Get Press Indonesia

ISBN 978-634-255-398-5

KECERDASAN BUATAN DAN MACHINE LEARNING DENGAN PYTHON:

TEORI, ALGORITMA, DAN IMPLEMENTASI

Penulis:

Jarot Budiasto | Deddy Rudhistiar | Risanto Darmawan | Dwi Sasmita Aji Pambudi | Ronald Belferik | Chairi Nur Insani | Dedy Abdianto Nggego | Selfina Pare | Marsujitullah | Hasanudin Jayawardana | Tri Kustanti Rahayu | Agustan Latif | Muhammad Hasbi

KECERDASAN BUATAN DAN MACHINE LEARNING DENGAN PYTHON:

TEORI, ALGORITMA, DAN IMPLEMENTASI

Jarot Budiasto
Deddy Rudhistiar
Risanto Darmawan
Dwi Sasmita Aji Pambudi
Ronald Belferik
Chairi Nur Insani
Dedy Abdianto Nggego
Selfina Pare
Marsujitullah
Hasanudin Jayawardana
Tri Kustanti Rahayu
Agustan Latif
Muhammad Hasbi

GET PRESS INDONESIA

KECERDASAN BUATAN DAN MACHINE LEARNING DENGAN PYTHON: TEORI, ALGORITMA, DAN IMPLEMENTASI

Penulis :
Jarot Budiasto
Deddy Rudhistiar
Risanto Darmawan
Dwi Sasmita Aji Pambudi
Ronald Belferik
Chairi Nur Insani
Dedy Abdianto Nggego
Selfina Pare
Marsujitullah
Hasanudin Jayawardana
Tri Kustanti Rahayu
Agustan Latif
Muhammad Hasbi

ISBN : 978-634-255-398-5

Editor : Hendra Nusa Pratama, M.Kom.
Desain Sampul dan Tata Letak : Yuliatri Novita, M.Hum.

Penerbit : GET PRESS INDONESIA
Anggota IKAPI No. 033/SBA/2022
Jl. Palarik RT 01 RW 06 Kelurahan Air Pacah
Kecamatan Koto Tangah Padang Sumatera Barat
website: www.getpress.co.id
email: adm.getpress@gmail.com

Cetakan pertama, Maret 2026

KATA PENGANTAR

Puji dan syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa, yang telah memberikan rahmat dan karunia-Nya sehingga buku berjudul *Kecerdasan Buatan dan Machine Learning dengan Python: Teori, Algoritma, dan Implementasi* ini dapat diselesaikan. Isi buku ini mencakup pembahasan tentang pendahuluan kecerdasan buatan (AI), sejarah dan perkembangan ai, konsep dasar *machine learning,* lingkungan pemrograman python untuk AI dan ML, struktur data dan *library python (numpy, pandas, matplotlib*), pemrosesan data: pembersihan, transformasi, dan visualisasi, algoritma machine learning: supervised learning, algoritma *machine learning: unsupervised learning, deep learning: neural network* dan arsitektur dasar, *framework* AI : *tensorflow*, keras, dan pytorch, *natural language processing* (NLP) dengan python, *computer vision* dengan opencv dan deep learning, evaluasi model dan optimasi kinerja *machine learning,* dan aplikasi ai dalam berbagai bidang (kesehatan, keuangan, industri, pendidikan). Akhir kata, semoga buku ini dapat menjadi referensi dan dapat memberikan manfaat serta inspirasi bagi semua pembaca.

Padang, Februari 2026

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN KECERDASAN BUATAN (AI)

Perkembangan pesat teknologi informasi dan komputasi telah membawa perubahan mendasar dalam cara manusia memproses informasi, mengambil keputusan, dan menyelesaikan berbagai permasalahan kompleks (Russell and Norvig, 2021). Salah satu teknologi yang paling berpengaruh dalam perubahan tersebut adalah Kecerdasan Buatan (*Artificial Intelligence*/AI). AI tidak lagi sekadar konsep teoretis atau imajinasi fiksi ilmiah, melainkan telah hadir sebagai teknologi nyata yang digunakan secara luas dalam berbagai aspek kehidupan, mulai dari komunikasi, bisnis, kesehatan, pendidikan, hingga industri.

Kecerdasan Buatan memungkinkan sistem komputer untuk meniru kemampuan kognitif manusia, seperti belajar dari pengalaman, mengenali pola, memahami bahasa, serta mengambil keputusan berdasarkan data (Baskoro *et al.*, 2025; Nampira *et al.*, 2025). Kemampuan ini menjadikan AI sebagai fondasi utama dalam pengembangan sistem cerdas yang adaptif dan responsif terhadap perubahan lingkungan. Seiring dengan meningkatnya ketersediaan data dalam jumlah besar (*big data*) dan kemajuan komputasi berperforma tinggi, AI berkembang dengan sangat cepat dan menjadi tulang punggung transformasi digital di era modern.

Bab pendahuluan ini bertujuan untuk memberikan gambaran komprehensif mengenai konsep dasar Kecerdasan Buatan sebagai landasan sebelum memasuki pembahasan yang lebih teknis dan implementatif pada bab-bab berikutnya. Pembaca akan diperkenalkan pada definisi dan ruang lingkup AI, hubungan konseptual antara AI, *Machine Learning*, dan *Deep Learning*, serta peran strategis AI dalam mendorong inovasi dan efisiensi di berbagai sektor. Selain itu, bab ini juga menyajikan contoh penerapan AI dalam kehidupan sehari-hari serta mengulas tantangan dan peluang yang muncul dalam pengembangannya.

Dengan memahami konsep dasar yang dibahas dalam bab ini, pembaca diharapkan memiliki perspektif yang utuh mengenai pentingnya Kecerdasan Buatan dalam konteks teknologi dan masyarakat modern. Pemahaman ini menjadi bekal awal yang penting untuk mengikuti pembahasan selanjutnya, khususnya terkait algoritma, pemrograman Python, serta implementasi *Machine Learning* dan *Deep Learning* secara praktis.

1.1 Definisi dan Ruang Lingkup Kecerdasan Buatan

Kecerdasan Buatan (*Artificial Intelligence*/AI) secara umum didefinisikan sebagai cabang ilmu komputer yang berfokus pada perancangan dan pengembangan sistem cerdas, yaitu sistem yang mampu melakukan tugas-tugas yang biasanya membutuhkan kecerdasan manusia (McCarthy, 2007; Russell and Norvig, 2021). Tugas-tugas tersebut mencakup kemampuan untuk bernalar (*reasoning*), belajar dari pengalaman (*learning*), memecahkan masalah (*problem solving*), mengenali pola (*pattern recognition*), memahami bahasa alami (*natural language understanding*), serta mengambil keputusan secara mandiri berdasarkan data dan lingkungan.

John McCarthy, yang dikenal sebagai salah satu pelopor AI, mendefinisikan AI sebagai ilmu dan rekayasa dalam membuat mesin cerdas, khususnya program komputer yang cerdas (McCarthy, 2007). Definisi ini menekankan bahwa AI bukan sekadar hasil akhir berupa sistem pintar, tetapi juga mencakup proses ilmiah dan rekayasa dalam merancang kecerdasan tersebut. Sementara itu, Russell and Norvig (2021) memandang AI sebagai studi tentang agen rasional (*rational agents*), yaitu entitas yang mampu bertindak secara optimal untuk mencapai tujuan tertentu berdasarkan persepsi terhadap lingkungannya.

Berdasarkan pendekatan dan tujuannya, definisi AI dalam literatur dapat dikelompokkan ke dalam beberapa perspektif utama, antara lain:

1. **Pendekatan Berpikir Seperti Manusia**, yang berfokus pada bagaimana mesin meniru proses kognitif manusia.
2. **Pendekatan Bertindak Seperti Manusia**, yang menilai kecerdasan mesin dari kemampuannya meniru perilaku manusia.
3. **Pendekatan Berpikir Rasional**, yang menekankan penggunaan logika formal dan penalaran matematis.
4. **Pendekatan Bertindak Rasional**, yang menitikberatkan pada kemampuan sistem dalam mengambil keputusan terbaik.

Untuk memperjelas perbedaan keempat pendekatan tersebut, Tabel 1.1 berikut menyajikan klasifikasi pendekatan Kecerdasan Buatan beserta fokus utama dan contoh penerapannya.

Tabel 1. 1 Klasifikasi Pendekatan dalam Kecerdasan Buatan

Pendekatan AI	Fokus Utama	Karakteristik Utama	Contoh Implementasi
Berpikir seperti manusia *(thinking humanly)*	Proses kognitif manusia	Meniru cara manusia berpikir dan memecahkan masalah	Model kognitif, simulasi proses berpikir
Bertindak seperti manusia *(acting humanly)*	Perilaku manusia	Menilai kecerdasan dari perilaku yang tampak	Chatbot, sistem Turing Test
Berpikir rasional *(thinking rationally)*	Logika dan penalaran	Menggunakan aturan logika formal dan inferensi	Sistem berbasis logika, expert system
Bertindak rasional *(acting rationally)*	Pencapaian tujuan optimal	Agen memilih tindakan terbaik berdasarkan lingkungan	Agen cerdas, sistem rekomendasi, robot otonom

Tabel 1.1 menunjukkan bahwa pendekatan Kecerdasan Buatan dapat dipahami dari sudut pandang proses internal maupun perilaku eksternal, serta dari aspek kesesuaian dengan cara berpikir manusia atau prinsip rasionalitas. Dalam praktik AI modern, pendekatan bertindak rasional menjadi pendekatan yang paling dominan karena berfokus pada pencapaian tujuan secara optimal berdasarkan data dan lingkungan.

Ruang lingkup Kecerdasan Buatan bersifat sangat luas dan multidisipliner. AI tidak hanya berakar pada ilmu komputer, tetapi juga memanfaatkan konsep dari

matematika, statistika, ilmu saraf, psikologi kognitif, linguistik, filsafat, hingga etika (Guo and Han, 2020). Kolaborasi lintas disiplin ini memungkinkan AI berkembang sebagai bidang ilmu yang tidak hanya kuat secara teknis, tetapi juga relevan secara sosial dan humanistik.

Dalam praktiknya, ruang lingkup AI mencakup berbagai subbidang utama yang saling berkaitan, antara lain:

- ***Machine Learning***, yaitu pendekatan yang memungkinkan sistem belajar dari data dan meningkatkan kinerjanya secara otomatis.
- ***Natural Language Processing*** **(NLP)**, yang berfokus pada pemrosesan, analisis, dan pemahaman bahasa manusia.
- ***Computer Vision***, yang memungkinkan sistem memahami dan menafsirkan informasi visual dari citra dan video.
- ***Expert System***, yaitu sistem berbasis pengetahuan yang meniru proses pengambilan keputusan seorang pakar.
- ***Robotics***, yang mengintegrasikan AI dengan sistem fisik untuk menghasilkan perilaku cerdas di dunia nyata.

Selain subbidang tersebut, perkembangan AI modern juga mencakup area seperti sistem rekomendasi, agen cerdas, pemrosesan data skala besar, serta kecerdasan buatan generatif. Dengan cakupan yang luas dan aplikatif tersebut, Kecerdasan Buatan menjadi fondasi utama bagi berbagai inovasi teknologi modern dan memainkan peran strategis dalam membentuk masa depan masyarakat digital.

1.2 *Artificial Intelligence, Machine Learning,* dan *Deep Learning:* Hubungan Konseptual

Dalam perkembangan teknologi modern, istilah *Artificial Intelligence* (AI), *Machine Learning* (ML), dan *Deep Learning* (DL) sering digunakan secara bersamaan dan bahkan dianggap memiliki makna yang sama. Padahal, ketiganya memiliki ruang lingkup, pendekatan, dan karakteristik yang berbeda, meskipun saling berkaitan erat (Ashenden *et al.*, 2021). Pemahaman yang tepat mengenai hubungan konseptual antara AI, *Machine Learning*, dan *Deep Learning* menjadi sangat penting sebagai landasan teoritis sebelum mempelajari algoritma dan implementasi teknis menggunakan Python.

Secara konseptual, *Artificial Intelligence* merupakan bidang paling luas yang mencakup seluruh upaya untuk menciptakan sistem atau mesin yang memiliki kemampuan cerdas. AI bertujuan untuk meniru atau mensimulasikan perilaku cerdas manusia, seperti berpikir, bernalar, belajar, dan mengambil keputusan. Pendekatan AI pada tahap awal banyak didominasi oleh metode simbolik, seperti sistem berbasis aturan (*rule-based systems*) dan sistem pakar (*expert systems*), di mana pengetahuan manusia direpresentasikan secara eksplisit dalam bentuk aturan logika.

Machine Learning muncul sebagai pendekatan yang lebih adaptif dalam pengembangan AI. Berbeda dengan sistem AI tradisional yang sangat bergantung pada aturan yang dirancang secara manual, *Machine Learning* memungkinkan sistem untuk belajar secara otomatis dari data. Dalam ML, model dibangun dengan memanfaatkan data historis untuk menemukan pola, hubungan, atau struktur tertentu, kemudian menggunakan pola tersebut untuk melakukan prediksi atau pengambilan keputusan pada data baru (Budiasto, Tallulembang, *et al.*, 2025; Narad *et al.*, 2025). Dengan demikian, *Machine Learning* dapat dipandang sebagai

jembatan antara AI konseptual dan implementasi praktis berbasis data.

Seiring dengan meningkatnya ketersediaan data dalam jumlah besar serta kemajuan komputasi, terutama penggunaan GPU dan komputasi paralel, berkembanglah *Deep Learning* sebagai subbidang khusus dari *Machine Learning*. *Deep Learning* menggunakan jaringan saraf tiruan berlapis banyak (*deep neural networks*) yang terinspirasi dari struktur dan cara kerja otak manusia. Arsitektur berlapis ini memungkinkan model *Deep Learning* untuk mempelajari representasi data yang sangat kompleks dan abstrak, sehingga sangat efektif untuk tugas-tugas seperti pengenalan citra, pemrosesan bahasa alami, dan pengenalan suara (Mandowen *et al.*, 2025; Narad *et al.*, 2025).

Hubungan antara AI, *Machine Learning*, dan *Deep Learning* dapat dipahami sebagai hubungan hierarkis. *Artificial Intelligence* berada pada tingkat paling atas sebagai konsep dan tujuan umum. *Machine Learning* berada di dalam AI sebagai pendekatan pembelajaran berbasis data, sedangkan *Deep Learning* berada di dalam *Machine Learning* sebagai teknik pembelajaran yang menggunakan jaringan saraf dalam skala besar (Ashenden *et al.*, 2021; Zhao and Liu, 2024). Dengan kata lain, setiap *Deep Learning* adalah *Machine Learning*, dan setiap *Machine Learning* merupakan bagian dari *Artificial Intelligence*, tetapi tidak semua AI menggunakan *Machine Learning*, dan tidak semua *Machine Learning* menggunakan *Deep Learning*.

Perbedaan fokus antara ketiganya dapat dirangkum sebagai berikut:

- ***Artificial Intelligence*** berfokus pada tujuan menciptakan sistem yang dapat bertindak cerdas dan rasional.
- ***Machine Learning*** berfokus pada metode dan algoritma yang memungkinkan sistem belajar dari data.

- ***Deep Learning*** berfokus pada arsitektur jaringan saraf berlapis dalam untuk mempelajari pola data yang kompleks.

Untuk memberikan gambaran yang lebih sistematis, Tabel 1.2 berikut menyajikan perbandingan antara *Artificial Intelligence*, *Machine Learning*, dan *Deep Learning* berdasarkan fokus, karakteristik data, pendekatan algoritmik, dan contoh penerapannya.

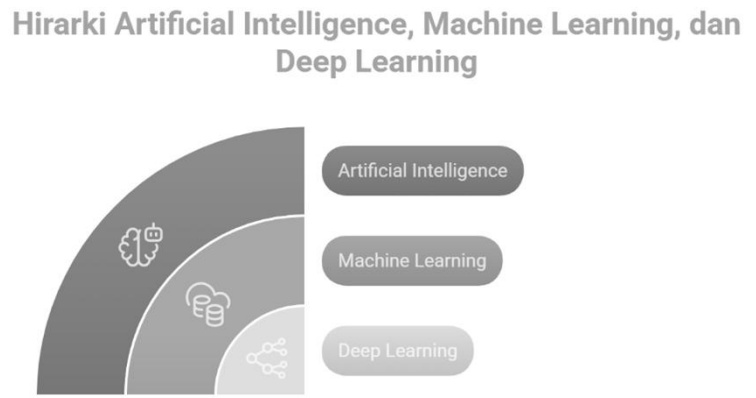
Tabel 1. 2 Perbandingan *Artificial Intelligence, Machine Learning,* dan *Deep Learning*

Aspek	***Artificial Intelligence*** **(AI)**	***Machine Learning*** **(ML)**	***Deep Learning*** **(DL)**
Fokus utama	Menciptakan sistem cerdas yang dapat bernalar dan mengambil keputusan	Membuat sistem yang dapat belajar dari data	Mempelajari representasi data kompleks menggunakan jaringan saraf berlapis
Ketergantungan data	Tidak selalu bergantung pada data dalam jumlah besar	Sangat bergantung pada data historis	Sangat bergantung pada data besar dan berkualitas tinggi
Pendekatan algoritma	Rule-based system, logika simbolik, agen rasional	Regresi, klasifikasi, clustering, decision tree, SVM	Neural network, CNN, RNN, Transformer

Aspek	*Artificial Intelligence* (AI)	*Machine Learning* (ML)	*Deep Learning* (DL)
Kompleksitas model	Relatif rendah hingga menengah	Menengah	Tinggi
Kebutuhan komputasi	Rendah hingga menengah	Menengah	Tinggi (GPU/TPU)

Tabel 1.2 menunjukkan bahwa perbedaan utama antara AI, *Machine Learning*, dan *Deep Learning* terletak pada fokus pendekatan, ketergantungan terhadap data, serta kompleksitas algoritma dan komputasi yang digunakan. Tabel ini juga menegaskan bahwa *Deep Learning* merupakan pendekatan yang paling kompleks namun sangat efektif untuk menangani permasalahan dengan data berskala besar dan pola yang kompleks.

Sebagai representasi visual dari hubungan tersebut, Gambar 1.1 menampilkan hierarki konseptual ketiga domain ini dalam struktur bertingkat. *Artificial Intelligence* (AI) berada pada lapisan terluar sebagai payung besar yang mencakup seluruh konsep dan pendekatan kecerdasan buatan. Di dalam AI terdapat *Machine Learning* (ML) sebagai pendekatan yang memungkinkan sistem belajar dari data, sedangkan *Deep Learning* (DL) berada di dalam *Machine Learning* sebagai teknik pembelajaran berbasis jaringan saraf berlapis banyak.



Gambar 1. 1 Hirarki *Artificial Intelligence, Machine Learning,* dan *Deep Learning*

Diagram hirarki ini membantu pembaca memahami posisi relatif berbagai teknik dan algoritma yang akan dibahas dalam buku ini. Dengan visualisasi tersebut, pembaca dapat lebih mudah mengaitkan konsep teoretis dengan implementasi praktis, serta memahami mengapa bab-bab selanjutnya difokuskan secara bertahap mulai dari *Machine Learning* hingga *Deep Learning*.

Dalam konteks buku ini, pembahasan *Artificial Intelligence* digunakan sebagai kerangka konseptual, sementara *Machine Learning* dan *Deep Learning* menjadi fokus utama dalam aspek algoritmik dan implementatif. Python dipilih sebagai bahasa pemrograman utama karena ekosistem pustakanya yang kaya dan dukungan luas terhadap pengembangan *Machine Learning* dan *Deep Learning* (Budiasto, Ismanto, *et al.*, 2025). Dengan memahami hubungan konseptual ketiga istilah ini, pembaca diharapkan mampu melihat posisi setiap algoritma dan teknik yang akan dipelajari pada bab-bab selanjutnya secara lebih sistematis dan terstruktur.

1.3 Peran AI dalam Transformasi Digital

Transformasi digital merupakan proses integrasi teknologi digital ke dalam seluruh aspek organisasi, proses bisnis, dan kehidupan sosial yang bertujuan untuk meningkatkan efisiensi, nilai tambah, serta kemampuan adaptasi terhadap perubahan lingkungan (Vial, 2021). Dalam konteks ini, Kecerdasan Buatan (AI) memainkan peran yang sangat strategis karena kemampuannya dalam mengolah data, belajar dari pola, serta menghasilkan rekomendasi dan keputusan secara cerdas. AI tidak hanya berfungsi sebagai teknologi pendukung, tetapi juga menjadi *enabler* utama yang mendorong lahirnya model bisnis, layanan, dan sistem kerja baru di era digital (Nalurita *et al.*, 2025).

Salah satu peran utama AI dalam transformasi digital adalah kemampuannya dalam mengolah dan menganalisis data dalam jumlah besar secara cepat dan akurat. Era digital ditandai dengan ledakan data (*big data*) yang berasal dari berbagai sumber, seperti media sosial, sensor IoT, transaksi digital, dan sistem informasi organisasi. AI, khususnya melalui *Machine Learning* dan *Deep Learning*, memungkinkan data tersebut diubah menjadi informasi dan pengetahuan yang bernilai untuk mendukung pengambilan keputusan strategis (Budiasto, Purnama, *et al.*, 2025). Tanpa AI, pemanfaatan data berskala besar akan menjadi sangat terbatas dan tidak efisien.

Selain itu, AI berperan penting dalam otomasi proses bisnis. Berbeda dengan otomasi tradisional yang bersifat statis dan berbasis aturan sederhana, otomasi berbasis AI bersifat adaptif dan mampu belajar dari data historis. Hal ini memungkinkan organisasi untuk mengotomatisasi proses yang kompleks, seperti pemrosesan dokumen, layanan pelanggan melalui chatbot, deteksi anomali, serta penjadwalan dan perencanaan operasional (Nampira *et al.*, 2025; Singh *et al.*, 2025). Dengan adanya AI, organisasi dapat

meningkatkan produktivitas, mengurangi kesalahan manusia, serta menekan biaya operasional secara signifikan (Yaakub *et al.*, 2025).

Peran strategis AI juga terlihat dalam peningkatan kualitas pengambilan keputusan. Sistem berbasis AI mampu memberikan rekomendasi yang didasarkan pada analisis data multidimensi dan simulasi berbagai skenario (Budiasto, Ketrin, *et al.*, 2025). Dalam konteks bisnis, AI digunakan untuk mendukung keputusan terkait pemasaran, manajemen risiko, peramalan permintaan, dan strategi investasi. Dalam sektor publik, AI dimanfaatkan untuk perencanaan kota cerdas (*smart city*), manajemen lalu lintas, dan pengambilan kebijakan berbasis data. Dengan demikian, AI membantu pengambil keputusan dalam menghasilkan keputusan yang lebih objektif, akurat, dan tepat waktu.

Di sisi lain, AI juga mendorong transformasi digital melalui personalisasi layanan. Dengan memanfaatkan data perilaku pengguna, AI mampu menyesuaikan produk, layanan, dan pengalaman pengguna secara individual (Govindaraj *et al.*, 2025; Impron *et al.*, 2025). Contoh penerapan personalisasi berbasis AI dapat ditemukan pada sistem rekomendasi e-commerce, platform pembelajaran daring adaptif, serta layanan kesehatan berbasis data pasien. Personalisasi ini tidak hanya meningkatkan kepuasan pengguna, tetapi juga menciptakan nilai tambah dan keunggulan kompetitif bagi organisasi.

Dalam skala yang lebih luas, AI berkontribusi pada transformasi digital lintas sektor. Di bidang industri, AI mendukung konsep *Industry 4.0* melalui sistem manufaktur cerdas dan pemeliharaan prediktif. Di bidang kesehatan, AI digunakan untuk diagnosis berbasis citra medis dan analisis data klinis. Di bidang pendidikan, AI memungkinkan pembelajaran yang lebih adaptif dan berbasis kebutuhan individu (Yaakub *et al.*, 2025). Integrasi AI dalam berbagai sektor tersebut menunjukkan bahwa transformasi digital

bukan hanya perubahan teknologi, melainkan juga perubahan cara berpikir, bekerja, dan berinteraksi.

Dengan demikian, peran AI dalam transformasi digital bersifat fundamental dan multidimensional. AI tidak hanya menjadi alat bantu teknis, tetapi juga elemen strategis yang mendorong inovasi, meningkatkan daya saing, serta membentuk ekosistem digital yang lebih cerdas dan berkelanjutan di era modern.

Tabel 1.3 berikut merangkum peran utama Kecerdasan Buatan dalam mendorong transformasi digital di berbagai sektor, termasuk bentuk penerapan AI serta dampak strategis yang dihasilkan.

Tabel 1. 3 Peran AI dalam Transformasi Digital Lintas Sektor

Sektor	Peran Utama AI	Contoh Penerapan	Dampak Transformasi
Industri & Manufaktur	Otomasi cerdas dan prediksi operasional	Predictive maintenance, robot industri, quality inspection berbasis vision	Peningkatan efisiensi produksi, pengurangan downtime, dan kualitas produk yang lebih konsisten
Bisnis & Keuangan	Analisis data dan pengambilan keputusan	Deteksi fraud, credit scoring, peramalan permintaan, trading algoritmik	Keputusan lebih akurat, manajemen risiko yang lebih baik, dan peningkatan profitabilitas

Sektor	Peran Utama AI	Contoh Penerapan	Dampak Transformasi
Kesehatan	Analisis data medis dan diagnosis	Deteksi penyakit dari citra medis, analisis rekam medis, sistem triase cerdas	Diagnosis lebih cepat dan akurat, peningkatan kualitas layanan kesehatan
Pendidikan	Pembelajaran adaptif dan evaluasi otomatis	Learning analytics, sistem rekomendasi materi, penilaian otomatis	Pembelajaran personal, peningkatan efektivitas proses belajar mengajar
Pemerintahan & Publik	Layanan publik berbasis data	Smart city, manajemen lalu lintas, pelayanan administrasi digital	Pelayanan publik lebih efisien, transparan, dan responsif
Transportasi & Logistik	Optimasi rute dan sistem otonom	Navigasi cerdas, kendaraan otonom, optimasi rantai pasok	Efisiensi waktu dan biaya, peningkatan keselamatan
E-commerce & Layanan Digital	Personalisasi layanan dan pengalaman pengguna	Sistem rekomendasi, chatbot layanan pelanggan	Peningkatan kepuasan pengguna dan loyalitas pelanggan

Tabel 1.3 menunjukkan bahwa peran AI dalam transformasi digital bersifat lintas sektor dan berdampak langsung pada peningkatan efisiensi, kualitas layanan, serta kemampuan adaptasi organisasi terhadap perubahan lingkungan digital.

1.4 Contoh Aplikasi AI dalam Kehidupan Sehari-hari

Kehadiran Kecerdasan Buatan dalam kehidupan sehari-hari semakin terasa seiring dengan meningkatnya digitalisasi layanan dan aktivitas manusia. Berbagai aplikasi yang digunakan masyarakat modern saat ini telah memanfaatkan AI sebagai komponen inti untuk meningkatkan kenyamanan, efisiensi, serta kualitas layanan (Baskoro *et al.*, 2025). Menariknya, sebagian besar penerapan AI tersebut bekerja di balik layar sehingga sering kali tidak disadari secara langsung oleh pengguna.

Salah satu contoh paling umum adalah asisten virtual berbasis suara, seperti Google Assistant, Siri, dan Amazon Alexa. Sistem ini memanfaatkan kombinasi *Natural Language Processing* (NLP), speech recognition, dan *Machine Learning* untuk memahami perintah pengguna dalam bahasa alami, menafsirkan maksudnya, serta memberikan respons yang relevan (Yadav *et al.*, 2023; Supriyono *et al.*, 2025). Asisten virtual digunakan untuk berbagai keperluan sederhana, seperti mengatur pengingat, mencari informasi, mengirim pesan, hingga mengendalikan perangkat pintar di rumah (*smart home*).

Aplikasi AI lainnya yang sangat dekat dengan kehidupan sehari-hari adalah sistem rekomendasi. Platform e-commerce, media sosial, dan layanan streaming seperti marketplace daring, YouTube, Netflix, dan Spotify menggunakan algoritma *Machine Learning* untuk

menganalisis perilaku pengguna, seperti riwayat pencarian, klik, dan waktu interaksi (Velankar and Kulkarni, 2023; Budiasto, Tallulembang, *et al.*, 2025). Berdasarkan analisis tersebut, sistem kemudian merekomendasikan produk, konten, atau layanan yang paling sesuai dengan preferensi masing-masing pengguna. Sistem rekomendasi ini tidak hanya meningkatkan pengalaman pengguna, tetapi juga berkontribusi besar terhadap strategi pemasaran dan peningkatan pendapatan penyedia layanan.

Dalam aspek keamanan dan autentikasi, AI banyak diterapkan melalui teknologi pengenalan wajah dan biometrik. Fitur pembukaan kunci ponsel menggunakan wajah atau sidik jari merupakan contoh nyata penerapan *Computer Vision* dan *Machine Learning*. Teknologi ini juga digunakan pada sistem pengawasan, absensi otomatis, serta manajemen identitas digital (Wairagade and Das, 2025). Dengan tingkat akurasi yang semakin tinggi, sistem biometrik berbasis AI mampu meningkatkan keamanan sekaligus kenyamanan pengguna.

AI juga memainkan peran penting dalam sistem navigasi dan transportasi. Aplikasi navigasi seperti Google Maps dan Waze memanfaatkan AI untuk menganalisis data lalu lintas secara real-time, memprediksi kemacetan, serta merekomendasikan rute tercepat. Lebih lanjut, pengembangan kendaraan otonom memanfaatkan AI untuk mengenali lingkungan sekitar, seperti rambu lalu lintas, kendaraan lain, dan pejalan kaki, serta mengambil keputusan secara mandiri. Meskipun kendaraan otonom masih dalam tahap pengembangan di banyak negara, teknologi ini menunjukkan potensi besar dalam meningkatkan keselamatan dan efisiensi transportasi di masa depan (Budiasto, Jayawardana, *et al.*, 2025).

Dalam bidang keuangan dan keamanan digital, AI digunakan untuk deteksi penipuan dan perlindungan siber. Sistem berbasis AI mampu mempelajari pola transaksi

normal dan mendeteksi aktivitas yang menyimpang secara otomatis (Yaakub *et al.*, 2025). Hal ini memungkinkan lembaga keuangan dan penyedia layanan digital untuk mengidentifikasi potensi penipuan, pencurian identitas, atau serangan siber secara lebih cepat dan akurat dibandingkan metode konvensional.

Selain contoh-contoh tersebut, AI juga hadir dalam berbagai aplikasi lain, seperti penyaring spam pada email, koreksi otomatis pada pengetikan, pengenalan objek pada kamera ponsel, serta chatbot layanan pelanggan. Beragam penerapan ini menunjukkan bahwa AI telah menjadi bagian integral dari ekosistem teknologi modern dan berkontribusi langsung terhadap peningkatan kualitas hidup masyarakat (Baskoro *et al.*, 2025). Seiring dengan perkembangan teknologi dan meningkatnya adopsi AI, peran AI dalam kehidupan sehari-hari diperkirakan akan semakin luas dan signifikan di masa mendatang.

Untuk memberikan gambaran yang lebih terstruktur mengenai penerapan Kecerdasan Buatan dalam kehidupan sehari-hari, Tabel 1.4 berikut merangkum berbagai contoh aplikasi AI beserta teknologi utama yang digunakan dan fungsi utamanya.

Tabel 1. 4 Peran AI dalam Transformasi Digital Lintas Sektor

Aplikasi AI	Bidang Penerapan	Teknologi yang Digunakan	Fungsi Utama
Asisten virtual (Siri, Google Assistant, Alexa)	Layanan digital & smart home	NLP, Speech Recognition, Machine Learning	Memahami perintah suara dan memberikan respons cerdas
Sistem rekomendasi	Bisnis digital & hiburan	Machine Learning,	Merekomendasikan produk atau

Aplikasi AI	Bidang Penerapan	Teknologi yang Digunakan	Fungsi Utama
(E-commerce, Streaming)		Collaborati ve Filtering	konten sesuai preferensi pengguna
Pengenalan wajah & biometrik	Keamanan & autentikasi	Computer Vision, Deep Learning (CNN)	Identifikasi dan verifikasi identitas pengguna
Navigasi cerdas (Googl e Maps, Waze)	Transportasi	Machine Learning, Analisis data real-time	Prediksi kemacetan dan rekomendasi rute optimal
Deteksi penipuan transaksi	Keuangan & perbankan	Machine Learning, Anomaly Detection	Mengidentifikasi transaksi mencurigakan secara otomatis
Filter spam email	Komunikasi digital	Machine Learning, NLP	Menyaring email tidak diinginkan atau berbahaya
Chatbot layanan pelanggan	Bisnis & layanan publik	NLP, Machine Learning	Menjawab pertanyaan dan melayani pengguna secara otomatis
Kamera cerdas pada ponsel	Multimedia	Computer Vision, Deep Learning	Pengenalan objek, wajah, dan peningkatan kualitas foto

Tabel 1.4 menunjukkan bahwa satu aplikasi AI umumnya memanfaatkan kombinasi beberapa teknologi kecerdasan buatan. Hal ini menegaskan bahwa AI bersifat multidisipliner dan integratif, di mana *Machine Learning*, *Deep Learning*, NLP, dan *Computer Vision* saling melengkapi untuk menghasilkan sistem yang cerdas dan fungsional.

1.5 Tantangan dan Peluang Pengembangan AI

Perkembangan Kecerdasan Buatan yang sangat pesat dalam beberapa tahun terakhir membawa dampak signifikan bagi berbagai sektor kehidupan. Di satu sisi, AI menawarkan potensi besar untuk meningkatkan efisiensi, akurasi, dan kualitas pengambilan keputusan. Namun di sisi lain, pengembangan dan penerapan AI juga dihadapkan pada berbagai tantangan teknis, sosial, dan etis yang tidak dapat diabaikan. Oleh karena itu, pemahaman yang seimbang mengenai tantangan dan peluang pengembangan AI menjadi sangat penting agar teknologi ini dapat dimanfaatkan secara optimal dan bertanggung jawab.

1.5.1 Tantangan dalam Pengembangan AI

Salah satu tantangan utama dalam pengembangan AI adalah kualitas dan ketersediaan data. Model AI, khususnya yang berbasis *Machine Learning* dan *Deep Learning*, sangat bergantung pada data dalam jumlah besar yang berkualitas tinggi. Data yang tidak lengkap, tidak akurat, atau tidak representatif dapat menyebabkan model menghasilkan prediksi yang keliru (Mohammed *et al.*, 2025; Nampira *et al.*, 2025). Selain itu, di banyak sektor, data yang relevan masih tersebar, tidak terstruktur, atau dibatasi oleh regulasi, sehingga menyulitkan proses pengumpulan dan pemanfaatannya secara optimal.

Tantangan berikutnya adalah bias dan keadilan algoritma. Model AI belajar dari data historis, sehingga jika data tersebut mengandung bias sosial, ekonomi, atau budaya, maka bias tersebut berpotensi direplikasi bahkan diperkuat oleh sistem AI (Kumar *et al.*, 2025). Hal ini dapat berdampak pada ketidakadilan dalam berbagai aplikasi, seperti sistem rekrutmen, penilaian kredit, atau penegakan hukum berbasis teknologi (A Sani *et al.*, 2025). Oleh karena itu, isu fairness, transparency, dan explainability menjadi fokus penting dalam pengembangan AI modern.

Aspek keamanan dan privasi data juga menjadi tantangan krusial. Pemanfaatan AI sering kali melibatkan pengolahan data sensitif, seperti data pribadi, data kesehatan, atau data keuangan (Asrul Sani *et al.*, 2025). Risiko kebocoran data, penyalahgunaan informasi, serta serangan siber terhadap sistem AI menuntut adanya mekanisme perlindungan yang kuat. Selain itu, model AI sendiri dapat menjadi target serangan, seperti *adversarial attacks*, yang dapat menurunkan keandalan dan keamanan sistem.

Tantangan lain yang tidak kalah penting adalah isu etika dan tanggung jawab hukum. Keputusan yang dihasilkan oleh sistem AI, terutama dalam konteks kritis seperti kesehatan, transportasi, dan keuangan, menimbulkan pertanyaan mengenai siapa yang bertanggung jawab jika terjadi kesalahan. Kurangnya kerangka hukum dan regulasi yang jelas di banyak negara membuat penerapan AI sering kali berada di wilayah abu-abu secara etis dan legal. Hal ini menuntut keterlibatan berbagai pemangku kepentingan dalam merumuskan prinsip dan kebijakan AI yang bertanggung jawab.

1.5.2 Peluang Pengembangan AI

Di balik berbagai tantangan tersebut, peluang pengembangan AI terbuka sangat luas. Salah satu peluang terbesar adalah kemajuan infrastruktur komputasi, seperti komputasi awan (*cloud computing*), GPU, dan teknologi komputasi terdistribusi. Infrastruktur ini memungkinkan pengolahan data dan pelatihan model AI berskala besar yang sebelumnya sulit dilakukan, sehingga mempercepat inovasi dan adopsi AI di berbagai sektor (El Morr *et al.*, 2022).

Selain itu, ketersediaan *framework* dan pustaka *open-source* seperti TensorFlow, PyTorch, Scikit-learn, dan Keras memberikan peluang besar bagi pengembang, peneliti, dan praktisi untuk membangun dan mengimplementasikan solusi AI secara lebih cepat dan efisien (Budiasto, Tallulembang, *et al.*, 2025). Ekosistem *open-source* ini juga mendorong kolaborasi global dan transfer pengetahuan, sehingga mempercepat perkembangan teknologi AI secara kolektif.

Peluang lainnya terletak pada kebutuhan sumber daya manusia yang kompeten di bidang AI. Permintaan terhadap talenta yang memiliki kemampuan dalam data science, *Machine Learning*, dan AI terus meningkat di berbagai sektor industri dan akademik. Hal ini membuka peluang besar bagi institusi pendidikan untuk mengembangkan kurikulum berbasis AI serta bagi individu untuk meningkatkan kompetensi dan daya saing di pasar kerja global (Su and Zhong, 2022).

Lebih jauh, AI memiliki potensi besar untuk berkontribusi dalam penyelesaian tantangan global, seperti peningkatan layanan kesehatan, mitigasi perubahan iklim, optimalisasi energi, dan peningkatan kualitas pendidikan. Dengan pemanfaatan data dan algoritma yang tepat, AI dapat membantu manusia

dalam merancang solusi yang lebih efektif dan berkelanjutan terhadap permasalahan kompleks yang dihadapi dunia saat ini (Baskoro *et al.*, 2025).

Dengan pendekatan yang bertanggung jawab, transparan, dan beretika, pengembangan AI tidak hanya menjadi peluang teknologi, tetapi juga peluang sosial dan ekonomi yang strategis. Oleh karena itu, tantangan yang ada seharusnya tidak menjadi penghambat, melainkan menjadi dasar untuk merancang sistem AI yang lebih adil, aman, dan bermanfaat bagi masyarakat luas.

Tabel 1.5 berikut merangkum secara sistematis berbagai tantangan utama yang dihadapi dalam pengembangan AI serta peluang strategis yang dapat dimanfaatkan untuk mendorong inovasi dan pemanfaatan AI secara berkelanjutan.

Tabel 1. 5 Tantangan dan Peluang Pengembangan AI

Aspek	Tantangan Utama	Peluang Strategis
Data	Keterbatasan data berkualitas, data tidak representatif, isu privasi	Pemanfaatan big data, data terbuka, dan teknik augmentasi data
Algoritma	Bias algoritma, kurangnya transparansi model	Pengembangan AI yang adil (fair AI) dan explainable AI
Infrastruktur	Kebutuhan komputasi tinggi dan biaya implementasi	Cloud computing, GPU, dan komputasi terdistribusi
Keamanan	Risiko kebocoran data dan serangan siber	Pengembangan AI security dan privacy-preserving AI
Etika & Regulasi	Ketidakjelasan tanggung jawab hukum dan etika	Penyusunan regulasi AI dan prinsip AI bertanggung jawab

Aspek	Tantangan Utama	Peluang Strategis
SDM	Keterbatasan tenaga ahli AI	Peningkatan pendidikan, pelatihan, dan literasi AI
Dampak Sosial	Kekhawatiran penggantian tenaga kerja	Transformasi pekerjaan dan lahirnya profesi baru

Tabel 1.5 memberikan gambaran bahwa tantangan dan peluang dalam pengembangan AI bersifat saling terkait. Tantangan yang ada tidak selalu menjadi penghambat, melainkan dapat menjadi pemicu lahirnya inovasi, kebijakan, dan pendekatan baru yang lebih matang dalam pengembangan dan penerapan Kecerdasan Buatan.

DAFTAR PUSTAKA

Ashenden, S.K. *et al.* (2021) 'Introduction to artificial intelligence and machine learning', in *The Era of Artificial Intelligence, Machine Learning, and Data Science in the Pharmaceutical Industry*, pp. 15–26. Available at: https://doi.org/10.1016/B978-0-12-820045-2.00003-9.

Baskoro, S.E. *et al.* (2025) *Kecerdasan Buatan dalam Kehidupan Nyata: Konsep, Algoritma dan Penerapan*. Star Digital Publishing. Available at: https://books.google.co.id/books?id=tGWGEQAAQBAJ.

Budiasto, J., Ketrin, D., *et al.* (2025) *DATA ANALYTICS: MENJADI AHLI DALAM MENGELOLAH DAN MENGANALISIS DATA*. Get Press Indonesia.

Budiasto, J., Purnama, S.G., *et al.* (2025) *Data Science Technology*. PT. Star Digital Publishing, Yogyakarta-Indonesia. Available at: https://books.google.co.id/books?id=3fxlEQAAQBAJ.

Budiasto, J., Tallulembang, T.M., *et al.* (2025) *Machine Learning untuk Pemula: Konsep dan Implementasi*. Edited by T.Y. Zalni. Jakarta Utara: Penerbit Buku Indonesia.

Budiasto, J., Jayawardana, H., *et al.* (2025) *Pengantar Ilmu Komputer: Pengenalan Dasar Komputer dan Teknologi Informasi Modern*. Yogyakarta: PT. Star Digital Publishing, Yogyakarta-Indonesia. Available at: https://books.google.co.id/books?id=JWNfEQAAQBAJ.

Budiasto, J., Ismanto, H., *et al.* (2025) *Python untuk Data Science: Panduan Praktis Menguasai Analisis Data*. Edited by W. Yuliani. Padang: Literasi Langsung Terbit.

Govindaraj, M. *et al.* (2025) 'Revolutionizing consumer engagement AI-driven personalization in modern marketing', in *AI Marketing and Ethical Considerations in Consumer Engagement*, pp. 123–142. Available at: https://doi.org/10.4018/979-8-3373-3476-9.ch007.

Guo, T. and Han, C. (2020) 'Artificial intelligence mechanism and mathematics implementation methods', *Scientia Sinica Mathematica*, 50(11), pp. 1541–1578. Available at: https://doi.org/10.1360/SSM-2020-0031.

Impron, A. *et al.* (2025) *MACHINE LEARNING: Konsep, Implementasi, dan Aplikasi*. Padang: Get Press Indonesia. Available at: www.getpress.co.id.

Kumar, A. *et al.* (2025) 'A Comprehensive Review of Bias in AI, ML, and DL Models: Methods, Impacts, and Future Directions', *Archives of Computational Methods in Engineering* [Preprint]. Available at: https://doi.org/10.1007/s11831-025-10483-6.

Mandowen, S.A. *et al.* (2025) *Pengantar Machine Learning: Teori, Algoritma, dan Implementasi Modern*. Available at: https://books.google.co.id/books?id=EOadEQAAQBAJ.

McCarthy, J. (2007) 'WHAT IS ARTIFICIAL INTELLIGENCE?' Available at: https://www-formal.stanford.edu/jmc/whatisai.pdf (Accessed: 20 June 2025).

Mohammed, S. *et al.* (2025) 'The effects of data quality on machine learning performance on tabular data', *Information Systems*, 132. Available at: https://doi.org/10.1016/j.is.2025.102549.

El Morr, C. *et al.* (2022) 'Future Directions and Ethical Considerations', in *International Series in Operations Research and Management Science*, pp. 449–460. Available at: https://doi.org/10.1007/978-3-031-16990-8_16.

Nalurita, F. *et al.* (2025) *Bisnis 5.0: Rancangan, Peluang dan Strategi Inovatif Dunia Bisnis dalam Menyongsong Era Society 5.0*. PT. Sonpedia Publishing Indonesia.

Nampira, A.A. *et al.* (2025) *Artificial Intelligence: A Guide for Thinking Humans*. PT. Green Pustaka Indonesia. Available at: https://books.google.co.id/books?id=_Y95EQAAQBAJ.

Narad, P. *et al.* (2025) 'Comparison of Machine Learning and Deep Learning', in *Artificial Intelligence and Biological Sciences*, pp. 111–120. Available at: https://www.scopus.com/inward/record.uri?eid=2-s2.0-105002529685&partnerID=40&md5=b4cf4faddd9084fb54539cdc2d7c38b0.

Russell, S. and Norvig, P. (2021) *Artificial Intelligence: A Modern Approach, Global Edition*. Pearson Education. Available at: https://books.google.co.id/books?id=cb0qEAAAQBAJ.

Sani, Asrul *et al.* (2025) *Mengenal Teknologi Informasi: Panduan Lengkap untuk Pemula*. Edited by A. Yanto. Padang: Get Press Indonesia.

Sani, A *et al.* (2025) *PENGANTAR TEKNOLOGI INFORMASI: Dampak dan Peluang Perkembangan Teknologi Informasi dalam Dunia Kerja dan Bisnis*. Edited by D.F. Romiza. Yogyakarta: PT. Star Digital Publishing, Yogyakarta-Indonesia. Available at:

https://books.google.co.id/books?id=EOVhEQAAQBAJ.

Singh, A.K. *et al.* (2025) 'The role of AI in digital transformation', in *Human-Centric AI in Digital Transformation and Entrepreneurship*, pp. 109–136. Available at: https://doi.org/10.4018/979-8-3693-8009-3.ch006.

Su, J. and Zhong, Y. (2022) 'Artificial Intelligence (AI) in early childhood education: Curriculum design and future directions', *Computers and Education: Artificial Intelligence*, 3. Available at: https://doi.org/10.1016/j.caeai.2022.100072.

Supriyono, L.A. *et al.* (2025) *Buku Ajar Pengantar Ilmu Komputer*. Available at: https://books.google.co.id/books?id=EqBxEQAAQBAJ.

Velankar, M. and Kulkarni, P. (2023) 'Music Recommendation Systems: Overview and Challenges', in *Signals and Communication Technology*, pp. 51–69. Available at: https://doi.org/10.1007/978-3-031-18444-4_3.

Vial, G. (2021) 'Understanding digital transformation: A review and a research agenda', *Managing digital transformation*, pp. 13–66.

Wairagade, A. and Das, R. (2025) 'AI-Driven Biometric Systems Using Facial Recognition and Fingerprint Analysis for Next Generation Identity Management', in *Advanced Interdisciplinary Applications of Large Language Models and AI Agents*, pp. 109–132. Available at: https://doi.org/10.4018/979-8-3373-2008-3.ch005.

Yaakub, S. *et al.* (2025) *AUTOMASI MASA DEPAN: AI, MACHINE LEARNING, DAN PERUBAHAN DUNIA KERJA*. Edited by A. Yanto. Get Press Indonesia.

Yadav, S.P. *et al.* (2023) 'Voice-Based Virtual-Controlled Intelligent Personal Assistants', in *2023 International Conference on Computational Intelligence, Communication Technology and Networking, CICTN 2023*, pp. 563–568. Available at: https://doi.org/10.1109/CICTN57981.2023.10141447.

Zhao, Q. and Liu, T. (2024) 'Application and Prospect of Deep Learning and Machine Learning Technology', in *Smart Innovation, Systems and Technologies*, pp. 259–269. Available at: https://doi.org/10.1007/978-981-99-6641-7_22.

BAB 2

SEJARAH DAN PERKEMBANGAN AI

2.1 Sejarah Kecerdasan Buatan

Kecerdasan Buatan (*Artificial Intelligence*/AI) merupakan salah satu bidang dalam ilmu komputer yang bertujuan untuk mengembangkan sistem atau mesin yang mampu meniru perilaku cerdas manusia. Konsep kecerdasan dalam AI mencakup kemampuan untuk memahami lingkungan, melakukan penalaran, belajar dari pengalaman, serta mengambil keputusan secara mandiri. Dalam beberapa dekade terakhir, AI berkembang sangat pesat dan menjadi teknologi kunci dalam transformasi digital global. (Birch,2024)

Meskipun demikian, perkembangan AI tidak terjadi secara instan. Teknologi yang saat ini digunakan dalam pengenalan wajah, kendaraan otonom, sistem rekomendasi, dan pemrosesan bahasa alami merupakan hasil akumulasi gagasan, eksperimen, dan penelitian selama lebih dari setengah abad. Sejarah AI memperlihatkan bahwa kemajuan teknologi sangat dipengaruhi oleh ketersediaan perangkat keras, metode matematis, serta perubahan paradigma ilmiah. (Birch,2024)

2.1.1 Pentingnya Memahami Sejarah AI

Pemahaman terhadap sejarah AI memiliki nilai edukatif dan praktis yang signifikan. Dari sudut pandang akademik, sejarah AI membantu mahasiswa memahami

evolusi konsep kecerdasan mesin, mulai dari pendekatan simbolik hingga pendekatan berbasis data. Dari sudut pandang praktis, sejarah AI memberikan pelajaran berharga tentang keberhasilan dan kegagalan teknologi. Banyak pendekatan AI awal dikembangkan dengan asumsi bahwa kecerdasan manusia dapat direpresentasikan sepenuhnya melalui aturan logika eksplisit. Pendekatan ini menghasilkan sistem yang mampu menyelesaikan masalah terbatas, namun gagal ketika menghadapi ketidakpastian dan kompleksitas dunia nyata. Kegagalan tersebut mengajarkan bahwa kecerdasan tidak hanya bergantung pada aturan, tetapi juga pada kemampuan belajar dari data. Selain itu, sejarah AI memperlihatkan bahwa kemajuan algoritma sering kali tertunda oleh keterbatasan teknologi. Algoritma jaringan saraf tiruan, misalnya, telah dikenal sejak 1950-an, namun baru mencapai keberhasilan besar pada abad ke-21 ketika daya komputasi dan data tersedia dalam jumlah besar. Oleh karena itu, memahami sejarah AI membantu praktisi mengembangkan perspektif kritis terhadap tren teknologi yang ada. (Birch,2024)

2.1.2 Periode Utama Perkembangan AI

Perkembangan kecerdasan buatan dapat dibagi ke dalam beberapa periode utama. Periode pertama adalah fase konseptual dan filosofis, di mana pemikiran tentang mesin berpikir masih bersifat teoritis. Periode ini meletakkan dasar konseptual bagi AI, meskipun belum didukung oleh teknologi komputasi yang memadai. (Birch,2024)

Periode kedua dimulai pada pertengahan abad ke-20 dengan munculnya komputer digital dan kelahiran AI sebagai disiplin ilmu formal. Optimisme tinggi pada masa ini mendorong lahirnya berbagai program AI

berbasis simbol dan logika. Periode ketiga ditandai dengan berkembangnya sistem pakar yang berhasil diterapkan dalam domain terbatas, seperti kedokteran dan kimia. (Birch,2024)

Namun, keterbatasan sistem tersebut memicu periode stagnasi yang dikenal sebagai AI Winter. Kebangkitan kembali AI terjadi ketika pendekatan statistik dan machine learning mulai menggantikan pendekatan simbolik. Periode ini berlanjut hingga era deep learning dan AI generatif yang mendominasi penelitian dan industri saat ini. (Birch,2024)

2.1.3 Hubungan Sejarah dengan Machine Learning Python

Sejarah AI memiliki keterkaitan erat dengan dominasi Python dalam pengembangan AI modern. Pergeseran paradigma dari symbolic AI ke data-driven AI menuntut bahasa pemrograman yang fleksibel, mudah digunakan, dan didukung oleh ekosistem pustaka yang kuat. Python memenuhi kebutuhan tersebut dan menjadi standar de facto dalam pengembangan machine learning. Framework Python seperti Scikit-learn, TensorFlow, dan PyTorch merepresentasikan evolusi historis AI. Algoritma klasik hingga model deep learning modern dapat diimplementasikan secara efisien menggunakan Python. Dengan memahami sejarah AI, pembaca dapat memahami bahwa Python bukan sekadar alat teknis, melainkan bagian integral dari evolusi AI itu sendiri. (Bishop,2026)

2.2 Akar Filosofis dan Fondasi Komputasi

2.2.1 Pemikiran Filsafat tentang Mesin Berpikir

Diskusi mengenai mesin berpikir berakar dari filsafat epistemologi dan ontologi. Para filsuf

mempertanyakan apakah proses berpikir manusia dapat direduksi menjadi mekanisme fisik atau simbolik. René Descartes berpendapat bahwa pikiran manusia bersifat non-mekanis, sementara Leibniz membayangkan mesin logis yang dapat melakukan penalaran simbolik. Perdebatan filosofis ini memengaruhi arah penelitian AI, khususnya dalam mendefinisikan kecerdasan dan kesadaran. Alan Turing mengalihkan perdebatan filosofis tersebut ke pendekatan empiris, dengan mengusulkan pengujian berbasis perilaku sebagai ukuran kecerdasan mesin. (Bishop,2026)

2.2.2 Logika Formal dan Aljabar Boolean

Logika formal menjadi fondasi penting dalam pengembangan AI awal. George Boole memperkenalkan aljabar Boolean sebagai sistem logika matematika, yang kemudian diimplementasikan secara fisik dalam sirkuit digital. Konsep ini memungkinkan komputer melakukan operasi logika dasar yang menjadi dasar penalaran simbolik. Dalam AI awal, pengetahuan direpresentasikan dalam bentuk simbol dan aturan logika. Pendekatan ini memberikan struktur formal bagi pemodelan pengetahuan, meskipun memiliki keterbatasan dalam menangani ketidakpastian. (Bishop,2026)

2.2.3 Mesin Turing dan Konsep Komputasi Universal

Mesin Turing memperkenalkan konsep komputasi universal, yang menyatakan bahwa satu mesin dapat menjalankan berbagai algoritma jika diberikan instruksi yang tepat. Konsep ini memberikan landasan teoretis bahwa kecerdasan dapat diwujudkan melalui proses komputasi. Makalah Turing tahun 1950 menjadi tonggak penting dalam AI, karena menghubungkan filsafat, matematika, dan komputasi dalam satu kerangka ilmiah. (Bishop,2026)

2.3 Kelahiran AI: Konferensi Dartmouth hingga 1960-an

Kelahiran kecerdasan buatan sebagai disiplin ilmu yang mandiri tidak dapat dilepaskan dari perkembangan komputer digital pada pertengahan abad ke-20. Pada masa ini, komputer mulai dipandang bukan sekadar mesin hitung, tetapi sebagai perangkat yang berpotensi meniru proses kognitif manusia. Perubahan cara pandang ini mendorong para ilmuwan dari berbagai disiplin matematika, psikologi, logika, dan teknik untuk mengeksplorasi kemungkinan penciptaan mesin cerdas. Tonggak utama dalam sejarah AI terjadi pada tahun 1956 melalui Dartmouth Summer Research Project on Artificial Intelligence. Konferensi ini menjadi forum resmi pertama yang menyatukan para peneliti dengan visi bersama untuk mengembangkan kecerdasan mesin. Optimisme yang tinggi pada periode ini membuat banyak peneliti percaya bahwa kecerdasan manusia dapat direplikasi sepenuhnya dalam waktu relatif singkat. (Bishop,2026)

2.3.1 Tes Turing dan Visi Awal Kecerdasan Mesin

Alan Turing memainkan peran sentral dalam merumuskan visi awal kecerdasan mesin. Dalam makalahnya *Computing Machinery and Intelligence* (1950), Turing mengajukan pertanyaan fundamental *"Can machines think?"* dan menawarkan pendekatan praktis untuk menjawabnya melalui apa yang kini dikenal sebagai Tes Turing. Tes ini mengukur kecerdasan mesin berdasarkan kemampuannya meniru perilaku manusia dalam percakapan teks. (Bishop,2026)

Pendekatan Turing bersifat revolusioner karena mengalihkan fokus dari definisi abstrak kecerdasan menuju pengukuran berbasis perilaku. Gagasan ini memberikan kerangka evaluasi yang dapat diuji secara

empiris, meskipun hingga saat ini masih menjadi bahan perdebatan filosofis dan ilmiah. Tes Turing juga memengaruhi arah penelitian AI awal, khususnya dalam pengembangan sistem berbasis bahasa dan dialog. (Bishop,2026)

2.3.2 Program AI Awal: Logic Theorist dan ELIZA

Setelah Konferensi Dartmouth, berbagai program AI awal mulai dikembangkan sebagai bukti bahwa mesin dapat melakukan tugas kognitif. Salah satu program paling berpengaruh adalah Logic Theorist yang dikembangkan oleh Allen Newell dan Herbert A. Simon. Program ini dirancang untuk membuktikan teorema matematika menggunakan pendekatan simbolik dan heuristik. Keberhasilan Logic Theorist memperkuat keyakinan bahwa penalaran manusia dapat dimodelkan secara komputasional. Namun, pendekatan ini juga menunjukkan keterbatasan karena bergantung pada aturan yang telah didefinisikan sebelumnya. Pada periode yang sama, Joseph Weizenbaum mengembangkan ELIZA, sebuah program yang mensimulasikan percakapan manusia menggunakan teknik pencocokan pola sederhana. Meskipun ELIZA tidak memiliki pemahaman semantik yang mendalam, respons pengguna menunjukkan bahwa manusia cenderung mengatribusikan kecerdasan pada sistem yang mampu berinteraksi secara linguistik. Fenomena ini menyoroti dimensi psikologis dalam interaksi manusia–mesin yang masih relevan dalam pengembangan AI modern. (Bishop,2026)

2.4 Masa Keemasan Awal dan Sistem Pakar

Pada akhir 1960-an hingga 1980-an, AI memasuki fase yang sering disebut sebagai masa keemasan awal. Pada periode ini, fokus penelitian beralih ke pengembangan sistem pakar (*expert systems*), yaitu sistem komputer yang dirancang untuk meniru kemampuan pengambilan keputusan seorang ahli dalam domain tertentu. Sistem pakar menjadi aplikasi AI pertama yang digunakan secara nyata di lingkungan profesional. (Christian,2020)

2.4.1 Pengembangan Bahasa LISP

Pengembangan sistem AI awal sangat bergantung pada bahasa pemrograman yang mendukung manipulasi simbol dan struktur data kompleks. John McCarthy mengembangkan bahasa LISP sebagai solusi untuk kebutuhan tersebut. LISP memungkinkan representasi pengetahuan dalam bentuk simbolik dan mendukung pemrograman rekursif, yang sangat sesuai untuk penalaran AI. Bahasa LISP mendominasi riset AI selama beberapa dekade dan memengaruhi banyak bahasa pemrograman modern. Konsep seperti garbage collection, dynamic typing, dan functional programming yang populer saat ini dapat ditelusuri kembali ke LISP. (Christian,2020)

2.4.2 Sistem Pakar: MYCIN dan DENDRAL

Dua contoh sistem pakar yang sering dikutip adalah DENDRAL dan MYCIN. DENDRAL dikembangkan untuk membantu ahli kimia dalam mengidentifikasi struktur molekul berdasarkan data spektrometri massa. Sistem ini menggunakan basis pengetahuan dan aturan heuristik yang dirancang oleh pakar manusia. MYCIN dikembangkan untuk membantu diagnosis infeksi bakteri dan rekomendasi antibiotik. Sistem ini mampu memberikan keputusan yang sebanding dengan dokter

ahli dalam kondisi tertentu. Namun, sistem pakar juga memiliki keterbatasan signifikan, seperti kesulitan memperbarui basis pengetahuan dan ketergantungan pada aturan eksplisit. Keterbatasan inilah yang kemudian memicu penurunan minat terhadap pendekatan symbolic AI. (Christian,2020)

2.5 Kebangkitan Machine Learning dan Data-Driven AI

Keterbatasan sistem pakar dan pendekatan simbolik mendorong munculnya paradigma baru dalam AI, yaitu machine learning. Pendekatan ini berfokus pada kemampuan sistem untuk belajar dari data, bukan sekadar mengikuti aturan yang telah ditentukan.

2.5.1 Pergeseran dari Symbolic ke Statistical AI

Pendekatan statistik memungkinkan AI menangani ketidakpastian dan variasi data dunia nyata. Alih-alih merepresentasikan pengetahuan secara eksplisit, machine learning mengekstraksi pola secara implisit melalui proses pelatihan. Pergeseran ini menandai perubahan mendasar dalam cara kecerdasan mesin dipahami dan diimplementasikan. Pendekatan statistical AI juga membuka jalan bagi integrasi metode probabilistik, seperti Bayesian networks dan model Markov, yang meningkatkan kemampuan prediksi dan pengambilan keputusan AI. (Goodfellow,2016)

2.5.2 Algoritma ML Klasik: *Perceptron* dan *Backpropagation*

Perceptron merupakan model jaringan saraf sederhana yang memperkenalkan konsep pembelajaran melalui penyesuaian bobot. Meskipun awalnya menuai

kritik karena keterbatasannya, konsep perceptron menjadi fondasi bagi pengembangan jaringan saraf multilapis. Algoritma *backpropagation* kemudian memperluas kemampuan jaringan saraf dengan memungkinkan pembaruan bobot secara efisien melalui propagasi kesalahan. Metode ini menjadi tulang punggung pelatihan jaringan saraf modern dan berperan penting dalam kebangkitan AI pada akhir abad ke-20. (Iqbal Hajamydeen,2025)

2.5.3 Peran Data Besar dan Komputasi Paralel

Kemajuan teknologi penyimpanan data dan komputasi paralel merupakan salah satu faktor kunci yang mendorong akselerasi perkembangan machine learning dan kecerdasan buatan modern. Pada era awal AI, keterbatasan kapasitas penyimpanan dan kecepatan komputasi menjadi hambatan utama dalam pengolahan data berskala besar. Model pembelajaran mesin pada masa tersebut hanya dapat dilatih menggunakan dataset berukuran kecil, sehingga kemampuan generalisasi dan performa model sangat terbatas. Munculnya teknologi big data mengubah kondisi tersebut secara fundamental. Perkembangan sistem basis data terdistribusi, penyimpanan berbasis cloud, serta infrastruktur data modern memungkinkan pengumpulan, penyimpanan, dan pengelolaan data dalam jumlah yang sangat besar dan beragam. Data yang berasal dari berbagai sumber, seperti media sosial, sensor IoT, transaksi digital, dan aktivitas daring, menyediakan informasi yang kaya dan representatif bagi proses pembelajaran mesin. Dalam konteks ini, data berperan sebagai “bahan bakar” utama yang memungkinkan model machine learning mempelajari pola kompleks yang sebelumnya sulit atau bahkan tidak mungkin diidentifikasi. Selain ketersediaan data, kemajuan dalam komputasi paralel, khususnya

melalui penggunaan *Graphics Processing Unit* (GPU), memberikan dampak signifikan terhadap efisiensi pelatihan model. GPU dirancang untuk menangani operasi matriks dan vektor secara paralel, yang sangat sesuai dengan kebutuhan komputasi dalam pelatihan jaringan saraf tiruan. Dibandingkan dengan *Central Processing Unit* (CPU) tradisional, GPU mampu mempercepat proses pelatihan model secara drastis, bahkan hingga beberapa orde magnitudo. Komputasi paralel juga memungkinkan eksperimen yang lebih luas dan kompleks. Peneliti dan praktisi dapat melatih model dengan arsitektur yang lebih dalam, jumlah parameter yang lebih besar, serta melakukan proses hyperparameter tuning secara lebih intensif. Hal ini berkontribusi langsung terhadap peningkatan performa model machine learning dan deep learning dalam berbagai aplikasi nyata. (Mbiazi,2023)

2.6 Revolusi *Deep Learning* dan Model Modern

2.6.1 Munculnya *Deep Neural Networks*

Deep Neural Networks (DNN) merupakan pengembangan dari jaringan saraf tiruan tradisional dengan menambahkan banyak lapisan tersembunyi (hidden layers) di antara lapisan input dan output. Setiap lapisan bertugas mempelajari representasi data pada tingkat abstraksi yang berbeda. Lapisan awal biasanya mengekstraksi fitur sederhana, sementara lapisan yang lebih dalam mempelajari pola yang lebih kompleks dan abstrak. Pendekatan hierarkis ini terbukti sangat efektif dalam berbagai bidang, khususnya pengenalan citra, pengenalan suara, dan pemrosesan bahasa alami. Dalam pengenalan citra, misalnya, lapisan awal jaringan saraf dapat mendeteksi tepi dan bentuk sederhana, sedangkan

lapisan selanjutnya mampu mengenali objek secara utuh. Kemampuan ini sulit dicapai dengan pendekatan machine learning klasik yang mengandalkan fitur buatan manusia. Keberhasilan deep neural networks tidak hanya disebabkan oleh struktur jaringan yang lebih dalam, tetapi juga oleh beberapa faktor pendukung penting. Pertama, ketersediaan data dalam jumlah besar memungkinkan model deep learning untuk belajar secara lebih representatif dan general. Kedua, perkembangan perangkat keras seperti *Graphics Processing Unit* (GPU) memungkinkan komputasi paralel yang sangat efisien untuk pelatihan model berskala besar. Ketiga, pengembangan algoritma optimasi dan teknik regularisasi, seperti dropout dan *batch* normalization, membantu mengatasi masalah *overfitting* dan stabilitas pelatihan. (Mitchell,1997)

2.6.2 AI Generatif: GAN dan Aplikasi Kontemporer

Salah satu cabang penting dari deep learning modern adalah AI generatif, yaitu pendekatan yang memungkinkan sistem AI tidak hanya menganalisis atau mengklasifikasikan data, tetapi juga menghasilkan data baru yang menyerupai data asli. Pendekatan ini membuka paradigma baru dalam AI, di mana mesin berperan sebagai pencipta (generator) konten. *Generative Adversarial Networks* (GAN) merupakan salah satu model generatif yang paling berpengaruh. GAN terdiri dari dua jaringan saraf yang dilatih secara bersamaan, yaitu generator dan discriminator. Generator bertugas menghasilkan data sintetis, sementara discriminator berfungsi membedakan antara data asli dan data hasil generasi.

Proses pelatihan bersifat kompetitif, di mana kedua jaringan saling meningkatkan kemampuan hingga generator mampu menghasilkan data yang sulit

dibedakan dari data asli. Selain GAN, model generatif berbasis transformer telah merevolusi pemrosesan bahasa alami dan domain lain. Model ini menggunakan mekanisme self-attention untuk mempelajari hubungan kontekstual dalam data secara efisien. Transformer memungkinkan pengembangan model bahasa skala besar yang mampu menghasilkan teks, kode program, dan bahkan konten multimedia dengan tingkat koherensi yang tinggi. Aplikasi AI generatif saat ini mencakup berbagai bidang, seperti pembuatan gambar dan video, sintesis suara, penulisan teks otomatis, desain produk, hingga pengembangan perangkat lunak. Perkembangan ini menunjukkan bahwa AI tidak lagi terbatas pada tugas analitis, tetapi juga berperan dalam aktivitas kreatif, yang sebelumnya dianggap eksklusif bagi manusia. (Russell,2021)

2.7 Implikasi Sejarah bagi Praktik AI dengan *Python*

Sejarah perkembangan AI memberikan landasan penting bagi praktik pengembangan AI modern, khususnya dalam konteks penggunaan bahasa pemrograman Python. Python telah menjadi bahasa utama dalam pengembangan AI karena kemampuannya menjembatani teori dan implementasi secara efektif. Evolusi AI dari pendekatan simbolik hingga deep learning tercermin secara jelas dalam ekosistem pustaka Python yang terus berkembang. (Smith,2024)

2.7.1 Evolusi Paradigma AI dan *Framework Python*

Framework dan pustaka Python mencerminkan perjalanan historis AI. Untuk algoritma machine learning klasik, pustaka seperti Scikit-learn menyediakan implementasi yang stabil dan mudah digunakan. Pustaka

ini memungkinkan pengguna menerapkan metode statistik dan machine learning tradisional secara efisien, sekaligus memahami keterbatasan dan keunggulan pendekatan tersebut. Dalam konteks deep learning, framework seperti TensorFlow dan PyTorch mendukung pengembangan model jaringan saraf yang kompleks dan berskala besar. Framework ini dirancang dengan mempertimbangkan fleksibilitas dan performa, sehingga cocok untuk riset akademik maupun aplikasi industri. Selain itu, Python juga mendukung integrasi dengan pustaka analisis data seperti NumPy dan Pandas, yang memperkuat peran Python sebagai ekosistem terpadu untuk pengembangan AI. Evolusi framework Python menunjukkan bagaimana sejarah AI memengaruhi alat dan praktik pengembangan saat ini. Dengan memahami sejarah tersebut, praktisi dapat memilih metode dan framework yang sesuai dengan karakteristik masalah yang dihadapi, bukan sekadar mengikuti tren teknologi. (Smith,2024)

2.7.2 Prospek Masa Depan dan Etika Praktisi

Sejarah kecerdasan buatan menunjukkan bahwa setiap lompatan teknologi selalu diiringi oleh tantangan baru, khususnya dalam aspek etika dan tanggung jawab sosial. Sejak awal perkembangan AI, para peneliti telah menyadari bahwa kemampuan mesin untuk meniru atau bahkan melampaui kecerdasan manusia menimbulkan konsekuensi yang tidak hanya bersifat teknis, tetapi juga sosial dan moral. Pada era AI modern, tantangan etika ini menjadi semakin kompleks seiring dengan meningkatnya otonomi dan skala penerapan sistem AI dalam kehidupan sehari-hari. Salah satu isu etika utama dalam AI modern adalah bias algoritmik. Model machine learning belajar dari data historis, dan apabila data tersebut mengandung bias sosial, ekonomi, atau budaya,

maka model yang dihasilkan berpotensi mereproduksi bahkan memperkuat bias tersebut. Hal ini dapat berdampak serius, misalnya dalam sistem rekrutmen, penilaian kredit, atau penegakan hukum berbasis AI. Sejarah AI menunjukkan bahwa kepercayaan berlebihan terhadap sistem otomatis tanpa pemahaman kritis dapat menghasilkan keputusan yang tidak adil dan diskriminatif. Selain bias, kurangnya transparansi model atau fenomena black-box models juga menjadi perhatian utama. Model deep learning dengan jutaan parameter sering kali sulit dijelaskan secara intuitif, sehingga menyulitkan pengguna dan pemangku kepentingan untuk memahami dasar pengambilan keputusan sistem. Ketidakjelasan ini menimbulkan masalah akuntabilitas, terutama ketika sistem AI digunakan dalam domain sensitif seperti kesehatan, pendidikan, dan keamanan publik. Sejarah AI mengajarkan bahwa sistem yang tidak dapat dijelaskan cenderung menimbulkan resistensi dan ketidakpercayaan dari masyarakat. (Smith,2024)

Isu etika lainnya berkaitan dengan keamanan sistem dan perlindungan privasi data. Model AI modern membutuhkan data dalam jumlah besar, sering kali mencakup data pribadi yang sensitif. Tanpa mekanisme perlindungan yang memadai, penggunaan data ini berpotensi melanggar privasi individu dan membuka celah penyalahgunaan informasi. Selain itu, sistem AI juga rentan terhadap serangan, seperti adversarial attacks, yang dapat dimanfaatkan untuk memanipulasi keputusan model. Hal ini menunjukkan bahwa keamanan harus menjadi bagian integral dari desain dan implementasi AI. Dalam praktik AI berbasis Python, tantangan etika ini menjadi semakin relevan karena kemudahan akses terhadap pustaka dan framework AI memungkinkan siapa pun untuk mengembangkan dan menerapkan model dengan cepat. Python, dengan

ekosistemnya yang luas, menurunkan hambatan teknis pengembangan AI, namun pada saat yang sama meningkatkan risiko penerapan sistem tanpa pertimbangan etis yang matang. Praktisi AI tidak hanya dituntut memiliki kompetensi teknis, tetapi juga pemahaman etis dalam setiap tahap siklus pengembangan, mulai dari pengumpulan data, pemilihan model, hingga evaluasi dan penerapan sistem. Kesalahan dalam desain model atau penggunaan data yang tidak tepat dapat berdampak langsung pada individu dan masyarakat luas. Sejarah AI mencatat berbagai contoh di mana kegagalan mempertimbangkan aspek etika menyebabkan penolakan sosial dan kegagalan implementasi teknologi. Oleh karena itu, refleksi terhadap sejarah AI menjadi penting untuk menghindari pengulangan kesalahan yang sama dalam konteks teknologi yang lebih canggih. (Tegmark,2017)

Dengan demikian, pemahaman sejarah kecerdasan buatan berperan sebagai pengingat bahwa perkembangan teknologi harus selalu disertai dengan refleksi kritis dan tanggung jawab sosial. Dengan belajar dari keberhasilan dan kegagalan masa lalu, praktisi AI diharapkan mampu mengembangkan sistem yang tidak hanya unggul secara teknis, tetapi juga adil, transparan, aman, dan bertanggung jawab. Pendekatan ini menjadi kunci dalam memastikan bahwa AI memberikan manfaat jangka panjang bagi manusia dan masyarakat secara berkelanjutan.

DAFTAR PUSTAKA

Birch, J., 2024. The Edge of Sentience: Risk and Precaution in Humans, Other Animals, and AI. Oxford University Press.

Bishop, C. M., 2006. Pattern Recognition and Machine Learning. Springer.

Christian, B., 2020. The Alignment Problem: Machine Learning and Human Values. W. W. Norton & Company.

Goodfellow, I., Bengio, Y. & Courville, A., 2016. Deep Learning. MIT Press.

Iqbal Hajamydeen, A. & Kumar, S., 2025. A Survey of Deep Learning Techniques: Applications Across Industries and Ethical Considerations. Preprints.org.

Mbiazi, D., Bhange, M., Babaei, M., Sheth, I., & Kenfack, P. J., 2023. Survey on AI Ethics: A Socio-technical Perspective. arXiv (preprint).

Mitchell, T. M., 1997. Machine Learning. McGraw-Hill.

Russell, S. & Norvig, P., 2021. Artificial Intelligence: A Modern Approach. 4th ed. Pearson Education.

Smith, J. J., Deng, W. H., Smith, W. H., Sap, M., DeCario, N., & Dodge, J., 2024. The Generative AI Ethics Playbook. arXiv (preprint).

Tegmark, M., 2017. Life 3.0: Being Human in the Age of Artificial Intelligence. Knopf.

BAB 3

KONSEP DASAR *MACHINE LEARNING*

3.1 Definisi Formal dan *Paradigma Learning*

Pembelajaran Mesin (ML) adalah cabang kecerdasan buatan yang secara sistematis menerapkan algoritma untuk mensintesis hubungan mendasar antara data dan informasi. Misalnya, sistem ML dapat dilatih pada sistem pengenalan suara otomatis (seperti Siri pada iPhone) untuk mengubah informasi akustik dalam urutan data ucapan menjadi struktur semantik yang dinyatakan dalam bentuk rangkaian kata. (Awad & Khanna, 2015).

- **Definisi Tom Mitchell:** Menjelaskan konsep ML sebagai pembelajaran dari pengalaman ***E***, sehubungan dengan tugas ***T***, dan ukuran kinerja ***P*** (Mitchell, 1997).
- **Perbandingan:** Model Tradisional vs. Model Machine Learning (aturan eksplisit vs. aturan yang dipelajari).
- **Terminologi Kunci:** Pengenalan istilah esensial: **Dataset**, **Fitur** (*Features*), **Label/Target**, **Model**, **Pelatihan** (*Training*), **Generalisasi**, dan **Inferensi** (*Inference*).

3.1.1 Menganalisis Definisi Formal Tom Mitchell

Seperti yang telah disebutkan, definisi Machine Learning yang paling diterima adalah dari Tom Mitchell:

Definisi formalnya adalah sebagai berikut:

"Sebuah program komputer dikatakan belajar dari pengalaman (E) sehubungan dengan beberapa tugas (T) dan ukuran kinerja (P), jika kinerjanya pada T, yang diukur oleh P, meningkat dengan pengalaman E."(Mitchell, 1997)

Definisi ini menetapkan tiga pilar operasional yang harus ada dalam setiap proyek Machine Learning.

1. Tugas (T): Task (Apa yang Ingin Dicapai)

   Tugas (T) mendefinisikan apa yang harus dipelajari atau dilakukan oleh sistem ML. Tugas-tugas ini biasanya diklasifikasikan berdasarkan jenis pembelajaran yang didiskusikan (Supervised, Unsupervised, dll.):

   - Klasifikasi: Tugas untuk menetapkan label kategori diskrit ke suatu input.

     Contoh: Mengidentifikasi apakah sebuah gambar adalah kucing

     atau anjing.

   - Regresi: Tugas untuk memprediksi nilai kontinu (numerik).

     Contoh: Memprediksi harga rumah berdasarkan luas dan lokasi.

   - Clustering: Tugas untuk mengidentifikasi kelompok-kelompok yang kohesif dalam data yang tidak berlabel.

2. Pengalaman (E): Experience (Bagaimana Sistem Belajar)

   Pengalaman (E) adalah esensi dari proses pembelajaran, yaitu data yang digunakan model untuk pelatihan. Ini adalah sumber informasi yang

digunakan oleh algoritma untuk menyimpulkan pola dan membangun "aturan" internal model.

- Dalam konteks Supervised Learning, E biasanya adalah dataset berpasangan yang terdiri dari input (fitur) dan output (label) yang benar.
  Contoh: Ribuan pasangan gambar kucing/anjing (input) beserta label yang benar (output).
- Proses Training (Pelatihan) adalah ketika model berinteraksi dengan E. Melalui iterasi berulang pada data ini, model menyesuaikan bobot dan bias internalnya untuk meminimalkan kesalahan antara prediksinya dan label yang sebenarnya.

3. Ukuran Kinerja (P): Performance Measure (Seberapa Baik Model Berhasil)

   Ukuran Kinerja (P) adalah metrik kuantitatif yang digunakan untuk mengevaluasi seberapa efektif model dalam menyelesaikan tugas T setelah mendapatkan pengalaman E.

   - P harus relevan dengan tugas T.
   - Jika T adalah Klasifikasi: P dapat diukur dengan Akurasi (persentase prediksi benar) atau F1-Score.
   - Jika T adalah Regresi: P dapat diukur dengan *Mean Squared Error* (MSE), yang mengukur seberapa dekat prediksi model dengan nilai aktual.

   Inti Pembelajaran: Menurut Mitchell, pembelajaran terjadi jika mengamati bahwa nilai P (misalnya, Akurasi) meningkat seiring dengan semakin banyaknya model berinteraksi dengan E

(data pelatihan). Jika model tidak menunjukkan peningkatan kinerja pada tugas tersebut, maka ia belum "belajar."

3.1.2 Perbandingan: Model Tradisional vs. Model *Machine Learning*

Pemrograman tradisional dan Machine Learning mewakili dua paradigma fundamental yang berbeda dalam mencapai output dari input data. Memahami perbedaannya sangat penting untuk mengapresiasi keunikan ML.

1. Paradigma Pemrograman Tradisional (Aturan Eksplisit)

   Dalam model tradisional, interaksi antara data dan aturan sangat eksplisit:

   User (Programmer) Menyediakan:

   Data (Input)

   Aturan (Logika hard-coded, seperti fungsi, kondisi IF-ELSE, dan loops).

   Komputer Menghasilkan:

   Jawaban (Output)

   Di sini, setiap langkah pemrosesan ditentukan oleh manusia. Komputer hanyalah alat yang secara setia menjalankan algoritma dan logika yang sudah ditetapkan oleh programmer.

   Contoh Tradisional:Membuat program untuk menentukan apakah suatu bilangan adalah genap atau ganjil. User harus secara eksplisit menulis aturan: "JIKA bilangan % 2 = 0 MAKA output Ganjil, SELAINNYA output Genap."

2. Paradigma Machine Learning (Aturan yang Dipelajari)

Machine Learning membalikkan proses ini. Model tidak diberi tahu aturan secara spesifik; sebaliknya, model yang bertugas menemukan aturan tersebut.

User (Programmer) Menyediakan:

Data (Input)

Jawaban (Output/Label yang benar untuk setiap data input).

Komputer Menghasilkan:

Aturan (Model atau Algoritma yang dipelajari sendiri).

Dalam ML, tujuannya adalah menciptakan fungsi matematis (Model) yang dapat memetakan X ke Y. Model ini belajar pola dari data input dan output yang disajikan selama fase pelatihan, tanpa perlu hard-code aturan untuk setiap skenario. Inilah yang memungkinkan ML mengatasi masalah yang terlalu kompleks atau ambigu untuk ditangani oleh logika eksplisit.

Fitur	**Pemrograman Tradisional**	***Machine Learning***
Fokus Utama	Logika yang sudah ditentukan (Aturan)	Pola data (Pengalaman E)
Output	Hasil berdasarkan aturan yang kaku	Model (Aturan yang Dipelajari)

Fitur	Pemrograman Tradisional	*Machine Learning*
Kesalahan	Mudah di-*debug* dan diperbaiki dengan meninjau logika	Sulit di-*debug*, membutuhkan data yang lebih baik atau penyesuaian model
Kompleksitas	Terbaik untuk tugas yang terstruktur dan terdefinisi dengan jelas	Terbaik untuk tugas yang ambigu dan kompleks (contoh: pengenalan gambar)

Perbedaan ini adalah alasan mengapa Machine Learning menjadi alat yang sangat kuat untuk memecahkan masalah modern, seperti pengenalan suara, penerjemahan bahasa, atau diagnosis medis, di mana aturan yang mendasarinya terlalu rumit, dinamis, atau tidak diketahui oleh manusia.

3.1.3 Terminologi Kunci: Bahasa Dasar *Machine Learning*

Memahami istilah-istilah berikut adalah krusial karena terminologi ini akan digunakan secara konsisten dalam proses membangun, melatih, dan mengevaluasi model Python di bab-bab berikutnya.

1. Dataset

   Definisi: Kumpulan data terorganisir yang digunakan untuk melatih dan menguji model ML. Dataset biasanya berbentuk tabel, di mana setiap baris mewakili satu Observasi atau Sampel, dan setiap kolom mewakili karakteristik data.

Contoh: Kumpulan data historis harga rumah, di mana setiap baris adalah satu rumah, dan kolomnya berisi informasi seperti luas, jumlah kamar, dan lokasi.

2. Fitur (*Features*)

   Definisi: Variabel input atau karakteristik dari data yang digunakan model untuk membuat prediksi. Fitur adalah kolom-kolom input dalam dataset.

   Contoh: Dalam dataset harga rumah, Fitur adalah Luas Rumah, Jumlah Kamar, dan Usia Bangunan.

3. Label/Target

   Definisi: Variabel output yang coba diprediksi atau dipelajari oleh model. Label adalah jawaban yang benar (atau output yang diinginkan) untuk setiap observasi. Istilah ini sering digunakan dalam konteks Supervised Learning.

   Contoh: Dalam dataset harga rumah, Label adalah Harga Jual rumah tersebut.

4. Model

   Definisi: Struktur atau program matematis yang telah dilatih dan mampu memetakan Fitur (input) ke Label (output). Model pada dasarnya adalah fungsi yang telah menemukan pola dalam data.

   Peran: Setelah pelatihan, model bertindak sebagai "kotak aturan" yang dipelajari, siap menerima input baru untuk menghasilkan prediksi.

5. Pelatihan (*Training*)

   Definisi: Proses di mana algoritma menggunakan data (Pengalaman E) untuk menemukan pola dan menyesuaikan parameter internal Model (bobot dan bias) agar kesalahan antara prediksi model dan Label yang sebenarnya diminimalkan.

   Siklus: Proses pelatihan bersifat iteratif, di mana model berulang kali melihat data, membuat prediksi, menghitung kesalahan, dan menyesuaikan diri.

6. Generalisasi

   Definisi: Kemampuan model yang telah dilatih untuk membuat prediksi akurat pada data baru yang belum pernah dilihat sebelumnya (data uji).

   Tujuan Utama ML: Model yang baik harus mampu melakukan generalisasi dengan baik, bukan hanya sekadar menghafal data pelatihan (menghindari *overfitting*).

7. Inferensi (*Inference*)

   Definisi: Proses di mana model yang sudah terlatih (sudah memiliki "aturan" yang dipelajari) digunakan untuk memproses input baru dan menghasilkan prediksi output yang sesuai.

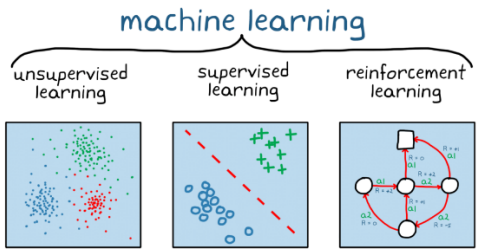
   Penerapan: Ini adalah tahap implementasi model di dunia nyata.

   Contoh: memberikan gambar baru kepada model klasifikasi, dan model tersebut mengeluarkan hasil prediksi: "Anjing."

Penguasaan terminologi ini akan mempermudah dalam memahami algoritma, mengevaluasi kinerja, dan mengimplementasikan solusi Machine Learning menggunakan Python.

3.2 Tiga Pilar Utama: Jenis-Jenis Pembelajaran dalam ML

Semua algoritma dan model ML, terlepas dari kompleksitasnya, dapat diklasifikasikan ke dalam tiga kategori besar atau Pilar Utama berdasarkan bagaimana mereka berinteraksi dengan data untuk belajar.Tiga pilar ini mendefinisikan strategi yang digunakan model untuk mengekstrak pengetahuan dari pengalaman (E) (Shalev-Shwartz, 2014).



Gambar 3. 1 Tiga Jenis *Machine Learning*
(Sumber : https://www.devopsschool.com/blog/machine-learning-compare-supervised-learning-vs-unsupervised-learning-vs-reinforcement-learning/)

3.2.1 *Supervised Learning*

Pembelajaran mesin dilatih menggunakan data pelatihan yang diberi label dengan baik. Data yang diberikan untuk pelatihan harus sangat akurat tanpa data palsu karena mesin sepenuhnya bergantung pada data yang diberikan untuk pelatihannya. Mesin dirancang dengan memanfaatkan informasi yang

diketahui sebelumnya dalam bentuk contoh pelatihan "langsung" yang terdiri dari nilai-nilai yang diamati dari keadaan sistem/vektor input dan respons/output untuk setiap keadaan (Devenderan, 2025). Supervised Learning adalah kategori Machine Learning yang paling banyak digunakan dalam aplikasi praktis saat ini.

Dalam paradigma ini, model dilatih menggunakan dataset yang berlabel. Artinya, untuk setiap data input yang diberikan selama proses pelatihan, juga memberikan "jawaban yang benar" atau target yang harus dicapai oleh model. Secara matematis, tujuan dari Supervised Learning adalah untuk menemukan atau mempelajari sebuah fungsi pemetaan f yang dapat menghubungkan variabel input (X) ke variabel output (y):

$$y = f(X)$$

Setelah fungsi ini dipelajari melalui proses pelatihan, model dapat digunakan untuk memprediksi output (y) dari data baru yang belum pernah dilihat sebelumnya.

Dua Tugas Utama dalam Supervised LearningBerdasarkan jenis output atau label yang ingin diprediksi, Supervised Learning dibagi menjadi dua tugas utama:

1. Klasifikasi (*Classification*)

   Tugas ini digunakan ketika label atau target bersifat diskret atau berupa kategori (kelas). Model bertugas untuk menentukan ke dalam kategori mana sebuah data baru harus dimasukkan.Contoh: Memprediksi apakah sebuah transaksi kartu kredit adalah "Penipuan" atau "Normal".

   Contoh: Mengidentifikasi gambar sebagai "Kucing", "Anjing", atau "Kelinci".

2. Regresi (*Regression*)

   Tugas ini digunakan ketika label atau target bersifat kontinu atau berupa nilai numerik. Model bertugas untuk memprediksi angka tertentu berdasarkan pola yang ditemukan.

   Contoh 1: Memprediksi harga rumah (misalnya Rp750.000.000) berdasarkan luas tanah dan lokasi.

   Contoh 2: Memprediksi perkiraan suhu udara esok hari.

Karakteristik Utama

Kunci keberhasilan dari Supervised Learning adalah ketersediaan data historis yang berkualitas dan telah dilabeli secara akurat. Sebagaimana didefinisikan dalam kerangka kerja (Mitchell, 1997), pengalaman (E) yang diperoleh model dari dataset berlabel ini akan menentukan seberapa baik kinerja (P) model dalam menjalankan tugas (T) prediksinya.

3.2.2 *Unsupervised Learning*

Unsupervised learning adalah subbidang pembelajaran mesin yang melibatkan pelatihan model pada data tanpa label untuk menemukan pola, struktur, atau pengelompokan yang melekat tanpa bimbingan manusia secara eksplisit. Tidak seperti pembelajaran dengan pengawasan (supervised learning), tidak ada keluaran target (label) yang harus dipetakan oleh algoritma ke masukan; sebaliknya, algoritma tersebut mengidentifikasi kesamaan dan perbedaan dalam data itu sendiri (Nurhalizah, 2024) .

Konsep Dasar: Menemukan Struktur Tersembunyi

Seperti diberikan sekeranjang buah-buahan yang belum pernah dilihat sebelumnya dan tidak tahu nama-

namanya. Meskipun tidak ada "guru" yang memberi tahu nama buah tersebut, Tetapi tetap bisa mengelompokkan buah yang memiliki warna, bentuk, atau tekstur kulit yang serupa. Inilah esensi dari Unsupervised Learning.Dalam konteks matematis, model hanya menerima variabel input (X) tanpa variabel output (y). Tugas model adalah untuk mempelajari distribusi data dan hubungan antar fitur di dalamnya.Tugas Utama dalam Unsupervised LearningTugas dalam kategori ini biasanya berfokus pada pengorganisasian data:

1. *Clustering* (Pengelompokan)

   Ini adalah tugas yang paling umum, di mana model mengelompokkan titik data ke dalam beberapa klaster berdasarkan tingkat kemiripannya. Data dalam satu kelompok akan sangat mirip satu sama lain, namun sangat berbeda dengan data di kelompok lain.

   Contoh: Segmentasi Pelanggan. Sebuah perusahaan menggunakan ML untuk mengelompokkan jutaan pelanggan mereka ke dalam beberapa grup (misalnya: "pemburu diskon", "pembeli setia", atau "pembeli jarang") berdasarkan riwayat transaksi mereka tanpa perlu melabeli mereka satu per satu secara manual.

2. *Dimensionality Reduction* (Pengurangan Dimensi)

   Tugas ini bertujuan untuk menyederhanakan data yang sangat kompleks dengan mengurangi jumlah fitur (kolom) yang digunakan, namun tetap mempertahankan informasi penting di dalamnya.Manfaat: Membantu dalam visualisasi data yang rumit (mengubah data 10 dimensi menjadi 2 dimensi agar bisa dilihat di grafik) dan mempercepat proses pelatihan model lainnya.

Contoh: Teknik Principal Component Analysis (PCA).

3. *Association* (Asosiasi)

   Mencari aturan yang menjelaskan proporsi besar dari dataset. Ini sering digunakan untuk menemukan hubungan antar item dalam dataset yang besar.

   Contoh: Market Basket Analysis. Menemukan pola bahwa pelanggan yang membeli roti cenderung juga membeli selai dalam waktu yang bersamaan.

Dalam dunia nyata, mendapatkan data berlabel sangatlah mahal dan memakan waktu (membutuhkan manusia untuk melabeli setiap baris data). Sebaliknya, data tidak berlabel tersedia dalam jumlah yang sangat melimpah. *Unsupervised Learning* memungkinkan untuk menggali wawasan berharga dari data yang terlihat "berantakan" tersebut tanpa intervensi manusia yang konstan.

3.2.3 *Reinforcement Learning*

Reinforcement Learning (RL) adalah paradigma pembelajaran mesin di mana agen otonom belajar untuk membuat serangkaian keputusan dengan berinteraksi dengan lingkungan untuk memaksimalkan imbalan kumulatif (AlMahamid, 2022).

Konsep Dasar: Belajar dari Umpan Balik

Dalam RL, sebuah entitas yang disebut Agent belajar untuk mengambil keputusan di dalam sebuah Environment (lingkungan). Agent tidak diberi tahu aksi mana yang harus diambil, tetapi ia harus menemukan aksi mana yang menghasilkan Reward (hadiah) terbanyak melalui serangkaian percobaan.

Ini seperti sedang melatih seekor anjing. Jika anjing tersebut melakukan perintah "duduk" dengan benar, Maka akan diberikan makanan (*Reward*). Jika ia tidak melakukannya, ia tidak mendapatkan apa-apa atau bahkan mendapatkan teguran (*Penalty*). Seiring waktu, anjing tersebut akan belajar bahwa "duduk" adalah aksi terbaik untuk mendapatkan makanan.

Komponen Utama *Reinforcement Learning*

Untuk memahami bagaimana RL bekerja, perlu mengenal lima elemen kuncinya:

- *Agent*: Subjek yang belajar dan mengambil keputusan (misalnya: AI dalam video game).
- *Environment*: Segala sesuatu di luar Agent tempat ia berinteraksi (misalnya: level atau arena permainan).
- *Action* (A): Semua langkah atau keputusan yang mungkin diambil oleh Agent.
- *State* (S): Kondisi atau situasi Agent saat ini di dalam lingkungan.
- *Reward* (R): Umpan balik numerik yang diterima Agent (bisa positif sebagai hadiah, atau negatif sebagai hukuman).

Tujuan Akhir: Memaksimalkan Reward Kumulatif

Tujuan dari algoritma RL bukanlah untuk mendapatkan hadiah instan, melainkan untuk menentukan Policy (kebijakan) atau strategi terbaik yang dapat memaksimalkan total hadiah dalam jangka panjang.

Contoh Penerapan:Permainan (Game): Program seperti AlphaGo yang belajar mengalahkan pemain profesional dunia melalui jutaan simulasi pertandingan.

Robotika: Robot yang belajar berjalan atau mengambil benda dengan mencoba berbagai gerakan hingga menemukan keseimbangan yang tepat.

Sistem Rekomendasi: Algoritma yang belajar menampilkan konten yang paling mungkin diklik pengguna berdasarkan interaksi sebelumnya.

Pendekatan dalam RL: *Model-Free vs. Model-Based*

Dalam perjalanannya memaksimalkan reward, seorang agen RL dapat menggunakan dua pendekatan utama untuk memahami lingkungannya. Perbedaan dasarnya terletak pada apakah agen tersebut mencoba membangun "peta mental" tentang bagaimana dunia bekerja atau tidak (Sutton&Barto, 2015).

1. *Model-Free Reinforcement Learning*

   Ini adalah pendekatan yang paling umum. Di sini, agen tidak mencoba memahami hukum fisika atau aturan internal dari lingkungannya. Agen benar-benar belajar murni dari pengalaman langsung (trial and error).

   Cara Kerja: Agen mencoba suatu aksi, melihat hasilnya, dan memperbarui kebijakannya. Jika aksi tersebut menghasilkan hadiah, agen akan cenderung mengulanginya di masa depan tanpa peduli mengapa lingkungan memberikan hadiah tersebut.

   Hal ini seperti seseorang yang belajar naik sepeda. dimana tidak perlu memahami hukum gravitasi atau momentum secara matematis; dimana hanya terus mencoba, jatuh, dan menyesuaikan keseimbangan sampai bisa meluncur.

Contoh Algoritma: Q-Learning, SARSA, dan Deep Q-Network (DQN).

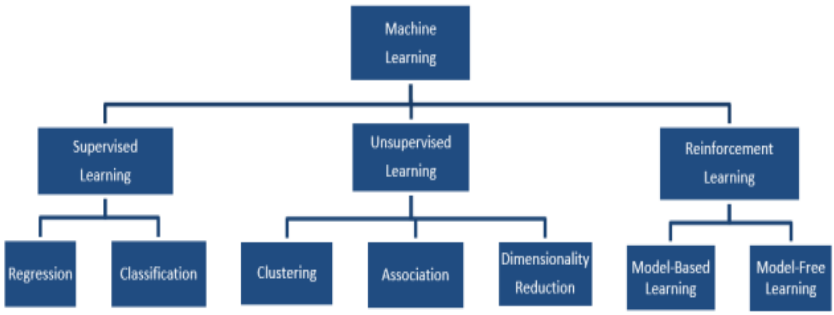
2. *Model-Based Reinforcement Learning*

Dalam pendekatan ini, agen mencoba membangun model internal atau simulasi tentang bagaimana lingkungan akan bereaksi terhadap aksinya. Agen mencoba memprediksi: "Jika saya berada di posisi A dan mengambil aksi B, apa yang akan terjadi selanjutnya dan berapa hadiah yang saya dapatkan?"

Cara Kerja: Agen belajar memahami transisi antar kondisi di lingkungannya. Dengan memiliki "model" ini, agen bisa melakukan perencanaan (planning) atau membayangkan masa depan sebelum benar-benar mengambil tindakan di dunia nyata.

Analogy: Seperti seorang pemain catur. Sebelum menggerakkan bidak, ia membangun model di kepalanya tentang kemungkinan langkah lawan dan hasil akhir dari beberapa langkah ke depan.

Contoh Algoritma: AlphaZero, program AI untuk catur (menggunakan pencarian pohon/tree search untuk merencanakan langkah).



Gambar 3. 2 Algoritma *Machine Learning*
(Sumber : https://www.nerc.com/globalassets/our-work/reports/white-papers/whitepaper-ai-and-ml-in-real-time-system-operations.pdf)

Perbandingan Singkat Tiga Pilar ML:

Pilar	Input Data	Tujuan Utama
Supervised	Data Berlabel (Input + Label)	Memprediksi label untuk data baru.
Unsupervised	Data Tanpa Label (Input saja)	Menemukan struktur atau pola tersembunyi.
Reinforcement	Interaksi (Aksi + Reward)	Menemukan aksi terbaik melalui umpan balik.

3.3 Data: Bahan Baku, Struktur, dan Pembagian Set

Jika algoritma *Machine Learning* adalah mesinnya, maka data adalah bahan bakarnya. Seberapa canggih pun mesin yang dibangun (seperti Deep Learning atau arsitektur kompleks lainnya), mesin tersebut tidak akan memberikan hasil yang optimal jika bahan bakar yang digunakan kotor atau tidak sesuai. Dalam dunia *Machine Learning,* terdapat sebuah prinsip emas yang dikenal sebagai "*Garbage In, Garbage Out*" (GIGO)—jika data yang dimasukkan buruk, maka prediksi yang dihasilkan pun akan buruk.

3.3.1 Struktur Data dalam *Machine Learning*

Dalam kebanyakan kasus, data yang diolah menggunakan Python akan berbentuk Data Tabular (seperti spreadsheet atau tabel database). Memahami anatomi tabel ini adalah langkah pertama sebelum melakukan proses coding.

- Sampel (*Observation/Row*): Setiap baris dalam dataset mewakili satu unit observasi tunggal. Misalnya, dalam dataset medis, satu baris mewakili satu pasien.
- Fitur (*Feature/Column*): Setiap kolom (selain kolom target) mewakili karakteristik atau variabel yang mendeskripsikan sampel tersebut.
- Target (*Label*): Kolom khusus yang berisi jawaban atau nilai yang ingin diprediksi oleh model.

3.3.2 Pembagian Dataset (*Data Splitting*)

Salah satu kesalahan yang paling umum adalah melatih model dan mengujinya menggunakan data yang sama. Hal ini akan menyebabkan model "menghafal" data daripada "belajar", sebuah kondisi yang disebut *overfitting*. Untuk mencegah hal ini dan memastikan

model dapat melakukan generalisasi pada data baru, harus membagi dataset menjadi tiga bagian utama:

1. Data Latih (*Training Set*): Porsi terbesar dari data (biasanya 70-80%) yang digunakan oleh algoritma untuk mempelajari pola. Di sinilah model mendapatkan "pengalaman" (E) sebagaimana didefinisikan oleh (Mitchell, 1997).
2. Data Validasi (*Validation Set*): Digunakan selama proses pengembangan untuk memantau kinerja model dan menyetel parameter (*hyperparameter*) agar mendapatkan hasil terbaik tanpa "melihat" data uji.
3. Data Uji (*Test Set*): Sekumpulan data yang benar-benar asing bagi model. Data ini hanya digunakan di bagian paling akhir untuk memberikan penilaian objektif tentang seberapa baik model akan bekerja di dunia nyata.

Penggunaan 2 bagian (*split*) atau 3 bagian sangat bergantung pada ketersediaan data dan kompleksitas model yang sedang dibangun.

1. Skenario 2 Bagian: *Training Set* dan *Test Set*

   Skenario ini adalah yang paling sederhana dan sering digunakan dalam tutorial dasar atau ketika dataset yang dimiliki terbatas (sedikit).

   Pembagian: Biasanya 70% Training dan 30% Test, atau 80:20.

   Kapan digunakan penggunaannya, saat dataset kecil, sehingga jika dibagi menjadi 3, jumlah data di setiap bagian menjadi terlalu sedikit untuk dipelajari model.

   Saat hanya menggunakan satu algoritma dengan parameter standar (tidak melakukan banyak eksperimen untuk mencari setelan terbaik).

Risiko: Jika mencoba berbagai parameter (misal: mengubah jumlah K pada K-NN) dan berulang kali mengecek hasilnya pada Test Set, secara tidak sengaja model mulai mengenali pola di Test Set. Hal Ini disebut Data Leakage (kebocoran data), yang membuat hasil evaluasi menjadi tidak murni lagi.

2. Skenario 3 Bagian: Training, Validation, dan Test Set

   Ini adalah Standar kebanyakan dalam penelitian Machine Learning, terutama untuk model yang kompleks.

   Pembagian: Biasanya 60% Training, 20% Validation, dan 20% Test atau 70% Training, 15% Validation, dan 15% Test

   Mengapa butuh Validation Set, dimana:

   - *Training Set*: Tempat model belajar.
   - *Validation Set:* Digunakan sebagai "ujian percobaan" untuk memilih model terbaik atau menyetel hyperparameter (seperti menentukan jumlah lapisan pada *Neural Network*).
   - *Test Set:* Digunakan hanya satu kali di akhir proyek untuk memberikan skor final yang jujur.

   Dalam hal ini training adalah buku pelajaran, Validation adalah latihan soal/try-out, dan Test adalah ujian nasional yang sesungguhnya.

3.3.3 Pentingnya Kualitas Data

Sebelum data siap digunakan oleh model, seringkali harus melakukan pembersihan. Data raw di dunia nyata jarang sekali sempurna; seringkali terdapat nilai yang

hilang (*missing values*), data yang tidak konsisten, atau outliers (pencilan) yang dapat mengganggu logika matematis model. Oleh karena itu, tahap pemrosesan data—yang akan dibahas mendalam pada berikut menjadi kunci kesuksesan implementasi (Krüger, 2025).

Dalam konteks pembelajaran mesin (ML), Lucas Rotta Krüger menekankan bahwa kualitas data (DQ) merupakan hambatan kritis, dengan menyatakan bahwa kinerja model ML apa pun dibatasi oleh kualitas data pelatihannya. Beberapa alasan utama mengapa kualitas data sangat penting:

Keandalan dan Akurasi Model, Data berkualitas tinggi yang dicirikan oleh akurasi, kelengkapan, dan konsistensi diperlukan agar model dapat menangkap pola mendasar yang sebenarnya. Data yang buruk menyebabkan model yang bias atau tidak akurat yang menghasilkan prediksi yang tidak dapat diandalkan.

Dimensi Kualitas, identifikasi tujuh dimensi inti kualitas data yang harus dikelola: akurasi, kelengkapan, konsistensi, validitas, ketepatan waktu, keunikan, dan integritas.

Dampak pada Keamanan, Di bidang khusus seperti keamanan jaringan, DQ sangat penting karena prinsip "sampah masuk, sampah keluar" berlaku; data jaringan berkualitas rendah dapat membuat sistem rentan dengan menghasilkan model deteksi ancaman yang tidak efektif.

Keberlanjutan Operasional, Memastikan kualitas data merupakan keharusan strategis untuk keberhasilan penerapan dan pemeliharaan pipeline ML. Perlu dicatat bahwa tanpa profil dan validasi data yang ketat, model menjadi rapuh dan menurun kualitasnya seiring waktu.

Interpretabilitas dan Kepercayaan, Kualitas data secara langsung memengaruhi seberapa baik suatu model dapat diinterpretasikan. Wawasan yang menyesatkan dari data yang buruk mengurangi kepercayaan pengguna dan dapat menyebabkan pengambilan keputusan yang salah.

Meskipun banyak penelitian ML berfokus pada peningkatan algoritma, bidang ini harus bergeser ke arah rekayasa yang berpusat pada data untuk mengatasi biaya dan risiko tinggi yang terkait dengan dataset berkualitas rendah.

DAFTAR PUSTAKA

Adams Rob et al (NERC), 2024. Artificial Intelligence and Machine Learning in Real-Time System Operations: White Paper – Revision 1. NERC, E-ISAC, the ERO Enterprise, SITES, SWG, EMSWG, and others.

AlMahamid Fadi, Grolinger Katarina, 2022. Reinforcement Learning Algorithms: An Overview and Classification, Department of Electrical and Computer Engineering Western University

Awad, M., Khanna, R., 2015. Machine Learning. In: Efficient Learning Machines. Apress, Berkeley, CA.

https://doi.org/10.1007/978-1-4302-5990-9_1

Devenderan V, 2025 Machine Learning DECAP737. Directorate of Distance Education at Lovely Professional University (LPUDE)

Kruger Lucas Rotta, 2025. Data Quality in Machine Learning and Network Security: A Systematic Literature Review, UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Mitchell Tom, 1997. Machine Learning, McGraw Hill

Nurhalizah Ria Suci, Ardianto Rian, Purwono, 2024, Analisis Supervised dan Unsupervised Learning pada Machine Learning: Systematic Literature Review, Jurnal Ilmu Komputer dan Informatika (JIKI) . DOI: https://doi.org/10.54082/jiki.168.

Shai Shalev-Shwartz and Shai Ben-David, 2014. Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press.

Sutton Richard S. and Barto Andrew G, 2015 Reinforcement Learning: An Introduction, Second edition, Bradford Book, The MIT Press

Waghmare, A.V., Singh, V.P., Varshney, T. *et al.* A systematic review of reinforcement learning-based control for microgrids: trends, challenges, and emerging algorithms. *Discov Appl Sci* 7, 939 (2025). https://doi.org/10.1007/s42452-025-07529-6

BAB 4

LINGKUNGAN PEMROGRAMAN PYTHON UNTUK AI DAN ML

Bab ini membahas tentang lingkungan pemrograman Python yang digunakan dalam pengembangan *Artificial Intelligence* (AI) dan *Machine Learning* (ML). Pembahasan difokuskan pada aspek konseptual sekaligus praktis, mulai dari alasan pemilihan Python, proses instalasi dan konfigurasi, manajemen lingkungan dan dependensi, hingga *workflow* pengembangan model AI/ML yang terstruktur. Bab ini menjadi fondasi penting sebelum memasuki pembahasan struktur data, pemrosesan data, algoritma, dan *framework* AI pada bab-bab selanjutnya.

4.1 Pengenalan Python dalam Konteks AI dan ML

Python adalah bahasa pemrograman *high-level* yang bersifat universal, dikembangkan oleh **Guido van Rossum** dan pertama kali diperkenalkan pada tahun **1991**. Bahasa ini sejak awal dirancang dengan sintaks yang sederhana dan mudah dibaca, sehingga memudahkan pengguna dalam memahami dan menulis kode program. Karakteristik tersebut membuat Python sangat ramah bagi pemula, namun tetap memiliki kemampuan yang memadai untuk menangani permasalahan komputasi yang kompleks (Blue Coding, 2024).

Dari sisi efisiensi, Python memiliki kinerja yang memadai untuk sebagian besar aplikasi, khususnya pada pengembangan contoh kasus berskala kecil hingga menengah. Apabila performa menjadi kendala, pendekatan umum yang digunakan adalah mengidentifikasi bagian kode yang paling banyak memakan waktu eksekusi, kemudian mengoptimalkan bagian tersebut menggunakan bahasa tingkat rendah seperti *C* atau *C++*. Python mendukung interoperabilitas yang sangat baik dengan berbagai bahasa tingkat rendah, sehingga hanya bagian kritis yang dioptimalkan, sementara logika utama tetap ditulis dalam Python. Pendekatan ini menghasilkan kode yang lebih efisien dengan usaha pemrograman yang lebih sedikit dibandingkan menulis seluruh program dalam bahasa tingkat rendah, terlebih karena banyak fungsi intensif telah diimplementasikan secara optimal dalam *library* Python yang lengkap(Poole and Mackworth, 2025).

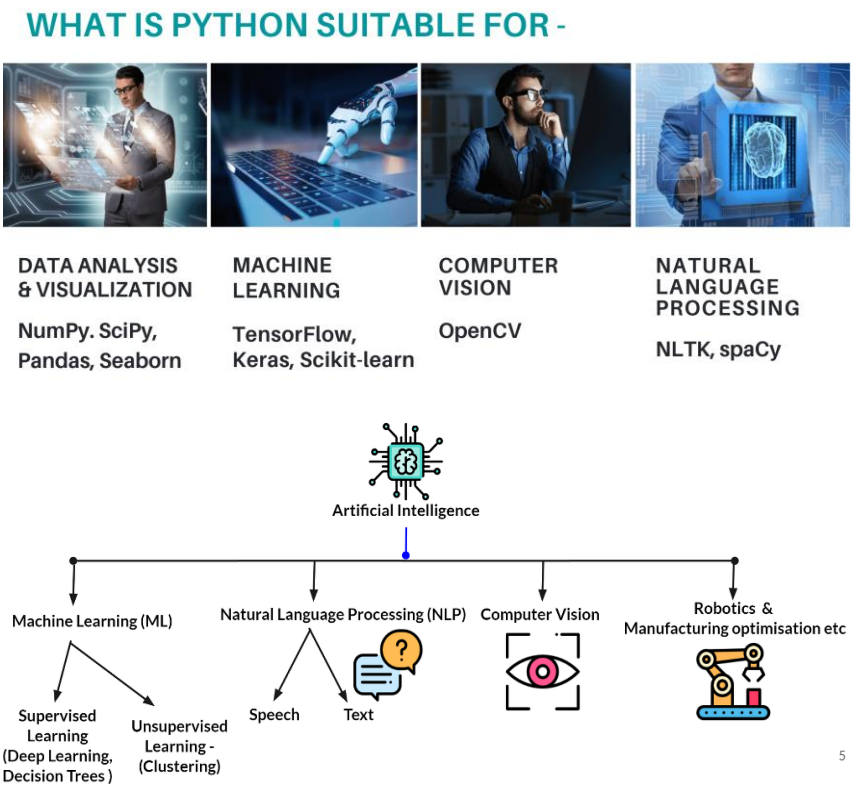
Python sendiri dikembangkan oleh Guido van Rossum di *Stichting Mathematics Centrum* (CWI), Belanda, sebagai penerus bahasa pemrograman ABC. Seiring perkembangannya, Python berevolusi menjadi bahasa pemrograman tingkat tinggi yang bersifat umum dan digunakan secara luas di berbagai bidang, mulai dari pengembangan perangkat lunak hingga komputasi ilmiah dan kecerdasan buatan (Gupta and Bagchi, 2024). Python kini berkembang pesat sebagai bahasa pemrograman utama yang dipilih oleh para pengembang untuk membangun aplikasi berbasis ***Aritificial Intelligence***, ***Machine Learning***, dan ***Deep Learning***.

Machine learning merupakan pengembangan lanjutan dari metode ***Artificial Intelligence*** yang menggunakan algoritma untuk menganalisis data, mengenali pola, dan menghasilkan keputusan yang lebih tepat. Sedangkan, ***Deep Learning*** bekerja dengan prinsip serupa, namun memiliki kemampuan yang lebih kompleks karena dapat menghasilkan

kesimpulan yang menyerupai cara manusia mengambil keputusan. Hal ini dimungkinkan melalui penggunaan lapisan (*layer*) algoritma yang tersusun baik yang diwujudkan dalam bentuk jaringan saraf tiruan (***artificial neural networks***) yang terinspirasi dari pemahaman kita mengenai cara kerja otak manusia.

Dalam konteks ***Machine Learning***, Python sangat membantu karena memungkinkan implementasi ide secara cepat. Seorang *machine learning engineer* atau *data scientist* dapat dengan mudah mengekstraksi, memproses, dan menganalisis data untuk memvalidasi suatu gagasan dalam waktu yang relatif singkat dan dengan usaha yang lebih efisien, sehingga mempercepat proses pengujian dan pengambilan keputusan (Gupta and Bagchi, 2024).

Berkat fleksibilitas dan kemudahan penggunaannya, Python digunakan secara luas di berbagai bidang teknologi. Python banyak dimanfaatkan dalam pengembangan aplikasi web, analisis dan pengolahan data, **Kecerdasan Buatan (*Artificial Intelligence/AI*)**, ***Machine Learning* (*ML*)**, *automation*, hingga pengembangan *game*. Cakupan penggunaan yang luas ini menjadikan Python sebagai salah satu bahasa pemrograman paling populer dan relevan dalam dunia komputasi modern.

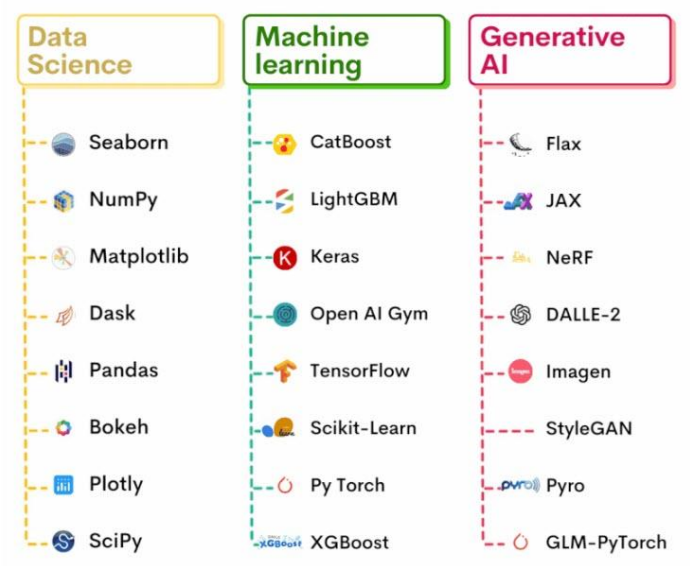


Gambar 4. 1 Python dalam Pengembangan AI/ML
(Sumber: (K. Thiyagu, no date; Jayesh Totla, 2025))

Python merupakan bahasa pemrograman tingkat tinggi yang saat ini banyak digunakan sebagai standar utama dalam pengembangan **Kecerdasan Buatan (*Artificial Intelligence/AI*)** dan **Pembelajaran Mesin (*Machine Learning/ML*)**. Popularitas Python didorong oleh sintaks yang sederhana, mudah dipelajari, serta dukungan ekosistem *library* yang sangat luas (Gupta & Bagchi, 2023). Kondisi ini menjadikan Python bersifat inklusif, sehingga dapat digunakan oleh berbagai kalangan, mulai dari mahasiswa hingga peneliti tingkat lanjut. Selain itu, Python bersifat *open-*

source dan didukung oleh komunitas global yang sangat aktif, sehingga pengembangan *library* AI dan ML berlangsung cepat dan berkelanjutan (Kamlesh Singad, 2024).

Python memiliki sejumlah karakteristik yang mendukung pengembangan sistem komputasi cerdas. Salah satu keunggulannya adalah fleksibilitas dalam mendukung berbagai paradigma pemrograman, seperti *procedural programming*, *object-oriented programming*, dan *functional programming*. Python juga terintegrasi dengan *library* numerik seperti *NumPy* dan *SciPy*, yang memudahkan manipulasi data dan perhitungan matematis kompleks (Klein, 2021). Selain itu, Python mendukung *parallel computing* dan pemanfaatan *Graphics Processing Unit* (*GPU*) melalui *framework* seperti ***TensorFlow*** dan ***PyTorch***, sehingga mampu digunakan untuk pelatihan model ***deep learning*** berskala besar secara efisien (Educate360, 2025). Dengan karakteristik tersebut, Python menjadi bahasa yang ideal untuk riset maupun implementasi aplikasi AI di dunia industri.



Gambar 4. 2 Ekosistem Python dalam Bidang *Data Science*, *Machine Learning*, dan *Generative AI* (Sumber: (*Pustaka Python dalam Data Science, Machine Learning, dan Generative AI*, 2025))

Ekosistem Python untuk AI dan ML tergolong sangat luas dan lengkap. Python menyediakan *library* fundamental untuk komputasi numerik dan analisis data, seperti ***NumPy***, ***Pandas***, dan ***Matplotlib***, serta *framework deep learning* berskala industri seperti ***TensorFlow***, ***PyTorch***, dan ***Keras*** (Aurélien Géron, 2019). Selain itu, Python juga memiliki pustaka khusus untuk bidang tertentu, misalnya Pemrosesan Bahasa Alami (*Natural Language Processing/NLP*) melalui ***NLTK*** dan ***spaCy*** (Yuli Vasiliev, 2020), serta visi komputer (*computer vision*) menggunakan ***OpenCV*** (Rick van Hattem, 2022). Meskipun Python memiliki keterbatasan dalam performa komputasi murni, keterbatasan tersebut dapat diatasi melalui integrasi dengan *library* berbasis ***C/CUDA***,

sehingga performa tetap dapat dioptimalkan secara signifikan ((Zinovyev, 2021)).

Kelebihan utama Python terletak pada kesederhanaan (***simplicity***), ekosistem *library* yang matang, serta dukungan komunitas global yang luas. Sintaks Python yang mudah dipahami membuat bahasa ini cepat dipelajari oleh pemula (Mark Lutz, 2009). Ekosistem *library* yang lengkap memungkinkan pengembangan aplikasi AI dilakukan lebih cepat dan efisien, sementara komunitas global menyediakan dokumentasi, forum diskusi, dan sumber belajar daring (*online*) yang sangat membantu proses pembelajaran (GeeksforGeeks, 2025).

Gambar 4. 3 Karakteristik dan Keunggulan Python
(Sumber: (Blue Coding, 2024))

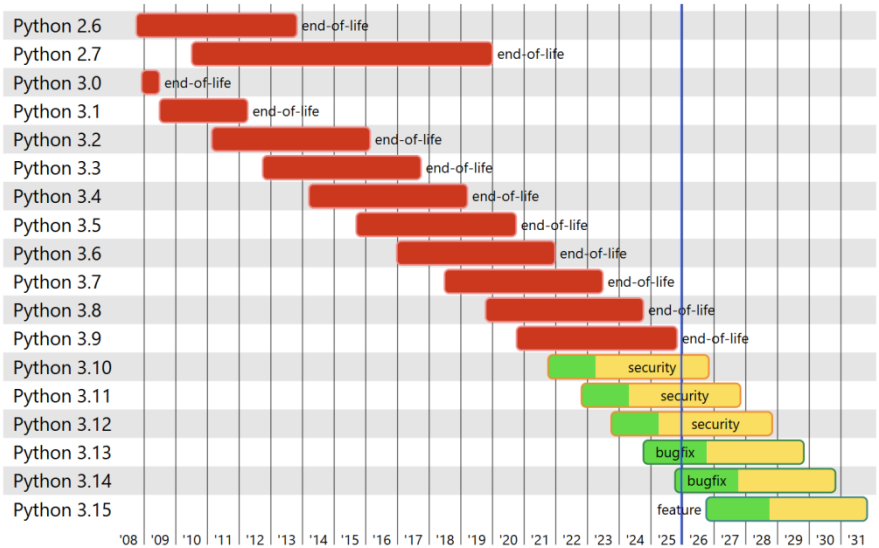
Namun demikian, Python juga memiliki beberapa keterbatasan. Salah satunya adalah kecepatan eksekusi yang relatif lebih lambat dibandingkan bahasa terkompilasi (*compiled*) seperti *C++* atau *Java*. Bahasa terkompilasi adalah bahasa pemrograman yang kode sumbernya diterjemahkan terlebih dahulu (*compiled*) oleh ***compiler*** menjadi kode mesin sebelum dijalankan oleh komputer. Proses ini menghasilkan program yang dapat dieksekusi langsung oleh perangkat keras, sehingga memiliki performa yang lebih tinggi. Sebaliknya, Python termasuk bahasa terinterpretasi (*interpreted*), di mana kode dijalankan baris demi baris oleh ***interpreter*** pada saat program dieksekusi. Perbedaan mekanisme ini menyebabkan Python memiliki beban tambahan (*overhead*), sehingga kurang optimal untuk aplikasi waktu nyata (*real-time*) yang menuntut performa sangat tinggi (Yamuna, 2025).

Dalam praktik pengembangan AI dan ML, keterbatasan tersebut umumnya tidak menjadi hambatan utama. Python sering digunakan sebagai antarmuka tingkat tinggi (*high-level interface*), sementara komputasi berat dijalankan oleh *library* yang ditulis dalam bahasa terkompilasi seperti *C* atau *C++*. Pendekatan ini memungkinkan Python tetap efisien dan andal dalam aplikasi AI berskala besar.

Versi Python yang paling umum digunakan dalam pengembangan AI dan ML saat ini adalah **Python 3**, khususnya pada rentang versi **3.8 hingga 3.11**. Versi-versi ini menawarkan peningkatan performa, kompatibilitas dengan *library* modern, serta fitur-fitur baru yang mendukung pengembangan perangkat lunak secara lebih efisien, seperti *structural pattern matching* (Python Software Foundation, 2025). Sebagian besar *framework* AI, seperti *TensorFlow* dan *PyTorch*, telah mendukung *Python* versi 3.9 hingga 3.11. Jika dibandingkan dengan bahasa lain, Python unggul dalam fleksibilitas dan kelengkapan ekosistem *library*. Sementara itu, ***R*** lebih kuat dalam analisis statistik, ***MATLAB*** unggul

dalam komputasi numerik dan simulasi, serta ***Java*** lebih sesuai untuk aplikasi perusahaan (*enterprise*) yang membutuhkan skalabilitas (*scalability*) tinggi (Deitel and Deitel, 2017; MathWorks, 2025; Venables, Smith and R Core Team, 2025).

Seiring perkembangan terkini, Python terus dikembangkan dan dirilis secara berkala dengan siklus tahunan, di mana setiap versi baru membawa peningkatan kinerja, stabilitas, dan keamanan. Versi Python di atas 3.11 menunjukkan fokus yang semakin kuat pada optimasi *runtime*, efisiensi pengelolaan memori, serta dukungan yang lebih baik terhadap komputasi modern, termasuk pemrosesan paralel dan integrasi akselerator perangkat keras. Namun, dalam konteks **AI dan ML**, adopsi versi Python umumnya dilakukan secara bertahap karena menunggu kesiapan dan kompatibilitas penuh dari *library* utama. Sebagian besar versi Python lama (termasuk seluruh Python 2.x serta Python 3.0 hingga 3.8) telah mencapai status *end of life* (EOL) dan tidak lagi menerima pembaruan resmi, sementara Python 3.9 saat ini berada pada fase dukungan keamanan dan dijadwalkan mencapai EOL pada Oktober 2025. Oleh karena itu, pemilihan versi Python yang masih aktif didukung dan stabil menjadi faktor penting untuk menjamin keamanan sistem, keberlanjutan pengembangan, serta kompatibilitas dengan ekosistem AI yang terus berkembang, seiring dengan rencana rilis versi Python terbaru pada tahun 2025 yang membawa peningkatan performa dan fitur bahasa baru.



Gambar 4. 4 Perkembangan Versi Python
(Sumber:(Python Software Foundation, 2025))

Untuk mempersiapkan serta mengetahui versi Python yang digunakan, berikut ditampilkan baris program yang dapat dijalankan melalui *interpreter* Python maupun ***Jupyter Notebook***.

Kode Program Phyton

```
# Menjalankan interpreter Python dan
menampilkan versi Python

import sys

print("Versi Python yang digunakan:",
sys.version)


# Contoh komputasi sederhana dengan NumPy

import numpy as np

A = np.array([[1, 2], [3, 4]])
```

```
B = np.array([[2, 0], [1, 3]])
print("Hasil perkalian matriks:\n", np.dot(A, B))
```

Kode program ini menunjukkan dua hal penting yang dapat dilakukan pada tahap awal penggunaan Python dalam konteks AI dan ML. Pertama, baris import sys diikuti dengan perintah print("Versi Python yang digunakan:", sys.version) berfungsi untuk menampilkan versi Python yang sedang aktif. Informasi ini sangat penting untuk memastikan kompatibilitas dengan pustaka AI/ML yang akan digunakan, karena beberapa *library* hanya mendukung versi tertentu dari Python.

Kedua, bagian kode yang menggunakan pustaka **NumPy** (import numpy as np) memberikan contoh komputasi dasar berupa perkalian matriks. Pada contoh ini, dua *array* dua dimensi didefinisikan sebagai matriks A dan B, kemudian dilakukan operasi perkalian matriks dengan fungsi np.dot(A, B). Hasil perhitungan ditampilkan melalui perintah print, sehingga pengguna dapat langsung mengamati output yang dihasilkan.

Contoh sederhana ini memperlihatkan bagaimana Python dapat digunakan untuk keperluan verifikasi lingkungan pemrograman sekaligus menjalankan operasi matematis dasar.

4.2 Fitur Phyton

Python merupakan bahasa pemrograman berorientasi objek (*object-oriented programming*) yang bersifat jelas, kuat, dan fleksibel, serta sebanding dengan bahasa pemrograman lain seperti *Perl*, *Ruby*, *Scheme*, dan *Java*. Keunggulan Python tercermin dari berbagai fitur berikut (*BeginnersGuide/Overview - Python Wiki*, 2024).

1. **Sintaks yang Elegan dan Mudah Dibaca**
   Python menggunakan sintaks yang sederhana dan konsisten, sehingga program yang ditulis menjadi lebih mudah dibaca, dipahami, dan dipelihara oleh pengembang.
2. **Mudah Digunakan dan Cepat Dikembangkan**
   Python dirancang agar mudah digunakan, sehingga program dapat dijalankan dengan cepat. Hal ini menjadikan Python sangat sesuai untuk pengembangan prototipe (*prototype development*) dan pemrograman *ad-hoc* tanpa mengorbankan keterpeliharaan (*maintainability*) kode.
3. **Pustaka Standar yang Lengkap**
   Python dilengkapi dengan *standard library* yang sangat besar, yang mendukung berbagai tugas pemrograman umum, seperti koneksi ke *web server*, pencarian teks menggunakan *regular expression*, serta proses membaca dan memodifikasi berkas.
4. **Mode Interaktif dan Lingkungan Pengembangan Bawaan**
   Python menyediakan mode interaktif yang memungkinkan pengujian potongan kode secara langsung. Selain itu, Python menyertakan *Integrated Development Environment* bawaan yang disebut *IDLE* untuk mendukung proses pembelajaran dan pengembangan dasar.
5. **Mudah Diperluas dengan Bahasa Terkompilasi**
   Python dapat diperluas dengan menambahkan modul baru yang diimplementasikan menggunakan bahasa terkompilasi seperti *C* atau *C++*, sehingga kinerja program dapat ditingkatkan pada bagian tertentu yang membutuhkan efisiensi tinggi.
6. **Dapat Disematkan ke dalam Aplikasi Lain**
   Python dapat di-*embed* ke dalam aplikasi lain untuk menyediakan antarmuka pemrograman yang dapat dikonfigurasi sesuai kebutuhan.

7. Lintas Platform (*Cross-Platform*)

Program Python dapat dijalankan pada berbagai sistem operasi seperti *Windows*, *macOS*, *Linux*, dan *Unix*. Selain itu, tersedia pula versi tidak resmi untuk *Android* dan *iOS*.

8. Perangkat Lunak Bebas dan *Open-Source*

Python merupakan perangkat lunak bebas, baik dalam arti bebas biaya untuk diunduh dan digunakan, maupun bebas dimodifikasi serta didistribusikan ulang karena dilisensikan secara *open-source*.

9. Tipe Data Dasar yang Beragam

Python menyediakan berbagai tipe data dasar, seperti bilangan numerik (bilangan pecahan, kompleks, dan bilangan bulat dengan panjang tak terbatas), *string* dalam format ASCII dan Unicode, serta struktur data seperti *list* dan *dictionary*.

10. Dukungan Pemrograman Berorientasi Objek

Python mendukung pemrograman berorientasi objek melalui penggunaan kelas dan pewarisan ganda (*multiple inheritance*), yang memungkinkan pengembangan perangkat lunak yang terstruktur.

11. Modularitas Program

Kode Python dapat dikelompokkan ke dalam *module* dan *package*, sehingga memudahkan pengelolaan, pemeliharaan, dan pengembangan kode program.

12. Penanganan Kesalahan dengan *Exception*

Python mendukung mekanisme *exception handling* untuk menangani kesalahan secara lebih bersih dan terstruktur.

13. Tipe Data Kuat dan Dinamis

Python menggunakan sistem tipe data yang bersifat kuat (*strongly typed*) dan dinamis (*dynamically typed*), sehingga kesalahan akibat pencampuran tipe data yang tidak kompatibel dapat terdeteksi lebih awal.

14. **Fitur Pemrograman Lanjutan**
    Python menyediakan fitur pemrograman tingkat lanjut seperti *generator* dan *list comprehension*, yang memungkinkan penulisan kode lebih ringkas dan efisien.
15. **Manajemen Memori Otomatis**
    Python dilengkapi dengan sistem manajemen memori otomatis yang membebaskan pengembang dari kewajiban mengatur alokasi dan pelepasan memori secara manual.

4.3 Instalasi dan Konfigurasi Python

Bahasa pemrograman *Python* dikelola dan disediakan secara resmi oleh *Python Software Foundation* melalui situs *https://www.python.org*, yang menjadi sumber utama untuk mengunduh *Python* dan dokumentasi pendukungnya. Paket unduhan *Python* mencakup fitur dasar yang diperlukan untuk mulai memprogram, yaitu *Python interpreter* serta sebuah editor kode sederhana atau *Integrated Development Environment* (IDE) bawaan yang dikenal sebagai *IDLE*. Paket ini merepresentasikan inti (*core*) dari *Python*, yang menyediakan komponen minimal untuk menulis dan menjalankan program *Python* dasar. Meskipun *Python* mendukung paradigma *Object-Oriented Programming* (OOP), pengguna tetap dapat memanfaatkan *Python* untuk aplikasi sederhana tanpa harus memahami atau menerapkan konsep OOP secara mendalam. Selain itu, *Python* merupakan bahasa pemrograman *interpreted*, di mana berkas program berekstensi *.py* dapat ditulis menggunakan editor teks, kemudian dijalankan melalui *Python interpreter*. Proses eksekusi ini dapat berlangsung secara otomatis atau manual, tergantung pada editor atau lingkungan pengembangan yang digunakan (Gupta and Bagchi, 2024).

4.3.1 Menyiapkan Lingkungan Pengembangan Python

Untuk membangun lingkungan pengembangan Python, ikuti langkah-langkah berikut (Educate360, 2025).

1. Instalasi Python

Unduh dan pasang versi Python terbaru (disarankan menggunakan Python 3.9 atau versi yang lebih baru agar kompatibel dengan library AI dan ML modern) melalui situs resmi Python (*python.org*) atau menggunakan *package manager* seperti *Anaconda* agar pengelolaan dependensi lebih mudah dan terstruktur.

***Download* Phyton**

a) Untuk pengguna *Windows*, Python dapat diunduh langsung dari *python.org*. Saat proses instalasi, penting untuk mencentang opsi "Add Python to PATH" agar Python dapat dijalankan dari *Command Prompt* atau *PowerShell* tanpa konfigurasi tambahan.

b) Pada sistem operasi *Linux* berbasis *Ubuntu* atau *Debian*, perintah berikut digunakan untuk memperbarui repositori dan memasang Python 3.10 beserta modul pengembangannya serta *pip* sebagai pengelola paket.

```
# Ubuntu/Debian
sudo apt update
sudo apt install python3.10
python3.10-dev python3-pip
```

c) Pada *macOS*, Python dapat diinstal dengan mudah melalui *Homebrew*, yaitu pengelola paket yang memudahkan instalasi dan pembaruan perangkat lunak.

```
# macOS (menggunakan Homebrew)
brew install python@3.10
```

Install Phyton

a) Lakukan proses instalasi Python sesuai dengan sistem operasi yang digunakan, baik itu *Windows*, *macOS*, maupun *Linux*, dengan mengikuti panduan yang tersedia pada situs resminya.
b) Saat instalasi berlangsung, pastikan opsi untuk menambahkan Python ke *PATH* diaktifkan agar Python dapat dijalankan langsung melalui terminal atau *command line*.

Verifikasi Instalasi

a) Buka *terminal*, *Command Prompt*, atau *PowerShell*, lalu ketik perintah berikut untuk memastikan Python dan *pip* telah terpasang dengan benar:

```
python --version
pip –version
```

b) Jika instalasi berhasil, perintah tersebut akan menampilkan nomor versi Python dan *pip* yang sedang terpasang di sistem.

Pengaturan *Virtual Environment*

Untuk menjaga setiap proyek tetap terpisah dan mencegah bentrokan antar *library* atau dependensi, penggunaan *virtual environment* sangat dianjurkan.

a) Membuat *Virtual Environment*

Perintah berikut akan membuat sebuah lingkungan Python terisolasi dengan nama *env*, yang berisi Python dan *package manager* tersendiri.

```
python -m venv env
```

b) Aktivasi *Virtual Environment*

Setelah diaktifkan, semua *library* yang diinstal akan tersimpan di dalam lingkungan ini sehingga tidak memengaruhi instalasi Python global di sistem.

- ✓ **Pada Windows**

```
.\env\Scripts\activate
```

- ✓ **Pada macOS/Linux**

```
source env/bin/activate
```

2. Pilih *Integrated Development Environment* (IDE)

Tentukan IDE yang sesuai dengan kebutuhan kerja Anda dan menyediakan fasilitas penulisan kode, *debugging*, serta eksekusi program, seperti *Jupyter Notebook*, *PyCharm*, atau *Visual Studio Code*.

Jupyter Notebook menyediakan lingkungan kerja yang interaktif untuk menulis, menjalankan, dan mengevaluasi kode program serta visualisasi data, sehingga sangat cocok digunakan dalam proses pembelajaran dan eksperimen AI/ML.

a) Instalasi *Jupyter Notebook*

```
pip install notebook
```

b) Menjalankan *Jupyter Notebook*

Perintah berikut akan membuka antarmuka berbasis web pada *browser*.

jupyter notebook

c) Membuat Notebook Baru
Klik menu **New > Python 3 Notebook**, lalu mulai menuliskan dan menjalankan kode.

Menyiapkan *Google Colab* (Opsional)

Bagi pengguna yang tidak ingin melakukan instalasi di komputer lokal, *Google Colab* dapat menjadi pilihan yang praktis. Layanan ini gratis dan menyediakan akses ke GPU untuk mempercepat pelatihan model AI.

a) Mengakses *Google Colab*
Buka situs *colab.research.google.com* melalui *internet browser*.
b) Membuat Notebook Baru
Pilih menu **New Notebook** untuk memulai sesi kerja baru.
c) Menginstal *Library* (jika diperlukan)
Sebagian besar *library* seperti *NumPy* dan *pandas* sudah tersedia. Namun, jika membutuhkan *library* lain, instalasi dapat dilakukan dengan perintah:

```
!pip install nama-library
```

3. Instalasi *library* dan *framework*

Setelah Python siap digunakan, tambahkan *library* serta *framework* Python seperti *TensorFlow*, *PyTorch*, dan *Scikit-Learn* untuk memanfaatkan fungsi dan algoritma siap pakai yang dapat mempercepat pengembangan aplikasi *Artificial Intelligence* dan *Machine Learning*.

a) *NumPy* digunakan untuk komputasi numerik dan pengolahan array.

```
pip install numpy
```

b) *Pandas* digunakan ntuk pengelolaan, pembersihan, dan analisis data.

pip install pandas

c) *Matplotlib* dan *Seaborn* digunakan ntuk membuat grafik dan visualisasi data.

pip install matplotlib seaborn

d) *Scikit-learn* digunakan untuk menyediakan berbagai algoritma dan alat pembelajaran mesin dasar.

pip install scikit-learn

e) *TensorFlow / PyTorch* digunakan untuk pengembangan dan pelatihan model *deep learning*.

```
pip install tensorflow
```

atau

```
pip install torch torchvision
```

Pengguna dapat memilih salah satu dari *framework* tersebut sesuai dengan kebutuhan dan preferensi dalam membangun model AI dan ML.

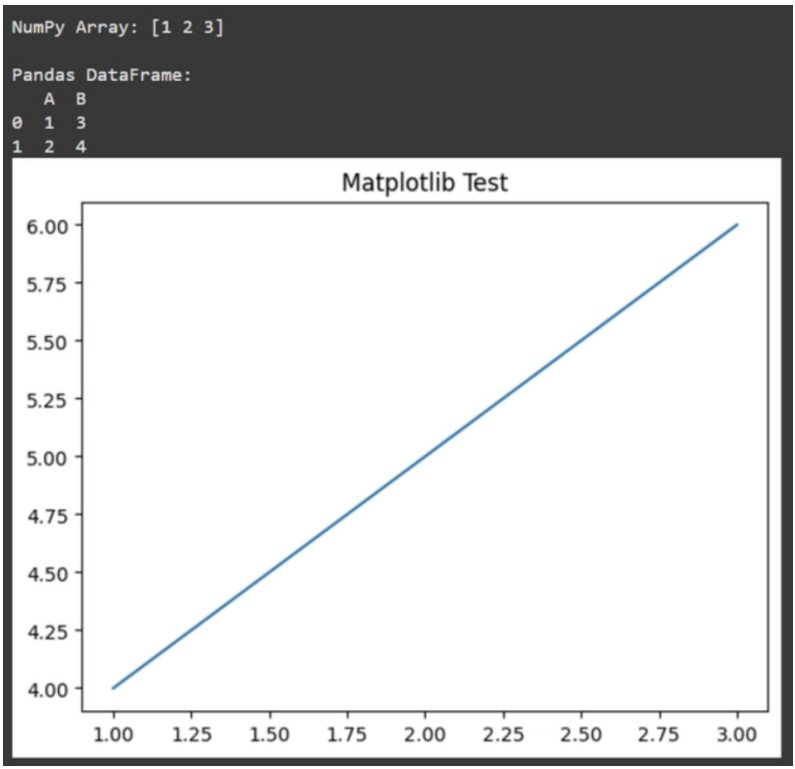
4. Menguji Konfigurasi Sistem

Untuk memastikan seluruh komponen telah terpasang dengan benar, jalankan contoh kode program berikut di *Jupyter Notebook* atau *Google Colab* (Trix Cyrus, 2024).

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Uji NumPy
arr = np.array([10, 20, 30])
print(f"Array NumPy: {arr}")

# Uji pandas
df = pd.DataFrame({'X': [5, 6], 'Y': [7, 8]})
print(f"\nDataFrame pandas:\n{df}")

# Uji Matplotlib
plt.plot([1, 2, 3], [3, 2, 1])
plt.title("Uji Matplotlib")
plt.show()
```

Output yang Diharapkan

Jika instalasi sudah benar, maka akan muncul tampilan *array* dari *NumPy*, tabel *DataFrame* dari *pandas*, serta sebuah grafik sederhana dari *Matplotlib*.



Gambar 4. 5 Luaran Program Pengujian Instalasi Phyton
(Sumber: (Trix Cyrus, 2024))

4.3.2 Dasar-Dasar Sintaks Python

Sebelum mempelajari lebih jauh tentang AI dan ML, pengguna perlu memahami terlebih dahulu dasar-dasar sintaks Python sebagai fondasi utama dalam pemrograman. Beberapa konsep penting yang perlu dikuasai meliputi hal-hal berikut (Educate360, 2025).

1. *Variabel* dan Tipe Data

Python menyediakan mekanisme untuk menyimpan data dalam bentuk variabel. Bahasa ini mendukung berbagai tipe data seperti *integer*, *float*, *string*, *list*, *tuple*, *dictionary*, dan lainnya.

Pemahaman terhadap tipe data sangat penting agar data dapat diolah dengan tepat sesuai kebutuhan program.

2. Control Flow

Python memiliki pernyataan pengendali seperti *if–else*, *for*, dan *while* yang memungkinkan program mengambil keputusan dan mengatur jalannya eksekusi. Dengan struktur ini, program dapat merespons kondisi tertentu dan melakukan pengulangan sesuai logika yang ditentukan.

3. Function

Function merupakan sekumpulan perintah yang dirancang untuk menjalankan tugas tertentu dan dapat digunakan berulang kali. Dengan penggunaan *function*, kode program menjadi lebih terstruktur, mudah diatur, dan lebih efisien karena bagia/proses yang sama tidak perlu ditulis berulang.

4. File Handling

Python menyediakan fasilitas yang sederhana untuk membaca dan menulis data ke dalam berkas. Kemampuan ini sangat penting dalam AI dan ML, misalnya untuk memproses dataset atau menyimpan hasil pelatihan model.

4.3.4 *Library Phyton* untuk Aplikasi AI dan ML

Popularitas Python dalam bidang AI dan ML tidak terlepas dari ketersediaan *library* yang sangat lengkap. Beberapa *library* utama yang sering digunakan dijelaskan sebagai berikut (Educate360, 2025).

1. *NumPy*

NumPy merupakan *library* dasar untuk komputasi numerik yang efisien. Dalam AI dan ML, *library* ini banyak digunakan untuk mengolah data dalam bentuk *array* dan matriks serta melakukan operasi matematika seperti aljabar linear dan pengolahan sinyal. Dalam bidang pengolahan citra, misalnya, gambar direpresentasikan sebagai *array NumPy* sehingga dapat dimanipulasi, difilter, atau ditransformasikan dengan cepat.

2. Pandas

Pandas dirancang untuk memudahkan pengolahan dan analisis data terstruktur. Dengan struktur data seperti *DataFrame*, pengguna dapat melakukan pembersihan data, penyaringan, pengelompokan, dan analisis statistik secara efisien. Dalam dunia keuangan, *Pandas* sering digunakan untuk menganalisis transaksi pelanggan guna menemukan pola atau mendeteksi aktivitas yang mencurigakan.

3. *Matplotlib*

Matplotlib merupakan *library* visualisasi yang digunakan untuk menampilkan data dalam bentuk grafik dan diagram. Dalam AI dan ML, visualisasi sangat penting untuk memahami pola dan tren dalam data. Misalnya, pada analisis sentimen media sosial, hasil pengolahan teks dapat divisualisasikan dalam bentuk grafik untuk menunjukkan perubahan opini publik dari waktu ke waktu.

4. Scikit-Learn

Scikit-Learn menyediakan berbagai algoritma *Machine Learning*, teknik *data preprocessing*, serta metode evaluasi model. *Library* ini memudahkan

proses pelatihan, pengujian, dan penerapan model ML. Salah satu penerapannya adalah pada sistem penyaring spam, di mana algoritma seperti *Naive Bayes* atau *Support Vector Machine* digunakan untuk membedakan email yang resmi dan yang mengandung spam.

DAFTAR PUSTAKA

Aurélien Géron (2019) *Hands-on Machine Learning with Scikit-Learn, Keras, and Tensor Flow (Concepts, Tools, and Techniques to Build Intelligent Systems)*. Second Edition. O'Reilly Media, Inc.

BeginnersGuide/Overview - Python Wiki (2024). Available at: https://wiki.python.org/moin/BeginnersGuide/Overview (Accessed: 7 January 2026).

Blue Coding (2024) *Why Python Is Still the Best Option for Machine Learning and AI*. Available at: https://www.bluecoding.com/post/why-python-remains-the-top-choice-for-ai-and-machine-learning (Accessed: 8 January 2026).

Kamlesh Singad (2024) *Python for AI and Machine Learning: The Complete Guide*. Available at: https://kamleshsingad.com/python-for-ai-and-machine-learning/ (Accessed: 7 January 2026).

Deitel, P.J.. and Deitel, H.M.. (2017) *Java 9 for programmers*. 4th ed. Prentice Hall.

GeeksforGeeks (2025) *Machine Learning with Python Tutorial*. Available at: https://www.geeksforgeeks.org/machine-learning/machine-learning-with-python/ (Accessed: 12 January 2026).

Gupta, P. and Bagchi, A. (2024) *Synthesis Lectures on Engineering, Science, and Technology Essentials of Python for Artificial Intelligence and Machine Learning*.

Jayesh Totla (2025) *Python: The AI & ML Powerhouse*. Available at: https://synergytop.com/blog/why-python-is-the-most-adaptable-technology-for-

artificial-intelligence-and-machine-learning/ (Accessed: 8 January 2026).

K. Thiyagu (no date) *Artificial Intelligence (AI) in Education.pdf*. Available at: https://www.slideshare.net/slideshow/ai-in-educationpdf/258907475 (Accessed: 8 January 2026).

Klein, B. (2021) *Python Tutorial*. Available at: www.python-course.eu.

Mark Lutz (2009) *Learning Python*. O'Reilly Media, Inc.

MathWorks (2025) *MATLAB Documentation*. Available at: https://www.mathworks.com/help/matlab/index.html (Accessed: 12 January 2026).

Poole, D.L. and Mackworth, A.K. (2025) *Python code for Artificial Intelligence Foundations of Computational Agents*. Available at: https://aipython.org (Accessed: 13 January 2026).

Pustaka Python dalam Data Science, Machine Learning, dan Generative AI (2025). Available at: https://buymeacoffee.com/pembelajarseumurhidup/pustaka-python-dalam-data-science-machine-learning-dan-generative-ai (Accessed: 8 January 2026).

Python Software Foundation (2025) *Status of Python versions*. Available at: https://devguide.python.org/versions/ (Accessed: 12 January 2026).

Rick van Hattem (2022) *Mastering_Python*. Packt Publishing.

Trix Cyrus (2024) *Part 2: Building Your Own AI - Setting Up the Environment for AI/ML Development - DEV Community*. Available at: https://dev.to/trixsec/part-1-building-your-own-ai-setting-up-the-environment-for-aiml-development-315g (Accessed: 13 January 2026).

Educate360 (2025) *Ultimate Guide to Python for AI & ML*.

Venables, W.N., Smith, D.M. and R Core Team (2025) *An Introduction to R Notes on R: A Programming Environment for Data Analysis and Graphics*.

Yamuna, M. (2025) *Python for Artificial Intelligence and Machine Learning the Creative Commons Attribution License (CC BY 4.0)*, *International Journal of Trend in Scientific Research and Development (IJTSRD)*. Available at: www.ijtsrd.com/papers/ijtsrd79847.pdf.

Yuli Vasiliev (2020) *Natural Language Processing with Python and spaCy: A Practical Introduction*. No Starch Press.

Zinovyev, A. (2021) *TP 'Python for AI': a crash course focused on linear regression models*. Available at: https://auranic.github.io/teaching/2021-python_for_ai.

BAB 5

STRUKTUR DATA DAN *LIBRARY PYTHON (NUMPY, PANDAS, MATPLOTLIB)*

Python telah menjadi bahasa pemrograman utama dalam pengembangan komputasi ilmiah modern, khususnya pada bidang Kecerdasan Buatan (*Artificial Intelligence*), *Machine Learning*, dan *Data Science*. Popularitas Python tidak semata-mata disebabkan oleh kemudahan sintaksnya, tetapi juga oleh kekuatan ekosistem library yang memungkinkan pengolahan data dilakukan secara efisien, terstruktur, dan terintegrasi. Dalam praktik komputasi modern, Python tidak hanya berfungsi sebagai bahasa pemrograman, tetapi juga sebagai *platform analisis data* yang komprehensif.

Dalam konteks AI dan Machine Learning, data merupakan komponen sentral yang menentukan kualitas dan keberhasilan sistem. Data yang besar, beragam, dan kompleks membutuhkan struktur penyimpanan dan pengolahan yang tepat agar dapat dimanfaatkan secara optimal. Kesalahan dalam pengelolaan data pada tahap awal akan berdampak langsung pada performa analisis dan akurasi model di tahap selanjutnya. Oleh karena itu, pemahaman mendalam terhadap struktur data Python dan library pendukungnya menjadi prasyarat fundamental sebelum mempelajari algoritma pembelajaran mesin.

Bab ini secara khusus membahas struktur data dasar Python serta tiga library inti yang menjadi fondasi dalam hampir seluruh workflow analisis data, yaitu **NumPy**, **Pandas**, dan **Matplotlib**. Ketiga library ini membentuk satu kesatuan ekosistem yang memungkinkan data direpresentasikan secara numerik, dikelola secara tabular, dan divisualisasikan secara informatif dalam satu lingkungan kerja terpadu.

5.1 Konsep Struktur Data dalam Python

Struktur data dapat didefinisikan sebagai cara sistematis untuk menyimpan, mengatur, dan mengelola data agar dapat diakses serta dimodifikasi secara efisien. Dalam pemrograman Python, struktur data memiliki peran strategis karena Python dirancang sebagai bahasa tingkat tinggi yang mengutamakan fleksibilitas dan keterbacaan kode.

Pemilihan struktur data yang tepat akan memengaruhi:

1. Efisiensi penggunaan memori
2. Kecepatan pemrosesan data
3. Kemudahan manipulasi data
4. Skalabilitas sistem komputasi

Dalam pengembangan AI, struktur data Python sering digunakan sebagai wadah awal data mentah sebelum data tersebut dikonversi ke bentuk numerik yang lebih kompleks melalui library seperti NumPy dan Pandas. Oleh karena itu, pemahaman terhadap struktur data dasar Python menjadi fondasi awal yang tidak dapat diabaikan.

5.2 Struktur Data Dasar Python

5.2.1 List

List merupakan struktur data paling umum dan fleksibel dalam Python. List bersifat mutable, yang berarti elemen di dalamnya dapat diubah setelah list dideklarasikan.

```
data = [80, 85, 90, 95]
```

List mampu menyimpan berbagai tipe data dalam satu variabel dan mendukung operasi penambahan, penghapusan, serta pengubahan elemen. Dalam praktik analisis data, list sering digunakan untuk menyimpan data mentah, hasil pembacaan file, atau kumpulan nilai sementara sebelum diproses lebih lanjut.

Namun, dari sudut pandang komputasi numerik, list memiliki keterbatasan karena setiap elemen disimpan sebagai objek Python yang terpisah. Hal ini menyebabkan operasi matematis berskala besar menjadi kurang efisien dibandingkan dengan array NumPy.

5.3.2 Tuple

Tuple memiliki kemiripan dengan list, tetapi bersifat **immutable**, sehingga nilainya tidak dapat diubah setelah dibuat.

```
parameter = (0.01, 100, "relu")
```

Tuple sering digunakan untuk menyimpan data yang bersifat tetap, seperti konfigurasi sistem, parameter model, atau konstanta. Sifat immutable pada tuple memberikan keuntungan dalam menjaga konsistensi data dan mencegah perubahan yang tidak disengaja dalam sistem komputasi.

5.3.3 *Dictionary*

Dictionary menyimpan data dalam bentuk pasangan *key–value* dan memungkinkan akses data yang sangat cepat berdasarkan kunci tertentu.

```
model_config = {
    "learning_rate": 0.01,
    "epoch": 100,
    "optimizer": "adam"
}
```

Dalam konteks AI dan analisis data, dictionary sangat berguna untuk:

- Menyimpan parameter eksperimen
- Menyimpan metadata dataset
- Menyimpan hasil evaluasi

Dictionary memungkinkan pengelolaan data yang lebih semantik dibandingkan list atau tuple karena setiap nilai memiliki makna yang jelas melalui kuncinya.

5.3.4 Set

Set merupakan struktur data yang hanya menyimpan elemen unik tanpa urutan tertentu.

```
label_unik = {"spam", "ham"}
```

Set sering digunakan dalam tahap *data cleaning* untuk menghilangkan duplikasi nilai dan memastikan keunikan elemen dalam suatu kumpulan data.

5.4 *Library NumPy*

5.4.1 Pengantar *NumPy*

NumPy (*Numerical Python*) merupakan library fundamental untuk komputasi numerik di Python. NumPy memperkenalkan struktur data utama berupa ndarray (N-dimensional array) yang dirancang untuk efisiensi komputasi dan penggunaan memori. (Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, A, 2020) menegaskan bahwa NumPy menjadi tulang punggung ekosistem komputasi ilmiah Python karena kemampuannya dalam menangani operasi numerik berskala besar secara efisien.

5.4.2 Array NumPy

```
import numpy as np
data = np.array([1, 2, 3, 4, 5])
```

Array NumPy bersifat homogen, artinya seluruh elemen memiliki tipe data yang sama. Data disimpan secara kontigu di memori, sehingga memungkinkan operasi matematika dilakukan dengan sangat cepat.

5.4.3 Operasi Vektorisasi

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
c = a + b
```

Operasi vektorisasi menghilangkan kebutuhan penggunaan perulangan eksplisit (*loop*), sehingga meningkatkan performa komputasi secara signifikan. Pendekatan ini sangat penting dalam analisis data dan pengolahan numerik berskala besar.

5.4.4 Array Multidimensi

```
matriks = np.array([[1, 2], [3, 4]])
```

Array multidimensi digunakan untuk merepresentasikan data tabular, matriks, hingga citra digital. Dalam konteks komputasi modern, struktur ini menjadi dasar bagi berbagai aplikasi ilmiah dan rekayasa.

5.4.5 Fungsi Statistik NumPy

NumPy menyediakan berbagai fungsi statistik bawaan, seperti rata-rata, simpangan baku, nilai minimum, dan maksimum.

```
np.mean(data)
np.std(data)
np.min(data)
np.max(data)
```

Fungsi-fungsi ini digunakan secara luas dalam analisis eksploratif dan normalisasi data.

5.5 *Library* Pandas

5.5.1 Konsep Dasar Pandas

Pandas merupakan library yang dirancang untuk manipulasi dan analisis data berbasis tabel. Pandas dibangun di atas NumPy dan menyediakan struktur data yang lebih intuitif, yaitu Series dan DataFrame. Menurut (McKinney, 2022), Pandas telah menjadi standar utama dalam pengolahan data tabular di Python.

5.5.2 Series

```
import pandas as pd
s = pd.Series([10, 20, 30])
```

Series merupakan struktur satu dimensi yang memiliki indeks dan sering digunakan untuk merepresentasikan data runtun waktu (*time series*).

5.5.3 DataFrame

```
data = {
 "Nama": ["Rudi", "Budi", "Cindi"],
"Nilai": [85, 90, 88]
}
df = pd.DataFrame(data)
```

DataFrame merupakan struktur data dua dimensi yang menyerupai tabel dan menjadi elemen paling dominan dalam analisis data modern.

5.5.4 Operasi Analisis Data

```
df.head()
df.describe()
df["Nilai"].mean()
```

Operasi ini digunakan dalam *Exploratory Data Analysis (EDA)* untuk memahami karakteristik dataset sebelum dilakukan analisis lebih lanjut.

5.5.5 Pembersihan Data

```
df.dropna()
df.fillna(0)
```

Pembersihan data merupakan tahap krusial karena kualitas data akan sangat menentukan kualitas hasil analisis dan pemodelan.

5.6 *Library* Matplotlib

5.6.1 Konsep dan Peran Matplotlib dalam Visualisasi Data

Matplotlib merupakan library visualisasi data berbasis Python yang dirancang untuk menghasilkan grafik dua dimensi secara fleksibel dan presisi. Library ini dikembangkan untuk mendukung kebutuhan visualisasi dalam komputasi ilmiah, analisis data, dan rekayasa perangkat lunak. Dalam ekosistem Python, Matplotlib berfungsi sebagai jembatan antara data numerik dan interpretasi visual yang dapat dipahami manusia.

Visualisasi data memiliki peran krusial dalam proses analisis karena membantu peneliti dan pengembang memahami pola, tren, dan anomali yang sulit diidentifikasi melalui representasi numerik semata. Grafik yang disusun dengan baik dapat mempercepat proses pengambilan keputusan, meningkatkan kualitas analisis, serta mendukung komunikasi hasil analisis kepada pihak non-teknis.

Dalam konteks *data science*, Matplotlib sering digunakan bersamaan dengan NumPy dan Pandas. Data yang telah diproses secara numerik atau tabular dapat langsung divisualisasikan tanpa perlu konversi format tambahan. Integrasi ini menjadikan Matplotlib sebagai komponen fundamental dalam workflow analisis data berbasis Python.

5.6.2 Matplotlib sebagai Bagian dari Ekosistem Library *Data Science Python*

Ekosistem Python untuk data science terdiri dari sejumlah library yang saling terintegrasi, di mana Matplotlib menempati posisi sentral sebagai alat visualisasi. Studi (João Eduardo Montandon, Luciana Lourdes Silva, Cristiano Politowski, Daniel Prates, Arthur de Brito Bonifácio, 2025) menegaskan bahwa Matplotlib termasuk dalam tiga library utama yang paling banyak digunakan dalam pipeline data science modern, bersama dengan NumPy dan Pandas

Menurut temuan tersebut, visualisasi data tidak hanya berfungsi sebagai tahap akhir analisis, tetapi juga sebagai alat bantu eksploratif yang digunakan sepanjang siklus pengolahan data. Praktik ini umum dilakukan dalam *interactive computing environments* seperti Jupyter Notebook, di mana visualisasi dapat dihasilkan secara langsung dari variabel Python.

Namun demikian, ketergantungan pada default parameter dalam library visualisasi juga memiliki implikasi teknis. Perubahan perilaku default pada fungsi visualisasi, termasuk pada Matplotlib, berpotensi memengaruhi konsistensi hasil visualisasi antar versi library. Oleh karena itu, pemahaman mendalam terhadap parameter visualisasi menjadi aspek penting dalam menjaga reprodusibilitas analisis.

5.6.3 Arsitektur Dasar Matplotlib

Secara konseptual, Matplotlib terdiri atas beberapa komponen utama, yaitu:

1. ***Figure*** – kanvas utama tempat seluruh elemen visual digambar
2. ***Axes*** – area tempat grafik diplot
3. ***Axis*** – sumbu koordinat horizontal dan vertikal
4. ***Plot Elements*** – garis, titik, batang, teks, dan anotasi

Pendekatan berbasis objek (*object-oriented approach*) yang digunakan Matplotlib memungkinkan pengguna mengontrol setiap aspek visualisasi secara rinci. Hal ini menjadikan Matplotlib sangat fleksibel, meskipun memerlukan pemahaman konsep yang lebih mendalam dibandingkan library visualisasi tingkat tinggi.

5.6.4 Visualisasi Data Dasar dengan Matplotlib

a. Grafik Garis (*Line Plot*)

Grafik garis digunakan untuk merepresentasikan perubahan nilai data secara berurutan, misalnya terhadap waktu atau iterasi.

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

plt.plot(x, y)
plt.xlabel("Iterasi")
plt.ylabel("Nilai")
plt.title("Grafik Garis Sederhana")
plt.show()
```

Grafik garis sangat umum digunakan untuk memantau tren data, seperti pertumbuhan nilai, penurunan kesalahan, atau perubahan variabel pengamatan.

b. Scatter Plot

Scatter plot digunakan untuk menganalisis hubungan antara dua variabel numerik.

```
plt.scatter(x, y)
plt.xlabel("Variabel X")
plt.ylabel("Variabel Y")
plt.title("Scatter Plot")
plt.show()
```

Visualisasi ini membantu mengidentifikasi pola korelasi, klaster, atau pencilan (*outliers*) dalam dataset.

c. Histogram

Histogram digunakan untuk menampilkan distribusi frekuensi suatu variabel.

```
data = [10, 20, 20, 30, 30, 30, 40]
plt.hist(data, bins=4)
plt.title("Histogram Distribusi Data")
plt.show()
```

Histogram sangat penting dalam analisis awal data untuk memahami sebaran nilai dan kecenderungan statistik.

5.6.5 Kustomisasi dan Interpretabilitas Visualisasi

Salah satu keunggulan utama Matplotlib adalah kemampuannya dalam kustomisasi visualisasi. Pengguna dapat mengatur warna, label, skala, anotasi, dan gaya grafik secara detail. Kustomisasi ini bukan sekadar estetika, tetapi berperan penting dalam meningkatkan interpretabilitas visual.

Dalam pengembangan perangkat lunak berbasis data, visualisasi yang konsisten dan terkontrol membantu menjaga keseragaman hasil analisis antar eksperimen dan antar versi library. (João Eduardo Montandon, Luciana Lourdes Silva, Cristiano Politowski, Daniel Prates, Arthur de Brito Bonifácio, 2025) menekankan bahwa perubahan kecil pada parameter

default library visualisasi dapat berdampak pada interpretasi hasil, sehingga eksplisitasi parameter visualisasi menjadi praktik yang disarankan dalam pengembangan perangkat lunak data science

5.6.6 Matplotlib dan Reprodusibilitas Analisis

Reprodusibilitas merupakan prinsip penting dalam komputasi ilmiah. Visualisasi yang dihasilkan menggunakan Matplotlib harus dapat direproduksi secara konsisten pada lingkungan dan versi library yang berbeda. Oleh karena itu, praktik terbaik yang disarankan meliputi:

1. Penulisan parameter visualisasi secara eksplisit
2. Penguncian versi library
3. Dokumentasi kode visualisasi

Dengan pendekatan ini, Matplotlib tidak hanya berfungsi sebagai alat visualisasi, tetapi juga sebagai bagian integral dari sistem analisis yang dapat dipertanggungjawabkan secara ilmiah.

5.7 Integrasi NumPy, Pandas, dan Matplotlib

Ketiga library ini dirancang untuk saling terintegrasi dan digunakan secara bersamaan dalam satu workflow analisis data. Data dapat diproses menggunakan Pandas, dikonversi ke NumPy untuk komputasi numerik, dan divisualisasikan menggunakan Matplotlib.

Pendekatan terintegrasi ini secara signifikan meningkatkan efisiensi kerja dan mengurangi ketergantungan pada proses ekspor–impor data eksternal. Studi terbaru menunjukkan bahwa integrasi langsung library Python dalam satu lingkungan kerja mampu meningkatkan efisiensi komputasi dan manajemen data, khususnya dalam

analisis berskala besar (Orlando Arroyo, Dirsa Feliciano, Dirsa Feliciano, 2024)

5.8 Peran *Library NumPy*, Pandas, dan Matplotlib dalam *Workflow* Ilmiah Modern

Penggunaan Python dalam komputasi ilmiah dan rekayasa modern mengalami pergeseran paradigma signifikan seiring berkembangnya library pendukung yang terintegrasi. Praktik lama yang mengandalkan ekspor hasil perhitungan ke berkas teks untuk diproses secara terpisah kini mulai ditinggalkan karena dinilai tidak efisien dan rentan kesalahan.(Orlando Arroyo, Dirsa Feliciano, Dirsa Feliciano, 2024) menjelaskan bahwa integrasi langsung antara Python dan library ilmiah seperti NumPy, Pandas, dan Matplotlib memungkinkan seluruh proses analisis mulai dari pemodelan, pengolahan data, hingga visualisasi dilakukan dalam satu lingkungan kerja terpadu.

Pendekatan ini secara signifikan mengurangi kebutuhan *input-output* berkas eksternal yang sebelumnya menjadi bottleneck dalam pengolahan data berskala besar.

NumPy berperan sebagai fondasi komputasi numerik dengan menyediakan struktur array multidimensi yang efisien dan mendukung operasi vektorisasi. Pandas memperluas kemampuan tersebut dengan menyediakan struktur DataFrame yang memudahkan manipulasi data tabular, termasuk penyaringan, agregasi, dan pembersihan data. Sementara itu, Matplotlib memungkinkan visualisasi langsung dari variabel Python tanpa memerlukan konversi data ke format lain.

Dalam konteks Kecerdasan Buatan dan Machine Learning, workflow terpadu ini memungkinkan peneliti dan pengembang untuk:

1. Memproses data dalam jumlah besar secara efisien
2. Melakukan eksplorasi data secara interaktif
3. Menghasilkan visualisasi analitis yang informatif
4. Meminimalkan kesalahan akibat pemindahan data antar sistem

Dengan demikian, struktur data Python dan library pendukungnya tidak hanya berfungsi sebagai alat bantu pemrograman, tetapi telah menjadi infrastruktur utama dalam ekosistem komputasi ilmiah modern.

Pembahasan mengenai workflow terpadu ini menjadi landasan konseptual untuk memahami penerapan nyata library Python dalam berbagai perangkat lunak ilmiah berskala besar yang akan dibahas pada subbab berikutnya.

5.9 Peran NumPy, Pandas, dan Matplotlib dalam Perangkat Lunak Ilmiah Modern

Perkembangan perangkat lunak ilmiah modern menunjukkan pergeseran signifikan dari sistem komputasi berbasis bahasa proprietary menuju ekosistem terbuka berbasis Python. Salah satu faktor utama yang mendorong pergeseran ini adalah kematangan dan integrasi library Python seperti NumPy, Pandas, dan Matplotlib yang mampu mendukung komputasi numerik, manajemen data, serta visualisasi secara terpadu dalam satu lingkungan kerja.

Peran struktur data dan library Python dalam komputasi ilmiah modern tidak hanya bersifat konseptual, tetapi telah terbukti secara nyata dalam pengembangan perangkat lunak ilmiah berskala besar. Studi (Janko Slavič, Klemen Zaletelj, Domen Gorjup, Dag Pasquale Pasca, Angelo Aloisio, Knut Andreas Kvåle, Gunnstein Thomas Frøseth, Wout Weijtjens, Ahmed Mujtaba, Martin Česnik, Aleš Zorman, George Tsialiamanis, Ivan Tomac, Thiago G. Ritto, Domen Ocepek,

Dan R, 2025) menunjukkan bahwa ekosistem Python, dengan NumPy sebagai fondasi komputasi numerik dan Matplotlib sebagai alat visualisasi ilmiah, telah menjadi tulang punggung dalam penelitian sinyal, dinamika struktur, dan pemodelan ilmiah terbuka. Temuan ini menegaskan bahwa Python scientific stack telah berfungsi sebagai infrastruktur utama dalam berbagai domain komputasi ilmiah lintas disiplin.

Selain digunakan dalam pengembangan perangkat lunak ilmiah berbasis simulasi dan pemodelan, ekosistem Python juga telah menjadi fondasi utama dalam sistem yang mengintegrasikan struktur data, komputasi numerik, visualisasi, dan Machine Learning secara terpadu. Beniwal dan Ray melalui pengembangan MAPAL (Mapping Alloys**)** menunjukkan bahwa library Python seperti NumPy, Pandas, dan Matplotlib digunakan sebagai komponen inti dalam pemetaan fitur material, pengelolaan data komposisi, serta visualisasi hasil prediksi model pembelajaran mesin pada ruang komposisi yang kompleks.

Dalam arsitektur MAPAL, struktur data berbasis array NumPy dimanfaatkan untuk memastikan efisiensi komputasi numerik, sementara Pandas digunakan untuk merepresentasikan dan memanipulasi data komposisi dan fitur material secara terstruktur. Matplotlib berperan dalam menyajikan hasil pemetaan fitur dan prediksi Machine Learning dalam bentuk visualisasi ilmiah yang mendukung interpretasi pola dan hubungan antar variabel. Pendekatan ini memperlihatkan bahwa integrasi struktur data Python dan library ilmiah tidak hanya relevan untuk pembelajaran, tetapi juga menjadi infrastruktur utama dalam pengembangan sistem berbasis kecerdasan buatan pada domain ilmiah yang kompleks.

Contoh penerapan Python scientific stack pada perangkat lunak ilmiah berskala besar juga ditunjukkan melalui pengembangan PySTRA. (Colin Caprani, Mohammad Shihabuddin Khan, 2025) menunjukkan bahwa perangkat

lunak ilmiah dapat dibangun sepenuhnya di atas Python, dengan NumPy sebagai mesin komputasi numerik, Pandas sebagai pengelola data kompleks, dan Matplotlib sebagai alat visualisasi hasil analisis. Dalam arsitektur PySTRA, ketiga library tersebut ditempatkan sebagai *Python Core* yang menjadi fondasi seluruh modul komputasi dan analisis.

Berdasarkan uraian arsitektur PySTRA, sebagian besar algoritma komputasi numerik dibangun menggunakan array NumPy untuk menjamin efisiensi perhitungan. Pandas digunakan secara intensif pada modul lanjutan untuk penyimpanan, manipulasi, dan penyajian data hasil analisis, sedangkan Matplotlib dimanfaatkan untuk visualisasi hasil komputasi agar dapat diinterpretasikan oleh pengguna secara intuitif. Lebih lanjut, arsitektur berlapis PySTRA menunjukkan bagaimana integrasi NumPy, Pandas, dan Matplotlib memungkinkan pengembangan perangkat lunak ilmiah yang modular, extensible, dan mudah dipelihara.

Penggunaan library Python sebagai fondasi komputasi ilmiah juga ditunjukkan dalam bidang kimia analitik dan chemometrics. (Luiz Renato Rosa Leme de Souza, Carlos Alberto Rios, 2026) mengembangkan rutinitas berbasis Python untuk preprocessing data Raman mapping dan penerapan metode multivariat seperti CLS, PCA, dan PLS. Dalam rutinitas tersebut, NumPy digunakan untuk pengelolaan array berdimensi tinggi, Pandas untuk manipulasi data spektral, dan Matplotlib untuk visualisasi spektrum serta peta kimia. Hasil ini menunjukkan bahwa Python scientific stack telah berkembang menjadi solusi terbuka yang setara dengan perangkat lunak proprietary dalam analisis data ilmiah yang kompleks.

Penggunaan struktur data Python dan library ilmiah juga terlihat pada pengembangan TensorClus**,** sebuah library Python untuk tensor (co)-clustering. Dalam TensorClus, NumPy dan Pandas digunakan sebagai fondasi utama dalam pemuatan dan representasi data tensor berdimensi tinggi,

sementara Matplotlib dimanfaatkan untuk visualisasi evolusi log-likelihood, parameter model, serta hasil reorganisasi data. Integrasi ini menunjukkan bahwa library Python dasar berperan langsung dalam pengembangan algoritma pembelajaran tak terawasi dan analisis data kompleks berskala besar.

Perkembangan terbaru semakin menegaskan peran strategis struktur data dan library Python dasar dalam menjembatani data terstruktur dengan model Deep Learning. Melalui pengembangan TINTOlib, data tabular yang dikelola menggunakan Pandas dan NumPy ditransformasikan menjadi representasi citra sintetis, sementara Matplotlib digunakan untuk visualisasi dan inspeksi hasil transformasi. Pendekatan ini memungkinkan penerapan Convolutional Neural Networks dan Vision Transformers pada data non-visual, mempertegas bahwa ekosistem Python menyediakan fondasi menyeluruh bagi pipeline Machine Learning modern.

Dengan demikian, berbagai studi kasus lintas domain tersebut memperkuat posisi NumPy, Pandas, dan Matplotlib sebagai fondasi teknis universal yang tidak hanya relevan untuk pembelajaran, tetapi juga untuk implementasi nyata pada perangkat lunak ilmiah dan sistem berbasis Kecerdasan Buatan.

DAFTAR PUSTAKA

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, A, C.G.& T.E.O. (2020) 'Array programming with NumPy', nature, 585, pp. 357–362. doi:10.1038/s41586-020-2649-2.

Colin Caprani, Mohammad Shihabuddin Khan, J.H. (2025) 'PySTRA: Python structural reliability analysis', Elsevier Science Inc. [Preprint]. doi:10.1016/j.softx.2025.102047.

Janko Slavič, Klemen Zaletelj, Domen Gorjup, Dag Pasquale Pasca, Angelo Aloisio, Knut Andreas Kvåle, Gunnstein Thomas Frøseth, Wout Weijtjens, Ahmed Mujtaba, Martin Česnik, Aleš Zorman, George Tsialiamanis, Ivan Tomac, Thiago G. Ritto, Domen Ocepek, Dan R, M.B. (2025) 'Python open-source signal processing and modeling/simulation for scientific research in structural dynamics', Elsevier Science Inc. [Preprint]. doi:10.1016/j.ymssp.2025.113465.

João Eduardo Montandon, Luciana Lourdes Silva, Cristiano Politowski, Daniel Prates, Arthur de Brito Bonifácio, G.E.B.I.& C. (2025) 'Unboxing Default Argument Breaking Changes in 1 + 2 data science libraries', Elsevier Science Inc., 229(C). doi:10.1016/j.jss.2025.112460.

Luiz Renato Rosa Leme de Souza, Carlos Alberto Rios, C.A.R. (2026) 'Raman mapping and Chemometrics: An open access Python-based routine to preprocess and generate chemical maps applying CLS, PCA and PLS

methods', Elsevier Science Inc. [Preprint]. doi:10.1016/j.chemolab.2025.105616.

McKinney, W. (2022) Python for Data Analysis, 3rd Edition. 3rd edn. Edited by S.S. Jessica Haberman, Angela Rufino, Christopher Faucher. O'Reilly Media. Available at: https://www.oreilly.com/catalog/errata.csp?isbn=063692051982.

Orlando Arroyo, Dirsa Feliciano, Dirsa Feliciano, J.V. (2024) 'opseestools: A Python library to streamline OpenSeesPy workflows', Elsevier Science Inc., 30. doi:10.1016/j.softx.2024.101832.

BAB 6

PEMROSESAN DATA: PEMBERSIHAN, TRANSFORMASI, DAN VISUALISASI

Gambar 6.1 berikut ini merepresentasikan proses konseptual dalam transformasi data mentah menjadi informasi yang terstruktur dan siap analisis. Elemen visual utama berupa ikon lembar kertas yang dicentang mengisyaratkan penyelesaian suatu tahap kritis dalam alur pemrosesan data, yakni validasi dan penyempurnaan data setelah melalui serangkaian proses kurasi. Di sekelilingnya, term-term seperti "Suhu Udara", "Derajat Celcius", "Berat Badan", "Kilogram", dan "Lembar" berfungsi sebagai contoh konkret atribut data yang umum ditemui dalam konteks empiris, sekaligus mewakili beragam tipe data, baik kuantitatif maupun kategorikal.

Gambar 6. 1 Abstraksi Data

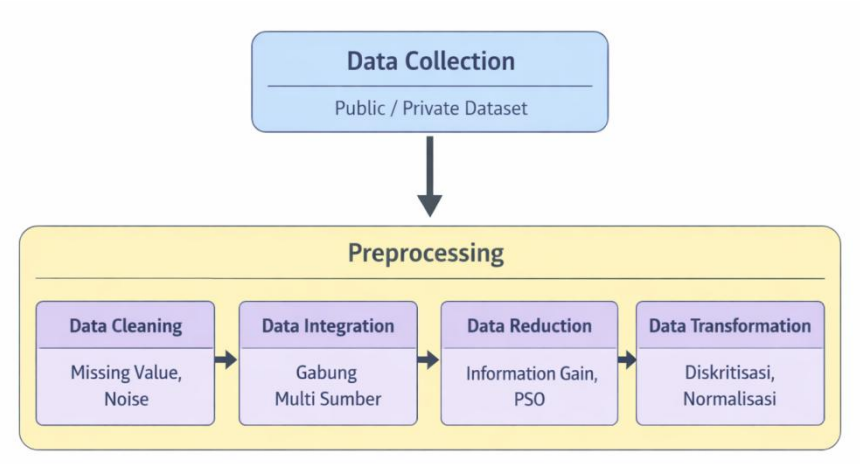
(Sumber: dicoding.com)

Secara implisit, gambar 6.1 mengilustrasikan tiga fase fundamental yang dibahas dalam bab ini. Pembersihan data tercermin dari simbol centang yang menandakan bahwa data telah dikoreksi dari anomali, inkonsistensi, atau missing value. Transformasi data terlihat dari hubungan antar-konsep seperti "Suhu Udara" dengan "Derajat Celcius", yang mengisyaratkan proses standarisasi satuan, normalisasi, atau pengubahan format. Sementara itu, penyusunan elemen-elemen tersebut dalam tata letak yang teratur mengarah pada fase visualisasi data, di mana informasi yang telah diproses disajikan dalam representasi visual yang sistematis untuk memfasilitasi interpretasi.

6.2 Alur Proses Data

Gambar berikut menggambarkan kerangka konseptual alur pengolahan data dalam konteks data mining yang diawali dengan tahap pengumpulan data (*data collection*) dari berbagai sumber, baik dataset publik maupun privat. Data yang diperoleh pada tahap ini umumnya masih bersifat mentah, heterogen, dan mengandung berbagai permasalahan kualitas, sehingga belum dapat digunakan secara langsung dalam proses analisis lanjutan.

Oleh karena itu, tahap prapemrosesan data (*preprocessing*) menjadi fase yang sangat krusial dalam keseluruhan proses data mining. Tahap ini mencakup pembersihan data (*data cleaning*) untuk menangani nilai hilang dan gangguan data (*noise*), integrasi data (*data integration*) untuk menggabungkan data dari berbagai sumber yang berbeda, reduksi data (*data reduction*) untuk menyederhanakan struktur data serta memilih atribut yang paling relevan, serta transformasi data (*data transformation*) yang meliputi proses diskritisasi dan normalisasi. Implementasi tahapan preprocessing secara sistematis bertujuan untuk meningkatkan kualitas, konsistensi, dan representativitas data, sehingga data yang dihasilkan lebih andal dan siap digunakan sebagai dasar dalam proses pemodelan dan analisis data mining pada tahap berikutnya.



Gambar 6. 2 Alur Pengolahan Data

Proses ini diawali dengan pengumpulan data dari berbagai sumber, baik publik maupun privat, yang sering kali masih mengandung berbagai permasalahan seperti nilai yang hilang, noise, inkonsistensi format, dan redundans(Suyanto, 2017)i. Tanpa melalui tahap prapemrosesan yang matang, kualitas analisis dan keandalan model yang dibangun dapat terancam karena data yang tidak terolah rentan terhadap bias, ketidakakuratan, dan ketidakstabilan hasil. Oleh karena itu, prapemrosesan data tidak hanya bertujuan untuk memperbaiki kualitas data, tetapi juga untuk memastikan bahwa data yang digunakan telah memenuhi asumsi dan persyaratan teknikal dari metode analisis yang akan diterapkan.

Alur kerja prapemrosesan data (*data preprocessing*) yang berfungsi sebagai tahapan awal sebelum dilakukan proses analisis dan pemodelan lebih lanjut(Firgiawan *et al.*, 2024). Alur dimulai dari tahap data collection, yaitu pengumpulan data yang bersumber dari dataset publik maupun privat. Data pada tahap ini masih bersifat mentah dan berpotensi mengandung berbagai permasalahan, seperti

nilai hilang, noise, redundansi, serta perbedaan format dan skala antar sumber data.

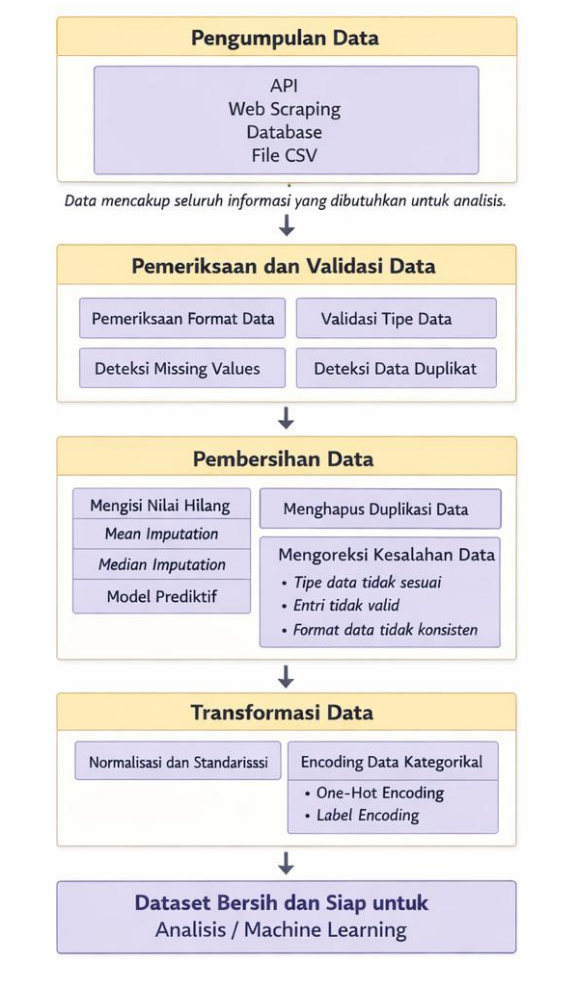
Tahap berikutnya adalah preprocessing, yang merupakan inti dari proses pengolahan data awal. Pada tahap data cleaning, dilakukan upaya untuk meningkatkan kualitas data dengan menangani nilai yang hilang (*missing values*) dan mengurangi gangguan data (*noise*) yang dapat memengaruhi akurasi analisis. Selanjutnya, data integration bertujuan untuk menggabungkan data yang berasal dari berbagai sumber menjadi satu kesatuan dataset yang konsisten, sehingga konflik data dan redundansi dapat diminimalkan.

Tahap data reduction berfokus pada penyederhanaan data tanpa menghilangkan informasi yang bersifat penting. Proses ini dilakukan melalui seleksi atribut dan reduksi dimensi dengan memanfaatkan metode seperti *Information Gain* dan *Particle Swarm Optimizatio*n (PSO), yang bertujuan untuk memilih fitur-fitur yang paling relevan terhadap tujuan analisis(Margareth, 2017). Tahap terakhir dalam preprocessing adalah data transformation, yaitu proses mengubah bentuk data agar sesuai dengan kebutuhan algoritma data mining. Transformasi ini meliputi diskritisasi untuk mengubah data kontinu menjadi kategori tertentu serta normalisasi untuk menyamakan skala data.

Pembersihan data dilakukan untuk menangani nilai hilang dan mengurangi gangguan noise. Selanjutnya, integrasi data berperan dalam menggabungkan berbagai sumber data menjadi satu kesatuan yang koheren sambil meminimalkan konflik dan duplikasi(Rahayu *et al.*, 2018). Tahap reduksi data kemudian diterapkan untuk menyederhanakan struktur data tanpa menghilangkan informasi esensial. Terakhir, transformasi data dilakukan guna mengubah data menjadi bentuk yang kompatibel dengan algoritma analisis, meliputi proses seperti diskritisasi untuk mengkategorikan data kontinu dan normalisasi untuk menyelaraskan skala antar variabel(Wahono, 2023). Dengan

melalui seluruh tahapan ini, data yang dihasilkan tidak hanya lebih berkualitas, tetapi juga lebih representatif dan siap untuk dieksplorasi, dianalisis, atau dimodelkan secara efektif.

Gambar berikut menyajikan alur sistematis proses pembersihan dan transformasi data sebagai bagian fundamental dalam tahapan prapemrosesan data sebelum dilakukan analisis lanjutan atau penerapan metode *machine learning*.



Gambar 6. 3 Proses Pembersihan Data

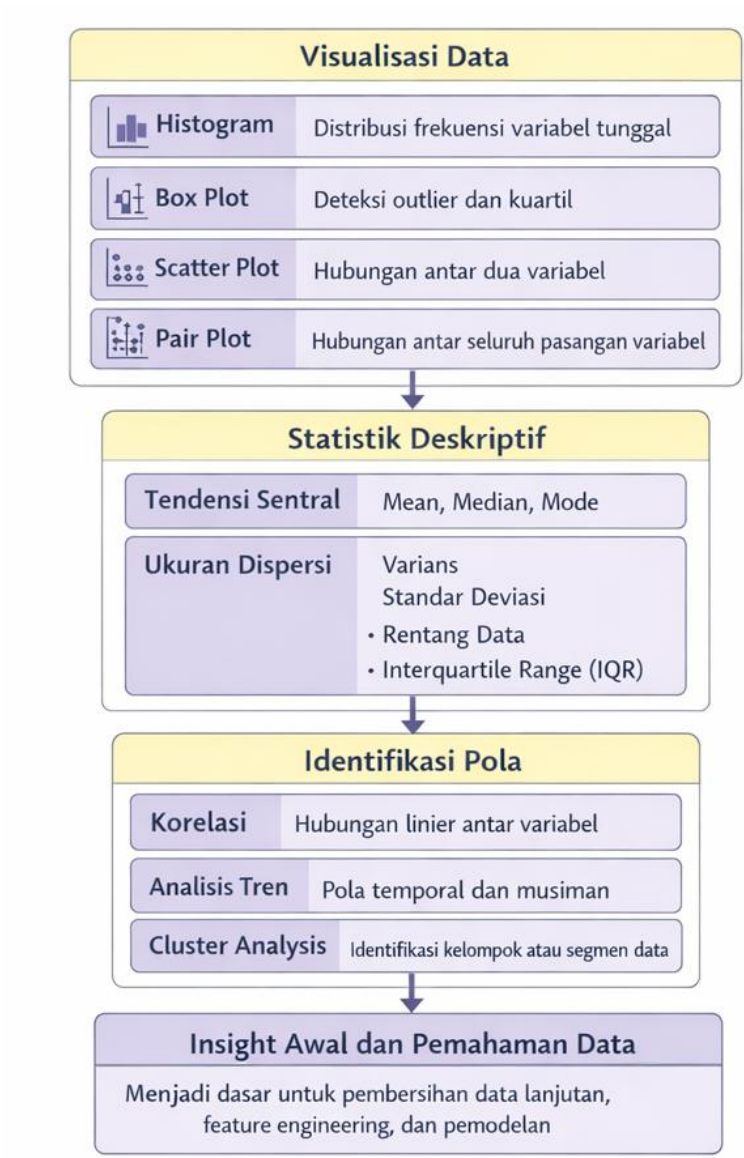
Proses diawali dengan tahap pengumpulan data dari berbagai sumber, seperti Application Programming Interface (API), *web scraping*, basis data, dan berkas CSV, dengan tujuan memastikan bahwa data yang diperoleh mencakup seluruh informasi yang dibutuhkan untuk analisis. Selanjutnya, data yang telah dikumpulkan menjalani tahap pemeriksaan dan

validasi untuk mengidentifikasi kesesuaian format dan tipe data, serta mendeteksi keberadaan nilai hilang (*missing values*) dan data duplikat (Sudipa *et al.*, 2024). Tahap pembersihan data kemudian dilakukan untuk mengatasi permasalahan yang teridentifikasi, melalui pengisian nilai hilang menggunakan pendekatan statistik maupun model prediktif, penghapusan duplikasi data, serta koreksi terhadap kesalahan dan inkonsistensi data(Sulianta, 2024). Setelah data berada dalam kondisi bersih, tahap transformasi data dilakukan untuk menyesuaikan struktur dan representasi data dengan kebutuhan analisis, yang meliputi proses normalisasi, standarisasi, dan pengkodean data kategorikal. Keseluruhan rangkaian tahapan ini bertujuan untuk menghasilkan dataset yang berkualitas tinggi, konsisten, dan siap digunakan sebagai dasar yang andal dalam proses analisis data maupun pengembangan model *machine learning*(Muttaqin, 2023).

6.3 *Exploratory Data Analysis* (EDA)

Diagram ini menguraikan tahapan sistematis dalam *Exploratory Data Analysis* (EDA) sebagai pondasi awal dalam siklus ilmu data. Tujuannya adalah untuk mengkaji karakteristik mendasar data melalui pendekatan visual dan statistik sebelum melakukan analisis inferensial atau pemodelan prediktif.

Proses diawali dengan Visualisasi Data, yang berperan dalam merepresentasikan struktur data secara grafis. Teknik visualisasi yang digunakan meliputi *histogram* untuk mengamati distribusi univariat, *box plot* untuk mendeteksi pencilan (*outliers*) dan persebaran kuartil, *scatter plot* untuk mengeksplorasi hubungan bivariat, serta *pair plot* sebagai alat eksplorasi multivariat yang memetakan seluruh relasi antar variabel secara simultan(Firgiawan *et al.*, 2024). Dapat dilihat pada gambar 6.4.



Gambar 6. 4 *Exploratory Data Analysis* (EDA)

Selanjutnya, dilakukan Analisis Statistik Deskriptif untuk mengkuantifikasi karakteristik data. Aspek ini mencakup pengukuran *tendensi sentral* seperti rerata (*mean*), median, dan modus, serta pengukuran *dispersi* seperti varians, simpangan baku (*standard deviation*), rentang (*range*), dan rentang antar-kuartil (*interquartile range/IQR*)(Wahono, 2023). Kedua pendekatan ini saling melengkapi antara representasi grafis dan deskripsi numerik.

Tahap ketiga adalah Identifikasi Pola dan Struktur Data, yang bertujuan untuk mengungkap hubungan yang lebih kompleks. Pada tahap ini, analisis *korelasi* digunakan untuk mengukur kekuatan hubungan linier antar variabel, *analisis tren* diterapkan untuk data temporal guna mengidentifikasi pola musiman atau kecenderungan waktu, dan *analisis klaster* (*cluster analysis*) dilakukan untuk mengelompokkan observasi berdasarkan kemiripan karakteristik(Solichin *et al.*, 2020).

Hasil dari keseluruhan proses EDA adalah Pemahaman Awal yang mendalam terhadap dataset. Insight ini menjadi landasan empiris untuk tahapan lanjutan dalam pipeline data, seperti *pembersihan data* (*data cleaning*), *rekayasa fitur* (*feature engineering*), dan *pemodelan statistik atau machine learning*(Suyanto, 2017). Dengan demikian, EDA berfungsi sebagai fase diagnostik yang kritis untuk memastikan validitas, kelayakan, dan kesiapan data sebelum diproses ke tahap analitik yang lebih kompleks.

6.4 Integrasi Pemrosesan Data dalam Alur Kerja Penelitian

Pemrosesan data melalui tahapan pembersihan, transformasi, dan visualisasi merupakan fondasi kritis untuk membangun sistem kecerdasan buatan yang akurat dan andal. Tahap pembersihan data tampak dalam berbagai bentuk, mulai dari standarisasi kualitas citra hingga penanganan data yang tidak relevan. Pada penelitian deteksi alat pelindung diri (APD), pembersihan mencakup penyiapan dataset yang seimbang antara citra penggunaan APD lengkap dan tidak lengkap, serta pemisahan data latih dan uji secara proporsional untuk menghindari bias evaluasi(Arifin *et al.*, 2025). Sementara itu, dalam penelitian bahasa isyarat, pembersihan data melibatkan ekstraksi dan pemfilteran koordinat tangan dari sensor Leap Motion untuk memastikan hanya fitur gerakan yang signifikan yang diproses lebih lanjut, menghilangkan noise atau data yang tidak utuh akibat gangguan selama akuisisi(Insani, Arifin and Rasyid, 2023).

Tahap transformasi data merupakan proses inti di mana data mentah diubah menjadi representasi fitur yang bermakna dan siap untuk diklasifikasi. Transformasi ini sangat bervariasi sesuai dengan karakteristik data dan tujuan penelitian. Pada domain citra, transformasi sering melibatkan konversi ruang warna (seperti dari RGB ke HSV atau L*a*b*) dan ekstraksi fitur tekstur (seperti GLCM dan LBP)(Arifin, Insani and Rasyid, 2023). Penelitian klasifikasi penyakit daun jagung dan kematangan tomat secara intensif memanfaatkan transformasi warna HSV untuk mengisolasi daerah daun atau buah berdasarkan karakteristik hue, saturation, dan value(Arifin and Insani, 2025). Di sisi lain, penelitian deteksi gerakan bahasa isyarat melakukan transformasi spasial dengan menormalisasi koordinat 3D jari tangan relatif terhadap pusat telapak tangan dan menghitung jarak Euclidean, mengubah gerakan fisik menjadi vektor numerik yang dapat dibandingkan.

Visualisasi data, meskipun tidak selalu menjadi output akhir yang eksplisit dalam semua penelitian, hadir sebagai alat bantu yang esensial dalam proses analisis dan interpretasi model. Dalam konteks pengolahan citra, visualisasi sering muncul dalam bentuk citra hasil segmentasi (seperti bounding box pada deteksi helm dan rompi, atau masking pada buah tomat) yang memungkinkan peneliti memverifikasi keakuratan pra-pemrosesan sebelum ekstraksi fitur(Arifin *et al.*, 2025). Lebih dari itu, visualisasi yang paling kritis adalah dalam bentuk matriks konfusi (*confusion matrix*) dan grafik performa metrik (akurasi, presisi, recall, F1-score) yang digunakan di hampir semua penelitian untuk mengevaluasi dan membandingkan kinerja berbagai algoritma klasifikasi, seperti CNN, SVM, Decision Tree, dan Backpropagation(Arifin and Insani, 2025). Visualisasi ini mentransformasikan hasil numerik kompleks menjadi insight yang mudah dipahami mengenai kekuatan dan kelemahan model.

Siklus ketiga tahapan ini bersifat iteratif dan saling bergantung. Sebagai contoh, hasil visualisasi dari matriks konfusi pada penelitian deteksi APD—yang menunjukkan misklasifikasi akibat kemiripan warna helm dengan penutup kepala—dapat menjadi umpan balik untuk memperbaiki tahap transformasi, misalnya dengan menambahkan fitur tekstur selain warna atau menyesuaikan threshold segmentasi HSV(Arifin *et al.*, 2025). Demikian pula, dalam penelitian perbandingan algoritma untuk penyakit daun jagung, visualisasi performa setiap kombinasi fitur (warna vs. tekstur) memandu peneliti dalam menyimpulkan bahwa fitur warna lebih dominan, sebuah insight yang dapat menyempurnakan strategi transformasi data pada penelitian selanjutnya(Arifin and Insani, 2025).

Secara keseluruhan, integrasi yang kokoh antara pembersihan, transformasi, dan visualisasi data dalam kelima penelitian ini tidak hanya menjamin validitas ilmiah dan

reproduktibilitas eksperimen, tetapi juga menjadi kunci dalam mentranslasikan masalah dunia nyata, dari keselamatan kerja, inklusi sosial, hingga ketahanan pangan, menjadi solusi komputasional yang praktis dan terukur. Proses ini menegaskan bahwa sebelum algoritma canggih dapat berjalan optimal, diperlukan fondasi data yang bersih, terstruktur dengan baik, dan dapat diinterpretasikan secara visual.

DAFTAR PUSTAKA

Arifin, N. *et al.* (2025) "Classification of Helmet and Vest Usage for Occupational Safety Monitoring using Backpropagation Neural Network," 6(3), pp. 1255–1266.

Arifin, N. and Insani, C.N. (2025) "Comparative Analysis of CNN , SVM , Decision Tree , Random Forest , and KNN for Maize Leaf Disease Detection Using Color and Texture Feature Extraction," 6(5), pp. 3572–3586.

Arifin, N., Insani, C.N. and Rasyid, M.R. (2023) "Klasifikasi Tingkat Kematangan Buah Tomat menggunakan Computer Vision untuk Smart Agriculture," *Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika dan Komputer)*, 22(2), p. 509. Available at: https://doi.org/10.53513/jis.v22i2.8387.

Firgiawan, W. *et al.* (2024) *DATA MINING DAN MANAJEMEN*.

Insani, C.N., Arifin, N. and Rasyid, M.R. (2023) "Deteksi Gerakan Bahasa Isyarat Menggunakan Euclidean Distance," *Informatik : Jurnal Ilmu Komputer*, 19(1), pp. 99–106. Available at: https://doi.org/10.52958/iftk.v19i1.5658.

Margareth, H. (2017) "Data Mining Algoritma C4.5," *Экономика Региона*, p. 32.

Muttaqin (2023) *Penganalan Data Mining*. 2023rd ed. Yayasan Kita Menulis,.

Rahayu, P. *et al.* (2018) *Buku Ajar Data Mining*.

Solichin *et al.* (2020) "Klasterisasi Persebaran Virus Corona (Covid-19) Di DKI Jakarta," *Fountain of Informatics Journal*, 5(2), pp. 52–59.

Sudipa, I.G.I. *et al.* (2024) *Buku ajar data mining*.

Sulianta, F. (2024) "Buku Dasar Data Mining from A to Z," (January).

Suyanto (2017) *Data mining untuk klasifikasi dan klasterisasi data*. 1st ed. Indonesia: Informatika.

Wahono, R.S. (2023) *Data Mining Data mining*, *Mining of Massive Datasets*. Available at: https://www.cambridge.org/core/product/identifier/CBO9781139058452A007/type/book_part.

BAB 7

ALGORITMA *MACHINE LEARNING: SUPERVISED LEARNING*

Oleh Dedy Abdianto Nggego

Machine Learning (ML) merupakan cabang dari kecerdasan buatan yang berfokus pada pengembangan algoritma yang memungkinkan sistem komputer untuk belajar dari data tanpa diprogram secara eksplisit. Salah satu paradigma utama dalam machine learning adalah *supervised learning*, yaitu pendekatan pembelajaran yang menggunakan data berlabel sebagai dasar pembentukan model prediktif. Dalam supervised learning, setiap data masukan (*input*) memiliki pasangan keluaran (*output*) yang telah diketahui sebelumnya, sehingga model dapat mempelajari hubungan antara keduanya secara sistematis.

Supervised learning memiliki peran yang sangat penting dalam berbagai aplikasi modern, seperti pengenalan citra, klasifikasi teks, prediksi cuaca, diagnosis penyakit, hingga pertanian cerdas (*smart farming*). Keberhasilan pendekatan ini sangat dipengaruhi oleh kualitas data berlabel, pemilihan fitur, serta algoritma yang digunakan. Oleh karena itu, pemahaman mendalam mengenai konsep, metode, dan evaluasi supervised learning menjadi hal yang krusial dalam pengembangan sistem berbasis data.

Bab ini bertujuan untuk membahas konsep supervised learning secara komprehensif, meliputi dasar teori, algoritma utama, formulasi matematis, implementasi menggunakan

bahasa pemrograman Python, serta evaluasi kinerja model. Pembahasan disertai dengan ilustrasi dan referensi ilmiah untuk mendukung landasan akademik.

7.1 Konsep Dasar *Machine Learning*

Machine learning dapat didefinisikan sebagai metode komputasi yang memungkinkan sistem untuk meningkatkan kinerjanya berdasarkan pengalaman atau data. Mitchell (1997) mendefinisikan machine learning sebagai berikut: *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"*.

Secara umum, machine learning diklasifikasikan ke dalam beberapa kategori utama:

1. ***Supervised Learning***, yaitu pembelajaran dengan data berlabel.
2. ***Unsupervised Learning***, yaitu pembelajaran tanpa label untuk menemukan pola tersembunyi.
3. ***Semi-Supervised Learning***, yaitu kombinasi data berlabel dan tidak berlabel.
4. ***Reinforcement Learning***, yaitu pembelajaran berbasis interaksi dengan lingkungan dan pemberian reward.

Dari keempat kategori tersebut, supervised learning merupakan pendekatan yang paling luas digunakan karena hasilnya dapat dievaluasi secara kuantitatif dan relatif mudah diinterpretasikan.

7.2 *Supervised Learning*

Supervised learning adalah metode pembelajaran di mana model dilatih menggunakan dataset yang terdiri dari pasangan input dan output. Misalkan diberikan dataset:

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

Dengan:

- $x_i \in \mathbb{R}^d$ sebagai vektor fitur
- y_i sebagai label atau target

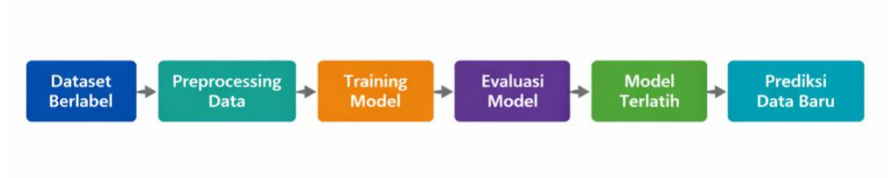
Tujuan supervised learning adalah mencari suatu fungsi hipotesis $f(x)$ sehingga:

$$f(x) \approx y$$

Model dilatih dengan meminimalkan fungsi kerugian (*loss function*):

$$\mathcal{L}(y, f(x))$$

Adapun alur kerja dari supervised learning adalah seperti digambarkan pada Gambar 7.1.



Gambar 7. 1 Alur Kerja Pembelajaran Mesin

Alur tersebut menunjukkan bahwa *supervised learning* tidak hanya berfokus pada pemodelan, tetapi juga mencakup tahapan persiapan data dan evaluasi.

7.3 Jenis Masalah dalam *Supervised Learning*

7.3.1 Klasifikasi

Klasifikasi bertujuan untuk memetakan data masukan ke dalam kelas diskrit. Secara matematis:

$$f: \mathbb{R}^d \rightarrow \{1,2, \dots, K\}$$

Contoh algoritma klasifikasi meliputi Logistic Regression, k-NN, SVM, dan Decision Tree.

Masalah klasifikasi banyak ditemui dalam pengenalan pola, diagnosis medis, dan analisis citra digital (Bishop, 2006; Russell & Norvig, 2021). Algoritma supervised learning yang umum digunakan pada permasalahan klasifikasi antara lain Logistic Regression, k-Nearest Neighbor (k-NN), dan Support Vector Machine (SVM).

Logistic Regression merupakan model probabilistik berbasis fungsi sigmoid yang digunakan untuk memodelkan peluang suatu kelas berdasarkan kombinasi linear fitur input. Metode ini banyak digunakan karena interpretabilitasnya yang tinggi dan efisiensi komputasi (Hosmer, Lemeshow & Sturdivant, 2013).

k-Nearest Neighbor (k-NN) adalah algoritma non-parametrik yang melakukan klasifikasi berdasarkan mayoritas kelas dari *k* tetangga terdekat dalam ruang fitur. Metode ini diperkenalkan oleh Cover dan Hart (1967) dan efektif untuk dataset berukuran kecil hingga menengah.

Support Vector Machine (SVM) bertujuan untuk menemukan hyperplane optimal yang memaksimalkan margin antar kelas. Pendekatan ini terbukti sangat kuat dalam menangani data berdimensi tinggi dan kasus non-

linear melalui penggunaan kernel (Cortes & Vapnik, 1995).

Berikut contoh kode klasifikasi dalam bahasa pemrograman python.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

7.3.2 Regresi

Regresi bertujuan untuk memprediksi nilai kontinu. Fungsi regresi umumnya dinyatakan sebagai:

$$y = f(x) + \varepsilon$$

dengan ε merupakan error.

7.4.3 Perbandingan Algoritma *Supervised Learning*

Tabel 7.1 berikut menyajikan perbandingan beberapa algoritma supervised learning yang paling umum digunakan, ditinjau dari karakteristik utama, kelebihan, keterbatasan, dan referensi ilmiah utama.

Tabel 7. 1 Perbandingan Algoritma *Supervised Learning*

Algoritma	Jenis Masalah	Karakteristik Utama	Kelebihan	Keterbatasan	Referensi utama
Logistic Regression	Klasifikasi	Model linear probabilistik berbasis fungsi sigmoid	Mudah diinterpretasi, efisien	Kurang baik untuk pola non-linear	Hosmer et al. (2013)
k-Nearest Neighbor (k-NN)	Klasifikasi / Regresi	Berbasis jarak, non-parametrik	Sederhana, tanpa proses training eksplisit	Sensitif terhadap noise dan skala data	Cover & Hart (1967)
Support Vector Machine (SVM)	Klasifikasi / Regresi	Margin maksimum, kernel trick	Akurat pada data berdimensi tinggi	Pemilihan kernel dan parameter kompleks	Cortes & Vapnik (1995)
Decision Tree	Klasifikasi / Regresi	Struktur pohon keputusan	Mudah dipahami, interpretatif	Rentan overfitting	Quinlan (1986)
Linear Regression	Regresi	Model linear berbasis least squares	Cepat, interpretatif	Asumsi linearitas	Montgomery et al. (2012)

Tabel tersebut menunjukkan bahwa tidak ada satu algoritma yang selalu unggul untuk semua permasalahan. Pemilihan algoritma harus mempertimbangkan karakteristik data, kompleksitas model, serta kebutuhan interpretabilitas.

7.5 Representasi Data dan *Feature Engineering*

Dalam *supervised learning*, performa model sangat dipengaruhi oleh bagaimana data direpresentasikan. Data mentah sering mengandung noise, perbedaan skala antar fitur, serta atribut yang tidak relevan. Feature engineering bertujuan mentransformasikan data tersebut agar lebih informatif dan sesuai dengan asumsi algoritma pembelajaran.

Proses *preprocessing* mencakup normalisasi dan standarisasi. Normalisasi min–max didefinisikan sebagai:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Sedangkan standarisasi (z-score) dirumuskan sebagai:

$$x' = \frac{x - \mu}{\sigma}$$

Feature selection dilakukan untuk memilih fitur paling relevan sehingga kompleksitas model dapat dikurangi dan risiko overfitting diminimalkan.

7.6 Algoritma *Supervised Learning*

7.6.1 *Linear Regression*

Linear Regression memodelkan hubungan linear antara variabel input dan output:

$$y = \beta_0 + \sum_{i=1}^{d} \beta_i x_i$$

Parameter diestimasi dengan meminimalkan *Mean Squared Error* (MSE):

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

7.6.2 *Logistic Regression*

Logistic Regression menggunakan fungsi sigmoid untuk klasifikasi biner:

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-(w^T x + b)}}$$

7.6.3 k-Nearest Neighbor (k-NN)

k-NN mengklasifikasikan data berdasarkan kedekatan jarak, umumnya menggunakan Euclidean distance:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2}$$

7.6.4 *Support Vector Machine* (SVM)

SVM mencari hyperplane dengan margin maksimum melalui optimasi:

$$\min_{w,b} \frac{1}{2} \| w \|^2$$

7.7 *Training* dan Optimasi Model

Proses *training* dalam supervised learning bertujuan untuk mempelajari parameter model dengan cara meminimalkan fungsi kerugian (*loss function*) yang merepresentasikan selisih antara prediksi model dan nilai target sebenarnya. Secara umum, proses ini dirumuskan sebagai masalah optimasi:

$$\min_{\mathrm{w}} \; J(\mathrm{w})$$

di mana w merupakan vektor parameter model dan $J(\mathrm{w})$ adalah fungsi objektif atau fungsi loss, seperti *mean squared error* (MSE) pada regresi atau *cross-entropy loss* pada klasifikasi.

Salah satu metode optimasi yang paling banyak digunakan adalah Gradient Descent, yang bekerja dengan memperbarui parameter model secara iteratif ke arah negatif gradien fungsi loss. Aturan pembaruan parameter dinyatakan sebagai:

$$\mathrm{w}_{t+1} = \mathrm{w}_t - \eta \nabla J(\mathrm{w}_t)$$

dengan:

- η adalah *learning rate*,
- $\nabla J(\mathrm{w}_t)$ adalah gradien fungsi loss terhadap parameter pada iterasi ke-t.

Learning rate η memainkan peran yang sangat krusial dalam menentukan stabilitas dan kecepatan konvergensi proses training. Nilai learning rate yang terlalu besar dapat menyebabkan proses optimasi menjadi tidak stabil dan gagal konvergen karena langkah pembaruan yang terlalu jauh dari titik minimum. Sebaliknya, learning rate yang terlalu kecil dapat menyebabkan konvergensi yang sangat lambat dan meningkatkan waktu komputasi secara signifikan.

Selain *batch gradient descent*, dalam praktik sering digunakan variasi lain seperti *Stochastic Gradient Descent* (SGD) dan Mini-Batch Gradient Descent**.** SGD memperbarui parameter menggunakan satu sampel data pada setiap iterasi, sehingga lebih efisien untuk dataset besar namun cenderung menghasilkan fluktuasi pada nilai loss. Mini-batch gradient descent merupakan kompromi antara efisiensi dan stabilitas dengan menggunakan sebagian kecil data pada setiap iterasi.

Untuk mengatasi keterbatasan learning rate statis, berbagai metode optimasi adaptif telah dikembangkan, seperti Momentum**,** AdaGrad**,** RMSProp**,** dan Adam, yang secara dinamis menyesuaikan langkah pembaruan parameter

berdasarkan riwayat gradien. Metode-metode ini terbukti mampu mempercepat konvergensi dan meningkatkan performa model, khususnya pada permasalahan dengan ruang parameter berdimensi tinggi dan lanskap loss yang kompleks.

7.8 Evaluasi Model *Supervised Learning*

Evaluasi model merupakan tahap krusial dalam supervised learning yang bertujuan untuk mengukur sejauh mana model mampu melakukan generalisasi, yaitu menghasilkan prediksi yang akurat pada data yang belum pernah dilihat selama proses pelatihan. Evaluasi yang baik tidak hanya menilai kinerja numerik model, tetapi juga memberikan pemahaman tentang karakteristik kesalahan, ketidakseimbangan kelas, serta implikasi praktis dari hasil prediksi.

Secara umum, proses evaluasi dilakukan dengan memisahkan data menjadi data pelatihan (training set) dan data pengujian (*testing set*), atau melalui teknik cross-validation untuk memperoleh estimasi performa yang lebih robust.

7.8.1 Evaluasi pada Masalah Klasifikasi

Pada permasalahan klasifikasi, evaluasi model biasanya didasarkan pada confusion matrix, yang menggambarkan hubungan antara kelas sebenarnya dan kelas hasil prediksi. Dari confusion matrix, berbagai metrik evaluasi dapat diturunkan, antara lain:

- ***Accuracy***, yang mengukur proporsi prediksi yang benar terhadap seluruh data:

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

- ***Precision***, yang mengukur ketepatan model dalam memprediksi kelas positif:

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

- ***Recall*** (*Sensitivity*), yang mengukur kemampuan model dalam mendeteksi seluruh sampel positif:

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

- **F1-*score***, yang merupakan rata-rata harmonik antara precision dan recall:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Metrik *accuracy* sering digunakan pada dataset yang seimbang, namun pada kasus ketidakseimbangan kelas (*imbalanced data*), precision, recall, dan F1-score menjadi lebih representatif dalam menggambarkan kinerja model.

Selain metrik di atas, evaluasi juga dapat diperluas menggunakan *Receiver Operating Characteristic* (ROC) *curve* dan *Area Under the Curve* (AUC), yang mengukur kemampuan model dalam membedakan kelas positif dan negatif pada berbagai threshold keputusan.

7.8.2 Evaluasi pada Masalah Regresi

Pada permasalahan regresi, evaluasi model berfokus pada besar kesalahan prediksi numerik. Beberapa metrik evaluasi yang umum digunakan antara lain:

- *Mean Absolute Error* (MAE):

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

- *Mean Squared Error* (MSE)**:**

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

- *Root Mean Squared Error* (RMSE)**:**

$$\text{RMSE} = \sqrt{\text{MSE}}$$

MAE memberikan bobot kesalahan yang sama pada setiap data, sedangkan MSE dan RMSE memberikan penalti yang lebih besar pada kesalahan yang ekstrem, sehingga lebih sensitif terhadap outlier. Pemilihan metrik evaluasi harus disesuaikan dengan karakteristik data dan tujuan aplikasi.

Selain itu, koefisien determinasi (R^2) sering digunakan untuk mengukur proporsi variansi target yang dapat dijelaskan oleh model:

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

7.8.3 *Cross-Validation* dan Generalisasi Model

Untuk meningkatkan keandalan evaluasi, teknik k-*fold cross-validation s*ering digunakan. Pada metode ini, dataset dibagi menjadi *k* subset, di mana model dilatih dan diuji secara bergantian sehingga setiap data digunakan sebagai data uji satu kali. Pendekatan ini

membantu mengurangi bias evaluasi dan memberikan estimasi performa yang lebih stabil.

7.9 *Overfitting* dan *Underfitting*

Overfitting dan *underfitting* merupakan dua permasalahan utama dalam pengembangan model supervised learning yang berkaitan dengan kemampuan generalisasi model. Underfitting terjadi ketika model terlalu sederhana sehingga tidak mampu menangkap pola penting dalam data, yang ditandai dengan kinerja buruk baik pada data latih maupun data uji. Sebaliknya, overfitting terjadi ketika model terlalu kompleks dan menyesuaikan diri secara berlebihan terhadap data latih, sehingga menghasilkan kinerja yang sangat baik pada data pelatihan tetapi menurun secara signifikan pada data pengujian.

Fenomena overfitting umumnya disebabkan oleh kompleksitas model yang tinggi, jumlah parameter yang besar, serta keterbatasan jumlah data pelatihan. Secara konseptual, hubungan antara kompleksitas model dan kesalahan prediksi sering digambarkan melalui *bias–variance trade-off*, di mana model dengan kompleksitas tinggi memiliki bias rendah namun varians tinggi, sehingga rentan terhadap overfitting.

7.9.1 Regularisasi sebagai Teknik Pencegahan *Overfitting*

Salah satu pendekatan yang paling umum digunakan untuk mengatasi overfitting adalah regularisasi, yaitu penambahan penalti terhadap kompleksitas model ke dalam fungsi loss. Regularisasi mendorong model untuk memilih parameter dengan nilai yang lebih kecil, sehingga mengurangi sensitivitas terhadap fluktuasi data latih.

Pada regularisasi L2 (*Ridge Regression*), fungsi objektif dimodifikasi sebagai berikut:

$$J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \parallel \mathbf{w} \parallel^2$$

dengan:

- $\mathbf{w}$ adalah vektor parameter model,
- $\lambda \geq 0$ adalah parameter regularisasi,
- $\parallel \mathbf{w} \parallel^2 = \sum_j w_j^2$ merupakan norma L2.

Parameter λ mengontrol tingkat regularisasi. Nilai λ yang besar akan menghasilkan model yang lebih sederhana dengan parameter kecil, sedangkan nilai λ yang terlalu kecil dapat menyebabkan model tetap overfitting. Oleh karena itu, pemilihan λ yang optimal biasanya dilakukan melalui teknik validasi seperti cross-validation.

7.9.2 Regularisasi L1 dan Elastic Net

Selain L2, terdapat pula regularisasi L1 (Lasso) yang menggunakan norma L1:

$$J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \parallel \mathbf{w} \parallel_1$$

Regularisasi L1 memiliki sifat menghasilkan solusi *sparse*, yaitu banyak parameter bernilai nol, sehingga sekaligus berperan sebagai metode seleksi fitur.

Kombinasi antara regularisasi L1 dan L2 dikenal sebagai Elastic Net, yang dirumuskan sebagai:

$$J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda_1 \parallel \mathbf{w} \parallel_1 + \lambda_2 \parallel \mathbf{w} \parallel^2$$

Pendekatan ini menggabungkan keunggulan stabilitas L2 dan kemampuan seleksi fitur dari L1.

7.9.3 Strategi Lain Mengatasi *Overfitting* dan *Underfitting*

Selain regularisasi, beberapa strategi lain yang umum digunakan antara lain:

1. Penambahan jumlah data pelatihan untuk meningkatkan representativitas data.
2. *Early stopping,* yaitu menghentikan proses training sebelum model mulai overfitting.
3. Reduksi dimensi fitur, seperti *Principal Component Analysis* (PCA).
4. *Ensemble learning,* seperti bagging dan boosting, yang menggabungkan beberapa model untuk meningkatkan generalisasi.

Pemahaman yang baik mengenai overfitting dan underfitting memungkinkan peneliti untuk merancang model yang seimbang antara kompleksitas dan kemampuan generalisasi, sehingga menghasilkan performa optimal pada data nyata.

7.10 Aplikasi dan *Tren Supervised Learning*

Supervised learning banyak diterapkan pada computer vision, pemrosesan bahasa alami, sistem rekomendasi, kesehatan, dan pertanian cerdas. Tren terkini mencakup deep learning, transfer learning, dan AutoML.

DAFTAR PUSTAKA

Bishop, C.M., 2006. *Pattern Recognition and Machine Learning*. New York: Springer.

Cortes, C. and Vapnik, V., 1995. Support-vector networks. *Machine Learning*, 20(3), pp.273–297.

Cover, T. and Hart, P., 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1), pp.21–27.

Goodfellow, I., Bengio, Y. and Courville, A., 2016. *Deep Learning*. Cambridge, MA: MIT Press.

Hosmer, D.W., Lemeshow, S. and Sturdivant, R.X., 2013. *Applied Logistic Regression*. 3rd ed. New York: Wiley.

Mitchell, T.M., 1997. *Machine Learning*. New York: McGraw-Hill.

Montgomery, D.C., Peck, E.A. and Vining, G.G., 2012. *Introduction to Linear Regression Analysis*. 5th ed. Hoboken, NJ: Wiley.

Quinlan, J.R., 1986. Induction of decision trees. *Machine Learning*, 1(1), pp.81–106.

BAB 8

ALGORITMA *MACHINE LEARNING: UNSUPERVISED LEARNING*

Oleh Selfina Pare

8.1 Pendahuluan *Unsupervised Learning*

Unsupervised Learning merupakan salah satu pendekatan fundamental dalam machine learning yang berfokus pada pengolahan data tanpa label (Budiasto, Tallulembang, *et al.*, 2025). Berbeda dengan supervised learning yang mengandalkan data beranotasi sebagai acuan pembelajaran, unsupervised learning bekerja dengan mengekstraksi pola, struktur tersembunyi, dan hubungan intrinsik langsung dari data mentah (Chander and Vijaya, 2021; Murphy, 2022). Pendekatan ini sangat relevan pada kondisi nyata di mana sebagian besar data yang tersedia belum memiliki label atau klasifikasi yang jelas. Perbedaan mendasar antara unsupervised learning dan supervised learning, khususnya dari aspek jenis data, tujuan analisis, dan metode evaluasi, dapat dilihat secara ringkas pada Tabel 8.1.

Tabel 8. 1 Perbandingan Konseptual: *Supervised vs Unsupervised Learning*

Aspek Perbandingan	*Supervised Learning*	*Unsupervised Learning*
Jenis Data	Berlabel (labeled data)	Tidak berlabel (unlabeled data)
Tujuan Utama	Prediksi dan klasifikasi	Penemuan pola dan struktur data
Contoh Masalah	Prediksi harga, klasifikasi spam	Segmentasi pelanggan, analisis pola
Evaluasi Model	Akurasi, precision, recall	Visualisasi, validasi internal
Ketergantungan Label	Sangat tinggi	Tidak bergantung label

Dalam praktik analisis data modern, unsupervised learning sering menjadi tahap awal eksplorasi data sebelum dilakukan pemodelan lanjutan. Melalui teknik-teknik tertentu, sistem mampu mengelompokkan data berdasarkan kemiripan karakteristik, mengidentifikasi anomali, serta menyederhanakan data berdimensi tinggi agar lebih mudah dipahami (Chander and Vijaya, 2021). Dengan demikian, unsupervised learning berperan penting dalam membantu analis dan peneliti memperoleh wawasan awal (*insight discovery*) yang tidak selalu dapat terlihat melalui analisis statistik konvensional.

Salah satu kekuatan utama *unsupervised learning* terletak pada kemampuannya untuk menemukan struktur alami dalam data. Misalnya, dalam dataset pelanggan, algoritma unsupervised dapat secara otomatis membagi pelanggan ke dalam segmen-segmen tertentu berdasarkan

pola perilaku tanpa perlu definisi kelas sebelumnya (Chilla *et al.*, 2024; Baskoro *et al.*, 2025). Hal ini menjadikan pendekatan ini sangat populer dalam berbagai bidang seperti analisis pasar, bioinformatika, pengolahan citra, keamanan siber, dan eksplorasi data skala besar.

Meskipun demikian, unsupervised learning juga memiliki tantangan tersendiri. Tidak adanya label pembanding menyebabkan hasil model sulit dievaluasi secara kuantitatif menggunakan metrik standar seperti akurasi. Interpretasi hasil sangat bergantung pada pemahaman konteks domain dan visualisasi data yang tepat. Oleh karena itu, keberhasilan penerapan unsupervised learning tidak hanya ditentukan oleh algoritma yang digunakan, tetapi juga oleh kualitas data, pemilihan fitur, serta kemampuan analis dalam menafsirkan pola yang dihasilkan (Wang *et al.*, 2019).

Bab ini akan membahas unsupervised learning secara sistematis, dimulai dari konsep dasar, jenis-jenis utama, hingga algoritma populer yang sering digunakan dalam praktik. Pemahaman terhadap unsupervised learning menjadi landasan penting sebelum memasuki topik-topik lanjutan seperti deep learning dan optimasi model, karena pendekatan ini membantu membangun intuisi terhadap struktur dan karakteristik data yang dianalisis.

8.2 Konsep Dasar dan Jenis *Unsupervised Learning*

Unsupervised learning merupakan pendekatan pembelajaran mesin yang bekerja dengan data tanpa label, di mana sistem tidak diberikan informasi target atau kelas yang harus dipelajari. Tujuan utama dari pendekatan ini adalah menemukan pola tersembunyi, struktur internal, dan hubungan alami di dalam data (Murphy, 2022; Budiasto, Ismanto, *et al.*, 2025). Oleh karena itu, unsupervised learning sering digunakan pada tahap awal analisis data untuk

memahami karakteristik dataset secara menyeluruh sebelum dilakukan pemodelan lanjutan.

Secara konseptual, unsupervised learning bergantung pada gagasan bahwa data memiliki tingkat kemiripan (*similarity*) dan perbedaan (*dissimilarity*) yang dapat diukur secara matematis. Konsep jarak (*distance metrics*) seperti *Euclidean Distance*, *Manhattan Distance*, dan *Cosine Similarity* menjadi fondasi utama dalam berbagai algoritma unsupervised learning (Hossain *et al.*, 2012; Chander and Vijaya, 2021; Liu, Chen and Jiao, 2023). Melalui pengukuran jarak inilah algoritma mampu mengelompokkan data (*clustering*), mendeteksi pola tersembunyi, serta menyederhanakan struktur data berdimensi tinggi ke dalam representasi yang lebih informatif. Berbagai metrik jarak yang umum digunakan beserta rumus matematisnya disajikan secara ringkas pada Tabel 8.2.

Tabel 8. 2 Rumus dan Karakteristik Dasar Metrik Jarak pada *Unsupervised Learning*

Metode	Rumus Matematis	Makna Konseptual
Euclidean Distance	$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$	Mengukur jarak geometris lurus antara dua titik dalam ruang berdimensi n
Manhattan Distance	$d(x,y) = \sum_{i=1}^{n}\lvert x_i - y_i\rvert$	Mengukur total perbedaan absolut antar dimensi secara terpisah
Cosine Similarity	$\text{Cosine}(x,y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$	Mengukur tingkat kemiripan arah vektor tanpa mempertimbangkan besar nilai

Metode	Rumus Matematis	Makna Konseptual
Keterangan: • $x = (x_1, x_2, \dots, x_n)$ dan $y = (y_1, y_2, \dots, y_n)$ merepresentasikan dua vektor data. • Pada *Euclidean* dan *Manhattan Distance*, nilai yang lebih kecil menunjukkan tingkat kemiripan yang lebih tinggi. • Pada *Cosine Similarity*, nilai yang semakin mendekati 1 menunjukkan kemiripan yang semakin kuat.		

Berbeda dengan supervised learning yang berorientasi pada prediksi, unsupervised learning lebih menekankan pada eksplorasi dan pemahaman data. Hasil yang diperoleh bukan berupa label prediksi, melainkan representasi baru dari data, kelompok-kelompok objek yang serupa, atau aturan hubungan antar variabel. Oleh karena itu, interpretasi hasil sangat bergantung pada konteks domain dan visualisasi yang digunakan.

Secara umum, unsupervised learning dapat diklasifikasikan ke dalam beberapa jenis utama. *Clustering* merupakan jenis yang paling populer, di mana data dikelompokkan berdasarkan tingkat kemiripan tertentu tanpa mengetahui jumlah atau jenis kelompok sebelumnya (Sonawane and Patil, 2020). Pendekatan ini banyak digunakan dalam segmentasi pelanggan, pengelompokan dokumen, dan analisis citra. Clustering membantu mengungkap struktur alami data yang sebelumnya tidak terlihat.

Jenis kedua adalah reduksi dimensi (*dimensionality reduction*), yang bertujuan menyederhanakan data berdimensi tinggi menjadi representasi yang lebih ringkas tanpa kehilangan informasi penting. Teknik ini sangat berguna untuk visualisasi data, mengurangi kompleksitas komputasi, serta mengatasi masalah *curse of dimensionality* (Oskolkov, 2022). Reduksi dimensi memungkinkan analis

memahami pola global data dalam ruang yang lebih sederhana.

Selain itu, terdapat pula *association rule learning*, yaitu pendekatan yang berfokus pada pencarian hubungan atau keterkaitan antar item dalam dataset. Teknik ini banyak diterapkan dalam analisis transaksi, seperti *market basket analysis*, untuk menemukan pola pembelian yang sering muncul secara bersamaan (Kaur and Kang, 2016). Meskipun tidak selalu dibahas secara mendalam dalam pembelajaran dasar, konsep ini tetap menjadi bagian penting dari ekosistem unsupervised learning.

Tabel 8. 3 Perbandingan Jenis *Unsupervised Learning*

Jenis Unsupervised Learning	Tujuan Utama	Contoh Penerapan	Output yang Dihasilkan
Clustering	Mengelompokkan data berdasarkan kemiripan	Segmentasi pelanggan, clustering dokumen	Kelompok/ klaster data
Reduksi Dimensi	Menyederhanakan data berdimensi tinggi	Visualisasi data, preprocessing model	Representasi fitur baru
Association Rule	Menemukan hubungan antar item	Market basket analysis	Aturan asosiasi

Secara keseluruhan, konsep dasar unsupervised learning menekankan pada kemampuan sistem untuk belajar secara mandiri dari data mentah. Dengan memahami jenis-jenis utama unsupervised learning, pembaca diharapkan mampu menentukan pendekatan yang tepat sesuai dengan karakteristik data dan tujuan analisis. Pemahaman ini menjadi fondasi penting sebelum mempelajari algoritma clustering dan teknik reduksi dimensi secara lebih teknis pada subbab berikutnya.

8.3 Algoritma *Clustering*

Clustering merupakan salah satu teknik paling penting dalam unsupervised learning yang bertujuan untuk mengelompokkan data ke dalam beberapa kelompok (*cluster*) berdasarkan tingkat kemiripan tertentu. Objek-objek dalam satu cluster memiliki karakteristik yang lebih mirip satu sama lain dibandingkan dengan objek pada cluster yang berbeda. Tanpa adanya label awal, algoritma clustering berupaya menemukan struktur alami yang tersembunyi di dalam data (Sonawane and Patil, 2020; Mandowen *et al.*, 2025).

Secara umum, proses clustering bergantung pada konsep kemiripan dan jarak. Setiap data direpresentasikan sebagai titik dalam ruang fitur, kemudian jarak antar titik dihitung menggunakan metrik tertentu seperti Euclidean, Manhattan, atau Cosine Similarity. Pemilihan metrik jarak sangat memengaruhi hasil clustering, terutama pada data berdimensi tinggi atau data dengan skala fitur yang berbeda. Oleh karena itu, tahap prapemrosesan seperti normalisasi data sering kali menjadi langkah penting sebelum menerapkan algoritma *clustering*.

8.3.1 K-Means *Clustering*

Salah satu algoritma *clustering* yang paling populer adalah K-Means *Clustering*. Algoritma ini bekerja dengan menentukan sejumlah cluster (K) di awal, kemudian secara iteratif memperbarui posisi pusat cluster (*centroid*) berdasarkan rata-rata data yang tergabung di dalamnya. K-Means dikenal karena kesederhanaan dan efisiensinya, sehingga banyak digunakan pada dataset berukuran besar (Tan, 2024). Namun, algoritma ini memiliki keterbatasan, seperti keharusan menentukan jumlah cluster sejak awal dan sensitivitas terhadap outlier.

Secara matematis, tujuan utama algoritma K-Means adalah meminimalkan total jarak kuadrat antara setiap data dan centroid cluster-nya (Nie *et al.*, 2023; Budiasto, Purnama, *et al.*, 2025). Fungsi objektif K-Means dirumuskan sebagai berikut:

$$J = \sum_{i=1}^{K} \sum_{x_j \in C_i} \| x_j - \mu_i \|^2$$

dengan keterangan:

- K: jumlah cluster yang ditentukan di awal
- C_i: himpunan data yang tergabung dalam cluster ke-i
- x_j: data ke-j dalam cluster C_i
- μ_i: centroid cluster ke-i, yang dihitung sebagai rata-rata seluruh data dalam cluster tersebut

Centroid setiap cluster diperbarui menggunakan persamaan:

$$\mu_i = \frac{1}{| C_i |} \sum_{x_j \in C_i} x_j$$

Proses optimasi ini dilakukan secara iteratif melalui dua tahap utama, yaitu penugasan data ke cluster terdekat berdasarkan jarak (umumnya jarak Euclidean) dan pembaruan posisi centroid hingga nilai fungsi objektif mencapai kondisi konvergen atau perubahan antar iterasi menjadi sangat kecil.

Untuk memahami cara kerja K-Means secara lebih konkret, diperlukan contoh implementasi langsung menggunakan dataset nyata. Pendekatan ini tidak hanya memperjelas konsep matematis dan mekanisme iteratif

K-Means, tetapi juga membantu melihat bagaimana algoritma tersebut membentuk cluster berdasarkan kemiripan data. Pada contoh berikut, digunakan Dataset Iris, salah satu dataset klasik dalam machine learning, untuk menunjukkan bagaimana K-Means mengelompokkan data tanpa memanfaatkan label kelas yang sebenarnya.

Contoh Implementasi K-Means Clustering dengan Python pada Dataset Iris

Dataset Iris terdiri dari 150 data bunga dengan empat fitur numerik. Dalam implementasi ini, dua fitur dipilih, yaitu *petal length* dan *petal width*, agar hasil clustering dapat divisualisasikan secara dua dimensi.

Kode Python berikut menunjukkan tahapan utama dalam penerapan algoritma K-Means, dimulai dari import library, pemuatan dataset, pemilihan fitur, proses pelatihan model, hingga visualisasi hasil clustering.

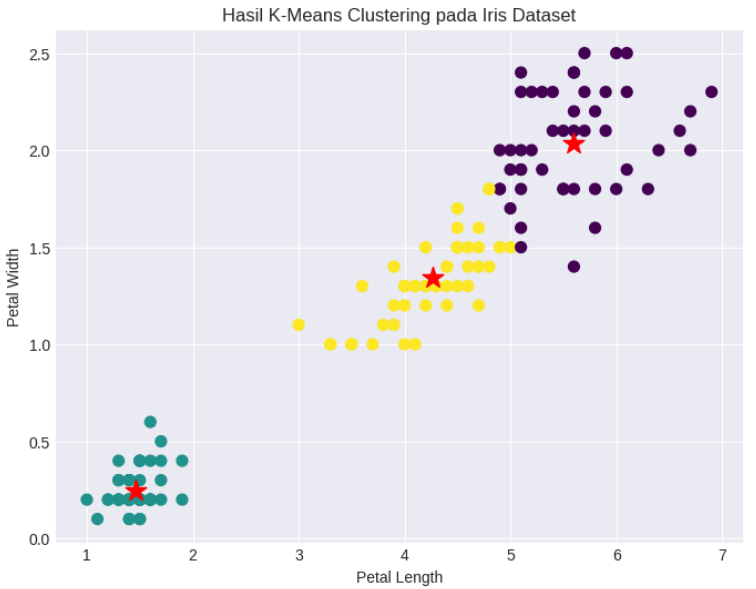
```
# Import Library dan Dataset
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans

# Memuat Dataset dan Memilih Fitur
iris = load_iris()
X = iris.data[:, 2:4]  # petal length dan
petal width

# Menerapkan Algoritma K-Means
```

```
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)


labels = kmeans.labels_
centroids = kmeans.cluster_centers_


# Visualisasi Hasil Clustering
plt.figure(figsize=(8,6))
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', s=50)
plt.scatter(centroids[:, 0], centroids[:, 1],
                        marker='*', s=200, color='red')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.title('Hasil K-Means Clustering pada Dataset Iris')
plt.show()
```

Interpretasi Hasil K-Means Clustering pada Dataset Iris

Berdasarkan hasil eksekusi kode tersebut, visualisasi yang dihasilkan ditampilkan pada Gambar 8.1. Titik-titik data direpresentasikan sebagai objek bunga Iris yang dipetakan berdasarkan nilai *petal length* dan *petal width*, sementara warna yang berbeda menunjukkan hasil pengelompokan (*cluster*) oleh algoritma K-Means. Simbol bintang berwarna merah merepresentasikan *centroid* dari masing-masing cluster

yang diperoleh melalui proses iteratif selama pelatihan model.



Gambar 8. 1 Visualisasi K-Means Clustering pada Dataset Iris

Visualisasi ini memperlihatkan bahwa meskipun algoritma K-Means tidak memanfaatkan label spesies asli, struktur cluster yang terbentuk tetap mencerminkan pola alami dalam dataset. Hal ini menegaskan peran K-Means sebagai metode yang efektif untuk eksplorasi data dan segmentasi awal, khususnya pada tahap awal analisis data tanpa label.

8.3.2 *Hierarchical Clustering*

Selain K-Means, terdapat *Hierarchical Clustering* yang membangun struktur pengelompokan secara bertingkat. Pendekatan ini tidak memerlukan penentuan jumlah cluster di awal, melainkan menghasilkan

representasi berupa *dendrogram* yang menunjukkan hubungan hierarkis antar data. Pengguna dapat menentukan jumlah cluster dengan memotong dendrogram pada tingkat tertentu. Hierarchical clustering sangat berguna untuk analisis eksploratif, meskipun kompleksitas komputasinya relatif lebih tinggi dibandingkan K-Means (Ma *et al.*, 2020).

Dalam praktiknya, hierarchical clustering umumnya diterapkan menggunakan pendekatan *agglomerative*, di mana data digabungkan secara bertahap berdasarkan tingkat kemiripan tertentu (Ran *et al.*, 2023). Metode *linkage* seperti Ward sering digunakan karena mampu menghasilkan cluster yang lebih kompak. Untuk memperjelas konsep ini, contoh implementasi berikut menggunakan dataset Iris dengan dua fitur asli tanpa reduksi dimensi, sehingga struktur pengelompokan dan hasil visualisasinya dapat diinterpretasikan secara langsung.

Contoh Implementasi Hierarchical Clustering dengan Python pada Dataset Iris

Implementasi berikut mendemonstrasikan penerapan hierarchical clustering dengan pendekatan agglomerative dan metode linkage Ward pada dataset Iris, termasuk visualisasi dendrogram dan hasil clustering dua dimensi.

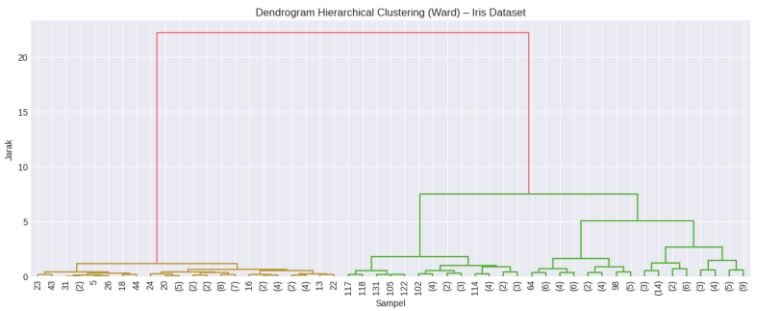
```
# Import Library dan Dataset
import numpy as np
import matplotlib.pyplot as plt


from sklearn.datasets import load_iris
from sklearn.preprocessing import
StandardScaler
```

```
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage


# Memuat Dataset dan Memilih Fitur
# Load dataset Iris
iris = load_iris()


# Gunakan dua fitur asli: petal length dan petal width
X = iris.data[:, [2, 3]]  # kolom ke-2 dan ke-3
feature_names = [iris.feature_names[2], iris.feature_names[3]]


# Normalisasi Data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


# Membuat Dendrogram
Z = linkage(X_scaled, method="ward")


plt.figure(figsize=(12, 5))
dendrogram(
    Z,
    truncate_mode="level",
```

```
    p=5,
    leaf_rotation=90,
    leaf_font_size=10
)
plt.title("Dendrogram Hierarchical
Clustering (Ward) - Iris Dataset")
plt.xlabel("Sampel")
plt.ylabel("Jarak")
plt.tight_layout()
plt.show()

# Melakukan Hierarchical Clustering
hc = AgglomerativeClustering(
    n_clusters=3,
    linkage="ward"
)

labels = hc.fit_predict(X_scaled)

# Visualisasi Hasil Clustering (2 Fitur
Asli)
plt.figure(figsize=(8, 6))
plt.scatter(
    X[:, 0], X[:, 1],
    c=labels,
    edgecolor="k"
)
```

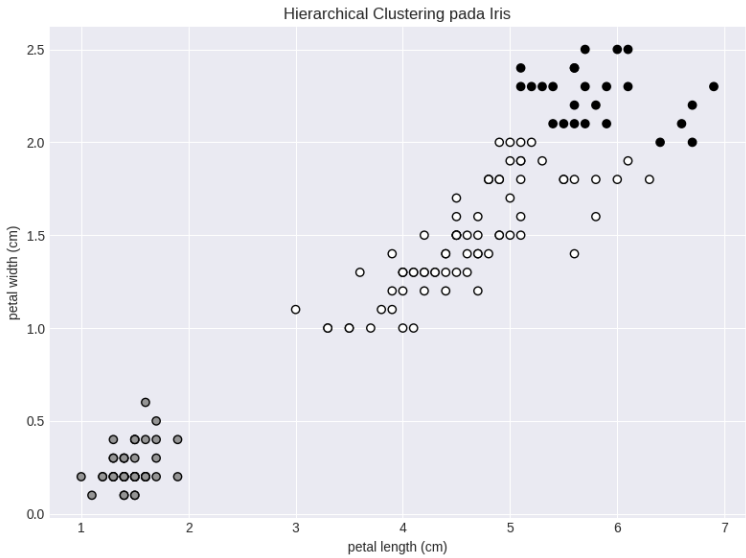
```
plt.xlabel(feature_names[0])
plt.ylabel(feature_names[1])
plt.title("Hierarchical Clustering pada Iris")
plt.tight_layout()
plt.show()
```

Interpretasi Hasil *Hierarchical Clustering* pada Dataset Iris



Gambar 8. 2 Visualisasi Dendrogram *Hierarchical Clustering* pada Dataset Iris

Dendrogram hasil *hierarchical clustering* dengan metode Ward sebagaimana ditunjukkan pada Gambar 8.2 memperlihatkan adanya tiga kelompok utama yang terbentuk secara alami pada dataset Iris. Pemotongan dendrogram pada jarak tertentu (ditandai dengan garis horizontal) menunjukkan bahwa data terbagi ke dalam cluster yang memiliki tingkat kemiripan internal tinggi dan perbedaan yang jelas antar cluster. Jarak penggabungan yang relatif besar antar cabang utama menandakan pemisahan cluster yang kuat, khususnya antara kelompok pertama dengan dua kelompok lainnya.



Gambar 8. 3 Visualisasi *Hierarchical Clustering* pada Dataset Iris

Visualisasi hasil clustering pada bidang *petal length* dan *petal width* yang ditampilkan pada Gambar 8.3 memperkuat temuan tersebut. Terlihat satu cluster yang sangat terpisah pada nilai *petal length* dan *petal width* yang rendah, yang merepresentasikan kelompok data dengan karakteristik morfologi bunga yang berbeda secara signifikan. Dua cluster lainnya berada pada rentang nilai yang lebih tinggi dan relatif berdekatan, namun tetap membentuk kelompok terpisah berdasarkan kepadatan dan pola distribusi data.

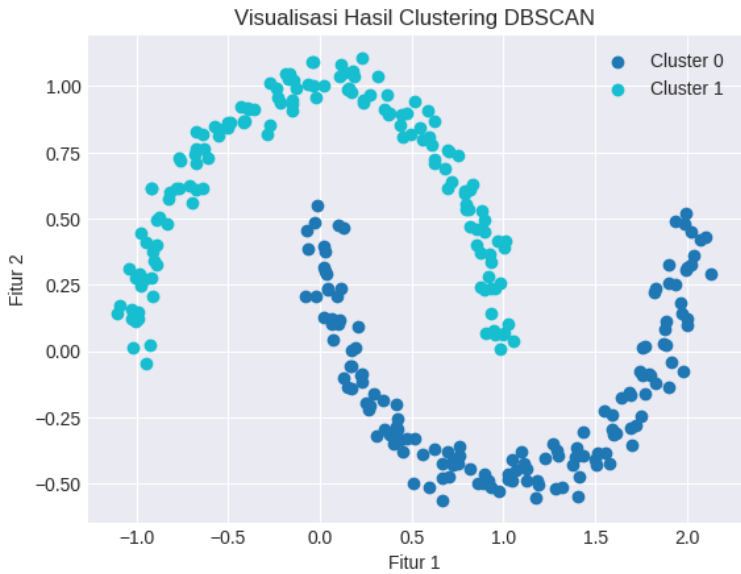
Secara keseluruhan, hasil ini menunjukkan bahwa hierarchical clustering mampu mengungkap struktur alami data tanpa memanfaatkan informasi label, sekaligus memberikan interpretasi visual yang intuitif melalui dendrogram dan *scatter plot*. Pendekatan ini sangat efektif untuk analisis eksploratif, khususnya

dalam memahami hubungan dan kedekatan antar objek data sebelum menerapkan metode machine learning lanjutan.

8.3.3 DBSCAN

Algoritma clustering lainnya yang cukup banyak digunakan adalah DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*). Berbeda dari pendekatan berbasis jarak atau hierarki, DBSCAN mengelompokkan data berdasarkan kepadatan. Algoritma ini mampu mengidentifikasi cluster dengan bentuk yang tidak beraturan serta mendeteksi *noise* atau outlier secara otomatis (Kalpavruksha *et al.*, 2025). DBSCAN sangat efektif pada data dengan distribusi kompleks, tetapi kinerjanya sangat bergantung pada pemilihan parameter kepadatan yang tepat.

Gambar 8.4 memperlihatkan contoh hasil clustering menggunakan algoritma DBSCAN pada data dengan struktur non-linear. Terlihat bahwa DBSCAN mampu memisahkan dua cluster utama dengan bentuk melengkung secara jelas, tanpa memaksakan bentuk cluster tertentu. Visualisasi ini menegaskan keunggulan DBSCAN dalam mengenali pola berbasis kepadatan sekaligus menghindari kesalahan pengelompokan yang umum terjadi pada algoritma clustering berbasis jarak.



Gambar 8. 4 Contoh Visualisasi DBSCAN

Dalam praktik nyata, pemilihan algoritma *clustering* tidak bersifat universal. Setiap algoritma memiliki kelebihan dan keterbatasan yang perlu disesuaikan dengan karakteristik data dan tujuan analisis. Clustering sering digunakan dalam berbagai aplikasi seperti segmentasi pelanggan, pengelompokan dokumen, analisis citra, dan deteksi anomali (Schiffers and Struchtrup, 2019; Jeena, Chaudhary and Thakur, 2023; Yaakub *et al.*, 2025). Pemahaman terhadap prinsip kerja dan karakteristik masing-masing algoritma *clustering* menjadi fondasi penting sebelum melangkah ke teknik unsupervised learning yang lebih lanjut, seperti reduksi dimensi dan visualisasi pola data.

8.4 Reduksi Dimensi dan Visualisasi Pola Data

Reduksi dimensi merupakan teknik penting dalam unsupervised learning yang bertujuan untuk menyederhanakan data berdimensi tinggi menjadi representasi dengan dimensi yang lebih rendah, tanpa menghilangkan informasi esensial yang terkandung di dalamnya (Nampira *et al.*, 2025; Venkatesh *et al.*, 2025). Dalam praktik machine learning modern, data sering kali memiliki puluhan hingga ribuan fitur, sehingga menyulitkan analisis, visualisasi, dan pemodelan. Reduksi dimensi hadir sebagai solusi untuk mengatasi kompleksitas tersebut sekaligus meningkatkan efisiensi komputasi.

Salah satu alasan utama diterapkannya reduksi dimensi adalah fenomena yang dikenal sebagai *curse of dimensionality*. Semakin tinggi dimensi data, semakin jarang distribusi data dalam ruang fitur, sehingga pengukuran jarak menjadi kurang bermakna. Kondisi ini dapat menurunkan kinerja algoritma machine learning, khususnya yang berbasis jarak seperti clustering. Dengan mereduksi dimensi, struktur global data menjadi lebih jelas dan stabil untuk dianalisis.

8.4.1 *Principal Component Analysis* (PCA)

Teknik reduksi dimensi yang paling umum digunakan adalah *Principal Component Analysis* (PCA). PCA bekerja dengan mentransformasikan data ke dalam sekumpulan komponen baru yang saling tidak berkorelasi dan diurutkan berdasarkan besarnya varians yang dijelaskan. Komponen utama pertama menangkap variasi terbesar dalam data, diikuti oleh komponen-komponen berikutnya. Melalui PCA, dimensi data dapat dikurangi secara signifikan dengan tetap

mempertahankan sebagian besar informasi penting (Benesty *et al.*, 2024).

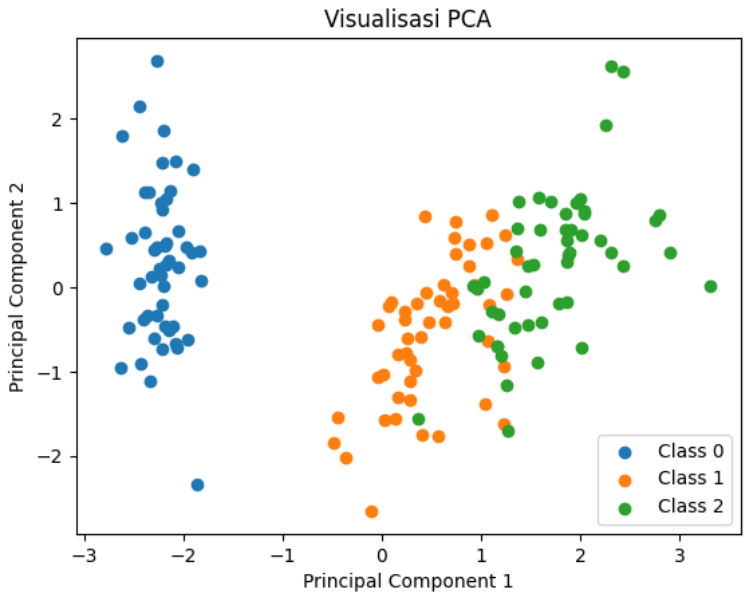
Untuk memberikan gambaran intuitif mengenai cara kerja *Principal Component Analysis* dalam mereduksi dimensi data sekaligus mempertahankan variasi informasi yang dominan, berikut disajikan contoh visualisasi PCA menggunakan dataset Iris. Pada contoh ini, seluruh fitur numerik terlebih dahulu distandarisasi, kemudian ditransformasikan ke dalam dua komponen utama sehingga distribusi data dapat divisualisasikan secara dua dimensi dan perbedaan antar kelas dapat diamati dengan lebih jelas.

```
# Import Library
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Load Dataset
data = load_iris()
X = data.data
y = data.target

# Standarisasi Data
X_scaled = StandardScaler().fit_transform(X)

# PCA ke 2 Dimensi
```

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Visualisasi
plt.figure()
for label in set(y):
    plt.scatter(
        X_pca[y == label, 0],
        X_pca[y == label, 1],
        label=f"Class {label}"
    )

plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.title("Visualisasi PCA")
plt.legend()
plt.show()
```

Visualisasi PCA sebagaimana ditunjukkan pada Gambar 8.5 memperlihatkan bahwa transformasi data ke dalam dua komponen utama mampu memisahkan data Iris ke dalam tiga kelompok yang relatif jelas berdasarkan distribusi pada *Principal Component 1* dan *Principal Component 2*. Terlihat bahwa satu kelompok data berada cukup terpisah di sisi kiri ruang PCA, sedangkan dua kelompok lainnya berada pada rentang nilai yang lebih berdekatan namun tetap menunjukkan pola distribusi yang berbeda. Hal ini mengindikasikan bahwa komponen utama pertama menangkap variasi

terbesar dalam data yang berkontribusi signifikan terhadap pemisahan antar kelas.



Gambar 8. 5 Visualisasi PCA

Selain itu, sebaran data pada komponen utama kedua memberikan informasi tambahan terkait variasi internal masing-masing kelompok. Meskipun PCA tidak menggunakan informasi label dalam proses transformasi, hasil visualisasi menunjukkan bahwa struktur kelas asli masih tercermin secara alami dalam ruang berdimensi lebih rendah (Vidal, Ma and Sastry, 2016). Dengan demikian, PCA terbukti efektif sebagai teknik reduksi dimensi untuk eksplorasi data, visualisasi awal, serta sebagai tahap pra-pemodelan sebelum diterapkannya algoritma machine learning lanjutan seperti *clustering* atau klasifikasi.

8.4.2 t-Distributed Stochastic Neighbor Embedding (t-SNE)

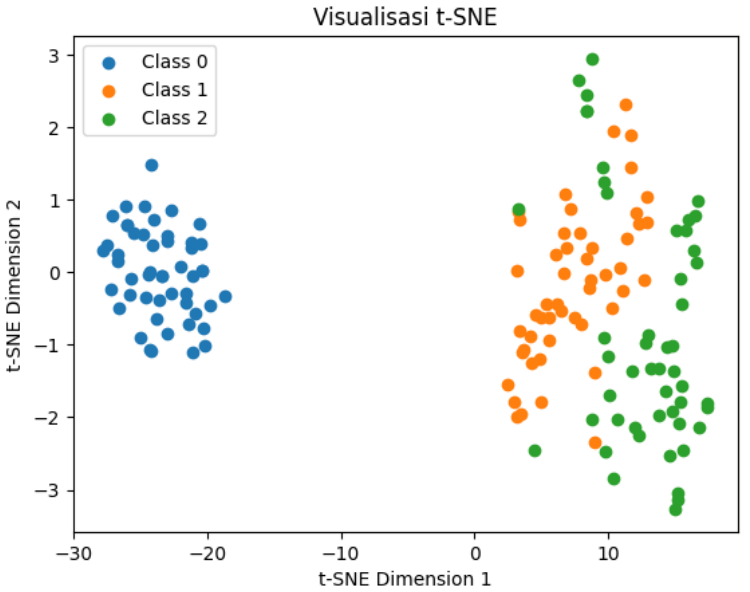
Selain PCA, terdapat teknik non-linear seperti *t-Distributed Stochastic Neighbor Embedding* (t-SNE) yang sangat efektif untuk visualisasi data berdimensi tinggi ke dalam ruang dua atau tiga dimensi (Yang, Li and Nai, 2023). Berbeda dengan PCA yang berfokus pada varians global, t-SNE menekankan pada pelestarian hubungan lokal antar data. Teknik ini sering digunakan untuk memvisualisasikan hasil clustering atau representasi fitur dari model deep learning, meskipun kurang cocok untuk tujuan reduksi dimensi pada tahap produksi karena kompleksitas komputasinya.

Untuk mengilustrasikan penerapan t-SNE dalam memetakan data berdimensi tinggi ke dalam ruang dua dimensi, berikut disajikan contoh implementasi t-SNE menggunakan dataset Iris yang telah distandarisasi. Visualisasi ini bertujuan untuk menunjukkan bagaimana t-SNE merepresentasikan kedekatan lokal antar data dalam bentuk sebaran dua dimensi yang mudah diinterpretasikan.

```
from sklearn.manifold import TSNE


# t-SNE ke 2 dimensi
tsne = TSNE(n_components=2,
perplexity=30, random_state=42)
X_tsne = tsne.fit_transform(X_scaled)


# Visualisasi
plt.figure()
for label in set(y):
```

```
    plt.scatter(
        X_tsne[y == label, 0],
        X_tsne[y == label, 1],
        label=f"Class {label}"
    )

plt.xlabel("t-SNE Dimension 1")
plt.ylabel("t-SNE Dimension 2")
plt.title("Visualisasi t-SNE")
plt.legend()
plt.show()
```

Visualisasi t-SNE sebagaimana ditunjukkan pada Gambar 8.6 memperlihatkan pemisahan kelompok data yang lebih tegas dibandingkan teknik reduksi dimensi linier seperti PCA. Terlihat bahwa masing-masing kelas pada dataset Iris membentuk klaster yang relatif kompak dan terpisah secara jelas, terutama pada dimensi pertama t-SNE. Hal ini mencerminkan kemampuan t-SNE dalam mempertahankan hubungan lokal antar data, sehingga objek-objek dengan kemiripan tinggi dipetakan berdekatan dalam ruang dua dimensi.



Gambar 8. 6 Visualisasi t-SNE

Selain itu, distribusi data yang tidak selalu mengikuti pola linier menegaskan sifat non-linear dari t-SNE dalam memproyeksikan data berdimensi tinggi. Meskipun jarak antar klaster pada visualisasi t-SNE tidak dapat diinterpretasikan secara kuantitatif sebagai jarak asli dalam ruang fitur, hasil visualisasi ini sangat berguna untuk eksplorasi data dan pemahaman awal terhadap struktur klaster yang tersembunyi. Oleh karena itu, t-SNE sering dimanfaatkan sebagai alat visualisasi pelengkap dalam analisis unsupervised learning, khususnya untuk mengevaluasi separabilitas data sebelum atau setelah proses clustering.

Untuk memperjelas perbedaan mendasar antara teknik reduksi dimensi linier dan non-linier yang telah dibahas, Tabel 8.4 menyajikan ringkasan perbandingan utama antara *Principal Component Analysis (PCA)* dan *t-*

Distributed Stochastic Neighbor Embedding (t-SNE) sebagai panduan cepat dalam pemilihan metode yang sesuai dengan tujuan analisis (Pareek and Jacob, 2021; Oskolkov, 2022).

Tabel 8. 4 Perbandingan PCA dan t-SNE

Aspek Kunci	PCA	t-SNE
Jenis Metode	Linier	Non-linier
Fokus Utama	Struktur global (varians)	Struktur lokal (kedekatan)
Tujuan	Reduksi dimensi & preprocessing	Visualisasi pola/klaster
Interpretasi	Tinggi (komponen bermakna)	Rendah (eksploratif)
Penggunaan	Pipeline machine learning	Analisis visual awal

Visualisasi pola data merupakan manfaat langsung dari penerapan reduksi dimensi. Dengan memetakan data ke dalam ruang dua atau tiga dimensi, analis dapat mengamati pola pengelompokan, keterpisahan antar cluster, serta keberadaan outlier secara intuitif. Visualisasi ini membantu proses interpretasi hasil unsupervised learning dan mendukung pengambilan keputusan berbasis data, terutama pada tahap eksplorasi awal.

Secara keseluruhan, reduksi dimensi dan visualisasi pola data berperan sebagai jembatan antara kompleksitas data mentah dan pemahaman manusia. Teknik-teknik ini tidak hanya meningkatkan efisiensi algoritma, tetapi juga memperkuat kemampuan analis dalam menafsirkan struktur dan hubungan tersembunyi

dalam data. Pemahaman yang baik terhadap reduksi dimensi menjadi bekal penting sebelum melanjutkan ke evaluasi dan integrasi model machine learning pada konteks yang lebih luas.

DAFTAR PUSTAKA

Al-Qerem, A. and Nashwan, S. (2025) 'Exploring Cybersecurity Data Patterns: A Comparative Analysis of Dimensionality Reduction Techniques', in *ACM International Conference Proceeding Series*, pp. 29–34. Available at: https://doi.org/10.1145/3711954.3711956.

Baskoro, S.E. *et al.* (2025) *Kecerdasan Buatan dalam Kehidupan Nyata: Konsep, Algoritma dan Penerapan*. Star Digital Publishing. Available at: https://books.google.co.id/books?id=tGWGEQAAQBAJ.

Benesty, J. *et al.* (2024) 'Principal Component Analysis in Noise Reduction and Beamforming', in *Springer Topics in Signal Processing*, pp. 57–85. Available at: https://doi.org/10.1007/978-3-031-36974-2_4.

Budiasto, J., Purnama, S.G., *et al.* (2025) *Data Science Technology*. PT. Star Digital Publishing, Yogyakarta-Indonesia. Available at: https://books.google.co.id/books?id=3fxlEQAAQBAJ.

Budiasto, J., Tallulembang, T.M., *et al.* (2025) *Machine Learning untuk Pemula: Konsep dan Implementasi*. Edited by T.Y. Zalni. Jakarta Utara: Penerbit Buku Indonesia.

Budiasto, J., Ismanto, H., *et al.* (2025) *Python untuk Data Science: Panduan Praktis Menguasai Analisis Data*. Edited by W. Yuliani. Padang: Literasi Langsung Terbit.

Chander, S. and Vijaya, P. (2021) 'Unsupervised learning methods for data clustering', in *Artificial Intelligence in Data Mining: Theories and Applications*, pp. 41–64.

Available at: https://doi.org/10.1016/B978-0-12-820601-0.00002-1.

Chilla, P.K. *et al.* (2024) 'Efficient Customer Segmentation Using Silhouette Based K-Means Algorithm', in *2024 4th International Conference on Artificial Intelligence and Signal Processing, AISP 2024*. Available at: https://doi.org/10.1109/AISP61711.2024.10870813.

Hossain, M.K. *et al.* (2012) 'Algorithms to calculate the Manhattan (L1) distance for vertical data represented in pTrees', in *Proceedings of the ISCA 27th International Conference on Computers and Their Applications, CATA 2012*, pp. 65–70. Available at: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84872004517&partnerID=40&md5=0ccc529ce84a53061517b595fddffe98.

Jeena, S., Chaudhary, A. and Thakur, A. (2023) 'Implementation & Analysis of Online Retail Dataset Using Clustering Algorithms', in *4th International Conference on Intelligent Engineering and Management, ICIEM 2023*. Available at: https://doi.org/10.1109/ICIEM59379.2023.10166552.

Kalpavruksha, R.N. *et al.* (2025) 'Kernel Density Based Spatial Clustering of Applications with Noise', in *Proceedings of the International Florida Artificial Intelligence Research Society Conference, FLAIRS*. Available at: https://doi.org/10.32473/flairs.38.1.138998.

Kaur, M. and Kang, S. (2016) 'Market Basket Analysis: Identify the Changing Trends of Market Data Using Association Rule Mining', in *Procedia Computer Science*, pp. 78–85. Available at: https://doi.org/10.1016/j.procs.2016.05.180.

Liu, S., Chen, Z. and Jiao, F. (2023) 'Detection of maize seed germination rate based on improved locally linear embedding', *Computers and Electronics in Agriculture*, 204. Available at: https://doi.org/10.1016/j.compag.2022.107514.

Ma, R. *et al.* (2020) 'Evaluation of Hierarchical Structures for Time Series Data', in *2020 5th IEEE International Conference on Big Data Analytics, ICBDA 2020*, pp. 94–99. Available at: https://doi.org/10.1109/ICBDA49040.2020.9101255.

Mandowen, S.A. *et al.* (2025) *Pengantar Machine Learning: Teori, Algoritma, dan Implementasi Modern*. Available at: https://books.google.co.id/books?id=EOadEQAAQBAJ.

Murphy, K.P. (2022) *Probabilistic machine learning: an introduction*. MIT press.

Nampira, A.A. *et al.* (2025) *Artificial Intelligence: A Guide for Thinking Humans*. PT. Green Pustaka Indonesia. Available at: https://books.google.co.id/books?id=_Y95EQAAQBAJ.

Nie, F. *et al.* (2023) 'An Effective and Efficient Algorithm for K-Means Clustering With New Formulation', *IEEE Transactions on Knowledge and Data Engineering*, 35(4), pp. 3433–3443. Available at: https://doi.org/10.1109/TKDE.2022.3155450.

Oskolkov, N. (2022) 'Dimensionality Reduction: Overview, Technical Details, and Some Applications', in *Tourism on the Verge*, pp. 151–167. Available at: https://doi.org/10.1007/978-3-030-88389-8_9.

Pareek, J. and Jacob, J. (2021) 'Data compression and visualization using pca and t-sne', in *Lecture Notes in Networks and Systems*, pp. 327–337. Available at: https://doi.org/10.1007/978-981-15-5421-6_34.

Ran, X. *et al.* (2023) 'Comprehensive survey on hierarchical clustering algorithms and the recent developments', *Artificial Intelligence Review*, 56(8), pp. 8219–8264. Available at: https://doi.org/10.1007/s10462-022-10366-3.

Schiffers, R. and Struchtrup, A.S. (2019) 'Anomaly detection in injection molding process data using cluster analysis', in *Annual Technical Conference - ANTEC, Conference Proceedings*. Available at: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85072967487&partnerID=40&md5=463ce4ff62cb7ce7129cba7c791ee708.

Sonawane, R.C. and Patil, H.D. (2020) 'Clustering Techniques and Research Challenages in Machine Learning', in *Proceedings of the 4th International Conference on Computing Methodologies and Communication, ICCMC 2020*, pp. 290–293. Available at: https://doi.org/10.1109/ICCMC48092.2020.ICCMC-00054.

Tan, Q. (2024) 'Intelligent Analysis And Processing Of Big Data Based On Improved K-Means Clustering Algorithm', in *Proceedings - 2024 International Conference on Internet of Things, Robotics and Distributed Computing, ICIRDC 2024*, pp. 254–259. Available at: https://doi.org/10.1109/ICIRDC65564.2024.00051.

Venkatesh, P.H. *et al.* (2025) 'Unsupervised learning and dimensionality reduction: Applications in drug discovery', in *Harnessing AI for Drug Discovery:*

Machine Learning Perspectives, pp. 95–118. Available at: https://www.scopus.com/inward/record.uri?eid=2-s2.0-105012698016&partnerID=40&md5=c2febf0d940b64f381e81d77228315b0.

Vidal, R., Ma, Y. and Sastry, S.S. (2016) 'Principal component analysis', in *Interdisciplinary Applied Mathematics*, pp. 25–62. Available at: https://doi.org/10.1007/978-0-387-87811-9_2.

Vybhavi, G.Y. *et al.* (2024) 'Clustering algorithms in data science: Evaluating the time and space complexities of K-means, DBSCAN, and hierarchical methods', in *AIP Conference Proceedings*. Available at: https://doi.org/10.1063/5.0215042.

Wang, S. *et al.* (2019) 'An Overview of Unsupervised Deep Feature Representation for Text Categorization', *IEEE Transactions on Computational Social Systems*, 6(3), pp. 504–517. Available at: https://doi.org/10.1109/TCSS.2019.2910599.

Yaakub, S. *et al.* (2025) *AUTOMASI MASA DEPAN: AI, MACHINE LEARNING, DAN PERUBAHAN DUNIA KERJA*. Edited by A. Yanto. Get Press Indonesia.

Yang, Z., Li, D. and Nai, W. (2023) 'T-SNE Based on Halton Sequence Initialized Butterfly Optimization Algorithm', in *ICEIEC 2023 - Proceedings of 2023 IEEE 13th International Conference on Electronics Information and Emergency Communication*, pp. 111–115. Available at: https://doi.org/10.1109/ICEIEC58029.2023.10199379.

BAB 9

DEEP LEARNING: NEURAL NETWORK DAN ARSITEKTUR DASAR

Oleh Marsujitullah

Perkembangan pesat teknologi komputasi dan ketersediaan data yang masif telah membuka babak baru dalam bidang kecerdasan buatan (*Artificial Intelligence*/AI). Salah satu cabang AI yang mengalami kemajuan sangat signifikan adalah *Deep Learning*, yang merupakan bagian dari *Machine Learning*. Deep Learning telah mengubah paradigma dalam pengolahan data dan membantu memecahkan berbagai masalah yang sebelumnya dianggap sulit, seperti pengenalan wajah, pemrosesan bahasa alami, dan kendaraan otonom (LeCun, Bengio, & Hinton, 2015).

Deep Learning menggunakan konsep dasar jaringan saraf tiruan (*Artificial Neural Network*) yang terinspirasi dari sistem saraf manusia. Model ini berusaha meniru cara kerja otak untuk mengolah dan memahami informasi. Perbedaan utama Deep Learning dengan pendekatan Machine Learning tradisional adalah kemampuannya untuk belajar secara otomatis dari data mentah tanpa perlu melakukan ekstraksi fitur manual yang rumit. Hal ini memungkinkan model Deep Learning untuk menghasilkan representasi fitur yang kompleks dan abstrak yang membantu meningkatkan presisi prediksi (Goodfellow, Bengio, & Courville, 2016).

Jaringan neural terdiri dari banyak neuron buatan yang tersusun berlapis-lapis, mulai dari lapisan input yang menerima data, lapisan tersembunyi yang mengolah data secara bertingkat, hingga lapisan output yang menghasilkan prediksi atau klasifikasi. Keberadaan banyak lapisan tersembunyi inilah yang membedakan Deep Learning dengan jaringan saraf tradisional dan memberikan kemampuan model untuk mengenali pola-pola rumit di dalam data. Konsep ini dikenal sebagai *deep neural networks* (jaringan saraf dalam) (Schmidhuber, 2015).

Pada bab ini, kita akan membahas teori dasar jaringan saraf mulai dari struktur dan cara kerjanya, proses pembelajaran menggunakan algoritma backpropagation dan optimasi gradien, hingga berbagai arsitektur dasar yang umum dipakai dalam Deep Learning. Pemahaman terhadap komponen ini sangat penting sebagai pijakan awal untuk memahami model deep learning yang lebih kompleks dan aplikasinya dalam dunia nyata (Nielsen, 2015).

Selain teori, bab ini juga menampilkan contoh implementasi sederhana neural network menggunakan bahasa pemrograman Python dengan library TensorFlow dan Keras. Dengan pendekatan praktis ini, pembaca dapat langsung melihat bagaimana konsep teoritis dapat diterapkan untuk menyelesaikan problema nyata. Diharapkan, setelah membaca bab ini, pembaca mampu memahami cara kerja neural network dan membuat model dasar deep learning dengan Python (Chollet, 2018).

Seiring dengan perkembangan aplikasi deep learning, berbagai bidang telah merasakan manfaatnya, mulai dari komputer visi, natural language processing, analisis data medis, hingga sistem rekomendasi. Contohnya, dalam komputer visi, teknologi deep learning dapat secara akurat mengenali objek dalam gambar dan video, mendeteksi anomali, serta melakukan segmentasi citra yang penting dalam bidang medis atau otomotif (LeCun et al., 2015). Di

bidang pemrosesan bahasa, model deep learning seperti RNN dan Transformer memungkinkan mesin untuk menerjemahkan bahasa, menjawab pertanyaan, dan berinteraksi dengan manusia secara natural (Vaswani et al., 2017).

Meskipun *Deep Learning* memiliki banyak keunggulan, bidang ini juga menghadapi tantangan tersendiri, seperti kebutuhan komputasi yang besar, pemahaman yang sulit terhadap model yang "*black-box*", serta perlunya data dalam jumlah besar dan berkualitas. Namun, berbagai riset dan inovasi terus dilakukan untuk mengatasi kendala tersebut, sehingga *Deep Learning* semakin mudah diakses dan diaplikasikan secara luas (Goodfellow et al., 2016).

Ringkasnya, bab ini adalah pintu gerbang untuk memahami Deep Learning dimulai dari jaringan saraf dan arsitektur dasar yang mendasari berbagai model canggih yang digunakan saat ini. Pembelajaran dimulai dari konsep fundamental akan mempersiapkan pembaca untuk menghadapi tantangan dan peluang dalam pengembangan kecerdasan buatan modern.

9.1Apa Itu Neural *Network*?

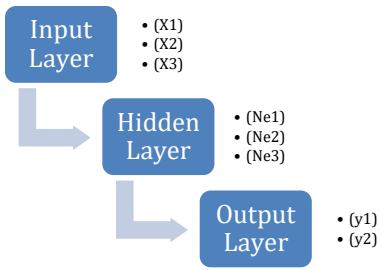
Jaringan saraf tiruan atau *Neural Network* adalah model komputasi yang terinspirasi dari struktur dan fungsi saraf biologis pada otak manusia. Dalam dunia kecerdasan buatan dan machine learning, neural network digunakan sebagai model yang mampu mempelajari pola dan hubungan kompleks dalam data, sehingga dapat digunakan untuk melakukan prediksi, klasifikasi, dan berbagai tugas lainnya.

Pada dasarnya, sebuah jaringan saraf terdiri dari kumpulan unit-unit pemrosesan sederhana yang disebut *neuron* atau *node*. Neuron-neuron ini tersusun dalam beberapa lapisan: lapisan input, satu atau beberapa lapisan

tersembunyi (*hidden layers*), dan lapisan output. Data mentah masuk ke lapisan input, kemudian diproses secara bertahap melalui lapisan tersembunyi yang menerapkan bobot (*weights*) dan bias (*biases*) pada sinyal input, dan akhirnya menghasilkan output yang mewakili keputusan atau prediksi jaringan (Goodfellow, Bengio, & Courville, 2016).

9.1.1 Struktur dan Elemen Neural *Network*

1. Neuron: Setiap neuron menerima input berupa sejumlah nilai numerik, mengalikan setiap input dengan bobotnya, menjumlahkannya, menambahkan bias, lalu menerapkan fungsi aktivasi. Fungsi aktivasi ini memberikan non-linearitas yang memungkinkan jaringan belajar pola-pola rumit dan kompleks (Nielsen, 2015).
2. Lapisan Input: Meliputi fitur dari data mentah yang akan diproses; contoh, pixel gambar atau nilai sensor.
3. Lapisan Tersembunyi: Memproses input menggunakan bobot, bias, dan fungsi aktivasi. Kehadiran fitur ini memungkinkan neural network untuk menangkap pola berlapis, mulai fitur sederhana hingga fitur kompleks.
4. Lapisan Output: Menghasilkan nilai akhir berupa prediksi, seperti kelas label pada masalah klasifikasi atau nilai kontinu pada masalah regresi (Goodfellow et al., 2016).



Gambar 9. 1 Ilustrasi Sederhana Jaringan Saraf Tiga Lapis

Jaringan saraf dengan banyak lapisan tersembunyi disebut sebagai *Deep Neural Networks*, sedangkan yang hanya memiliki satu atau dua lapisan tersembunyi disebut *shallow* neural networks. Setiap panah mewakili koneksi dengan bobot yang akan dioptimalkan selama proses pelatihan

9.1.2 Elemen Utama

Terdapat 3 elemen utama yang menjadi landasan penerapan Neural Network yakni Bobot, Bias dan Fungsi Aktivasi.

1. Bobot (*Weights*)

   Bobot menunjukkan kekuatan koneksi antar neuron. Bobot menentukan seberapa besar kontribusi output neuron di lapisan sebelumnya terhadap neuron pada lapisan berikutnya.

2. Bias (Bias)

   Bias adalah nilai konstan yang ditambahkan untuk membantu model mempelajari pola yang lebih kompleks dan memberikan fleksibilitas dalam penyesuaian fungsi aktivasi.

3. Fungsi Aktivasi (*Activation Function*)

   Fungsi aktivasi memperkenalkan non-linearitas agar jaringan dapat mempelajari pola-pola rumit yang tidak bisa dipetakan oleh fungsi linear saja.

Beberapa fungsi aktivasi yang umum:

a) Sigmoid: Mengubah input menjadi nilai antara 0 dan 1, cocok untuk klasifikasi biner.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

b) ReLU (*Rectified Linear Unit*): Menghasilkan input jika positif, dan nol jika negatif, sangat populer karena efisiensi dan kemampuan mengatasi masalah *vanishing gradient* pada jaringan dalam.

$$f\ (x) = \max\ (0, x)$$

c) Tanh: Memetakan nilai input ke kisaran antara -1 dan 1, memberikan output yang bersifat "mean zero" sehingga mempercepat proses belajar (LeCun, Bengio, & Hinton, 2015).

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

9.1.3 Proses Perhitungan di Neuron

Setiap neuron menghitung nilai keluaran secara matematis sebagai berikut:

$$z = \sum_{i=1}^{n} w_i x_i + b$$

di mana:

- (x_i) = input neuron
- (w_i) = bobot koneksi ke input tersebut
- (b) = bias neuron
- (z) = nilai yang akan diproses ke fungsi aktivasi

Kemudian, output neuron:

$$a = \phi(z)$$

dengan (\phi) adalah fungsi aktivasi.

9.1.4 Contoh Penerapan Neural *Network*

Salah satu contoh aplikasi klasik neural network adalah pengenalan tulisan tangan pada dataset MNIST, yang berisi gambar angka 0 hingga 9 berukuran 28x28 piksel. Neural network menerima pixel sebagai input, lalu memprosesnya melalui lapisan tersembunyi untuk mengekstrak fitur yang relevan, dan pada akhirnya mengklasifikasikan gambar ke dalam 10 kelas. Berkat kemampuannya menangkap pola visual yang kompleks, neural network dapat mencapai akurasi yang sangat tinggi pada tugas ini (LeCun et al., 1998).

Selain di bidang pengenalan gambar, neural network juga digunakan dalam bidang seperti pengolahan bahasa alami (natural language processing), di mana model dapat mempelajari makna kalimat dan konteks kata-kata yang terdapat dalam teks, serta dalam sistem rekomendasi seperti yang digunakan oleh platform streaming dan belanja daring (Chollet, 2018).

Neural network memungkinkan model untuk memetakan hubungan yang kompleks dan non-linear dari data input ke output, sehingga mampu menangani

aplikasi yang sulit seperti pengenalan gambar dan suara. Namun, mereka juga membutuhkan data pelatihan yang besar, komputasi tinggi, serta berpotensi sulit diinterpretasikan.

Tabel 9. 1 Perbandiangan Fungsi Aktivasi

Fungsi Aktivasi	Rentang Output	Kelebihan	Kekurangan	Umum Digunakan Untuk
Sigmoid	(0, 1)	Mudah dipahami, output probabilistik	Vanishing gradient, lambat	Klasifikasi biner, lapisan output
ReLU	[0, ∞)	Cepat, membantu pelatihan jaringan dalam	Tidak aktif saat nilai negatif	Lapisan tersembunyi jaringan deep
Tanh	(-1, 1)	Output mean zero, mempercepat konvergensi	Masih rentan vanishing gradient	Hidden layer dalam beberapa kasus

9.1.5 Perbedaan Dengan *Machine Learning* Tradisional

Model *machine learning* tradisional seperti regresi linier atau pohon keputusan bergantung pada fitur yang dirancang secara manual dan memiliki keterbatasan dalam menangani data dengan dimensi tinggi dan pola-pola yang sangat kompleks. Sebaliknya, neural network secara otomatis dapat mengekstrak fitur dari data mentah, seperti gambar atau suara, sehingga sangat cocok untuk aplikasi yang memerlukan analisis data kompleks dan non-linear (Goodfellow et al., 2016).

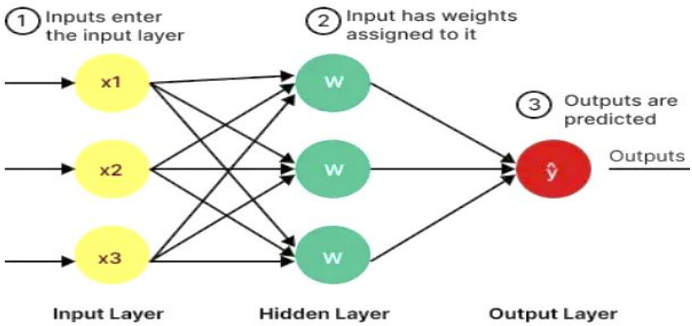
9.2 Proses Pembelajaran Neural *Network*

Neural network bukan hanya sekedar struktur yang tersusun dari neuron-neuron, tapi juga sebuah sistem yang dapat belajar dari data. Proses pembelajaran inilah yang membuat neural network mampu mengenali pola dan membuat prediksi yang akurat. Secara singkat, pembelajaran neural network melibatkan penyesuaian bobot (*weights*) dan bias (biases) berdasarkan data latih, dengan tujuan meminimalkan kesalahan prediksi

Pembelajaran neural network dilakukan melalui siklus berulang yang melibatkan dua tahap utama: *feedforward* dan *backpropagation*, serta mekanisme optimisasi bobot menggunakan algoritma gradient descent atau variasinya

9.2.1 Tahap *Feedforward*

Pada tahap *feedforward*, data input diberikan ke jaringan mulai dari lapisan input, kemudian dihantarkan maju menuju lapisan tersembunyi dan lapisan output. Setiap neuron di lapisan selanjutnya menerima sinyal yang merupakan kombinasi bobot dan output dari neuron lapisan sebelumnya, dihitung menggunakan rumus linear dan fungsi aktivasi. Proses ini menghasilkan output prediksi jaringan untuk data tersebut.



Gambar 9. 2Alur *Feedforward*

Sumber : www.geeksforgeeks.org

- (x_i) adalah tiap input fitur
- (w_{ij}) adalah bobot koneksi dari neuron ke neuron di lapisan berikutnya
- Output dari neuron dihitung dengan (a = \phi(\sum w_i x_i + b)), di mana (\phi) fungsi aktivasi

9.3.2 Menghitung Fungsi Loss

Setelah output dihasilkan, tahap selanjutnya adalah mengukur seberapa bagus prediksi tersebut dibandingkan dengan nilai target (label sebenarnya). Ukuran ini dikenal sebagai *fungsi loss* atau *cost function*.

Beberapa fungsi loss umum:

Tabel 9. 2 Fungsi Loss Umum

Fungsi Loss	Digunakan Untuk	Formula Singkat
Mean Squared Error (MSE)	Regresi	(\frac{1}{n} \sum (y_i - \hat{y}_i)^2)
Cross-Entropy Loss	Klasifikasi	(- \sum y_i \log(\hat{y}_i))

Nilai loss yang dihasilkan adalah ukuran kesalahan pada prediksi semakin kecil nilai loss, semakin baik kinerja jaringan

9.3.2 Tahap *Backpropagation*

Backpropagation adalah algoritma inti dalam pelatihan neural network yang digunakan untuk menghitung gradien (turunan) fungsi loss terhadap setiap bobot jaringan. Gradien ini menunjukkan arah dan besarnya perubahan bobot agar loss berkurang.

Proses ini menggunakan prinsip *chain rule* dari kalkulus untuk "mengalirkan balik" kesalahan dari output ke lapisan-lapisan terdahulu, sehingga setiap bobot dapat diperbarui secara tepat.

Secara ringkas, backpropagation melakukan tiga langkah utama:

1. Hitung error pada lapisan output: selisih antara prediksi dan target.
2. Propagasi error ke lapisan tersembunyi: menghitung kontribusi error tiap neuron.

3. Hitung gradien dan update bobot: gunakan gradien tersebut dalam algoritma optimasi.

9.4 Optimasi Bobot: Gradient Descent

Bobot dan bias diperbarui menggunakan algoritma optimasi yang meminimalkan fungsi loss berdasarkan gradien yang dihitung dari backpropagation. Algoritma dasar yang digunakan adalah *Gradient Descent*.

Rumus pembaruan bobot:

$$\omega := \omega - \eta \frac{\partial \mathcal{L}}{\partial \omega}$$

di mana:

- (w) adalah bobot yang akan diperbarui
- (\eta) (*learning rate*) adalah kecepatan pembelajaran (nilai kecil seperti 0.01)
- (\frac{\partial L}{\partial w}) adalah turunan fungsi loss terhadap bobot

Jenis-jenis *gradient descent:*

- *Batch Gradient Descen*t: menggunakan seluruh data latih untuk sekali update bobot. Akurat tapi lambat untuk data besar.
- *Stochastic Gradient Descent* (SGD): menggunakan satu sampel data tiap update bobot, lebih cepat tapi hasilnya lebih bervariasi.
- *Mini-batch Gradient Descent*: mengkombinasikan kedua metode di atas, menggunakan subset data (batch) untuk update, populer pada deep learning.

9.5 Epoch dan Iterasi

Epoch adalah satu kali proses penuh melewatkan seluruh data latih melalui feedforward dan backpropagation.

Satu epoch mungkin terdiri dari beberapa iterasi tergantung batch size. Model biasanya dilatih dalam banyak epoch agar bobot semakin baik dan jaringan dapat belajar secara optimal.

Contoh Penerapannya:

Misalnya, dalam klasifikasi citra anjing dan kucing, input berupa gambar melewati lapisan-lapisan jaringan. Setelah output prediksi diterima, dicari seberapa jauh prediksi tersebut meleset dari label sebenarnya (misal anjing atau kucing). Kesalahan tersebut kemudian dipropagasi ke belakang untuk memperbaiki bobot koneksi agar jaringan semakin akurat dalam mengidentifikasi gambar di iterasi berikutnya.

Tabel 9. 3 Proses Pelatihan Neural Network

Tahap	Deskripsi	Tujuan
Feedforward	Menghasilkan output berdasarkan input	Prediksi awal model
Hitung Loss	Menghitung kesalahan output vs target	Ukur performa
Backpropagation	Hitung gradien fungsi loss terhadap bobot	Tentukan arah pembaruan bobot
Optimasi	Update bobot menggunakan gradien	Turunkan loss (perbaiki model)

Tahap	Deskripsi	Tujuan
Epoch	Ulangi proses untuk seluruh data	Latih model sampai konvergen

9.6 Arsitek Dasar Neural *Network*

9.6.1 Pengantar Arsitektur Neural *Network*

Arsitektur neural network mengacu pada cara neuron-neuron disusun dan dihubungkan dalam jaringan. Pemilihan arsitektur yang tepat adalah kunci sukses penerapan deep learning karena mempengaruhi kapasitas belajar dan performa model.

Secara umum, arsitektur dasar neural network terdiri dari beberapa jenis, masing-masing dirancang untuk tipe data dan masalah yang spesifik. Tiga arsitektur dasar yang paling umum adalah:

- *Feedforward Neural Network* (FNN)
- *Convolutional Neural Network* (CNN)
- *Recurrent Neural Network* (RNN)

9.6.2 Pengantar Arsitektur Neural *Network*

Feedforward Neural *Network* adalah bentuk paling sederhana dari jaringan saraf tiruan. Arsitektur ini memiliki aliran informasi satu arah dari input ke output tanpa mekanisme pengulangan atau siklus.

Ciri-ciri FNN:

- Tersusun atas lapisan bertingkat: input, satu atau lebih hidden layer, dan output.
- Setiap neuron terhubung hanya ke neuron pada lapisan berikutnya.

- Tidak menyimpan state atau memori dari input sebelumnya.

Tabel 9. 4 Kelebihan dan Kekurangan

Kelebihan	Kekurangan
Struktur sederhana dan mudah dipahami	Tidak cocok untuk data berurutan atau waktu
Cepat dilatih pada data berdimensi rendah	Kurang efektif untuk data spasial/temporal

Penerapannya :

FNN banyak digunakan dalam menghadapi masalah klasifikasi umum pada data terstruktur (misal, klasifikasi spam email) atau regresi sederhana.

9.6.3 Convolutional Neural *Network* (CNN)

CNN dirancang khusus untuk data visual seperti gambar dan video. CNN menggunakan lapisan konvolusi yang dapat mengekstrak fitur spasial dari data input secara efisien tanpa perlu koneksi penuh antar neuron.

Struktur CNN:

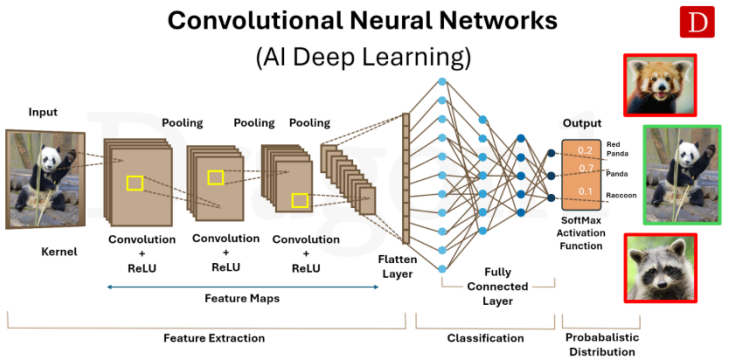
- Lapisan Konvolusi (*Convolutional Layer*): Menerapkan filter (kernel) yang bergerak melintasi gambar untuk mendeteksi fitur lokal seperti tepi, sudut, atau tekstur.
- Lapisan Pooling: Melakukan downsampling guna mengurangi dimensi dan mengontrol overfitting.
- Lapisan Fully-connected: Bagian akhir jaringan yang menginterpretasikan fitur yang telah diekstraksi dan menghasilkan output klasifikasi atau regresi.

Kelebihan CNN:

- Memanfaatkan parameter sharing dan sparcity koneksi sehingga efisien secara komputasi.
- Mampu secara otomatis mengekstrak fitur hierarkis mulai dari detail sederhana ke fitur kompleks.
- Sangat efektif untuk pengolahan citra, video, dan data spasial lainnya.

Penerapan CNN:

- Pengenalan wajah dan objek dalam gambar (misal: ImageNet *Challenge*).
- Analisis citra medis, seperti deteksi tumor dalam MRI.
- Sistem pengawasan video dan augmented reality.



Gambar 9. 3 Arsitektur CNN

Sumber : www.dragon1.com

9.6.4 *Recurrent Neural Network* (RNN)

RNN dirancang untuk menangani data berurutan atau sequential seperti teks, audio, dan deretan waktu (*timeseries*). RNN memiliki koneksi rekursif yang memungkinkan informasi sebelumnya digunakan dalam pemrosesan data saat ini.

Ciri dan Karakteristik

- Memiliki *state* internal yang secara dinamis diperbarui tiap input baru.
- Dapat memodelkan dependensi jangka pendek dan jangka panjang dalam data urutan.
- Versi populer seperti LSTM (*Long Short-Term Memory*) dan GRU (*Gated Recurrent Unit*) mengatasi masalah vanishing gradient.

Penerapan RNN:

1. Mesin penerjemah bahasa otomatis.
2. Analisis sentimen dalam teks.
3. Speech recognition dan prediksi seri waktu

Tabel 9. 5 Perbandingan Arsitektur

Aspek	***Feedforward Neural Network***	***Convolutional Neural Network***	***Recurrent Neural Network***
Data yang diproses	Data statis, tidak berurutan	Data spasial (gambar, video)	Data urutan / sequential
Koneksi antar neuron	Satu arah dari lapisan ke lapisan	Koneksi lokal berbasis filter	Koneksi rekursif (feedback loop)
Memori state sebelumnya	Tidak	Tidak	Ya, menyimpan konteks sebelumnya
Kelebihan utama	Sederhana dan cepat	Efisien pada gambar, ekstraksi fitur otomatis	Memahami konteks dan urutan

9.7 Implementasi NN Dengan Phyton Sederhana

Setelah memahami teori dasar dan arsitektur, langkah penting berikutnya adalah mengaplikasikan neural network secara praktis. Python menjadi bahasa paling populer karena memiliki library deep learning yang kuat dan mudah dipakai seperti TensorFlow dan Keras.

Pada bagian ini, kita akan membangun sebuah neural network sederhana untuk klasifikasi data gambar digit angka tangan menggunakan dataset MNIST.

9.7.1 Implementasi pada Phyton

Install library yang diperlukan dengan perintah :

```
pip install tensorflow

Neural Network untuk MNIST

import tensorflow as tf

from tensorflow.keras.models import
Sequential

from tensorflow.keras.layers import Dense,
Flatten

from tensorflow.keras.optimizers import
Adam

# Load dataset MNIST

(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.mnist.load_data()

# Normalisasi pixel 0-255 menjadi 0-1 float

x_train, x_test = x_train / 255.0, x_test /
255.0


# Definisikan model sequential

model = Sequential([
```

```
    Flatten(input_shape=(28, 28)),   # 
Flatten 2D gambar ke 1D
    Dense(128, activation='relu'),   # 
Hidden layer 128 neuron, aktivasi ReLU
    Dense(10, activation='softmax')  # 
Output layer 10 neuron, softmax klasifikasi 
10 kelas
])
# Kompilasi model dengan optimizer Adam dan 
loss fungsi cross-entropy
model.compile(optimizer=Adam(),

loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])


# Latih model selama 5 epoch dengan batch 
size default
model.fit(x_train, y_train, epochs=5)


# Evaluasi model pada data test
test_loss, test_acc = 
model.evaluate(x_test, y_test)
print(f'Akurasi pada test set: 
{test_acc:.4f}')
```

Penjelasan Code yang diterapkan :

- *Flatten Layer*: Mengubah input 28x28 pixel menjadi vektor 784 elemen agar bisa diproses layer Dense.
- *Dense Layer*: Lapisan fully-connected neural dengan 128 neuron. Fungsi aktivasi ReLU memperkenalkan non-linearitas.

- *Output Layer*: 10 neuron, satu untuk setiap digit 0-9. Fungsi aktivasi softmax mengubah output menjadi probabilitas kelas.
- *Loss Function: Sparse Categorical Crossentropy* cocok untuk klasifikasi multikelas.
- *Optimizer*: Adam adalah variasi adaptive gradient descent yang efisien dan umum digunakan.

Model simple ini biasanya mencapai akurasi di atas 97% pada dataset MNIST setelah 5 epoch. Ini membuktikan neural network walau sederhana dapat memberikan hasil kuat untuk klasifikasi gambar kecil dan jelas seperti digit tulisan tangan.

Untuk masalah lebih kompleks dan dataset besar, arsitektur bisa diperluas dengan:

- Menambah jumlah dan ukuran hidden layer
- Mengganti Dense layer dengan *Convolutional layers* (CNN) untuk pengolahan citra
- Menggunakan teknik regularisasi seperti dropout untuk menangani overfitting

DAFTAR PUSTAKA

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

Nielsen, M. A. (2015). Neural networks and deep learning. Determination Press. https://neuralnetworksanddeeplearning.com/

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533–536. https://doi.org/10.1038/323533a0

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.

Chollet, F. (2018). Deep learning with Python. Manning Publications.

Abadi, M., et al. (2016). TensorFlow: Large-scale machine learning on heterogeneous systems. arXiv preprint arXiv:1603.04467.

BAB 10

FRAMEWORK AI : *TENSORFLOW*, KERAS, DAN PYTORCH

10.1 Apa yang Dimaksud dengan Framework

Dalam pengembangan AI dan ML, framework adalah pustaka (*library*) perangkat lunak yang menyediakan abstraksi tinggi atas operasi matematika, optimisasi, dan komputasi untuk neural network dan algoritma pembelajaran mesin. Tanpa framework, seorang pengembang harus menulis ulang banyak bagian fundamental seperti diferensiasi otomatis (*automatic differentiation*), manajemen graf komputasi, dan optimisasi perangkat keras. Framework memudahkan proses ini, memungkinkan fokus lebih pada arsitektur model, eksperimen, dan deployment.

Framework AI modern memungkinkan:

a. Abstraksi matematika: Tensor, tensor operasi, backpropagation, dan optimisasi tidak perlu diimplementasikan dari nol.

b. Eksekusi efisien: *Framework* dapat menjalankan operasi di CPU, GPU, atau TPU dengan paralelisasi dan optimasi.

c. Modularitas dan penggunaan kembali (*reuse*) : Dengan modul-layer, optimizer, dan loss function yang sudah ada, pengembang bisa menyusun model dengan cepat dan konsisten.

d. Reproduksibilitas dan pengujian : *Loging training, checkpoint,* dan visualisasi (misalnya melalui TensorBoard) mempermudah eksperimen ilmiah dan industri.

10.2 Efisiensi energi menggunakan Framework

Dalam beberapa tahun terakhir, konsep *Green AI* semakin mendapatkan perhatian, Di mana konsumsi energi sangat dipengaruhi oleh backend framework dan bagaimana tensor computation dijalankan pada perangkat keras yang berbeda (Alizadeh & astor, 2024). Framework yang lebih efisien dapat menghasilkan pengurangan konsumsi energi secara signifikan, terutama bila digunakan untuk inference dalam deployment jangka panjang. Hal ini penting untuk aplikasi edge (misalnya pada perangkat IoT) atau skenario produksi skala besar di mana setiap watt menghitung. Pemilihan framework yang "ramah energi" bisa menjadi salah satu pertimbangan utama selain performa murni.

10.3 Manfaat lain *Framework*

Framework AI sangat penting untuk reproduksibilitas riset. Dengan API yang konsisten (seperti Keras), logging dan checkpoint, dan eksekusi deterministik di berbagai perangkat keras, para peneliti dapat:

a. Membagikan skrip pelatihan (training script) dengan rekan kerja dan komunitas.

b. Melacak eksperiment dengan eksperimen hyperparameter, versi framework, dan konfigurasi perangkat keras.

c. Mengulangi hasil penelitian di lingkungan lain, menetapkan baseline yang dapat diverifikasi.

Beberapa manfaat praktis framework dalam konteks tim:

a. Kolaborasi tim: Dengan modul-layer dan API yang jelas, anggota tim (misalnya peneliti, *engineer*, data *scientist*) dapat bekerja bersama lebih mudah.

b. Prototipe cepat: Framework seperti Keras sangat membantu dalam membuat prototipe arsitektur baru secara cepat — sangat berguna dalam riset awal atau proof-of-concept.

c. *Deployment*: Framework lengkap (seperti *TensorFlow*) menyediakan alat untuk deployment ke berbagai lingkungan, misalnya server produksi, mobile, ataupun *edge device.*

Efisiensi ini mempercepat siklus pengembangan, menghemat waktu, dan menurunkan risiko teknis saat memproduksi model AI.

10.4 Resiko Penggunaan *Framework*

Meskipun *framework* memberikan kemudahan, literatur baru juga menunjukkan bahwa framework juga memiliki risiko bug dan keandalan yang perlu dicermati, misalnya dalam studi *On Reporting Performance and Accuracy Bugs for Deep Learning Frameworks* (Long & Chen, 2022), para peneliti menganalisis lebih dari 600 laporan bug di berbagai framework populer (termasuk TensorFlow, PyTorch, Keras) di GitHub. Mereka menemukan bahwa bug performa sering disebabkan oleh "kecepatan rendah (low speed)" dalam pelatihan, sedangkan bug akurasi tidak memiliki pola yang konsisten. Banyak laporan bug menyebutkan masalah selama fase training, tetapi hanya sebagian kecil akhirnya diperbaiki langsung melalui patch. Ada juga analisis *multi-language bugs* (Li et al., 2023) yang menunjukkan bahwa framework seperti TensorFlow dan PyTorch, yang menggunakan kombinasi

bahasa pemrograman (seperti Python dan C/C++), rentan terhadap bug yang kompleks dan sulit diperbaiki karena perbedaan bahasa dan arsitektur. Di samping itu itu, riset *Fuzzing Automatic Differentiation in Deep-Learning Libraries* (Yang, Deng, Yao, dkk., 2023) memperlihatkan bahwa komponen *automatic differentiation* (AD) dalam banyak framework masih mengandung bug numerik.

10.5 Sejarah dan latar belakang pengembangan *Framework TensorFlow*

TensorFlow merupakan salah satu tonggak penting dalam perkembangan kecerdasan buatan modern. Framework ini tidak hanya mendominasi ekosistem machine learning di industri, tetapi juga menjadi dasar dari banyak inovasi besar dalam deep learning. Sejak diperkenalkan pada tahun 2015 oleh Google Brain Team, TensorFlow telah menjadi standar de facto dalam membangun model berskala besar, mengelola komputasi paralel, dan melakukan deployment model ke berbagai platform mulai dari server, perangkat mobile, hingga browser web.

Framework ini dirancang untuk menangani komputasi numerik berskala besar dan mendukung berbagai aplikasi machine learning seperti:

a. pengenalan suara,

b. pengenalan gambar,

c. rekomendasi konten,

d. pemahaman bahasa alami,

e. serta analisis data berskala besar.

Dalam perkembangannya menuju TensorFlow 2.x (dirilis 2019 dan terus diperbarui sampai 2025), framework ini mengalami transformasi besar:

a. berpindah dari static computation graph ke eager execution,
b. integrasi penuh dengan Keras sebagai high-level API,
c. peningkatan kompatibilitas dengan NumPy,
d. peningkatan pipeline produksi melalui ekosistem lengkap seperti TFX, TensorFlow Lite, dan TensorFlow Serving.

10.5.1 Arsitektur TensorFlow

TensorFlow memiliki empat komponen inti:

1. Tensor

   "Tensors" adalah representasi data multidimensi yang menjadi dasar seluruh operasi matematika dalam TensorFlow. Setiap tensor memiliki:

   a. rank (dimensi),
   b. *shape*,
   c. data *type*.

2. *Computation Graph*

   Pada awalnya TensorFlow menggunakan static computation graph, yaitu graf yang mendefinisikan seluruh operasi sebelum dieksekusi. Mulai TensorFlow 2.x, paradigma ini diganti oleh eager execution, namun graf tetap tersedia ketika pengguna menjalankan model dalam mode tf.function, yang mengkonversi operasi menjadi graf yang dapat dioptimalkan lebih lanjut. Manfaat graf TensorFlow:

   a. optimisasi automatic parallelism,
   b. eksekusi lebih cepat di GPU/TPU,
   c. optimasi memory planning,
   d. portabilitas ke platform lain.

3. *Automatic Differentiation System*

   TensorFlow menyediakan autodiff melalui tf.GradientTape. Sistem ini melacak seluruh operasi tensor dan secara otomatis menghitung gradient untuk proses training model neural network.

4. Runtime Eksekusi (*Device Abstraction Layer*)

   TensorFlow secara otomatis menentukan perangkat optimal untuk menjalankan operasi:

   a. CPU (komputasi standar),
   b. GPU (komputasi paralel masif),
   c. TPU (chip khusus yang dikembangkan Google untuk training skala besar).

Melalui abstraksi perangkat ini, pengguna dapat menulis kode yang sama tanpa menyesuaikan driver atau modul perangkat keras secara manual.

10.5.2 Fitur-Fitur Utama Transflow Modern

TensorFlow banyak digunakan pada aplikasi:

1. Computer Vision

   Digunakan untuk object detection, face recognition, dan semantic segmentation. Model seperti EfficientNet dan MobileNet dikembangkan di ekosistem TensorFlow.

2. Natural *Language Processing*

   Google menggunakan TensorFlow untuk melatih model seperti:

   a. Transformer,
   b. BERT (*Bidirectional Encoder Representations from Transformers*),
   c. T5 (Text-To-Text T*ransfer Transformer*).

10.6 *Framework* Keras

Keras adalah high-level neural network API yang dirancang untuk memudahkan pembuatan dan eksperimen model deep learning. Keras diperkenalkan oleh François Chollet (*Google Engineer*) dan pertama kali dirilis pada tahun 2015. Filosofi utama di dalamnya adalah:

a. User-friendly

b. Modular dan fleksibel

c. Mudah di-extend

d. Cepat dalam prototipe

Sejak TensorFlow 2.0 (2019) dan terus hingga 2025, Keras menjadi bagian inti dari TensorFlow. Keras sangat populer di dunia akademik maupun industri karena kesederhanaan sintaks dan kemampuannya membuat prototipe cepat. Banyak peneliti menggunakan Keras untuk experimentation awal sebelum memindahkan model ke framework tingkat rendah.

10.6.1 Arsitektur Keras

Keras terdiri dari 3 inti yaitu :

1. *Simplicity / User Experience* (UX)

   Keras menyediakan abstraksi tinggi sehingga peneliti dapat fokus pada desain model, bukan detail teknis:

   a. Pembuatan layer sangat intuitif.
   b. Hanya beberapa baris kode untuk membuat model deep learning.
   c. Error message lebih mudah dipahami daripada framework lain.

Keras dirancang untuk meminimalkan *cognitive load* bagi pengguna —sebuah faktor penting bagi pemula (Chollet, 2021).

2. Modular / *Modularity*, yang terdiri dari :
   a. Layers, merupakan blok bangunan utama model, memiliki sifat di mana setiap layer adalah objek yang terpisah, dapat disusun ulang, dan dapat diganti tanpa mengubah arsitektur lain.
   b. Models, modul yang menyediakan dua pendekatan modular yaitu :
   c. *Sequential* API, bersifat linear dan sederhana, cocok untuk model berlapis lurus.
   d. *Funcational* API, modul yang mendukung multi-input, multi-output, dan shared layer. Layer ini dapat digunakan ualng (*reusable component*)
   e. *Optimizers*, modul ini dapat mengatur cara model belajar dan dapat diganti kapan saja tanpa mengubah layer atau data.
   f. *Loss Functions*, merupakan modul yang teripsah dari model
   g. *Metrics*, modul yang hanya digunakan untuk evaluasi, tidak mempengaruhi training.
   h. *Callbacks*, merupakan modul kendali proses training.
3. *Extensibility*, di mana Keras dapat diperluas melalui:
   a. custom loss
   b. custom layer
   c. custom training loop

d. *model subclassing*

10.6.2 Kelebihan Keras

Beberapa kelebihan dari framework keras antara lain :

1. Cocok untuk pemula

   Sintaksnya ramah bagi mahasiswa dan engineer baru.

2. Cepat dalam prototipe

   Pembuatan model cepat tanpa konfigurasi rumit.

3. Integrasi kuat dengan TensorFlo

   Mendukung GPU/TPU otomatis.

4. Dokumentasi lengkap dan stabil

   Dokumentasi Keras dianggap salah satu yang terbaik.

5. Skala dari riset kecil → produksi besar

   Berbeda dari PyTorch yang awalnya lebih riset, Keras sangat kompatibel dengan pipeline produksi.

10.7 *Framework* PyTorch

10.7.1 Pendahuluan: PyTorch dan Posisinya dalam Ekosistem AI Modern

PyTorch adalah framework deep learning yang dikembangkan oleh *Facebook AI Research* (FAIR) dan dirilis secara *open-source* pada tahun 2016. PyTorch dengan cepat menjadi framework favorit komunitas riset karena:

a. fleksibilitas tinggi,
b. eksekusi dinamis (*dynamic computation graph*),

c. kemudahan debugging,
d. integrasi kuat dengan Python.

Sejak 2020 hingga 2025, PyTorch mendominasi publikasi ilmiah dan menjadi standar de facto untuk penelitian cutting-edge seperti:

a. GPT-series dan t*ransformer modern,*
b. stable diffusion dan *model generatif,*
*c. graph neural networks (*GNN*),*
d. model *reinforcement learning* canggih.

10.7.2 Filosofi dan Arsitektur Framework PyTorch

PyTorch memiliki filosofi utama:

1. Pythonic: bekerja seperti Python biasa.
2. Dynamic: computation graph dibuat saat runtime.
3. Flexible: cocok untuk eksperimen model baru.
4. Transparent: debugging mudah dengan print() atau breakpoint.

Framework PyTorch ini memiliki beberapa komponen inti yaitu :

1. Tensor, objek tensor PyTorch mirip dengan NumPy namun memiliki kelebihan yaitu dapat dijalankan di CPU/GPU, diberi gradient, dan digunakan dalam operasi matematis tinggi.
2. Autograd (*Automatic Differentation*) modul paling penting di PyTorch yang dapat melacak operasi Tensor, menghasilkan computation graph dinamis, serta menghitung gradient otomatis melalui backprogagation.
3. Torch.nn, modul yang berisi layer neural network, loss function, dan komponen model lain.
4. Optim, berisi algoritma seperti Adam, SGD dan RMSprop yang merupakan optimizer dalam

machine learning/ *deep learning* untuk mengoptimalkan bobot (*weights*) dan bias pada model neural network agar nilai loss (kesalahan prediksi) semakin kecil.

5. TorchScript, digunakan untuk konversi model PyTorch ke bentuk statis agar lebih cepat untuk deployment.
6. TorchVision, TorchAudio, dan TOrchText, merupakan leibrary yang pendukung untuk computer vision, audio processing, dan NLP

10.7.3 Keunggulan Khas Framework PyTorch

Salah satu alasan utama PyTorch populer dalam riset adalah *dynamic computation graph* (DCG). Berbeda dari static graph TensorFlow 1.x (yang kini sudah digantikan *eager execution*), PyTorch membuat graph secara real-time saat program berjalan.

Keuntungannya:

1. Sangat fleksibel untuk model kompleks
   Ideal untuk RNN non-standar, model recursive, model adaptive, dan arsitektur eksperimental.
2. Debugging mudah
   Bisa menggunakan print(), pdb, atau IDE debugger seperti debugging Python biasa.
3. Lebih intuitif bagi peneliti
   Alur model mengikuti alur eksekusi kode, bukan graph yang dituliskan sebelumnya.

10.7.4 PyTorch dalam Industri

Walaupun dominan di riset, sejak 2021 PyTorch juga berkembang sebagai framework industri melalui:

a. PyTorch Lightning high-level training framework yang mempermudah manajemen loop training.

b. PyTorch 2.0 (2023) memperkenalkan compiler *Dynamo*, *Inductor*, dan optimasi tingkat tinggi.

c. TorchServe server untuk deployment model produksi.

d. Mobile Deployment PyTorch Mobile dan ONNX Runtime.

10.7.5 Tantangan dan Kekurangan Framework PyTorch

Sebagai framework, PyTorch memiliki beberapa kekurangan antara lain

1. Bug Framework yang Kompleks

   Studi *An Empirical Study on Bugs Inside PyTorch* (Ho et al., 2023) menunjukkan:

   a. PyTorch mengandung *ratusan bug*, termasuk memory leak, shape mismatch, dan kesalahan gradient.

   b. 27% bug berhubungan dengan dynamic graph.

   c. Banyak bug berada di implementasi C++ yang membungkus operasi GPU.

2. Silent Bugs dalam Kode Pengguna

   Penelitian oleh Shinhwei et al. (2024) menemukan bahwa pengguna PyTorch rentan pada:

   a. missing operations,

   b. gradient yang salah karena variable detach(),

   c. tensor operation yang tidak kompatibel,

   d. shape mismatch yang tidak terdeteksi. Hal ini sangat umum pada proyek riset dengan model kompleks.

3. Deployment Masih Kurang Matang Dibanding TensorFlow

   Beberapa studi industri menunjukkan bahwa:

   a. TensorFlow (dengan TFX) lebih baik untuk pipeline produksi,

   b. PyTorch lebih unggul dalam model development, bukan deployment.

4. Konsumsi Energi

   Dalam studi *Green AI* (Alizadeh & Castor, 2024):

   a. PyTorch sering lebih boros energi daripada TensorFlow pada model vision,

   tetapi lebih efisien pada beberapa workload NLP.

Ini penting untuk perusahaan yang menjalankan model besar secara berkelanjutan.

Tabel 10. 1 Perbandingan PyTorch dan TensorFlow / Keras

Aspek	PyTorch	TensorFlow/Keras
Fleksibilitas Riset	**Sangat unggul** (Ho 2023)	Baik
Deployment Industri	Baik (TorchServe)	Unggul (TFX, TF Serving)
Dynamic Graph	Unggul	Ada, tetapi kurang fleksibel untuk riset
Efisiensi Energi	Tergantung workload (Alizadeh 2024)	Stabil
Silent Bugs	Tinggi (Shinhwei 2024)	Tinggi (Tambon 2023)

Aspek	PyTorch	TensorFlow/Keras
Kemudahan Pemula	Menengah	Sangat tinggi (Keras)

10.8 *Framework* Apa yang Digunakan ?

Memahami karakteristik TensorFlow, Keras, dan PyTorch bukanlah tujuan akhir. Langkah terpenting adalah mengetahui kapan dan dalam konteks apa masing-masing framework tepat digunakan. Sebuah framework yang sangat unggul dalam riset mungkin tidak optimal dalam produksi industri, demikian pula framework yang efisien di perangkat mobile mungkin tidak ideal untuk riset model baru.

Sub bab ini memberikan analisis mendalam berdasarkan:

a. kebutuhan pengguna (pemula, engineer, peneliti, industri),
b. kompleksitas model,
c. lingkungan komputasi,
d. performa dan efisiensi energi,
e. pipeline produksi,
f. serta temuan penelitian 5 tahun terakhir terkait reliabilitas dan bug framework.

10.8.1 Memahami Kriteria Pemilihan Framework

Pemilihan framework dipengaruhi oleh enam faktor utama:

1. Tujuan Penggunaan
    a. Belajar / edukasi
    b. Riset dan eksperimen
    c. Prototipe cepat
    d. *Deployment* skala besar
    e. *Mobile / edge computing*

2. Kompleksitas Arsitektur Model

   Model sederhana misalnya model CNN dasar, MLP
   Model kompleks misalnya model Transformer, Diffusion, GNN, Meta-learning

3. Kinerja Komputasi dan Energi

   Apakah model dijalankan pada GPU/TPU atau perangkat edge?

4. Ketergantungan Ekosistem

   Apakah memerlukan TFX, TensorBoard, TorchScript, atau Keras Core?

5. Risiko Bug & *Reproducibility*

   Framework yang fleksibel sering lebih rawan bug (Ho et al., 2023). Framework high-level rawan *silent bugs* (Tambon et al., 2023).

6. Dukungan Komunitas & Dokumentasi

   Ini sangat memengaruhi pembelajaran dan troubleshooting.

10.8.2 Kapan sebaiknya menggunakan TensorFlow?

TensorFlow adalah framework "industry-grade" yang dirancang untuk skala besar, eksekusi teroptimasi, dan deployment jangka panjang.

Cocok digunakan ketika:

1. Mengembangkan sistem produksi skala besar (*enterprise-grade*)

   Penelitian industri (Google Research, 2022–2024) menegaskan bahwa

   1. TensorFlow unggul untuk:

a. sistem rekomendasi real-time,
b. pipeline Machen Learning dengan validasi otomatis,
c. deployment ke server besar atau cluster,
d. aplikasi yang membutuhkan reliability jangka panjang.

2. TensorFlow Lite adalah framework paling matang untuk:
   a. Android, iOS, ARM CPUs
   b. Raspberry Pi, Coral Edge TPU
   c. perangkat IoT dengan RAM rendah

   Studi *Energy Index for Deep Learning Models* (Aquino-Brítez et al., 2025) menunjukkan bahwa TF Lite menunjukkan efisiensi energi paling stabil untuk model lightweight dibandingkan PyTorch Mobile.

2. Training model besar dengan TPU
   TensorFlow adalah framework native untuk TPU dan masih paling stabil hingga 2025. Jika HPC dan efisiensi training menjadi prioritas, TensorFlow lebih unggul.
3. Jika membutuhkan workflow yang lengkap & terstandar
   TensorFlow menyediakan:
   a. *TensorBoard* (visualisasi training),
   b. tf.data (pipeline data efisien),
   c. tf.*distribute* (*distributed training*).

Kapan sebaiknya tidak menggunakan TensorFlow ?

a. Jika Anda pemula dan membutuhkan sintaks sangat sederhana maka lebih baik Keras.

b. Jika Anda peneliti yang sering membuat arsitektur baru maka PyTorch lebih fleksibel.
c. Untuk debugging step-by-step, TensorFlow lebih sulit dibanding PyTorch.

10.8.3 Kapan sebaiknya menggunakan Keras?

Keras adalah framework paling ramah pemula dan ideal untuk rapid prototyping. Ia menyediakan API tingkat tinggi yang abstraktif, sehingga proses pembuatan model cepat dan mudah. Dengan munculnya Keras Core (2023–2025) yang mendukung JAX, PyTorch, dan TensorFlow, penggunaannya makin luas.

Framework Keras tepat digunakan ketika :

1. Anda pemula yang baru belajar *Machine Learning / Deep Learning*

Dalam Buku *Deep Learning with Python* (Chollet, 2021) menegaskan bahwa "Keras didesain untuk human-centered ML — membuat deep learning dapat dipahami pemula tanpa mengorbankan kekuatan teknis". Kelas universitas dunia seperti MIT, Stanford, UIUC, juga menggunakan Keras sebagai framework pengantar.

2. Anda ingin prototipe model dengan cepat
   Keras sangat ideal bila Anda ingin:
   a. membangun model baseline,
   b. bereksperimen dengan arsitektur sederhana,
   c. membuat model untuk riset awal sebelum eksperimen mendalam. Dari studi empiris (Tambon et al., 2023), Keras menurunkan jumlah kesalahan sintaks pemula hingga 40% dibanding TensorFlow dan PyTorch.
3. Anda membangun model standard seperti:
   Keras sangat ideal bila Anda ingin:
   - membangun model baseline,
   - bereksperimen dengan arsitektur sederhana,

- membuat model untuk riset awal sebelum eksperimen mendalam. Dari studi empiris (Tambon et al., 2023), Keras menurunkan jumlah kesalahan sintaks pemula hingga 40% dibanding TensorFlow dan PyTorch.

4. Anda ingin integrasi dengan TensorFlow untuk produksiKeras mempermudah proses koding, sedangkan TensorFlow mempermudah deployment.

Kapan sebaiknya tidak menggunakan framework Keras ?

Keras kurang cocok digunakan untuk :

a. Untuk penelitian kompleks (GNN, *meta-learning*).
b. Untuk model yang memerlukan kontrol penuh atas gradient.
c. Untuk arsitektur non-standar yang butuh dynamic execution seperti PyTorch.

Dalam penelitian Shinhwei et al. (2024) bahwa Keras memiliki sejumlah silent bugs, terutama ketika model terlalu kompleks atau menggunakan kombinasi layer custom.

10.8.4 Kapan menggunakan PyTorch?

PyTorch sebaiknya digunakan ketika :

1. Sedang melakukan riset atau penelitian akademik

Dalam PapersWithCode (2021–2024) menunjukkan bahwa lebih dari **70%** publikasi top-tier (NeurIPS, ICML, ICLR) menggunakan PyTorch. Model SOTA seperti GPT, LLaMA, Diffusion Models, ViT dibuat di PyTorch.

2. Membuat arsitektur baru yang tidak tersedia di API standar
   a. Diffusion models (Stable Diffusion)
   b. Reinforcement Learning dengan *dynamic reward*
   c. Neural ODEs
   d. *Graph Neural Networks* (GNN)
   e. Meta-learning
   f. Multimodal transformer
   g. *Dynamic graph* sangat memudahkan eksperimen ini.
3. butuh debugging yang cepat dan intuitif
4. ingin memanfaatkan PyTorch 2.0 Compiler Stack seperti TorchDynamo, AOTautograd, TorchInductor.

Framework PyTorch kurang cocok bila digunakan dalam kondisi berikut :

a. Kurang optimal untuk dployment mobile (PyTorch Mobile masih teritinggal dari TF Lite)
b. Produksi pipeline end-to-end lebih lemah dibanding TFX
c. Konsumsi energi cenderung lebih tinggi di beberapa workload (Alizadeh & Castor, 2024)

10.9 Menghubungkan kebutuhan dengan *Framework*

Tabel 10. 2 Menghubungkan Kebutuhan Framework

Kebutuhan	Framework Paling Sesuai	Alasan Modern (2020–2025)
Belajar dasar deep learning	Keras	API mudah, dokumentasi kuat
Prototipe cepat	Keras	Sedikit kode, cepat iterasi
Riset akademik	PyTorch	Dynamic graph, fleksibel
Eksperimen arsitektur baru	PyTorch	Mudah modifikasi model
Deploy ke mobile/IoT	TensorFlow Lite	Paling efisien di edge device
Pipeline produksi enterprise	TensorFlow / TFX	Stabil, mature, skala besar
Komputasi TPU besar	TensorFlow	Native TPU support
Model generatif modern	PyTorch	Ekosistem SOTA berbasis PyTorch
KIHL (komputasi hemat energi)	TensorFlow	Konsumsi energi lebih stabil
Model vision klasik	Keras / TensorFlow	Ada banyak pretrained models
Model NLP transformer	PyTorch	HuggingFace ecosystem

DAFTAR PUSTAKA

Alizadeh, N., & Castor, F. 2024. "Green AI: Energy Consumption in Deep Learning Frameworks." *Journal of Systems and Software*.

Aquino-Brítez, S., Pérez, R., & Benítez, C. 2025. "Energy Consumption Index for Deep Learning Models".

Ho, S. C. Y., Wang, X., & Chen, L. 2023. "An Empirical Study on Bugs Inside PyTorch: A Replication Study." *Empirical Software Engineering.*

Tambon, F., et al.2023. "Silent Bugs in TensorFlow and Keras: An Empirical Study." *IEEE Transactions on Software Engineering*.

Shinhwei, H., et al. 2024. "Silent Bugs in Keras and PyTorch Programs." *Journal of Software: Evolution and Process*.

Li, Z., Wang, M., & Zhang, T. (2023). "Understanding Bugs in Multi-language Deep Learning Frameworks." *ACM Transactions on Software Engineering and Methodology*.

Meta AI Research. 2023. *PyTorch 2.0 Technical Report*. Meta AI / FAIR.

Google Research / Google Brain. (2022–2024). *TensorFlow Extended (TFX) Whitepapers*. Google Research.

Yang, C., et al. 2023. "Fuzzing Automatic Differentiation in Deep-Learning Libraries." *ACM Transactions on Programming Languages and Systems*.

Long, G., & Chen, T. 2022. "Performance and Accuracy Bugs in Deep Learning Frameworks." *Journal of Machine Learning Research.*

BAB 11

NATURAL LANGUAGE PROCESSING (NLP) DENGAN PYTHON

Natural Language Processing (NLP) merupakan cabang Kecerdasan Buatan yang berfokus pada pemrosesan, pemahaman, dan analisis bahasa manusia dalam bentuk teks maupun ucapan menggunakan pendekatan komputasional. Berbeda dengan data terstruktur yang telah dibahas pada bab-bab sebelumnya, data bahasa alami bersifat tidak terstruktur, ambigu, serta sangat dipengaruhi oleh konteks linguistik dan situasional. Karakteristik ini menjadikan NLP sebagai salah satu bidang yang paling menantang dalam Kecerdasan Buatan, sekaligus strategis karena kemampuannya menjembatani interaksi antara manusia dan sistem komputasi secara lebih natural.

Dalam beberapa tahun terakhir, NLP mengalami perkembangan yang sangat pesat seiring dengan kemajuan algoritma machine learning, deep learning, serta meningkatnya ketersediaan data teks dalam skala besar. NLP tidak lagi terbatas pada penelitian akademik, tetapi telah menjadi komponen inti dalam berbagai produk teknologi yang digunakan secara luas. Beberapa produk NLP yang terkenal seperti di gambar 1 dan banyak diadopsi antara lain **Google Search** dan **Google Assistant**, yang memanfaatkan NLP untuk memahami maksud pengguna dan menyajikan informasi yang relevan; **ChatGPT** serta model bahasa besar lainnya seperti **GPT-4**, **BERT**, dan **T5**, yang mampu menghasilkan teks alami, menjawab pertanyaan, dan

melakukan dialog (*conversational AI*); serta asisten suara seperti **Siri**, **Alexa**, dan **Cortana**, yang memproses perintah bahasa manusia dalam bentuk lisan. Selain itu, teknologi *machine translation* seperti **Google Translate** menggunakan NLP untuk menerjemahkan teks antarbahasa secara akurat, sementara sistem rekomendasi konten dan analisis opini banyak diterapkan pada media sosial dan platform e-commerce untuk memahami sentimen pengguna serta preferensi konsumen.

Perkembangan aplikasi-aplikasi tersebut didukung oleh kemajuan riset NLP modern, khususnya melalui penerapan arsitektur *Transformer* dan konsep *transfer learning*, yang memungkinkan model bahasa dilatih pada korpus besar dan diadaptasi untuk berbagai tugas spesifik. Di sisi lain, Python berperan sebagai bahasa pemrograman utama yang menjadi fondasi penelitian dan implementasi NLP. Ekosistem pustaka Python yang matang, seperti **NLTK**, **scikit-learn**, **spaCy**, **Gensim**, serta pustaka *deep learning* seperti **TensorFlow** dan **PyTorch**, menjadikan Python sebagai platform yang fleksibel dan powerful untuk mengembangkan sistem NLP, baik pada level pembelajaran dasar maupun riset lanjutan. Kombinasi antara kematangan algoritma dan ekosistem Python ini menjadikan NLP sebagai bidang yang sangat relevan untuk dipelajari oleh mahasiswa, khususnya sebagai dasar penelitian dan penyusunan skripsi berbasis data teks.

Gambar 11. 1 Beberapa Teknologi Berbasis NLP

11.1 Konsep Dasar *Natural Language Processing* (NLP)

Natural Language Processing (NLP) adalah bidang interdisipliner yang berada pada persimpangan ilmu komputer, linguistik, dan kecerdasan buatan, dengan tujuan memungkinkan sistem komputasi untuk memahami, memproses, dan menghasilkan bahasa manusia secara bermakna. Secara konseptual, NLP berupaya menjembatani perbedaan mendasar antara cara manusia menggunakan bahasa—yang kaya konteks, fleksibel, dan sering kali tidak eksplisit—dengan cara mesin memproses informasi yang bersifat formal, eksplisit, dan berbasis simbol atau numerik. Oleh karena itu, NLP tidak hanya berfokus pada pemrosesan teks secara teknis, tetapi juga pada pemodelan makna, struktur, dan penggunaan bahasa dalam konteks tertentu.

11.2 Masalah Pemrosesn Bahasa Alami

Bahasa alami memiliki sejumlah karakteristik yang menimbulkan tantangan signifikan dalam pemrosesan komputasional. Salah satu masalah utama adalah **ambiguitas**, baik pada tingkat kata (*lexical ambiguity*), frasa, maupun kalimat, di mana sebuah kata atau ungkapan dapat memiliki lebih dari satu makna tergantung pada konteks penggunaannya. Fenomena ini telah lama menjadi perhatian utama dalam penelitian NLP karena menyulitkan proses pemetaan bahasa ke representasi formal yang konsisten (Daniel Jurafsky & James H. Martin, 2025). Selain itu, bahasa alami bersifat **kontekstual**, di mana makna suatu ujaran sering kali tidak dapat ditentukan hanya dari kata-kata yang digunakan, tetapi juga dipengaruhi oleh situasi komunikasi, tujuan penutur, serta latar belakang pengetahuan bersama antara penutur dan pendengar (Christopher D. Manning et al., 2009).

Masalah lain yang penting adalah **variasi bahasa**, termasuk perbedaan gaya penulisan, penggunaan bahasa informal, singkatan, dan campuran bahasa, khususnya pada data yang bersumber dari media sosial. Struktur kalimat yang tidak selalu mengikuti kaidah tata bahasa formal juga menjadi tantangan tersendiri. Literatur NLP menekankan bahwa kompleksitas ini membuat bahasa alami sulit direpresentasikan secara langsung dalam bentuk yang dapat diproses mesin tanpa tahapan normalisasi dan pemodelan yang tepat (Daniel Jurafsky & James H. Martin, 2025; Khurana et al., 2017).

11.3 Ekosistem Python Untuk NLP

Python telah berkembang menjadi bahasa pemrograman utama dalam penelitian dan pengembangan Natural Language Processing (NLP). Popularitas Python

dalam bidang ini tidak terlepas dari kemudahan sintaks, sifatnya yang terbuka (*open source*), serta dukungan komunitas riset dan industri yang sangat luas. Python memungkinkan peneliti dan pengembang untuk memfokuskan perhatian pada perancangan algoritma dan analisis data bahasa, tanpa terbebani oleh kompleksitas teknis bahasa pemrograman tingkat rendah. Keunggulan utama Python dalam NLP terletak pada **ekosistem pustaka (*library*)** yang kaya, stabil, dan saling terintegrasi. Pustaka-pustaka ini mencakup seluruh tahapan NLP, mulai dari pemrosesan linguistik dasar hingga pemodelan bahasa tingkat lanjut. Beberapa library penting yang umum digunakan dalam NLP antara lain:

1. **NLTK (*Natural Language Toolkit*)**.

   NLTK merupakan salah satu pustaka NLP paling awal dan paling banyak digunakan dalam konteks akademik dan pembelajaran. NLTK menyediakan berbagai fungsi dasar NLP, seperti tokenisasi kata dan kalimat, *stopword removal*, stemming, lemmatization, *part-of-speech tagging*, serta akses ke berbagai korpus linguistik. Karena sifatnya yang modular dan edukatif, NLTK sangat cocok digunakan untuk memahami konsep-konsep fundamental NLP dan sebagai sarana eksplorasi awal dalam penelitian berbasis teks.

2. **spaCy**

   spaCy dirancang sebagai pustaka NLP berperforma tinggi yang ditujukan untuk kebutuhan industri dan aplikasi skala besar. Berbeda dengan NLTK yang berorientasi pembelajaran, spaCy menekankan efisiensi, kecepatan pemrosesan, dan integrasi dengan sistem produksi. spaCy menyediakan fitur seperti *named entity recognition*, dependency parsing, dan pipeline NLP terintegrasi yang siap digunakan pada data nyata.

3. scikit-learn

scikit-learn berperan penting dalam menghubungkan NLP dengan Machine Learning. Pustaka ini menyediakan berbagai teknik ekstraksi fitur teks, seperti *Bag of Words* dan *Term Frequency–Inverse Document Frequency (TF-IDF)*, serta algoritma supervised learning yang umum digunakan dalam NLP, seperti Naive Bayes, Support Vector Machine, dan Logistic Regression. Dalam konteks pendidikan dan penelitian dasar, scikit-learn sering digunakan untuk membangun model klasifikasi teks yang transparan dan mudah dievaluasi.

4. Gensim

Gensim merupakan pustaka yang berfokus pada pemodelan semantik dan representasi vektor kata. Library ini banyak digunakan untuk *topic modeling* (misalnya Latent Dirichlet Allocation) dan pembelajaran *word embedding* seperti Word2Vec dan FastText. Gensim memungkinkan eksplorasi makna laten dalam kumpulan dokumen besar, sehingga sering dimanfaatkan dalam analisis tematik dan eksplorasi korpus teks.

11.4 Tahapan NLP

Tahapan NLP disusun sebagai sebuah **pipeline** yang dapat langsung dijadikan kerangka metodologi penelitian atau sering dikenal juga sebagai *backbone* eksperimen. Pipeline ini menekankan bahwa performa sistem NLP tidak hanya ditentukan oleh algoritma klasifikasi, tetapi juga oleh kualitas data, prapemrosesan, serta representasi fitur (Christopher D. Manning et al., 2009; Sebastiani, 2002). Kerangka pipeline berikut selaras dengan praktik umum dalam literatur NLP modern, mulai dari pengumpulan data

hingga evaluasi dan interpretasi (Daniel Jurafsky & James H. Martin, 2025; Khurana et al., 2017).

11.4.1 Akuisisi dan Pemahaman Data Teks

Tahap awal adalah memastikan **sumber data**, **tujuan analisis**, serta **label/kelas** (jika menggunakan supervised learning). Untuk studi kasus analisis sentimen, data biasanya berupa teks pendek yang telah diberi label *positif* atau *negatif* (Pang & Lee, 2008). Pada skala penelitian, data dapat berasal dari ulasan produk, komentar media sosial, berita daring, atau dokumen organisasi.

Contoh implementasi (membuat dataset kecil):

```
import pandas as pd


data = [
("produk ini bagus banget, kualitasnya
mantap", "pos"),
("pengiriman cepat dan kemasan rapi",
"pos"),
("saya puas, sesuai deskripsi", "pos"),
("parah, barang rusak dan mengecewakan",
"neg"),
("tidak sesuai deskripsi, sangat kecewa",
"neg"),
("pengiriman lama, pelayanan buruk",
"neg"),
]


df = pd.DataFrame(data, columns=["text",
"label"])
```

```
print(df)
```

11.4.2 Prapemrosesan Teks (*Text Processing*)

Tahap prapemrosesan teks bertujuan untuk menormalkan dan membersihkan data bahasa alami agar lebih konsisten dan informatif sebelum direpresentasikan dalam bentuk numerik. Bahasa alami mentah mengandung banyak variasi penulisan, simbol, dan kata-kata yang tidak berkontribusi langsung terhadap proses pembelajaran model. Oleh karena itu, tahap ini menjadi krusial dalam pipeline NLP. Tahap ini umumnya mencakup: *case folding*, pembersihan karakter, tokenisasi, stopword removal, serta stemming atau lemmatization (Jurafsky & Martin, 2025). Pemilihan teknik prapemrosesan perlu disesuaikan dengan sumber data (misalnya teks media sosial cenderung lebih informal dibanding berita).

Contoh implementasi (NLTK: tokenisasi + *stopword removal + stemming*):

```
import re
import nltk
nltk.download('stopwords')


from nltk.corpus import stopwords
from Sastrawi.Stemmer.StemmerFactory import
StemmerFactory


stop_id =
set(stopwords.words('indonesian'))


factory = StemmerFactory()
stemmer = factory.create_stemmer()
```

```
def simple_tokenize(text):
 return re.findall(r"[a-zA-Z]+", text)


def preprocess(text: str) -> str:
text = text.lower()
tokens = simple_tokenize(text)
tokens = [t for t in tokens if t not in
stop_id and len(t) > 2]
tokens = [stemmer.stem(t) for t in tokens]
   return " ".join(tokens)


df["text_clean"] =
df["text"].apply(preprocess)
print(df[["text", "text_clean"]])
```

```
                                      text  \
0  produk ini bagus banget, kualitasnya mantap
1           pengiriman cepat dan kemasan rapi
2                saya puas, sesuai deskripsi
3        parah, barang rusak dan mengecewakan
4       tidak sesuai deskripsi, sangat kecewa
5            pengiriman lama, pelayanan buruk

                          text_clean
0  produk bagus banget kualitas mantap
1              kirim cepat kemas rapi
2               puas sesuai deskripsi
3           parah barang rusak kecewa
4             sesuai deskripsi kecewa
5                  kirim layan buruk
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Gambar 11. 2 Output Prapemrosesan Teks

- *Library* dan modul yang digunakan

  Modul re digunakan untuk memanipulasi teks berbasis pola (*pattern matching*), khususnya dalam menghapus karakter non-alfabet. Pustaka nltk

digunakan untuk menyediakan resource linguistik dasar seperti daftar *stopwords*. Proses stemming bahasa Indonesia dilakukan menggunakan pustaka Sastrawi.

- Inisialisasi Stopword dan Stemmer

```
stop_id                                    =
set(stopwords.words('indonesian'))

factory = StemmerFactory()
stemmer = factory.create_stemmer()
```

Daftar *stopwords* bahasa Indonesia dimuat menggunakan modul stopwords dari pustaka NLTK dan disimpan dalam struktur data set untuk mempercepat proses pencarian kata saat tahap *stopword removal*. Penggunaan struktur set memungkinkan proses penyaringan token dilakukan secara efisien karena memiliki kompleksitas pencarian yang lebih rendah dibandingkan struktur data list.

Proses stemming dilakukan menggunakan Sastrawi Stemmer, yaitu pustaka yang secara khusus dirancang untuk menangani morfologi bahasa Indonesia. Sastrawi mampu menghapus berbagai imbuhan bahasa Indonesia, seperti prefiks (*me-*, *di-*, *ke-*), sufiks (*-kan*, *-an*), dan konfiks, sehingga menghasilkan bentuk kata dasar yang lebih akurat dibandingkan stemmer generik. Dengan menggunakan Sastrawi, ualitas representasi fitur teks dapat meningkat karena variasi bentuk kata berkurang dan menjadi lebih konsisten.

- Tokenisasi Kata

```
tokens = re.findall(r"[a-zA-Z]+", text)
```

Tokenisasi dilakukan untuk memecah teks menjadi unit kata (*token*). Pada implementasi ini, tokenisasi dilakukan menggunakan re.findall() agar tidak bergantung pada resource tokenisasi NLTK yang kadang berbeda antar versi. Pola [a-zA-Z]+ akan mengekstrak rangkaian huruf alfabet sebagai token.

Contoh:

```
"produk ini sangat bagus" →
["produk", "ini", "sangat", "bagus"]
```

- *Case Folding*

```
text = text.lower()
```

Tahap case folding bertujuan untuk menormalkan teks dengan mengubah seluruh karakter menjadi huruf kecil. Langkah ini penting untuk menghindari perbedaan representasi kata akibat variasi kapitalisasi, misalnya kata *"Produk"* dan *"produk"* yang secara semantik identik tetapi dapat dianggap berbeda oleh model.

Contoh:

```
"Pengiriman Cepat" → "pengiriman cepat"
```

- Pembersihan Karakter Non-Alfabet

```
text = re.sub(r"[^a-zA-Z ]", " ", text)
```

Pada tahap ini dilakukan pembersihan karakter non-alfabet, seperti angka, tanda baca, dan simbol khusus, menggunakan *regular expression*. Pola [^a-zA-Z] berarti seluruh karakter selain huruf dan spasi akan dihapus. Tujuan tahap ini adalah mengurangi *noise* yang tidak memberikan kontribusi semantik pada analisis teks.

Contoh:

```
"produk bagus!!! kualitas 100%" →
"produk bagus kualitas"
```

- Stopword Removal

```
tokens = [t for t in tokens if t not
in stop_id and len(t) > 2]
```

Tahap stopword removal bertujuan menghapus kata-kata umum yang sering muncul tetapi tidak membawa makna diskriminatif, seperti *"dan"*, *"yang"*, atau *"ini"*. Selain itu, token dengan panjang kurang dari atau sama dengan dua karakter juga dihapus karena umumnya tidak bermakna secara semantik.

Contoh:

```
["produk", "ini", "sangat", "bagus"] →
["produk", "sangat", "bagus"]
```

- Stemming

```
tokens = [stemmer.stem(t) for t in
tokens]
```

Tahap stemming dilakukan untuk mengubah kata ke bentuk dasarnya dengan menghilangkan imbuhan. Proses ini bertujuan mengurangi variasi morfologis kata sehingga ruang fitur menjadi lebih sederhana dan kompak, serta membantu model mengenali kata yang semakna sebagai fitur yang sama.

Contoh:

```
"pengiriman", "pelayanan", "memuaskan" →
"kirim", "layan", "puas"
```

- Rekonstruksi Teks dan Penerapan ke Dataset

```
return " ".join(tokens)

df['text_clean'] =
df['text'].apply(preprocess)
print(df[['text', 'text_clean']])
```

Token hasil prapemrosesan digabung kembali menjadi satu string agar dapat digunakan sebagai input pada tahap representasi fitur, seperti TF-IDF. Fungsi preprocess kemudian diterapkan ke seluruh data teks menggunakan apply(), sehingga dihasilkan kolom baru text_clean yang berisi teks hasil prapemrosesan.
Contoh hasil akhir:

```
"Pengiriman cepat dan pelayanan sangat memuaskan" →
"kirim cepat layan puas"
```

Tahapan pra pemrosesan ini menghasilkan teks yang lebih bersih, ringkas, dan informatif, sehingga siap digunakan pada tahap representasi fitur dan pembelajaran model machine learning.

11.4.3 Representasi Fitur Teks (TF-IDF)

Setelah tahap prapemrosesan, data teks masih berbentuk string, sementara algoritma *machine learning* hanya dapat memproses data numerik. Oleh karena itu, diperlukan tahap representasi fitur untuk mengubah teks menjadi vektor numerik yang merepresentasikan informasi penting dari dokumen. Pada penelitian ini, teknik Term Frequency–Inverse Document Frequency (TF-IDF) digunakan karena terbukti efektif dan stabil sebagai representasi fitur dalam berbagai tugas klasifikasi teks.

```
from sklearn.feature_extraction.text import
TfidfVectorizer


# Inisialisasi TF-IDF Vectorizer
# Menggunakan unigram dan bigram untuk
menangkap konteks sederhana
```

```
vectorizer =
TfidfVectorizer(ngram_range=(1, 2))


# Transformasi teks hasil prapemrosesan
menjadi matriks fitur TF-IDF
X =
vectorizer.fit_transform(df['text_clean'])


# Label kelas untuk supervised learning
y = df['label']


# Informasi hasil representasi fitur
    print("Jumlah dokumen       :",
X.shape[0])
    print("Jumlah fitur (TF-IDF):",
X.shape[1])
    print("\nContoh fitur TF-IDF:")
    print(vectorizer.get_feature_names_out()
[:15])
```

```
Jumlah dokumen      : 6
Jumlah fitur (TF-IDF): 33

Contoh fitur TF-IDF:
['bagus' 'bagus banget' 'banget' 'banget kualitas' 'barang' 'barang rusak'
 'buruk' 'cepat' 'cepat kemas' 'deskripsi' 'deskripsi kecewa' 'kecewa'
 'kemas' 'kemas rapi' 'kirim']
```

Gambar 11. 3 Output Representasi Fitur Teks TF-IDF

- Library dan modul yang digunakan
  `from sklearn.feature_extraction.text import TfidfVectorizer`
  Modul TfidfVectorizer berasal dari pustaka scikit-learn dan digunakan untuk mengubah teks menjadi representasi numerik berbasis TF-IDF.

Modul ini menangani proses pembentukan kosakata (*vocabulary*), perhitungan bobot TF-IDF, serta menghasilkan matriks fitur yang siap digunakan dalam pemodelan supervised.

Metode ***Term Frequency–Inverse Document Frequency*** **(TF-IDF)** dipilih sebagai teknik representasi fitur karena memiliki sejumlah keunggulan dibandingkan representasi teks sederhana seperti *Bag of Words (BoW)*. TF-IDF mampu menekankan kata-kata yang bersifat informatif, yaitu kata yang sering muncul dalam suatu dokumen tetapi relatif jarang muncul pada keseluruhan korpus, sehingga memiliki daya diskriminatif yang lebih tinggi terhadap kelas tertentu.

Sebaliknya, kata-kata umum yang muncul hampir di seluruh dokumen, seperti *produk* atau *barang*, akan memperoleh bobot rendah sehingga pengaruhnya terhadap proses klasifikasi dapat diminimalkan. Karakteristik ini menjadikan TF-IDF sangat efektif dalam mengurangi noise linguistik pada data teks. Selain itu, TF-IDF terbukti cocok untuk data berdimensi tinggi, yang merupakan ciri khas data teks, karena mampu merepresentasikan dokumen dalam ruang fitur besar secara efisien tanpa kehilangan informasi penting. Berbagai penelitian menunjukkan bahwa TF-IDF merupakan representasi fitur yang stabil dan efektif sebagai baseline dalam tugas-tugas NLP klasik, seperti klasifikasi teks dan analisis sentimen, sebelum menerapkan pendekatan berbasis *deep learning* (Salton & Buckley, 1987; Sebastiani, 2002). Studi lanjutan juga menegaskan bahwa kombinasi TF-IDF dengan algoritma supervised learning sederhana, seperti Logistic Regression dan Support Vector Machine, sering menghasilkan performa kompetitif

pada berbagai dataset teks (Christopher D. Manning et al., 2009; Joachims, 2001).

- Inisialiasasi TF-IDF vectorizer

```
vectorizer =
TfidfVectorizer(ngram_range=(1, 2))
```

Pada bagian ini dibuat objek vectorizer menggunakan TfidfVectorizer. Parameter ngram_range=(1, 2) berarti fitur yang dibentuk mencakup:

- unigram (1-gram): satu kata, misalnya *"bagus"*
- bigram (2-gram): dua kata berurutan, misalnya *"sangat kecewa"* atau *"pengiriman cepat"*

Penggunaan bigram membantu model menangkap konteks sederhana yang sering lebih bermakna dibanding kata tunggal.

- Hasil representasi fitur

```
print("Ukuran matriks fitur (X):",
X.shape)
print("Contoh fitur pertama:",
vectorizer.get_feature_names_out()[:15])
```

Bagian ini digunakan untuk memastikan bahwa TF-IDF sudah berhasil membentuk fitur:

- `X.shape` menunjukkan jumlah dokumen dan jumlah fitur
- `get_feature_names_out()` menampilkan daftar kata/bigram yang menjadi fitur

Contoh konseptual transformasi teks
Teks hasil prapemrosesan:
"kirim cepat layan puas"
Setelah TF-IDF:

- direpresentasikan sebagai vektor numerik, misalnya
[0.00, 0.42, 0.00, 0.58, ...]

Setiap nilai merepresentasikan tingkat kepentingan kata terhadap dokumen tersebut dalam keseluruhan korpus.

11.4.4 Pemodelan *Supervised: Logistic Regression*

Model regresi yang digunakan adalah Logistic Regression, yang secara matematis memodelkan probabilitas kelas menggunakan fungsi logistik (*sigmoid*).

Secara teoritis, Logistic Regression memodelkan hubungan antara fitur dan kelas sebagai:

$$P(y = 1|x) = \sigma(w^T x + b)$$

Dimana:

x = vektor fitur TF/IDF

w = bobot regresi (parameter yang dipelajari model)

b = bias

$\sigma(z) = \frac{1}{1+e^{-1}}$ = fungsi sigmoid

```
    from sklearn.model_selection import
train_test_split
    from sklearn.linear_model import
LogisticRegression
    import numpy as np


    # 1. Split data (supervised learning)
    X_train, X_test, y_train, y_test =
    train_test_split(
        X, y,
        test_size=0.3,
```

```
    random_state=42,
    stratify=y)


# 2. Inisialisasi & training model Logistic
Regression
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)


# 3. Prediksi kelas & probabilitas
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)


print("Contoh probabilitas prediksi (5 data
pertama):")
print(y_proba[:5])


# 4. Interpretasi regresi pada teks (kata
dominan)
feature_names =
vectorizer.get_feature_names_out()
coef = model.coef_[0]


top_pos = np.argsort(coef)[-10:]
top_neg = np.argsort(coef)[:10]


print("\nKata paling berkontribusi ke
sentimen POSITIF:")
print(feature_names[top_pos])
```

```
print("\nKata paling berkontribusi ke
sentimen NEGATIF:")
print(feature_names[top_neg])
```

- Split Data

  Tahap ini bertujuan membagi data hasil representasi TF-IDF (X) dan label sentimen (y) ke dalam data latih dan data uji. Data latih digunakan untuk membangun model, sedangkan data uji digunakan untuk mengevaluasi kemampuan generalisasi model terhadap data yang belum pernah dilihat sebelumnya. Parameter stratify=y digunakan untuk menjaga proporsi kelas positif dan negatif tetap seimbang pada kedua subset data, yang penting dalam studi kasus analisis sentimen.
- Inisialisasi dan Training Model Logistic Regression

  ```
  model =
  LogisticRegression(max_iter=1000)
  model.fit(X_train, y_train)
  ```

  Logistic Regression merupakan implementasi resmi model regresi logistik pada pustaka *scikit-learn*. Pada tahap ini, fungsi fit() menjalankan proses optimisasi berbasis *maximum likelihood* untuk mempelajari parameter model, yaitu bobot regresi (w) dan bias (b). Perlu ditekankan bahwa input model bukan teks mentah, melainkan vektor numerik TF-IDF (X_train). Dengan demikian, proses regresi logistik dilakukan setelah teks berhasil diubah menjadi representasi fitur numerik.
- Prediksi kelas dan probabilitas

  ```
  y_pred = model.predict(X_test)
  y_proba = model.predict_proba(X_test)
  ```

```
print(y_proba[:5])
```

Setelah model dilatih, fungsi `predict()` digunakan untuk menghasilkan label kelas sentimen (positif atau negatif) pada data uji. Di balik proses ini, model terlebih dahulu menghitung probabilitas kelas menggunakan fungsi sigmoid, kemudian mengonversinya menjadi label berdasarkan ambang batas (*threshold*) default sebesar 0,5.

Untuk melihat hasil regresi logistik secara eksplisit, fungsi `predict_proba()` digunakan. Fungsi ini menghasilkan probabilitas masing-masing kelas untuk setiap dokumen. Sebagai contoh, keluaran [0.12, 0.88] menunjukkan bahwa sebuah teks memiliki probabilitas 12% sebagai sentimen negatif dan 88% sebagai sentimen positif.

- Interpretasi regresi pada teks

```
feature_names =
vectorizer.get_feature_names_out()
coef = model.coef_[0]

top_pos = np.argsort(coef)[-10:]
top_neg = np.argsort(coef)[:10]

print("Kata paling berkontribusi ke
sentimen POSITIF:")
print(feature_names[top_pos])

print("Kata paling berkontribusi ke
sentimen NEGATIF:")
print(feature_names[top_neg])
```

Salah satu keunggulan Logistic Regression adalah sifatnya yang interpretabel. Setiap fitur TF-IDF (kata atau bigram) memiliki bobot regresi yang tersimpan dalam atribut coef_, sedangkan nilai bias disimpan dalam intercept_. Bobot positif yang besar

menunjukkan bahwa suatu kata mendorong prediksi ke kelas positif, sedangkan bobot negatif yang besar menunjukkan kecenderungan ke kelas negatif.

Melalui proses pengurutan bobot regresi, dapat diidentifikasi kata-kata yang paling dominan dalam menentukan sentimen. Analisis ini memberikan pemahaman linguistik terhadap keputusan model dan membantu menjelaskan alasan di balik hasil klasifikasi, sehingga sangat bermanfaat dalam konteks pembelajaran dan penelitian NLP.

```
Contoh probabilitas prediksi (5 data pertama):
[[0.49293641 0.50706359]
 [0.48103371 0.51896629]]

Kata paling berkontribusi ke sentimen POSITIF:
['bagus banget' 'kirim' 'kirim cepat' 'kemas' 'kemas rapi' 'cepat' 'rapi'
 'cepat kemas' 'puas sesuai' 'puas']

Kata paling berkontribusi ke sentimen NEGATIF:
['kecewa' 'deskripsi kecewa' 'barang rusak' 'barang' 'rusak kecewa'
 'rusak' 'parah barang' 'parah' 'deskripsi' 'sesuai']
```

Gambar 11. 4 Output Permodelan

11.4.5 Evaluasi Model (*Accuracy, Precission, Recall,* F1 + *Confussion Matrix*)

Setelah model *Logistic Regression* dilatih menggunakan data latih, tahap selanjutnya adalah evaluasi model untuk menilai kinerjanya pada data uji yang belum pernah dilihat sebelumnya. Evaluasi ini bertujuan untuk mengukur kemampuan model dalam mengklasifikasikan teks ke dalam kelas sentimen positif dan negatif secara akurat. Pada tahap ini digunakan beberapa metrik evaluasi, yaitu akurasi untuk melihat ketepatan prediksi secara keseluruhan, serta precision, recall, dan F1-score untuk menilai performa model pada masing-masing kelas. Selain itu, confusion matrix digunakan untuk memberikan gambaran kesalahan

klasifikasi yang dilakukan model. Kombinasi metrik ini memberikan evaluasi yang lebih komprehensif terhadap performa model klasifikasi teks.

```
from sklearn.metrics import accuracy_score,
classification_report, confusion_matrix


# 1) Prediksi pada data uji
y_pred = model.predict(X_test)


# 2) Accuracy
acc = accuracy_score(y_test, y_pred)
print("Accuracy:", acc)


# 3) Classification report (precision,
recall, f1-score)
print("\nClassification Report:")
print(classification_report(y_test,
y_pred))


# 4) Confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(cm)
```

```
Accuracy: 0.5

Classification Report:
              precision    recall  f1-score   support

         neg       0.00      0.00      0.00         1
         pos       0.50      1.00      0.67         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
weighted avg       0.25      0.50      0.33         2


Confusion Matrix:
[[0 1]
 [0 1]]
```

Gambar 11. 5 Output Evaluasi

- Library dan Modul Evaluasi yang Digunakan
  from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

  Tahap evaluasi model menggunakan modul metrics dari pustaka scikit-learn. Modul ini menyediakan berbagai fungsi evaluasi standar untuk *supervised learning*, khususnya klasifikasi, yang banyak digunakan dalam penelitian NLP.
- Akurasi

  Akurasi memberikan gambaran umum performa model, tetapi tidak selalu cukup, terutama jika data tidak seimbang.

```
acc = accuracy_score(y_test, y_pred)
print("Accuracy:", acc)
```

  Fungsi accuracy_score digunakan untuk menghitung akurasi, yaitu proporsi prediksi yang benar dibandingkan dengan seluruh data uji. Nilai akurasi memberikan gambaran umum tentang seberapa baik model melakukan klasifikasi secara keseluruhan, namun belum menggambarkan performa model pada masing-masing kelas secara detail.

Contoh interpretasi:

- Akurasi = 0.85 → 85% data uji berhasil diklasifikasikan dengan benar

- *Classification Report* (Precision, Recall, F1-score)

  Fungsi classification_report digunakan untuk menampilkan metrik evaluasi yang lebih rinci, yaitu precision, recall, dan F1-score.

```
y_pred = model.predict(X_test)

print("Accuracy:",
accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test,
y_pred))
```

  Precision mengukur ketepatan prediksi model terhadap suatu kelas.

  Recall mengukur kemampuan model dalam menemukan seluruh data yang benar pada kelas tersebut.

  F1-score merupakan rata-rata harmonik dari precision dan recall, sehingga memberikan ukuran performa yang lebih seimbang.

- *Confusion Matrix*

  Confusion matrix digunakan untuk menampilkan ringkasan kesalahan klasifikasi dalam bentuk matriks. Matriks ini menunjukkan jumlah data yang diprediksi dengan benar dan salah untuk masing-masing kelas, sehingga peneliti dapat mengidentifikasi pola kesalahan model, misalnya kecenderungan model mengklasifikasikan teks negatif sebagai positif atau sebaliknya.

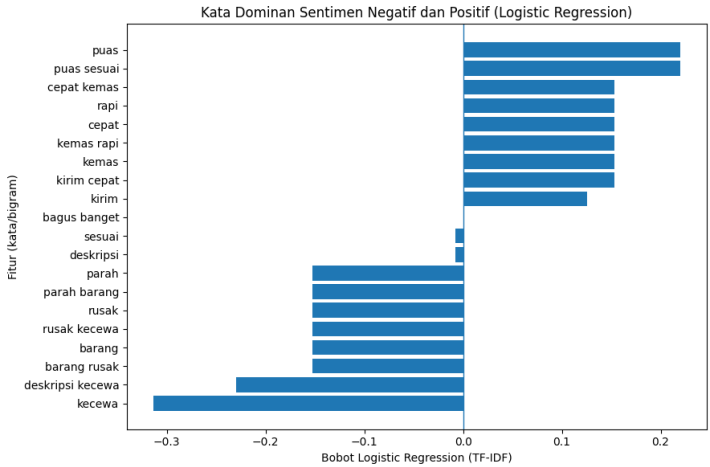
```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

11.4.6 Visualisasi Hasil

Visualisasi hasil digunakan untuk memahami kontribusi kata atau frasa terhadap keputusan model Logistic Regression dalam klasifikasi sentimen. Bobot regresi TF-IDF menunjukkan kata yang paling dominan mendorong sentimen positif maupun negatif, sehingga meningkatkan interpretabilitas model dan membantu menjelaskan pola linguistik yang dipelajari. Namun, visualisasi bobot fitur saja belum cukup untuk menilai kinerja model secara menyeluruh. Oleh karena itu, analisis dilanjutkan dengan *error analysis* untuk mengidentifikasi contoh teks yang salah diklasifikasikan dan memahami keterbatasan model dalam menangani ambiguitas bahasa.

```
import numpy as np
import matplotlib.pyplot as plt

# Ambil nama fitur dan bobot regresi
feature_names =
vectorizer.get_feature_names_out()
coef = model.coef_[0]

# Ambil top-k kata negatif dan positif
k = 10
top_neg_idx = np.argsort(coef)[:k]        #
bobot paling negatif
top_pos_idx = np.argsort(coef)[-k:]       #
bobot paling positif

top_neg_words = feature_names[top_neg_idx]
top_pos_words = feature_names[top_pos_idx]

top_neg_weights = coef[top_neg_idx]
top_pos_weights = coef[top_pos_idx]

# Gabungkan (negatif dulu, lalu positif)
dan urutkan untuk tampilan rapi
```

```
top_words = list(top_neg_words) +
list(top_pos_words)
top_weights = list(top_neg_weights) +
list(top_pos_weights)

# Sort berdasarkan bobot agar bar plot
terurut dari negatif ke positif
sorted_idx = np.argsort(top_weights)
top_words_sorted = [top_words[i] for i in
sorted_idx]
top_weights_sorted = [top_weights[i] for i
in sorted_idx]

# Plot
plt.figure(figsize=(9, 6))
plt.barh(top_words_sorted,
top_weights_sorted)
plt.axvline(0, linewidth=1)  # garis nol
untuk membedakan negatif vs positif
plt.title("Kata Dominan Sentimen Negatif
dan Positif (Logistic Regression)")
plt.xlabel("Bobot Logistic Regression (TF-
IDF)")
plt.ylabel("Fitur (kata/bigram)")
plt.tight_layout()
plt.show()
```



Gambar 11. 6 Output Bagian Visualisasi

Selain itu juga bisa ditampilkan tampilkan daftar kata dominan sebagai berikut:

```
print("Top kata NEGATIF:")
for w, v in sorted(zip(top_neg_words,
top_neg_weights), key=lambda x: x[1]):
    print(f"{w:20s} {v:.4f}")

print("\nTop kata POSITIF:")
for w, v in sorted(zip(top_pos_words,
top_pos_weights), key=lambda x: x[1],
reverse=True):
    print(f"{w:20s} {v:.4f}")
```

```
Top kata NEGATIF:
kecewa                -0.3138
deskripsi kecewa      -0.2298
barang rusak          -0.1528
barang                -0.1528
rusak kecewa          -0.1528
rusak                 -0.1528
parah barang          -0.1528
parah                 -0.1528
deskripsi             -0.0080
sesuai                -0.0080

Top kata POSITIF:
puas sesuai           0.2200
puas                  0.2200
kirim cepat           0.1532
kemas                 0.1532
kemas rapi            0.1532
cepat                 0.1532
rapi                  0.1532
cepat kemas           0.1532
kirim                 0.1256
bagus banget          0.0000
```

Gambar 11. 7 Output Visualisasi Daftar Kata Dominan

DAFTAR PUSTAKA

Christopher D. Manning, Manning, C. D., Raghava, P., & Schütze, H. (2009). *Introduction to Information Retrieval*. Cambridge University Press. https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf

Daniel Jurafsky, & James H. Martin. (2025). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models* (3rd Draft Edition). Stanford University.

Joachims, T. (2001). *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*.

Khurana, D., Koli, A., Khatter, K., & Singh, S. (2017). Natural Language Processing: State of The Art, Current Trends and Challenges. *Multimedia Tools and Applications*, *82*.

Salton, G., & Buckley, C. (1987). *Term Weighting Approaches in Automatic Text Retrieval*. 87–881.

Sebastiani, F. (2002). Machine Learning In Automated Text Categorization. *ACM Computing Surveys (CSUR)*, *34*, 1–47.

BAB 12

COMPUTER VISION DENGAN *OPENCV* DAN *DEEP LEARNING*

12.1 Computer Vision

Computer Vision merupakan salah satu cabang penting dalam kecerdasan buatan yang berfokus pada kemampuan komputer untuk melihat, memahami, dan mengekstraksi informasi bermakna dari citra maupun video secara otomatis. Dalam bidang ini, sistem komputasi dirancang untuk melakukan tugas-tugas persepsi visual yang sebelumnya hanya dapat dilakukan oleh manusia, seperti mengenali objek, mendeteksi pola, memahami struktur ruang, hingga menafsirkan konteks visual dari lingkungan sekitar. Menurut Marpaung, Aulia and Nabila (2022), kemampuan ini dicapai melalui tahapan pemrosesan citra digital yang memungkinkan komputer mengubah data visual mentah menjadi representasi yang dapat dianalisis lebih lanjut. Di era modern, Computer Vision tidak hanya digunakan untuk keperluan penelitian akademik, tetapi juga diaplikasikan dalam berbagai sistem cerdas seperti pengenalan wajah, navigasi kendaraan otonom, sistem pengawasan otomatis, hingga analisis citra medis, menjadikannya salah satu fondasi utama dalam pengembangan teknologi berbasis AI. (Hassaballah and Hosny, 2019). Dengan demikian, Computer Vision bertujuan meniru kemampuan persepsi manusia dalam sistem komputasi sehingga komputer mampu melakukan tugas seperti mengenali objek, mendeteksi pola,

memahami struktur adegan, hingga mengambil keputusan berdasarkan informasi visual tersebut.

Dalam ekosistem kecerdasan buatan dan machine learning, Computer Vision menempati posisi yang sangat strategis karena data visual merupakan salah satu jenis data yang paling melimpah dan kompleks di era digital. Aplikasi seperti kamera ponsel, sistem pengawasan, drone, mesin industri, dan platform media sosial setiap hari menghasilkan miliaran citra dan video. Untuk mengolah data sebesar itu secara efisien, integrasi antara Computer Vision dan machine learning menjadi sangat diperlukan. Algoritma pembelajaran mesin memberi kemampuan untuk melakukan klasifikasi, deteksi objek, pengenalan wajah, segmentasi citra, dan berbagai operasi lain yang melibatkan pemahaman terhadap pola visual (Ghosh *et al.*, 2019). Python menjadi bahasa pemrograman yang paling banyak digunakan untuk tujuan ini berkat ekosistem library yang luas seperti OpenCV, NumPy, Pandas, dan berbagai framework deep learning seperti TensorFlow dan PyTorch, yang memudahkan pengembangan solusi AI berbasis visual.

Perbedaan mendasar antara persepsi visual manusia dan komputer menjadi tantangan sekaligus landasan bagi seluruh teknik dalam Computer Vision. Bagi komputer, citra bukanlah gambar utuh yang mudah dikenali, melainkan kumpulan nilai numerik yang tersusun dalam bentuk matriks piksel. Oleh karena itu, berbagai teknik pemrosesan citra dikembangkan untuk membantu komputer memahami struktur di balik data tersebut. Operasi dasar seperti filtering digunakan untuk mengurangi noise, edge detection digunakan untuk mengenali tepi objek, sementara transformasi citra membantu mengubah perspektif atau struktur gambar. Tahap-tahap ini sangat penting dalam menyiapkan citra agar dapat dianalisis lebih lanjut oleh algoritma machine learning maupun model deep learning.

Sebelum era deep learning mendominasi, Computer Vision sangat bergantung pada metode klasik yang mengharuskan perancangan fitur secara manual oleh pakar. Teknik seperti SIFT, SURF, dan HOG digunakan untuk mengekstraksi fitur yang relevan dari sebuah citra, sementara algoritma klasifikasi seperti SVM, KNN, dan Random Forest digunakan untuk melakukan identifikasi objek. Pendekatan ini terbukti efektif pada banyak kasus, tetapi memiliki keterbatasan ketika data visual semakin kompleks. Revolusi besar terjadi dengan hadirnya Convolutional Neural Networks (CNN), yang memungkinkan sistem belajar fitur langsung dari data mentah tanpa perlu perancangan manual. Model-model modern seperti VGG, ResNet, YOLO, dan U-Net meningkatkan akurasi dan efisiensi secara signifikan dalam berbagai tugas Computer Vision, mulai dari klasifikasi citra hingga deteksi dan segmentasi objek dalam waktu nyata (Zhao *et al.*, 2024).

Computer Vision kini telah menjadi teknologi kunci dalam berbagai bidang kehidupan modern. Dalam dunia kesehatan, CV digunakan untuk menganalisis citra medis seperti MRI dan CT-scan. Pada industri manufaktur, teknologi ini membantu melakukan inspeksi kualitas produk secara otomatis. Di bidang keamanan, pengenalan wajah dan deteksi aktivitas menjadi lebih canggih dengan bantuan model deep learning. Bahkan dalam kendaraan otonom, Computer Vision bekerja bersama sensor lain untuk memahami lingkungan sekitar dalam waktu nyata. Pesatnya perkembangan ini menunjukkan bahwa Computer Vision bukan hanya konsep akademis, tetapi juga fondasi bagi berbagai inovasi praktis di industri.

12.2 Dasar-Dasar OpenCV untuk Pemrosesan Citra

OpenCV (*Open Source Computer Vision Library*) adalah library open-source yang digunakan untuk pemrosesan citra dan visi komputer. OpenCV dirancang untuk mendukung berbagai jenis tugas dalam pengolahan gambar dan video secara real-time, mulai dari pengenalan objek, deteksi wajah, hingga pengolahan citra medis. Karena sifatnya yang open-source, OpenCV menjadi salah satu library yang paling banyak digunakan dalam penelitian dan aplikasi komersial di bidang visi komputer dan kecerdasan buatan (AI). Library ini memiliki implementasi yang sangat efisien, mendukung berbagai bahasa pemrograman, namun Python menjadi pilihan utama karena integrasinya yang kuat dengan berbagai tools lain dalam ekosistem data science dan machine learning, seperti NumPy dan SciPy.

Gambar 12. 1 OpenCV

Untuk memulai menggunakan OpenCV di Python, Anda hanya perlu menginstal paket opencv-python menggunakan pip, manajer paket Python yang sangat populer. Berikut adalah langkah-langkah untuk menginstal OpenCV di sistem Anda:

```
pip install opencv-python
```

Setelah menginstal library tersebut, Anda sudah dapat mulai menggunakan OpenCV untuk membaca, memanipulasi, dan memproses citra serta video. Penggunaan OpenCV dalam Python sangat mudah dan intuitif, karena sebagian besar operasi dasar pemrosesan citra dapat dilakukan dalam beberapa baris kode saja. OpenCV mendukung berbagai format citra, seperti PNG, JPEG, dan TIFF, serta format video seperti MP4 dan AVI. Fungsi-fungsi dasar yang disediakan dalam OpenCV memungkinkan Anda untuk membaca citra, memodifikasi konten citra, menerapkan filter, mendeteksi objek, dan banyak lagi.

Salah satu operasi dasar yang paling sering digunakan adalah membaca dan menampilkan citra. Berikut adalah contoh kode sederhana yang menunjukkan cara membaca gambar menggunakan OpenCV dan menampilkannya dalam jendela:

```
import cv2


# Membaca gambar
image = cv2.imread('gambar.jpg')


# Menampilkan gambar
cv2.imshow('Gambar', image)


# Menunggu hingga tombol ditekan
cv2.waitKey(0)
```

Pada kode di atas, fungsi cv2.imread() digunakan untuk membaca gambar dari disk, sementara `cv2.imshow()` menampilkan gambar dalam jendela. Fungsi `cv2.waitKey(0)` membuat program menunggu hingga pengguna menekan tombol apa saja sebelum menutup jendela dengan `cv2.destroyAllWindows()`.

OpenCV tidak hanya memungkinkan untuk membaca dan menampilkan citra, tetapi juga menyediakan berbagai teknik pemrosesan citra dasar yang sangat berguna untuk analisis gambar lebih lanjut. Misalnya, salah satu teknik dasar adalah konversi citra ke grayscale. Grayscale adalah citra yang hanya menggunakan satu saluran warna untuk intensitas cahaya, sehingga membuat analisis lebih sederhana dengan mengurangi kompleksitas data.

Untuk mengonversi citra ke grayscale, kita dapat menggunakan fungsi `cv2.cvtColor()` yang ada di `OpenCV`. Berikut adalah contoh kode yang mengubah citra warna menjadi citra grayscale:

```
Python


# Mengonversi citra ke grayscale
gray_image = cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY)


# Menampilkan gambar grayscale
```

Gambar 12. 2 Citra RGB dan Citra Grayscale

OpenCV juga dapat bekerja bersama berbagai library pendukung seperti NumPy, yang memungkinkan manipulasi array dalam memori. Setiap citra yang dimuat oleh OpenCV sebenarnya direpresentasikan sebagai objek NumPy array, yang memungkinkan pemrosesan citra dalam bentuk matriks

numerik. Ini memudahkan pengguna untuk menerapkan operasi matematis dan pemrosesan data dalam citra menggunakan teknik-teknik yang ada di NumPy.

Misalnya, kita dapat menggunakan NumPy untuk mengakses dan memodifikasi nilai piksel dalam citra. Berikut adalah contoh kode untuk mengakses nilai piksel di citra grayscale:

```
Python


import numpy as np


# Mengakses piksel di baris ke-50 dan kolom ke-
100

pixel_value = gray_image[50, 100]

print(f'Nilai piksel pada baris 50 dan kolom
100: {pixel_value}')
```

Dengan menggunakan NumPy, kita dapat mengakses nilai intensitas piksel dalam citra dan memodifikasinya jika diperlukan untuk aplikasi tertentu, seperti pemrograman pengolahan citra berbasis algoritma.

12.3 *Fitur Extraction* dan Klasifikasi Berbasis *Machine Learning*

Sebelum hadirnya pendekatan berbasis deep learning, pemrosesan citra dalam Computer Vision (CV) sangat bergantung pada teknik feature extraction dan klasifikasi berbasis machine learning. Pada era ini, algoritma dirancang untuk mengekstraksi fitur penting dari citra—seperti tepi,

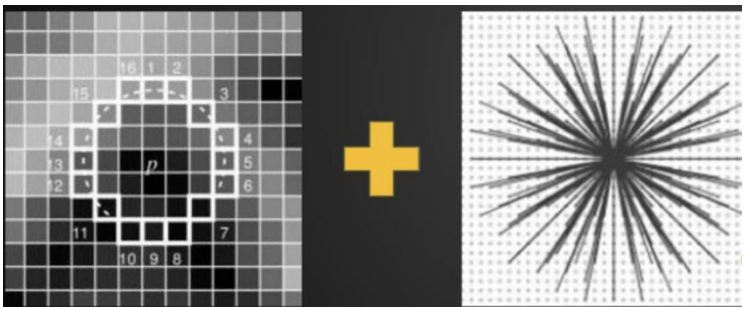
sudut, tekstur, maupun pola lokal—secara manual dengan mengandalkan metode matematika dan teknik pengolahan citra digital. Setelah fitur diperoleh, proses selanjutnya adalah mengklasifikasikan fitur tersebut menggunakan algoritma pembelajaran mesin seperti SVM, KNN, dan Random Forest. Pendekatan ini dikenal sebagai shallow learning, dan menjadi fondasi awal evolusi Computer Vision modern (Archana and Jeevaraj, 2024).

12.3.1 Ekstraksi Fitur: SIFT, SURF, dan ORB

Scale-Invariant Feature Transform (SIFT) merupakan salah satu algoritma paling berpengaruh yang mampu mengekstraksi titik-titik penting (*keypoints*) pada citra yang bersifat invariant terhadap perubahan skala, rotasi, dan iluminasi. SIFT bekerja melalui empat tahap utama: deteksi titik karakteristik, lokalisasi keypoint, penentuan orientasi, dan pembuatan deskriptor (Huang *et al.*, 2011). Meski sangat akurat, SIFT relatif lambat sehingga kurang cocok untuk aplikasi real-time.

Speeded-Up Robust Features (SURF) dikembangkan untuk mempercepat kinerja SIFT dengan memanfaatkan integral image dan Hessian matrix. SURF mampu menghasilkan deskriptor yang lebih kompak dan cepat, namun metode ini pernah memiliki pembatasan paten sehingga tidak tersedia secara bebas dalam semua implementasi.

Oriented FAST and Rotated BRIEF (ORB) merupakan metode open-source yang dirancang sebagai alternatif SIFT/SURF, dengan komputasi jauh lebih cepat dan cocok untuk aplikasi real-time. ORB menggunakan FAST untuk deteksi keypoint dan BRIEF untuk membuat deskriptor biner yang efisien, sehingga dapat bekerja baik pada perangkat dengan keterbatasan komputasi.



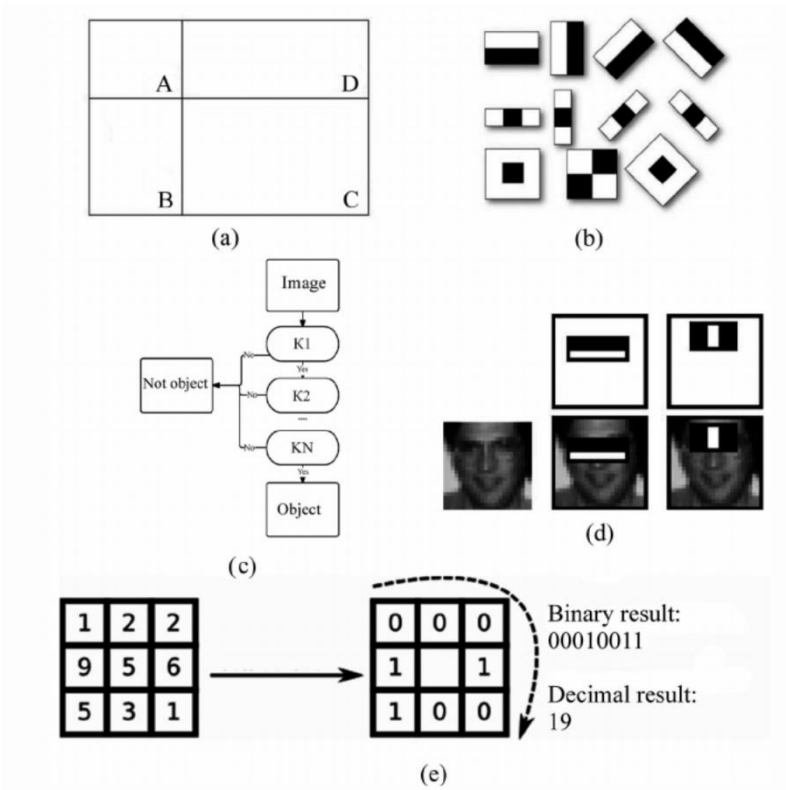
Gambar 12. 3 *ORB Fiture Detection*

Metode ini memainkan peran penting dalam tugas seperti pencocokan citra (image matching), image stitching, rekonstruksi 3D, hingga pelacakan objek sederhana.

12.3.2 Deteksi Objek Berbasis Haar Cascade

Sebelum era CNN, Haar Cascade Classifier menjadi salah satu teknik paling populer untuk deteksi objek, terutama deteksi wajah. Metode ini diperkenalkan oleh (Viola and Jones, 2001), menggunakan pendekatan Haar-like features, yang diolah melalui struktur Cascade of Classifiers untuk mempercepat proses inferensi. Algoritma ini mampu memindai citra pada berbagai ukuran jendela (sliding window) secara efisien, menjadikannya solusi real-time pertama yang dapat berjalan pada komputer dengan sumber daya terbatas.

Contoh umum implementasinya adalah face detection menggunakan dataset Haar Cascade yang telah dilatih sebelumnya pada ribuan contoh citra wajah.



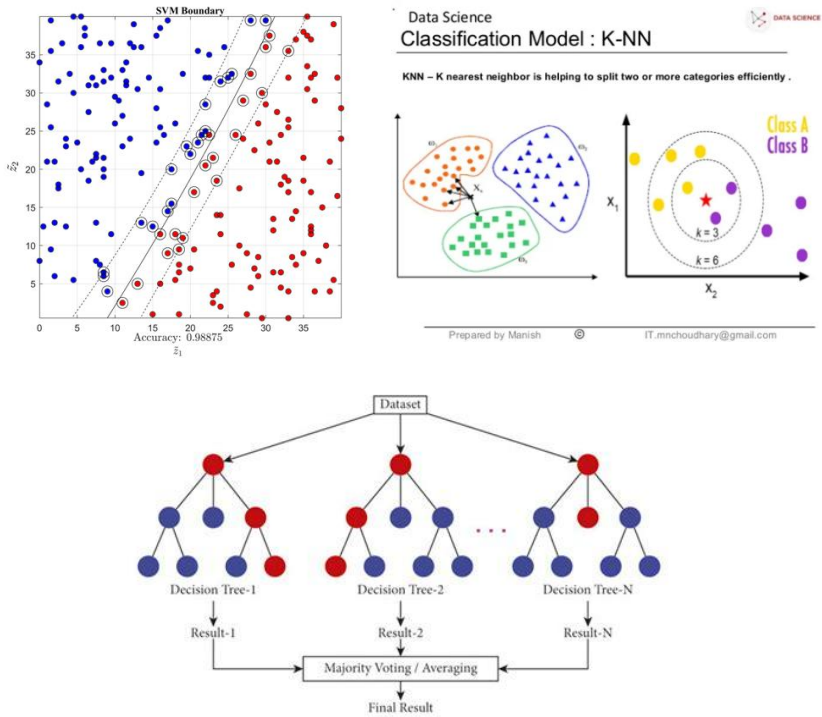
Gambar 12. 4 Algoritma Haar Cascade, yang juga Dikenal sebagai Algoritma Viola-Jones (Viola and Jones, 2001)

Meskipun cukup kuat untuk tugas sederhana, Haar Cascade memiliki keterbatasan dalam kondisi pencahayaan buruk, pose bervariasi, atau objek yang tidak terdefinisi dengan jelas.

12.3.3 Klasifikasi Menggunakan SVM, KNN, dan *Random Forest*

Setelah fitur diekstraksi, citra diklasifikasikan menggunakan algoritma pembelajaran mesin:

1. ***Support Vector Machine* (SVM)**: Dirancang untuk mencari hyperplane terbaik yang memisahkan kelas dengan margin maksimum. Cocok untuk data fitur berdimensi tinggi seperti SIFT dan ORB. Efektif untuk klasifikasi objek sederhana (Cortes and Vapnik, 1995).
2. ***K-Nearest Neighbor* (KNN):** Metode sederhana yang mengklasifikasikan berdasarkan kedekatan jarak antar fitur. Sering digunakan dalam aplikasi pemrosesan citra yang tidak memerlukan pelatihan model yang kompleks.
3. ***Random Forest*:** Algoritma ensemble berbasis pohon keputusan yang kuat terhadap data berisik. Cocok untuk fitur heterogen dan tidak memerlukan banyak praproses.



Gambar 12. 5 SVM, KNN, dan *Random Forest* (Anubi and Konstantinou, 2020; Khan *et al.*, 2021)

Ketiga algoritma ini bekerja baik dalam konteks CV klasik karena fitur sudah diekstraksi dengan jelas, sehingga model hanya perlu mempelajari pola pada domain fitur. Tabel berikut merangkum kelebihan dan keterbatasan metode berbasis fitur dibandingkan metode deep learning modern:

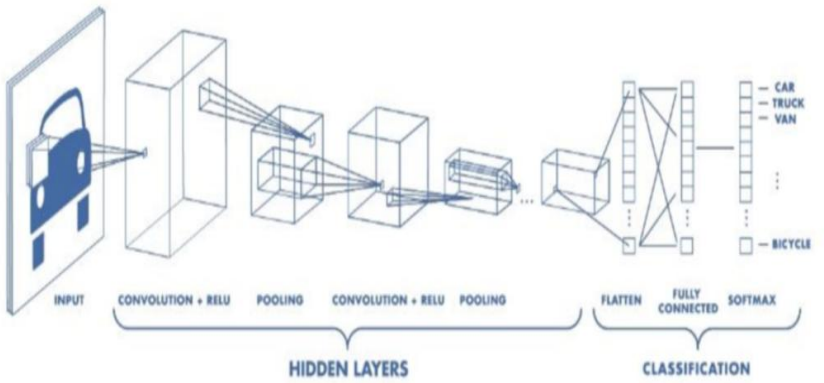
Tabel 12. 1 Perbandingan CV Klasik dan *Deep Learning*

Aspek	CV Klasik (SIFT/SURF/ORB + ML)	*Deep Learning* (CNN, YOLO, ResNet)
Representasi fitur	Dirancang manual (*handcrafted features*)	Dipelajari otomatis dari data (*learned features*)
Kebutuhan data	Relatif sedikit	Membutuhkan dataset besar
Kecepatan inferensi	Cepat pada ORB, lambat pada SIFT/SURF	Sangat cepat dengan GPU
Akurasi	Baik untuk pola sederhana	Sangat tinggi untuk tugas kompleks
Robustness	Tidak stabil terhadap rotasi ekstrim, noise, pose	Sangat robust terhadap variasi pencahayaan, pose, dan noise
Kemudahan implementasi	Cukup mudah, terutama ORB	Memerlukan pelatihan model dan tuning parameter
Aplikasi	Image matching, tracking sederhana, face detection	Face recognition, detection, segmentation, autonomous systems

Metode *feature extraction* dan klasifikasi berbasis machine learning merupakan pondasi penting dalam sejarah perkembangan Computer Vision. Pendekatan ini mengandalkan desain fitur secara manual, seperti SIFT, SURF, dan ORB, untuk merepresentasikan citra dalam bentuk fitur numerik yang kemudian diproses oleh algoritma machine learning seperti SVM, KNN, atau Random Forest. Meskipun teknik klasik ini masih memiliki relevansi dalam beberapa aplikasi, khususnya yang memerlukan kecepatan pada perangkat terbatas, metode ini memiliki keterbatasan dalam menghadapi kerumitan data visual modern. Kemunculan deep learning—khususnya CNN—membawa lompatan besar dalam akurasi dan kemampuan generalisasi, sehingga kini menjadi pendekatan dominan dalam Computer Vision modern.

12.4 *Deep Learning* untuk Computer Vision

Perkembangan *Deep Learning* membawa perubahan paradigma yang sangat signifikan dalam bidang Computer Vision. Berbeda dengan pendekatan klasik yang mengandalkan ekstraksi fitur manual, Deep Learning memungkinkan sistem mempelajari fitur visual secara otomatis langsung dari data mentah. Pendekatan ini terbukti sangat efektif dalam menangani kompleksitas data visual modern, seperti variasi pencahayaan, sudut pandang, latar belakang, dan skala objek. Model deep learning, khususnya *Convolutional Neural Networks* (CNN), kini menjadi tulang punggung utama dalam berbagai aplikasi Computer Vision, mulai dari klasifikasi citra hingga deteksi dan segmentasi objek (Zhao *et al.*, 2024).



Gambar 12. 6 Arsitektur CNN (Purwono *et al.*, 2022)

CNN merupakan arsitektur jaringan saraf yang secara khusus dirancang untuk memproses data berbentuk grid, seperti citra dua dimensi. CNN bekerja dengan memanfaatkan operasi konvolusi untuk mengekstraksi fitur spasial dari citra, sehingga mampu mengenali pola lokal seperti tepi, sudut, tekstur, hingga bentuk kompleks. Secara umum, CNN terdiri dari beberapa komponen utama, yaitu convolutional layer, activation function, pooling layer, dan fully connected layer. Lapisan konvolusi bertugas mengekstraksi fitur, sedangkan pooling digunakan untuk mereduksi dimensi dan meningkatkan ketahanan terhadap pergeseran kecil pada citra. Dengan menumpuk banyak lapisan konvolusi, CNN mampu membangun representasi fitur secara hierarkis, dari fitur sederhana hingga fitur tingkat tinggi yang lebih abstrak.

Seiring perkembangan riset, berbagai arsitektur CNN dikembangkan untuk meningkatkan performa dan efisiensi model. LeNet merupakan salah satu arsitektur awal yang digunakan untuk pengenalan karakter tulisan tangan. AlexNet menandai kebangkitan deep learning dalam Computer Vision dengan memenangkan kompetisi ImageNet 2012. Selanjutnya, VGGNet memperkenalkan arsitektur yang

lebih dalam dengan kernel kecil yang konsisten, sementara ResNet memperkenalkan konsep residual learning untuk mengatasi masalah degradasi pada jaringan yang sangat dalam (He *et al.*, 2016). Arsitektur-arsitektur ini menjadi dasar bagi banyak model Computer Vision modern.

Selain klasifikasi citra, *Deep Learning* juga memungkinkan pengembangan tugas Computer Vision yang lebih kompleks. *Object Detection* bertujuan untuk mendeteksi sekaligus menentukan lokasi objek dalam citra. Model populer seperti YOLO (*You Only Look Once*) dan Faster R-CNN mampu melakukan deteksi objek secara real-time dengan akurasi tinggi. Sementara itu, Semantic Segmentation bertujuan untuk mengklasifikasikan setiap piksel dalam citra, sehingga sangat berguna dalam aplikasi medis, kendaraan otonom, dan pemetaan lingkungan. Model seperti U-Net dan DeepLab banyak digunakan dalam tugas ini.

Salah satu keunggulan utama *Deep Learning* adalah konsep transfer learning, yaitu memanfaatkan model yang telah dilatih pada dataset besar (seperti ImageNet) untuk tugas lain yang lebih spesifik. Dengan transfer learning, proses pelatihan menjadi lebih cepat dan efisien, serta mengurangi kebutuhan dataset berukuran besar. Pendekatan ini sangat relevan dalam konteks implementasi praktis, terutama ketika data terbatas. Model seperti VGG, ResNet, dan MobileNet sering digunakan sebagai backbone dalam transfer learning untuk berbagai aplikasi Computer Vision.

Berikut contoh implementasi sederhana CNN menggunakan TensorFlow/Keras untuk klasifikasi citra:

```
Python


from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense


model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(128,128,3)),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

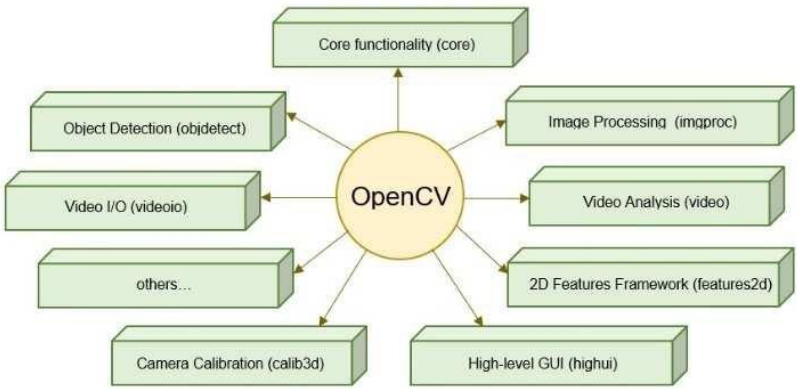
Kode di atas menunjukkan struktur CNN dasar yang terdiri dari lapisan konvolusi, pooling, dan fully connected, yang umum digunakan dalam berbagai aplikasi Computer Vision berbasis Python.

Deep Learning telah menjadikan Computer Vision lebih adaptif, akurat, dan skalabel dibandingkan pendekatan klasik. Dengan kemampuan belajar otomatis dari data, model deep learning mampu menangani kompleksitas visual dunia nyata yang sebelumnya sulit dicapai. Namun demikian, pendekatan

ini juga memiliki tantangan, seperti kebutuhan komputasi tinggi, ketergantungan pada data besar, serta interpretabilitas model yang relatif rendah.

12.5 Integrasi OpenCV dengan *Deep Learning*

Integrasi OpenCV dengan *Deep Learning* merupakan langkah penting dalam membangun sistem Computer Vision yang siap digunakan di dunia nyata. OpenCV berperan sebagai alat utama untuk akuisisi dan prapemrosesan data visual, sementara model deep learning bertugas melakukan analisis tingkat lanjut seperti klasifikasi, deteksi, dan segmentasi objek. Kombinasi ini memungkinkan pengembangan aplikasi Computer Vision yang tidak hanya akurat, tetapi juga efisien dan dapat berjalan secara real-time. Dalam konteks pengembangan AI berbasis Python, OpenCV menjadi jembatan antara data visual mentah dan model deep learning yang telah dilatih menggunakan framework seperti TensorFlow, Keras, dan PyTorch.



Gambar 12. 7 Struktur Modular OpenCV (Ralev and Krastev, 2022)

Dalam sistem Computer Vision modern, OpenCV umumnya digunakan pada tahap awal dan akhir pipeline deep learning. Pada tahap awal, OpenCV bertanggung jawab atas pembacaan citra atau video, konversi ruang warna, resizing, normalisasi, serta augmentasi data. Proses ini sangat penting karena kualitas input sangat mempengaruhi performa model deep learning. Setelah model menghasilkan prediksi, OpenCV kembali digunakan untuk menampilkan hasil inferensi, menggambar bounding box, label kelas, serta menampilkan skor kepercayaan pada citra atau video. Dengan demikian, OpenCV berfungsi sebagai antarmuka visual antara model AI dan pengguna.

OpenCV menyediakan modul khusus bernama DNN (Deep Neural Network) yang memungkinkan pemuatan dan inferensi model deep learning tanpa harus menjalankan framework pelatihan secara langsung. Modul ini mendukung berbagai format model seperti ONNX, TensorFlow (.pb), Caffe, dan Darknet. Keunggulan utama OpenCV DNN adalah kemampuannya menjalankan model deep learning dengan efisien, bahkan pada perangkat dengan sumber daya terbatas.

Contoh pemuatan model *deep learning* menggunakan OpenCV DNN ditunjukkan pada kode berikut:

```
import cv2

net = cv2.dnn.readNetFromONNX("model.onnx")

image = cv2.imread("input.jpg")
blob = cv2.dnn.blobFromImage(
    image, scalefactor=1/255.0, size=(224, 224),
    mean=(0, 0, 0), swapRB=True, crop=False
)
```

Kode tersebut menunjukkan bagaimana OpenCV mengubah citra menjadi blob sebagai input model deep learning, lalu menjalankan proses inferensi.

Salah satu implementasi paling populer dari integrasi OpenCV dan deep learning adalah deteksi objek secara real-time, misalnya menggunakan YOLO. OpenCV digunakan untuk membaca frame video dari kamera, sementara model YOLO mendeteksi objek dalam setiap frame. Hasil deteksi kemudian divisualisasikan menggunakan fungsi `drawing OpenCV` seperti `cv2.rectangle()` dan `cv2.putText()`.

OpenCV mendukung akuisisi video secara langsung dari kamera melalui cv2.VideoCapture(). Hal ini memungkinkan implementasi sistem Computer Vision berbasis deep learning yang berjalan secara real-time. Dengan memproses setiap frame video dan menerapkannya ke model deep learning, sistem dapat melakukan pengenalan objek, pelacakan, atau analisis visual lainnya secara kontinu.

Integrasi OpenCV dengan deep learning menawarkan berbagai keunggulan, antara lain kemudahan implementasi, fleksibilitas, serta kemampuan real-time. Namun, terdapat pula tantangan seperti kebutuhan optimasi performa, keterbatasan perangkat keras, dan manajemen latensi. Oleh karena itu, pemilihan model yang tepat, penggunaan teknik optimasi, serta pemanfaatan GPU atau akselerator menjadi faktor penting dalam pengembangan sistem Computer Vision berbasis deep learning.

DAFTAR PUSTAKA

Anubi, O. and Konstantinou, C. (2020) *Adversarial Examples on Power Systems State Estimation*. Available at: https://doi.org/10.1109/ISGT45199.2020.9087789.

Archana, R. and Jeevaraj, P.S.E. (2024) "Deep learning models for digital image processing: a review," *Artificial Intelligence Review*, 57(1), p. 11.

Cortes, C. and Vapnik, V. (1995) "Support-vector networks," *Machine learning*, 20(3), pp. 273–297.

Ghosh, S. *et al.* (2019) "Understanding deep learning techniques for image segmentation," *ACM computing surveys (CSUR)*, 52(4), pp. 1–35.

Hassaballah, M. and Hosny, K.M. (2019) "Recent advances in computer vision," *Studies in computational intelligence*, 804(1–84), p. 3.

He, K. *et al.* (2016) "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778. Available at: https://doi.org/10.1109/CVPR.2016.90.

Huang, F.-C. *et al.* (2011) "High-performance SIFT hardware accelerator for real-time image feature extraction," *IEEE Transactions on Circuits and Systems for Video Technology*, 22(3), pp. 340–351.

Khan, M.Y. *et al.* (2021) "Automated Prediction of Good Dictionary EXamples (GDEX): A Comprehensive Experiment with Distant Supervision, Machine Learning, and Word Embedding-Based Deep Learning Techniques," *Complexity* [Preprint]. Available at: https://doi.org/10.1155/2021/2553199.

Marpaung, F., Aulia, F. and Nabila, R.C. (2022) "Computer Vision Dan Pengolahan Citra Digital." PUSTAKA AKSARA.

Purwono, P. *et al.* (2022) "Understanding of convolutional neural network (cnn): A review," *International Journal of Robotics and Control Systems*, 2(4), pp. 739–748.

Ralev, I. and Krastev, G. (2022) *Components and model of implementation of a hand gesture recognition system*. Available at: https://doi.org/10.1109/HORA55278.2022.9799919.

Viola, P. and Jones, M. (2001) "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*. Ieee, pp. I–I.

Zhao, X. *et al.* (2024) "A review of convolutional neural networks in computer vision," *Artificial Intelligence Review*, 57(4), p. 99.

BAB 13

EVALUASI MODEL DAN OPTIMASI KINERJA *MACHINE LEARNING*

Oleh Muhammad Hasbi

13.1 Pentingnya Evaluasi Model dalam *Machine Learning*

Evaluasi model merupakan tahap krusial dalam siklus pengembangan machine learning karena menentukan seberapa baik model dapat memprediksi atau mengambil keputusan pada data yang belum pernah dilihat sebelumnya. Tanpa evaluasi yang tepat, model yang tampak akurat pada data pelatihan berpotensi menghasilkan prediksi buruk saat dihadapkan pada data nyata, sehingga menurunkan keandalan dan kegunaan model tersebut dalam praktik sesungguhnya. Evaluasi model machine learning memiliki peran penting dalam memastikan keandalan, generalisasi, selection model yang tepat, serta nilai praktis dalam aplikasi nyata. Evaluasi tidak hanya membantu memilih model terbaik, tetapi juga mendukung monitoring dan peningkatan performa model dalam konteks dunia nyata secara berkelanjutan

13.1.1 Menjamin Generalisasi Model

Salah satu tujuan utama evaluasi adalah memastikan bahwa model tidak hanya "*to memorize*" terhadap data pelatihan tetapi juga dapat bekerja dengan

baik pada data baru. Proses ini membantu mendeteksi *overfitting* dan *underfitting*, dua masalah umum yang dapat memengaruhi performa model jika tidak diidentifikasi secara dini melalui evaluasi yang sistematis. Kegagalan untuk mendeteksi fenomena ini dapat membuat model tidak relevan di dunia nyata meskipun menunjukkan performa tinggi pada fase pelatihan.

13.1.2 Pemilihan Model Terbaik

Evaluasi model memungkinkan peneliti dan praktisi untuk membandingkan beberapa pendekatan algoritmik secara kuantitatif. Dengan menerapkan metrik evaluasi yang tepat, seperti akurasi, presisi, recall, F1-score untuk klasifikasi, atau *Mean Squared Error* (MSE) dan R^2 untuk regresi, pengembang dapat menentukan model mana yang paling sesuai menurut tujuan spesifik masalah. Ini penting karena performa model seringkali bergantung pada karakteristik dataset dan konteks penerapan (Sekeroglu et al., 2022).

13.1.3 Dampak pada Keputusan dan Dampak Nyata

Evaluasi model bukan hanya aspek teknis tetapi juga berkaitan langsung dengan efektivitas model dalam aplikasi dunia nyata. Misalnya, dalam konteks kesehatan, model yang tidak dievaluasi secara akurat bisa memberikan diagnosis yang salah, yang berpotensi menyebabkan risiko serius bagi pasien. Evaluasi model yang memadai memastikan bahwa model yang digunakan memberikan prediksi yang dapat diandalkan serta mengurangi implikasi berbahaya dari kesalahan prediksi (Research, Society and Development, 2023).

13.1.4 Pemantauan dan Perbaikan Berkelanjutan

Evaluasi bukan hanya satu kali kegiatan saat pelatihan model selesai, tetapi juga harus dilakukan secara berkala setelah model diterapkan *(post-deployment)*. Perubahan data nyata dalam lingkungan operasional (data drift) bisa menyebabkan penurunan performa seiring waktu. Evaluasi berkelanjutan membantu dalam pemantauan performa model serta memicu retraining atau penyesuaian saat diperlukan, mempertahankan relevansi model dalam jangka Panjang (Research, Society and Development, 2023).

13.1.5 Nilai Tambah bagi Organisasi

Penelitian terbaru menunjukkan bahwa pendekatan evaluasi model harus mempertimbangkan bukan hanya metrik numerik, tetapi juga nilai yang dihasilkan model dalam konteks organisasi atau pengguna akhir. Model yang baik bukan hanya yang optimal secara statistik, tetapi yang memberikan nilai tambah dalam alur kerja nyata, misalnya integrasi prediksi dengan keahlian manusia untuk pengambilan keputusan hibrid. Pendekatan evaluasi yang lebih holistik seperti ini mengarah pada implementasi AI yang lebih efektif (Sayin et al., 2025).

13.1.6 Metrik Evaluasi untuk Klasifikasi

Sathyanarayanan dan Tantri (2025) menekankan bahwa confusion matrix bukan hanya alat visual, tetapi fondasi perhitungan metrik performa yang penting dalam menilai kualitas model klasifikasi.

1. *Accuracy* (Akurasi)

   Accuracy mengukur proporsi prediksi yang benar dari seluruh data yang diuji. Metrik ini cocok digunakan ketika data memiliki distribusi kelas yang seimbang, namun kurang representatif pada

data tidak seimbang karena dapat memberikan kesan performa yang tinggi meskipun model gagal mengenali kelas minoritas. Berikut adalah rumus metrik evaluasi klasifikasi untuk *Accuracy*.

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

2. *Precision*

   Precision menunjukkan tingkat ketepatan prediksi positif, yaitu seberapa banyak prediksi positif yang benar dari seluruh prediksi positif yang dihasilkan model. Metrik ini penting pada aplikasi yang harus meminimalkan kesalahan positif palsu (*false positive*), seperti deteksi penipuan atau spam. Berikut adalah rumus metrik evaluasi klasifikasi untuk *Precision*.

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

3. *Recall* (Sensitivity)

   Recall mengukur kemampuan model dalam menangkap seluruh data positif yang sebenarnya. Metrik ini berfokus pada minimisasi kesalahan negatif palsu (*false negative*) dan sangat penting pada bidang medis atau keselamatan, di mana kegagalan mendeteksi kasus positif dapat berdampak serius. Berikut adalah rumus metrik evaluasi klasifikasi untuk *Recall*

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

4. *F1-Score*

   F1-Score merupakan rata-rata harmonik antara precision dan recall. Metrik ini digunakan untuk menyeimbangkan kedua ukuran tersebut, terutama

pada dataset tidak seimbang, sehingga memberikan gambaran performa model yang lebih adil dibandingkan *accuracy* saja. Berikut adalah rumus metrik evaluasi klasifikasi untuk *F1-Score.*

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

Perbandingan antara label aktual dan hasil prediksi model dalam bentuk *True Positive, True Negative, False Positive, dan False Negative* disebut *confusion matrix*. Matriks ini menjadi dasar perhitungan seluruh metrik evaluasi klasifikasi.

Adapun keterangan simbol dari rumus -rumus metrik evaluasi klasifikasi sebagai berikut:
TP = True Positive
TN = True Negative
FP = False Positive
FN = False Negative

13.1.6 Metrik Evaluasi untuk Regresi

Sekeroglu et al. (2022) menyatakan bahwa MAE, MSE, RMSE, dan R^2 merupakan metrik utama dalam evaluasi kinerja model regresi.

1. *Mean Absolute Error (MAE)*

   Mean Absolute Error (MAE) mengukur rata-rata nilai absolut dari selisih antara nilai aktual dan nilai prediksi. MAE memberikan gambaran kesalahan rata-rata tanpa memperhatikan arah kesalahan (positif atau negatif). Metrik ini relatif mudah diinterpretasikan karena memiliki satuan yang sama dengan data asli dan tidak terlalu sensitif terhadap nilai ekstrem (outlier). Berikut adalah rumus metrik evaluasi regresi untuk *Mean Absolute Error (MAE).*

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

2. *Mean Squared Error (MSE)*

   Mean Squared Error (MSE) menghitung rata-rata kuadrat dari selisih antara nilai aktual dan nilai prediksi. Dengan melakukan pengkuadratan, MSE memberikan penalti lebih besar pada kesalahan yang besar. Oleh karena itu, metrik ini sangat sensitif terhadap outlier dan sering digunakan ketika kesalahan besar harus dihindari. Berikut adalah rumus metrik evaluasi regresi untuk *Mean Squared Error (MSE)*

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

3. *Root Mean Squared Error (RMSE)*

   Root Mean Squared Error (RMSE) merupakan akar kuadrat dari MSE. RMSE mengembalikan satuan error ke skala yang sama dengan data asli, sehingga lebih mudah diinterpretasikan dibandingkan MSE. Namun, seperti MSE, RMSE tetap sensitif terhadap kesalahan besar. Berikut adalah rumus metrik evaluasi regresi untuk *Root Mean Squared Error (RMSE)*

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

4. *R-Squared (R^2)*

   R-Squared (R^2) atau koefisien determinasi mengukur proporsi variasi data aktual yang dapat dijelaskan oleh model regresi. Nilai R^2 berada pada

rentang 0 hingga 1, di mana nilai yang mendekati 1 menunjukkan bahwa model memiliki kemampuan prediksi yang baik. Namun, R^2 tidak selalu mencerminkan kesalahan prediksi secara langsung dan dapat menyesatkan jika digunakan tanpa metrik *error* lainnya. Berikut adalah rumus metrik evaluasi regresi untuk *R-Squared (R^2)*

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

Keterangan simbol dari rumus-rumus metrik evaluasi kinerja model regresi.

y_i = nilai aktual ke-i
$\hat{y}_i$ = nilai prediksi ke-i
$\bar{y}$ = rata-rata nilai aktual
n = jumlah data
Σ = penjumlahan

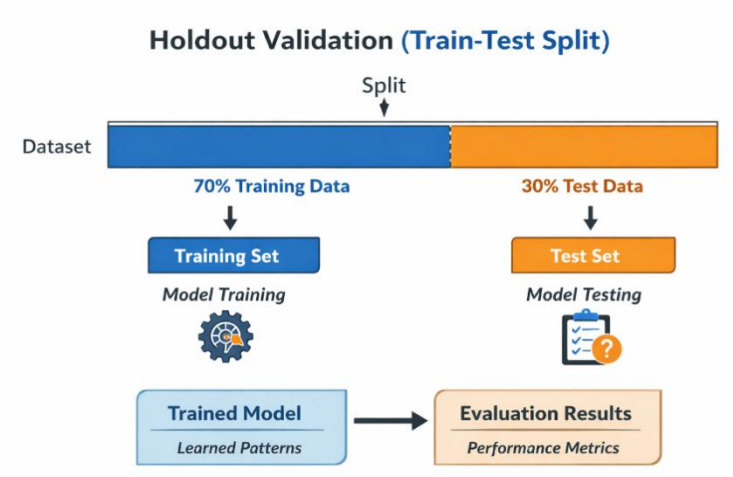
13.2 Metode Validasi Model

Validasi model adalah serangkaian teknik yang digunakan untuk menilai seberapa baik model machine learning dapat menggeneralisasi terhadap data yang belum pernah dilihat sebelumnya. Tujuan utama validasi model menurut Qiu (2024) adalah untuk memperoleh estimasi performa model yang stabil, tidak bias, dan dapat diandalkan sehingga model tersebut layak digunakan dalam aplikasi nyata. Tanpa validasi yang tepat, perkiraan performa model hanya berdasarkan data pelatihan dapat menyesatkan dan tidak mencerminkan kemampuan model pada data baru.

13.2.1 *Holdout Validation (Train-Test Split)*

Holdout validation atau *train-test split* adalah metode paling sederhana dalam validasi model. Caranya dengan membagi dataset menjadi dua bagian (School of Information Systems BINUS University, 2024), yaitu:

1. Data pelatihan (*training set*): digunakan untuk melatih atau fit model.
2. Data uji (*test set*): digunakan untuk mengevaluasi performa model setelah pelatihan.



Gambar 13. 1 *Holdout Validation (Train-Test Split)*
(Sumber : Qiu, J. 2024)

Rasio umum pembagian adalah 70:30 atau 80:20 antara data pelatihan dan uji. Metode ini cepat dan mudah diimplementasikan, tetapi hasilnya dapat sensitif terhadap cara pembagian data. Jika pembagian tidak representatif, estimasi performa bisa bias tinggi dan menghasilkan model yang kurang stabil. Kelebihannya adalah cepat, cocok saat dataset besar. Kekurangannya adalah performa estimasi bisa bervariasi tergantung

pada pemilihan subset; tidak memaksimalkan penggunaan data untuk pelatihan

13.2.2 *Cross-Validation* (Validasi Silang)

Cross-validation adalah teknik yang lebih canggih dibanding metode holdout karena membagi dataset menjadi beberapa bagian (folds) dan melakukan pelatihan serta validasi secara bergantian pada setiap kombinasi *fold*. Tujuannya adalah mendapatkan estimasi performa yang lebih stabil dan kurang tergantung pada pembagian data semata (GeeksforGeeks, 2025).

1. *k-Fold Cross-Validation*

   Pada k-fold cross-validation, dataset dibagi menjadi k subset yang berukuran sama. Model dilatih pada k-1 subset dan diuji pada subset yang tersisa. Proses ini diulang k kali sehingga setiap subset menjadi data uji sekali, lalu hasil metrik performa dirata-ratakan
2. *Stratified k-Fold Cross-Validation*

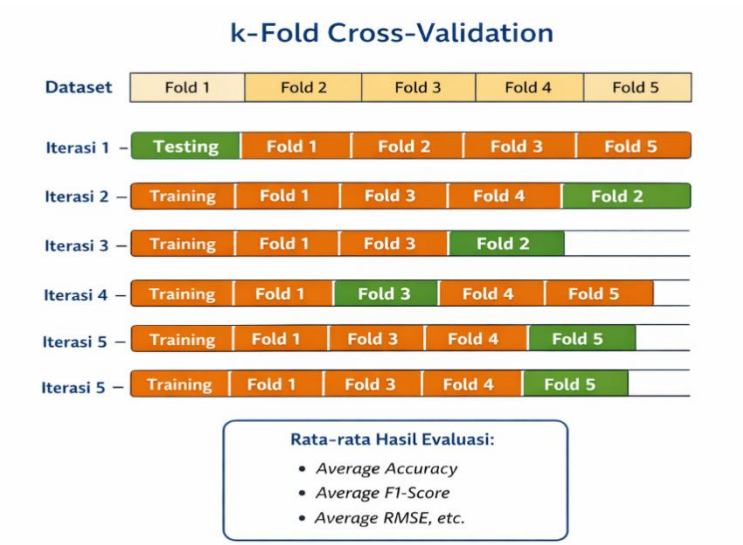
   Versi k-fold ini menjaga proporsi label kelas dalam setiap fold agar sesuai dengan distribusi asli dataset. Teknik ini sangat penting untuk kasus klasifikasi dengan data imbalanced karena fold yang menyeragamkan distribusi kelas akan menghasilkan estimasi performa yang lebih akurat.
3. *Leave-One-Out Cross-Validation (LOOCV)*

   LOOCV adalah kasus khusus k-fold dengan k = jumlah sampel. Setiap iterasi hanya satu sampel yang menjadi data uji sementara sisanya digunakan untuk pelatihan. Teknik ini memaksimalkan penggunaan data, namun biaya komputasi sangat tinggi untuk dataset besar dan bisa menyebabkan varians evaluasi besar.

4. *Repeated & Nested Cross-Validation*

   Repeated Cross-Validation: k-fold diulang beberapa kali dengan pengacakan berbeda untuk mengurangi varians estimasi performa lebih jauh dan Nested Cross-Validation: Teknik ini digunakan saat melakukan tuning hyperparameter sekaligus evaluasi performa agar tidak terjadi bias dalam pemilihan model serta optimasi parameter.

Berikut diagram ilustrasi proses *Cross-Validation* (*k-Fold Cross-Validation)* yang disajikan secara konseptual pada gambar 13.2



Gambar 13. 2 *k-Fold Cross-Validation*

Gambar 13.2 tersebut mengilustrasikan proses *k-Fold Cross-Validation,* yaitu metode validasi model yang membagi dataset menjadi beberapa bagian *(fold)* dengan ukuran yang relatif sama. Pada ilustrasi ini, dataset dibagi menjadi lima fold, di mana setiap fold digunakan

secara bergantian sebagai data uji (*testing*), sementara fold lainnya digunakan sebagai data latih (*training*).

Pada setiap iterasi, model dilatih menggunakan data latih dan kemudian dievaluasi menggunakan data uji untuk menghasilkan nilai metrik kinerja, seperti *accuracy, F1-score, atau RMSE*. Setelah seluruh iterasi selesai, nilai evaluasi dari masing-masing fold dirata-ratakan untuk memperoleh performa akhir model. Pendekatan ini menghasilkan estimasi kinerja yang lebih stabil dan representatif terhadap kemampuan model dalam melakukan generalisasi pada data baru, sehingga *k-Fold Cross-Validation* banyak digunakan dalam praktik dan penelitian *machine learning*

```
#Dataset Awal

| Fold1 | Fold2 | Fold3 | Fold4 | Fold5 |

#Iterasi 1
Training Data : Fold2 + Fold3 + Fold4 + Fold5
Testing Data  : Fold1
#Iterasi 2
Training Data : Fold1 + Fold3 + Fold4 + Fold5
Testing Data  : Fold2
#Iterasi 3
Training Data : Fold1 + Fold2 + Fold4 + Fold5
Testing Data  : Fold3
#Iterasi 4
Training Data : Fold1 + Fold2 + Fold3 + Fold5
Testing Data  : Fold4
#Iterasi 5
Training Data : Fold1 + Fold2 + Fold3 + Fold4
Testing Data  : Fold5
#Performa Akhir Model =
Rata-rata (Metrik Fold1 + Fold2 + Fold3 + Fold4 +
Fold5)
Dataset Asli (Imbalanced)
Kelas A : 80%
Kelas B : 20%
#Dataset Asli (Imbalanced)
Kelas A : 80%
Kelas B : 20%
#Setiap Fold:
Fold1 → A:80% | B:20%
Fold2 → A:80% | B:20%
Fold3 → A:80% | B:20%
..
```

Gambar 13. 3 *k-Fold Cross-Validation Process in Python*

13.2.3 Pentingnya Validasi dalam *Preventing Overfitting*

Validasi model membantu mendeteksi jika model *overfitting* yaitu kondisi di mana model sangat sesuai terhadap data pelatihan tetapi gagal pada data baru. Metode *cross-validation* secara khusus dirancang untuk menilai kestabilan model di berbagai subset data sehingga dapat mengidentifikasi tanda awal overfitting

jika performa model bervariasi tajam antar fold (Qiu, J. 2024).

13.2.4 Konteks Pemilihan Teknik Validasi

Metode validasi model adalah bagian penting dari alur kerja *machine learning* yang memastikan model tidak hanya belajar dari data pelatihan, tetapi juga dapat melakukan generaliasi ke data baru dengan baik. *Holdout validation* memberikan metode sederhana dan cepat, sementara berbagai teknik *cross-validation* menawarkan estimasi performa yang lebih andal terlepas dari pembagian data awal. Pemilihan metode validasi harus mempertimbangkan ukuran data, tujuan aplikasi model, dan sumber daya komputasi yang tersedia (School of Information Systems BINUS University, 2024).

Pemilihan teknik validasi bergantung pada beberapa faktor, diantaranya: Jumlah dataset: Dataset kecil lebih cocok dengan cross-validation untuk memaksimalkan data, Tugas ML: Untuk dataset dengan kelas yang tidak seimbang, stratified *cross-validation* sering lebih efektif dan Sumber daya komputasi: Jika biaya komputasi menjadi batas utama, metode *holdout* sederhana dengan rasio dan random state yang tepat bisa menjadi alternatif.

13.3 *Overfitting* dan *Underfitting*

Dalam pengembangan model machine learning, tujuan utama adalah menciptakan model yang mampu melakukan generalisasi, yakni memberikan prediksi yang baik pada data yang belum pernah dilihat sebelumnya. Namun, dua kondisi yang sering menjadi hambatan utama dalam mencapai generalisasi yang baik adalah *overfitting* dan *underfitting.*

Berbagai metrik evaluasi dan visualisasi seperti *accuracy, F1-score, ROC Curve, dan Precision–Recall Curve* mampu memberikan gambaran kuantitatif dan visual tentang kinerja model, hasil evaluasi tersebut tidak selalu mencerminkan kemampuan generalisasi model secara menyeluruh. Nilai metrik yang tinggi pada data uji tertentu bisa saja menutupi masalah mendasar dalam proses pembelajaran model, terutama jika model terlalu menyesuaikan diri dengan data pelatihan atau justru gagal menangkap pola yang ada.

Kondisi inilah yang mengarah pada dua permasalahan klasik dalam machine learning, yaitu *overfitting* dan *underfitting. Overfitting* terjadi ketika model menunjukkan performa sangat baik pada data pelatihan tetapi mengalami penurunan kinerja yang signifikan pada data baru, sedangkan *underfitting* muncul ketika model terlalu sederhana sehingga tidak mampu mempelajari pola penting dari data. Kedua kondisi ini sering kali tidak dapat diidentifikasi hanya dengan satu metrik evaluasi, melainkan memerlukan analisis perbandingan performa antara data pelatihan dan data validasi

Overfitting adalah kondisi ketika *model machine learning* terlalu menyesuaikan diri dengan data pelatihan sehingga tidak hanya mempelajari pola umum, tetapi juga menangkap noise dan variasi acak yang tidak merepresentasikan fenomena sebenarnya. Akibatnya, model memiliki kinerja sangat baik pada data pelatihan, namun kinerja menurun secara signifikan pada data validasi atau data uji, yang menunjukkan kegagalan dalam melakukan generalisasi (Zhang et al., 2022). Fenomena overfitting umumnya terjadi pada model dengan kompleksitas tinggi, jumlah parameter besar, atau ketika jumlah data pelatihan relatif terbatas dibandingkan kapasitas model.

Qiu (2024) menjelaskan bahwa *underfitting* merupakan kondisi ketika model terlalu sederhana sehingga gagal menangkap pola utama yang terkandung dalam data, baik pada tahap pelatihan maupun pengujian. Model yang mengalami underfitting menunjukkan *error* yang tinggi secara konsisten pada data pelatihan dan data uji, yang menandakan bahwa kapasitas model tidak memadai untuk merepresentasikan hubungan antara fitur dan target.

13.3.1 Analisis *Bias–Varians* (Perspektif Teoretis)

Permasalahan *overfitting* dan *underfitting* dalam machine learning dapat dipahami secara konseptual melalui *trade-off* bias varians, yang merupakan kerangka fundamental dalam analisis kesalahan prediksi model. *Trade-off* ini menjelaskan bagaimana kompleksitas model memengaruhi kemampuan generalisasi terhadap data baru.

Bias tinggi (*high bias*) terjadi ketika model terlalu sederhana sehingga tidak mampu menangkap pola mendasar yang terdapat dalam data. Kondisi ini menyebabkan kesalahan prediksi yang besar baik pada data pelatihan maupun data pengujian, yang secara umum dikenal sebagai underfitting. Model dengan bias tinggi biasanya memiliki asumsi yang terlalu kuat atau struktur yang terlalu terbatas, sehingga gagal merepresentasikan hubungan yang sebenarnya antara fitur dan variabel target. Literatur menyatakan bahwa underfitting berkaitan erat dengan keterbatasan kapasitas model dan pemilihan fitur yang kurang informatif (Zhang et al., 2022).

Sebaliknya, varians tinggi (*high variance*) muncul ketika model terlalu kompleks dan sangat sensitif terhadap fluktuasi kecil dalam data pelatihan. Model dalam kondisi ini cenderung mempelajari tidak hanya pola umum, tetapi juga *noise* pada data, sehingga

menghasilkan performa yang sangat baik pada data pelatihan namun menurun secara signifikan pada data pengujian. Fenomena ini dikenal sebagai *overfitting* dan menunjukkan kegagalan model dalam melakukan generalisasi. Studi empiris menunjukkan bahwa overfitting sering terjadi pada model dengan jumlah parameter besar atau ketika jumlah data pelatihan tidak mencukupi dibandingkan kompleksitas model (Sekeroglu et al., 2022).

Oleh karena itu, tujuan utama dalam pengembangan model machine learning adalah menemukan titik keseimbangan antara bias dan varians, di mana model cukup fleksibel untuk menangkap pola penting dalam data namun tetap mampu melakukan generalisasi dengan baik. Pendekatan seperti regularisasi, validasi silang (*cross-validation*), dan pemilihan *hyperparameter* secara sistematis direkomendasikan dalam literatur untuk mengelola *trade-off bias–varians* ini secara efektif (Qiu, 2024).

13.3.2 Ciri - Ciri dan Penyebab Umum *Overfitting* & *Underfitting*

1. Ciri-Ciri *Overfitting* dan *Underfitting*

   Overfitting terjadi ketika model terlalu menyesuaikan diri dengan data pelatihan. Secara empiris, kondisi ini dapat diamati dari nilai kesalahan pada data pelatihan yang sangat rendah, sementara kesalahan pada data validasi atau uji meningkat secara signifikan. Literatur menyebutkan bahwa model dalam kondisi *overfitting* tidak hanya mempelajari pola utama, tetapi juga noise atau fluktuasi acak dalam data, sehingga performanya tidak stabil pada data baru.

Dari sisi visualisasi, overfitting ditandai oleh kurva training *error* yang terus menurun, sedangkan validation *error* mulai meningkat setelah titik tertentu, yang mengindikasikan bahwa peningkatan kompleksitas model justru menurunkan kemampuan generalisasi. Buku teks Pattern Recognition and Machine Learning menegaskan bahwa perilaku ini merupakan ciri klasik dari model dengan varians tinggi (*high variance*) (Bishop, 2006).

Selain itu, kesenjangan yang besar antara nilai metrik evaluasi (misalnya accuracy atau loss) pada data latih dan data uji juga menjadi indikator kuat terjadinya *overfitting* (Goodfellow et al., 2016).

Underfitting terjadi ketika model terlalu sederhana sehingga gagal menangkap struktur dasar data. Kondisi ini ditandai oleh nilai error yang tetap tinggi baik pada data pelatihan maupun data pengujian, yang menunjukkan bahwa model tidak mampu mempelajari pola secara efektif.

Dalam kurva pembelajaran (*learning curve*), *underfitting* terlihat dari kurva *training* error yang stagnan dan tidak mengalami penurunan signifikan, meskipun jumlah data atau iterasi pelatihan ditambah. Literatur menyatakan bahwa fenomena ini berkaitan dengan bias yang tinggi (*high bias*), di mana asumsi model terlalu sederhana untuk merepresentasikan hubungan antarvariabel (Hastie et al., 2017).

2. Penyebab Umum *Overfitting* dan *Underfitting*

Secara umum, *overfitting* disebabkan oleh kompleksitas model yang berlebihan dibandingkan dengan jumlah dan kualitas data yang tersedia. Model seperti pohon keputusan dengan kedalaman besar atau jaringan saraf dengan banyak parameter sangat rentan mengalami *overfitting* apabila tidak

diimbangi dengan data yang memadai. Selain itu, keberadaan noise dan fitur yang tidak relevan juga memperbesar risiko *overfitting* jika tidak ditangani melalui seleksi fitur atau regularisasi (Bishop, 2006; Goodfellow et al., 2016).

Underfitting umumnya disebabkan oleh model yang terlalu sederhana, penggunaan regularisasi yang terlalu kuat, atau fitur input yang tidak cukup informatif. Dalam kondisi ini, model tidak memiliki kapasitas yang memadai untuk mempelajari pola kompleks dalam data, sehingga performanya buruk pada seluruh tahap evaluasi. Hastie et al. (2017) menekankan bahwa pemilihan model dan representasi fitur yang tepat merupakan faktor kunci untuk menghindari *underfitting.*

13.3.3 Strategi Deteksi dengan Evaluasi & Kurva Pembelajaran

Deteksi *overfitting* dan *underfitting* dalam *machine learning* tidak dapat dilakukan hanya dengan mengamati satu nilai metrik evaluasi secara terpisah. Literatur menegaskan bahwa interpretasi kinerja model harus dilakukan dengan membandingkan pola performa antara data pelatihan dan data validasi atau pengujian. Pendekatan ini bertujuan untuk menilai kemampuan generalisasi model terhadap data yang belum pernah dilihat sebelumnya.

Salah satu metode visual yang paling banyak digunakan untuk tujuan tersebut adalah kurva pembelajaran (*learning curves*), yang memvisualisasikan perubahan nilai loss atau akurasi pada data pelatihan dan validasi seiring dengan bertambahnya jumlah data atau iterasi pelatihan. Menurut Géron (2022), learning curves memberikan indikasi langsung mengenai apakah

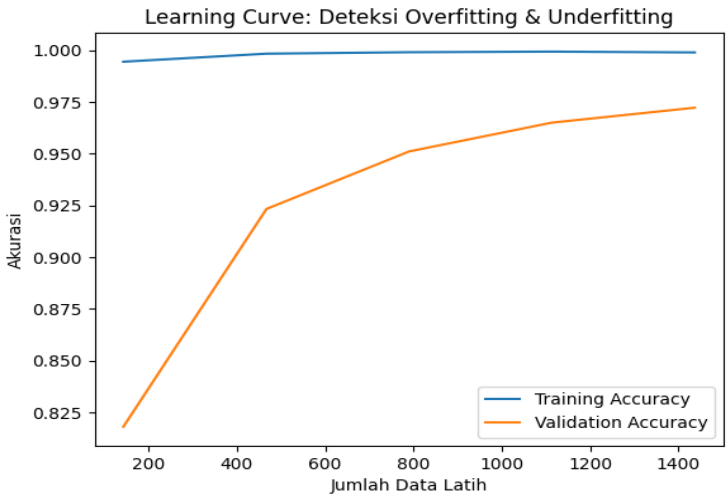
model mengalami keterbatasan kapasitas atau justru terlalu kompleks terhadap data yang tersedia.

Dalam konteks *overfitting*, kurva pembelajaran menunjukkan pola di mana *training loss* terus menurun atau *training accuracy* terus meningkat, sementara *validation loss* mulai meningkat atau *validation accuracy* stagnan bahkan menurun setelah titik tertentu. Pola ini menunjukkan bahwa model semakin menyesuaikan diri dengan data pelatihan, namun gagal mempertahankan performa pada data validasi. Literatur statistik pembelajaran menyebutkan bahwa kondisi ini mencerminkan varians yang tinggi, di mana model sangat sensitif terhadap karakteristik spesifik data latih (James et al., 2021).

Sebaliknya, *underfitting* ditandai oleh pola kurva pembelajaran di mana *training loss* dan *validation loss* sama-sama tinggi dan relatif sejajar, serta tidak menunjukkan penurunan signifikan meskipun proses pelatihan dilanjutkan. Kondisi ini mengindikasikan bahwa model tidak memiliki kompleksitas yang cukup untuk menangkap struktur dasar data. Dalam hal ini, model mengalami bias tinggi, yang menyebabkan kesalahan sistematis baik pada data pelatihan maupun data pengujian (Shalev-Shwartz & Ben-David, 2014).

Lebih lanjut, buku teks pembelajaran mesin modern menekankan bahwa jarak (*gap*) antara kurva pelatihan dan validasi menjadi indikator utama dalam diagnosis model. *Gap* besar mengindikasikan *overfitting*, sedangkan gap kecil tetapi *erro*r tinggi menunjukkan *underfitting.* Oleh karena itu, learning curves berfungsi sebagai alat diagnostik yang esensial dalam evaluasi model dan pengambilan keputusan terkait kompleksitas model, regularisasi, dan kebutuhan data tambahan (Géron, 2022). Berikut Script 13.2 dan Gambar 13.3

menjelaskan contoh pola *overfitting* dan *underfitting* secara empiris.



Gambar 13. 4 Grafik *Learning Curve*

```python
from sklearn.datasets import load_digits
from sklearn.model_selection import learning_curve
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import numpy as np
# Load dataset
X, y = load_digits(return_X_y=True)
# Model
model = SVC(kernel="rbf", gamma=0.001)
# Learning curve
train_sizes, train_scores, val_scores = learning_curve(
    model, X, y, cv=5, scoring="accuracy",
    train_sizes=np.linspace(0.1, 1.0, 5))
# Mean scores
train_mean = train_scores.mean(axis=1)
val_mean = val_scores.mean(axis=1)
# Plot learning curve
plt.figure()
plt.plot(train_sizes, train_mean, label="Training
Accuracy")
plt.plot(train_sizes, val_mean, label="Validation
Accuracy")
plt.xlabel("Jumlah Data Latih")
plt.ylabel("Akurasi")
plt.title("Learning Curve: Deteksi Overfitting &
Underfitting")
plt.legend()
plt.show()
```

Gambar 13. 5 Kode Python: *Learning Curve*

Learning curve pada Gambar 13.3 memperlihatkan hubungan antara jumlah data latih dan kinerja model pada data pelatihan serta data validasi. Kurva biru menunjukkan akurasi pada data latih, sedangkan kurva oranye menunjukkan akurasi pada data validasi. Pada ukuran data latih yang kecil, terlihat adanya selisih akurasi yang cukup besar antara data latih dan data validasi. Kondisi ini mengindikasikan kecenderungan overfitting, di mana model sangat baik dalam mengklasifikasikan data latih tetapi belum

mampu melakukan generalisasi secara optimal pada data yang tidak terlihat.

Seiring bertambahnya jumlah data latih, akurasi pada data validasi meningkat dan jarak antara kedua kurva semakin menyempit. Fenomena ini menunjukkan bahwa penambahan data membantu model mempelajari pola yang lebih umum dan mengurangi efek *overfitting*. Pada titik ini, model mulai mencapai keseimbangan antara bias dan varians, yang merupakan kondisi ideal dalam proses pembelajaran mesin.

Apabila kedua kurva berhenti pada nilai akurasi yang rendah dan saling berdekatan, hal tersebut akan menandakan *underfitting,* yaitu model terlalu sederhana untuk menangkap kompleksitas data. Namun, pada grafik ini, performa validasi yang terus meningkat menegaskan bahwa model memiliki kapasitas yang memadai dan memperoleh manfaat signifikan dari penambahan data latih.

13.3.4 Teknik Mengatasi *Overfitting* dan *Underfitting*

Setelah model *machine learning* dibangun dan dievaluasi dengan metrik yang tepat, langkah penting berikutnya adalah mengoptimalkan kinerjanya agar prediksi yang dihasilkan lebih akurat, stabil, dan sesuai kebutuhan aplikasi nyata. Optimalisasi ini mencakup berbagai strategi, mulai dari pemilihan fitur hingga penyetelan parameter model (*hyperparameter tuning*) dan teknik lain seperti *ensemble learning*. Dalam konteks machine learning modern, kombinasi teknik-teknik ini sering diterapkan untuk meningkatkan performa di berbagai domain (Nurzatil Aqmar et al., 2025).

DAFTAR PUSTAKA

A systematic literature review on Machine Learning Model evaluation on healthcare

applications. 2023. *Research, Society and Development*, 12(6), e5412642042. https://doi.org/10.33448/rsd-v12i6.42042.

Bishop, C.M. 2006. *Pattern recognition and machine learning*. Springer.

Géron, A. 2022. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. 3rd edn. O'Reilly Media.

GeeksforGeeks. 2025. Cross Validation in Machine Learning. Available at: https://www.geeksforgeeks.org/machine-learning/cross-validation-machine-learning/ (Accessed: 25 December 2025).

Goodfellow, I., Bengio, Y. and Courville, A. 2016. *Deep learning*. MIT Press.

Hastie, T., Tibshirani, R. and Friedman, J. 2017. *The elements of statistical learning: Data mining, inference, and prediction.* 2nd edn. Springer.

James, G., Witten, D., Hastie, T. and Tibshirani, R. 2021. *An introduction to statistical learning: With applications in R*. 2nd edn. Springer.

Nurzatil Aqmar, Wijayanto, H. and Afendi, F.M. 2025. Performance analysis of machine learning models using RFE feature selection and Bayesian optimization in imbalanced data classification with SHAP-based explanations. *Scientific Journal of Informatics*, 12(3), pp. 539–554. Available at:

https://journal.unnes.ac.id/journals/sji/article/view/31459.

Qiu, J. 2024. An analysis of model evaluation with cross-validation: Techniques, applications, and recent advances, *Advances in Economics Management and Political Sciences,* 99(1), pp. 69–72. https://doi.org/10.54254/2754-1169/99/2024OX0213.

Sathyanarayanan, S. and Tantri, B.R. 2025. Confusion matrix–based performance evaluation metrics. *African Journal of Biomedical Research*. https://doi.org/10.53555/AJBR.v27i4S.4345.

Sayin, B., Yang, J., Chen, X. et al. 2025. Rethinking and recomputing the value of machine learning models. *Artificial Intelligence Review*, 58, 238. https://doi.org/10.1007/s10462-025-11242-6.

School of Information Systems BINUS University. 2024*. Train validate test dan cross validation.* Available at: https://sis.binus.ac.id/2024/11/01/train-validate-test-dan-cross-validation/ (Accessed: 25 December 2025).

Sekeroglu, B., Ever, Y.K., Dimililer, K. and Al-Turjman, F. 2022. Comparative evaluation and comprehensive analysis of machine learning models for regression problems. *Data Intelligence*, 4(3), pp. 620–652. https://doi.org/10.1162/dint_a_00155.

Shalev-Shwartz, S. and Ben-David, S. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge University Press.

Zhang, Y., Chen, X. and Wang, L. 2022. Rethinking and recomputing the value of machine learning models. *Artificial Intelligence Review*, 55, pp. 1–20. https://doi.org/10.1007/s10462-022-10105-8.

BAB 14

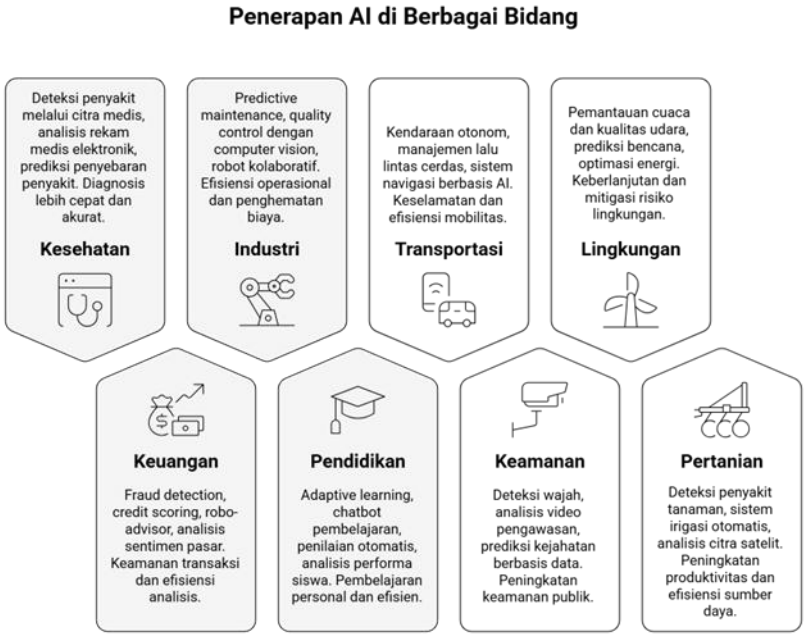
APLIKASI AI DALAM BERBAGAI BIDANG (KESEHATAN, KEUANGAN, INDUSTRI, PENDIDIKAN)

14.1 Transformasi Lintas Sektor Melalui Kecerdasan Buatan

Kecerdasan buatan (*Artificial Intelligence*/AI) kini menjadi salah satu pilar utama dalam perkembangan teknologi modern. Keberadaannya telah membawa perubahan besar dalam cara manusia bekerja, berpikir, dan berinteraksi dengan sistem digital. AI memungkinkan komputer untuk meniru kemampuan berpikir manusia, seperti belajar, menalar, mengenali pola, hingga membuat keputusan berdasarkan data (Budiasto, Tallulembang, *et al.*, 2025). Melalui integrasi dengan *machine learning* dan *big data*, teknologi ini tidak hanya mampu menganalisis informasi dalam jumlah besar, tetapi juga menghasilkan prediksi yang akurat untuk mendukung pengambilan keputusan di berbagai bidang (Baskoro *et al.*, 2025; Venkateswaran and Sriramkumar, 2025).

Dalam era industri 4.0 dan masyarakat 5.0, AI menjadi motor utama transformasi digital. Hampir semua sektor kini memanfaatkan teknologi ini untuk meningkatkan efisiensi dan kualitas layanan (Rashid and Kausik, 2024). Sebagai

contoh, di bidang kesehatan, AI membantu dokter dalam mendiagnosis penyakit dan menentukan terapi yang sesuai (Yaakub *et al.*, 2025). Di sektor keuangan, algoritma cerdas digunakan untuk mendeteksi penipuan dan mengelola risiko (Nampira *et al.*, 2025). Dalam industri manufaktur, AI berperan dalam otomasi proses produksi dan perawatan mesin secara prediktif (Budiasto, Tallulembang, *et al.*, 2025; Sani *et al.*, 2025). Sementara di dunia pendidikan, AI menghadirkan sistem pembelajaran adaptif yang menyesuaikan materi dengan kemampuan setiap siswa (Budiasto, Purnama, *et al.*, 2025; Yaakub *et al.*, 2025). Gambar 14.1 memberikan visualisasi yang komprehensif mengenai ragam aplikasi AI di berbagai sektor ini, termasuk di dalamnya transportasi, lingkungan, keamanan, dan pertanian.



Gambar 14. 1 Penerapan AI di Berbagai Bidang

Penerapan AI yang luas ini tidak terlepas dari kemajuan teknologi pendukung seperti komputasi awan dan *Internet of Things* (IoT), yang memungkinkan pengumpulan dan pengolahan data secara cepat dan real time (Saadia, 2021). Namun, seiring dengan peluang besar yang ditawarkan, tantangan baru juga muncul. Masalah seperti bias algoritma, privasi data, serta tanggung jawab etis menjadi perhatian penting dalam pengembangan sistem AI yang andal dan adil (Grzybowski, Jin and Wu, 2024; Budiasto, Jayawardana, *et al.*, 2025).

Bab ini akan mengulas berbagai penerapan AI di beberapa sektor utama: kesehatan, keuangan, industri, dan pendidikan. Melalui pembahasan ini, pembaca diharapkan dapat memahami bagaimana teori dan algoritma yang dipelajari pada bab sebelumnya diterapkan dalam kehidupan nyata, serta bagaimana AI menjadi kekuatan yang mengubah cara dunia beroperasi di era digital.

14.2 AI dalam Bidang Kesehatan (*Healthcare*)

Perkembangan kecerdasan buatan (AI) telah membawa perubahan besar dalam dunia kesehatan modern. Dengan kemampuannya menganalisis data medis dalam jumlah besar dan mengenali pola yang sulit dideteksi manusia, AI kini menjadi alat bantu penting dalam diagnosis, pengobatan, serta pemantauan pasien. Teknologi ini membantu para profesional medis membuat keputusan yang lebih cepat, tepat, dan berbasis data, sehingga kualitas layanan kesehatan meningkat secara signifikan (Kumar, Sareen and Arora, 2023).

Salah satu penerapan utama AI dalam bidang ini adalah analisis citra medis. Algoritma *deep learning*, terutama *Convolutional Neural Network* (CNN), mampu mengidentifikasi kelainan pada hasil pemindaian seperti X-ray, MRI, atau CT scan dengan akurasi yang tinggi (Manolescu,

Buckley and Secco, 2024). Misalnya, sistem yang dikembangkan oleh DeepMind Health dapat mendeteksi berbagai penyakit mata hanya dari citra retina, dengan hasil diagnosis yang sebanding dengan dokter spesialis. Inovasi ini mempercepat proses deteksi dini penyakit dan memungkinkan pengobatan dilakukan sebelum kondisi menjadi parah.

Selain untuk diagnosis, AI juga berperan dalam analisis data rekam medis elektronik (*Electronic Health Records*/EHR). Melalui algoritma pembelajaran mesin, sistem dapat mengekstraksi informasi penting dari catatan pasien untuk memprediksi risiko penyakit kronis seperti diabetes atau serangan jantung (Rashid *et al.*, 2022). Pendekatan ini dikenal sebagai *predictive analytics*, yang membantu dokter memberikan perawatan preventif berdasarkan data riil pasien.

Di bidang penemuan obat (*drug discovery*), AI mempercepat proses penelitian dengan memprediksi interaksi molekul dan mengidentifikasi senyawa potensial sebelum dilakukan uji klinis (Bachhar *et al.*, 2025). Platform seperti Atomwise dan Insilico Medicine menggunakan *deep learning* untuk menyaring ribuan kandidat obat, sehingga waktu penelitian dapat dipangkas dari bertahun-tahun menjadi hitungan bulan (Chowdhury *et al.*, 2025). Inovasi ini terbukti sangat membantu, terutama pada masa pandemi COVID-19, ketika kecepatan penemuan terapi menjadi faktor krusial.

Tabel 14. 1 Penerapan AI di Berbagai Bidang Kesehatan

Bidang Penerapan	Teknologi AI yang Digunakan	Contoh Implementasi	Manfaat Utama
Diagnosis Penyakit	CNN, Deep Learning	Deteksi kanker payudara, pneumonia	Akurasi diagnosis tinggi
Analisis EHR	NLP, Predictive Analytics	Prediksi risiko diabetes dan serangan jantung	Pencegahan dini penyakit
Penemuan Obat	Deep Neural Network	Atomwise, Insilico Medicine	Percepatan riset farmasi
Kesehatan Mental	NLP, Sentiment Analysis	Chatbot terapi seperti Wysa, Woebot	Dukungan psikologis digital
Telemedicine	AI Chatbot, Computer Vision	Deteksi gejala jarak jauh	Akses kesehatan lebih luas

Selain itu, perkembangan perangkat *wearable* dan sistem pemantauan berbasis AI memungkinkan pengawasan kesehatan pasien dilakukan secara real-time. Sensor pada jam tangan pintar dapat mengukur detak jantung, kadar oksigen, dan pola tidur, lalu menganalisis data tersebut untuk mendeteksi gangguan kesehatan lebih awal (Kanakaprabha *et al.*, 2024). Beberapa sistem rehabilitasi pasien stroke bahkan memanfaatkan robot terapi yang dikendalikan oleh algoritma *reinforcement learning* agar dapat menyesuaikan intensitas latihan secara adaptif terhadap kemampuan pasien.

Namun, di balik berbagai kemajuan tersebut, penerapan AI di bidang kesehatan juga menghadapi tantangan. Isu

privasi data medis, keamanan informasi pasien, serta bias algoritmik masih menjadi perhatian utama (Grzybowski, Jin and Wu, 2024). Data kesehatan bersifat sangat sensitif, sehingga penggunaannya harus diatur secara ketat untuk menghindari penyalahgunaan. Selain itu, model AI perlu dilatih dengan data yang beragam agar tidak menghasilkan keputusan yang diskriminatif (Kamikubo *et al.*, 2022).

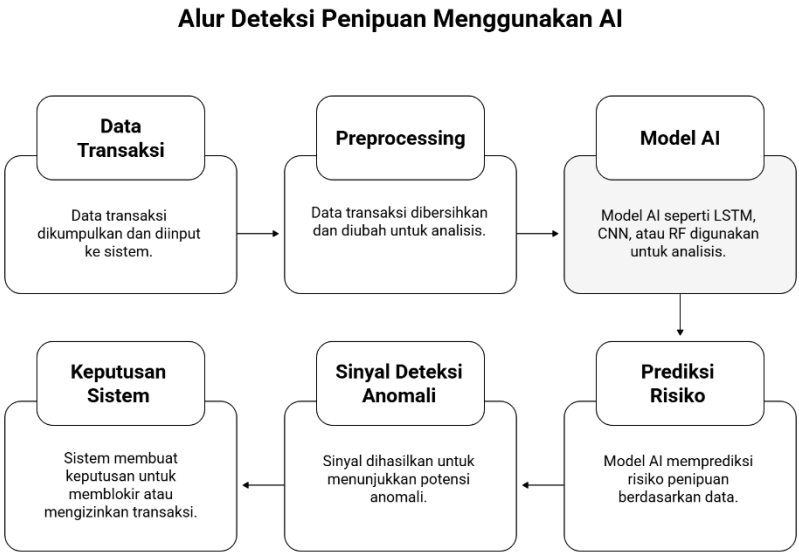
Secara keseluruhan, AI telah membawa era baru bagi dunia kesehatan—sebuah era di mana teknologi tidak menggantikan dokter, melainkan menjadi mitra cerdas yang membantu mereka bekerja dengan lebih efisien dan akurat. Dengan perkembangan yang berkelanjutan, kolaborasi antara tenaga medis, peneliti, dan teknologi AI akan terus membuka peluang baru menuju sistem kesehatan yang lebih personal, prediktif, dan manusiawi.

14.3 AI dalam Bidang Keuangan (*Finance*)

Kecerdasan buatan telah mengubah wajah industri keuangan secara menyeluruh. Di era digital ini, lembaga keuangan tidak lagi hanya bergantung pada analisis manual atau intuisi manusia dalam membuat keputusan, tetapi mulai memanfaatkan kemampuan AI untuk memproses data besar secara cepat, akurat, dan berkelanjutan. Melalui algoritma machine learning, sistem dapat mengenali pola transaksi, memprediksi tren pasar, serta mendeteksi potensi risiko jauh sebelum manusia menyadarinya (Nampira *et al.*, 2025).

Salah satu penerapan paling penting dari AI di sektor ini adalah pendeteksian penipuan (*fraud detection*), sebagaimana diilustrasikan pada Gambar 14.2. Setiap hari jutaan transaksi keuangan terjadi, dan sistem berbasis AI mampu memantau aktivitas tersebut secara real-time (Vetrivel *et al.*, 2024). Model pembelajaran mesin seperti *Random Forest* dan *Deep Neural Network* digunakan untuk mengenali pola-pola mencurigakan yang menyimpang dari

perilaku normal pengguna. Misalnya, PayPal dan Mastercard telah berhasil memanfaatkan teknologi ini untuk menurunkan angka penipuan dengan akurasi yang semakin tinggi.



Gambar 14. 2 Ilustrasi Alur Deteksi Penipuan (*Fraud Detection*)

Selain aspek keamanan, AI juga berperan besar dalam penilaian risiko dan analisis kredit (Budiasto, Tallulembang, *et al.*, 2025). Dahulu, bank hanya menilai kelayakan pinjaman berdasarkan data historis dan rasio keuangan sederhana. Kini, sistem berbasis *machine learning* mampu menilai risiko dengan mempertimbangkan ratusan variabel, termasuk perilaku digital dan pola pengeluaran pengguna (Ramakrishnan *et al.*, 2024). Contohnya, JPMorgan Chase memanfaatkan algoritma *explainable AI* untuk menjelaskan keputusan kredit agar tetap transparan dan sesuai regulasi.

Dalam dunia investasi, AI menjadi otak di balik perdagangan otomatis (*automated trading*) yang bereaksi terhadap perubahan pasar dalam hitungan detik. Algoritma seperti *Reinforcement Learning* dan *Long Short-Term Memory (LSTM)* digunakan untuk menyesuaikan strategi perdagangan secara adaptif, berdasarkan data harga, berita, dan sentimen pasar (Zou *et al.*, 2024). Sistem ini digunakan oleh perusahaan besar seperti Goldman Sachs dan BlackRock untuk mengoptimalkan portofolio investasi dan meminimalkan risiko kerugian.

Penerapan lain yang kini semakin populer adalah AI dalam layanan pelanggan. Chatbot keuangan seperti Erica dari Bank of America atau Cleo AI memanfaatkan *Natural Language Processing (NLP)* untuk memberikan layanan konsultasi keuangan secara interaktif (Polireddi, 2024). Pengguna dapat menanyakan laporan pengeluaran, merencanakan anggaran, bahkan menerima saran finansial berbasis perilaku mereka sendiri.

Tabel 14. 2 Penerapan AI dalam Sektor Keuangan

Bidang Aplikasi	Teknologi AI yang Digunakan	Contoh Implementasi	Manfaat Utama
Deteksi Penipuan	Random Forest, DNN, SVM	PayPal, Mastercard	Keamanan transaksi real-time
Analisis Kredit	Explainable AI, Gradient Boosting	ZestFinance, JPMorgan Chase	Penilaian risiko lebih akurat
Perdagangan Otomatis	Reinforcement Learning, LSTM	BlackRock Aladdin, Goldman Sachs Marquee	Optimasi strategi investasi
Analisis Sentimen	NLP (BERT, LSTM)	BloombergGPT, FinBERT	Prediksi tren pasar dan

Bidang Aplikasi	Teknologi AI yang Digunakan	Contoh Implementasi	Manfaat Utama
			reaksi investor
Layanan Pelanggan	Chatbot NLP, Speech Recognition	Erica, Cleo, Kasisto	Pelayanan nasabah 24/7 dan personalisasi

Meski memberikan banyak manfaat, penggunaan AI di sektor keuangan tetap menghadapi tantangan. Salah satunya adalah transparansi algoritma, karena model AI sering kali beroperasi sebagai "kotak hitam" yang sulit dijelaskan. Selain itu, bias data dapat menyebabkan keputusan yang tidak adil, seperti diskriminasi terhadap kelompok tertentu dalam penilaian kredit. Oleh karena itu, banyak lembaga keuangan kini menekankan konsep "Responsible AI", yakni penggunaan AI yang etis, transparan, dan dapat diaudit (Meduri *et al.*, 2024).

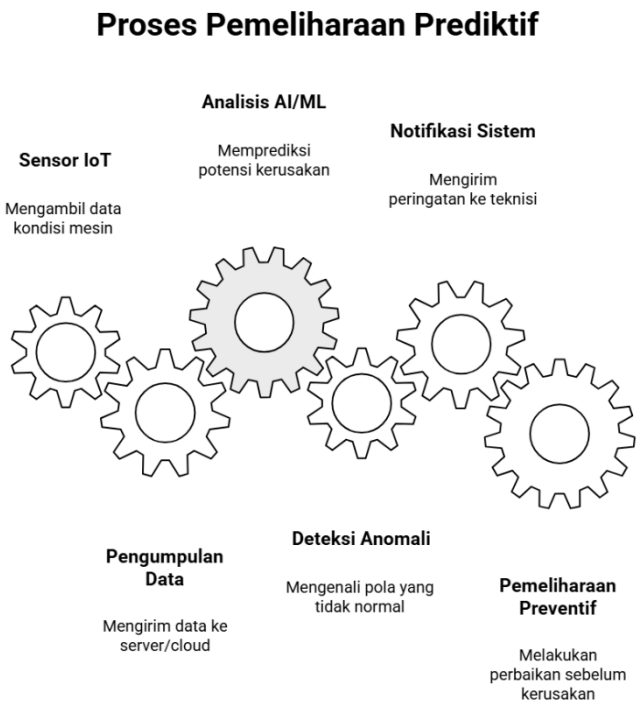
Secara keseluruhan, AI telah membawa industri keuangan ke arah yang lebih adaptif, efisien, dan cerdas. Teknologi ini tidak hanya memperkuat sistem keamanan dan akurasi analisis, tetapi juga membantu menciptakan pengalaman keuangan yang lebih personal bagi setiap pengguna. Di masa depan, kolaborasi antara AI, *blockchain*, dan *quantum computing* diyakini akan membentuk fondasi baru bagi sistem keuangan global yang lebih aman, transparan, dan inklusif.

14.4 AI dalam Bidang Industri dan Manufaktur

Kecerdasan buatan telah menjadi motor penggerak utama dalam transformasi industri modern. Di era Industri 4.0, AI tidak hanya berfungsi sebagai alat bantu, tetapi sebagai

sistem cerdas yang mampu belajar, menyesuaikan diri, dan membuat keputusan secara mandiri berdasarkan data produksi yang terus mengalir. Integrasi antara AI, *Internet of Things* (IoT), big data, dan komputasi awan membentuk fondasi bagi konsep *smart manufacturing*—sebuah sistem produksi yang efisien, adaptif, dan berorientasi pada prediksi (Bunian, Al-Ebrahim and Nour, 2024).

Salah satu penerapan paling menonjol dari AI di bidang manufaktur adalah *predictive maintenance*, yaitu pemeliharaan mesin secara prediktif untuk mencegah kerusakan sebelum terjadi (Baskoro *et al.*, 2025), seperti ditunjukkan pada Gambar 14.3. Sistem ini bekerja dengan menganalisis data sensor seperti getaran, suhu, atau tekanan untuk mendeteksi gejala awal gangguan. Perusahaan seperti Siemens dan General Electric telah memanfaatkan teknologi ini untuk mengurangi waktu henti produksi hingga setengahnya serta menekan biaya operasional secara signifikan.



Gambar 14. 3 Ilustrasi Proses *Predictive Maintenance*

Selain itu, AI juga memainkan peran penting dalam pengendalian kualitas (*quality control*). Teknologi *computer vision* yang berbasis *deep learning* memungkinkan mesin mengenali cacat produk dengan ketepatan yang jauh melampaui kemampuan manusia (Kim, Shin and Hwang, 2024; Baskoro *et al.*, 2025). Pabrik Tesla, misalnya, menggunakan sistem visi mesin yang memeriksa ribuan komponen kendaraan secara otomatis untuk memastikan kualitas setiap unit tetap konsisten.

Di sisi lain, AI membantu mengoptimalkan rantai pasokan (*supply chain*) melalui analisis permintaan pasar dan data logistik secara real-time. Dengan algoritma seperti *reinforcement learning* dan *genetic algorithm*, sistem dapat

memprediksi kebutuhan bahan baku, menentukan rute pengiriman paling efisien, serta menyesuaikan kapasitas produksi sesuai kondisi pasar (Palakurti, 2025). Perusahaan seperti Amazon menggunakan pendekatan ini untuk mempercepat distribusi barang dan mengurangi biaya pengiriman melalui sistem robotik cerdas di gudangnya.

AI juga membuka jalan bagi kolaborasi antara manusia dan robot (*human–robot collaboration*). Robot industri modern, atau *cobots*, kini mampu bekerja berdampingan dengan manusia secara aman berkat sistem sensor adaptif dan pembelajaran berbasis demonstrasi (Bi *et al.*, 2021; Yaakub *et al.*, 2025). Cobots ini dapat mempelajari tugas baru tanpa pemrograman kompleks, menjadikannya solusi fleksibel bagi industri yang dinamis.

Tabel 14. 3 Penerapan AI dalam Sektor Industri dan Manufaktur

Bidang Penerapan	Teknologi AI	Contoh Implementasi	Manfaat Utama
Predictive Maintenance	Machine Learning, IoT Analytics	Siemens MindSphere, GE Predix	Mengurangi downtime dan biaya perawatan
Quality Control	Computer Vision, CNN	Tesla AI Inspection System	Deteksi cacat produk otomatis
Supply Chain Optimization	Reinforcement Learning, Genetic Algorithm	Amazon Robotics, DHL SmartChain	Efisiensi distribusi dan logistik
Robotika Industri	Reinforcement Learning, Motion Planning	KUKA, Universal Robots	Kolaborasi aman manusia–robot

Bidang Penerapan	Teknologi AI	Contoh Implementasi	Manfaat Utama
Smart Factory Management	Deep Learning, Cloud Computing	Foxconn AI Factory	Produksi adaptif dan efisien

Meskipun manfaatnya besar, penerapan AI di sektor manufaktur tetap menghadapi tantangan, seperti integrasi dengan sistem lama, perlindungan data industri, dan kebutuhan sumber daya manusia yang terampil. Namun, arah perkembangan teknologi menunjukkan bahwa hambatan ini semakin mudah diatasi seiring hadirnya platform terbuka dan layanan *AI-as-a-Service* (Griesch, Rittelmeyer and Sandkuhl, 2024).

Secara keseluruhan, penerapan AI dalam industri dan manufaktur telah membawa perubahan besar: dari proses produksi yang statis menuju sistem yang cerdas, efisien, dan berkelanjutan. Dengan kemampuan menganalisis, memprediksi, dan beradaptasi, AI menjadikan pabrik modern tidak sekadar tempat produksi, tetapi pusat inovasi digital yang terus berkembang.

14.5 AI dalam Bidang Pendidikan (*Education*)

Kecerdasan buatan (AI) semakin memegang peranan penting dalam dunia pendidikan. Melalui pemanfaatan teknologi seperti *machine learning*, *natural language processing* (NLP), dan *data analytics*, AI membantu menciptakan proses belajar yang lebih personal, adaptif, dan efisien (Prabhu Chakkaravarthy *et al.*, 2025). Sistem pembelajaran kini tidak lagi bersifat seragam, tetapi mampu menyesuaikan materi, tempo, dan tingkat kesulitan sesuai kemampuan masing-masing siswa (Betaubun, Laila Rokhmah and Budiasto, 2023).

Salah satu bentuk penerapan paling menonjol adalah pembelajaran adaptif. Sistem seperti *Knewton* atau *Smart Sparrow* menggunakan algoritma pembelajaran mesin untuk menganalisis interaksi siswa dan menyesuaikan materi berdasarkan kemajuan individu. Dengan pendekatan ini, siswa yang mengalami kesulitan dapat memperoleh bantuan tambahan, sementara mereka yang lebih cepat memahami materi dapat melanjutkan ke tingkat berikutnya.

Selain itu, analisis pembelajaran (*learning analytics*) menjadi alat penting dalam memahami pola belajar siswa. Melalui data yang diambil dari platform daring seperti *Coursera* atau *edX*, AI dapat memprediksi risiko kegagalan siswa, memberikan peringatan dini, dan membantu pendidik melakukan intervensi yang lebih tepat sasaran (Quadri and Shukor, 2021).

AI juga memperluas peran NLP dalam pembelajaran bahasa (Betaubun, Budiasto and Rokhmah, 2025). Aplikasi seperti *Duolingo* atau *Grammarly* memanfaatkan NLP untuk mengenali kesalahan tata bahasa, memberi saran perbaikan, serta menyesuaikan latihan agar sesuai dengan kemampuan pengguna. Bahkan, sistem penilaian otomatis kini mampu memeriksa esai, tugas pemrograman, hingga ujian berbasis gambar dengan akurasi tinggi.

Dalam konteks pembelajaran jarak jauh, chatbot edukatif dan asisten virtual menjadi pendamping belajar yang efektif. Contohnya, *Jill Watson*, asisten virtual di Georgia Tech, mampu menjawab pertanyaan mahasiswa di forum daring tanpa mereka sadari bahwa ia bukan manusia. Di Indonesia, platform seperti *Ruangguru* juga telah mengintegrasikan chatbot cerdas untuk membantu siswa belajar kapan saja.

Tabel 14. 4 Penerapan AI dalam Bidang Pendidikan

Bidang Aplikasi	Teknologi AI yang Digunakan	Contoh Implementasi	Manfaat Utama
Pembelajaran Adaptif	Machine Learning	Knewton, Smart Sparrow	Personalisasi pembelajaran
Analisis Performa	Predictive Analytics, Data Mining	Coursera, edX	Identifikasi risiko kegagalan belajar
NLP & Penilaian Bahasa	NLP, Reinforcement Learning	Duolingo, Grammarly	Pembelajaran bahasa adaptif
Penilaian Otomatis	Computer Vision, Pattern Recognition	Gradescope, Write & Improve	Efisiensi evaluasi akademik
Chatbot Edukatif	NLP, Conversational AI	Jill Watson, Ruangguru Bot	Dukungan pembelajaran 24/7

Meski banyak manfaat yang ditawarkan, penerapan AI di bidang pendidikan tetap menghadapi tantangan seperti privasi data siswa, kesiapan guru terhadap teknologi, serta potensi ketimpangan akses digital (Salloum, Salloum and Alfaisal, 2024). Oleh karena itu, integrasi AI perlu diimbangi dengan kebijakan pendidikan yang inklusif dan beretika.

Secara keseluruhan, AI membuka jalan menuju sistem pendidikan yang lebih adaptif, humanis, dan berorientasi pada peserta didik. Teknologi ini bukan sekadar alat bantu, tetapi mitra yang mampu mendukung terciptanya pengalaman belajar yang mendalam dan berkelanjutan.

14.6 Penutup: Sinergi Manusia dan Mesin Menuju Era Inovasi Berkelanjutan

Kecerdasan buatan kini menjadi fondasi utama dalam transformasi berbagai sektor kehidupan. Dari kesehatan hingga pendidikan, dari industri hingga keuangan, AI telah menunjukkan kemampuannya untuk mempercepat proses, meningkatkan efisiensi, serta memberikan wawasan yang sebelumnya sulit dicapai manusia. Melalui kemampuan belajar dari data, mengenali pola, dan membuat keputusan cerdas, AI tidak hanya mengubah sistem kerja, tetapi juga cara berpikir dan berinovasi dalam masyarakat modern (Dwivedi *et al.*, 2021; Budiasto, Tallulembang, *et al.*, 2025).

Dalam bidang kesehatan, AI membantu tenaga medis mendiagnosis penyakit lebih akurat dan cepat. Di sektor keuangan, teknologi ini mendukung deteksi penipuan dan analisis risiko yang lebih efisien. Industri dan manufaktur kini memanfaatkan AI untuk mengelola produksi, memprediksi kerusakan mesin, dan meningkatkan kualitas hasil produksi. Sementara di dunia pendidikan, AI menghadirkan pengalaman belajar yang lebih personal dan adaptif bagi setiap siswa. Keempat sektor ini memperlihatkan bahwa AI bukan sekadar alat bantu teknologi, tetapi telah menjadi motor penggerak perubahan yang membawa efisiensi, ketepatan, dan inovasi.

Meski begitu, kehadiran AI juga membawa tantangan baru. Permasalahan seperti keamanan data, bias algoritmik, dan dampak sosial terhadap lapangan pekerjaan perlu menjadi perhatian serius. Implementasi AI yang berhasil bukan hanya soal teknologi, tetapi juga tentang tanggung jawab etika, transparansi, dan keadilan dalam penggunaannya (Morante *et al.*, 2024).

Ke depan, AI diharapkan mampu memperkuat kolaborasi antara manusia dan mesin. Teknologi ini sebaiknya tidak menggantikan peran manusia, melainkan

memperluas kapasitasnya dalam berpikir, mencipta, dan mengambil keputusan. Masa depan AI akan sangat bergantung pada bagaimana manusia mengarahkannya — apakah untuk menciptakan dunia yang lebih efisien semata, atau untuk membangun kehidupan yang lebih cerdas, inklusif, dan berkeadilan bagi semua.

DAFTAR PUSTAKA

Bachhar, S. *et al.* (2025) 'Emerging horizons of AI in pharmaceutical research', *Advances in Pharmacology*, 103, pp. 325–348. Available at: https://doi.org/10.1016/bs.apha.2025.01.016.

Baskoro, S.E. *et al.* (2025) *Kecerdasan Buatan dalam Kehidupan Nyata: Konsep, Algoritma dan Penerapan*. Star Digital Publishing. Available at: https://books.google.co.id/books?id=tGWGEQAAQBAJ.

Betaubun, M., Budiasto, J. and Rokhmah, D.E.L. (2025) 'ChatGPT In EFL Learning Navigating the Impact of Technological Advancements on Educational Policy, Equity, and Legal Framework', *Nusantara Science and Technology Proceedings* [Preprint]. Available at: https://doi.org/https://doi.org/10.11594/nstp.2025.4816.

Betaubun, M., Laila Rokhmah, D.E. and Budiasto, J. (2023) 'Personalized Pathways to Proficiency: Exploring the Synergy of Adaptive Learning and Artificial Intelligence in English Language Learning.', *Technium*, 17.

Bi, Z.M. *et al.* (2021) 'Safety assurance mechanisms of collaborative robotic systems in manufacturing', *Robotics and Computer-Integrated Manufacturing*, 67. Available at: https://doi.org/10.1016/j.rcim.2020.102022.

Budiasto, J., Purnama, S.G., *et al.* (2025) *Data Science Technology*. PT. Star Digital Publishing, Yogyakarta-Indonesia. Available at: https://books.google.co.id/books?id=3fxlEQAAQBAJ.

Budiasto, J., Tallulembang, T.M., *et al.* (2025) *Machine Learning untuk Pemula: Konsep dan Implementasi*. Edited by T.Y. Zalni. Jakarta Utara: Penerbit Buku Indonesia.

Budiasto, J., Jayawardana, H., *et al.* (2025) *Pengantar Ilmu Komputer: Pengenalan Dasar Komputer dan Teknologi Informasi Modern*. Yogyakarta: PT. Star Digital Publishing, Yogyakarta-Indonesia. Available at: https://books.google.co.id/books?id=JWNfEQAAQBAJ.

Bunian, S., Al-Ebrahim, M.A. and Nour, A.A. (2024) 'Role and Applications of Artificial Intelligence and Machine Learning in Manufacturing Engineering: A Review', *Engineered Science*, 29. Available at: https://doi.org/10.30919/es1088.

Chowdhury, I.J. *et al.* (2025) 'Revolutionizing Drug Discovery: A Systematic Review of AI and Machine Learning Application', in *Proceedings - 3rd International Conference on Self Sustainable Artificial Intelligence Systems, ICSSAS 2025*, pp. 1814–1821. Available at: https://doi.org/10.1109/ICSSAS66150.2025.11081368.

Dwivedi, Y.K. *et al.* (2021) 'Artificial Intelligence (AI): Multidisciplinary perspectives on emerging challenges, opportunities, and agenda for research, practice and policy', *International Journal of Information Management*, 57. Available at: https://doi.org/10.1016/j.ijinfomgt.2019.08.002.

Griesch, L., Rittelmeyer, J. and Sandkuhl, K. (2024) 'Towards AI as a Service for Small and Medium-Sized Enterprises (SME)', in *Lecture Notes in Business Information Processing*, pp. 37–53. Available at: https://doi.org/10.1007/978-3-031-48583-1_3.

Grzybowski, A., Jin, K. and Wu, H. (2024) 'Challenges of artificial intelligence in medicine and dermatology', *Clinics in Dermatology*, 42(3), pp. 210–215. Available at: https://doi.org/10.1016/j.clindermatol.2023.12.013.

Kamikubo, R. *et al.* (2022) 'Data Representativeness in Accessibility Datasets: A Meta-Analysis', in *ASSETS 2022 - Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility*. Available at: https://doi.org/10.1145/3517428.3544826.

Kanakaprabha, S. *et al.* (2024) 'Wearable Devices and Health Monitoring: Big Data and AI for Remote Patient Care', in *Intelligent Data Analytics for Bioinformatics and Biomedical Systems*, pp. 291–311. Available at: https://doi.org/10.1002/9781394270910.ch12.

Kim, B., Shin, M. and Hwang, S. (2024) 'Design and Development of a Precision Defect Detection System Based on a Line Scan Camera Using Deep Learning', *Applied Sciences (Switzerland)*, 14(24). Available at: https://doi.org/10.3390/app142412054.

Kumar, A., Sareen, P. and Arora, A. (2023) 'Healthcare Engineering Using AI and Distributed Technologies', in *Smart Distributed Embedded Systems for Healthcare Applications*, pp. 1–14. Available at: https://doi.org/10.1201/9781003254119-1.

Manolescu, D., Buckley, N. and Secco, E.L. (2024) 'Convolutional Neural Network Applied to X-ray Medical Imagery for Pneumonia Identification', in *Lecture Notes in Networks and Systems*, pp. 183–197. Available at: https://doi.org/10.1007/978-981-97-2053-8_14.

Meduri, K. *et al.* (2024) 'Accountability and Transparency Ensuring Responsible AI Development', in *Ethical Dimensions of AI Development*, pp. 83–102. Available at: https://doi.org/10.4018/979-8-3693-4147-6.ch004.

Morante, G. *et al.* (2024) 'Proposal of an Ethical and Social Responsibility Framework for Sustainable Value Generation in AI', in *2024 IEEE Technology and Engineering Management Society, TEMSCON LATAM 2024*. Available at: https://doi.org/10.1109/TEMSCONLATAM61834.2024.10717855.

Nampira, A.A. *et al.* (2025) *Artificial Intelligence: A Guide for Thinking Humans*. PT. Green Pustaka Indonesia. Available at: https://books.google.co.id/books?id=_Y95EQAAQBAJ.

Palakurti, N.R. (2025) 'Leveraging AI-Driven Predictive Analytics for Demand Forecasting, Route Optimization, and Minimizing Food Waste in the Supply Chain', in *Cutting-Edge Solutions for Advancing Sustainable Development: Exploring Technological Horizons for Sustainability - Part 2*, pp. 157–172. Available at: https://doi.org/10.2174/9789815322408125010011.

Polireddi, N.S.A. (2024) 'Artificial intelligence in banking and financial services', in *Research Advances in Intelligent Computing: Volume 2*, pp. 139–152. Available at: https://doi.org/10.1201/9781003433941-9.

Prabhu Chakkaravarthy, A. *et al.* (2025) 'Personalized learning through AI on the power of tailored education', in *AI Applications and Pedagogical*

Innovation, pp. 455–479. Available at: https://doi.org/10.4018/979-8-3373-5092-9.ch016.

Quadri, A.T. and Shukor, N.A. (2021) 'The Benefits of Learning Analytics to Higher Education Institutions: A Scoping Review', *International Journal of Emerging Technologies in Learning*, 16(23), pp. 4–15. Available at: https://doi.org/10.3991/ijet.v16i23.27471.

Ramakrishnan, R. *et al.* (2024) 'Employing AI and ML in Risk Assessment for Lending for Assessing Credit Worthiness', in *2024 2nd International Conference on Disruptive Technologies, ICDT 2024*, pp. 561–566. Available at: https://doi.org/10.1109/ICDT61202.2024.10489313.

Rashid, A.B. and Kausik, M.A.K. (2024) 'AI revolutionizing industries worldwide: A comprehensive overview of its diverse applications', *Hybrid Advances*, 7. Available at: https://doi.org/10.1016/j.hybadv.2024.100277.

Rashid, M.M. *et al.* (2022) 'Artificial Intelligence Algorithms for Treatment of Diabetes', *Algorithms*, 15(9). Available at: https://doi.org/10.3390/a15090299.

Saadia, D. (2021) 'Integration of cloud computing, big data, artificial intelligence, and internet of things: review and open research issues', *International Journal of Web-Based Learning and Teaching Technologies*, 16(1), pp. 10–17. Available at: https://doi.org/10.4018/IJWLTT.2021010102.

Salloum, S.A., Salloum, A. and Alfaisal, R. (2024) 'Objectives and Obstacles of Artificial Intelligence in Education', in *Studies in Big Data*, pp. 605–614. Available at: https://doi.org/10.1007/978-3-031-52280-2_38.

Sani, A. *et al.* (2025) *PENGANTAR TEKNOLOGI INFORMASI: Dampak dan Peluang Perkembangan Teknologi*

Informasi dalam Dunia Kerja dan Bisnis. Edited by D.F. Romiza. Yogyakarta: PT. Star Digital Publishing, Yogyakarta-Indonesia. Available at: https://books.google.co.id/books?id=EOVhEQAAQBAJ.

Venkateswaran, P.S. and Sriramkumar, M. (2025) 'Predictive analytics: Utilizing machine learning and big data for forecasting future trends in business and consumer behavior', in *Strategic Brand Management in the Age of AI and Disruption*, pp. 463–491. Available at: https://doi.org/10.4018/979-8-3693-9461-8.ch019.

Vetrivel, S.C. *et al.* (2024) 'Impact of AI Adoption in Current Trends of the Financial Industry', in *Artificial Intelligence for Risk Mitigation in the Financial Industry*, pp. 103–131. Available at: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85199836736&partnerID=40&md5=2bd11dc7f0de87f9295e3a29e3c1097c.

Yaakub, S. *et al.* (2025) *AUTOMASI MASA DEPAN: AI, MACHINE LEARNING, DAN PERUBAHAN DUNIA KERJA*. Edited by A. Yanto. Get Press Indonesia.

Zou, J. *et al.* (2024) 'A novel Deep Reinforcement Learning based automated stock trading system using cascaded LSTM networks', *Expert Systems with Applications*, 242. Available at: https://doi.org/10.1016/j.eswa.2023.122801.

BIODATA PENULIS

Ir. Jarot Budiasto, S.T., M.T.

Dosen Program Studi Sistem Informasi

Fakultas Teknik Universitas Musamus

Penulis lahir di Magelang pada 4 Maret 1981. Menyelesaikan pendidikan Sarjana (S1) di Program Studi Teknik Informatika, Universitas Atma Jaya Yogyakarta, pada tahun 2006, dan melanjutkan pendidikan Magister (S2) di bidang Teknik Informatika di universitas yang sama, lulus pada tahun 2008. Pada tahun 2022, penulis memperoleh gelar Insinyur dari Universitas Muslim Indonesia.

Sejak tahun 2008, penulis aktif berkarier sebagai dosen di Program Studi Sistem Informasi, Universitas Musamus, dengan keahlian utama di bidang *Data Analytics* dan *Machine Learning*. Selain berkecimpung di dunia akademik, penulis juga aktif menulis dan telah menerbitkan beberapa buku, di antaranya *Pengantar Ilmu Komputer*, *Pengantar Teknologi Informasi*, *Sistem Pendukung Keputusan*, *Database*, *Data Analytics*, *Artificial intelligence*, *Machine Learning* dan *Data Science*.

Melalui karya dan dedikasinya, penulis berkomitmen untuk terus berkontribusi dan berupaya menciptakan dampak positif yang signifikan dalam pengembangan ilmu pengetahuan dan teknologi di Indonesia.

BIODATA PENULIS

Deddy Rudhistiar, S.Kom., M.Cs.

Dosen Program Studi S1 Teknik Informatika

Fakultas Teknologi Industri Institut Teknologi Nasional Malang

Saat ini, penulis menjabat sebagai dosen tetap di Program Studi S1 Teknik Informatika, Fakultas Teknologi Industri, Institut Teknologi Nasional Malang. Penulis menyelesaikan pendidikan Sarjana di Jurusan Teknik Informatika dan melanjutkan pendidikan Magister di Jurusan Ilmu Komputer. Keahlian penulis meliputi Jaringan Komputer, Keamanan Jaringan, Kriptografi, serta Teknologi terkait dengan Cloud Computing.

BIODATA PENULIS

Risanto Darmawan, MM, MKom.

Dosen Program Studi Teknik Infomatika

Fakultas Teknik, Universitas Krisnadwipayana

Penulis lahir di Jakarta di tahun 1967. Penulis menamatkan pendidikan Diploma D2 Informatika di Fakultas Politeknik Pertanian IPB Bogor di 1991. Program Sarjana (S1) di Universitas Budi Luhur, Jakarta, Fakultas Teknologi Informasi, Sistem Informasi di tahun 2000 dan Universitas Terbuka, Tangerang Selatan, Fakultas Ekonomi, Prodi Manajemen di 2003. Penulis juga menyelesaikan program Pasca Sarjana (S2) di Universitas Pelita Harapan, Karawaci, Fakultas Business & Management, spesialisasi Marketing di tahun 2003 dan di Sekolah Pascasarjana, IPB Bogor, Program Studi Ilmu Komputer, lulus di tahun 2006.

Penulis telah bekerja selama beberapa tahun pada Perusahaan swasta nasional dan multinasional dan telah melaksanakan berbagai proyek Sistem Informasi mulai dari Project Retail, Research, Finance, Marketing serta Human Capital.

Penulis memegang sertifikasi BNSP untuk Perekayasaan Perangkat IoT, Data Scientist dan Cyber Security tahun 2024.

BIODATA PENULIS

Dwi Sasmita Aji Pambudi, S.T., M.T.

Dosen Program Studi Teknik Kelistrikan Kapal

Politeknik Perkapalan Negeri Surabaya (PPNS)

Penulis lahir di Lumajang tanggal 21 April 1992. Penulis adalah dosen tetap pada Program Studi Teknik Kelistrikan Kapal, Jurusan Teknik Kelistrikan Kapal, Politeknik Perkapalan Negeri Surabaya (PPNS). Penulis menyelesaikan pendidikan S1 pada Jurusan Teknik Elektro di Universitas Telkom, Bandung pada tahun 2013 dan melanjutkan jenjang S2 pada Jurusan Teknik Elektro, Bidang Keahlian Elektronika di Institut Teknologi Sepuluh Nopember (ITS), Surabaya pada tahun 2018. Penulis mulai menekuni bidang menulis sejak awal menjadi dosen di tahun 2019. Ketertarikan bidang penelitiannya adalah elektronika industri, elektronika kontrol cerdas, elektronika daya, elektronika navigasi, *Internet of Things* (IoT), Energi Baru Terbarukan (EBT), sistem kelistrikan dan elektronika kapal. Dalam mewujudkan karir sebagai dosen profesional, penulis terus berupaya aktif dalam melakukan penelitian, pengabdian masyarakat, dan menulis baik dalam bentuk jurnal nasional maupun internasional.

BIODATA PENULIS

Ronald Belferik, S.Kom., M.Kom.

Dosen Program Studi Informatika

Fakultas Teknologi Informasi Universitas Pelita Harapan

Penulis lahir di Kabupaten Deli Serdang, Provinsi Sumatera Utara tanggal 05 April 1987. Penulis adalah dosen tetap pada Program Studi Informatika Fakultas Teknologi Informasi, Universitas Pelita Harapan. Menyelesaikan pendidikan S1 pada Jurusan Sistem Informasi di Universitas Mikroskil dan melanjutkan S2 pada Jurusan Teknik Informatika di Universitas Sumatera Utara. Penulis menekuni bidang Menulis.

BIODATA PENULIS

Chairi Nur Insani, S.Kom., M.T.

Dosen Program Studi Informatika

Fakultas Teknik Universitas Sulawesi Barat

Penulis lahir di Wonomulyo pada tanggal 27 Juli 1994. Penulis adalah dosen tetap pada Program Studi Informatika Fakultas Teknik, Universitas Sulawesi Barat. Menyelesaikan pendidikan S1 pada Jurusan Teknik Informatika di STMIK Dipanegara Makassar yang telah berubah menjadi Universitas Dipa Makassar dan melanjutkan S2 pada Jurusan Teknik Informatika di Universitas Hasanuddin. Penulis menekuni bidang Menulis. Selama masa studi tersebut, penulis mendalami bidang-bidang khusus dalam dunia informatika, mempelajari pemahaman mendalam tentang aspek-aspek teknis dan konseptual yang mendukung perkembangan ilmu pengetahuan dan teknologi.

Dalam proses penyusunan bab ini, penulis memanfaatkan teknologi kecerdasan buatan (Artificial Intelligence/AI) secara terbatas dan bertanggung jawab, khususnya sebagai alat bantu linguistik dalam penyusunan, perbaikan, dan parafrase kalimat, serta pembuatan bagan ilustrasi. Pemanfaatan AI dilakukan semata-mata untuk

meningkatkan kejelasan bahasa, konsistensi istilah, dan kualitas penyajian akademik, tanpa mengubah substansi ilmiah, gagasan utama, maupun kerangka konseptual yang sepenuhnya merupakan hasil pemikiran dan analisis penulis.

BIODATA PENULIS

Dedy Abdianto Nggego, S.SI., M.Kom

Dosen Program Studi Teknik Informatika

Fakultas Teknik Universitas Musamus Merauke

Penulis lahir di Poso tanggal 11 September 1992. Penulis adalah dosen tetap pada Program Studi Teknik Informatika Fakultas Teknik, Universitas Musamus. Menyelesaikan pendidikan S1 pada Jurusan Sistem Informasi dan melanjutkan S2 pada Jurusan Teknik Informatika. Penulis menekuni bidang kecerdasan buatan dan soft-computing.

BIODATA PENULIS

Ir. Selfina Pare, S.Kom., M.T.

Dosen Program Studi Sistem Informasi

Fakultas Teknik Universitas Musamus

Penulis lahir di Merauke pada tanggal 16 September 1981. Menempuh pendidikan Sarjana (S1) pada Program Studi Sistem Informasi, Sekolah Tinggi Manajemen Informatika dan Komputer (Makassar) dan menyelesaikannya pada tahun 2005. Kemudian melanjutkan ke jenjang Strata-2 (S2) pada tahun 2009 di Universitas Hasanuddin (Makassar) pada Program Studi Teknik Elektro, dan berhasil memperoleh gelar Magister pada tahun 2011. Penulis kemudian memperkuat bidang keteknikan melalui profesi insinyur pada tahun 2024.

Saat ini, penulis menjadi Dosen Tetap di Universitas Musamus pada Program Studi Sistem Informasi, dengan bidang keahlian Sistem Pendukung Keputusan. Penulis memiliki pengalaman mengajar pada beberapa mata kuliah, seperti Manajemen Sains, Proyek Pengembangan Sistem Informasi, Sistem Basis Data, dan Analisis Proses Bisnis. Penulis juga aktif dalam penulisan buku ilmiah, di antaranya buku *Machine Learning untuk Pemula: Konsep dan*

Implementasi dan *Augmented Reality Dasar pada Pemrograman Mobile dengan Unity 3D*.

BIODATA PENULIS

Ir. Marsujitullah, S.Kom.,M.T.

Dosen Program Studi Teknik Informatika

Fakultas Teknik Universitas Musamus

Penulis lahir di Merauke tanggal 27 Mei 1989. Penulis adalah dosen tetap pada Program Studi Teknik Informatika Fakultas Teknik, Universitas Musamus. Menyelesaikan pendidikan S1 pada Jurusan Teknik Informatika pada Universitas Musamus dan melanjutkan S2 pada Jurusan Teknik Informatika pada Universitas Hasanuddin Makassar. Selain itu penulis menyelesaikan program profesi Insinyur pada universitas yang sama di Makassar. Saat ini penulis sedang menempuh pendidikan doktoral pada Universitas Negeri Yogyakarta pada Prodi Ilmu Teknik, Teknik Informatika. Penulis menekuni bidang Sistem Informasi Geografis, Pengolahan Citra dan juga Internet of Things. Dalam kesehariannya, penulis juga aktif pada berbagai organisasi yang menunjang fokus bidangnya, seperti pada Federasi Aero Sport Indonesia pada tigkat provinsi, menjabat sebagai ketua harian, Asosiasi Pilot Drone Indonesia Region Merauke, menjabat sebagai Penasihat. Serta berbagai organisasi masyarakat, keagamaan maupun olah raga.

BIODATA PENULIS

Hasanudin Jayawardana, S.T., MMSI.

Dosen Jurusan Sistem Informasi

Fakultas Teknik Universitas Musamus

Penulis menempuh pendidikan Sarjana (S1) pada Program Studi Teknik Elektro, kosentrasi Elektronika, Institut Teknologi Nasional (Malang) dan menyelesaiannya pada tahun 2005, seelah lulus pernah menjadi tenaga pengajar di salah satu Sekolah Menengah Kejuruan Negri (Teknik) di Merauke. Kemudian melanjutkan ke jenjang Strata-2 (S2) pada tahun 2008 di Universitas Gunadarma (Jakarta) pada bidang Manajemen Sistem Informasi, dan berhasil memperoleh gelar Magister pada tahun 2011.

Saat ini, penulis menjadi Dosen Tetap di Universitas Musamus Merauke pada jurusan Sistem Informasi, dengan bidang keahlian Manajemen dan Strategi pengembangan Sistem Informasi. Penulis memiliki pengalaman mengajar pada beberapa mata kuliah, seperti Analisis dan Desain Sistem Informasi, Sistem Informasi Manajemen, Pengantar Teknologi Informasi, Sistem Basis Data, Sistem Operasi, Algoritma dan Pemrograman serta Sistem Pendukung Keputusan.

BIODATA PENULIS

Tri Kustanti Rahayu, S.Kom., M.Kom.

Dosen Program Studi Sistem Informasi

Fakultas Teknik Universitas Musamus Merauke

Penulis lahir di Mojokerto pada tahun 1982, adalah seorang dosen dan penulis yang memiliki minat besar pada dunia teknologi dan literasi. Ia menyelesaikan pendidikan Sarjana (S1) di Universitas Dr. Soetomo Surabaya pada tahun 2005, kemudian melanjutkan studi Magister (S2) di Universitas Diponegoro, Semarang, dan meraih gelar pada tahun 2018. Menulis dan traveling menjadi bagian penting dalam hidupnya, memberikan inspirasi dalam setiap karya dan penelitiannya. Bidang keilmuan yang menjadi favoritnya adalah Natural Language Processing (NLP) dan Computer Vision, yang mencerminkan dedikasinya untuk mengembangkan inovasi berbasis teknologi demi kemajuan ilmu pengetahuan.

BIODATA PENULIS

Ir. Agustan Latif, S.Kom., M.Cs

Dosen Program Studi Sistem Informasi

Fakultas Teknik, Universitas Musamus

Penulis lahir di Bade tanggal 08 Agustus 1983. Menyelesaikan pendidikan Sarjana (S1) di Program Studi Teknik Informatika Fakultas Teknik dan Ilmu Komputer Universitas Komputer Indonesia Bandung lulus tahun 2006, melanjutkan pendidikan Magister (S2) di Program Studi Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Gadjah Mada Yogyakarta lulus tahun 2013. Memperoleh Gelar Insinyur dari Universitas Hasanuddin Makassar lulus tahun 2024.

Penulis memulai karir sebagai dosen pada tahun 2008 di Program Studi Sistem Informasi Universitas Musamus, dengan keahlian utama di bidang Teknologi Informasi dan Visi Komputer. Selain berkecimpung di dunia akademik, penulis juga aktif dalam riset dan publikasi ilmiah pada bidang tersebut. Penulis juga aktif menulis dan telah menerbitkan buku yang berjudul Sistem Pendukung Keputusan; Mechine Learning: Konsep, Implementasi, dan

Aplikasi; Automasi Masa Depan: AI, Mechine Learning, dan Perubahan Dunia Kerja; dan Python untuk Data Science: Panduan Praktis Menguasai Analisis Data.

Melalui karya dan dedikasinya, penulis berkomitmen untuk terus berkontribusi dan berupaya menciptakan dampak positif yang signifikan dalam pengembangan ilmu pengetahuan dan teknologi di Indonesia khususnya di wilayah Perbatasan Papua Selatan.

BIODATA PENULIS

Muhammad Hasbi, M.Kom

Dosen Program Studi Sistem Informasi

Fakultas Teknik Universitas Musamus

Penulis lahir pada 16 Februari 1992 di Kabupaten Maros, Sulawesi Selatan, Indonesia. Penggiat teknologi ini telah menamatkan pendidikan sarjana di Universitas Negeri Makassar (UNM) pada jurusan Pendidikan Teknik Informatika dan Komputer. Setelah menamatkan pendidikannya tersebut, memutuskan untuk melanjutkan pendidikan tingkat magister (S-2) pada Jurusan Ilmu Komputer, Universitas Brawijaya, Malang, Indonesia. Selama kuliah, menekuni dan melakukan pengembangan ilmu pengetahuan dengan fokus riset dibidang Multimedia, Game & Mobile Development, termasuk didalamnya teknologi Augmented reality dan Virtual Reality (AR/VR). Memulai karir sebagai pengajar/dosen pada tahun 2021 pada Jurusan Teknik Informatika, Politeknik Harapan Bersama (PHB) Tegal, Jawa Tengah. Ditahun yang sama juga, menjadi Staf Pengajar/Dosen pada Jurusan Informatika, Universitas Muhammadiyah Makassar

Pada tahun 2022, ditetapkan menjadi Aparatur Sipil Negara (ASN) sebagai Dosen pada Jurusan Sistem Informasi, Universitas Musamus Merauke. Di Universitas inilah melanjutkan tugas untuk mengembangkan dan menyebarluaskan ilmu pengetahuan melalui Tri Dharma Pendidikan Tinggi hingga sekarang.