasarkan gradien dari satu sampel acak, lebih cepat tetapi berisik.
- Mini-Batch Gradient Descent: Kompromi antara SGD dan Gradient Descent, menggunakan subset kecil data untuk setiap pembaruan.

- Optimisasi Lanjutan: Algoritma seperti Adam, RMSprop, atau Adagrad menggabungkan momentum dan adaptasi learning rate untuk konvergensi yang lebih cepat dan stabil.

Tantangan

- Learning Rate: Jika terlalu besar, model mungkin tidak konvergen; jika terlalu kecil, pelatihan menjadi lambat.
- Local Minima dan Saddle Points: Fungsi kerugian yang kompleks dapat menyebabkan model terjebak di titik yang tidak optimal.
- Vanishing/Exploding Gradients: Dalam jaringan dalam, gradien dapat menjadi sangat kecil atau besar, menyulitkan pelatihan. Teknik seperti gradient clipping atau inisialisasi bobot yang baik (misalnya Xavier initialization) dapat membantu.

Dalam pelatihan model untuk klasifikasi gambar, backpropagation menghitung bagaimana setiap bobot di lapisan konvolusi memengaruhi kesalahan prediksi (misalnya, mengenali kucing vs. anjing). Bobot kemudian diperbarui untuk meningkatkan akurasi.

Overfitting, Regularisasi, Dropout

Overfitting terjadi ketika model belajar terlalu banyak dari data pelatihan, termasuk noise, sehingga gagal menggeneralisasi ke data baru. Berikut adalah teknik untuk mengatasi overfitting:

Overfitting

- Gejala: Performa model sangat baik pada data pelatihan tetapi buruk pada data uji.
- Penyebab: Model terlalu kompleks (terlalu banyak parameter), data pelatihan terlalu sedikit, atau data berisik.
- Deteksi: Pantau perbedaan antara metrik pelatihan (training loss) dan validasi (validation loss). Jika validation loss meningkat sementara training loss menurun, model kemungkinan overfitting.

Regularisasi

Regularisasi menambahkan penalti pada kompleksitas model untuk mencegah overfitting:

- L1 Regularization (Lasso): Menambahkan penalti berbasis norma L1 ke fungsi kerugian, mendorong bobot menjadi nol (sparsity).
- L2 Regularization (Ridge): Menambahkan penalti berbasis norma L2, mendorong bobot tetap kecil untuk model yang lebih sederhana.
- Elastic Net: Kombinasi L1 dan L2 regularization. Fungsi kerugian dengan regularisasi L2

Dropout

Dropout adalah teknik regularisasi khusus untuk neural networks:

- Selama pelatihan, sejumlah neuron secara acak "dimatikan" (outputnya disetel ke nol) dengan probabilitas tertentu (misalnya, 0.5).

- Ini mencegah model bergantung pada neuron tertentu, memaksa jaringan untuk belajar representasi yang lebih robust.
- Pada saat pengujian, semua neuron aktif, tetapi outputnya diskalakan untuk menyesuaikan efek dropout selama pelatihan.
- Aplikasi: Dropout sering digunakan di lapisan tersembunyi fully connected atau lapisan konvolusi dalam CNN.

Teknik Lain untuk Mengatasi Overfitting

- Data Augmentation: Meningkatkan jumlah data pelatihan dengan transformasi seperti rotasi, flipping, atau cropping pada gambar.
- Early Stopping: Menghentikan pelatihan ketika performa pada data validasi berhenti membaik.

Batch Normalization: Menormalkan output setiap lapisan untuk mempercepat pelatihan dan meningkatkan stabilitas, yang juga membantu mengurangi overfitting.

"Machine learning adalah cabang kecerdasan buatan yang memungkinkan sistem belajar dari data untuk mengenali pola dan membuat prediksi tanpa pemrograman eksplisit. Secara dasar, prosesnya melibatkan representasi data, pembentukan model matematis, serta penyesuaian parameter agar model mampu meminimalkan kesalahan dan meningkatkan akurasi prediksi".

BAGIAN 4
MATEMATIKA UNUK MACHINE LEARNING

4.1. Representasi Data

Jika Anda ingin membangun algoritma machine learning dari nol atau bahkan merancang algoritma baru, maka Anda membutuhkan pemahaman mendalam tentang beberapa bidang matematika penting. Tidak mungkin sepenuhnya menghindari matematika ketika berbicara tentang machine learning. Hal ini karena akar keilmuan machine learning sangat kuat di bidang matematika dan statistika. Keberhasilan machine learning tidak terlepas dari fondasi matematis dan statistik yang kuat. Melalui kombinasi keduanya, algoritma machine learning dapat:

- memahami struktur numerik dari data,
- mengukur hubungan antar variabel, dan
- membuat model yang mampu melakukan generalisasi terhadap data baru.

Dengan kata lain, matematika dan statistika adalah bahasa yang digunakan oleh mesin untuk memahami dunia. Untuk memahami bagaimana data diproses oleh algoritma machine learning, kita perlu mengenal bentuk representasi data dasar yang menjadi pondasi dari hampir semua operasi matematis dalam model, yaitu:

1. Skalar (Scalar)

Skalar merupakan satu nilai data tunggal. Contoh: angka 2, suhu 27°C, atau kecepatan 60 km/jam. Skalar biasanya digunakan untuk merepresentasikan nilai tunggal dari suatu pengukuran atau parameter.

2. Vektor (Vector)

Vektor adalah kumpulan nilai-nilai skalar yang tersusun dalam satu dimensi. Contoh: [2, 4, 6] dapat merepresentasikan posisi, arah, atau fitur dari satu sampel data. Dalam machine learning, vektor digunakan untuk menggambarkan fitur (feature vector) dari sebuah entitas, misalnya profil pelanggan atau representasi piksel dari gambar.

Misalkan (x,y) jika x adalah panah kekanan lalu y adalah panah keatas maka jika kita lihat titik vektor (2,1) artinya adalah panah sepanjang 2 ke kanan dan 1 ke atas

3. Matriks (Matrix)

Matriks adalah kumpulan vektor yang tersusun dalam dua dimensi (baris dan kolom). Contoh: sebuah matriks dapat menyimpan banyak data sampel sekaligus, di mana setiap baris mewakili satu data, dan setiap kolom mewakili fitur tertentu. Operasi pada matriks , seperti perkalian atau invers, merupakan inti dari banyak algoritma machine learning, termasuk neural networks dan dimensionality reduction.

Bayangkan kamu punya tabel angka, seperti lembar Excel kecil. Tabel ini punya baris (ke samping) dan kolom (ke bawah). Nah, tabel angka seperti itu disebut matriks. Contohnya:

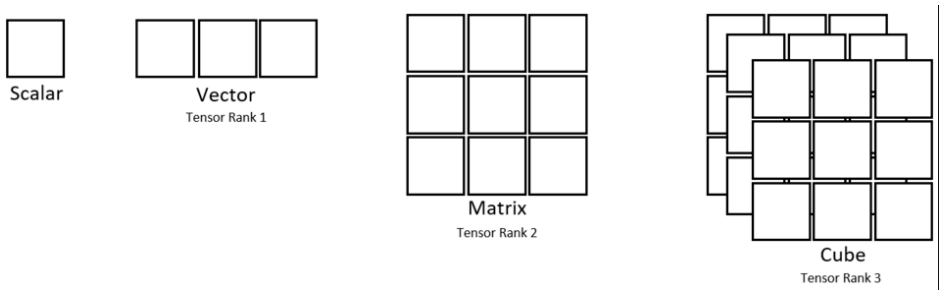
$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Matriks A diatas memiliki:

- 2 Baris (Karena ada 2 baris angka)
- 3 Kolom (Karena setiap baris punya 3 angka)

Maka kita bisa tuliskan bahwa matriks A berukuran 2 x 3

Dengan memahami bagaimana skalar, vektor, dan matriks digunakan untuk merepresentasikan data, Anda mulai melihat bagaimana algoritma machine learning bekerja di balik layar. Melalui operasi matematis terhadap representasi data ini, mesin dapat belajar mengenali pola, membuat prediksi, dan menghasilkan keputusan cerdas.



Gambar 14
Ilustrasi Scalar, Vector, Matriks, dan Tensor
(Sumber: Wikimedia)

Tensor

Dalam dunia matematika dan komputasi, tensor merupakan bentuk umum yang mencakup skalar, vektor, dan matriks sebagai kasus khusus dari dirinya. Konsep ini sangat penting dalam machine learning karena hampir semua data dan parameter model modern disimpan dalam bentuk tensor.

Untuk memahami tensor secara intuitif, kita dapat mulai dari bentuk data yang paling sederhana. Sebuah skalar adalah nilai tunggal, misalnya angka 7. Ia tidak memiliki arah maupun dimensi, hanya menyatakan satu besaran tertentu. Kemudian, jika kita menyusun beberapa skalar dalam satu baris, kita memperoleh sebuah vektor, misalnya [3, 4]. Vektor ini memiliki satu dimensi, dan

bisa kita bayangkan sebagai sebuah garis dengan dua titik data di dalamnya. Jika kita menyusun beberapa vektor menjadi baris dan kolom, terbentuklah matriks, misalnya:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Matriks ini bersifat dua dimensi, seperti sebuah tabel yang memiliki baris dan kolom. Setiap angka di dalamnya tetaplah skalar, tetapi kini tersusun dalam ruang dua arah. Ketika kita melangkah lebih jauh, misalnya memiliki sekumpulan matriks yang disusun bertumpuk ke arah ketiga maka terbentuklah tensor tiga dimensi. Tensor seperti ini bisa dibayangkan sebagai balok atau kubus data. Contohnya:

$$Tensor3D = \left[\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix}, \begin{bmatrix} 8 & 9 \\ 10 & 11 \end{bmatrix} \right]$$

Secara visual, tensor ini dapat dibayangkan sebagai tiga lapisan matriks yang bertumpuk — seperti tiga halaman tabel yang menumpuk ke atas. Jika setiap lapisan tersebut mewakili satu warna pada gambar (merah, hijau, dan biru), maka kita telah merepresentasikan gambar berwarna (RGB) dalam bentuk tensor 3D: tinggi × lebar × kanal warna.

Apabila kita memiliki banyak gambar sekaligus — misalnya 100 gambar RGB dengan ukuran 64 × 64 piksel — maka seluruh kumpulan itu menjadi tensor 4D dengan ukuran: (100 × 64 × 64 × 3) artinya, 100 gambar, masing-masing memiliki tinggi 64, lebar 64, dan 3 lapisan warna.

Dengan cara ini, tensor dapat menampung data dalam berbagai bentuk dan dimensi. Skalar adalah tensor berdimensi nol, vektor berdimensi satu, matriks berdimensi dua, dan tensor-tensor selanjutnya mewakili dimensi tiga ke atas.

Dalam konteks machine learning, tensor adalah bentuk utama dari semua data yang digunakan oleh model. Pada jaringan Convolutional Neural Network (CNN) untuk pengenalan citra, tensor menyimpan data gambar dan hasil transformasi pada setiap lapisan konvolusi. Pada Recurrent Neural Network (RNN) atau Transformer di bidang pemrosesan bahasa alami, tensor digunakan untuk menyimpan urutan kata dalam kalimat, di mana setiap kata direpresentasikan sebagai vektor numerik. Bahkan dalam reinforcement learning, tensor menyimpan kondisi lingkungan, aksi, dan hasil observasi yang diterima oleh agen selama proses pelatihan.

Tensor memungkinkan model untuk melakukan operasi matematis kompleks seperti perkalian, penjumlahan, dan transformasi dalam ruang multidimensi. Setiap operasi di dalam jaringan syaraf tiruan (neural network) pada dasarnya bekerja di atas tensor ini. Karena itu, tensor dapat dianggap sebagai "bahasa universal data" dalam deep learning. Secara intuitif, tensor adalah wadah multidimensi yang mampu menyimpan struktur data apapun:

angka tunggal, deretan nilai, tabel dua dimensi, hingga kumpulan gambar atau video. Ia memperluas konsep ruang dari titik (skalar), garis (vektor), bidang (matriks), hingga volume (tensor). Dengan representasi ini, komputer mampu mempelajari pola-pola kompleks dalam data dari gambar dan suara hingga teks dan sinyal waktu secara matematis dan efisien.

"Representasi data adalah cara kita menerjemahkan dunia nyata ke dalam bentuk yang dapat dipahami oleh mesin. Dari skalar hingga tensor, setiap bentuk membawa kita selangkah lebih dekat untuk membuat komputer melihat, mendengar, dan memahami sebagaimana manusia berpikir".

4.2. Operasi Matriks

Setelah memahami cara merepresentasikan data dalam bentuk skalar, vektor, matriks, dan tensor, kini saatnya kita melangkah ke fondasi penting berikutnya: operasi matriks. Nyatanya, hampir semua komputasi dalam pembelajaran mesin—mulai dari model linear sederhana hingga jaringan saraf yang rumit—berakar dari manipulasi dan transformasi matriks.

Matriks bukan sekadar susunan angka dalam baris dan kolom. Ia adalah alat matematika yang ampuh untuk merepresentasikan hubungan antar variabel, melakukan transformasi ruang, dan menyederhanakan perhitungan yang melibatkan banyak data sekaligus. Dalam komputasi, operasi matriks memungkinkan proses perhitungan berjalan efisien dan paralel—hal yang sangat krusial ketika kita bekerja dengan data dalam skala besar.

Pada bab ini, kita akan mengeksplorasi operasi dasar matriks seperti penjumlahan, perkalian, transpos, dan invers, serta melihat bagaimana konsep-konsep ini menjadi dasar bagi perhitungan yang lebih kompleks di tahap selanjutnya. Memahami operasi matriks bukan hanya tentang cara menghitung, tetapi juga tentang cara berpikir dalam ruang vektor—bagaimana data dan parameter model saling berinteraksi secara matematis.

Dengan pemahaman ini, kita akan menyadari bahwa setiap langkah dalam algoritma pembelajaran mesin—mulai dari *forward propagation* hingga pembaruan bobot melalui gradien—pada dasarnya adalah rangkaian operasi matriks yang diulang dengan pola tertentu. Itulah mengapa menguasai operasi matriks bukanlah

sekadar pelengkap, melainkan syarat mutlak untuk benar-benar mengerti apa yang terjadi di balik layar *machine learning* modern.

1. Perkalian Skalar Dengan Matriks

Ketika kita mengalikan matriks dengan skalar, itu berarti kita mengalikan setiap angka di dalam matriks dengan angka skalar tersebut. Tidak ada langkah rumit — semuanya dikalikan satu per satu.

Misalkan kita memiliki matriks A dan skalar k:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \qquad k = 3$$

Kita ingin menghitung:

$$C = k\ x\ A$$

Artinya: Setiap elemen di dalam matriks A dikalikan dengan 3

Jadi proses perhitungan adalah sebagai berikut:

$$C = 3\ x\ \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 3x1 & 3x2 \\ 3x3 & 3x4 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 9 & 12 \end{bmatrix}$$

2. Perkalian Matriks Dengan Matriks

Dalam aljabar linear, dua operasi yang paling penting setelah penjumlahan dan pengurangan adalah perkalian matriks (matrix multiplication) dan transposisi (matrix transpose)**.** Keduanya menjadi inti dalam formulasi matematis algoritma *machine learning*, termasuk dalam perhitungan prediksi, pembaruan bobot, serta dalam operasi jaringan syaraf tiruan (*neural network*).

Perkalian matriks tidak dilakukan secara elemen-per-elemen, melainkan melibatkan perkalian baris dari matriks pertama

dengan kolom dari matriks kedua. Syarat agar dua matriks dapat dikalikan adalah jumlah kolom pada matriks pertama harus sama dengan jumlah baris pada matriks kedua.

Sekarang bayangkan kita memiliki 2 tabel angka yaitu A dan B. Agar A dan B bisa dikalikan maka jumlah kolom matriks pertama (A) harus sama dengan jumlah baris matriks kedua (B). Jika syarat ini terpenuhi maka barulah kita bisa melakukan perkalian.

Kemudian hasil dari perkalian dua matriks diatas bisa disebut matriks C. Jika ukutan A adalah m x n dan ukuran B adalah n x p maka ukuran matriks C hasil perkalian adalah m x p.

Artinya jumlah baris hasilnya sama seperti matriks A dan jumlah kolom hasilnya sama seperti matriks B. Agar lebih jelas mari kita jabarkan tahap per tahap perkalian matriks:

a. Pilih satu baris dari matriks A
b. Pilih satu kolom dari matriks B
c. Kalikan angka-angka yang sejajar
d. Jumlahkan smua hasil perkaliannya

Tentunya akan lebih jelas jika kita pelajari sebuah contoh. Misalkan matriks A dan B adalah sebagai berikut:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

Kita mau cari hasilnya $C = A\ x\ B$

a) Langkah 1 : Ambil baris pertama $A \rightarrow [1,2]$ dan ambil kolom pertama dari $B \rightarrow [5,7]$.
   Kalikan yang berhadapan dan jumlahkan:

$$1x5 + 2x7 = 5 + 14 = 19$$

Ini digunakan sebagai angka pertama di hasil (baris 1, kolom 1)

b) Langkah 2: Ambil baris pertama dari $A \rightarrow [1,2]$ dan ambil kolom kedua $B \rightarrow [6,8]$

Kalikan dan jumlahkan:

$$1\ x\ 6 + 2\ x\ 8 = 6 + 16 = 22$$

Ini digunakan sebagai nilai di baris 1 kolom 2

c) Langkah 3 : Ambil baris kedua $A \rightarrow [3,4]$ dan kolom pertama $B \rightarrow [5,7]$

Kalikan dan jumlahkan:

$$3x5 + 4\ x\ 7 = 15 + 28 = 43$$

Ini digunakan sebagai nilai di baris 2 kolom 1

d) Langkah 4: Ambil baris kedua $A \rightarrow [3,4]$ dan kolom kedua $B \rightarrow [6,8]$

Kalikan dan jumlahkan:

$$3x6 + 4x8 = 18 + 32 = 50$$

Maka matriks C akan berbentuk sebagai berikut:

$$C = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

A. Transposisi Matriks

Transposisi memiliki arti menukar posisi. Dalam konteks matriks, transposisi berarti kita menukar baris menjadi kolom dan kolom menjadi baris. Dengan kata lain kita "putar" matriksnya dimana yang tadinya baris menjadi kolom dan yang tadinya kolom menjadi baris.

Misalkan kita punya matriks A, yang berbentuk sebagai berikut:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

Maka transposenya (ditulis sebagai A^T) adalah:

$$A^T = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{12} & a_{23} \end{bmatrix}$$

B. Invers Matriks

Inverse secara bahasa artinya kebalikan, jadi inverse matriks adalah matriks kebalikan dari suatu matriks. Jika dalam aritmatika biasa kita kenal kebalikan dari 2 adalah $\frac{1}{2}$ karena $2\ x \frac{1}{2} = 1$. Maka pada matriks juga sama. Jika matriks A memiliki inverse A^{-1} maka saat dikalikan hasilnya adalah identitas (I)

$$A\ x\ A^{-1} = I$$

Sebagai reminder matriks identitas adalah matriks yang isinya:

- angka 1 di diagonal utama
- dan 0 di tempat lainnya.

Contoh:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Kalau dikalikan dengan matriks apa pun hasilnya sama seperti semula, misalnya $A\ x\ I = A$

Tidak semua matriks bisa dicari inversenya. Matriks hanya punya inverse jika:

- Matriksnya persegi (jumlah baris = jumlah kolom)
- Determinannya tidak nol

Kalau determinannya = 0 → tidak punya inverse

C. Matriks Kovarian

Kovarian digunakan untuk mengukur dua hal bergerak searah atau tidak. Jika kita hanya punya dua hal (tinggi dan berat) maka kita bisa membuat satu matriks kecil 2x2 yang berisi kovarian antar pasangan. Contohnya

$$Covariance\ Matrix = \begin{bmatrix} Cov(tinggi, tinggi) & Cov(tinggi, berat) \\ Cov(berat, tinggi) & Cov(berat, berat) \end{bmatrix}$$

Matriks diatas dinamakan sebagai matriks kovarian. Matriks kovarian seperti peta hubungan antar semua variabel. Pada diagonal menyatakan seberapa besar variasi tiap variabel. Diluar diagonal menyatakan seberapa kuat dua variabel berhubungan.

Matriks kovarian digunakan untuk PCA, karena PCA perlu mengetahui arah mana dalam data yang paling banyak memiliki variasi (perbedaan). Untuk mengetahui ara itu, kita perlu mengetahui:

- Mana fitur yang bervariasi besar
- Mana fitur yang saling berhubungan (naik-turun) bersama

Kedua poin diatas bisa kita dapatkan dari matriks kovarian. PCA mengambil matriks kovarian lalu mencari eigenvector dan eigenvalue.

D. Determinant Matriks

Determinan adalah satu nilai angka yang mewakili seluruh isi sebuah matriks persegi. Sebagai reminder matriks persegi artinya jumlah barisnya sama dengan jumlah kolomnya (misalnya 2x2, 3x3, 4x4, dan seterusnya)

Determinan bisa dianggap sebagai ukuran atau besaran dari matriks tersebut dan sering digunakan untuk:

- mengetahui apakah suatu matriks bisa dibalik (inverse)
- menyelesaikan sistem persamaan linear
- banyak digunakan dalam algoritma machine learning atau grafik komputer

Untuk matriks $2x2,$ cara menghitungnya sangat mudah:

$$\det(A) = ad - bc$$

Artinya:

- Kalikan elemen diagonal utama (axd)
- Kalikan elemen diagonal lainnya (bxc)
- Lalu kurangkan hasilnya.

"Operasi matriks adalah bahasa sejati dari pembelajaran mesin—tempat di mana angka-angka berinteraksi membentuk pola dan makna. Melalui operasi inilah data tidak hanya dihitung, tetapi diubah menjadi pengetahuan yang dapat dipelajari oleh mesin".

4.3. Aljabar Linear

Aljabar linear adalah bahasa matematika inti yang mendasari hampir semua metode dalam *machine learning* modern. Di balik setiap model—mulai dari regresi sederhana hingga jaringan saraf yang kompleks—tersimpan operasi dan konsep yang berakar pada prinsip dasar aljabar linear. Melalui aljabar linear, data dapat diubah, diputar, atau disederhanakan tanpa kehilangan inti informasinya.

Dalam bab ini, kita akan menelusuri bagaimana konsep seperti *eigenvalue* dan *eigenvector* membantu kita memahami arah variasi terpenting dalam data, serta mengapa keduanya menjadi kunci dalam teknik *Principal Component Analysis* (PCA)—sebuah metode andalan untuk reduksi dimensi dan ekstraksi fitur. Selain itu, kita juga akan membahas bagaimana proses vektorisasi membuat komputasi menjadi jauh lebih efisien, dengan mengubah operasi berulang menjadi bentuk matematika yang dapat dijalankan secara paralel.

Dengan memahami aljabar linear, kita tidak hanya sekadar mempelajari cara memanipulasi matriks, tetapi juga belajar melihat struktur tersembunyi di dalam data. Bab ini akan membuka perspektif baru: bahwa setiap transformasi, proses pembelajaran, dan pola yang muncul dalam *machine learning*, pada hakikatnya bersumber dari keteraturan matematika yang elegan—sebuah keteraturan yang diungkap oleh aljabar linear.

1. EIGENVALUE DAN EIGENVECTOR

Eigenvalue dan eigenvector bisa diartikan sebagai nilai dan arah khas dari sebuah matriks yang menunjukkan bagaimana matriks itu mengubah suatu vektor".

Untuk memahami definisi diatas mari kita kaji sebuah contoh. Misalkan kita memiliki vektor sebagai berikut:

$$v = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Misalkan ini kita artikan sebagai panah 1 langkah ke kanan. Berikutnya kita punya matriks.

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

Maka perkalian vektor v dengan matriks A adalah sebagai berikut:

$$Av = \begin{bmatrix} 2x1 + 0x0 \\ 0x1 + 1x0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

Disini arah (misalkan ke kanan) tidak berubah arah hanya panjangnya yang berubah. Arah inilah yang dinamakan sebagai eigenvector. Seberapa besar nilai perpanjangannya (misalkan pada contoh diatas adalah 2) adalah eigenvalue.

Secara sederhana:

- eigenvector adalah arah tetap yang tidak berubah saat dikalikan oleh matriks, hanya memanjang atau memendek
- eigenvalue adalah angka pengali yang menunjukkan seberapa besar perubahan panjang tersebut

2. PRINCIPAL COMPONENT ANALYSIS

Ide utama dari PCA (Principal Component Analysis) adalah mencari arah utama tempat data paling banyak menyebar. Sebagai gambaran sebaran data points memang tidak merata

namun data points cenderung berderet miring ke satu arah tertentu (misalnya diagonal). Jika kita bisa menggambar satu garis yang mewakili arah utama penyebaran data itu maka garis itu bisa menunjukkan pola terpenting dari data.

Secara matematis, arah itu disebut komponen utama (principal component) dan arah tersebut adalah eigenvector dari matriks kovarian data. Sedangkan seberapa besar sebaran data disepanjang arah itu dinyatakan oleh eigenvalue-nya.

Jadi di PCA:

- Eigenvector → arah pola utama data
- Eigenvalue → seberapa besar variasi (informasi) di arah itu

Langkah-langkah PCA sebenarnya hanya menerjemahkan ide diatas ke bentuk matematis:

- ✓ **Langkah 1 –** Ambil data
  Menentukan fitur dan menempatkan titik data
- ✓ **Langkah 2 –** Pusatkan data
  Kurangi setiap nilai dengan rata-ratanya agar pusatnya di nol, tujuannya adalah agar tidak bisa
- ✓ **Langkah 3 -** Buat matriks kovarian
  Matriks kovarian ini menceritakan hubungan antar fitur apakah tinggi dan berat naik bersama atau tidak
- ✓ **Langkah 4 –** Cari eigenvalue dan eigenvector dari matriks kovarian
  Eigenvector → arah baru data
  Eigenvalue → Besarnya variasi di arah itu

- ✓ **Langkah 5 –** Pilih arah dengan eigenvalue terbesar
  Karena arah dengan data paling tersebar berarti memuat informasi paling banyak
- ✓ **Langkah 6 –** Proyeksikan data ke arah itu
  Kita melihat data dari arah yang paling penting saja, dengan melakukan hal ini maka kita bisa mengurangi dimensi namun tetap mempertahankan informasi utama.

3. VEKTORISASI

Dalam dunia komputasi modern, terutama di bidang machine learning dan data science, kita sering menjumpai istilah vektorisasi. Secara sederhana, vektorisasi adalah cara mengubah data atau perhitungan menjadi bentuk vektor (atau matriks) agar komputer dapat memprosesnya secara paralel dan efisien. Daripada melakukan perhitungan satu per satu, vektorisasi memungkinkan semua operasi dilakukan sekaligus dalam satu langkah komputasi. Hal ini dimungkinkan karena komputer (terutama CPU dan GPU) dirancang untuk memproses data dalam blok besar menggunakan operasi linier yang teroptimasi.

Sebagai contoh, misalkan kita memiliki empat angka: 2,4,6, dan 8, maka menggunakan vektorisasi bentuknya adalah sebagai berikut:

$$a = [2,4,6,8]$$

Sekarang, kita ingin mengalikan semua angka tersebut dengan 2. Tanpa vektorisasi, kita mungkin akan menulis operasi ini satu per satu:

$$2x2, 2x4, 2x6, 2x8$$

Namun dengan vektorisasi, kita cukup menuliskan:

$$2\ x\ a = [4,8,12,16]$$

Seluruh operasi dilakukan sekali jalan, tanpa perlu iterasi eksplisit.

Secara konseptual, vektorisasi bukan hanya soal menulis kode yang lebih singkat, tetapi juga mencerminkan cara berpikir matematis dalam bentuk linier. Dengan vektorisasi:

- Data diubah menjadi struktur matematis (vektor atau matriks)
- Operasi dijalankan dalam bentuk aljabar linier
- Perhitungan menjadi lebih cepat karena memanfaatkan arsitektur paralel komputer

Inilah sebabnya vektorisasi menjadi fondasi penting dalam banyak bidang, mulai dari perhitungan statistik, pelatihan jaringan saraf tiruan, hingga pemrosesan gambar dan sinyal.

"Aljabar linear bukan sekadar kumpulan rumus, tetapi bahasa universal yang memungkinkan kita memahami dan mengendalikan data, ruang, serta transformasi di dalamnya. Dari eigenvector hingga vektorisasi, setiap konsepnya membuka jalan bagi kecerdasan buatan untuk melihat, berpikir, dan belajar seperti manusia".

4.4. Probabilitas dan Statistika

Setiap keputusan dalam dunia sains, teknologi, maupun kehidupan sehari-hari pada dasarnya lahir dari ketidakpastian. Probabilitas dan statistika hadir sebagai bahasa formal untuk memahami dan mengelola ketidakpastian tersebut. Melalui probabilitas, kita belajar memperkirakan kemungkinan terjadinya suatu peristiwa; sementara statistika mengajarkan cara mengambil kesimpulan yang bermakna dari data yang terbatas (Ross, 2014).

Bab ini akan memandu Anda memahami fondasi berpikir ilmiah berbasis data. Kita akan mulai dari konsep dasar probabilitas, kemudian beralih ke probabilitas bersyarat, independensi, dan hukum probabilitas total, sebelum akhirnya sampai pada mahkota teori inferensi: Teorema Bayes. Untuk memperdalam pemahaman tentang cara kerja model probabilistik, kita juga akan mengenal parameter model dan likelihood—dua konsep kunci dalam pembelajaran mesin modern.

Selanjutnya, kita akan memasuki dunia statistika dan distribusi probabilitas, yang menggambarkan bagaimana kemungkinan tersebar di antara berbagai nilai acak. Mulai dari distribusi Bernoulli yang sederhana hingga distribusi binomial yang lebih kompleks, Anda akan melihat bagaimana fenomena acak dapat dimodelkan dengan sistematis dan elegan.

Dengan menguasai bab ini, Anda tidak hanya sekadar mempelajari angka dan rumus, tetapi juga mengembangkan cara berpikir ilmiah—sebuah kemampuan untuk mengubah ketidakpastian menjadi pengetahuan yang dapat diandalkan.

1. Probabilitas

Probabilitas adalah cara kita mengukur seberapa mungkin suatu peristiwa akan terjadi. Ia bukan sekadar angka, tetapi cerminan dari keyakinan atau frekuensi suatu kejadian dalam dunia yang penuh ketidakpastian. Secara matematis, probabilitas suatu peristiwa E dinyatakan sebagai $P(E)$ dan nilainya selalu berada diantara 0 dan 1:

$$0 \leq P(E) \leq 1$$

Jika $P(E) = 0,$ artinya peristiwa tersebut tidak mungkin terjadi sama sekali. Sedangkan jika $P(E) = 1$ berarti peristiwa itu pasti akan terjadi. Sementara nilai-nilai di antara keduanya, seperti 0.25, 0.5, atau 0.75, menggambarkan tingkat kemungkinan peristiwa tersebut muncul.

Sebagai contoh sederhana, bayangkan Anda melempar sebuah koin. Ada dua kemungkinan hasil: angka atau gambar. Karena keduanya sama-sama mungkin, maka peluang munculnya angka adalah 0.5, begitu pula dengan gambar. Nilai 0.5 ini tidak berarti bahwa setengah koin akan menjadi angka dan setengahnya lagi gambar, melainkan bahwa jika percobaan dilakukan berulang kali dalam jumlah besar, maka sekitar setengah dari seluruh lemparan akan menghasilkan angka.

Dengan cara pandang ini, probabilitas menjadi jembatan antara teori dan kenyataan: ia membantu kita membuat perkiraan berdasarkan pola yang muncul dari banyak percobaan. Di sinilah kekuatan probabilitas ia tidak menjanjikan kepastian, tetapi memberikan kerangka rasional untuk memahami ketidakpastian.

2. Probabilitas Bersyarat/Conditional Probability

Dalam kehidupan sehari-hari, banyak keputusan kita bergantung pada informasi tambahan yang kita miliki. Misalnya, peluang seseorang kehujanan akan berbeda jika kita tahu bahwa langit sedang mendung. Inilah inti dari probabilitas bersyarat, yaitu cara untuk memperbarui penilaian terhadap suatu peristiwa ketika kita sudah mengetahui bahwa peristiwa lain telah terjadi. Secara formal, probabilitas bersyarat menyatakan peluang terjadinya peristiwa A dengan syarat bahwa peristiwa B sudah terjadi. Notasinya ditulis sebagai:

$$P(A|B)$$

dibaca "probabilitas A dengan syarat B". Artinya, kita tidak lagi melihat A dalam kondisi umum, tetapi dalam konteks baru: dengan asumsi B telah terjadi lebih dulu. Misalnya, jika A adalah "seseorang basah kuyup" dan B adalah "langit mendung," maka $P(A|B)$ menunjukkan seberapa besar kemungkinan seseorang basah kuyup jika kita tahu langit memang mendung.

Secara matematis, probabilitas bersyarat dirumuskan sebagai:

Rumus probabilitas bersyarat:

$$P(A|B) = \frac{P(A \text{ and } B)}{P(B)}$$

dengan syarat $P(B) > 0$

Artinya:

- Pembilang : $P(A\ and\ B)$ menggambarkan peluang kedua peristiwa terjadi bersamaan, yakni A dan B terjadi dalam waktu yang sama
- Penyebut : $P(B)$ adalah peluang B terjadi, karena B adalah kondisi yang sudah kita ketahui kebenarannya.

Dengan membagi keduanya, kita sebenarnya sedang menghitung "seberapa sering A muncul dalam kumpulan kejadian di mana B benar-benar terjadi." Konsep ini menjadi sangat penting karena menggambarkan bagaimana pengetahuan baru mengubah pandangan kita terhadap peluang. Dalam dunia nyata, hampir semua keputusan — dari diagnosis medis, prediksi cuaca, hingga sistem kecerdasan buatan — menggunakan prinsip probabilitas bersyarat untuk menyesuaikan keyakinan berdasarkan informasi yang tersedia.

3. Parameter Model

Bayangkan kamu sedang mencoba menebak bentuk hubungan antara dua hal, misalnya antara jumlah jam belajar dan nilai ujian. Kita tahu bahwa semakin lama seseorang belajar, biasanya nilainya naik. Tapi seberapa besar kenaikannya. Apakah setiap jam belajar menambah nilai sebesar 2 poin atau 5 poin. Nah, angka-angka seperti 2 atau 5 itulah yang disebut parameter model.

Misal kita punya timbangan digital. Timbangan ini punya pengaturan sensitifitas yang jika kita ubah sedikit hasil timbangannya bisa berubah. Model probabilistik juga begitu, ia

punya pengaturan internal yaitu parameter yang menentukan bagaimana model itu memandang dunia. Jadi kalau parameter diatur ke satu nilai model bisa memprediksi sesuatu dengan cara tertentu dan kalau parameternya diatur ke nilai lain maka hasil prediksinya bisa berbeda.

Misalkan model kita:

$$y = a \cdot x + b$$

dimana:

- a dan b adalah parameter model
- x adalah data input (misal jumlah jam belajar)
- y adalah hasil yang diprediksi (misal nilai ujian)

Kalau kita sudah mengetahui a dan b, kita bisa menghitung kemungkinan nilai ujian seseorang yang belajar 3 jam. Tetapi kalau kita hanya memiliki data jam belajar dan nilai ujiannya, kita bisa mencari nilai a dan b yang paling cocok agar model mendekati kenyataan.

Hubungan dengan probability dan likelihood adalah dalam sudut pandang view, kita sudah tahu parameter (misal $a = 5$ dan $b = 20$), lalu menghitung peluang data muncul. Contohnya : "Kalau model mengatakan tiap jam belajar menambah 5 poin, seberapa besar kemungkinan siswa mendapat nilai 85". Berikutnya dalam likelihood view, kita sudah mengetahui datanya tetapi belum mengetahui parameternya. Contoh : "Dari data jam belajar dan nilai ujian siswa, nilai a dan b berapa yang paling masuk akal".

4. Likelihood

Setiap kali berhadapan dengan data dan model probabilistik, kita sebenarnya sedang berinteraksi dengan dua sudut pandang berbeda:

- Probability view → mencari peluang suatu kejadian terjadi jika parameter model sudah diketahui
- Likelihood view → mencari parameter model yang paling sesuai dengan kejadian yang sudah diamati

Dua-duanya menggunakan bentuk yang sama, yaitu conditional probability tetapi arah pandangnya berbeda.

Dalam pandangan probabilistik, kita sudah mengetahui parameternya modelnya dan ingin mengetahui seberapa mungkin kejadian muncul.

Contoh:

Sebuah koin diketahui seimbang, berarti peluang munculnya kepala = 0.5. Kita ingin mengetahui berapa peluang muncul 7 kepala dari 10 lemparan.

$$\mathrm{P}(\mathrm{X} = 7|\theta = 0.5)$$

Kita menilai kemungkinan terjadinya data (hasil lemparan) berdasarkan parameter yang sudah diketahui$(\theta = 0.5)$.

Arah pikirnya : "Untuk θ tertentu, puncak-puncak mana (x) yang paling tinggi". Secara visual kita menyisir sumbu-x (nilai data) sementara θ tetap.

Sebaliknya dalam sudut pandang likelihood, kita sudah mengetahui data hasil pengamatan dan ingin mengetahui parameter mana yang paling mungkin menghasilkan data itu.

$$L(\theta) = P(X = x_{observed}|\theta)$$

Contoh :

Kita telah melakukan 10 kali lemparan dan memperoleh 7 kepala.

Maka pertanyaannya adalah berapa nilai θ (peluang kepala) yang paling cocok menjelaskan hasil ini. Jawabannya kita ingin mencari dengan memvariasiakan θ dan menghitung likelihoodnya. Arah pikirnya adalah untuk data tertentu, puncak mana θ yang paling tinggi. Secara visual kita menyisir sumbu θ (nilai parameter) sementara x tetap.

Secara matematis baik probabilitas dan likelihood sering ditulis dengan rumus yang identik karena sama-sama menggunakan:

$$P(X = x|\theta)$$

Namun maknanya berbeda:

- Pada probability θ (parameter) sudah diketahui, dan mencari x (data), fungsi dari x
- Pada likelihood, x (data) sudah diketahui, θ (parameter) dicari, fungsi dari θ

Jadi meskipun secara numerik nilainya bisa sama, arah interpretasi dan tujuannya berbeda. Walaupun demikian keliru menafsirkan Likelihood(θ) = 0.4 berarti θ=0.4. Karena likelihood adalah nilai fungsi bukan nilai parameternya. Kita harus struktur fungsi ini:

$$L(\theta) = P(X = x_{observed}|\theta)$$

- θ adalah input fungsi (parameter yang diuji)
- $L(\theta)$ adalah output fungsi (tingkat kecocokan antara model dan data)

Jadi jika $L(\theta) = 0.4,$ artinya jika model menggunakan parameter θ tertentu, peluang menghasilkan data ini adalah 0.4 dan bukan berarti $\theta = 0.4$

5. Independensi dan Ketergantungan dalam Probabilitas

Dalam probabilitas, dua peristiwa dikatakan independen (tidak saling bergantung) apabila terjadinya salah satu tidak memengaruhi kemungkinan terjadinya yang lain. Sebaliknya, dua peristiwa disebut dependen (bergantung) apabila terjadinya satu peristiwa mempengaruhi peluang terjadinya peristiwa lain.

Dua peristiwa A dan B disebut independn jika dan hanya jika berlaku:

$$P(A \cap B) = P(A)x\ P(B)$$

Artinya probabilitas bahwa keduanya terjadi bersamaan sama dengan hasil kali probabilitas masing-masing. Contoh ketika melempar dua dadu:

- Peluang mendapatkan angka 6 dari dadu pertama P(A) = $\frac{1}{6}$
- Peluang mendapatkan angka 6 dari dadu kedua P(B) = $\frac{1}{6}$

Maka peluang mendapatkan dua angka 6 sekaligus dari dadu pertama dan dadu kedua adalah:

$$P(A \cap B) = \frac{1}{6}\ x \frac{1}{6} = \frac{1}{36}$$

Karena hasil kali ini benar-benar sama dengan peluang sebenarnya, maka A dan B independen.

Berikutnya dua peristiwa dikatakan dependn jika:

$$P(A \cap B) \neq P(A)\ x\ P(B)$$

atau dengan kata lain:

$$P(A|B) \neq P(A)$$

Artinya peluang A beruba ketika kita tahu bahwa B telah terjadi.

Misalkan dari 52 kartu, kita mengambil satu kartu tanpa mengembalikan ke deck. Sehingga kondisinya adalah sebagai berikut:

- A = kartu pertama adalah As → P(A) = $\frac{4}{52}$
- B = kartu kedua adalah As → P(B | A) = $\frac{3}{51}$ (karena satu As sudah diambil)

Karena $P(B|A) \neq P(B)$, maka kedua peristiwa tidak independen

6. Hukum Probabilitas Total

Konsep ini digunakan ketika sebuah peristiwa A dapat terjadi melalui beberapa cara berbeda yang masing-masing bergantung pada kondisi atau peristiwa lain $B_1, B_2, \ldots, B_n$ yang saling ekslusif (tidak bisa terjadi bersamaan) dan lengkap (menutupi semua kemungkinkan). Rumusnya adalah:

$$P(A) = \sum_{i=1}^{n} P(A|B_i)x\ P(B_i)$$

Artinya probabilitas suatu peristiwa A adalah jumlah dari semua kemungkinan cara terjadinya A dikalikan dengan peluang dari kondisi masing-masing.

Contoh misalkan sebuah test medis digunakan untuk mendeteksi penyakit langka.

- B_1 : orang sakit (probabilitas $P(B_1) = 0.01$)
- B_2 : orang sehat (probabilitas $P(B_2) = 0.99$)
- Tes positif untuk orang sakit $P(A|B_1) = 0.9$
- Tes positif untuk orang sehat (false positive) $P(A|B_2) = 0.05$

Maka probabilitas total seseorang mendapat hasil test positif:

$$P(A) = P(A|B_1)P(B_1) + P(A|B_2)P(B_2)$$

$$P(A) = (0.9)(0.01) + (0.05)(0.99) = 0.009 + 0.0495 = 0.0585$$

Jadi hanya sekitar 5.85% dari semua orang akan mendapat hasil tes positif.

7. Teorema Bayes

Dalam kehidupan sehari-hari, peluang atau probabilitas sering kali dipahami sebagai sesuatu yang bersifat tetap dan universal. Namun dalam kenyataannya, peluang sangat bergantung pada konteks — pada kondisi dan informasi yang kita miliki saat melakukan perhitungan. Misalnya, ketika kita melempar koin yang adil, peluang munculnya gambar (heads) adalah 50%. Nilai ini disebut sebagai peluang a priori (a priori probability) — yaitu peluang umum suatu kejadian tanpa mempertimbangkan kondisi khusus yang mungkin memengaruhinya. Namun, dalam banyak situasi di dunia nyata, konteks dapat mengubah nilai peluang tersebut. Ketika kondisi berubah, maka peluang pun ikut berubah. Dalam kasus seperti ini, kita menggunakan istilah peluang a posteriori (a posteriori probability) — yaitu peluang setelah adanya informasi baru yang memengaruhi keyakinan kita terhadap suatu kejadian.

Untuk memahami perbedaan priori dengan posteriori, bayangkan dua contoh berikut:

1. Contoh populasi umum:
   Secara umum, peluang seseorang berjenis kelamin perempuan adalah sekitar 50%. Ini adalah peluang a priori.
2. Contoh dengan kondisi tambahan:
   Namun jika kita hanya mempertimbangkan kelompok usia lanjut, peluang seseorang menjadi perempuan meningkat karena perempuan memiliki harapan hidup lebih lama. Dengan kata lain, peluang a posteriori pada konteks ini menjadi lebih tinggi dari 50%.

Perubahan ini menunjukkan bahwa informasi tambahan (kondisi atau evidence) dapat mengubah peluang awal. Inilah yang menjadi dasar dari konsep probabilitas bersyarat (conditional probability) yang ditulis dalam bentuk:

Teorema Bayes menghubungkan dua probabilitas bersyarat, yaitu:

P(A | B) untuk peluang A jika B terjadi dan P(B | A) untuk peluang B jika A terjadi. Bayes kemudian menghitung satu sama lain dengan rumus:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Arti dari rumus itu:

- P(A | B) yang ingin kita cari – peluang A terjadi jika kita tahu B sudah terjadi
- P(B | A) sebaliknya – peluang B terjadi jika kita tahu A sudah terjadi
- P(A) adalah peluang A secara umum (tanpa syarat)
- P(B) adalah peluang B secara umum (tanpa syarat)

Bayangkan:

- 1% orang punya penyakit X→P(A) = 0.01
- Jika punya penyakit X, tes medis positif 99% → P(B | A) = 0.99
- Jika tidak punya penyakit X, test tetap bisa positif 5% → $P(B|\neg A)$=0.05

Jika seseorang melakukan test dan hasilnya positif (B). Kita ingin mengetahui berapa peluang kita benar-benar memiliki punya penyakit X.

Menggunakan Bayes:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Tapi P(B) dihitung dari dua kemungkinan (punya penyakit dan tidak):

$$P(B) = P(B|A) \cdot P(A) + P(B|\neg A) \cdot P(\neg A)$$

Jika kita memasukan angka:

$$P(B) = 0.99(0.01) + 0.05(0.99) = 0.0594$$

Maka:

$$P(A|B) = \frac{0.99\ x\ 0.01}{0.0594} \approx 0.166$$

Jadi peluang orang tersebut benar-benar sakit hanya 16.6% meski hasil tes positif.

Jadi teorema bayes menghubungkan dua arah berpikir:

- Dari penyebab → gejala (misalnya: punya penyakit → tes positif)
- Ke gejala → penyebab (tes positif → kemungkinan punya penyakit)

Teorema Bayes menjadi dasar bagi berbagai algoritma machine learning berbasis probabilitas, seperti Naïve Bayes classifier.

Naïve Bayes menggunakan ide bahwa setiap fitur (kondisi/evidence) memberikan kontribusi terhadap prediksi secara independen. Meskipun sederhana, pendekatan ini terbukti sangat efektif dalam berbagai tugas, seperti:

- Klasifikasi teks dan email spam,
- Analisis sentimen,
- Diagnosa medis berbasis data,
- Prediksi kategori objek berdasarkan ciri visual.

Dalam konteks pembelajaran mesin, memahami kondisi (evidence) dan mengupdate keyakinan terhadap hipotesis (posterior probability) adalah inti dari proses pembelajaran dari data.

Teorema Bayes mengajarkan kita bahwa:

1. Peluang bukanlah nilai mutlak — ia tergantung pada informasi yang kita miliki.
2. Informasi baru dapat mengubah keyakinan lama, menghasilkan peluang a posteriori.
3. Machine learning bekerja dengan prinsip yang sama: mempelajari hubungan antara kondisi (fitur) dan hasil (target) berdasarkan data masa lalu.

Teorema sederhana dari abad ke-18 ini menjadi salah satu fondasi utama kecerdasan buatan modern — membuktikan bahwa pemahaman terhadap *peluang bersyarat* adalah langkah pertama menuju pembelajaran mesin yang cerdas dan adaptif.

8. Statistik

Setelah memahami konsep peluang (probability), langkah berikutnya dalam teori matematika dalam machine learning adalah memahami statistik — ilmu yang berperan penting untuk menggambarkan, menganalisis, dan menafsirkan data yang digunakan oleh algoritma pembelajaran mesin. Probabilitas membantu kita memprediksi kemungkinan suatu kejadian, sedangkan statistik membantu kita menggambarkan kenyataan berdasarkan data yang kita amati.

Statistik merupakan cabang ilmu matematika yang mempelajari cara mengumpulkan, menganalisis, menafsirkan, menyajikan, dan mengorganisasi data. Dalam konteks machine learning, statistik digunakan untuk memahami data dan membantu algoritma

dalam menemukan pola, membuat prediksi, serta menilai performa model (Freedman et al., 2007).

Inti hubungan antara statistik dan machine learning:

- ✓ Statistik → menafsirkan data dan ketidakpastian
- ✓ Machine Learning → belajar dari data untuk membuat keputusan.
    - Populasi (Population): keseluruhan elemen yang ingin kita pelajari. Contoh: semua pengguna aplikasi e-commerce di Indonesia.
    - Sampel (Sample): sebagian kecil dari populasi yang digunakan untuk melakukan analisis. Contoh: 10.000 pengguna yang dipilih secara acak.

Biasanya kita tidak bisa mengakses seluruh populasi, sehingga kita menggunakan inferensi statistik untuk menarik kesimpulan tentang populasi dari sampel.

- Variabel adalah karakteristik yang dapat diukur atau diamati. Contoh: umur, pendapatan, jenis kelamin, warna, tinggi badan, dan sebagainya.
- Jenis Data: Kategorikal (Nominal): data berbentuk label (contoh: jenis kelamin, warna mobil)
- Ordinal: data dengan urutan, tapi jaraknya tidak pasti (contoh: peringkat kepuasan: rendah, sedang, tinggi)
- Interval: jarak antar nilai bermakna, tapi tidak punya nol absolut (contoh: suhu dalam °C)
- Rasio: punya jarak dan nol absolut (contoh: berat badan, umur, gaji)

9. Distribusi : Sebaran Data

Setiap kumpulan data memiliki pola penyebaran nilai yang disebut distribusi (distribution). Distribusi menjelaskan seberapa sering nilai-nilai tertentu muncul di dalam data, atau dengan kata lain, peluang terjadinya suatu nilai.

Distribusi dapat digambarkan secara:

- ✓ Matematis, melalui fungsi distribusi probabilitas (probability density function), atau
- ✓ Visual, melalui grafik seperti histogram dan plot distribusi.

Distribusi membantu kita memahami karakteristik alami data, seperti apakah data cenderung simetris, miring ke satu sisi, memiliki nilai ekstrem (outlier), dan sebagainya. Untuk menggambarkan data secara ringkas, kita sering mencari nilai pusat atau titik keseimbangan dari distribusi. Ada dua ukuran utama:

a. Mean (Rata-rata)

Mean diperoleh dengan menjumlahkan semua nilai dan membaginya dengan jumlah data.

$$Mean = \frac{\sum x_i}{n}$$

mean menggambarkan nilai yang paling diharapkan dalam data. Rata-rata cocok digunakan untuk distribusi yang simestris, dimana nilai-nilai diatras dan dibawah mean tersebar secara seimbang.

Contoh distribusi seperti ini adalah distribusi normal (Gaussian) – berbentuk lonceng (bell-sharped curve) yang sering muncul dalam fenomena alami seperti tinggi badan, berat badan, atau error pengukuran.

b. Median

Median adalah nilai tengah dari data yang sudah diurutkan. Jika jumlah data ganjil , median adalah nilai di posisi tengah. Jika genap, median adalah rata-rata dari dua nilai tengah. Median lebih tahan terhadap nilai ekstrem (outlier) dibanding mean. Misalnya, jika satu orang diantara seratus memiliki penghasilan sangat tinggi, nilai mean akan naik drastis, tetapi median tetap stabil. Karena itu, median sering dianggap lebih adil untuk menggambarkan distribusi yang tidak simetris (skewed distribution)

Selain nilai pusat, machine learning juga memperhatikan keragaman (variasi) data, biasanya menggunakan:

a. Variance

Variance mengukur seberapa jauh nilai-nilai data menyebar dari rata-ratanya.

$$Variance = \frac{\sum(x_i - mean)^2}{n}$$

Nilai variance yang besar berarti data tersebar luas, nilai kecil berarti data berdekatan dengan mean

b. Standard deviation

Simpangan baru menggambarkan jarak rata-rata antara setiap titik data dengan mean-nya. Standar deviasi diambil dari variance. Karena variance berbentuk kuadrat, maka kita mengambil akar kuadratnya

$$\text{Standar Deviation} = \sqrt{\text{Variance}}$$

10. Distribusi Probabilitas (Probability Distributions)

Dalam dunia yang tampak acak, sebenarnya selalu ada pola tersembunyi. Distribusi probabilitas adalah cara kita menggambarkan pola tersebut, ia menunjukkan bagaimana peluang tersebar di antara semua kemungkinan nilai dari suatu variabel acak.

Bayangkan Anda melempar sebuah dadu. Hasilnya bisa 1, 2, 3, 4, 5, atau 6. Setiap angka memiliki peluang yang sama, yaitu 1/6. Jika kita menggambarkannya dalam bentuk grafik, akan tampak bahwa probabilitas tersebar merata di antara keenam kemungkinan itu. Grafik inilah yang disebut distribusi probabilitas , peta peluang yang memperlihatkan nilai mana yang lebih mungkin muncul, dan mana yang jarang.

Agar lebih jelas, mari kita pahami dulu apa yang dimaksud dengan variabel acak (random variable). Variabel acak adalah variabel yang nilainya ditentukan oleh hasil suatu proses acak. Dalam eksperimen seperti melempar koin, dadu, atau mengukur suhu, kita tidak tahu hasil pastinya sebelum percobaan dilakukan. Nilai yang dihasilkan , entah itu "angka 5 pada dadu" atau "suhu 27°C", disebut sebagai realisasi dari variabel acak.

Berdasarkan jenis nilainya, variabel acak dibagi menjadi dua:

1. Variabel Acak Diskret (Discrete Random Variable)

   Jenis ini hanya dapat mengambil nilai-nilai tertentu yang terpisah satu sama lain. Contohnya, jumlah kepala yang muncul saat melempar koin tiga kali bisa bernilai 0, 1, 2, atau 3 — tidak ada nilai di antaranya. Karena itu, kita dapat menghitung peluang untuk setiap nilai secara terpisah.

2. Variabel Acak Kontinu (Continuous Random Variable)

   Berbeda dengan diskret, variabel acak kontinu bisa mengambil nilai di sepanjang suatu rentang yang tidak terputus. Misalnya tinggi badan, waktu, atau suhu. Nilainya bisa 160 cm, 160.2 cm, 160.25 cm, dan seterusnya — tak terbatas banyaknya kemungkinan. Untuk jenis ini, kita tidak lagi menghitung peluang untuk satu nilai tertentu, melainkan untuk rentang nilai, seperti "peluang suhu berada antara 26°C dan 28°C."

Dengan memahami distribusi probabilitas, kita belajar melihat dunia acak dengan cara yang lebih teratur. Distribusi ini menjadi fondasi penting dalam banyak bidang, mulai dari statistika inferensial hingga pembelajaran mesin, karena ia membantu kita menebak, memprediksi, dan memahami perilaku data dalam jangka panjang.

11. Distribusi Bernoulli

Bayangkan Anda melempar sebuah koin. Hasilnya hanya dua: angka atau gambar. Tidak ada kemungkinan lain. Sederhana, bukan? Nah, konsep sesederhana itulah yang menjadi dasar dari distribusi Bernoulli, distribusi paling sederhana dalam dunia probabilitas. Distribusi Bernoulli termasuk dalam kelompok distribusi diskret, karena hasilnya hanya bisa berupa dua nilai yang terpisah:

- 1 → menandakan sukses (dengan probabilitas p)
- 0 → menandakan gagal (dengan probabilitas $1 - p$)

Secara matematis, jika kita menyebut variabel acak X mengikuti distribusi Bernoulli, maka:

$$P(X = 1) = p, P(X = 0) = 1 - p$$

dengan $0 \leq p \leq 1$.

Meskipun tampak sederhana, distribusi Bernoulli memiliki peran yang sangat besar. Ia adalah pondasi dari banyak model probabilistik dan algoritma machine learning modern, terutama yang melibatkan klasifikasi biner, yaitu situasi di mana hasilnya hanya dua kemungkinan.

Contohnya:

- Apakah sebuah email termasuk spam (1) atau bukan spam (0)?
- Apakah sensor mendeteksi rintangan (1) atau tidak (0)?
- Apakah pasien terdiagnosis positif (1) atau negatif (0) terhadap suatu penyakit?

Setiap peristiwa seperti ini bisa dimodelkan sebagai percobaan Bernoulli, percobaan dengan dua kemungkinan hasil, sukses atau gagal. Distribusi Bernoulli juga menjadi blok pembangun bagi distribusi lain yang lebih kompleks, seperti distribusi binomial, yang memodelkan banyak percobaan Bernoulli yang dilakukan berulang kali. Dengan memahami Bernoulli, kita belajar prinsip dasar dari segala bentuk keputusan biner, antara ya dan tidak, hidup dan mati, benar dan salah, yang sering kali menjadi inti dari pengambilan keputusan dalam dunia nyata maupun sistem kecerdasan buatan.

12. Distribusi Binomial

Menentukan peluang mendapatkan k keberhasilan dari n percobaan independen, dimana setiap percobaan memiliki peluang sukses p. Dirumuskan sebagai berikut:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Contoh :

Berapa peluang muncul 3 kepala dari 5 kali lempar koin? Maka kita masukan kedalam rumus:

$$P(X = 3) = \binom{5}{3} (0.5)^3 (0.5)^2 = 10$$

dimana $\binom{n}{k}$ adalah koefisien binomial:

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

Masukan nilai n=5 dan k=3:

$$\binom{5}{3} = \frac{5!}{3!\,(5-3)!} = \frac{5!}{3!\,2!}$$

Jika hitung satu per satu:

$$5! = 5\ x\ 4\ x\ 3\ x\ 2\ x\ 1 = 120$$

$$3! = 3\ x\ 2\ x\ 1 = 6$$

$$2! = \ 2\ x\ 1 = 2$$

Maka:

$$\binom{5}{3} = \frac{120}{6x2} = \frac{120}{12} = 10$$

Kemudian jika kita masukan ke persamaan :

$$P(X = 3) = \binom{5}{3}(0.5)^3(0.5)^2 = 10$$

Maka jika kita lanjutkan perhitungannya:

$$P(X = 3) = 10\,(0.125)(0.25) = 0.3125$$

"Probabilitas dan statistika mengajarkan kita bahwa di balik setiap ketidakpastian selalu ada pola yang bisa dipahami. Dengan keduanya, kita tidak sekadar menebak masa depan, tetapi menalar dan memprediksinya dengan dasar yang ilmiah".

4.5. Limit

Gambar 15
Gambar Timer

Bayangkan Anda sedang mengikuti sebuah permainan sederhana yaitu 10-second timer challenge. Atur stopwatch Anda, lalu tekan tombol "start" dan "stop" dengan tujuan menghentikannya tepat di detik ke-10. Namun, apa yang biasanya terjadi? Sebagian besar orang berhenti di 9.97 detik, atau 10.02 detik, atau bahkan 9.99 detik. Jarang yang benar-benar bisa menghentikannya tepat di angka 10, tetapi semua orang mendekatinya.

Di sinilah konsep limit dalam matematika muncul mengenai Limit mengenai seberapa dekat kita bisa mendekati angka itu.

Dalam contoh ini:

- Detik ke-10 adalah nilai yang ingin dicapai.
- Waktu yang dicatat (9.99, 10.01, 9.98, dst.) adalah nilai-nilai yang mendekati.
- Semakin kecil selisihnya, semakin dekat kita pada limit waktu 10 detik.

Contoh lainnya, misalkan kita sedang mengendari mobil dijalan yang lurus. Kita menyalakan perekam jarak tempuh yang menampilkan posisi mobil setiap detik. Hasil perekaman yag dilakukan adalah sebagai berikut:

Tabel 2
Hasil Rekaman Posisi Mobil

Waktu (detik)	Posisi (meter)
1	10
2	25
3	45
4	70

Dari tabel ini, kita bisa tahu bahwa:

- antara detik ke-1 dan ke-2, mobil menempuh 15 meter,
- antara detik ke-2 dan ke-3, menempuh 20 meter,
- antara detik ke-3 dan ke-4, menempuh 25 meter.

Kalau kita hitung rata-ratanya, mobil tampak semakin cepat.

Namun jika muncul pertanyaan mengenai berapa kecepatan mobil tepat pada detik ke-3? Kita tidak bisa langsung menjawabnya hanya dari tabel. Karena tabel ini menunjukkan jarak setiap satu detik, bukan saat yang persis detik ke-3.

Namun, kita bisa memperkirakan:

- Dari detik 2 ke 3 → kecepatan rata-rata = (45–25)/(3–2) = 20 m/s
- Dari detik 3 ke 4 → kecepatan rata-rata = (70–45)/(4–3) = 25 m/s

Kalau kita ambil jeda waktu semakin kecil, misalnya:

- dari 2,9 ke 3,0 detik,
- dari 2,99 ke 3,0 detik,
- dari 2,999 ke 3,0 detik,

maka kita akan mendapatkan angka kecepatan yang makin mendekati nilai tertentu. Nilai yang terus "didekati" itulah yang dalam matematika disebut limit. Limit membantu kita mendefinisikan nilai yang didekati oleh suatu besaran (fungsi), meskipun nilai itu tidak bisa atau belum tentu bisa dihitung secara langsung.

Kalau posisi mobil ditulis sebagai fungsi:

$$s(t) = \text{posisi mobil pada waktu } t,$$

maka kecepatan rata-rata antara dua waktu, t dan $t + h$, adalah:

$$v = \frac{s(t+h) - s(t)}{h}.$$

Sekarang, kalau kita ingin tahu kecepatan sesaat di waktu t, kita buat jeda waktu h makin kecil, makin kecil, dan makin kecil lagi. Matematikanya ditulis sebagai:

$$v(t) = \lim_{h \to 0} \frac{s(t+h) - s(t)}{h}.$$

Inilah yang nantinya menjadi dasar turunan (derivative) dalam kalkulus.

Mari kita lanjutkan contoh sederhana dalam fungsi matematis sebagai berikut:

Jika kita ingin mengetahui apa yang terjadi pada fungsi

$$f(x) = \frac{x^2 - 4}{x - 2}$$

ketika x mendekati 2, kita sedang mencari

$$\lim_{x \to 2} \frac{x^2 - 4}{x - 2}$$

kita tidak sedang tanya nilai fungsi tepat di $x = 2$ karena di sana fungsi ini tidak terdefinisi, melainkan nilai yang didekati fungsi ketika x makin mendekati 2. Maksudnya adalah jika kita masukkan $x = 2$ ke pembilang dan penyebut:

- Pembilang: $2^2 - 4 = 4 - 4 = 0$
- Penyebut: $2 - 2 = 0$

Jadi kita dapat $\frac{0}{0}$. Itu disebut bentuk tak tentu, jadi tidak bisa langsung dipakai. Kita perlu cara lain.

Ambil beberapa angka dekat 2 (tapi bukan 2), lalu hitung $f(x)$:

Tabel 3
Hasil Hitung Normalisasi

x	$x^2 - 4$	$x - 2$	$f(x) = \frac{x^2 - 4}{x - 2}$
1.9	3.61−4 = −0.39	−0.1	$-0.39/(-0.1) = 3.9$
1.99	3.9601−4 = −0.0399	−0.01	3.99
2.01	4.0401−4 = 0.0401	0.01	4.01
2.1	4.41−4 = 0.41	0.1	4.1

Perhatikan pola: ketika $x$semakin dekat ke 2 (dari kiri atau kanan), $f(x)$mendekati 4. Itu berarti limitnya adalah 4:

$$\lim_{x \to 2} \frac{x^2 - 4}{x - 2} = 4$$

Jika kita menggunakan perhitungan limit hasilnya pun tidak jauh berbeda, dengan tahapan sebagai berikut:

$$f(x) = \frac{x^2 - 4}{x - 2}$$

Jika kita faktorkan pembilangnya:

$$x^2 - 4 = (x - 2)(x + 2)$$

Maka :

$$f(x) = \frac{(x-2)(x+2)}{(x-2)}$$

Sehingga kita bisa menyederhanakan:

g(x)=x+2

maka limit $f(x)$ saat $x \to 2$ akan sama dengan nilai $g(x)$ di $x = 2$.

$$\lim_{x \to 2} f(x) = g(2) = 2 + 2 = 4$$

Analogi singkat: bayangkan mobil mendekati persimpangan. Kita tidak peduli posisi mobil tepat pada persimpangan, melainkan arah dan kecepatannya saat mendekat — nilai yang "didekati" itulah limit.

Kebetulan saja, karena $x = 2$ membuat $x + 2 = 4$, hasilnya bulat. Kalau titik yang didekati lain, hasilnya bisa pecahan atau desimal.

Misalnya:

$$\lim_{x \to 1.5} \frac{x^2 - 4}{x - 2} = 1.5 + 2 = 3.5$$

Limit berguna karena banyak fungsi yang *tidak bisa langsung disubstitusi*, dan limit membantu kita menemukan nilai yang mendekati di sekitar titik tersebut. Limit sebenarnya menjawab pertanyaan yang lebih filosofis mengenai apa yang terjadi pada fungsi ketika kita makin dekat ke titik tertentu, terlepas dari apakah fungsi itu punya nilai di titik itu atau tidak. Jadi limit tidak diciptakan hanya untuk memperbaiki fungsi yang "rusak", tapi untuk mendefinisikan makna kedekatan dan perubahan yang halus dalam matematika.

Beberapa kegunaan limit adalah:

a. Menangani titik yang tidak bisa langsung dihitung (seperti 0/0), seperti:

$$f(x) = \frac{x^2 - 4}{x - 2}$$

Nilai di x=2 tidak ada (karena 0/0), tapi kita bisa tahu menuju ke mana f(x) saat x mendekati 2 → hasilnya 4. Jadi limit "mengisi lubang" di situ.

b. Menentukan *perilaku mendekati*, meskipun fungsi bisa dihitung
Contoh:

$$f(x) = 3x + 1$$

Kalau kita tanya:

$$\lim_{x \to 2} f(x)$$

hasilnya jelas 7.
Tapi apakah kita butuh limit di sini?

Mungkin tidak untuk fungsi ini, tapi justru konsep limit inilah yang memungkinkan kita mendefinisikan turunan dan integral di kemudian hari.

c. Limit adalah pondasi turunan (derivative)
Turunan didefinisikan dengan limit:

$$f'(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}$$

Tanpa konsep limit, rumus ini tidak bisa didefinisikan secara matematis. Jadi — meskipun untuk fungsi sederhana limit terasa "tidak perlu", limit tetap dibutuhkan agar kita bisa bicara tentang kecepatan perubahan yang halus, kemiringan garis singgung, dan kontinuitas.

Konsep limit mengajarkan kita untuk memahami perilaku fungsi ketika mendekati titik tertentu, bukan hanya di titik itu sendiri. Dengan memahami limit:

- Kita dapat menganalisis kontinuitas fungsi,
- Menentukan sifat perubahan fungsi,
- Dan membuka jalan menuju turunan (derivative) serta integral dalam kalkulus.

Singkatnya, limit adalah jembatan antara fungsi biasa dan kalkulus.
Ia memberi kita cara untuk memahami perubahan yang "hampir terjadi" dengan presisi matematika.

Limit itu seperti dasar fondasi sebelum kita mempelajari:

1. Turunan (derivative) → menggambarkan seberapa cepat sesuatu berubah, misalnya kecepatan dari jarak terhadap waktu.
   Tapi "kecepatan sesaat" hanya bisa dihitung dengan limit — karena kita mencari perubahan pada waktu yang "hampir nol".

2. Integral → menghitung luas di bawah kurva, atau total perubahan dalam periode tertentu. Konsepnya juga muncul dari menjumlahkan potongan kecil yang jumlahnya "mendekati tak hingga banyak".

Kontinuitas fungsi → apakah grafik fungsi nyambung atau terputus di suatu titik. Limit membantu menentukan apakah nilai fungsi di titik itu sejalan dengan arah datangnya grafik dari kiri dan kanan.

Limit digunakan untuk menentukan nilai yang didekati suatu fungsi ketika variabelnya mendekati titik tertentu, meskipun fungsi tersebut tidak selalu terdefinisi tepat di titik itu. Konsep limit menjadi dasar bagi turunan dan integral, karena ia menjelaskan perilaku fungsi secara mendekati — bukan hanya pada titik tertentu.

4.6. Derivative

Kemiringan garis menggambarkan seberapa cepat suatu nilai vertikal (biasanya y) berubah terhadap nilai horizontal (biasanya x).

Secara matematis:

$$\text{slope} = \frac{\Delta y}{\Delta x}$$

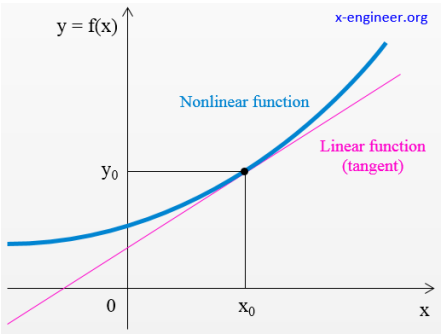
dimana:

- Δy: perubahan nilai pada sumbu vertikal (y),
- Δx: perubahan nilai pada sumbu horizontal (x).

Simbol segitiga Yunani Δ(dibaca *delta*) berarti "perubahan". Dalam Analogi sederhananya, misalkan kita sedang menaiki bukit:

- Jika jalannya curam, maka $\Delta y/\Delta x$ besar → slope tinggi.
- Jika jalannya landai, maka $\Delta y/\Delta x$ kecil → slope rendah.
- Jika jalan datar, maka $\Delta y/\Delta x = 0$.

Pembahasan diatas valid untuk garis lurus, dimana kemiringan selalu konstan di setiap titik. Namun, bagaimana jika yang kita hadapi bukan garis lurus, melainkan kurva seperti lintasan pelari atau grafik pertumbuhan penduduk?



Gambar 16
Ilustrasi Garis (Linear) dan Kurva (NonLiear)
(Sumber: https://x-engineer.org)

Kurva menunjukkan bahwa laju perubahan tidak konstan; ia bisa meningkat atau menurun di setiap titik. Untuk mengukur rata-rata laju perubahan antara dua titik pada kurva, kita dapat menggambar garis sekan (secant line), yaitu garis yang menghubungkan dua titik pada kurva.

Rumusnya tetap sama:

$$\text{slope rata-rata} = \frac{\Delta y}{\Delta x}$$

Namun, nilai slope ini tergantung pada dua titik yang kita pilih. Jika titiknya berbeda, nilai slope-nya juga bisa berbeda, dengan kata lain laju perubahan tidak konstan. Untuk memahaminya mari kita bahas contoh yang lebih detail. Pada garis lurus laju perubahannya selalu sama. Misalkan kalau kita punya garis lurus :

$$y = 2x + 1$$

Berarti, setiap kali x naik 1 satuan maka y selalu naik 2 satuan. Mari kita ilustrasikan pada tabel dibawah ini:

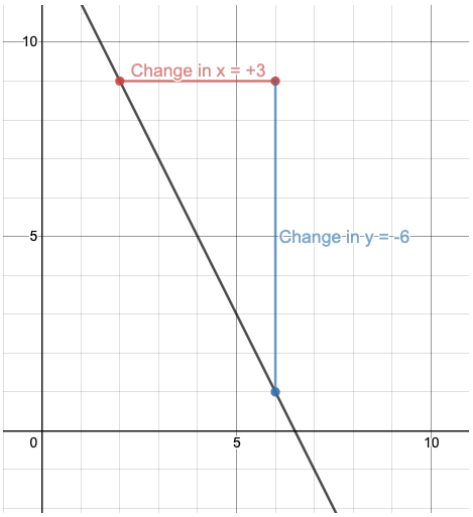
Tabel 4
Slope

x	y = 2x + 1	Perubahan y (Δy)	Perubahan x (Δx)	Slope (Δy/Δx)
0	1	-	-	-
1	3	+2	+1	2
2	5	+2	+1	2

Jadi berapa pun dua titik yang kita ambil, kemiringannya (slope) akan selalu sama:

$$\text{slope} = \frac{\Delta y}{\Delta x} = 2$$

Itulah sebabnya kita bilang, untuk garis lurus, laju perubahannya konstan.



Gambar 17
Ilustrasi Slope
(Sumber: Wikimedia)

Sekarang, bayangkan fungsi yang tidak lurus, misalnya:

$$y = x^2$$

Kalau kita hitung perubahan y terhadap x di dua titik berbeda, hasilnya tidak akan sama.

Tabel 5
Slope Kemiringan

Titik 1 (x_1, y_1)	Titik 2 (x_2, y_2)	Δx	Δy	Slope = Δy / Δx
(1, 1)	(2, 4)	1	3	3
(2, 4)	(3, 9)	1	5	5

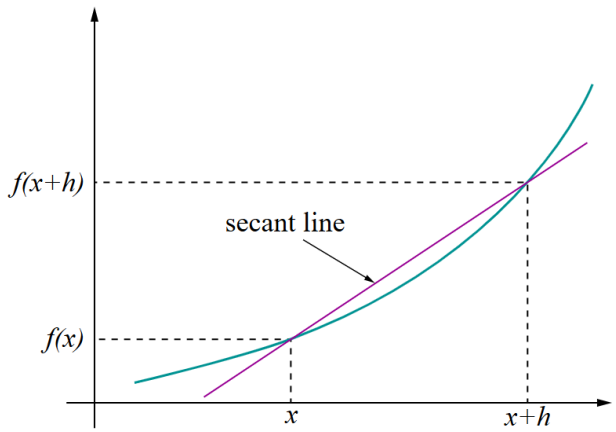
Perhatikan:

Antara $x = 1$ dan $x = 2$, slope = 3.

Antara $x = 2$ dan $x = 3$, slope = 5

Artinya: kemiringan (atau laju perubahan) tidak tetap. Semakin ke kanan, garisnya makin curam, dengan kata lain y bertambah semakin cepat terhadap x.

Jika kita gambar grafik $y = x^2$, dan ambil dua titik, misalnya titik A(1,1) dan B(2,4) diatas lalu kamu hubungkan keduanya dengan garis lurus, maka garis itu disebut garis sekan (Secant Line). Garis itu memotong kurva di dua titik (tidak hanya menyentuh satu titik). Garis sekan mewakili rata-rata laju perubahan antara dua titik.

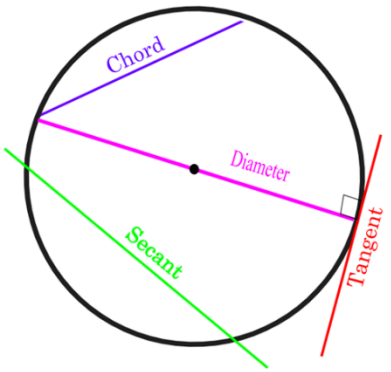


Gambar 18
Ilustrasi Secant Line
(Sumber : Wikimedia)

Kalau slope-nya berubah-ubah, bagaimana cara mencari kemiringan tepat di satu titik? Di sinilah muncul gagasan penting:

- Garis sekan memberi kita laju perubahan rata-rata antara dua titik.
- Tapi kita ingin tahu laju perubahan sesaat — yaitu pada satu titik saja.

Untuk menjawab itu, kita gunakan garis singgung (tangent line) — yaitu garis yang hanya menyentuh kurva di satu titik tanpa memotongnya.



Gambar 19
Ilustrasi Tangent Line
(Sumber: Wikimedia)

Secara matematis untuk mendapatkan laju perubahan tepat di satu titik, kita buat dua titik itu semakin berdekatan:

$$x_2 \to x_1$$

Maka:

$$\text{slope garis singgung} = \lim_{x_2 \to x_1} \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

Dan hasil limit inilah yang disebut turunan.

- Turunan menggambarkan laju perubahan sesaat suatu fungsi.
- Secara geometris, turunan adalah kemiringan garis singgung pada titik tertentu.

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Contoh :

Misalkan $y$menunjukkan posisi pelari (meter), dan $x$adalah waktu (detik). Jika kita menghitung $\Delta y/\Delta x$ antara dua waktu berbeda, kita mendapat kecepatan rata-rata. Namun, jika kita ingin tahu kecepatan Usain Bolt tepat pada detik ke-5, kita harus mencari laju perubahan sesaat, yaitu slope garis singgung pada titik itu.

Dengan kata lain:

$$\text{kecepatan sesaat} = \frac{dy}{dx}$$

yang artinya "perubahan posisi terhadap waktu dalam selang waktu yang sangat kecil".

Jadi secara Kkonsep turunan lahir dari ide memperkecil jarak antar titik pada kurva. Tahapan berpikirnya adalah sebagai berikut:

1. **Mulai dari sekan**

   Kita mulai dari yang bisa kita ukur:

   $$\text{slope sekan} = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

   Itu mudah, karena kita tahu dua titiknya. Namun kalau kita ingin tahu "kemiringan tepat di x_1", kita tidak bisa langsung ambil satu titik saja, itu disebabkan karena slope memerlukan dua titik. Sehingga kita memerlukan pendekatan yang membuat titik kedua semakin dekat ke yang pertama.

2. **Garis singgung muncul sebagai *limit* dari garis sekan**

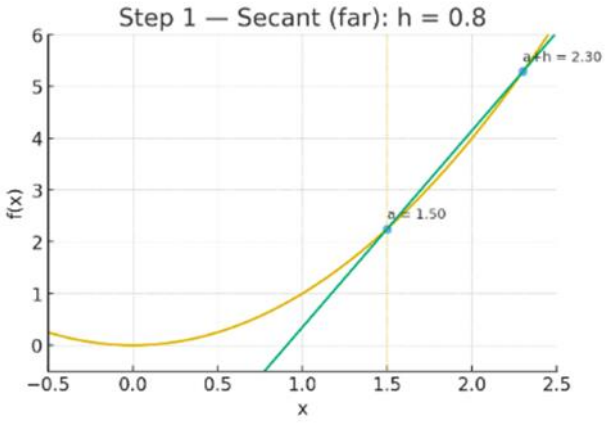
   Ketika jarak kedua titik semakin kecil, garis sekan makin "melekat" pada kurva, hingga pada akhirnya menyentuh kurva hanya di satu titik. Pada saat itulah, garis sekan berubah menjadi garis singgung (*tangent line*).

Secara matematis:

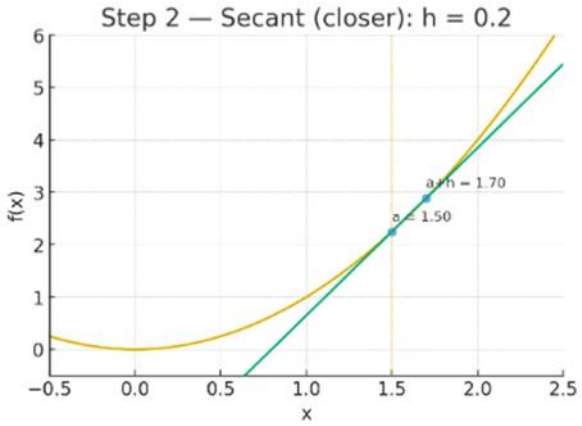
$$\text{slope tangent line di } x = a = \lim_{x \to a} \frac{f(x) - f(a)}{x - a}$$

Dan nilai limit itu adalah turunan, $f'(a)$.

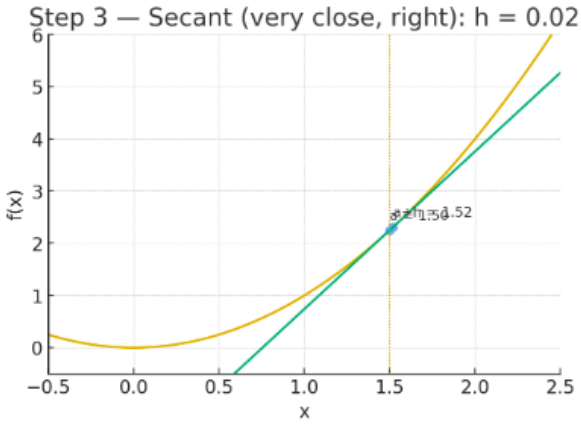
Ilustrasi secant line menuju tangent line adalah sebagai berikut:



Gambar 20
Step 1 Secant Line



Gambar 21
Step 2 Secant Line Bersinggungngan



Gambar 22
Step 3 Secant

Didefinisikan:

$$f(x) = x^2$$

dan titik pusat:

$$a = 1.5$$

maka:

$$f(a) = (1.5)^2 = 2.25$$

Turunan analitiknya:

$$f'(x) = 2x$$

jadi di $x = 1.5$:

$$f'(1.5) = 3$$

Artinya, kemiringan garis singgung adalah 3, dan garisnya melalui titik $(1.5, 2.25)$:

$$y - 2.25 = 3(x - 1.5)$$

atau:

$$y = 3x - 2.25$$

Jika:

$$y=31.5-2.25$$

$$y=4.5-2.25$$

$$y=2.25$$

Jadi secara matematis:

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$$

Inilah definisi formal dari turunan, kemiringan garis singgung pada titik tertentu sebagaimana yang sudah kita bahas diatas. Dengan memahami turunan, kita bisa:

- Mengetahui kecepatan sesaat,
- Mencari puncak atau lembah suatu fungsi (optimasi),
- Menganalisis dinamika sistem di berbagai bidang (fisika, ekonomi, biologi, AI).

Turunan bisa dihitung untuk setiap titik (setiap momen), sehingga kita memperoleh fungsi laju perubahan.

Contohnya:

$$f(x) = x^2 \Rightarrow f'(x) = 2x$$

- Jika $x = 1$, laju perubahannya 2.
- Jika $x = 2$, laju perubahannya 4.
- Jika $x = 3$, laju perubahannya 6.

Artinya, di setiap titik (momen) pada kurva, kita tahu "seberapa cepat" $f(x)$berubah di situ.

Tabel 6

Konsep dan Makna

Konsep	Makna
Slope (Kemiringan)	Laju perubahan rata-rata antara dua titik
Secant Line (Garis Sekan)	Garis yang menghubungkan dua titik pada kurva
Tangent Line (Garis Singgung)	Garis yang menyentuh kurva di satu titik
Derivative (Turunan)	Laju perubahan sesaat — slope dari garis singgung
Rumus Umum	$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$

"Turunan menyatakan laju perubahan sesaat suatu fungsi terhadap variabelnya, yaitu kemiringan garis singgung pada titik tertentu di kurva. Dengan turunan, kita dapat memahami bagaimana suatu besaran berubah dari waktu ke waktu atau dari satu kondisi ke kondisi lain secara tepat dan terukur".

4.7. Integral

Bayangkan kamu lagi mobil yang sedang melaju, dan kita punya grafik posisi terhadap waktu seperti ini:

$$y = f(t)$$

Artinya:

- y: posisi mobil (dalam km)
- t: waktu (dalam jam)

Sekarang:

- Kalau kita menurunkan posisi, kita dapat kecepatan.
- Tapi kalau kita mengintegralkan kecepatan, kita dapat posisi.

Jadi:

Turunan ⇒ dari posisi ke kecepatan
Integral ⇒ dari kecepatan ke posisi

Secara fisika dasar kecepatan adalah seberapa cepat posisi berubah terhadap waktu. Kalau dalam 1 jam posisi berubah 60 km, maka kecepatannya 60 km/jam.

Secara matematis:

$$v = \frac{\Delta y}{\Delta t}$$

dan ketika kita ingin tahu kecepatan tepat pada satu saat, maka:

$$v(t) = \frac{dy}{dt}$$

Artinya apa?

Turunan $\frac{dy}{dt}$ itu seperti melihat kemiringan grafik posisi terhadap waktu, dimana:

- Jika grafiknya curam → posisi cepat naik → mobil cepat.
- Jika grafiknya datar → posisi tidak berubah → mobil diam.

Jadi, turunan posisi terhadap waktu memberi tahu laju perubahan posisi, yaitu kecepatan. Sekarang kita balik, dari kecepatan ke posisi. Kalau turunan memberi tahu seberapa cepat posisi berubah, maka integral menjawab pertanyaan kebalikannya: "Kalau saya tahu kecepatan mobil setiap saat, berapa jauh total jarak yang sudah ditempuh?"

Misalkan kecepatan mobil bervariasi setiap jam seperti ini:

Tabel 7
Kecepatan Mobil

Waktu (jam)	Kecepatan (km/jam)
0–1	40
1–2	50
2–3	60

Kalau kecepatannya tetap 40 km/jam selama 1 jam, jarak yang ditempuh:

$$40 \times 1 = 40 \text{ km}$$

Kalau kecepatannya 50 km/jam di jam berikutnya:

$$50 \times 1 = 50 \text{ km}$$

Dan seterusnya.

$$\text{Total jarak} = 40 + 50 + 60 = 150 \text{ km}$$

Tapi bagaimana jika kecepatannya berubah terus? Maka kita pecah waktu jadi potongan-potongan kecil (Δt kecil sekali):

$$\text{Jarak total} \approx \sum v(t)\,\Delta t$$

Semakin kecil potongannya, semakin akurat hasilnya. Kalau Δt dibuat sangat kecil (mendekati nol), penjumlahan ini menjadi integral:

$$s(t) = \int v(t)\,dt$$

Jadi turunan dan integral adalah operasi kebalikan, misalkan dari posisi $y = f(t)$ lalu kita terapkan operasi turunan $\frac{dy}{dt}$ maka menjadi kecepatan $v(t)$ yang dapat kita interpretasikan sebagai laju perubahan posisi. Sedangkan jika dari kecepatan $v(t)$ lalu kita lakukan operasi integral $\int v(t)dt$, maka menjadi posisi $y(t)$ yang dapat kita intepretasikan sebagai akumulasi perubahan posisi.

Keduanya terkait oleh proses limit:

- Turunan menghitung limit dari perbandingan perubahan kecil (seberapa cepat sesuatu berubah di titik itu).

$$\frac{dy}{dt} = \lim_{\Delta t \to 0} \frac{\Delta y}{\Delta t}$$

- Integral menghitung limit dari penjumlahan kecil-kecil (berapa banyak perubahan yang terkumpul).

$$\int v(t)\,dt = \lim_{\Delta t \to 0} \sum v(t_i)\,\Delta t$$

Jadi secara filosofi:

Turunan memecah perubahan menjadi per-unit kecil (analisis mikro). Integral menyatukan perubahan kecil menjadi total (analisis makro). Dalam bahasa sederhananya: Turunan melihat perubahan kecil dari yang besar. Integral menyatukan perubahan kecil menjadi yang besar.

Mari kita tambahkan dengan contoh agar lebih memahami:
Bayangkan kita duduk di dalam mobil dengan jendela tertutup rapat. Kamu tidak bisa melihat ke luar, hanya bisa melihat speedometer (alat pengukur kecepatan). Selama 8 detik, mobil bergerak — mulai dari diam, melaju makin cepat, lalu melambat lagi hingga berhenti. Pertanyaannya: "Bagaimana caranya mengetahui berapa jauh mobil telah berjalan hanya dengan melihat kecepatan setiap saat?"

Kita misalkan kecepatan mobil (dalam meter per detik) berubah terhadap waktu (dalam detik) dengan fungsi berikut:

$$v(t) = t(8 - t)$$

Artinya:

- Saat $t = 0$, $v(0) = 0$(mobil diam).
- Saat $t = 4$, $v(4) = 16$m/s (kecepatan maksimum).
- Saat $t = 8$, $v(8) = 0$(mobil berhenti).

Angka 8 muncul karena mobil bergerak selama total 8 detik.

Fungsi $v(t) = t(8 - t)$ dipilih secara sengaja agar:

- Di awal ($t = 0$), $v(0) = 0\rightarrow$ mobil diam.
- Di akhir ($t = 8$), $v(8) = 0\rightarrow$ mobil berhenti lagi.
- Di antara keduanya, nilai $v(t)$positif dan berbentuk *parabola terbalik*, artinya kecepatan meningkat, mencapai puncak, lalu menurun kembali ke nol.

Kita punya:

$$v(t) = t(8 - t)$$

Mari kita uraikan dulu bentuknya:

$$v(t) = 8t - t^2$$

Kalau diperhatikan, ini adalah fungsi kuadratik dengan bentuk umum:

$$v(t) = -t^2 + 8t$$

yang memiliki koefisien negatif di depan t^2 yang artinya grafiknya parabola terbuka ke bawah (melengkung ke bawah seperti gunung). Karena pangkat tertinggi adalah t^2, grafik fungsi ini adalah parabola. Lalu kita menghitung turunan dari $y = at^2 + bt + c$ terhadap t sebagai berikut:

$$\frac{dy}{dt} = 2at + b$$

Kita ingin cari titik di mana turunan ini nol (karena di puncak, grafik "berhenti menanjak" karena slopenya datar sebelum turun lagi):

$$2\mathrm{at} + \mathrm{b} = 0$$

Dari sini:

$$t = -\frac{b}{2a}$$

Inilah rumus posisi puncak parabola pada sumbu-x (atau sumbu-t). Nah, pada $v(t) = -t^2 + 8t$:

- $a = -1$
- $b = 8$

Masukkan ke rumus:

$$t = -\frac{8}{2(-1)} = \frac{8}{2} = 4$$

Jadi puncak parabola (kecepatan maksimum) terjadi saat $t = 4$.

Tabel berikut menggambarkan nilai kecepatan dari mobil :

Tabel 8

Nilai Kecepatan Dari Mobil

Waktu t	Rumus $v(t) = t(8 - t)$	Hasil (m/s)	Makna Fisik
0	$0(8-0)$	0	mulai diam
2	$2(8-2)$	12	mulai cepat
4	$4(8-4)$	16	kecepatan maksimum
6	$6(8-6)$	12	melambat
8	$8(8-8)$	0	berhenti
0	$0(8-0)$	0	mulai diam

Karena kecepatan kita tidak konstan sebagaimana yang kita lihat diatas, maka kita perlu menjumlahkan jarak kecil dari setiap potongan waktu. Misalkan kita bagi waktu menjadi interval kecil sebesar Δt. Pada tiap interval kecil, kita anggap kecepatan *hampir konstan*, misalnya $v(t_i)$.

Maka jarak kecil yang ditempuh di interval itu adalah:

$$\Delta s_i = v(t_i) \cdot \Delta t$$

Jika kita jumlahkan semua potongan itu dari awal hingga akhir, kita peroleh perkiraan total jarak:

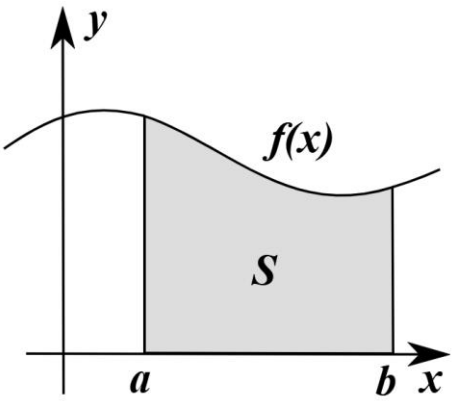
$$s \approx \sum v(t_i) \cdot \Delta t$$

Semakin kecil Δt, semakin akurat hasilnya.

Ketika $\Delta t \to 0$, penjumlahan ini menjadi integral:

$$s = \int_0^8 v(t)\, dt$$

yang berarti "Jumlahkan semua potongan kecil kecepatan × waktu dari t = 0 sampai t = 8.". Kalau kita menggambar grafik $v(t)$(kecepatan terhadap waktu), setiap $v(t_i) \cdot \Delta t$ adalah luas persegi panjang kecil di bawah kurva. Maka ketika Δt makin kecil dan jumlahnya makin banyak, penjumlahan semua persegi panjang itu menjadi luas di bawah kurva v(t). Luas inilah jarak total yang ditempuh.



Gambar 23
Ilustrasi Integral sebagai area dibawah kurva
(Sumber: wikimedia)

" Integral adalah proses matematis untuk menghitung luas di bawah kurva suatu fungsi, yang merepresentasikan penjumlahan terus-menerus dari perubahan kecil. Dalam konteks computer science dan machine learning, integral digunakan untuk menghitung probabilitas total di bawah fungsi kepadatan, mengestimasi area dalam distribusi kontinu, serta mengoptimalkan model melalui perhitungan gradien dan loss yang bersifat kontinu."

4.8. Partial Derivative

Dalam machine learning, terutama pada proses optimisasi seperti pelatihan model neural network, kita sering ingin tahu seberapa cepat fungsi berubah terhadap setiap variabel inputnya. Perubahan inilah yang ditangkap oleh turunan (derivative).

Ketika fungsi kita hanya memiliki satu variabel, misalnya

$$f(x) = x^2$$

turunan memberi tahu kita laju perubahan dari fungsi tersebut terhadap x, yaitu:

$$\frac{df}{dx} = 2x$$

Namun dalam machine learning, fungsi seringkali bergantung pada banyak variabel. Contohnya, fungsi loss pada neural network bisa ditulis:

$$L(w_1, w_2, \dots, w_n)$$

yang berarti nilai loss bergantung pada banyak bobot (weights) w_i. Untuk menganalisis bagaimana perubahan setiap bobot memengaruhi nilai loss, kita memerlukan turunan parsial (partial derivative). Sebelum memahami turunan parsial, mari kita ingat kembali turunan biasa.

Jika kita punya fungsi:

$$y = f(x)$$

maka turunan $\frac{dy}{dx}$ menunjukkan laju perubahan y terhadap x.

Sebagai contoh:

$$f(x) = 3x^2 + 2x$$
$$\frac{df}{dx} = 6x + 2$$

Turunan ini disebut total derivative karena perubahan f sepenuhnya bergantung pada satu variabel saja, yaitu x.

Sekarang, misalkan kita memiliki fungsi dua variabel:

$$f(x, y) = 3x^2 + 2y$$

Ketika kita ingin tahu bagaimana perubahan $x$memengaruhi f, kita hanya menurunkan terhadap $x$sambil menganggap $y$sebagai konstan. Inilah yang disebut turunan parsial, dan dilambangkan dengan simbol "∂" (dibaca: del atau partial).

$\frac{\partial f}{\partial x} = 6x$, karena y dianggap konstan.

Demikian pula, turunan parsial terhadap $y$berarti kita memperlakukan $x$sebagai konstan:

$$\frac{\partial f}{\partial y} = 2$$

Jadi, turunan parsial digunakan untuk mengisolasi efek satu variabel saja pada suatu fungsi yang memiliki banyak variabel.

Berbeda dengan turunan parsial, turunan total (total derivative) memperhitungkan kemungkinan bahwa satu variabel bergantung pada variabel lain. Misalnya, jika $y$bukan variabel bebas, melainkan fungsi dari x:

$$y = y(x)$$

maka fungsi kita menjadi:

$$f(x,y) = 3x^2 + 2y(x)$$

Dalam kasus ini, perubahan $x$akan memengaruhi $f$secara langsung melalui x dan secara tidak langsung melalui y(x).

Turunan totalnya menjadi:

$$\frac{df}{dx} = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dx}$$

Substitusi dari fungsi di atas:

$$\frac{df}{dx} = 6x + 2 \cdot \frac{dy}{dx}$$

Jadi perbedaan utama pada turunan total dan turunan parsial adalah:

- Turunan parsial: menganggap variabel lain konstan.
- Turunan total: memperhitungkan keterkaitan antar variabel.

" Turunan parsial menggambarkan bagaimana suatu fungsi multivariat berubah terhadap satu variabel saja, sementara variabel lainnya dianggap konstan. Konsep ini sangat penting dalam machine learning karena menjadi dasar perhitungan gradien yang digunakan untuk mengarahkan proses optimisasi model seperti pada algoritma gradient descent".

4.9. Probability Density Function

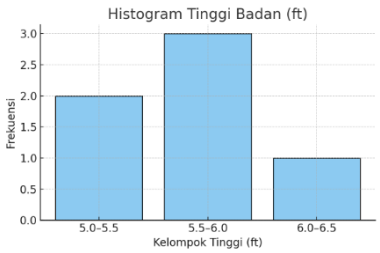
Dalam kehidupan sehari-hari, kita sering mendengar istilah *distribusi* dalam konteks statistik, misalnya "*distribusi tinggi badan,*" "*distribusi pendapatan,*" atau "*distribusi nilai ujian.*" Namun, apa sebenarnya makna distribusi itu? Secara sederhana, distribusi adalah cara untuk menggambarkan bagaimana data tersebar atau terdistribusi. Ia memberi tahu kita seberapa sering suatu nilai muncul dan seberapa besar kemungkinan kita mendapatkan nilai tertentu.

Untuk memahami apa itu distribusi, mari kita ambil contoh sederhana: mengukur tinggi badan sekelompok orang. Katakanlah kita mengukur tinggi enam orang:

Tabel 9
Data Tinggi Badan

No	Tinggi (ft)	Kategori Bin (kelompok)
1	5.2	5.0 – 5.5
2	5.8	5.5 – 6.0
3	5.6	5.5 – 6.0
4	5.9	5.5 – 6.0
5	5.1	5.0 – 5.5
6	6.3	6.0 – 6.5

Jika kita "menumpuk" data tinggi badan ini berdasarkan kelompok (*bin*), kita mendapatkan gambar berupa histogram.



Gambar 24
Ilustrasi Histogram

Histogram adalah grafik batang yang menunjukkan berapa banyak data jatuh ke dalam setiap kelompok. Dari tabel di atas:

Tabel 10
Hasil Bin Tinggi Badan

Bin (ft)	Jumlah Orang	Frekuensi
5.0 – 5.5	2	33.3%
5.5 – 6.0	3	50%
6.0 – 6.5	1	16.7%

Dari histogram, terlihat bahwa kebanyakan orang memiliki tinggi antara 5.5 hingga 6.0 kaki. Artinya, jika kita memilih seseorang secara acak, kemungkinan besar tinggi badannya berada dalam rentang ini. Dengan kata lain, histogram menunjukkan probabilitas relatif dari setiap rentang nilai.

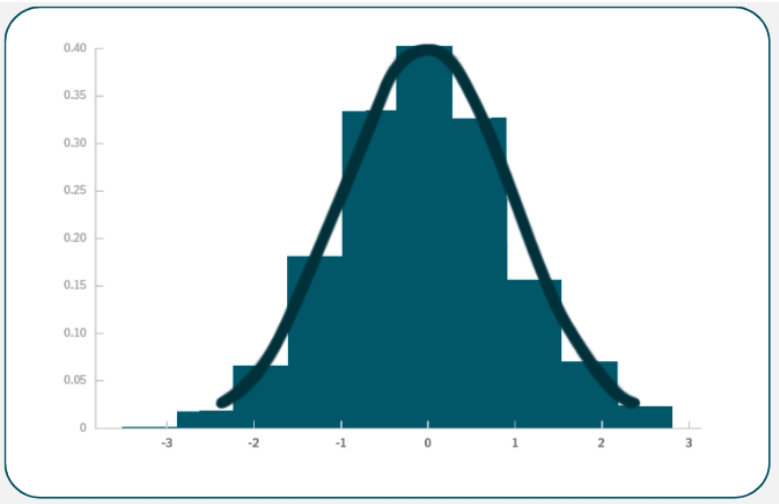
Ketika bin dikecilkan maka distribusi menjadi lebih detail. Sekarang, bayangkan kita memperkecil ukuran bin. Misalnya, bukan 0.5 ft per bin, tetapi 0.25 ft. Kita akan mendapatkan pembagian yang lebih detail seperti berikut:

Tabel 11
Hasil Pembagian Tinggi Badan

Bin (ft)	Jumlah Orang
5.0–5.25	1
5.25–5.5	1
5.5–5.75	1
5.75–6.0	2
6.0–6.25	0
6.25–6.5	1

Dengan semakin banyak data dan semakin kecil ukuran bin, kita memperoleh estimasi yang lebih akurat dan lebih halus tentang bagaimana data tersebar.Histogram memberikan gambaran kasar tentang distribusi data, tetapi sering kali tampak bergerigi atau tidak halus. Untuk membuatnya lebih mudah dianalisis, kita bisa menghaluskan histogram menjadi kurva distribusi.

Untuk memahami pola penyebaran data dengan lebih baik, kita dapat menghaluskan histogram sehingga terbentuk kurva yang kontinu. Kurva halus inilah yang kemudian dikenal sebagai kurva distribusi, yang secara matematis direpresentasikan oleh fungsi kepadatan probabilitas (Probability Density Function / PDF).



Gambar 25
Ilustrasi Kurva Distribusi
(Sumber: https://www.financestrategists.com)

Kurva distribusi memiliki beberapa kelebihan:

1. Kontinuitas: Tidak tergantung pada ukuran bin.
2. Kalkulasi presisi tinggi: Kita dapat menghitung probabilitas untuk rentang nilai sekecil apa pun, misalnya antara 5.021 dan 5.317 kaki.
3. Efisiensi: Kita tidak perlu mengukur ribuan orang — cukup gunakan data yang ada untuk memperkirakan bentuk kurva berdasarkan rata-rata (mean) dan simpangan baku (standard deviation).

Distribusi Normal

- Distribusi normal adalah salah satu jenis distribusi probabilitas kontinu.
- Setiap distribusi kontinu, termasuk distribusi normal diwakili oleh sebuah fungsi kepadatan probabilitas (PDF).
- Jadi, PDF dari distribusi normal adalah rumus matematis yang menggambarkan bentuk kurva lonceng khas distribusi normal.

Secara matematis, PDF dari distribusi normal dengan rata-rata μ dan simpangan baku σdituliskan sebagai:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

Artinya:

- $f(x)$memberi tahu kita seberapa padat nilai-nilai data di sekitar x.
- Luas total di bawah kurva PDF selalu = 1 (karena itu mewakili total probabilitas).

Keterangan:

- μ= rata-rata (mean)
- σ= simpangan baku (standard deviation)
- e= bilangan Euler (≈ 2.718)

Misalkan tinggi badan rata-rata (μ) = 5.7 ft, dan simpangan baku (σ) = 0.3 ft. Kita ingin tahu probabilitas relatif seseorang memiliki tinggi 6.0 ft.

$$f(6.0) = \frac{1}{\sqrt{2\pi(0.3)^2}} e^{-\frac{(6.0-5.7)^2}{2(0.3)^2}}$$

$$f(6.0) \approx \frac{1}{0.752} e^{-0.5}$$

$$f(6.0) \approx 1.33 \times 0.6065 = 0.806$$

Nilai $f(6.0)$ tidak menunjukkan probabilitas absolut, tetapi tingkat kepadatan relatif, makin besar nilainya, makin besar kemungkinan tinggi badan tersebut muncul.

Perlu diingat bahwa bentuk kurva adalah tinggi × lebar. Nilai PDF (f(x)) = tinggi gunung di titik itu. Total probabilitas = luas seluruh gunung (integral dari f(x)). Arti dari $f(6.0) = 0.806$ adalah kepadatan probabilitas (probability density) di titik 6.0 ft. Artinya, di sekitar tinggi 6.0 ft, data relatif padat atau lebih mungkin muncul dibandingkan titik lain dengan nilai f(x) lebih kecil.

Semakin besar nilai $f(x)$, semakin besar pula peluang nilai itu muncul dalam rentang kecil di sekitarnya. Misalnya, probabilitas orang memiliki tinggi antara 5.95–6.05 ft dapat dihitung dengan luas area di bawah kurva antara 5.95 dan 6.05:

$$P(5.95 < X < 6.05) = \int_{5.95}^{6.05} f(x)\, dx$$

Nah, hasil integral itulah yang merupakan probabilitas absolut, bukan nilai $f(6.0)$itu sendiri.

Bayangkan Anda menaburkan pasir di sepanjang garis tinggi badan.

- Di sekitar 5.7 ft, pasirnya menumpuk tinggi — artinya banyak orang di sekitar situ.
- Di 6.3 ft, pasirnya sedikit — artinya jarang ada orang setinggi itu.

Nilai $f(x)$menunjukkan tinggi tumpukan pasir pada titik x — bukan seberapa banyak total pasir di titik itu.

Dari hasil perhitungan tadi:

Tinggi (ft)	$f(x)$	Interpretasi
5.7(mean)	1.33	titik paling padat (banyak orang di sekitar sini)
6.0	0.806	lebih jarang dibanding rata-rata
6.3	0.325	semakin jarang
5.0	0.010	sangat jarang

Kita bisa melihat bahwa nilai $f(x)$ menurun ketika menjauh dari rata-rata, menandakan bahwa nilai ekstrem lebih jarang terjadi. Berikut adalah beberapa konsep penting yang kita pelajari:

- PDF (Probability Density Function) adalah Fungsi yang menggambarkan *kepadatan* peluang nilai-nilai kontinu.
- $f(x)$ adalah Tinggi kurva di titik x yang menunjukkan seberapa "padat" kemungkinan di sekitar nilai itu.
- PDF bukan probabilitas langsung, karena probabilitas tepat di satu titik = 0 (karena kontinu).
- Probabilitas menurut PDF adalah luas area di bawah kurva antara dua titik (misalnya $P(5.9 < X < 6.1)$).

" Fungsi kepadatan probabilitas (PDF) menggambarkan seberapa padat kemungkinan nilai suatu variabel acak kontinu muncul di setiap titik, bukan probabilitas munculnya nilai tunggal. Luas total di bawah kurva PDF selalu sama dengan 1, yang merepresentasikan keseluruhan probabilitas dari semua kemungkinan nilai".

BAGIAN 5
ALGORITMA PENTING YANG PERLU KAMU TAHU

5.1. Naïve Bayes

Naïve Bayes pertama kali digunakan pada tahun 1950-an di berbagai bentuk awalnya, dan hingga kini masih menjadi algoritma dasar yang sangat populer terutama untuk pemrosesan teks (text mining). Kesederhanaannya membuat algoritma ini menjadi tolok ukur (baseline) bagi banyak permasalahan klasifikasi berbasis teks, sebelum beralih ke model yang lebih kompleks seperti Logistic Regression atau Neural Network.

Konsep Naïve Bayes adalah mengalikan probabilitas-probabilitas antar fitur. Mengalikan probabilitas berarti mengasumsikan bahwa setiap fitur bersifat independen satu sama lain. Inilah alasan algoritma ini disebut naïve (naif) karena dalam kenyataannya, fitur-fitur dalam data sering kali saling bergantung. Namun menariknya, meskipun asumsi ini tidak realistis, Naïve Bayes sering kali memberikan hasil yang sangat baik, terutama pada data berukuran besar (large-scale data).

Motor dari algoritma ini adalah teorema bayes, dan untuk memahami ini kita perlu memahami Statistik Bayesian dan untuk memahami statistik Bayesian kita perlu memahami conditional probability. Conditional probability adalah probabilitas dari sebuah kejadian terjadi jika diberikan kejadian event yang lain, ini bisa dituliskan sebagai berikut:

$$P(Y|X) = \frac{P(X \; and \; Y)}{P(X)}$$

Teorema Bayes pada dasarnya menjawab pertanyaan : "Jika saya sudah memiliki dugaan awal (prior), lalu saya dapat informasi baru, bagaimana saya memperbaharui dugaan itu(posterior)". Dari pertanyaan ini maka muncul beberapa istilah yang perlu dipahami, antara lain:

1. **Priot Probability (Probabilitas Awal)**
    - Probabilitas yang kita tetapkan sebelum ada informasi tambahan
    - Disebut juga marginal probability, karena hanya melihat kejadian secara umum tanpa kondisi khusus
    - Contoh prior: peluang seseorang sakit flu di musim hujan mungkin 20%
2. **Posterior Probability (Probabilitas Sesudah Data Masuk)**
    - Setelah kita mendapat informasi baru (misalnya gejala: pilek dan demam), kita memperbaharui peluang orang itu benar-benar sakit flu
    - Posterior menghitung ulang probabilitas dengan mempertimbangkan data baru
    - Contoh posterior: setelah tahu orang itu demam, peluang sakit flu meningkat menjadi 70%
3. **Sequential Events (Peristiwa Berurutan)**
    - Teorema bayes bekerja dengan cara memperbaharui probabilitas secara bertahap
    - Setiap kali ada informasi baru, prior yang lama berubah menjadi posterior baru. Posterior ini bisa dipakai lagi jadi prior untuk informasi berikutnya

Misalkan pada teorema bayes ketika kita ingin mengetahui peluang suatu email apakah spam atau tidak maka dihitung sebagai berikut:

$$\mathrm{P(Spam|Fitur)} = \frac{\mathrm{P(Fitur|Spam)} \cdot \mathrm{P(Spam)}}{\mathrm{P(Fitur)}}$$

Tapi permasalahannya adalah ketika fiturnya banyak (misalkan ada ratusan kata dalam email), maka menghitung $P(Fitur|Spam)$ menjadi sangat rumit.

Naïve bayes menganggap semua fitur independen satu sama lain, misalkan kata diskon dalam email tidak ada hubungannya dengan kata tawaran padahal kenyataannya kedua kata tersebut saling berhubungan. Dengan asumsi ini, perhitungan menjadi jauh lebih mudah:

$$P(Fitur_1, \dots Fitur_n|Spam) \approx P(Fitur_1|Spam) \cdot P(Fitur_n|Spam)$$

Baik kita jabarkan contoh diatas. Bayangkan kita memiliki dua jenis pesan:

- Pesan normal (dari teman atau keluarga)
- Pesan spam (iklan atau penipuan)

Kita ingin agar komputer bisa otomatis mengenali mana yang spam dan mana yang bukan. Langkah awal yang dilakukan adalah:

- Mengumpulkan semua kata dari pesan normal dan pesan spam.
- Menghitung frekuensi kemunculan setiap kata di masing-masing jenis pesan.

Kata	Jumlah di Pesan Normal	Jumlah di Spam
dear	8	2
friend	5	2
lunch	3	0
money	1	3

Total kata di pesan normal = 17

Total kata di pesan spam = 7

Dari tabel di atas, kita bisa menghitung peluang kemunculan suatu kata pada masing-masing jenis pesan.

Contoh:

- Peluang munculnya kata dear di pesan normal

$$\mathrm{P}(\mathrm{dear}|\mathrm{normal}) = \frac{8}{17} = 0.47$$

- Peluang munculnya kata friend di pesan normal

$$\mathrm{P}(\mathrm{friend}|\mathrm{normal}) = \frac{5}{17} = 0.29$$

- Peluang munculnya kata money di pesan normal

$$\mathrm{P}(\mathrm{money}|\mathrm{normal}) = \frac{1}{17} = 0.06$$

- Peluang munculnya kata dear di spam

$$\mathrm{P}(\mathrm{dear}|\mathrm{spam}) = \frac{2}{7} = 0.29$$

- Peluang munculnya kata friend di spam

$$P(friend\ |spam) = \frac{2}{7} = 0.29$$

Nilai-nilai di atas disebut likelihood, yaitu peluang munculnya suatu kata jika diketahui jenis pesannya (normal atau spam). Catatan: Dalam konteks diskrit seperti kata, istilah probabilitas dan likelihood sering dipakai bergantian.

Sekarang, kita dapat pesan baru berbunyi:
"Dear Friend"
Kita ingin tahu apakah ini pesan normal atau spam.

- ✓ **Langkah 1:** Tentukan Prior Probability
  Kita mulai dengan menebak peluang awal (sebelum melihat kata apa pun):
  - Dari 12 pesan pelatihan, 8 adalah normal dan 4 adalah spam

  Jadi:
  - $P(normal) = \frac{8}{12} = 0.67$
  - $P(spam) = \frac{4}{12} = 0.33$
- ✓ **Langkah 2:** Hitung Posterior Score
  Gunakan Teorema Bayes (tanpa penyebut, cukup proporsionalitas):

$$Score(normal) \propto P(normal)\ x\ P(dear|normal)\ x\ P(friend\ |normal)$$
$$Score(normal) \propto 0.67\ x\ 0.47\ x\ 0.29 = 0.09$$

$$Score(spam) \propto P(spam)\ x\ P(dear|spam)\ x\ P(friend|spam)$$
$$Score(spam) \propto 0.33\ x\ 0.29\ x\ 0.29 = 0.01$$

Karena 0.09 > 0.01, maka pesan dear friend diklasifikasikan sebagai normal.

Sekarang kita coba pesan baru:
"Lunch money money money money"

Masalah muncul karena di data pelatihan, kata "lunch" tidak pernah muncul dalam spam. Sehingga:

$$P(lunch|spam) = 0$$

dan akibatnya hasil perhitungan jadi 0, karena berapa pun dikalikan 0 hasilnya 0. Artinya, pesan apapun yang mengandung kata lunch akan selalu dianggap normal, walaupun kata money muncul berkali-kali. Ini jelas tidak ideal.

Untuk menghindari nilai nol tersebut, kita gunakan Laplace Smoothing, yaitu dengan menambahkan satu hitungan (count) pada setiap kata dalam histogram.

Rumusnya :

$$P(kata \mid kelas) = \frac{jumlah\ kata + \alpha}{total\ kata + \alpha\ x\ jumlah\ unik\ kata}$$

Biasanya $\alpha = 1$.

Dengan cara ini, semua kata punya peluang minimal, sehingga tidak ada lagi yang bernilai 0.

Contoh:

$$P(lunch|spam) = \frac{0+1}{7+1x4} = 0.09$$

Setelah smoothing, kita hitung ulang skor:

- Normal message: nilainya tetap kecil.
- Spam message: sekarang memiliki nilai > 0.

Karena nilai spam lebih besar, maka pesan tersebut diklasifikasikan sebagai spam. Naive Bayes dianggap naive (naif) karena ia mengabaikan urutan kata dan konteks bahasa.

Contohnya:

“dear friend”

“friend dear”

Keduanya akan diberi skor yang sama persis, padahal secara bahasa berbeda. Model ini memperlakukan teks sebagai sekadar kumpulan kata acak (bag of words), tanpa memperhatikan struktur kalimat atau tata bahasa. Namun, meski sederhana, pendekatan ini sangat efektif untuk masalah seperti klasifikasi spam, analisis sentimen, dan topik teks lainnya.

” Naive Bayes adalah algoritma klasifikasi berbasis teorema Bayes yang mengasumsikan setiap fitur bersifat independen terhadap fitur lainnya. Meskipun sederhana, metode ini sering memberikan hasil yang efektif dan efisien terutama pada data berukuran besar seperti dalam analisis teks atau deteksi spam”.

5.2. Linear Regression

Linear Regression adalah salah satu metode paling sederhana dalam machine learning yang digunakan untuk memodelkan hubungan antara satu atau lebih variabel independen (prediktor) dengan sebuah variabel dependen (target). Inti dari linear regression adalah mencari garis lurus terbaik yang dapat menggambarkan hubungan data, sehingga garis tersebut bisa digunakan untuk melakukan prediksi. Misalnya, jika kita ingin memprediksi harga rumah berdasarkan ukuran rumah, linear regression akan mencoba menemukan garis yang menunjukkan hubungan antara ukuran rumah dan harga rumah.

Cara kerjanya dimulai dengan mengasumsikan bahwa hubungan antara variabel dapat dijelaskan dalam bentuk persamaan linear $y = a + bx$ dimana y adalah variabel target, x adalah variabel input, a adalah intercept, dan b adalah slope (kemiringan garis). Model kemudian menghitung nilai a dan b sedemikian rupa agar garis yang terbentuk meminimalkan selisih antara prediksi model dan data sebenarnya.

Selisih ini diukur dengan metode yang disebut *least squares*, yaitu menjumlahkan kuadrat dari semua error (perbedaan antara nilai aktual dan prediksi). Setelah model terlatih, kita dapat memasukkan nilai baru ke dalam persamaan linear tersebut untuk memperoleh prediksi. Linear regression banyak digunakan karena sederhana dan mudah dipahami, serta cocok untuk kasus-kasus prediksi dengan hubungan data yang cukup linear.

Kelebihan utama regresi adalah:

- Cepat dilatih (computationally efficient).
- Mudah dijelaskan kepada orang nonteknis.
- Mudah diimplementasikan dalam berbagai bahasa pemrograman.

Karena kesederhanaan dan keterbacaannya, regresi sering digunakan sebagai model pembanding (baseline model) sebelum mencoba algoritma yang lebih kompleks seperti decision tree, random forest, atau neural network. Selain itu, regresi juga sering digunakan untuk:

- Mengidentifikasi fitur-fitur penting dalam suatu permasalahan.
- Mencoba kombinasi fitur baru.
- Memberikan wawasan awal untuk rekayasa fitur (feature engineering).
- Regresi linear bekerja dengan cara menggabungkan setiap fitur numerik menggunakan penjumlahan berbobot. Tiap fitur $X_i$dikalikan dengan koefisien bobot β_i, lalu dijumlahkan bersama sebuah konstanta bias α(juga disebut intercept). Bias ini berfungsi sebagai titik awal prediksi, yaitu nilai keluaran ketika semua fitur bernilai nol.

  Secara matematis, persamaan regresi linear dapat ditulis sebagai:

$$Y = \beta X + \alpha$$

  Keterangan:

  - Y : vektor nilai target (misalnya harga rumah, jumlah penjualan, atau pendapatan)
  - X : matriks fitur yang berisi nilai-nilai numerik dari variabel masukan

- β : vektor koefisien (bobot) yang menunjukkan seberapa kuat pengaruh masing-masing fitur terhadap target
- α : bias (konstanta), yaitu nilai prediksi ketika semua fitur bernilai nol.

Apabila model hanya menggunakan satu fitur (x), maka model disebut regresi linear sederhana, dengan persamaan:

$$y = \beta x + \alpha$$

Contoh Kasus: Berat dan Panjang Tubuh Tikus

Tabel 12
Berat dan Panjang Tubuh Tikus

Tikus	Berat (gram)	Panjang (cm)
1	10	15
2	15	18
3	20	20
4	25	24
5	30	27

Kita ingin mengetahui apakah ada hubungan linier antara berat dan panjang tubuh? Jika kita plot titik-titik tersebut dalam grafik, kita akan melihat kecenderungan bahwa semakin berat seekor tikus, semakin panjang tubuhnya. Kemudian kita tarik garis regresi.

Garis regresi tidak ditarik sembarangan. Kita ingin garis yang paling mendekati semua titik data. Untuk menilai seberapa "dekat", kita menghitung jarak vertikal antara titik data dan garis prediksi, disebut residual (selisih prediksi).

$$\text{residual}_i = Y_i - \hat{Y}_i$$

Kita lalu mengkuadratkan setiap residual (agar positif), dan menjumlahkannya:

$$SS_{\text{fit}} = \sum (Y_i - \hat{Y}_i)^2$$

Tujuannya adalah mencari garis dengan nilai SS_{fit} paling kecil, inilah yang disebut metode Least Squares (*Kuadrat Terkecil*). Sehingga ilustrasi prosesnya adalah sebagai berikut:

1. Tarik garis acak di antara titik-titik data.
2. Hitung residual (selisih vertikal antara titik dan garis).
3. Kuadratkan setiap residual dan jumlahkan.
4. Putar sedikit garisnya, ulangi langkah 2–3.
5. Ulangi sampai menemukan rotasi (kemiringan) dengan jumlah kuadrat residual terkecil.

Hasil akhirnya adalah garis terbaik atau *best fit line*.

Kita sudah mengetahui bahwa persamaan linear regression adalah sebagai berikut:

$$\hat{Y} = a + bX$$

Misalnya setelah dihitung didapat:

$$\hat{Y} = 10 + 0.5X$$

Artinya: setiap kenaikan 1 gram berat, panjang tubuh diprediksi bertambah 0.5 cm.

Maknanya:

- Koefisien 0.5 di depan X
  Artinya: *setiap penambahan 1 satuan pada X (misalnya 1 gram)* akan menyebabkan $\hat{Y}$ bertambah 0.5 pada tinggi (misalnya 0.5 cm). Jadi benar bahwa hubungan antara X dan Y adalah linier dan proporsional.

- Nilai 10 (intercept)

  Artinya: ketika $X = 0$, maka nilai prediksi $\hat{Y} = 10$. Dengan kata lain, tanpa adanya X sama sekali, berat awal (atau nilai dasar Y) sudah 10.

Setelah kita mendapatkan garis terbaik, pertanyaan berikutnya adalah "Seberapa baik garis ini menjelaskan data?". Di sinilah kita menggunakan koefisien determinasi, yang dilambangkan dengan R^2.

Koefisien determinasi, yang biasa dilambangkan dengan R^2(dibaca: "R kuadrat") adalah ukuran seberapa baik model regresi menjelaskan variasi data yang diamati. Secara sederhana R^2 menunjukkan seberapa besar proporsi variasi dalam variabel dependen (y) yang dapat dijelaskan oleh variabel independen (x) melalui model regresi.

Interpretasi Nilai R^2

- $R^2 = 1 \rightarrow$ Model sempurna, semua data tepat di garis regresi (tidak ada error).
- $R^2 = 0 \rightarrow$ Model tidak menjelaskan apapun, prediksi sama buruknya dengan menggunakan rata-rata.
- $0 < R^2 < 1 \rightarrow$ Model menjelaskan sebagian variasi data (semakin tinggi semakin baik).
- Dalam kasus langka, R^2bisa negatif jika model sangat buruk (misalnya karena overfitting atau model non-linier dipaksa linier).

Fungsi R^2

1. Mengukur kualitas model regresi — seberapa baik model mampu menjelaskan data.
2. Membandingkan beberapa model — model dengan R^2lebih tinggi umumnya lebih baik dalam menjelaskan data (meskipun bukan satu-satunya kriteria).
3. Evaluasi kekuatan hubungan linier antara variabel independen dan dependen.

Bayangkan kamu mencoba memprediksi tinggi badan seseorang dari berat badannya.

- Jika $R^2 = 0.8$, artinya 80% variasi tinggi badan bisa dijelaskan oleh berat badan melalui garis regresi.
- Sisanya 20% dijelaskan oleh faktor lain (misalnya genetik, usia, nutrisi, dll).

Rumus R^2

$$R^2 = \frac{\text{Variasi di sekitar mean} - \text{Variasi di sekitar garis fit}}{\text{Variasi di sekitar mean}}$$

atau dalam bentuk kuadrat jumlah:

$$R^2 = \frac{SS_{\text{mean}} - SS_{\text{fit}}}{SS_{\text{mean}}}$$

di mana:

- $SS_{\text{mean}} = \sum(Y_i - \bar{Y})^2$
- $SS_{\text{fit}} = \sum(Y_i - \hat{Y}_i)^2$

Misal hasilnya:

$$SS_{\text{mean}} = 100, \qquad SS_{\text{fit}} = 40$$

$$R^2 = \frac{100 - 40}{100} = 0.6$$

REGRESI LINEAR BERGANDA (MULTIPLE LINEAR REGRESSION)

Terkadang, satu variabel tidak cukup untuk menjelaskan variasi data. Misalnya, panjang tubuh tikus mungkin tidak hanya dipengaruhi oleh berat, tetapi juga oleh panjang ekor.

$$Y = a + b_1X_1 + b_2X_2$$

- X_1: berat
- X_2: panjang ekor

Metode *least squares* masih digunakan, hanya saja kita kini mencari bidang datar (plane), bukan garis.

Contoh:

$$\text{Body Length} = 0.3 + 0.7 \times \text{Weight} + 0.5 \times \text{Tail Length}$$

Dari persamaan ini:

- setiap tambahan 1 gram berat meningkatkan panjang tubuh sekitar 0.7 cm
- setiap tambahan 1 cm panjang ekor menambah panjang tubuh sekitar 0.5 cm

MASALAH OVERFITTING DAN ADJUSTED R^2

Semakin banyak variabel kita tambahkan, semakin besar kemungkinan R^2naik bahkan jika variabelnya tidak relevan

Contoh ekstrem:

$$\text{Size} = 0.3 + \text{Weight} + \text{Favorite Color} + \text{Astrological Sign}$$

Meskipun warna favorit jelas tidak berhubungan dengan ukuran tikus, model masih bisa menghasilkan R^2 yang sedikit lebih tinggi karena kebetulan pola acak. Untuk mengatasi hal ini, digunakan Adjusted R^2, yang mengoreksi nilai R^2 terhadap jumlah variabel.

Jika penambahan variabel tidak memberi manfaat signifikan, nilai Adjusted R^2justru akan menurun.

Untuk mengatasi hal ini, digunakan Adjusted R², yang menghukum penambahan variabel tidak relevan.
Rumusnya:

$$\text{Adjusted } R^2 = 1 - (\frac{(1 - R^2)(n - 1)}{n - p - 1})$$

dengan:

- n: jumlah data (sampel)
- p: jumlah variabel independen (prediktor)

Misalkan kita ingin memprediksi ukuran tikus (Size) berdasarkan beberapa faktor. Kita punya 100 data (n = 100) dan akan bandingkan tiga model berbeda:

- Model 1, jumlah variabel 1 yaitu hanya weight
- Model 2, jumlah variabel 2 yaitu weight dan tail length dan ini masih relevant
- Model 3, jumlah variabel 4 yaitu dari model 2 (weight dan tail length) ditambah favorite color dan astrological sign yang ini tidak relevan.

Misal hasil regresinya seperti ini:

Model	R²
Model 1	0.80
Model 2	0.88
Model 3	0.89

Terlihat R^2 selalu naik, tapi belum tentu itu berarti modelnya lebih baik secara *nyata* karena kita mengetahui bahwa ada dua variabel di model 3 yang tidak relevan.

Sekarang kita hitung menggunakan adjusted R^2 dengan rumus:

$$\text{Adjusted } R^2 = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1}$$

✓ **Model 1**

$R^2 = 0.80, n = 100, p = 1$

$$\text{Adjusted } R^2 = 1 - \frac{(1 - 0.80)(99)}{98} = 1 - \frac{0.20 \times 99}{98} = 1 - 0.202 = 0.798$$

Adjusted R^2 = 0.798

✓ **Model 2**

$R^2 = 0.88, n = 100, p = 2$

$$\text{Adjusted } R^2 = 1 - \frac{(1 - 0.88)(99)}{97} = 1 - \frac{0.12 \times 99}{97} = 1 - 0.122 = 0.878$$

Adjusted R^2 = 0.878

Naik signifikan → berarti variabel "Tail Length" benar-benar membantu model.

✓ **Model 3**

$R^2 = 0.89, n = 100, p = 4$

$$\text{Adjusted } R^2 = 1 - \frac{(1 - 0.89)(99)}{95} = 1 - \frac{0.11 \times 99}{95} = 1 - 0.115 = 0.885$$

Adjusted R^2 = 0.885

Naik hanya 0.007 poin, jauh lebih kecil dari kenaikan sebelumnya artinya tambahan "Color" dan "Zodiak" tidak banyak membantu, bahkan bisa dianggap *noise*.

Interpretasinya adalah:

- Pada model 1, variabel weight menghasilkan adjusted R^2 sebesar 0.798 maka bisa dianggap sebagai model dasar yang cukup baik
- Model 2 dengan penambahan variabel tail length dari model 1 menghasilkan adjusted R^2 sebesar 0.878 maka dilihat sebagai peningkatan nyata yang menyatakan variabel yang ditambahkan relevan
- Model 3 dengan penambahan favorite color dan astrological sign dari model 2 menghasilkan adjusted R^2 sebesar 0.885 maka peningkatannya sangat kecil yang mengindikasikan noise.

Meskipun mudah dipahami dan efisien, regresi linear tidak bisa menyelesaikan semua jenis masalah. Hal ini karena tidak semua hubungan antarvariabel bersifat linier.

Bayangkan Anda memiliki data yang membentuk pola melengkung (non-linear). Jika Anda mencoba menggambarnya dengan garis lurus, maka sebagian titik data akan berada di atas garis (model underestimate) dan sebagian lainnya di bawah garis (model overestimate). Dalam kondisi seperti ini, model akan cenderung gagal menangkap pola sebenarnya dari data. Akibatnya, hasil prediksi akan selalu menyimpang secara sistematis dan di sinilah muncul apa yang disebut bias error (kesalahan bias).

Bias error menggambarkan ketidakmampuan model untuk merepresentasikan hubungan yang kompleks antara fitur dan target. Model dengan bias tinggi (seperti regresi linear sederhana) terlalu menyederhanakan kenyataan, sehingga tidak mampu menangkap pola data yang lebih rumit. Untuk mengatasi keterbatasan model linier, para peneliti mengembangkan algoritma yang lebih fleksibel dan adaptif, seperti decision tree, random forest, atau neural network.Model-model ini memang lebih sulit dijelaskan secara matematis, tetapi memiliki kemampuan luar biasa untuk mendekati bentuk fungsi apa pun baik linier maupun non-linier. Dengan kata lain, model yang lebih kompleks mampu mendekati fungsi target (target function) yang sebenarnya jauh lebih rumit dibanding apa yang bisa dilakukan oleh model linier.

" Linear regression adalah metode statistik yang digunakan untuk memodelkan hubungan linear antara variabel input (independen) dan output (dependen). Tujuannya adalah menemukan garis terbaik yang meminimalkan selisih antara nilai prediksi dan nilai aktual, sehingga dapat digunakan untuk memprediksi tren atau nilai di masa depan".

5.3. Logistic Regression

Regresi logistik adalah salah satu algoritma fundamental dalam *machine learning* yang digunakan untuk klasifikasi biner, yaitu ketika output yang ingin diprediksi hanya memiliki dua kemungkinan, seperti:

- Apakah email ini *spam* atau bukan?
- Apakah pasien ini memiliki penyakit tertentu atau tidak?
- Apakah pelanggan akan membeli produk atau tidak?

Berbeda dengan *linear regression* yang menghasilkan keluaran berupa nilai kontinu (misalnya prediksi harga rumah), *logistic regression* menghasilkan probabilitas bahwa suatu contoh data termasuk ke dalam kelas tertentu.

Tujuan utama dari regresi logistik bukanlah memprediksi nilai kontinu, melainkan memetakan kombinasi linier dari fitur menjadi probabilitas antara 0 dan 1.

Regresi logistik merupakan jembatan penting antara *machine learning* klasik dan model probabilistik modern. Dengan pemahaman yang baik tentang teori matematis di baliknya, algoritma ini menjadi fondasi bagi berbagai model canggih seperti:

- Softmax Regression
- Neural Network dengan output sigmoid
- Deep Learning Binary Classifier

Meskipun sederhana, regresi logistik tetap relevan dan banyak digunakan karena interpretabilitas dan kestabilannya.

Cara kerjanya cukup sederhana, misalkan kita ingin memprediksi apakah pinjaman seseorang akan disetujui atau tidak. Dimana ditolak bernilai 0 sedangkan diterima bernilai 1, kemudian kita tentukan batas threshold misalkan 0.5 yang menjadi batas apakah ajuan pinjaman tersebut diterima, jika nilainya dibawah treshold maka ditolak dan jika nilainya diatas threshold maka diterima.

Logistic Regression menggunakan rumus fungsi sigmoid yang akan membentuk garis seperti huruf "S". Fungsi ini mengubah angka apapun menjadi nilai antara 0 dan 1. Contoh:

- Input -10 → Output : 0,00004 (sangat dekat dengan 0)
- Input 0 → Output 0.5
- Input +10 → Output : 0,99995 (sangat dekat dengan 1)

Secara matematis tahapan logistic regression adalah sebagai berikut:

- ✓ **Langkah 1:** Model logistic regression menerima fitur-fitur (x) seperti usia, pendapatan, jam tidur, dll, dan menggabungkannya secara linear:

$$z = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

w_0 : bias (angka dasar)

$w_1, w_2, \dots, w_n$: bobot tiap fitur

$x_1, x_2, \dots, x_n$: nilai fitur

✓ **Langkah 2:** Mengubah hasilnya menjadi probabilitas

Hasil perhitungan z tadi bisa bernilai negatif atau sangat besar. Agar bisa ditafsirkan sebagai peluang (antara 0 dan 1), logistic regression menggunakan fungsi sigmoid:

$$p = \frac{1}{1 + e^{-z}}$$

Fungsi ini akan mengubah semua nilai menjadi di antara 0 sampai

Contoh:

- Jika $z = 0 \rightarrow p = 0.5$
- Jika z sangat besar $\rightarrow p$ mendekati 1
- Jika z sangat kecil $\rightarrow p$ mendekati 0

✓ **Langkah 3 :** Interpretasi Hasil

Setelah model menghitung peluang p, kita tinggal menentukan batas (*threshold*) untuk membuat keputusan. Biasanya digunakan batas 0.5:

- Jika $p > 0.5$: Prediksi positif (1)
- Jika $p < 0.5$: Prediksi negatif (0)

Tapi dalam beberapa kasus (misalnya deteksi penyakit), batas bisa diubah, Misal: gunakan 0.3 agar lebih sensitif mendeteksi pasien berisiko.

Contoh kasus:

Misalkan terdapat data pasien sebagai berikut:

Tabel 13
Data Pasien

Usia	Tekanan Darah	Sakit (1=Ya,0=Tidak)
25	120	0
45	160	1
35	140	1

Model logistic regression akan mencari hubungan antara usia, tekanan darah, dan kemungkinan sakit. Setelah dilatih kita bisa masukan data baru, misalkan Usia = 40 ; tekanan darah = 150 → model hasilkan peluang 0.78. Artinya pasien ini 78% kemungkinan sakit.

Berikutnya agar model bisa belajar, logistic regression menggunakan log loss (juga disebut binary cross-entropy):

$$L = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$$

dimana:

- L adalah nilai kerugian rata-rata seluruh data (semakin kecil, maka semakin baik)
- m jumlah total data pelatihan
- y_i label sbenarnya dari data ke-i (0 atau 1)
- p_i Probabilitas hasil prediksi model untuk data ke-i (antara 0 dan 1)
- log adalah Logaritma natural (basis e, biasa ditulis ln)
- $\sum$ penjumlahan seluruh data dari $i = 1$ sampai m

Model logistic regression menghasilkan probabilitas p_i untuk setiap data. Kita ingin memberi hukuman besar bisa model salah dan hukuman kecil bisa model benar. Untuk memahami cara kerja mari kita lihat komponen yang ada didalam kurung pada persamaan diatas.

$$y_i \log(p_i) + (1 - y_i)\log(1 - p_i)$$

Kita coba baca persamaan diatas dalam dua kasus:

- Jika data sebenarnya positif ($y_i = 1$)

  Bagian kiri menjadi : $1 \cdot \log(p_i) + 0 \cdot \log(1 - p_i) = \log(p_i)$

  Maka:

  Jika model yakin $p_i = 1 \rightarrow \log(1) = 0$ ini berarti tidak ada kerugian, sebaliknya jika model salah maka menyebabkan p_i kecil → $\log(p_i)$ menjadi negatif besar → kerugian besar.

- Jika data sebenarnya negatif ($y_i = 0$):

  Bagian kiri menjadi $0 \cdot \log(p_i) + 1 \cdot \log(1 - p_i) = \log(1 - p_i)$

  Maka:

  Jika model yakin $p_i = 0 \rightarrow \log(1) = 0$, sebaliknya jika model salah maka kemungkinan p_i mendekati 1 yang menyebabkan $\log(1 - p_i)$ menjadi negatif besar yang berujung kepada kerugian besar.

Model tidak hanya memprediksi satu data, tetapi banyak data berjumlah m. Oleh karena itu kemudian kita ambil rata-rata kerugian dari seluruh data.

Contoh perhitungan, misalkan kita memiliki 3 data sebagai berikut:

Tabel 14
Contoh Data

Data	Label sebenarnya y_i	Prediksi Model p_i
1	1	0.9
2	0	0.2
3	1	0.3

Maka:

$$L = -\frac{1}{3}\left[(1 \cdot (\log 0.9)) + (1-1)\log(0.1) + (0 \cdot (\log 0.2)) + (1-0)\log(0.8) + (1 \cdot \log(0.3)) + (1-1)\log(0.7)\right]$$

$$L = -\frac{1}{3}\left[\log(0.9) + \log(0.8) + \log(0.3)\right]$$

$$L \approx \left[(-0.105) + (-0.223) + (-1.204)\right]$$

$$L = 0.51$$

Artinya rata-rata kesalahan model adalah 0.51, dimana semakin kecil nilai ini maka semakin baik modelnya.

" Logistic regression adalah metode klasifikasi yang memodelkan probabilitas suatu kejadian dengan menggunakan fungsi logistik (sigmoid) untuk menghasilkan output antara 0 dan 1. Teknik ini banyak digunakan untuk memecahkan masalah klasifikasi biner, seperti menentukan apakah suatu kondisi "ya" atau "tidak".

5.4. Decision Tree

Bayangkan kamu ingin memutuskan apakah akan pergi piknik besok atau tidak. Kamu akan mempertimbangkan beberapa hal, kan?

Cuaca: Cerah atau hujan?
Suhu: Hangat atau dingin?
Teman: Ada yang ikut atau tidak?

Kamu membuat serangkaian pertanyaan dalam pikiranmu, dan setiap jawaban membawamu ke pertanyaan berikutnya, sampai akhirnya kamu punya keputusan: "Ya, piknik!" atau "Tidak, di rumah saja."

Nah, Decision Tree itu persis seperti proses berpikirmu itu! Ini adalah sebuah model yang menggunakan struktur seperti pohon untuk membuat keputusan atau prediksi berdasarkan serangkaian aturan. Setiap "cabang" atau "node" di pohon adalah sebuah pertanyaan tentang data, dan "daun" (node terakhir) adalah keputusan atau hasil prediksinya.

Gambar 26
Ilustrasi Decision Tree

Pada gambar diatas terdapat beberapa komponen penting pada Decision Tree:

- Node Akar (Root Node): Pertanyaan pertama (misalnya, "Prakiraan Cuaca?"). Ini adalah titik awal.
- Cabang (Branches): Jawaban dari pertanyaan (misalnya, "Cerah" atau "Hujan").
- Node Keputusan (Decision Nodes): Pertanyaan-pertanyaan selanjutnya (misalnya, "Suhu Udara?").
- Node Daun/Terminal (Leaf Nodes): Hasil akhir atau keputusan (misalnya, "YA, PIKNIK!" atau "TIDAK, di Rumah Saja").

Decision Tree adalah salah satu algoritma machine learning yang bekerja dengan membagi data ke dalam cabang-cabang keputusan, sehingga hasil akhirnya berbentuk struktur seperti pohon. Pohon ini dimulai dari sebuah akar (root), lalu bercabang menjadi simpul (node) yang berisi pertanyaan atau kondisi tertentu, hingga akhirnya berakhir pada daun (leaf) yang menunjukkan hasil prediksi atau klasifikasi. Cara kerjanya sederhana, dimana algoritma akan mencari atribut atau variabel yang paling baik untuk memisahkan data berdasarkan suatu ukuran, misalnya *Gini Impurity* atau *Information Gain*. Proses ini dilakukan secara berulang (rekursif) pada setiap cabang hingga data benar-benar terpisah ke dalam kelompok-kelompok yang homogen atau hingga mencapai batas tertentu.

Sebagai contoh, jika kita ingin mengklasifikasikan apakah seseorang membeli sebuah produk atau tidak, decision tree dapat memulai dengan pertanyaan seperti “Apakah usia > 30 tahun?”, lalu

bercabang lagi dengan pertanyaan "Apakah pendapatan tinggi?" hingga sampai ke keputusan "Membeli" atau "Tidak membeli." Keunggulan utama decision tree adalah mudah dipahami karena hasilnya menyerupai aturan logika yang jelas. Selain itu, model ini bisa menangani data numerik maupun kategorikal.

Pohon keputusan bekerja dengan cara membagi data menjadi beberapa bagian (partisi) berdasarkan fitur (fitur = kolom data) yang paling membantu membedakan antara kelas target. Setiap langkah pemisahan (split) berusaha memperbaiki kemampuan prediksi model dengan membuat data di setiap partisi menjadi semakin "murni"

Secara intuitif, proses ini seperti menelusuri kembali aturan yang membentuk suatu keputusan. Dari sejumlah contoh data, algoritma mencoba menemukan pola umum berupa aturan yang bisa menjelaskan bagaimana sebuah keputusan dibuat. Pendekatan ini mirip dengan cara berpikir manusia dalam inferensi terbalik yaitu menelusuri sebab dari akibat yang diamati. Algoritma pohon keputusan bersifat greedy. Artinya, pada setiap langkah pemisahan, algoritma selalu memilih keputusan yang memberikan hasil terbaik saat itu juga, tanpa memikirkan apakah keputusan tersebut akan optimal di langkah-langkah berikutnya.

Inti dari membangun Decision Tree adalah menemukan "pertanyaan" atau "fitur" mana yang paling efektif membagi data menjadi kelompok-kelompok yang lebih murni (homogen) dalam hal keputusan akhir. Misalnya, jika semua orang yang cuacanya cerah selalu piknik, maka "Cuaca" adalah pertanyaan yang sangat baik

untuk memulai, karena langsung mengelompokkan data dengan jelas. Untuk mencari pertanyaan terbaik ini, ada konsep matematika yang digunakan, yaitu Impurity (Ketidakmurnian).

Bayangkan kamu punya sekumpulan buah. Jika semuanya apel, kumpulan itu "murni." Jika ada apel, jeruk, dan pisang, kumpulan itu "tidak murni." Dalam Decision Tree, "murni" berarti semua data di kelompok itu punya keputusan akhir yang sama (misalnya, semua "YA, PIKNIK!" atau semua "TIDAK, di Rumah Saja").

Decision Tree ingin meminimalkan impurity. Setiap kali Decision Tree membuat "cabang" atau "pertanyaan," tujuannya adalah membuat kelompok-kelompok baru yang lebih murni. Ada beberapa cara untuk mengukur impurity, yang paling umum adalah:

1. Gini Impurity (Impurity Gini)
2. Entropy (Entropi)

Mari kita lihat Gini Impurity karena lebih sederhana untuk dijelaskan. Secara sederhana, Gini Impurity mengukur seberapa sering kita salah memprediksi label (keputusan) jika kita memilih label secara acak dari kumpulan data.

- Jika Gini Impurity = 0, artinya kumpulan itu murni sempurna. Semua data di sana punya label yang sama.
- Jika Gini Impurity = 0.5 (untuk dua kelas), artinya kumpulan itu paling tidak murni. Jumlah label di sana sama banyak (misalnya, 50% "YA" dan 50% "TIDAK").

Rumus Gini Impurity:

$$Gini = 1 - \sum(p_i)^2$$

dimana:

- p_i adalah proporsi (bagian) dari kelas i dalam kumpulan data tersebut
- $\sum$ (sigma) berarti jumlahkan semua

Contoh Sederhana Gini Impurity:
Misalkan kita punya 10 orang di suatu kelompok:

- 7 orang memutuskan "YA, Piknik!"
- 3 orang memutuskan "TIDAK, di Rumah Saja"

Maka:

- p_YA = 7/10 = 0.7
- p_TIDAK = 3/10 = 0.3

Gini = $1 - ((0.7)^2 + (0.3)^2)$
= $1 - (0.49 + 0.09)$
= $1 - (0.58)$
= $1 - (0.49 + 0.09)$
= 0.42

Angka 0.42 menunjukkan bahwa kelompok ini belum sepenuhnya murni.

Decision Tree akan mencoba semua kemungkinan "pertanyaan" (fitur) dan menghitung Gini Impurity sebelum dan sesudah pertanyaan tersebut membagi data. Information Gain (Perolehan Informasi) adalah seberapa banyak impurity yang berkurang setelah membagi data dengan suatu pertanyaan.

Information Gain = Gini Impurity (Sebelum Pembagian) - Gini Impurity (Setelah Pembagian, rata-rata tertimbang)

Tujuannya Decision Tree adalah memilih pertanyaan yang memberikan Information Gain terbesar, karena itu berarti pertanyaan tersebut paling efektif dalam membuat kelompok-kelompok yang lebih murni dan mendekati keputusan akhir.

Kelebihan:

1. Mudah Dipahami dan Diinterpretasikan: Mirip cara manusia berpikir, jadi mudah dijelaskan.
2. Tidak Memerlukan Banyak Persiapan Data: Tidak perlu normalisasi data seperti model lain.
3. Bisa Menangani Data Kategorikal dan Numerik: Bisa memproses cuaca (kategorikal) dan suhu (numerik) secara bersamaan.
4. Fleksibel: Bisa digunakan untuk masalah klasifikasi (Ya/Tidak) maupun regresi (memprediksi angka).

Kekurangan:

1. Cenderung Overfitting: Bisa terlalu "hapal" data latihan sehingga kurang baik saat bertemu data baru. Ini seperti kamu terlalu menghapal rute piknikmu yang biasa, tapi bingung kalau ada jalan ditutup.
2. Sensitif Terhadap Perubahan Kecil: Perubahan kecil pada data bisa menghasilkan struktur pohon yang sangat berbeda.
3. Bias Terhadap Fitur Dominan: Bisa cenderung memilih fitur yang memiliki banyak level atau kategori.

Contoh Penerapan di Dunia Nyata

1. Medis: Mendiagnosis penyakit berdasarkan gejala pasien.
2. Perbankan: Menentukan apakah seseorang layak mendapatkan pinjaman.
3. Pemasaran: Mengidentifikasi pelanggan yang kemungkinan besar akan membeli produk tertentu.
4. Manajemen Risiko: Memprediksi risiko gagal bayar.

Sebagaimana yang diketahui bahwa decision tree rawan mengalami *overfitting* jika pohonnya terlalu dalam, sehingga sering digunakan bersama teknik lain seperti *random forest* untuk meningkatkan performa.

" Decision tree adalah algoritma pembelajaran yang menggunakan struktur pohon untuk memetakan keputusan berdasarkan fitur-fitur dalam data. Metode ini mudah dipahami dan diinterpretasikan karena meniru cara manusia mengambil keputusan melalui serangkaian pertanyaan bercabang".

5.5. Random Forest

Random Forest adalah pengembangan dari decision tree yang berusaha mengatasi kelemahan utamanya, yaitu *overfitting*. Jika decision tree hanya membangun satu pohon keputusan, maka random forest membangun banyak pohon sekaligus (disebut "hutan"), lalu menggabungkan hasilnya untuk mendapatkan prediksi yang lebih stabil dan akurat. Proses ini bekerja dengan prinsip *ensemble learning*, yaitu mengombinasikan banyak model sederhana agar menghasilkan model yang lebih kuat.

Cara kerjanya adalah: pertama, algoritma memilih sampel data secara acak (*bootstrap sampling*) untuk membangun masing-masing pohon. Kedua, pada setiap percabangan pohon, algoritma juga memilih subset fitur secara acak, sehingga setiap pohon akan sedikit berbeda dalam cara memisahkan data. Setelah semua pohon terbentuk, random forest akan menggabungkan hasil prediksi: untuk masalah klasifikasi, prediksi diambil berdasarkan "suara terbanyak" (*majority voting*), sedangkan untuk regresi digunakan rata-rata hasil dari semua pohon.

Hasilnya adalah kumpulan pohon yang *beragam* (diverse trees). Karena setiap pohon "melihat" data dan fitur yang berbeda, kesalahan mereka tidak saling berkorelasi kuat — dan rata-rata prediksi dari banyak pohon akan lebih stabil dan akurat.

Keunggulan random forest adalah kemampuannya menghasilkan prediksi yang lebih akurat, lebih tahan terhadap *overfitting*, dan dapat menangani data dengan dimensi tinggi atau

fitur yang kompleks. Namun, kekurangannya adalah model ini lebih sulit diinterpretasikan dibanding decision tree tunggal, serta membutuhkan sumber daya komputasi lebih banyak karena jumlah pohon yang besar.

Secara konseptual, random forest terdiri atas:

- n buah pohon keputusan (biasanya disebut $T_1, T_2, \dots, T_n$)
- Masing-masing pohon dilatih pada sebuah acak dan subset acak fitur
- Setiap pohon memberikan hasil prediksi:
  - Untuk klasifikasi, hasilnya berupa kelas mayoritas (majority voting)
  - Untuk regresi, hasilnya berupa rata-rata prediksi dari seluruh pohon

Tahapan detailnya adalah sebagai berikut:

✓ **Langkah 1:** Bootstrapping (Sampling Data)

Dari data asli berukuran N, Random Forest membuat beberapa subset acak dengan pengambilan dengan pengembalian (sampling with replacement). Setiap subset ini digunakan untuk melatih satu pohon.

✓ **Langkah 2:** Pemilihan Fitur Acak

Ketika pohon membelah data pada setiap simpul (node), hanya sebagian kecil fitur yang dipilih secara acak untuk dipertimbangkan. Tujuannya adalah agar setiap pohon menjadi unik.

✓ **Langkah 3:** Pembuatan Banyak Pohon

Setiap pohon keputusan dilatih pada subset datanya sendiri dan fitur-fitur acak yang berbeda.

✓ **Langkah 4:** Voting atau Averaging

Untuk klasifikasi: hasil akhir diambil dari mayoritas suara seluruh pohon.

$$\hat{y} = mode\{h_1(x), h_2(x), \dots, h_M(x)\}$$

dimana :

$\hat{y}$ adalah prediksi akhir (hasil gabungan dari semua pohon)

$h_1(x), h_2(x), \dots, h_M(x)$ adalah prediksi dari masing-masing pohon (ada M pohon total)

mode{…} adalah nilai yang paling sering muncul (mayoritas suara)

Mari kita bahas sebuah contoh, katakanlah kita memiliki 5 pohon keputusan di hutan kita. Kita ingin menebak apakah seseorang akan membeli mobil (Ya) atau tidak (Tidak).

Tabel 15
Hasil Random Forest

Pohon	Prediksi
$\boldsymbol{h_1(x)}$	Ya
$\boldsymbol{h_2(x)}$	Tidak
$\boldsymbol{h_3(x)}$	Ya
$\boldsymbol{h_4(x)}$	Ya
$\boldsymbol{h_5(x)}$	Tidak

Maka daftar prediksinya = {Ya, Tidak, Ya, Ya, Tidak}

Nilai yang paling sering muncul = "Ya"

Itulah arti dari $mode\{h_1(x), h_2(x), \dots, h_M(x)\}$. Jadi hasil akhirnya :

$$\hat{y} = \text{Ya}$$

Sedangkan untuk regresi, hasil akhir diambil dari rata-rata seluruh prediksi pohon.

$$\hat{y} = \frac{1}{M}\sum_{m=1}^{M} h_m(x)$$

dimana:

- $h_m(x)$ adalah prediksi dari pohon ke-m
- M adalah jumlah total pohon dalam hutan

Jika disimpulkan, maka secara umum random forest adalah bentuk ensemble learning:

$$H(x) = \frac{1}{M}\sum_{m=1}^{M} h_m(x)$$

dimana $H(x)$ adalah model gabungan dari M pohon

" Random forest adalah metode ensemble learning yang menggabungkan banyak decision tree untuk meningkatkan akurasi dan mengurangi overfitting. Pendekatan ini bekerja dengan mengambil rata-rata atau voting dari prediksi tiap pohon, sehingga memberikan hasil yang lebih stabil dan andal dibandingkan single". decision tree".

5.6. Ridge Regression

Saat data kita sederhana, linear regression berjalan baik. Namun dalam dunia nyata, sering kali kita punya banyak variabel (bukan cuma luas tanah, tapi juga lokasi, jumlah kamar, umur bangunan, dll). Kadang ada variabel yang mirip atau berkorelasi tinggi satu sama lain (misalnya, luas tanah dan luas bangunan). Oleh karena itu dua masalah umum muncul adalah:

a. Overfitting

Model terlalu "menghafal" data pelatihan, bukan "memahami pola". Akibatnya, model bagus di data latihan tapi jelek di data baru. Ciri khas: garis atau permukaan prediksi terlalu "bergelombang".

b. Multikolinearitas

Beberapa variabel terlalu mirip sehingga perhitungan jadi tidak stabil. Misalnya, dua kolom input hampir sama yang menyebabkan hasil bobot w_i bisa menjadi sangat besar atau tidak menentu, walau prediksi tampak benar.

Ridge Regression adalah pengembangan dari linear regression yang digunakan untuk mengatasi masalah *overfitting* dan multikolinearitas (hubungan kuat antar variabel independen). Pada linear regression biasa, model mencari garis terbaik yang meminimalkan jumlah kuadrat selisih antara nilai sebenarnya dengan nilai prediksi. Namun, jika jumlah fitur sangat banyak atau beberapa fitur saling berkorelasi tinggi, model bisa menjadi terlalu kompleks dan hasil prediksinya tidak stabil. Ridge regression mengatasi masalah ini dengan menambahkan istilah penalti (regularisasi) pada fungsi loss, yaitu berupa jumlah kuadrat dari koefisien regresi. Artinya, selain

berusaha menyesuaikan data sebaik mungkin, ridge regression juga berusaha menjaga agar nilai koefisien tidak terlalu besar. Secara matematis, fungsi yang diminimalkan dalam ridge regression adalah:

$$Loss = \sum(y_i - \hat{y}_i)^2 + \lambda \sum \beta_j^2$$

Bagian pertama adalah error prediksi seperti pada linear regression biasa, sementara bagian kedua adalah penalti regulasi yang dikendalikan oleh parameter λ. Jika λ bernilai nol, maka ridge regression sama saja dengan linear regression biasa. Jika λ semakin besar, maka koefisien regresi akan dipaksa semakin kecil (didekatkan ke nol), sehingga model menjadi lebih sederhana dan lebih tahan terhadap *overfitting*.

Kelebihan ridge regression adalah kemampuannya memberikan prediksi yang lebih stabil pada dataset dengan banyak fitur dan mengurangi varians model. Namun, kekurangannya adalah ridge tidak dapat benar-benar membuat koefisien menjadi nol, sehingga tidak bisa digunakan untuk seleksi fitur (berbeda dengan Lasso Regression).

Kelebihan dan Kelemahan Ridge Regression

Kelebihan:

- Mengurangi overfitting melalui regularisasi.
- Menangani multikolinearitas.
- Memiliki solusi analitik yang stabil.
- Cocok untuk kasus dengan banyak fitur.

Kelemahan:

- Tidak dapat menghasilkan *feature selection* (semua fitur tetap ada).
- Interpretasi koefisien menjadi sulit jika terlalu banyak shrinkage.

Jadi ridge regression adalah metode regularisasi linear yang menyeimbangkan bias dan variansi melalui pinalti L_2. Dengan menambahkan penalti terhadap besar koefisien, ridge mampu memperbaiki kestabilan model dan meningkatkan kemampuan generalisasi terutama saat fitur saat itu saling berkorelasi tinggi.

Secara matematis, Ridge Regression meminimalkan:

$$J(\beta) = \|y - X\beta\|^2 + \lambda\|\beta\|^2$$

Dan menghasilkan solusi tertutup:

$$\hat{\beta}_{ridge} = (X^T X + \lambda I)^{-1} X^T y$$

Solusi tertutup disini artinya kita bisa langsung mendapatkan hasil akhir dari sebuah persamaan dengan rumus pasti, tanpa harus mencoba-coba atau menggunakan perulangan. Tujuan Ridge Regression adalah mencari bobot β yang membuat prediksi mendekati data asli, tetapi tidak terlalu besar supaya model tetap stabil, solusi tertutu diatas dapat dijabarkan sebagai berikut:

- $X^T X$, ini seperti melihat hubungan antar kolom data (fitur). Misalnya apakah luas rumah dan jumlah kamar saling berhubungan
- $+\lambda I$, disini kita menambahkan angka kecil pinalti supaya perhitungan tetap aman dan tidak error. Tanpa ini, rumus bisa macet kalau datanya terlalu mirip antar kolom.

" Ridge regression adalah teknik regresi linear yang menambahkan penalti terhadap besarnya koefisien untuk mengurangi overfitting dan meningkatkan kestabilan model. Metode ini efektif ketika terdapat multikolinearitas antar fitur, karena membantu menjaga koefisien tetap kecil dan model lebih generalisasi".

5.7. Lasso Regression

Lasso Regression adalah salah satu varian dari linear regression yang menggunakan teknik *regularisasi*, mirip dengan ridge regression, tetapi dengan perbedaan penting pada jenis penalti yang digunakan. Jika ridge regression menambahkan penalti berupa kuadrat koefisien ($\sum\beta_j^2$), maka lasso regression menambahkan penalti berupa nilai absolut koefisien ($\sum|\beta_j|$) Dengan kata lain, lasso regression berusaha menekan nilai koefisien agar kecil, bahkan bisa memaksanya menjadi nol.

Secara matematis, fungsi loss lasso adalah:

$$Loss = \sum(y_i - \hat{y}_i)^2 + \lambda\sum|\beta_j|$$

Di sini, λ adalah parameter regulasi yang menentukan seberapa kuat penalti diterapkan. Jika $\lambda = 0$ maka lasso sama saja dengan linear regression biasa. Jika λ semakin besar, maka semakin banyak koefisien yang ditekan menuju nol. Inilah keunggulan utama lasso regression dibanding ridge: selain membantu mencegah *overfitting*, lasso juga dapat melakukan seleksi fitur otomatis dengan menghilangkan variabel yang dianggap kurang penting (karena koefisiennya dipaksa nol).

Kelebihan lasso regression adalah kemampuannya menyederhanakan model, membuatnya lebih mudah diinterpretasikan, dan efektif saat bekerja dengan dataset berdimensi tinggi. Namun, kekurangannya adalah jika ada fitur yang saling berkorelasi tinggi, lasso cenderung memilih salah satunya saja dan mengabaikan yang lain, sehingga bisa kehilangan informasi.

Walaupun sama-sama menambahkan sebuah term penalti ke dalam fungsi objektif. Tujuannya adalah membatasi besar koefisien

agar model menjadi lebih sederhana. Regularisasi pada Ridge Regression dan Lasso Regression ada perbedaan yaitu:

- Ridge Regression (L2 Regularization) , yaitu menambahkan pinalti berdasarkan kuadrat nilai koefisien

$$\lambda \sum_{j=1}^{p} \beta_j^2$$

- Lasso Regression (L1 Regularization) , yaitu menambah pinaltu berdasarkan nilai absolut koefisien.

$$\lambda \sum_{j=1}^{p} |\beta_j|$$

Perbedaan bentuk pinalti inilah yang menghasilkan perilaku berbeda pada kedua metode. Lasso mampu menenkan beberapa koefisien hingga nol sementara Ridge hanya memperkecil nilainya tanpa benar-benar menjadi nol.

Fungsi objektif yang diminimalkan oleh Lasso Regression adalah:

$$\mathcal{L}(\beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

dengan:

- $\lambda \geq 0$ adalah parameter regularisasi yang mengontrol kekuatan pinalti
- jika $\lambda = 0$ maka lasso identik dengan OLS (Ordinary Linear Regression). Jika λ besar maka pinalti meningkat dan lebih banyak koefisien ditekan menuju nol

Contoh kasus:

Misalkan kita ingin memprediksi harga rumah berdasarkan tiga fitur:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon$$

Setelah dilakukan pelatihan dengan Lasso dan diperoleh:

$$\hat{\beta}_1 = 5.2\,, \hat{\beta}_2 = 0, \hat{\beta}_3 = 2.8$$

Artinya fitur kedua x_2 dianggap tidak relevan oleh model sehingga dieliminasi secara otomatis.

Kelebihan:

- Melakukan feature selection otomatis.
- Menghasilkan model sederhana dan mudah diinterpretasikan.
- Mengurangi overfitting.
- Dapat digunakan untuk dataset dengan jumlah fitur besar

Kekurangan:

- Sulit dipakai jika jumlah fitur > jumlah sampel kecil (walau masih bisa diadaptasi).
- Tidak stabil jika fitur sangat berkorelasi (karena bisa memilih salah satu dan mengabaikan lainnya).
- Tidak memiliki solusi analitik sederhana.

Aplikasi Nyata Lasso Regression

- Seleksi gen pada data biomedis (karena fitur sangat banyak).
- Pemodelan keuangan untuk memilih variabel makroekonomi yang paling berpengaruh.
- Pengolahan sinyal untuk menghapus noise.
- Pemilihan fitur otomatis pada model prediksi teks.

" Lasso Regression adalah metode regularisasi berbasis L1 norm yang menambahkan penalti terhadap nilai absolut koefisien regresi. Dengan menekan beberapa koefisien menjadi nol, Lasso tidak hanya mengurangi overfitting tetapi juga berperan sebagai metode seleksi fitur otomatis".

5.8. Elastic Net

Regresi Jaring Elastis, atau *Elastic Net Regression*, adalah metode regresi ter-regularisasi yang secara cerdas menggabungkan kekuatan dari dua pendahulunya: Regresi Ridge dan Regresi Lasso. Metode ini dirancang untuk mengatasi kelemahan masing-masing metode tersebut, menjadikannya alat yang sangat kuat dan fleksibel dalam pemodelan statistik dan *machine learning*, terutama saat berhadapan dengan data berdimensi tinggi dan fitur yang saling berkorelasi.

Dalam model regresi linear standar (*Ordinary Least Squares* atau OLS), tujuannya adalah untuk meminimalkan jumlah kuadrat galat (*Residual Sum of Squares* atau RSS). Namun, ketika dihadapkan pada dataset dengan banyak sekali fitur (lebih banyak fitur daripada jumlah data) atau fitur-fitur yang berkorelasi tinggi (multikolinearitas), model OLS cenderung mengalami *overfitting*. Artinya, model menjadi terlalu "hafal" dengan data latih dan gagal memberikan prediksi yang akurat pada data baru.

Untuk mengatasi ini, diperkenalkanlah teknik regularisasi, yang bekerja dengan cara menambahkan "penalti" ke dalam fungsi kerugian (*loss function*) untuk koefisien regresi yang terlalu besar. Penalti ini memaksa model untuk tetap sederhana dan mengurangi kecenderungannya untuk *overfitting*. Dua metode regularisasi yang paling populer adalah Ridge dan Lasso.

1. **Regresi Ridge (Penalti L2):** Ridge sangat efektif dalam menangani multikolinearitas. Ia menambahkan penalti yang sebanding dengan jumlah kuadrat dari besaran koefisien (norma L2). Hal ini "menyusutkan" (shrink) koefisien dari fitur-fitur yang berkorelasi secara bersama-sama, membuat model lebih stabil. Namun, kelemahannya adalah Ridge hanya mampu menyusutkan koefisien mendekati nol, tetapi tidak pernah membuatnya menjadi nol. Akibatnya, Ridge tidak dapat melakukan seleksi fitur.
2. **Regresi Lasso (Penalti L1):** Keunggulan utama Lasso adalah kemampuannya melakukan seleksi fitur Ia menambahkan penalti yang sebanding dengan jumlah nilai absolut dari koefisien (norma L1). Penalti ini dapat memaksa beberapa koefisien menjadi tepat nol, secara efektif menghilangkan fitur yang tidak relevan dari model. Namun, kelemahan Lasso muncul saat berhadapan dengan fitur-fitur yang berkorelasi tinggi; ia cenderung memilih salah satu fitur secara acak dari sebuah grup dan mengabaikan yang lainnya, sehingga membuatnya tidak stabil.

Elastic Net diperkenalkan untuk menyatukan kelebihan dari kedua metode tersebut. Ia menggabungkan penalti L1 dari Lasso dan penalti L2 dari Ridge ke dalam satu fungsi kerugian. Hal ini memungkinkan Elastic Net untuk melakukan seleksi fitur seperti Lasso, sekaligus menjaga stabilitas model dalam menghadapi multikolinearitas seperti Ridge.

Fungsi kerugian yang diminimalkan oleh Elastic Net adalah kombinasi dari RSS, penalti L1, dan penalti L2. Secara matematis, formula ini dapat ditulis dalam dua cara yang ekuivalen:

$$Loss = \sum(y_i - \hat{y}_i)^2 + \lambda_1 \sum|\beta_j| + \lambda_2 \sum \beta_j^2$$

Disini:

- $\sum(y_i - \hat{y}_i)^2$: Ini adalah komponen *Ordinary Least Squares* (OLS), yaitu jumlah kuadrat galat antara nilai aktual (y_i) dan nilai prediksi ($\hat{y}_i$).
- $\lambda_1 \sum|\beta_j|$: Ini adalah penalti L1 (Lasso), di mana $|\beta_j|$ adalah nilai absolut dari koefisien fitur ke-j, dan λ_1 adalah parameter tuning yang mengontrol kekuatan penalti ini.
- $\lambda_2 \sum \beta_j^2$: Ini adalah penalti L2 (Ridge), di mana β_j^2 adalah kuadrat dari koefisien, dan λ_2 adalah parameter tuning yang mengontrol kekuatan penalti L2.

Persamaan ini kemudian disederhanakan agar proses tuning menjadi dua parameter utama:

$$Loss = \sum \sum(y_i - \hat{y}_i)^2 + \lambda[\alpha \sum|\beta_j| + (1-\alpha)\sum \beta_j^2]$$

Di sini:

- λ (Lambda): Parameter ini mengontrol kekuatan keseluruhan regularisasi. Semakin besar nilai λ, semakin besar penalti yang diberikan, yang akan menyusutkan semua koefisien lebih kuat menuju nol. Jika λ = 0, model ini menjadi regresi OLS biasa.
- a (Alpha): Parameter ini disebut "*mixing parameter*" yang mengontrol proporsi antara penalti Lasso dan Ridge. Nilai a berada di rentang:

- Jika a = 1, penalti L2 menjadi nol, dan model ini menjadi Regresi Lasso murni.
- Jika a = 0, penalti L1 menjadi nol, dan model ini menjadi Regresi Ridge murni.
- Jika 0 < a < 1, model ini adalah kombinasi dari keduanya, yang merupakan esensi dari Elastic Net.

Bayangkan kita ingin memprediksi harga sebuah rumah dengan fitur-fitur berikut: luas_bangunan, jumlah_kamar_tidur, luas_garasi, usia_bangunan, dan tingkat_kejahatan_lokal.

Dalam dataset ini, sangat mungkin fitur luas_bangunan, jumlah_kamar_tidur, dan luas_garasi saling berkorelasi tinggi.

- Jika menggunakan Lasso (a=1): Lasso mungkin akan memilih salah satu dari tiga fitur tersebut (misalnya, luas_bangunan) sebagai prediktor yang kuat dan membuat koefisien untuk jumlah_kamar_tidur dan luas_garasi menjadi nol, meskipun keduanya juga relevan. Ini membuat hasil model menjadi kurang stabil.
- Jika menggunakan Ridge (a=0): Ridge akan mempertahankan ketiga fitur tersebut dan menyusutkan koefisiennya secara bersamaan. Namun, jika tingkat_kejahatan_lokal adalah fitur yang tidak relevan, Ridge akan tetap mempertahankannya dalam model dengan koefisien yang kecil, bukan menghilangkannya.
- Jika menggunakan Elastic Net (misalnya, a=0.5): Elastic Net mampu menangani situasi ini dengan lebih baik.

- Efek Pengelompokan (*Grouping Effect*): Berkat komponen Ridge (L2), Elastic Net akan mengidentifikasi luas_bangunan, jumlah_kamar_tidur, dan luas_garasi sebagai sebuah grup fitur yang berkorelasi dan menyusutkan koefisien mereka secara bersamaan.
- Seleksi Fitur: Pada saat yang sama, berkat komponen Lasso (L1), jika fitur tingkat_kejahatan_lokal terbukti tidak signifikan, Elastic Net dapat menyusutkan koefisiennya hingga menjadi tepat nol, sehingga fitur tersebut dihilangkan dari model akhir.

Keunggulan:

- Kombinasi Terbaik: Menggabungkan kemampuan seleksi fitur dari Lasso dengan kemampuan menangani multikolinearitas dari Ridge.
- Efek Pengelompokan: Sangat efektif untuk situasi di mana terdapat sekelompok fitur yang saling berkorelasi tinggi.
- Fleksibilitas Tinggi: Dengan menyesuaikan parameter a, model ini bisa menjadi Lasso, Ridge, atau kombinasi keduanya, membuatnya sangat mudah beradaptasi dengan berbagai jenis data.
- Performa Unggul: Seringkali memberikan performa prediksi yang lebih baik daripada Lasso atau Ridge saja, terutama pada dataset dengan banyak fitur (*high-dimensional*) atau multikolinearitas yang tinggi.

Kelemahan:

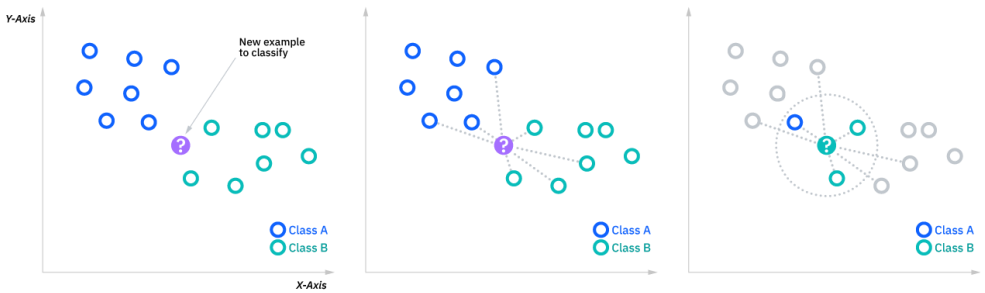
- Kompleksitas Tuning: Kelemahan utamanya adalah kebutuhan untuk menyetel dua *hyperparameter* (λ dan α), yang membuat proses optimasi menjadi lebih rumit dan memakan waktu komputasi lebih lama dibandingkan dengan Ridge atau Lasso yang hanya memiliki satu parameter.

" Elastic Net adalah metode regresi yang menggabungkan penalti L1 (Lasso) dan L2 (Ridge) untuk mengatasi multikolinearitas sekaligus melakukan seleksi fitur. Pendekatan ini efektif dalam menangani dataset dengan banyak fitur yang saling berkorelasi, sehingga menghasilkan model yang stabil dan lebih interpretatif".

5.9. K-Nearest Neighbors

Algoritma k-tetangga terdekat (KNN) adalah pengklasifikasi pembelajaran non-parametrik dan terawasi, yang menggunakan kedekatan untuk membuat klasifikasi atau prediksi tentang pengelompokan titik data individu. Ini adalah salah satu klasifikasi dan regresi yang populer dan paling sederhana yang digunakan dalam machine learning saat ini.

Meskipun algoritma KNN dapat digunakan untuk masalah regresi atau klasifikasi, algoritma ini biasanya digunakan sebagai algoritma klasifikasi, dengan asumsi bahwa titik-titik yang serupa dapat ditemukan berdekatan satu sama lain.



Gambar 27
Algoritma KNN

Singkatnya, tujuan dari algoritma k-tetangga terdekat adalah untuk mengidentifikasi tetangga terdekat dari titik kueri tertentu, sehingga kita dapat menetapkan label kelas ke titik itu. Untuk menentukan titik data mana yang paling dekat dengan titik kueri yang diberikan, jarak antara titik kueri dan titik data lainnya perlu dihitung. Metrik jarak ini membantu membentuk batas keputusan,

yang mempartisi titik kueri ke wilayah yang berbeda. Anda biasanya akan melihat batas-batas keputusan yang divisualisasikan dengan diagram Voronoi.

Algoritma K-Nearest Neighbors (KNN) merupakan salah satu algoritma tertua dan paling intuitif dalam machine learning, terutama dalam kategori algoritma berbasis instance (instance-based learning).

KNN digunakan baik untuk klasifikasi maupun regresi, bergantung pada jenis target yang diprediksi:

- Untuk *klasifikasi*, KNN menentukan kelas dari suatu data baru berdasarkan mayoritas kelas dari *K* tetangga terdekatnya.
- Untuk *regresi*, KNN mengambil rata-rata nilai target dari *K* tetangga terdekat.

Kelebihan KNN terletak pada kesederhanaan konsepnya: prediksi dihasilkan langsung dari data pelatihan tanpa model eksplisit. Namun kelemahannya, KNN tidak melakukan generalisasi eksplisit yang artinya, model hanya melakukan perhitungan saat prediksi dilakukan (*lazy learner*).

Bayangkan Anda memiliki sekumpulan titik data yang mewakili dua kelas (misalnya "A" dan "B"). Ketika ada titik baru yang belum diketahui kelasnya, Anda dapat memeriksa siapa saja tetangga terdekatnya di ruang fitur. Jika sebagian besar tetangganya dari kelas A, maka titik tersebut diasumsikan termasuk kelas A, dan sebaliknya. Secara konseptual, KNN dapat diibaratkan sebagai pendekatan "voting" di ruang fitur, di mana tetangga terdekat memiliki suara paling berpengaruh terhadap hasil prediksi.

Misalkan setiap contoh data $x^{(i)} = \left[x_1^{(i)}, x_2^{(i)}, \dots, x_d^{(i)}\right]^T \in \mathbb{R}^d$

dimana

- $x^{(i)}$ adalah data ke-i dari kumpulan data pelatihan
- $x_1^{(i)}, x_2^{(i)}, \dots, x_d^{(i)}$ adalah fitur-fitur (ciri-ciri) dari data ke-i
- d adalah jumlah fitur (dimensi ruang data). Misalkan jika datanya memiliki dua fitur (umur dan tinggi), maka $d = 2$ dan jika datanya memiliki 3 fitur (umur, tinggi, berat) maka $d = 3$
- $[\cdot]^T$ adalah tanda transpose, artinya vektor kolom
- $\mathbb{R}^d$ artinya ruang berdimensi d yaitu himpunan semua vektor dengan d komponen real

Kemudian target labelnya adalah $y^{(i)}$. Jika klasifikasi biner $y^{(i)} \in \{0,1\}$. Jika klasifikasi multi-kelas maka $y^{(i)} \in \{0,1,2,3, \dots, C\}$. Jika regresi maka → $y^{(i)} \in \mathbb{R}$

Jika kita memiliki data baru baru yang belum diketahui labelnya, misalkan:

$$\mathrm{x_q} = \left[\mathrm{x_1^{(q)}}, \mathrm{x_2^{(q)}}, \dots, \mathrm{x_d^{(q)}}\right]^{\mathrm{T}}$$

Yang ingin ketahui adalah label apa yang seharusnya dimiliki x_q. Ide utama KNN adalah bekerja berdasarkan kemiripan, lebih tepatnya kedekatan jarak antar titik. Jika dirangkum tugas KNN adalah mencari K tetangga terdekat di antara semua contoh pelatihan $\{x^{(1)}, x^{(2)}, \dots, x^{(N)}\}$. Artinya kita memiliki N data pelatihan (training examples), untuk titik uji x_q, kita hitung jaraknya ke setiap data pelatihan. Pilih K data yang jaraknya paling dekat dengan x_q. Data-data ini disebut K tetangga terdekat (K nearest neighbors).

Cara umum untuk mengukur jarak antara dua titik $x^{(i)}$ dan x_q adalah dengan jarak Euclidean:

$$d(x^{(i)}, x_q) = \sqrt{\left(x_1^{(i)} - x_1^{(q)}\right)^2 + \left(x_2^{(i)} - x_2^{(q)}\right)^2 + \cdots + \left(x_d^{(i)} - x_d^{(q)}\right)^2}$$

Dari persamaan ini jika jaraknya kecil maka lebih mirip antara label prediksi sedangkan jika jarak besar artinya tidak mirip antara label dengan prediksi. Kesimpulannya K-Nearest Neighbors adalah algoritma sederhana yang mengandalkan *kesamaan lokal* antar data. Dengan menilai tetangga terdekat di ruang fitur, KNN dapat beradaptasi dengan berbagai bentuk distribusi data tanpa perlu asumsi model eksplisit. Namun demikian, kelemahan utama KNN seperti waktu prediksi yang lama dan sensitivitas terhadap skala fitur membuat algoritma ini lebih cocok untuk dataset kecil hingga menengah, atau digunakan sebagai tolok ukur dasar (baseline) dalam eksperimen *machine learning*.

" Elastic Net adalah metode regresi yang menggabungkan penalti L1 (Lasso) dan L2 (Ridge) untuk mengatasi multikolinearitas sekaligus melakukan seleksi fitur. Pendekatan ini efektif dalam menangani dataset dengan banyak fitur yang saling berkorelasi, sehingga menghasilkan model yang stabil dan lebih interpretatif".

BAGIAN 6
CLUSTERING DAN TEMAN-TEMANNYA

6.1. K-Means

K-Means adalah salah satu algoritma unsupervised learning yang paling populer dalam analisis klaster (clustering). Tujuan dari algoritma ini adalah membagi sekumpulan data menjadi beberapa kelompok (klaster) sedemikian rupa sehingga data dalam satu klaster memiliki kesamaan yang tinggi, sementara antar-klaster memiliki perbedaan yang besar (Bishop & Nasrabadi, 2006b).

Secara intuitif, K-Means berusaha menemukan posisi titik-titik pusat (disebut centroid) yang paling baik untuk mewakili setiap klaster. Centroid dapat dianggap sebagai "rata-rata" atau prototipe ideal dari anggota-anggota klaster tersebut.

Agar lebih mudah dipahami, berikut adalah langkah-langkah umum yang dilakukan oleh algoritma K-Means:

1. **Menentukan jumlah klaster (k).**
   Pengguna harus menentukan di awal berapa banyak klaster yang diinginkan. Nilai ini dinyatakan sebagai k, sebuah bilangan bulat positif.
2. **Inisialisasi centroid secara acak.**
   Algoritma memilih secara acak k contoh data dari dataset sebagai posisi awal centroid masing-masing klaster.
3. **Menetapkan anggota klaster.**
   Setiap data (disebut *contoh*) dihitung jaraknya terhadap semua centroid menggunakan jarak Euclidean (Euclidean distance). Contoh data kemudian dimasukkan ke klaster dengan centroid

terdekat. Prinsipnya: *centroid terdekat "memenangkan" contoh tersebut.*

4. **Menghitung ulang posisi centroid.**
   Setelah semua contoh memiliki label klaster, algoritma menghitung ulang posisi tiap centroid dengan cara mengambil rata-rata seluruh contoh dalam klaster tersebut. Biasanya setelah langkah ini, posisi centroid tidak lagi bertepatan dengan data asli—ia menjadi representasi ideal dari klaster (sebuah *prototype*).
5. **Pemeriksaan konvergensi.**
   Algoritma kemudian memeriksa seberapa jauh posisi centroid berubah dibandingkan iterasi sebelumnya.
   - Jika perubahan posisi sangat kecil (di bawah ambang batas tertentu), algoritma menganggap bahwa solusi telah stabil.
   - Jika masih banyak perubahan, algoritma akan mengulangi langkah 3 dan 4 hingga posisi centroid berhenti berubah secara signifikan.

Proses pengulangan ini disebut iterasi. K-Means akan terus berulang hingga mencapai kondisi konvergensi, yaitu ketika sebagian besar data tidak lagi berpindah klaster.

Setelah algoritma K-Means mencapai titik stabil, hasil akhirnya memiliki beberapa karakteristik penting:

1. Setiap data memiliki satu label klaster.
   Semua contoh akan terpisah ke dalam k klaster, dan setiap contoh hanya menjadi anggota satu klaster saja.
2. Setiap klaster memiliki *kohesi internal maksimum*.

Artinya, jarak antara anggota klaster terhadap centroid-nya seminimal mungkin.

Setiap klaster memiliki *perbedaan eksternal maksimum*. Artinya, jarak antar centroid (antara klaster yang berbeda) sejauh mungkin. K-Means bersifat iteratif dan sensitif terhadap inisialisasi acak. Karena centroid awal dipilih secara acak, dua kali menjalankan K-Means pada dataset yang sama bisa menghasilkan hasil yang berbeda.

Oleh karena itu, praktik umum dalam penerapan K-Means adalah:

- Menjalankan algoritma berulang kali (misalnya 10–100 kali) dengan inisialisasi acak berbeda.
- Memilih hasil dengan WSS terendah dan BSS tertinggi sebagai solusi terbaik.

Pendekatan ini membantu menghindari hasil yang tidak stabil atau jebakan pada local minimum, yaitu kondisi di mana algoritma berhenti pada solusi yang tampak stabil tetapi bukan yang paling optimal secara global.

Salah satu tantangan utama dalam K-Means adalah menentukan nilai k yang tepat. Tidak ada aturan pasti, tetapi terdapat beberapa pendekatan untuk menentukannya:

1. Tujuan eksploratif (exploratory analysis). Jika K-Means digunakan untuk memahami struktur data, maka k dapat dipilih berdasarkan interpretasi yang masuk akal misalnya, jumlah kategori yang dapat diberi nama atau makna tertentu.

2. Metode statistik seperti *Elbow Method*. Metode ini mengamati penurunan nilai WSS terhadap peningkatan jumlah klaster. Ketika grafik mulai "melengkung" dan tidak banyak penurunan signifikan lagi, titik tersebut dianggap nilai k yang optimal.
3. Kebutuhan model lanjutan. Jika hasil K-Means akan digunakan sebagai masukan bagi algoritma *supervised learning* (misalnya klasifikasi), maka nilai k dapat ditentukan dengan cross-validation yaitu memilih k yang menghasilkan performa prediksi terbaik di model berikutnya.

Kelebihan:

- Sederhana dan mudah diimplementasikan.
- Cepat untuk dataset berukuran besar.
- Memberikan hasil yang mudah diinterpretasikan secara visual.

Keterbatasan:

- Harus menentukan jumlah klaster (k) di awal.
- Sensitif terhadap posisi inisialisasi centroid.
- Tidak cocok untuk klaster dengan bentuk non-sferis (misalnya, klaster memanjang atau kompleks).
- Kurang tahan terhadap *outlier*.

K-Means merupakan algoritma yang efektif untuk menemukan struktur tersembunyi dalam data, terutama ketika bentuk klasternya relatif sederhana. Meskipun sederhana, algoritma ini menjadi dasar bagi banyak metode clustering lanjutan, seperti K-Medoids, Fuzzy C-Means, dan Mini-Batch K-Means.

Pemahaman mendalam terhadap prinsip K-Means membantu kita memahami konsep penting dalam pembelajaran tanpa supervisi yaitu bagaimana mesin dapat "mengelompokkan" data berdasarkan kesamaan tanpa bantuan label manusia.

" K-Means adalah algoritma clustering yang membagi data menjadi sejumlah cluster berdasarkan kedekatan jarak antara titik-titik data dengan pusat cluster (centroid). Metode ini efektif untuk menemukan pola atau struktur dalam data tanpa memerlukan label sebelumnya".

6.2. Hierarchical Clustering

Hierarchical Clustering adalah algoritma *unsupervised learning* yang digunakan untuk mengelompokkan data ke dalam struktur bertingkat menyerupai pohon, yang disebut dendrogram. Berbeda dengan K-Means yang membutuhkan jumlah cluster kkk sejak awal, hierarchical clustering tidak selalu memerlukan jumlah cluster terlebih dahulu, karena kita bisa memotong dendrogram pada level tertentu untuk mendapatkan jumlah cluster yang diinginkan.

Cara kerjanya ada dua pendekatan utama:

1. Agglomerative (bottom-up) dimana metode yang paling umum. Algoritma mulai dengan menganggap setiap data sebagai cluster tersendiri. Kemudian, pada setiap langkah, dua cluster yang paling mirip digabungkan menjadi satu. Proses ini diulang terus hingga semua data bergabung menjadi satu cluster besar.
2. Divisive (top-down) adala kebalikannya, algoritma mulai dari satu cluster besar yang berisi semua data, lalu secara bertahap memecahnya menjadi cluster yang lebih kecil.

Kunci dari hierarchical clustering adalah bagaimana mengukur jarak atau kemiripan antar cluster. Ada beberapa metode populer, seperti single linkage (jarak terdekat antar anggota cluster), complete linkage (jarak terjauh antar anggota cluster), dan average linkage (rata-rata jarak antar semua anggota).

Kelebihan hierarchical clustering adalah hasilnya bisa divisualisasikan dalam bentuk dendrogram, sehingga kita bisa melihat struktur alami data dan memilih jumlah cluster secara fleksibel. Selain

itu, algoritma ini tidak memerlukan inisialisasi centroid seperti pada K-Means. Namun, kelemahannya adalah waktu komputasinya bisa sangat lama untuk dataset besar, dan hasil clustering sensitif terhadap cara pengukuran jarak yang dipilih.

Berbeda dengan algoritma seperti *K-Means* yang membutuhkan jumlah cluster $k$di awal, Hierarchical Clustering membangun struktur hierarki antar data yang dapat dipotong di berbagai level untuk menghasilkan berbagai jumlah cluster. Struktur tersebut divisualisasikan menggunakan dendrogram, sebuah diagram pohon yang menunjukkan jarak dan hubungan antar data atau antar kelompok.

Proses utamanya adalah:

1. Hitung jarak antara semua pasangan titik data.
2. Gabungkan dua titik (atau cluster) yang memiliki jarak paling kecil.
3. Hitung jarak antara cluster baru dengan cluster lain menggunakan aturan tertentu (*linkage criteria*).
4. Ulangi hingga semua data bergabung menjadi satu cluster.

Seperti yang telah disebutkan, kunci dari *Hierarchical Clustering* adalah bagaimana cara mengukur jarak atau kemiripan antar *cluster*. Metode untuk menghitung ketidakmiripan antar *cluster* ini disebut kriteria *linkage*. Pilihan metode *linkage* ini dapat sangat memengaruhi hasil akhir dan bentuk *cluster* yang terbentuk. Berikut adalah beberapa metode *linkage* yang paling umum digunakan:

- ***Single Linkage*:** Dikenal juga sebagai metode tetangga terdekat. Jarak antara dua *cluster* didefinisikan sebagai jarak minimum antara dua titik dari masing-masing *cluster*. Metode ini cenderung menghasilkan *cluster* yang bentuknya memanjang seperti rantai.
- ***Complete Linkage*:** Dikenal juga sebagai metode tetangga terjauh. Jarak antara dua *cluster* didefinisikan sebagai jarak maksimum antara dua titik dari masing-masing *cluster*. Metode ini cenderung menghasilkan *cluster* yang lebih padat dan berbentuk bulat.
- ***Average Linkage*:** Jarak antara dua *cluster* dihitung sebagai rata-rata jarak dari semua pasangan titik di kedua *cluster* tersebut. Metode ini merupakan penengah antara *single* dan *complete linkage*.
- ***Ward's Minimum Variance*:** Metode ini menggabungkan dua *cluster* yang menghasilkan kenaikan total varians dalam *cluster* yang paling kecil. Tujuannya adalah untuk meminimalkan varians di dalam setiap *cluster*. Metode ini cenderung menghasilkan *cluster-cluster* dengan ukuran yang serupa.
- ***Centroid Linkage*:** Jarak antara dua *cluster* dihitung berdasarkan jarak antara titik pusat (*centroid*) dari masing-masing *cluster*.

Hierarchical Clustering digunakan secara luas di berbagai disiplin ilmu, antara lain:

- **Biologi:** Digunakan untuk mengelompokkan gen dengan fungsi serupa atau untuk membangun pohon filogenetik yang menunjukkan hubungan evolusioner antar spesies.

- **Pemasaran:** Membantu dalam segmentasi pasar dengan mengelompokkan pelanggan berdasarkan perilaku pembelian, demografi, atau preferensi mereka. Hal ini memungkinkan perusahaan untuk menargetkan kampanye pemasaran dengan lebih efektif.
- **Analisis Dokumen:** Mengelompokkan dokumen, artikel, atau laporan berdasarkan topiknya.
- **Pengolahan Citra:** Digunakan untuk segmentasi gambar, misalnya untuk mengelompokkan piksel-piksel dengan warna atau tekstur yang serupa.

" Hierarchical clustering adalah metode pengelompokan data yang membangun struktur pohon (dendrogram) untuk menunjukkan hubungan hierarkis antar data. Pendekatan ini memungkinkan analisis cluster pada berbagai level granularitas, dari pengelompokan luas hingga pengelompokan yang lebih spesifik".

6.3. Gaussian Mixture Model

Gaussian Mixture Model (GMM) adalah algoritma *unsupervised learning* untuk clustering yang berbasis pada probabilitas. Tidak seperti K-Means yang membagi data ke dalam cluster secara tegas (setiap titik hanya masuk ke satu cluster), GMM menggunakan pendekatan soft clustering, yaitu memberikan probabilitas seberapa besar kemungkinan sebuah titik data termasuk ke dalam masing-masing cluster.

Ide dasarnya adalah bahwa data yang kita miliki berasal dari campuran beberapa distribusi Gaussian (normal). Setiap cluster dianggap sebagai sebuah distribusi Gaussian dengan parameter mean (μ) dan varians (σ^2), serta bobot probabilitas tertentu. Misalnya, jika kita ingin memisahkan data ke dalam tiga cluster, maka GMM akan mencoba memodelkan data sebagai campuran tiga distribusi normal.

Proses pembelajaran GMM biasanya menggunakan algoritma Expectation-Maximization (EM):

1. **Expectation (E-step):** menghitung probabilitas setiap titik data termasuk ke dalam tiap distribusi Gaussian (cluster) berdasarkan parameter saat ini.
2. **Maximization (M-step):** memperbarui parameter distribusi (mean, varians, dan bobot) agar sesuai dengan probabilitas yang dihitung pada E-step.

Langkah ini diulang berkali-kali sampai model stabil.

Keunggulan GMM adalah fleksibilitasnya: karena menggunakan distribusi Gaussian, ia mampu membentuk cluster dengan bentuk elips (tidak hanya bulat seperti K-Means), dan memberikan informasi probabilistik (misalnya, titik data A punya 70% kemungkinan di cluster 1, 30% di cluster 2). Namun, kekurangannya adalah GMM cukup sensitif terhadap inisialisasi parameter, dan memerlukan asumsi bahwa data memang cocok dimodelkan dengan distribusi Gaussian. Selain itu, kita tetap perlu menentukan jumlah cluster terlebih dahulu.

Gaussian Mixture Model (GMM) adalah salah satu algoritma unsupervised learning yang digunakan untuk clustering probabilistik. Berbeda dengan algoritma seperti *K-Means* atau *DBSCAN* yang mengelompokkan data secara tegas (hard clustering), GMM mengelompokkan data secara probabilistik (soft clustering) — artinya, setiap data memiliki probabilitas untuk menjadi anggota setiap cluster.

GMM mengasumsikan bahwa data yang diamati berasal dari campuran beberapa distribusi Gaussian (normal). Setiap distribusi Gaussian dianggap mewakili satu komponen cluster, dan keseluruhan data dianggap terbentuk dari superposisi (mixture) beberapa Gaussian tersebut.

GMM sangat populer karena:

- Dapat menangani cluster dengan bentuk elips (bukan hanya bulat seperti K-Means).
- Menyediakan model probabilistik yang fleksibel.
- Memiliki dasar teori statistik yang kuat.

❖ Kelebihan

1. Dapat melakukan soft clustering (memberikan probabilitas keanggotaan).
2. Dapat menangani cluster dengan bentuk elips.
3. Berdasarkan model probabilistik yang kuat.
4. Mampu memberikan estimasi distribusi data yang realistis.

❖ Kekurangan

1. Memerlukan inisialisasi yang baik, sensitif terhadap titik awal.
2. Dapat terjebak pada local maxima dalam proses EM.
3. Sulit untuk memilih jumlah komponen Kyang optimal.
4. Kovarians tinggi menyebabkan overfitting untuk dimensi besar.

Gaussian Mixture Model adalah algoritma clustering yang berbasis probabilistik dan berlandaskan teori statistik Gaussian multivariat.

Dengan menggunakan algoritma EM, GMM mempelajari parameter distribusi yang paling mungkin menjelaskan data. GMM merupakan generalisasi dari K-Means yang lebih fleksibel dan mampu memberikan hasil *soft assignment* yang realistis, menjadikannya salah satu model paling penting dalam *unsupervised learning* dan *density estimation*.

" Gaussian Mixture Model (GMM) adalah metode probabilistik untuk clustering yang menganggap data berasal dari kombinasi beberapa distribusi Gaussian. Metode ini fleksibel dalam menangkap bentuk cluster yang kompleks dan tumpang tindih, berbeda dengan algoritma clustering berbasis jarak seperti K-Means".

6.4. Principal Component Analysis

Principal Component Analysis (PCA) adalah teknik *unsupervised learning* yang digunakan terutama untuk reduksi dimensi, yaitu menyederhanakan data yang memiliki banyak fitur (dimensi) menjadi lebih sedikit, tanpa kehilangan terlalu banyak informasi penting. Bayangkan kita punya dataset dengan puluhan atau ratusan variabel; PCA membantu kita menemukan representasi baru yang lebih ringkas, sehingga analisis atau algoritma machine learning bisa berjalan lebih cepat dan lebih efisien.

Cara kerja PCA bisa dibayangkan seperti mencari "sudut pandang terbaik" untuk melihat data. PCA mencari sumbu baru (disebut *principal components*) yang merupakan kombinasi linier dari fitur-fitur asli. Sumbu pertama (PC1) adalah arah yang menjelaskan variasi data terbesar, sumbu kedua (PC2) menjelaskan variasi terbesar berikutnya (dan tegak lurus dengan PC1), dan seterusnya. Dengan hanya mengambil beberapa sumbu pertama, kita sudah bisa menangkap sebagian besar informasi dari data, meskipun dimensinya jauh lebih sedikit.

Sebagai contoh, jika kita punya data wajah dengan ribuan piksel, PCA bisa digunakan untuk mengekstrak beberapa *principal components* yang merepresentasikan pola utama wajah, sehingga pengolahan data wajah menjadi lebih ringan. Dalam konteks machine learning, PCA sering dipakai sebelum melakukan klasifikasi atau clustering agar model lebih cepat dan tidak terlalu terjebak pada *noise*.

Kelebihan PCA adalah kemampuannya mereduksi kompleksitas data, mengurangi *overfitting*, dan membantu visualisasi data berdimensi tinggi (misalnya menampilkan data 2D/3D dari ratusan dimensi). Namun, kelemahannya adalah hasil komponen PCA sulit diinterpretasikan secara langsung, karena setiap komponen merupakan kombinasi dari banyak variabel asli. Selain itu, PCA hanya menangkap pola linier, sehingga tidak selalu efektif jika struktur data sangat non-linier.

Principal Component Analysis (PCA) adalah salah satu metode paling fundamental dalam unsupervised learning dan data preprocessing.
PCA digunakan untuk reduksi dimensi — yaitu, mengubah data berdimensi tinggi menjadi berdimensi lebih rendah, sambil mempertahankan sebanyak mungkin informasi (variasi) yang terkandung dalam data.

Konsep utama PCA adalah:

"Mencari arah baru (komponen utama) di mana data memiliki variasi terbesar."

Dengan kata lain, PCA menemukan sumbu koordinat baru (basis baru) yang:

- Berasal dari kombinasi linier fitur asli.
- Tidak saling berkorelasi (ortogonal).
- Mengurutkan sumbu berdasarkan seberapa besar varians data yang dijelaskan oleh masing-masing sumbu tersebut.

Dalam banyak kasus, data memiliki banyak fitur yang:

- Redundan (berkorelasi satu sama lain).
- Tidak semuanya relevan untuk analisis atau pembelajaran.
- Menyulitkan visualisasi dan menyebabkan *curse of dimensionality*.

Sebagai contoh:

Jika dua fitur $x_1$dan $x_2$sangat berkorelasi, maka sebenarnya mereka mengandung informasi yang mirip.

PCA dapat menggabungkan keduanya menjadi satu fitur baru (komponen utama) tanpa kehilangan banyak informasi.

Bayangkan kumpulan titik data 2D yang memanjang sepanjang suatu garis diagonal.

Arah diagonal itu merupakan arah di mana data memiliki variasi terbesar.

Jika kita memproyeksikan data ke arah diagonal tersebut, maka:

- Variasi data tetap besar.
- Dimensi data berkurang dari 2D menjadi 1D.

Arah diagonal ini disebut komponen utama pertama (first principal component).

Kelebihan

1. Mengurangi dimensi tanpa kehilangan informasi signifikan.
2. Menghilangkan korelasi antar fitur (fitur baru ortogonal).
3. Meningkatkan efisiensi komputasi untuk algoritma lain.
4. Dapat digunakan untuk visualisasi data berdimensi tinggi (misalnya 2D/3D).

Kekurangan

1. Hanya menangkap hubungan linear antar fitur.
2. Tidak mempertahankan interpretabilitas fitur (fitur baru = kombinasi linier).
3. Sensitif terhadap skala data (perlu standardisasi).
4. Komponen utama bisa sulit diinterpretasikan secara semantik.

Principal Component Analysis adalah metode fundamental dalam pembelajaran mesin dan statistik modern. Dengan prinsip sederhana — menemukan arah dengan variansi terbesar — PCA mampu mereduksi dimensi data sambil mempertahankan struktur informasi terpenting. Dalam praktiknya, PCA sering menjadi langkah pertama sebelum analisis lanjutan seperti clustering, classification, atau visualization.

" Principal Component Analysis (PCA) adalah teknik reduksi dimensi yang mengubah variabel asli menjadi sekumpulan komponen utama yang saling ortogonal dan menangkap sebagian besar variansi data. Metode ini membantu menyederhanakan data, mengurangi kebisingan, dan memudahkan visualisasi atau pemrosesan lebih lanjut tanpa kehilangan informasi penting".

BAGIAN 7
REINFORCEMENT LEARNING

7.1. Q-Learning

Q-Learning adalah salah satu algoritma paling dasar dan populer dalam reinforcement learning yang digunakan untuk mengajarkan agen (misalnya robot atau program) bagaimana mengambil tindakan terbaik dalam suatu lingkungan agar memperoleh reward (hadiah) setinggi mungkin. Inti dari Q-Learning adalah membangun sebuah tabel yang disebut Q-Table, di mana setiap baris merepresentasikan keadaan (state) dan setiap kolom merepresentasikan aksi (action) yang bisa dilakukan. Nilai dalam tabel ini disebut Q-Value, yaitu perkiraan seberapa bagus suatu aksi jika diambil pada keadaan tertentu.

Cara kerjanya bisa dijelaskan begini: agen mulai dengan Q-Table kosong atau acak. Setiap kali agen mencoba suatu aksi di sebuah state, ia akan mendapat reward dari lingkungan serta berpindah ke state berikutnya. Q-Learning kemudian memperbarui nilai Q-Value menggunakan persamaan Bellman, yaitu:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

Dimana:

- **s** = state sekarang
- **a** = aksi yang dipilih
- **r** = reward yang diterima
- **s^'** = state baru
- **α** = learning rate (seberapa besar pembaruan dilakukan)

γ = discount factor (seberapa jauh agen mempertimbangkan reward dimasa depan)

Intinya, Q-Learning belajar dengan cara mencoba-coba (trial and error), lalu memperbaiki perkiraannya tentang mana aksi yang menghasilkan reward lebih baik. Seiring waktu, tabel Q akan semakin akurat, dan agen dapat memilih aksi optimal dengan melihat aksi mana yang punya nilai Q tertinggi pada setiap state.

Sebagai contoh sederhana, jika agen adalah robot yang berada di labirin, Q-Learning akan membuatnya mencoba berbagai jalur. Meskipun awalnya tersesat, lama-lama robot akan belajar bahwa ada jalur tertentu yang lebih cepat menuju pintu keluar, karena Q-Value jalur tersebut semakin tinggi setelah banyak pengalaman.

Kelebihan Q-Learning adalah sederhana, mudah diimplementasikan, dan terbukti efektif untuk banyak masalah kecil. Namun, kelemahannya adalah Q-Table hanya cocok untuk lingkungan dengan jumlah state dan action yang terbatas. Jika ruang state terlalu besar (misalnya gambar atau sensor kamera), Q-Learning menjadi tidak praktis, sehingga biasanya dikembangkan lebih lanjut menjadi Deep Q-Network (DQN) yang menggunakan neural network sebagai pengganti Q-Table.

" Q-Learning adalah algoritma reinforcement learning berbasis nilai yang memungkinkan agen belajar strategi optimal dengan memperbarui fungsi Q, yaitu estimasi nilai tindakan dalam setiap state. Metode ini bersifat off-policy, sehingga agen dapat mempelajari nilai terbaik dari pengalaman yang dikumpulkan tanpa harus mengikuti strategi saat ini".

PROFILE PENULIS

PENULIS

Handoko Supeno lahir pada 29 Juni 1986. Ia merupakan pengajar, praktisi dan peneliti di bidang **kecerdasan buatan** dengan fokus pada ***machine learning, computer vision, reinforcement learning***, dan ***autonomous control systems***. Ia menyelesaikan pendidikan tinggi di bidang Teknik Informatika dan melanjutkan studi lanjutan pada program pascasarjana dengan konsentrasi **rekayasa kecerdasan buatan**.

Sebagai peneliti aktif, ia telah menghasilkan berbagai karya ilmiah, modul pembelajaran, dan buku yang dipublikasikan secara nasional maupun internasional. Topik yang digelutinya meliputi ***deep learning, behavior modeling, visual-based autonomous driving***, serta sistem otonom berbasis pembelajaran mesin.

Selain berkecimpung dalam riset, Handoko juga terlibat dalam berbagai pengembangan perangkat lunak, proyek teknologi kreatif, dan inisiatif edukasi publik. Ia sering diundang sebagai pembicara dalam seminar, workshop, dan diskusi teknologi untuk membagikan wawasan mengenai perkembangan terbaru AI dan aplikasinya dalam industri modern.

PENULIS

Muhammad Fauzan Dwi Putera adalah praktisi pemula di bidang **kecerdasan buatan** yang berfokus pada pengembangan model *machine learning* dan automasi sistem berbasis data. Ia aktif mengerjakan berbagai proyek implementasi seperti pengembangan ***chatbot***, sistem deteksi perilaku transaksi, dan model prediksi terapan.

Fauzan memiliki pengalaman mengikuti berbagai pelatihan profesional dan sertifikasi teknologi, serta terlibat dalam beberapa proyek kolaboratif yang berkaitan dengan edukasi dan implementasi AI. Minatnya meliputi pengembangan model ML modern, eksplorasi algoritma, serta pengembangan solusi teknologi yang aplikatif.

Dalam penyusunan buku ini, Fauzan berperan sebagai **kontributor konten teknis**, melengkapi materi dengan perspektif implementatif dan membantu penyusunan beberapa bagian fundamental terkait konsep pembelajaran mesin.

Penulis

Anissa Nursafitri adalah kreator digital yang berfokus pada pengembangan antarmuka, desain visual, serta eksplorasi teknologi **kecerdasan buatan**. Ia memiliki pengalaman dalam pembuatan konten kreatif digital, perancangan tampilan aplikasi, serta pengembangan materi berbasis teknologi.

Dalam proyek-proyek yang ia kerjakan, Anissa menggabungkan kemampuan teknis dan kreativitas untuk menghasilkan karya yang relevan dengan kebutuhan era digital. Bidang yang ia tekuni meliputi **frontend development, desain antarmuka aplikasi, visual content creation**, serta eksplorasi model kecerdasan buatan untuk berbagai keperluan kreatif.

Sebagai bagian dari tim penulis buku ini, Anissa berperan dalam **penyusunan contoh ilustrasi konsep, desain visual, serta penguatan narasi pembelajaran**, sehingga menjadikan buku ini lebih mudah dipahami dan menarik bagi pembaca dari berbagai latar belakang.