

PERANCANGAN REINFORCEMENT LEARNING

Teori Pendukung, Lifecycle, dan Implementasi Python

Agent

Action

A

Muhammad Ardiansyah Azman
Tri Susanto
Almindio Vabiano

**Perancangan *Reinforcement Learning*:
Teori Pendukung, *Lifecycle*, dan Implementasi Python**

UU No. 28 tahun 2014 tentang Hak Cipta

Fungsi dan sifat hak cipta pada pasal 4

Hak Cipta sebagaimana dimaksud dalam Pasal 3 huruf a merupakan hak eksklusif yang terdiri atas hak moral dan hak ekonomi.

Pembatasan Perlindungan Pasal 26

Ketentuan sebagaimana dimaksud dalam Pasal 23, Pasal 24, dan Pasal 25 tidak berlaku terhadap:

- i. Penggunaan kutipan singkat Ciptaan dan/atau produk Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual.
- ii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan.
- iii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk keperluan pengajaran, kecuali pertunjukan dan Fonogram yang telah dilakukan Pengumuman sebagai bahan ajar; dan
- iv. Penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkinkan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

- Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan /atau pidana denda paling banyak Rp 100.000 (seratus juta rupiah).
- Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp 500.000.000 (lima ratus juta rupiah).

**Perancangan *Reinforcement Learning*:
Teori Pendukung, *Lifecycle*, dan Implementasi Python**

Penyusun:

Muhammad Ardiansyah Azman

Tri Susanto

Almindio Vabiano



2026

Perancangan *Reinforcement Learning*: Teori Pendukung, Lifecycle, dan Implementasi Python

Jumlah halaman : x, 234 halaman

Ukuran halaman : 18,2 x 25,7 cm

ISBN e-book : 978-634-7971-02-9 (PDF)

Penulis :

Muhammad Ardiansyah Azman

Tri Susanto

Almindo Vabiano

Desain Cover :

Nurhakim As'ad Wicaksono

Hak Cipta dilindungi Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

Siapapun dilarang keras menerjemahkan, mencetak, atau memperbanyak sebagian atau seluruh isi buku ini tanpa izin tertulis dari penulis dan penerbit.

Cetakan pertama :

Oktober 2026

Diterbitkan oleh :

Universitas Bakrie Press

Penerbit Anggota IKAPI No. 638/Anggota Luar Biasa/DKI/2024



Komplek Rasuna Epicentrum
Jl. HR. Rasuna Said, Setiabudi,
Kuningan Jakarta 12920
Email: ubakriepress@bakrie.ac.id
Website: ubakriepress.bakrieuniversity.ac.id

PRAKATA

Machine Learning (ML) kini telah menjadi bagian yang tidak terpisahkan dari kehidupan kita sehari-hari. Tanpa disadari, kita telah menikmati berbagai kemudahan yang ditawarkannya mulai dari hal sederhana seperti memerintahkan smartphone untuk memutar lagu favorit, hingga pemanfaatan yang lebih kompleks seperti analisis proses produksi di industri maupun sistem rekomendasi kuliner yang dipersonalisasi.

Melihat perkembangannya yang begitu pesat, penyusun mencoba mengemas buku ini secara praktis, sederhana, dan langsung tepat pada sasaran, yang dilengkapi dengan contoh implementasi nyata menggunakan bahasa pemrograman Python. Buku ini dirancang agar sangat layak dan bersahabat bagi para pemula yang baru saja menapakkan kaki di dunia Machine Learning. Teori-teori dasar disajikan secara ringkas, di mana pembaca akan dibimbing untuk memahami konsep ML secara lebih mendalam melalui ilustrasi dan analogi yang mudah dicerna.

Secara garis besar, buku ini disusun ke dalam empat kategori utama, yaitu:

- Pengenalan Reinforcement Learning
- Teori Pendukung Machine Learning
- Machine Learning Lifecycle
- Algoritma-Algoritma Machine Learning

Bagi pembaca yang baru pertama kali mempelajari bidang ini, sangat disarankan untuk membaca buku ini secara berurutan. Namun, bagi pembaca yang sudah memiliki pengetahuan dasar tentang ML, Pembaca dapat langsung memulainya dari bagian Machine Learning Lifecycle. Perlu diketahui bahwa pada bagian algoritma, penyusun secara khusus memilih algoritma-algoritma dasar yang diarahkan pada Reinforcement Learning, sedangkan untuk materi Supervised Learning dan Unsupervised Learning akan dibahas secara terpisah.

Penyusun menyampaikan rasa terima kasih yang sebesar-besarnya kepada seluruh pihak yang telah membantu dalam proses penyusunan dan penerbitan buku ini, terutama kepada Program Studi Teknik Industri Universitas Bakrie dan UBakrie Press.

Penyusun menyadari bahwa buku ini masih jauh dari kesempurnaan. Oleh karena itu, kritik dan saran yang membangun dari para pembaca akan sangat kami harapkan demi perbaikan di masa mendatang. Akhir kata, semoga buku ini dapat memberikan manfaat yang luas bagi pembaca, serta mampu menginspirasi generasi muda bangsa agar tumbuh menjadi generasi yang tanggap teknologi dan tangguh dalam menghadapi tantangan masa depan.

Jakarta, Juli 2026

DAFTAR ISI

PRAKATA	v
DAFTAR ISI	vi
DAFTAR GAMBAR.....	viii
DAFTAR TABEL	x
BAB I PEMAHAMAN <i>MACHINE LEARNING</i>.....	1
1.1 Definisi Machine Learning	4
1.2 Gambaran Ringkas ML.....	7
1.3 Machine Learning bagian AI.....	9
1.4 Contoh Aplikasi ML Sederhana	11
1.5 Klasifikasi Machine Learning	18
1.6 Algoritma Machine Learning	21
1.7 Aplikasi Machine Learning	24
BAB II TEORI TERKAIT MACHINE LEARNING.....	29
2.1 Kemampuan Komputasi.....	29
2.2 Pondasi Matematika dan Statistika.....	50
BAB III MACHINE LEARNING LIFECYCLE	65
3.1 Pengumpulan Data (Data Aquisition).....	66
3.2 Persiapan Data	72
3.3 Pelatihan dan Uji Coba Model	91
3.4 Model Deployment.....	96
3.5 Studi Kasus : Pengumpulan Data dengan Metode Scrapping berita detik.com	97
3.6 Studi Kasus - Data Preprocessing.....	102
BAB IV K-NEAREST NEIGHBORS.....	108
4.1 Definisi K-Nearest Neighbors (KNN)	108
4.2 Jarak Data KNN	109
4.3 Implementasi K-NN.....	120
4.4 Keunggulan serta Kelemahan K-NN	121
4.5 Studi Kasus – Klasifikasi Buah	123
4.6 K-Dimensional Tree	126
BAB V ROBOT NAVIGATION DENGAN REINFORCEMENT LEARNING	128
5.1 Penjelasan: Robot Navigation dengan Reinforcement Learning	128
5.2 Implementasi Inteligensi Artifisial (DQN) pada Optimasi Inventaris.....	161

BAB VI SKILL ACQUISITION	166
6.1 Skill Aquisition.....	166
6.2 Dasar Teori Hierarchical Reinforcement Learning (HRL)	169
6.3 Tahapan Proses Skill Acquisition (HRL)	169
6.4 Kelebihan dan Tantangan.....	170
BAB VII LEARNING TASK.....	179
7.1. Pembahasan Tahapan Learning Task	179
7.2. Penerapan tahapan learning task	189
7.3. Analisis Alur Learning Task	193
BAB VIII GAME AI REAL TIME DECISION	200
8.1. Pembahasan Tahapan Proses.....	200
8.2. Menghubungkan AI ke Game Engine (Unity & Unreal).....	206
8.3. Menggunakan Normalisasi (0 ke 1).....	212
8.4 Struktur Logika Behavior Tree.....	212
8.5 Menggunakan Pendekatan Hybrid.....	214
8.6 Analisis Data Debugger	216
8.7 Matriks Evaluasi Pengujian.....	221
BAB IX EVALUASI REINFORCEMENT LEARNING	224
DAFTAR PUSTAKA.....	230
BIOGRAFI PENULIS	234

DAFTAR GAMBAR

Gambar 1.1 Kerangka Mengambil Keputusan Berdasarkan Pengalaman	2
Gambar 1.2 Kategori Berat Badan Berdasarkan BMI	3
Gambar 1.3 Digit Angka yang ditulis menggunakan tulisan tangan (sumber: lecun.com)	4
Gambar 1.4 Perbedaan Deep Learning vs Machine Learning dalam mengambil keputusan.....	5
Gambar 1.5 Beberapa Penemuan penting dibidang Machine Learning.....	9
Gambar 1.6 Pembagian cabang-cabang ilmu data (Sumber: rifqimulyawan.com)	10
Gambar 1.7 Sekumpulan data Kelinci (kotak kiri) dan data Tupai (kotak kanan) yang akan digunakan untuk mempelajari objek Kelinci dan Tupai.	12
Gambar 1.8 Pemetaan data training pada feature space 2 dimensi.....	14
Gambar 1.9 Ilustrasi hasil pelatihan model menggunakan fungsi linear	15
Gambar 1.10 Sekumpulan data Kelinci (kotak kanan) dan data Tupai (kotak kiri) yang akan digunakan sebagai data uji	16
Gambar 1.11 Sekumpulan data Kelinci (kotak kiri) dan data Tupai (kotak kanan) yang akan digunakan sebagai data uji	16
Gambar 1.12 Masalah yang dapat diselesaikan dengan ML (sumber: Medium.com). ..	24
Gambar 1.13 Rekomendasi kata kunci pencarian Google	25
Gambar 1.14 Rekomendasi kata kunci pencarian Google	26
Gambar 1.15 Mengenali bagian wajah (sumber: Oxygen Forensic)	26
Gambar 1.16 Implementasi kendali otomatis mobil (sumber:dqlab.id).....	28
Gambar 2.1 Perkembangan Python (sumber: google trend,2021).....	30
Gambar 2.2 Aplikasi Jupyter Notebook	33
Gambar 2.3 Ilustrasi Hubungan Series dan DataFrame	42
Gambar 2.4 Anatomi sebuah chart (sumber: matplotlib.org)	46
Gambar 2.5 Topik Matematika Penting Pada Bidang Machine Leaning (sumber: Wale	51
Gambar 2.6 Cara Manusia & Komputer Melihat Gambar	52
Gambar 2.7 Perbedaan Scalar, Vektor, Matrik dan Tensor.....	52
Gambar 2.8 Vektor berukuran 2 menggambarkan posisi titik x dengan koordianat x_1, x_2 (gambar a) dan sebuah vektor x yang merepresentasikan perpindahan sejauh x_1 pada sumbu x dan x_2 pada sumbu y (gambar b).....	54
Gambar 2.9 Skewness pada data (www.savemyexams.com).....	61
Gambar 2.10 Quartile membagi data yang diurutkan menjadi 4 bagian. Setiap bagian yang terbagi disebut dengan quarter.....	62
Gambar 3.1 Machine Learning LifeCycle (www.geeksforgeeks.org).....	65
Gambar 3. 2 Komposisi yang menyita waktu data saintis (sumber: Forbes.com)	66
Gambar 3.3 Pembagian Tipe Data Statistik.....	68
Gambar 3.4 Visualisasi dalam bentuk Scatter Plot	77
Gambar 3.5 Visualisasi dalam bentuk Histogram.....	77
Gambar 3.6 Visualisasi Menggunakan Boxplot	78
Gambar 3.7 Contoh Correlation matrix	84

Gambar 3.9 Ilustrasi holdout sets	87
Gambar 3.10 Ilustrasi Leave one out cross-validation.....	90
Gambar 3.11 Ilustrasi K-Fold cross-validation.....	90
Gambar 3.12 Ilustrasi Stratified cross-validation	91
Gambar 3.13 Proses Ilustrasi Pelatihan dan Pengujian Model	92
Gambar 3.14 Jenis-Jenis Metrik Evaluasi Model berdasarkan task	93
Gambar 3.15 Contoh confusion matrix dengan dua kelas yaitu yes dan no	93
Gambar 3.16 Contoh confusion matrix dengan dua kelas yaitu yes dan no	94
Gambar 3.17 Contoh grafik ROC	95
Gambar 3.18 Inspeksi terhadap elemen data detik.com.....	98
Gambar 3.19 Struktur Index berita detik.com	100
Gambar 3.23 Luaran Berita Detik.com tanggal 22 Mei 2021	101
Gambar 4.1 Ilustrasi KNN.....	108
Gambar 4.2 Ilustrasi Euclidean distance	110
Gambar 4.3 Ilustrasi Manhattan distance	112
Gambar 4.4 Ilustrasi Chebyshev distance	113
Gambar 4.5 Ilustrasi Cosine distance.....	115
Gambar 4.6 Jarak Hamming.....	117
Gambar 4.7 Ilustrasi Haversine distance.....	118

DAFTAR TABEL

Tabel 1.1 Data sampel Kelinci dan Tupai (telinga dan hidung)	13
Tabel 1.2 Data uji fitur Kelinci dan Tupai (bentuk telinga dan ukuran hidung).....	18
Table 3.1 Contoh dataset play tenis (sumber:UCI machine learning)	69
Tabel 5.1 Perbandingan ROP Tradisional vs. AI "Riak Air"	142
Tabel 5.2 Perbandingan Kalkulasi tradisional vs AI “Riak Air”	143
Tabel 5.3 Perbandingan ROP Statis vs Dinamis.....	145
Tabel 5.4 Desain Reward Function.....	145
Tabel 5.5 Analisis Reward Hacking.....	146
Tabel 5.6 Desain Fungsi Reward Pinalti Expired	147
Tabel 5.7 Simulasi Strategi Stok Awet vs Cepat Basi	149
Tabel 7.1 Gambaran jenis Task berdasarkan interaksi dengan lingkungan.....	181
Tabel 7.2 Rekayasa fitur berdasarkan format atau skala data.....	184
Tabel 7.3 Penerapan Learning Task	191
Tabel 7.4 Alur Learning Task perancangan rumah	194
Tabel 8.1 AI Musuh dalam Game Shooter (FPS)	202
Tabel 8.2 : Ringkasan Perbandingan Fitur	204
Tabel 8.3 AI Musuh FPS	206
Tabel 8.4 Kombinasi antara wortel dengan cambuk	217

BAB I

PEMAHAMAN *MACHINE LEARNING*

Manusia sebagai makhluk yang senang belajar dan memiliki kemampuan untuk belajar yang sangat hebat, di mana proses belajar ini dimulai sejak dia lahir hingga akhir hidupnya.

Seiring berkembangnya jaman, Manusia mulai belajar berjalan, berbicara, berkomunikasi, membaca, menulis, dan berpikir untuk mengumpulkan dan memproses informasi, yang memungkinkan mereka untuk bertindak dan mengambil keputusan (R., & Madanagopal, 2020). Salah satu cara manusia belajar dan mengambil keputusan serta sering dilakukan merupakan berdasarkan pengalaman (Kolb, n.d.). Sebagai contoh ketika seorang anak menemukan jeruk untuk pertama kalinya, maka tanpa ragu anak tersebut akan memakan jeruk tersebut. Akibatnya dia akan merasa “keasaman” dan “tidak nyaman”. Pengalaman keasaman tersebut karena memakan jeruk akan menjadi sebuah pembelajaran baginya untuk berfikir mengenai keasaman jeruk. Dari pengalaman tersebut sang anak belajar mengenai jeruk memiliki sifat asam dan dapat menimbulkan rasa tidak nyaman, maka dia tidak akan berani untuk melahap jeruk tersebut atau setidaknya akan berhati-hati terhadap jeruk.

Berdasarkan contoh tersebut, bisa disimpulkan bahwa salah satu cara mengambil keputusan adalah dengan memakai pengalaman yang sudah pernah dialami sebelumnya. Proses pengambilan keputusan yang didasarkan pada pengalaman minimal terdiri dari tiga tahap berikut pada Gambar 1.1:

1. **Memorize.** Anak itu mengingat semua informasi dan peristiwa yang pernah terjadi, baik yang sudah terjadi sebelumnya maupun yang serupa dan terkait. Bagaimana rasanya jeruk, bentuknya jeruk, atau warnanya jeruk.
2. **Generalize.** Berdasarkan data-data tersebut, si anak akan membuat sebuah "pola" atau "aturan umum" dari kejadian tadi.
3. **Predict.** Ketika menghadapi kasus yang berhubungan atau serupa, aturan atau pola tersebut digunakan untuk memperkirakan tindakan yang akan dilakukan. Misalnya, jika anak diberi makanan yang terdiri dari potongan jeruk, maka dia akan langsung menunjukkan reaksinya.



Gambar 1.1 Kerangka Mengambil Keputusan Berdasarkan Pengalaman

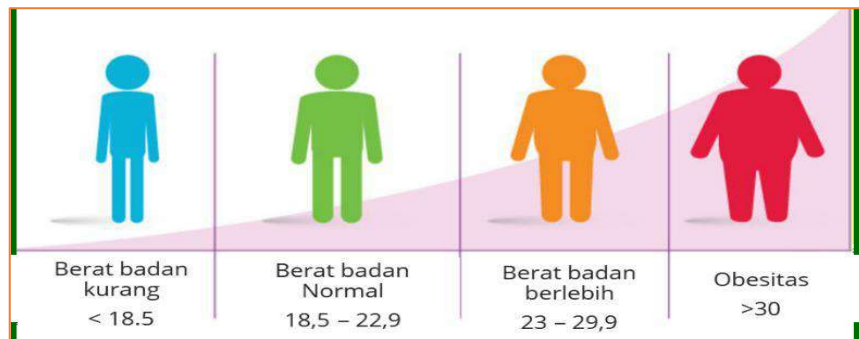
Komputer berbeda dengan manusia, karena ia hanya melakukan apa yang diperintahkan oleh orang yang membuat programnya. Agar bisa menyelesaikan sebuah masalah, komputer memerlukan sebuah algoritma yang terdiri dari langkah-langkah instruksi yang harus dijalankan oleh komputer untuk memecahkan masalah tersebut atau mengubah input menjadi output Turing, A. M. (1950). In *Computing Machinery and Intelligence. Mind* (Vol. 49). Instruksi-instruksi tersebut akan direalisasikan dalam bahasa pemrograman tertentu, dan tugas komputer hanya mengeksekusi baris-baris perintah tersebut Knuth, D. E. (1997). *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*. Reading, Mass.: Addison-Wesley.

Misalnya, perhatikan Gambar 1.2 yang menunjukkan kategori kondisi berat badan berdasarkan BMI atau Indeks Massa Tubuh. Kategori tersebut bisa dibagi menjadi 4 kelompok, yaitu Underweight, Normal, Overweight, dan Obese. Indeks Massa Tubuh adalah indeks yang paling sederhana dan paling banyak digunakan untuk menentukan tingkat kegemukan dan obesitas. Cara menghitungnya adalah dengan membagi berat badan seseorang dalam kilogram dengan tinggi badannya dalam meter yang dikalikan dua (kg/m^2). Setelah mendapatkan nilai BMI, aturan berikutnya digunakan untuk menentukan kelompoknya.

- Jika nilai BMI kurang dari 18.5 maka Pembaca termasuk dalam kategori underweight.
- Jika nilai BMI berada di antara 18,5 hingga 24,9 maka masuk ke kategori normal.
- Jika nilai BMI berada di antara 25 hingga 29.9, maka dikatakan berlebihan berat badan.
- Jika besar dari 30 dikategorikan Obese.

Menerapkan pengukuran BMI dalam aplikasi komputer dapat dilakukan dengan cara yang mudah. Dengan memanfaatkan perintah seleksi seperti if/else, masalah tersebut bisa teratasi. Aplikasi yang dihasilkan akan mampu mengelompokkan dengan tepat. Proses kategorisasi dimulai dengan standar yang sudah ditentukan, di

mana komputer hanya diberi instruksi tertentu lalu menjalankan tugas sesuai dengan instruksi itu.



Gambar 1.2 Kategori Berat Badan Berdasarkan BMI

Aplikasi ini tidak bisa belajar sendiri dan hanya menjalankan perintah yang sudah ditentukan.

Berbeda dengan standar BMI, dari sudut pandang pembelajaran mesin, sebuah komputer berusaha mempelajari sesuatu dengan data yang diberikan. Komputer hanya diberi beberapa data dari empat kategori BMI lalu mempelajari bagaimana cara memahami data tersebut.. Mitchell, T. M. . (1997). Machine Learning. McGraw-Hill. Misalnya, komputer diberikan daftar berat badan dari 24 orang yang berasal dari 4 kategori tersebut di mana:

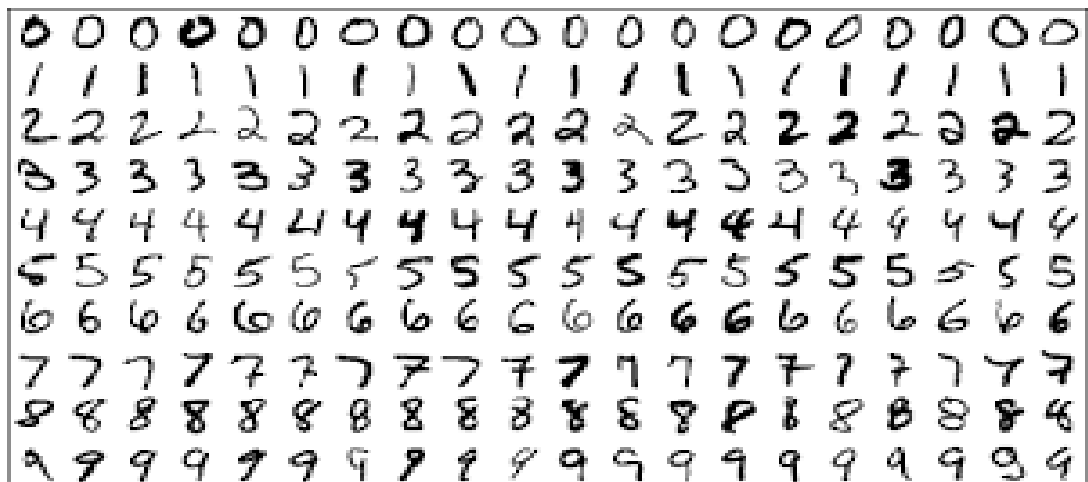
- Kategori Underwight merupakan {15.4, 17, 18,16,18,16.5}
- Kategori Normal merupakan {19, 20, 22.8, 22, 21, 23}
- Kategori Overwight merupakan {29, 26, 27, 28,24.9, 25}
- Kategori Obese merupakan {39, 45, 32, 41, 33, 37}.

Berdasarkan data itu, komputer mempelajari ciri-ciri orang berdasarkan kategorinya, sehingga bisa memperkirakan kategori BMI ketika diberi informasi berat badan. Apakah semua masalah bisa diselesaikan dengan pemrograman konvensional? Faktanya, terdapat kasus-kasus tertentu yang tidak dapat diselesaikan hanya dengan algoritma kaku (berbasis aturan tetap), salah satunya merupakan deteksi email spam. Berbeda dengan perhitungan BMI yang memiliki rumus baku, email spam tidak memiliki kriteria tunggal yang pasti. Karakteristik spam cenderung sangat variatif dan dinamis mengikuti perkembangan waktu. Selain itu, penyelesaian masalah ini sangat bergantung pada pengetahuan tentang isi dan ciri dokumen yang subjektif, karena preferensi mengenai apa yang dianggap 'spam' dapat berbeda-beda bagi setiap pengguna..

Sebagai contoh, seorang pemasar (*marketing*) sering kali mengirimkan email berisi promosi produk, diskon, hingga penawaran kerja sama. Namun, bagi seorang

pelajar yang belum berprestasi, email promosi tersebut bisa dianggap sebagai spam. Hal ini menunjukkan bahwa klasifikasi spam sangat bergantung pada preferensi pengguna, sehingga tidak ada aturan baku yang berlaku umum bagi setiap individu. Untuk menangani berbagai variasi kasus seperti ini secara lebih efektif, para ahli komputer mereplikasi proses pembelajaran manusia ke dalam mesin. Inilah yang melahirkan konsep Machine Learning..

Pemrograman secara eksplisit tidak efektif adalah proses mendeteksi angka (0-9) pada gambar tulisan tangan. Dengan ukuran gambar 100x100 pixel, jumlah kombinasi dari input yang hanya terdiri dari satu digit sangat beragam, sehingga tidak mungkin membuat sebuah program yang ditulis secara eksplisit oleh LeCun, Y., & Cortes, C.(2010). MNIST handwritten digit database. Yann LeCun's Website. Pada gambar 1.3 berikut ini menunjukkan sebagian dari gambar digit tulisan tangan. Untuk mengenali angka 0, terdapat banyak kemungkinan cara untuk membuat huruf tersebut.

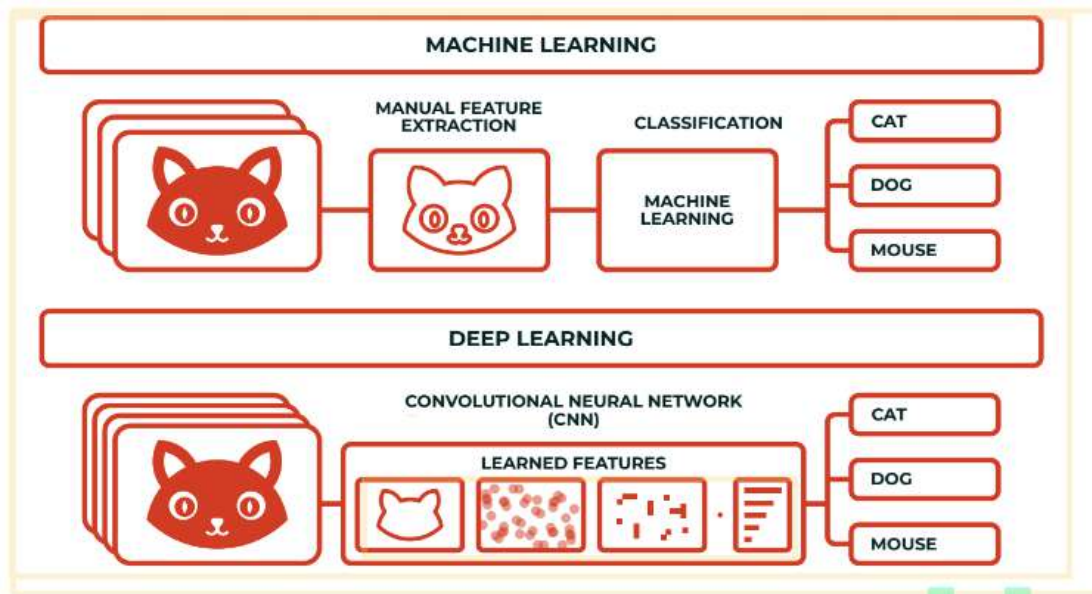


Gambar 1.3 Digit Angka yang ditulis menggunakan tulisan tangan (sumber: lecun.com)

1.1 Definisi *Machine Learning*

Machine Learning (ML) adalah bidang ilmu yang berfokus pada desain dan analisis algoritma agar komputer bisa belajar sendiri. Menurut Artur Samuel tahun 1959, ML adalah sebuah algoritma yang umum, di mana algoritma tersebut mampu menghasilkan sesuatu yang menarik atau berguna dari sekumpulan data, tanpa perlu menulis kode yang khusus. Pada algoritma yang umum itu, jika diberikan beberapa data, maka ia bisa membuat aturan, model, atau kesimpulan dari data tersebut. Sebagai contoh, sebuah algoritma yang digunakan untuk mengenali tulisan tangan bisa juga digunakan untuk menentukan apakah sebuah email adalah spam atau bukan spam tanpa mengganti kode. Jika algoritma yang sama diberikan data latihan yang berbeda, maka logika dalam mengklasifikasikan data juga akan berbeda. Machine learning juga dapat

diartikan sebuah komputer yang memiliki kemampuan belajar tanpa diprogram secara eksplisit. Program tersebut memanfaatkan data untuk membangun model dan mengambil keputusan berdasarkan model yang telah dibangun.



Gambar 1.4 Perbedaan Deep Learning vs Machine Learning dalam mengambil keputusan

Selain Samuel, Mitchell (1997) memberikan definisi yang ringkas dan jelas mengenai Machine Learning (ML). Menurutnya, sebuah program komputer dikatakan belajar dari pengalaman (E) terkait dengan sejumlah tugas (T) dan ukuran kinerja (P), jika kinerjanya pada tugas-tugas di T, sebagaimana diukur oleh P, meningkat seiring dengan bertambahnya pengalaman E.

Berdasarkan definisi tersebut, sebuah aplikasi *Machine Learning* setidaknya memiliki tiga komponen utama, yaitu:

- Task (T): Tugas atau pekerjaan yang dilakukan.
- Performance Measure (P): Ukuran untuk menilai keberhasilan kinerja.
- Experience (E): Pengalaman atau data yang dipelajari.

Oleh karena itu, dalam membangun aplikasi ML, ketiga komponen (T, P, dan E) ini harus dapat diidentifikasi dengan jelas sejak awal. Task (T) merupakan objektif utama dari program yang dibuat. ML membantu kita mengatasi masalah kompleks yang sulit diselesaikan menggunakan pemrograman tradisional yang bersifat statis. Beberapa contoh Task dalam program ML Mitchell, T. M. . (1997). *Machine Learning*. McGraw-Hill:

1. Klasifikasi adalah proses pengelompokan data berdasarkan kelas atau label tertentu yang telah ditentukan. Dalam jenis tugas ini, program komputer diminta untuk menebak atau memilih kelompok, kategori, atau kelas dari data yang diberikan. Pada task ini kelas telah ditetapkan sebelumnya. Contoh lainnya adalah pengelompokan perasaan pada gambar manusia dengan tiga jenis yaitu Senang, Marah, dan Netral. Aplikasi itu harus memilih salah satu (atau di beberapa kasus lebih dari satu) dari tiga kelas tersebut berdasarkan foto yang diberikan. Regresi sederhana adalah sebuah tugas di mana aplikasi pembelajaran mesin diminta untuk menebak angka (bilangan yang bisa diulang) berdasarkan sejumlah data yang sudah diberikan. Misalnya, kita ingin memperkirakan harga saham, maka hasil dari peramalan tersebut adalah sebuah angka yang merepresentasikan harga. Jadi, salah satu perbedaan antara regresi dan klasifikasi adalah pada objek yang diprediksi. Regresi digunakan untuk memprediksi angka, sedangkan klasifikasi digunakan untuk memprediksi kelas.
2. Transkripsi adalah tugas satu jenis di mana aplikasi pembelajaran mesin berusaha "memahami" data yang tidak terstruktur yang mewakili bentuk data lainnya. Misalnya, pengenalan tulisan pada gambar dimana gambar tersebut terdiri dari data pixel yang belum terorganisir, tetapi data tersebut bisa mewakili sebuah tulisan atau beberapa huruf. Contoh lainnya dari aplikasi text to speech adalah ketika data berupa gelombang yang berisi teks. Proses mengubah gelombang suara menjadi teks transkripsi.
3. Machine Translation. Tugas ini adalah proses mengubah sekelompok input yang sudah diurutkan dan berisi simbol-simbol tertentu menjadi simbol-simbol yang berbeda dalam bahasa lainnya. Contoh penerapannya adalah penerjemah antara bahasa Indonesia dan bahasa Inggris.
4. Anomaly Detection. Task ini bertujuan untuk memanfaatkan aplikasi pembelajaran mesin untuk memeriksa sejumlah data atau peristiwa, lalu menandainya sebagai sesuatu yang tidak biasa.
5. Syntesis dan Sampling. Tugas ini bertujuan untuk membuat atau menghasilkan sesuatu berdasarkan contoh yang diberikan. Misalnya, aplikasi pembelajaran mesin yang bisa membuat gambar pemandangan dari sketsa yang sudah dibuat.

Untuk menilai bagaimana baiknya aplikasi pembelajaran mesin bekerja, diperlukan cara mengukur kinerjanya secara kualitatif. Setiap tugas memiliki cara pengukuran yang berbeda, misalnya antara klasifikasi dan regresi, cara menilai kinerjanya juga berbeda. Klasifikasi mengevaluasi kinerja dengan membandingkan jumlah jawaban yang benar dan salah, sedangkan regresi dinilai berdasarkan sejauh mana hasil tebakan mendekati nilai sebenarnya. Macam-macam metode evaluasi bisa dilihat pada bab ketiga.

Aplikasi atau model pembelajaran mesin memperoleh pengalaman berdasarkan data yang diberikan selama proses pelatihan. Sebuah dataset adalah kumpulan contoh-contoh yang harus dipelajari oleh komputer agar bisa menyelesaikan tugasnya. Informasi yang lebih rinci tentang dataset akan dibahas di bab berikutnya. 1.2 Sejarah Singkat ML

1.2 Gambaran Ringkas ML

Istilah Machine Learning pertama kali diperkenalkan oleh Arthur Samuel, pionir pengembangan gim komputer dan kecerdasan buatan, pada tahun 1959. Samuel, A.L. (1959) Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development, 44, 206-226.. Bidang ini berkembang pesat seiring kemajuan Artificial Intelligence (AI) dan besarnya minat peneliti untuk menciptakan mesin yang mampu belajar dari data. Untuk mencapai tujuan tersebut, para ahli menggunakan berbagai pendekatan, mulai dari metode simbolik, jaringan saraf tiruan (neural networks), penalaran probabilistik, hingga berbagai model statistik..

Masa sebelum-1920an

Thomas Bayes, Adrien-Marie Legendre, Andrey Markov, serta para matematikawan lainnya mulai melakukan penelitian yang menjadi dasar teknik-teknik pada konsep-konsep pembelajaran mesin. Sejarah machine learning bisa dianggap dimulai dari penelitian Thomas Bayes yang dilakukan pada tahun 1763 dan kemudian dipublikasikan oleh temannya, Richard Price. Penelitian itu dilakukan oleh Pierre-Simon Laplace, sehingga sekarang dikenal sebagai Theorema Bayes tahun 1812. Selanjutnya, ada penelitian penting lainnya yang dilakukan oleh Adrien-Marie Legendre, seorang matematikawan dari Prancis. Ia mengembangkan metode Least Squares yang bertujuan untuk memasukkan data dengan menggunakan pendekatan aljabar pada tahun 1805. Selain itu, Andrey Markov juga menjelaskan sebuah teknik yang ia gunakan untuk menganalisis puisi. Teknik tersebut dikenal dengan Markov Chain. Teori-teori mereka itu menjadi dasar utama untuk pembelajaran mesin.

Masa 1940-1950an

Pada tahun 1943, McCulloch dan Pitts membuat model matematika yang diinspirasi cara kerja otak manusia, di mana dalam model itu terdapat neuron-neuron yang bisa menyala atau mati seperti saklar yang dinyalakan atau dimatikan. Selain itu, neuron-neuron tersebut mampu belajar dan memberikan respons yang berbeda tergantung pada input yang diterimanya.

Salah satu sumbangan besar lainnya datang dari Alan Turing, pada tahun 1950, ia membuat mesin Turing untuk menjawab pertanyaan “Apakah komputer bisa berpikir?”

Artikel Alan Turing tahun 1950 berjudul “Computing Machinery and Intelligence” membahas kriteria yang dibutuhkan agar sebuah mesin dianggap cerdas. Dia berpikir bahwa jika mesin bisa berperilaku seperti manusia dengan baik, maka kita bisa menganggapnya sebagai sesuatu yang cerdas.

Pengembangan berikutnya, pada tahun 1959, Allen Newell dan Herbert Simon membuat sebuah proyek yang dikenal sebagai General Problem Solver atau GPS. GPS berhasil menyelesaikan masalah manusia dengan menggunakan teknik analisis tujuan-ke-sasaran.

Masa 1960-1990an

Tahun 1962, Frank Rosenblatt menemukan sesuatu yang luar biasa, yaitu teorema Perceptron. Ia berhasil membuat aturan klasifikasi yang mampu membedakan objek dari dua kelompok yang berbeda.

Selanjutnya, Rumelhart pada tahun 1986 mencoba meningkatkan sistem Perceptron yang hanya memiliki satu lapisan menjadi sistem dengan beberapa lapisan. Metode ini kemudian dikenal sebagai back-propagation. Konsep back-propagation sebenarnya sudah ada sebelumnya, tetapi Rumelhart dan Hinton yang pertama kali membuatnya terkenal dan menunjukkan bahwa metode ini bisa diterapkan. Setelah itu, muncul beberapa model jaringan saraf tiruan lainnya yang dikembangkan oleh Kohonen tahun 1972, Hopfield tahun 1982, serta peneliti lainnya.

Selain algoritma berbasis jaringan saraf, di Rusia juga berkembang beberapa algoritma Generalized Portrait pada tahun 1960-an, yaitu tahun 1963 dan 1964. Algoritma itu didasarkan pada teori statistik pembelajaran yang sudah dikembangkan selama tiga puluh tahun oleh Vapnik dan Chervonenkis. SVM (Support Vector Machines) diperkenalkan oleh Vapnik, Boser, dan Guyer pada tahun 1992 dalam sebuah konferensi yang disebut COLT dan juga dalam sebuah jurnal. Sejak itu, metode ini berkembang dengan sangat pesat.

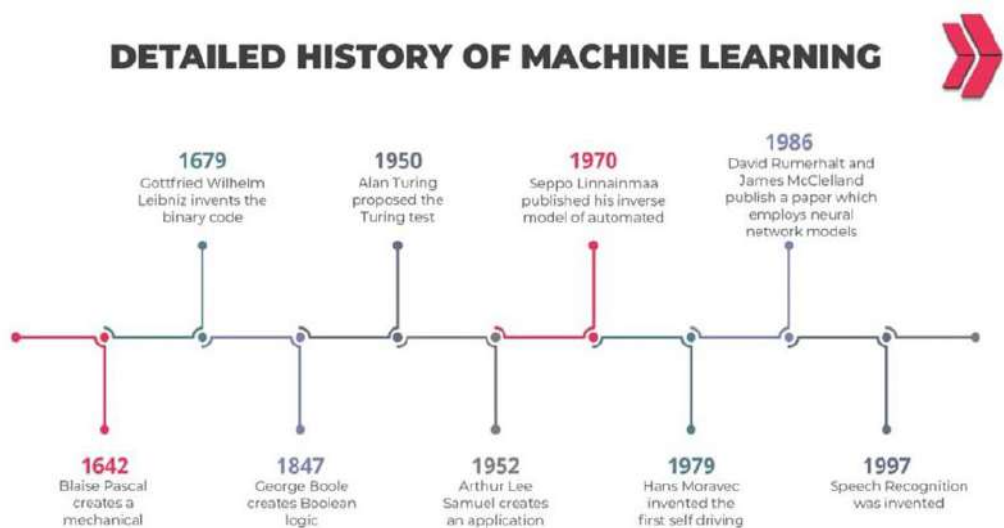
SVM pertama kali diperkenalkan ke dunia sejak akhir tahun 1970-an oleh Vapnik pada tahun 1979, namun Support Vector Machines tidak mendapat perhatian yang banyak dari para peneliti. Setelah itu, Vladimir Vapnik kembali menerbitkan bukunya di tahun 1990-an, yaitu pada tahun 1995 dan 1998. Sejak itu, SVM menjadi metode yang banyak digunakan dan masih aktif dipelajari dalam bidang Machine Learning.

Masa 2000an

Di akhir tahun 90-an hingga pertengahan tahun 2000-an, neural networks nyaris tidak digunakan lagi karena munculnya beberapa algoritma lain seperti Support Vector Machines dan AdaBoost yang bisa berjalan lebih cepat dan memberikan hasil yang

lebih baik pada masa itu. Neural networks kembali menjadi perhatian utama ketika Deep Belief Networks (DBN) membuat kemajuan besar dengan menjadi model yang paling akurat dalam mengenali tulisan tangan digit, yang akhirnya mengantarkan munculnya istilah Deep Learning.

Istilah Deep Learning semakin dikenal karena pengenalan Convolution Neural Network (CNN). CNN dengan desain khusus yang dikombinasikan dengan berbagai metode seperti dropout untuk regulasi, penggunaan ReLU sebagai fungsi aktivasi, serta peningkatan data melalui augmentasi, membuatnya mampu mengklasifikasi gambar dalam jumlah yang sangat besar, seperti dalam dataset ImageNet. ImageNet memiliki 1.000 kategori objek dan 1 juta gambar, sehingga performanya lebih baik dari manusia.



Gambar 1.5 Beberapa Penemuan penting dibidang Machine Learning

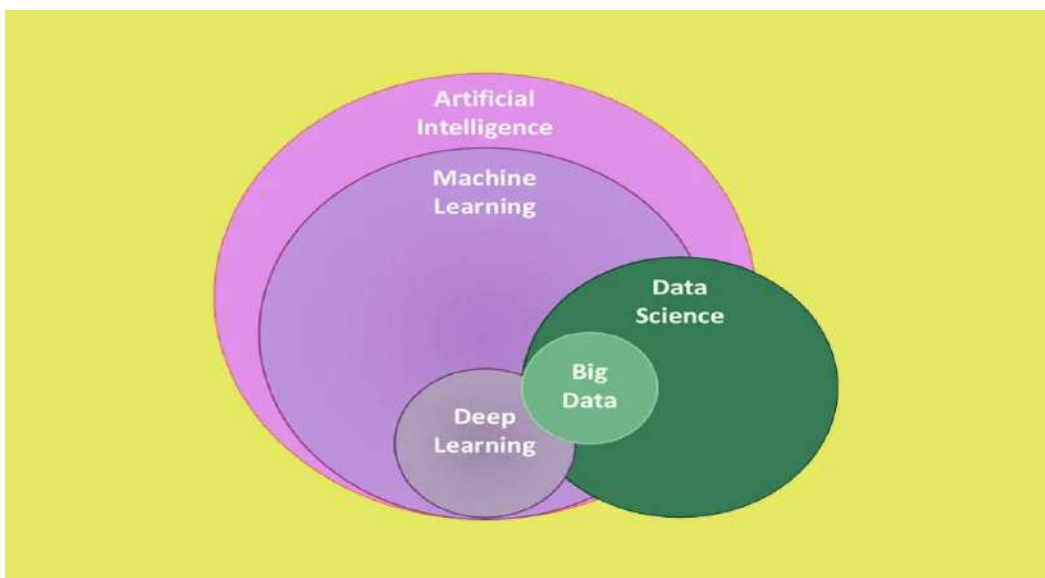
1.3 Machine Learning bagian AI

Jika kita lihat kembali, cara manusia mengambil keputusan tidak hanya melalui pengalaman, tetapi juga bisa menggunakan logika. Seperti yang sudah dikatakan sebelumnya, machine learning adalah bagian dari artificial intelligence yang bertugas memberi kemampuan kepada komputer untuk belajar dari data baru tanpa perlu diinstruksikan secara langsung. Fokus utamanya adalah membuat sebuah aplikasi komputer yang mampu belajar dari data, kemudian membuat model yang siap digunakan untuk menyelesaikan kasus tertentu.

Berbeda dengan Data Mining, di mana prosesnya dimulai dengan mengumpulkan database yang ukurannya besar, yang sebelumnya tidak bisa dipahami dan tidak diketahui, lalu menggunakan informasi tersebut untuk membuat keputusan bisnis yang sesuai. (Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P, 1996).

Secara sederhana, Data Mining adalah sekumpulan metode yang digunakan untuk menemukan pengetahuan dalam bentuk hubungan dan pola dari database yang besar, dan hal-hal tersebut sebelumnya belum diketahui.

Oleh karena itu, bisa disimpulkan bahwa Data Mining merupakan pertemuan antara berbagai bidang seperti kecerdasan buatan, basis data, pengenalan pola, visualisasi data, pembelajaran mesin, statistik, dan lain sebagainya. Tujuan utama dari proses data mining adalah mengambil informasi dari berbagai kumpulan data yang jumlahnya besar, lalu mengubahnya menjadi struktur yang tepat dan mudah dipahami untuk digunakan oleh orang lain. Dapat disimpulkan bahwa Data Mining adalah proses yang digunakan oleh para ahli data dan penggemar pembelajaran mesin untuk mengubah dataset yang besar menjadi sesuatu yang lebih berguna.



Gambar 1.6 Pembagian cabang-cabang ilmu data (Sumber: rifqimulyawan.com)

Secara umum, ML dan Data Mining mengikuti proses yang hampir sama. Perbedaan utamanya adalah Data Mining dibuat untuk mengekstrak informasi dari sekumpulan data yang besar, sedangkan Machine Learning bertujuan agar komputer bisa belajar dari data tersebut. Secara sederhana, Data Mining adalah sebuah metode untuk mengekstrak hasil tertentu dari sekumpulan dataset yang besar. Luarannya adalah sebuah informasi, sedangkan machine learning memiliki luaran berupa sebuah model yang bisa melakukan fungsi tertentu.

Pembelajaran mesin menggunakan metode analisis data yang bertugas untuk mengotomatiskan pembuatan model secara analitis. Menggunakan algoritma yang bekerja secara berulang untuk mempelajari dari data, dan dalam proses ini komputer

dapat menemukan pengetahuan yang seolah-olah tersembunyi, tanpa perlunya bantuan dari pihak luar.

Selain istilah Data Mining, terdapat istilah lain yang juga cukup membingungkan, yaitu Data Science. Data science mencakup berbagai kegiatan seperti mengumpulkan data yang terstruktur dan tidak terstruktur, termasuk proses pembersihan, persiapan, dan analisis data secara menyeluruh. Selain itu, data science juga melibatkan penerapan metode machine learning dan analisis prediktif untuk mengekstrak informasi penting dari dataset. Mason, H., & Wiggins, C. (2010). A taxonomy of data science. Informasi yang didapat selalu dilihat dari perspektif bisnis, sehingga bisa dipakai untuk memperkirakan masa depan atau mendapatkan pemahaman lebih dalam yang membantu dalam mengambil keputusan bisnis. Data science menggabungkan pemrograman, berpikir logis, matematika, dan statistik.

Data Science adalah cara yang cerdas untuk mengumpulkan dan memahami data, serta membantu melihat masalah atau peluang dari sudut pandang yang berbeda sesuai dengan kebutuhan dan strategi bisnis perusahaan. Kekuatan Data Science merupakan visualisasinya. Kemampuannya dalam bidang visualisasi membuat ilmu ini berbeda dengan Machine Learning, karena dengan visualisasi yang tepat, pengguna bisa memanfaatkan pengetahuan ke dalam bidang bisnis.

1.4 Contoh Aplikasi ML Sederhana

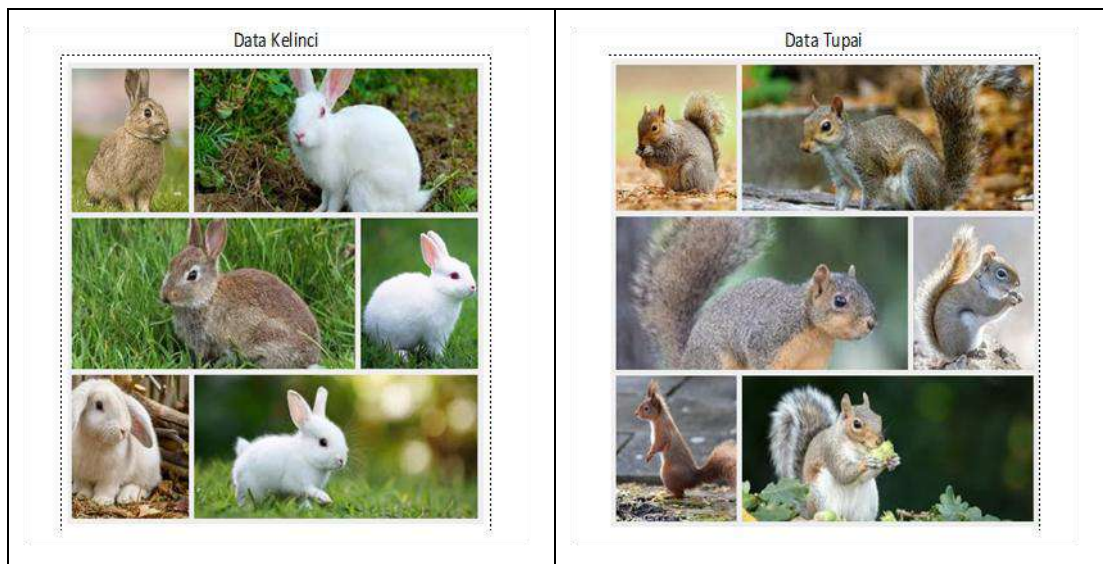
Membedakan tupai dan kelinci bukanlah hal yang sulit bagi manusia. Dengan memperhatikan beberapa contoh, otak manusia sudah mampu membedakan mana yang tupai dan mana yang kelinci. Tidak perlu cara yang rumit seperti berpikir secara ilmiah untuk membedakan mereka, cukup dengan mengenali ciri-ciri fisiknya, manusia sudah bisa membedakannya. Di bab ini, kita akan mencoba menjelaskan bagaimana komputer bisa mengenali Tupai dan Kelinci. Oleh karena itu, beberapa tahap perlu dilakukan, diantaranya:

1. Pengumpulan data gambar Kelinci dan Tupai.

Seperti manusia, komputer juga perlu dikenalkan dengan objek-objek yang akan dipelajari, yaitu objek Tupai dan objek Kelinci. Cara satu untuk “mengenalkan” Kelinci dan Tupai adalah dengan memberikan beberapa gambar Tupai dan Kelinci. Dalam kasus ini, kelinci dan tupai bisa dikatakan termasuk dalam kategori atau kelas yang sama.

Tugas utama komputer adalah memasukkan sebuah data ke dalam salah satu kategori yang telah ditentukan. Proses ini biasa disebut klasifikasi. Data yang digunakan untuk proses belajar biasanya disebut dengan data training. Data pelatihan harus berisi informasi yang dilengkapi dengan kategori atau label data. Dalam kasus ini, kita akan menggunakan sekelompok gambar yang telah ditentukan apakah gambar tersebut

menunjukkan kelinci atau tupai. Gambar 1.7 menampilkan sebuah dataset yang telah diberi label sebagai Kelinci dan Tupai. Data tersebut bisa didapat dari berbagai sumber atau dibuat langsung oleh pengembang.



Gambar 1.7 Sekumpulan data Kelinci (kotak kiri) dan data Tupai (kotak kanan) yang akan digunakan untuk mempelajari objek Kelinci dan Tupai.

2. Mendesain Fitur Kelinci dan Tupai.

Perhatikan dengan baik, lalu pelajarilah ciri-ciri yang membedakan kedua makhluk tersebut. Apakah warna, jenis bulu, ukuran, jenis telinga, atau ukuran hidung? Misalnya, fitur warna, apakah kita bisa membedakan antara kelinci dan tupai berdasarkan warna mereka? Jika bisa, gunakan fitur warna tersebut, jika tidak, coba periksa fitur lainnya, misalnya jenis telinga. Jika diperhatikan, terdapat perbedaan yang jelas antara telinga tupai dan telinga kelinci, maka fitur ini bisa dimanfaatkan. Pada intinya, carilah ciri atau fitur yang bisa membedakan kedua hewan tersebut. Fitur-fitur yang telah ditemukan akan digunakan oleh komputer sebagai bahan latihan data.

Proses memilih fitur atau mengekstrak fitur sangat penting karena akan mempengaruhi hasil yang dicapai dan sangat tergantung pada kasus tertentu. Misalnya fitur "jumlah kaki", fitur ini tidak terlalu berguna dalam kasus ini, tetapi untuk membedakan antara kelinci dan ayam, fitur ini sangat penting. Selanjutnya, fitur tersebut akan berbeda untuk setiap kasus dan proses untuk menemukannya cukup sulit serta bisa membingungkan karena tingkat kesamaannya yang sangat detail.

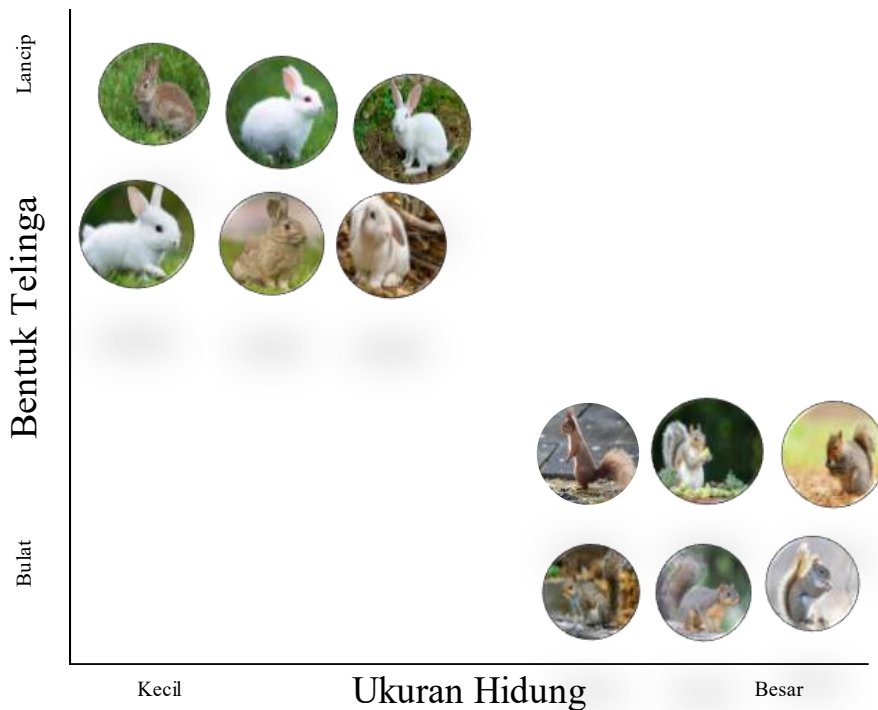
Untuk kasus Kelinci dan Tupai, kita akan menggunakan dua fitur agar proses pembelajaran dan visualisasi menjadi lebih mudah. Dua fitur itu adalah ukuran hidung dan bentuk telinga. Ukuran hidung bisa kecil atau besar, sedangkan bentuk telinga bisa

bulat atau lancip. Hasil pengukuran dari telinga dan hidung beberapa ekor kelinci serta tupai terlihat pada Tabel 1.1.

Tabel 1.1 Data sampel Kelinci dan Tupai (telinga dan hidung)

Gambar	Bentuk Telinga	Ukuran Ekor	Kelas
	Oval (4 cm)	Pendek & Bulat (6 cm)	Kelinci
	Oval (6 cm)	Pendek & Bulat (5,5 cm)	Kelinci
	Oval (5 cm)	Pendek & Bulat (5 cm)	Kelinci
	Oval (4 cm)	Pendek & Bulat (7 cm)	Kelinci
	Oval (7 cm)	Pendek & Bulat (5 cm)	Kelinci
	Oval (4,3 cm)	Pendek & Bulat (5,6 cm)	Kelinci
	Bulat Runcing (1,7 cm)	Panjang (17 cm)	Tupai
	Bulat Runcing (1,8 cm)	Panjang (19 cm)	Tupai
	Bulat Runcing (1,4 cm)	Panjang (18 cm)	Tupai
	Bulat Runcing (2,2 cm)	Panjang (19 cm)	Tupai
	Bulat Runcing (2,5 cm)	Panjang (21 cm)	Tupai
	Bulat Runcing (2,3. cm)	Panjang (20 cm)	Tupai

Berdasarkan data dari Tabel 1.1 dilakukan pemetaan kedalam ruang dua dimensi (*2-dimensional feature space*) agar mempermudah visualisasi fitur- fitur maka fitur ukuran hidung sebagai sumbu horizontal dan bentuk telinga sebagai sumbu vertikal seperti yang terlihat pada Gambar 1.8.



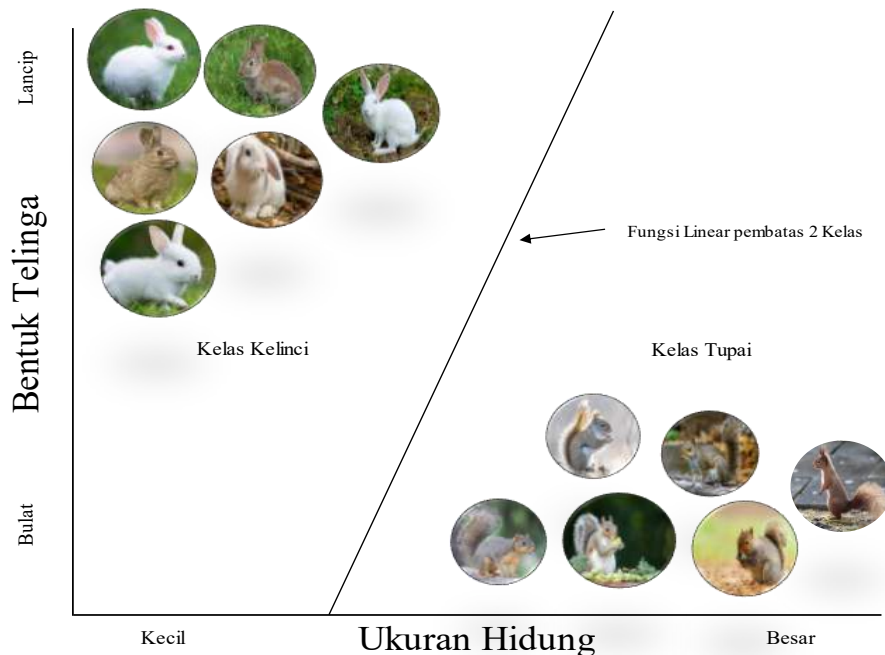
Gambar 1.8 Pemetaan data training pada feature space 2 dimensi.

3. Pelatihan Model.

Pada Gambar 1.8 dapat dilihat bahwa setiap data sudah ditempatkan dalam ruang dua dimensi, langkah berikutnya adalah melatih model berdasarkan data contoh tersebut. Secara visual dapat dilihat bahwa data Kelinci dan Tupai terkumpul di posisi tertentu karena hasil pemetaan dari fitur-fitur tersebut. Pelatihan bertujuan untuk mencari model linier yang bisa memisahkan kedua kelompok tersebut. Model linear dapat dibayangkan sebagai sebuah garis yang memisahkan antara Tupai dan Kelinci pada Gambar 1.8.

Gambar 1.9 menunjukkan model yang terbentuk dalam bentuk sebuah garis yang memisahkan anggota-anggota kelas Kelinci dan kelas Tupai. Setelah garis tersebut ditentukan, semua fitur yang berada di area berwarna merah akan dikategorikan ke dalam kelas Kelinci, sedangkan fitur yang berada di bawah garis (area berwarna biru) akan dikategorikan ke dalam kelas Tupai. Secara singkat, proses pelatihan adalah cara mencari fungsi pembatas, dalam kasus ini berupa garis pembatas yang memisahkan

anggota Kelinci dan Tupai. Dengan menggunakan pembatas tersebut, kita berhasil membedakan antara anggota Kelinci dan anggota Tupai berdasarkan posisi mereka ketika seekor makhluk (yang belum diketahui apakah termasuk Kelinci atau Tupai) ditempatkan dalam diagram atau model.



Gambar 1.9 Ilustrasi hasil pelatihan model menggunakan fungsi linear

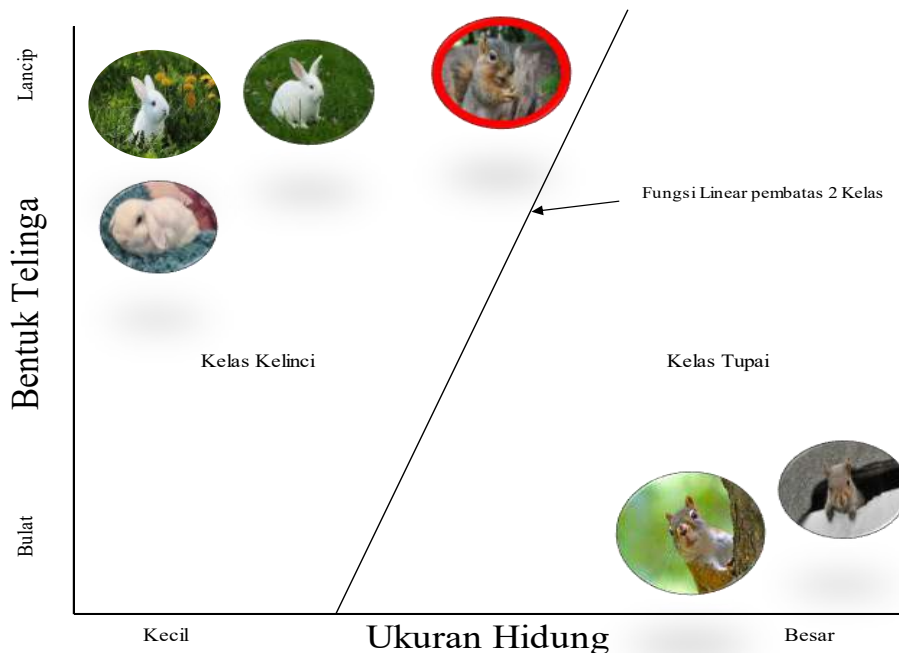
4. Uji coba model.

Proses ini bertujuan untuk menguji apakah model yang telah kita hasilkan memberikan hasil yang akurat. Untuk menguji diperlukan sejumlah data yang biasa disebut data uji. Data uji tersebut akan digunakan oleh model untuk memprediksi label. Akurasi atau performa akan dihitung dengan membandingkan label sebenar dan label hasil prediksi.



Gambar 1.10 Sekumpulan data Kelinci (kotak kanan) dan data Tupai (kotak kiri) yang akan digunakan sebagai data uji

Pada Gambar 1.10 berisi sejumlah data Kelinci dan Tupai baru yang belum pernah diidentifikasi oleh sistem. Langkah pertama yang harus dilakukan merupakan melakukan penghitungan terhadap fitur (mengukur ukuran hidung dan bentuk telinga), lalu memasukkan ke dalam model. Setelah dipetakan menggunakan model, kita dapat mengetahui masing-masing data berada di area merah atau biru (perhatikan Gambar 1.11)



Gambar 1.11 Sekumpulan data Kelinci (kotak kiri) dan data Tupai (kotak kanan) yang akan digunakan sebagai data uji

Tupai dengan ID T02 (lihat Tabel 1.2 Data uji fitur Kelinci dan Tupai (bentuk telinga dan ukuran ekor)) dikategorikan sebagai Kelinci karena telinganya tajam dan ekornya berukuran sedang. Ini adalah salah satu kesalahan dalam proses

pembelajaran mesin. Error ini bisa terjadi karena pemilihan fitur yang salah atau data latihan yang tidak lengkap, atau metode pelatihan yang digunakan. Dalam data latih, semua Tupai memiliki telinga berbentuk lonjong dan ekor yang kecil. Ketika ada tupai yang memiliki ekor pendek dan tajam, maka model yang sudah dibuat salah dalam mengklasifikasikannya. Agar mendapatkan model yang baik, pemilihan fitur, variasi, serta jumlah data latihan juga mempengaruhi hasil kerja model.

Untuk meningkatkan hasil algoritma, kita perlu mengumpulkan lebih banyak data serta fitur lainnya yang bisa membedakan, seperti jenis ekor, warna mata, atau tinggi leher. Setelah data berhasil dikumpulkan, kita perlu melatih model lagi agar dapat menemukan batas baru yang tepat.

Tabel 1.2 Data uji fitur Kelinci dan Tupai (bentuk telinga dan ukuran hidung)

Gambar	ID	Bentuk Telinga	Ukuran Ekor	Kelas Sebenarnya	Kelas Model
	T01	Bulat	Pendek	Tupai	Kelinci
	T02	Bulat	Kecil	Tupai	Tupai
	T03	Bulat	Kecil	Tupai	Tupai
	T04	Lancip	Sedang	Kelinci	Kelinci
	T05	Lancip	Sedang	Kelinci	Kelinci
	T06	Bulat Lonjong	Besar	Kelinci	Kelinci

1.5 Klasifikasi *Machine Learning*

Machine learning dapat dikelompokkan berdasarkan bagaimana cara belajar sehingga dapat melakukan tugasnya. Pembagian Machine learning berdasarkan cara belajarnya dibagi menjadi tiga kelompok yaitu :

1. **Supervised Learning**

Supervised Learning merupakan pembelajaran terarah atau terawasi. Dalam proses ini, mesin (komputer) mempelajari data training yang sudah memiliki label. Jika dianalogikan dengan dunia pendidikan, komputer berperan sebagai siswa yang belajar dari kumpulan soal yang sudah dilengkapi dengan kunci jawaban. Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer-Verlag.

Sebagai contoh, pada kasus pendeteksian kelinci dan tupai, memberikan data ke dalam sistem sama artinya dengan memberikan referensi visual yang jelas mengenai ciri-ciri keduanya. Komputer kemudian memproses informasi tersebut menggunakan algoritma tertentu untuk membangun sebuah model prediksi. Hasilnya, komputer

mampu mengenali dan mengklasifikasikan gambar kelinci atau tupai baru yang belum pernah dilihat sebelumnya. Intinya, sistem diberikan contoh kasus beserta solusi yang diinginkan sebagai panduan utama.

Selain klasifikasi, regresi juga termasuk dalam kategori pembelajaran terarah ini. Beberapa algoritma yang populer digunakan dalam supervised learning antara lain:

- Linear Regression
- Logistic Regression
- Linear Discriminant Analysis
- k-Nearest Neighbors (k-NN)
- Support Vector Machines (SVM)
- Random Forests
- Neural Networks

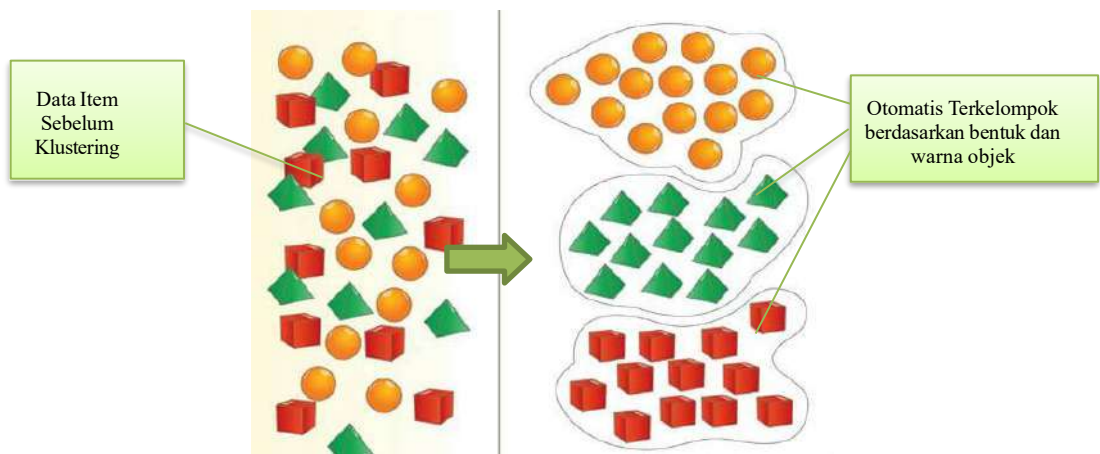
Poin-Poin Perbaikan yang Dilakukan:

- Konsistensi Subjek: Saya menyelaraskan analogi agar tetap fokus pada satu istilah (misalnya menggunakan "komputer dan siswa") agar tidak membingungkan pembaca dengan campuran istilah "dosen" dan "mahasiswa" di kalimat yang sama.
- Menghilangkan Redundansi: Kalimat seperti "data training yang berisi contoh data Tupai dan Kelinci" diulang beberapa kali pada teks asli. Saya menyederhanakannya agar lebih to-the-point.
- Ketepatan Istilah: Mengubah "mengelompokkan" menjadi "mengklasifikasikan". Dalam machine learning, mengelompokkan data tanpa label disebut clustering (unsupervised), sedangkan supervised lebih tepat disebut klasifikasi.
- Struktur Daftar: Menggunakan bullet points untuk daftar algoritma agar lebih mudah dibaca (scannable).

2. Unsupervised Learning

Tidak seperti pembelajaran terawasi, pembelajaran tidak terawasi adalah kebalikannya di mana proses pembelajaran dilakukan tanpa ada arahan atau petunjuk. Algoritma di dalam komputer bekerja untuk mencari pola-pola yang ada di dalam data. Dalam matematika, pembelajaran tanpa pengawasan terjadi ketika kita memiliki beberapa data masukan (X) dan tidak ada variabel output yang terkait dengan itu. Kohonen, T.(1990). The self-organizing map. Proceedings of the IEEE, volume 78, number 9, pages 1464 hingga 1480.[1, 2, 3, 4, 5]. Jika kita menggunakan analogi yang sama seperti contoh tadi, maka mahasiswa belajar tanpa ada solusi atau jawaban yang benar, dan siswa harus mencari jawabannya sendiri. Masalah pembelajaran tanpa pengawasan bisa dibagi menjadi dua macam, yaitu asosiasi dan clustering.

Misalnya, pada Gambar 1.12 terdapat dua jenis item yang berbeda. Kedua barang tersebut akan dibagi ke dalam beberapa kelompok berdasarkan datanya. Komputer hanya tahu fitur apa saja yang digunakan untuk membedakan dua barang tersebut, yaitu warna dan bentuk. Dengan menggunakan algoritma clustering, komputer bisa membagi berbagai item ke dalam dua kelompok tanpa perlu diberi informasi. Algoritma akan berjalan untuk membagi item-item tersebut ke dalam beberapa kelompok berdasarkan isi data masing-masing item. Teknik-teknik pengelompokan tersebut akan dibahas pada bab-bab berikutnya. Contoh algoritma unsupervised learning sederhana merupakan K-means.



Gambar 1.1 Ilustrasi proses klusterisasi data

3. Reinforcement Learning

Sebuah komputer akan bekerja dalam lingkungan yang sangat dinamis dan diminta untuk menyelesaikan tugas tertentu, seperti bermain catur atau mobil listrik sendiri. Mesin akan menggunakan algoritma untuk mempelajari bagaimana membuat keputusan khusus dalam lingkungan yang berubah-ubah.. Sutton, R. S., & Barto, A. G. (n.d.). Reinforcement Learning: An Introduction Second edition, in progress.

Selain dilihat dari cara pembelajaran, machine learning juga bisa dibagi berdasarkan bagaimana proses pembelajarannya dilakukan. Pembelajaran bisa dilakukan bertahap, yaitu dengan memproses data dalam batch, atau dilakukan langsung secara berturut-turut saat data masuk, yang disebut online learning. Prinsip dasar dari pembelajaran online adalah membuat sebuah model yang mampu belajar dari data baru secara real time atau hampir real time. Sementara itu, dalam Batch Learning, data latih dibagi menjadi beberapa bagian, lalu setiap bagian tersebut dipelajari secara terpisah pada waktu yang berbeda.

Machine learning dapat dikelompokkan menjadi dua jenis berdasarkan cara kerjanya.

- Pembelajaran berbasis instance (atau sering disebut pembelajaran berbasis memori) adalah sebuah kelompok algoritma pembelajaran mesin yang bekerja dengan membandingkan data uji dengan data yang sudah dipelajari dalam proses pelatihan. Algoritma kelompok ini tidak membuat kesimpulan umum, tetapi lebih fokus pada perbandingan dengan data yang tersimpan di memori. Contoh algoritma yang termasuk dalam kelompok ini adalah k-nearest neighbors, kernel machines, dan RBF network.
- Pembelajaran berbasis model merupakan kebalikan dari pembelajaran berbasis instance, di mana menggunakan memori untuk menyelesaikan masalah. Algoritma ini membuat sebuah model yang bersifat umum.

Kriteria-kriteria ini tidak eksklusif, kita dapat mengombinasikannya. Sebagai contoh, sebuah sistem anti-spam yang mampu belajar dari spam secara langsung menggunakan jaringan saraf mendalam yang dilatih dengan data spam dan ham; maka sistem ini disebut sebagai sistem online, model-based dan supervised learning system

1.6 Algoritma *Machine Learning*

Agar penggunaan lebih mudah, algoritma ML harus dibagi sesuai dengan masalah yang ingin diselesaikan.. Adapun pembagian tersebut

Algoritma Regresi

Permasalahan regresi berfokus pada pembuatan model yang menghubungkan variabel independen dengan variabel dependen. Algoritma ini biasanya digunakan untuk melakukan peramalan atau pengelompokan data. Penjelasan yang lebih lengkap mengenai materi ini dapat kamu temukan di buku yang berbeda. Algoritma yang sering digunakan dalam kelompok ini antara lain Regresi Ordinary Least Squares (OLSR), Regresi Linear, Regresi Logistik, Regresi Stepwise, Multivariate Adaptive Regression Splines (MARS), serta Locally Estimated Scatterplot Smoothing (LOESS).

Algoritma Regularisasi

Algoritma regularisasi adalah pengembangan dari metode lainnya (biasanya metode regresi). Manfaat dari algoritma ini adalah mencegah overfitting dengan membuat model yang lebih sederhana, sehingga dapat menggeneralisasi hasil dengan lebih baik. Beberapa algoritma regularisasi yang memiliki performa yang baik adalah Least Absolute Shrinkage and Selection Operator (LASSO) atau Regularisasi L1, Ridge Regression atau Regularisasi L2, Elastic Net, dan Least-Angle Regression (LARS).

Algoritma Instance-based Learning

Model pembelajaran berbasis instance secara sederhana adalah sebuah model yang membuat keputusan dengan memperhatikan contoh atau data latihan yang sudah ada. Algoritma ini membandingkan contoh dengan masalah, sehingga hasil yang diperoleh adalah yang paling serupa dengan contoh tersebut. Dalam metode ini, biasanya dibuat sebuah database contoh, lalu data baru dibandingkan dengan database tersebut menggunakan ukuran kesamaan untuk mencari yang paling cocok, kemudian membuat prediksi berdasarkan hasilnya. Algoritma-algoritma yang sering digunakan antara lain K-Nearest Neighbor (KNN), Learning Vector Quantization (LVQ), Self-Organizing Map (SOM), Locally Weighted Learning (LWL), dan masih banyak lagi.

Algoritma Decision Tree

Algoritma Decision Tree atau pohon keputusan adalah cara-cara yang digunakan untuk membuat model pengambilan keputusan dalam bentuk pohon atau tree, berdasarkan data latihan yang dimiliki. Berikut beberapa algoritma yang sering digunakan yaitu Classification and Regression Tree (CART), Iterative Dichotomiser 3 (ID3), C4.5, dan C5.0.

Algoritma Bayesian

Algoritma Bayesian adalah metode yang secara jelas menggunakan Teorema Bayes untuk menyelesaikan permasalahan klasifikasi dan regresi. Berikut beberapa metode yang termasuk dalam kategori ini yaitu Naive Bayes, Gaussian Naive Bayes, Multinomial Naive Bayes, dan Bayesian Belief Network (BBN).

Algoritma Klustering

Clustering atau klasterisasi merupakan metode pengelompokan data. Menurut Tan, 2006, clustering adalah proses mengelompokkan data menjadi beberapa kelompok atau kluster sehingga data dalam satu kelompok memiliki kesamaan yang sangat tinggi, sedangkan data di antara kelompok memiliki kesamaan yang sangat rendah. Berikut beberapa algoritma klasterisasi yang sering digunakan seperti k-Means, k-Medians, Expectation Maximisation (EM), Agglomerative Nesting (AGNES), dan Divisive Analysis (DIANA).

Algoritma Association Rule Learning

Association rule adalah metode yang digunakan untuk mengekstrak aturan sehingga diperoleh hubungan antar variabel dalam data. Algoritma yang sering digunakan adalah Apriori dan Eclat.

Algoritma Artificial Neural Network

Jaringan Syaraf Tiruan (Artificial Neural Networks) adalah cara membuat model yang menginspirasi struktur sistem saraf manusia. Beberapa algoritma yang sering digunakan antara lain Perceptron, Back-Propagation, Hopfield Network, dan Radial Basis Function Network (RBFN).

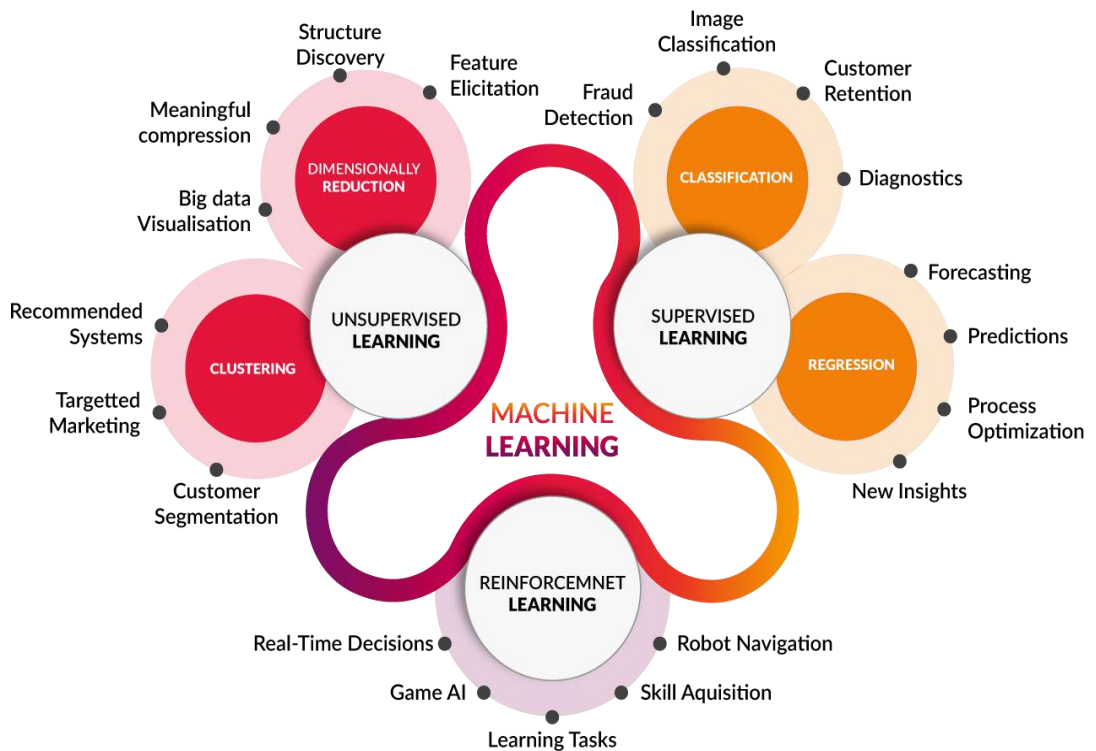
Algoritma Deep Learning

Deep Learning adalah pengembangan lebih lanjut dari Neural Network dengan struktur jaringan yang lebih rumit. Berikut beberapa algoritma yang sering digunakan dalam kategori ini. Jaringan Saraf Tiruan Convolutional (CNN), Jaringan Boltzmann Profund (DBM), Jaringan Percaya Profund (DBN), Encoder Berlapis, Jaringan Saraf Tiruan Berulang (RNN), dan lainnya.

Algoritma Ensemble

Metode Ensemble atau metode ansambel adalah sebuah algoritma dalam pembelajaran mesin yang berusaha mencari solusi prediksi terbaik dibandingkan dengan algoritma lainnya. Metode ini menggunakan beberapa algoritma pembelajaran untuk mencapai hasil prediksi yang lebih baik. Berikut beberapa algoritma ensambel yang sering digunakan, seperti Boosting, Bootstrapped Aggregation atau Bagging, AdaBoost, Gradient Boosting Machines atau GBM, Gradient Boosted Regression Trees atau GBRT, dan Random Forest.

1.7 Aplikasi Machine Learning

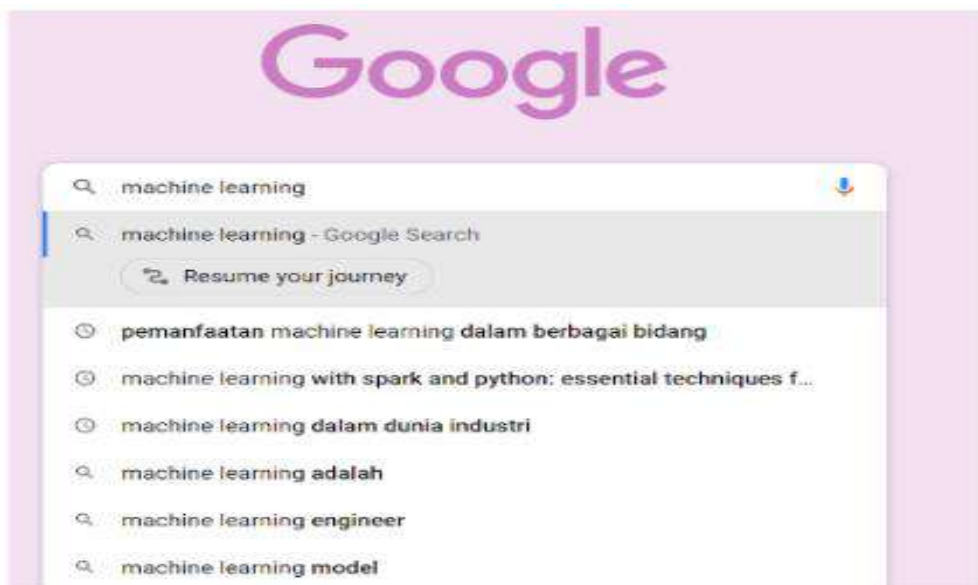


Gambar 1.12 Masalah yang dapat diselesaikan dengan ML (sumber: Medium.com)

Pemanfaatan aplikasi Machine Learning sudah terasa dalam aktivitas sehari-hari. Aktivitas-aktivitas seperti sistem rekomendasi, prediksi, peramalan, diagnosa, dan robot canggih sudah menggunakan teknologi pembelajaran mesin. Contoh sederhananya adalah produk-produk Google, seperti Google Translate yang bisa menerjemahkan teks, mengenali tulisan tangan, serta suara, Alpha Go, Google Search, dan Gmail.

Mesin Pencari

Semua mesin pencari seperti Google, Bing, dan Yandex menggunakan machine learning untuk menilai dan mengurutkan halaman sebuah website, memberi saran kata kunci, memperbaiki kesalahan pengejaan, serta menawarkan saran kata kunci yang lebih baik. Setiap mesin pencari memiliki cara sendiri dalam menggunakan algoritma untuk mencari informasi. Saat kita mengetikkan kata kunci, Google akan menunjukkan hasil pencarian yang paling sesuai dengan kata kunci tersebut. Jika kita memilih sebuah halaman dan menghabiskan waktu yang lama di halaman tersebut, Google akan mengetahui bahwa halaman itu cocok dengan kata kunci yang kita gunakan. Seperti itu pula, ketika kita melihat halaman pencarian berikutnya, misalnya halaman 2, 3, dan seterusnya. Google akan mendeteksi jika ada kata kunci yang tidak sesuai dengan hasil pencarian yang muncul. Itulah cara data tersebut dikumpulkan dan dijumlahkan dengan menggunakan Machine Learning oleh Google Search Engine, sehingga menghasilkan hasil pencarian yang selalu berubah-ubah dan berkualitas.



Gambar 1.13 Rekomendasi kata kunci pencarian Google

Toko Online dan Marketplace

Penerapan Machine Learning di marketplace atau toko online bisa memberi manfaat bagi penjual dan pembeli. Untuk membuat pengalaman belanja lebih baik dan meningkatkan penjualan, setiap akun harus menampilkan produk yang direkomendasikan sesuai dengan minat dan keinginan pembeli. Untuk menjalankan hal ini secara otomatis dan langsung, Machine Learning memainkan peran penting dalam meningkatkan keakuratan rekomendasi produk yang diberikan kepada setiap pembeli.

Google Cendekia

penelitian wanita karir

Artikel

Sekitar 24.000 hasil (0,03 detik)

Kapan saja
Sejak 2021
Sejak 2020
Sejak 2017

Rentang khusus...

Urutan menurut relevansi

Urutan menurut tanggal

☐ sertakan paten

☒ mencakup kutipan

☒ Buat lansiran

Pengaruh konflik peran terhadap kinerja **wanita karir** pada Universitas Sam Ratulangi Manado [PDF] unsrat.ac.id

Beberapa peneliti menemukan bahwa konflik pekerjaan-keluarga dan konflik keluarga-pekerjaan mempunyai pengaruh negatif yang signifikan terhadap ... Konflik peran berpengaruh negatif terhadap kinerja **wanita karir** pada Universitas Sam Ratulangi METODE PENELITIAN

☆ 99 Ditinjau 35 kali Artikel terkait 2 versi 00

Keseimbangan kerja-kehidupan pada **wanita karir** [PDF] ulm.ac.id

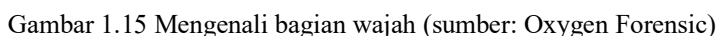
Selain itu peneliti juga menjabarkan ketekunan dengan melakukan pengamatan secara lebih ... Gambar hasil dan pembahasan penelitian ini menunjukkan pada bagaimana keseimbangan ... kerja seseorang, menurut Gilang (2014) di dalam penelitiannya menganal dukungan

☆ 99 Ditinjau 13 kali Artikel terkait 00

Pengaruh Gender dan **wanita karir**

78 Gambar 8. Model Relasi Gender yang Harmonis dan Kesuksesan **Wanita Karir** ... 122 Page

Media Sosial adalah salah satu aplikasi yang sering digunakan oleh banyak orang, karena itu penyedia layanan media sosial selalu memperkenalkan fitur yang canggih agar pengguna merasa lebih mudah, aman, dan nyaman. Misalnya media sosial Facebook menggunakan beberapa teknologi pembelajaran mesin untuk memudahkan pengguna, seperti rekomendasi teman yang mungkin kita kenal, pengenalan wajah, urutan konten yang tampil di halaman utama, dan masih banyak lagi.



Google AdWords adalah media iklan digital yang paling populer di dunia. Iklan di AdWords ditampilkan di situs web yang menjadi publisher pada Google AdSense. Iklan yang muncul adalah iklan yang terus berubah atau tidak tetap. Google Adwords

mengumpulkan informasi tentang situs web berdasarkan isinya, lalu menampilkan iklan-iklan yang sesuai dengan topik tersebut. Selain itu, Google AdWords juga memakai cookies untuk mengacu pada aktivitas pengunjung di suatu website, terkait dengan situs-situs yang telah dikunjungi sebelumnya. Adword mengumpulkan kedua faktor tersebut, sehingga bisa menampilkan iklan yang sesuai dengan pengunjung situs web.

Asisten Pribadi Virtual

Gadget sekarang memiliki asisten virtual pribadi, baik untuk laptop maupun ponsel, seperti Cortana di Windows, Siri di iPhone, dan Google Now di Android. Asisten virtual ini bisa membantu pengguna mencari informasi di internet, bertanya tentang arah, cuaca, melakukan panggilan telepon, serta membuka aplikasi. Di Windows, Cortana belajar mengenai pengguna untuk memberikan rekomendasi perintah yang akan dilakukan.

Penyaring Pesan Spam

Machine learning memiliki peran yang sangat penting dalam memfilter spam, baik di email, website, maupun media komunikasi yang menggunakan software yang sudah terinstal. Algoritma pohon keputusan atau decision tree adalah dasar dari algoritma penyaring spam, yang digunakan untuk menentukan apakah sebuah pesan termasuk spam atau bukan.

Mobil Kendali Otomatis

Mobil kendali otomatis adalah hasil penerapan dan pengembangan teknologi machine learning, khususnya bidang machine vision. Mobil kendali otomatis adalah contoh penerapan machine learning yang rumit dan memiliki risiko yang sangat besar. Mobil perlu mempelajari banyak hal, seperti tanda-tanda lalu lintas, arah dan tujuan, kondisi jalan, lampu lalu lintas, kondisi orang di sekitarnya, serta sensor lainnya yang terpasang.lalu lintas,



Gambar 1.16 Implementasi kendali otomatis mobil (sumber:dqlab.id)

BAB II

TEORI TERKAIT MACHINE LEARNING

Machine Learning (ML) adalah sebuah cabang ilmu yang didasarkan pada berbagai disiplin ilmu lainnya, seperti Aljabar Linier, Kalkulus, dan Statistika. Ilmu-ilmu dasar tersebut penting untuk memahami konsep dasar dari pembelajaran mesin. Selain memahami konsep dasar tersebut, kita juga membutuhkan kemampuan teknis seperti algoritma dan pemrograman agar dapat menerapkan ilmu-ilmu dasar tersebut dengan manfaat yang nyata. Kemampuan komputasi adalah alat yang diperlukan untuk menjalankan pembelajaran mesin, karena tanpa kemampuan komputasi, konsep-konsep itu tidak bisa diwujudkan. Jika kita bandingkan, dasar-dasar matematika seperti otak dan kemampuan komputasi seperti otot serta tubuhnya yang bergerak. Kedua hal tersebut saling melengkapi.

2.1 Kemampuan Komputasi

Kemampuan komputasi berarti kemampuan untuk menggunakan bahasa pemrograman atau alat berbasis komputer tertentu untuk memproses data. Sekarang ini ada banyak aplikasi atau perangkat lunak yang bisa dipakai untuk mengatasi masalah dalam bidang pembelajaran mesin atau machine learning. Wing, J. M.(2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35. Secara sederhana, kita hanya perlu memasukkan data, lalu aplikasi akan memproses data tersebut dan membuat model yang bisa digunakan untuk membuat prediksi. Sebagian besar aplikasi tersebut sudah menggunakan algoritma-algoritma yang umum dan bisa menyelesaikan masalah tersebut. Hutter, F., Kotthoff, L., & Vanschoren, J.(Eds.).(2019). *Automated Machine Learning: Methods, Systems, Challenges*. Springer Nature.

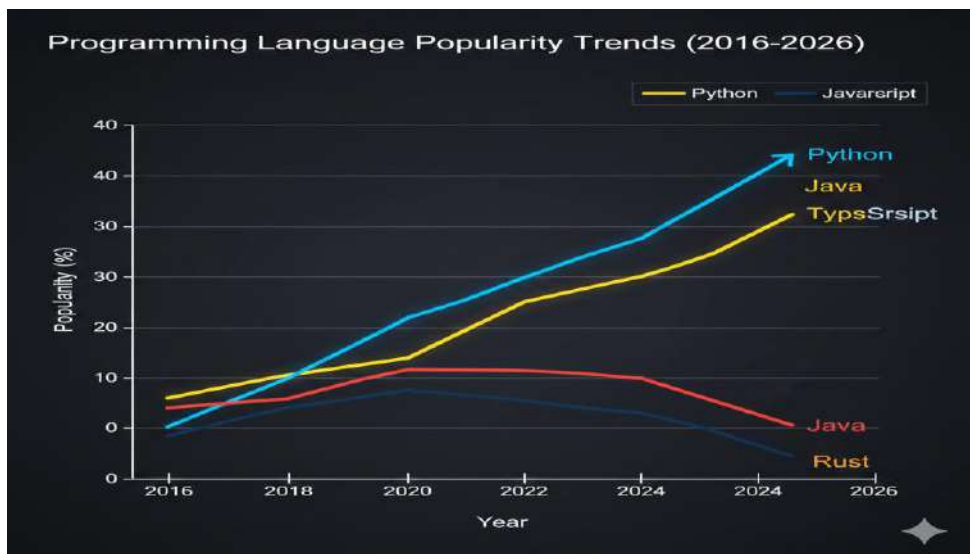
Seperti seorang tukang mekanik yang ingin memperbaiki mobil, dia harus bisa mengerti apa yang salah dengan mobil itu, lalu menggunakan alat-alatnya untuk mencari tahu masalahnya dan memperbaikinya. Setiap alat memiliki peran sendiri, dan sebuah kegiatan perbaikan terdiri dari beberapa langkah yang harus dilakukan secara berturut-turut. Terkadang dengan pengalaman yang dia miliki, dia melakukan "hacking" terhadap masalah kita sehingga masalah tersebut bisa terpecahkan.

Hal itu hampir mirip dengan cara yang digunakan oleh seorang programmer saat menyelesaikan sebuah masalah. Beberapa langkah harus dilakukan agar kasus tersebut bisa terungkap. Proses analisa, perancangan, dan implementasi adalah urutan yang tidak bisa dihindari. Oleh karena itu, ketika kita belajar tentang pembelajaran mesin, kita harus memiliki kemampuan di bidang pemrograman agar dapat menerapkan apa yang sudah kita analisis dan rancang.

Python

Python adalah bahasa pemrograman yang sangat populer di kalangan pengembang Machine Learning, ilmuwan data, serta ahli data mining. Alasannya karena bahasa ini mudah dan gratis. Selain itu, Python juga merupakan bahasa pemrograman yang interpretatif dan memiliki banyak kegunaan. Tidak seperti bahasa pemrograman lain yang sulit dibaca dan dipahami, Python lebih fokus pada kejelasan kode agar sintaksnya lebih mudah dipahami. Hal ini membuat Python cukup mudah dipelajari, baik bagi yang baru mulai belajar maupun bagi yang sudah mengenal bahasa pemrograman lainnya.

Bahasa ini pertama kali diperkenalkan tahun 1991, dan dibuat oleh seseorang bernama Guido van Rossum. Hingga saat ini, Python masih dikembangkan oleh Python Software Foundation. Van Rossum, G.(2009). A Brief History of Python. Artima Developer. Van Rossum, G., & Drake, F. L.(2009). Python 3 Reference Manual. Bahasa Python bisa digunakan di hampir semua jenis sistem operasi, termasuk Linux, di mana hampir semua versi distro sudah memiliki Python sebagai bagian dari sistemnya. CreateSpace, Python Software Foundation.(2024). PSF Mission Statement.python.org. Menurut Google Trend, hingga tahun 2021, penggunaan Python mengalami pertumbuhan yang cukup besar dan saat ini menjadi bahasa pemrograman yang paling sering dibicarakan.



Gambar 2.1 Perkembangan Python (sumber: google trend,2021)

Table 1.1 Perbandingan penggunaan bahasa pemrograman

Bahasa Pemrograman	Tren (2016 vs 2026)	Status Saat Ini	Alasan Utama Perubahan
Python	 Pesat	#1	Dominasi di bidang AI, Machine Learning, dan Data Science.
JavaScript	 Stabil	#1 (Usage)	Standar wajib untuk pengembangan web (Front-end & Back-end).
Java	 Menurun	Top 5	Masih kuat di enterprise, namun mulai tergeser oleh Kotlin dan Go.
Type Script	 Meningkat	Top 10	Menjadi standar baru bagi developer JS yang menginginkan keamanan tipe data.
C++ / C	 Stabil	Top 5	Tak tergantikan untuk sistem tingkat rendah, game engine, dan IoT.
Rust	 Tumbuh	Emerging	Dipilih sebagai bahasa paling dicintai karena keamanan memori yang tinggi.

Analisis Perubahan pada Dekade Terakhir

Masa Kejayaan Python (2018–2024): Sekitar tahun 2017-2018, Python mulai menyalip Java di berbagai indeks. Kemudahan sintaksisnya membuat Python menjadi gerbang utama bagi non-programmer untuk masuk ke dunia teknologi melalui analisis data (Cass, (2017) Modernisasi JavaScript melalui TypeScript: JavaScript tidak lagi berdiri sendiri. Penggunaan TypeScript melonjak karena perusahaan besar membutuhkan kode yang lebih terstruktur dan mudah dipelihara (scalable).

1. Penurunan Java dan PHP: Meski masih digunakan secara masif di sistem perbankan dan WordPress, kedua bahasa ini mengalami penurunan minat di kalangan pengembang baru yang lebih memilih bahasa yang lebih ringkas seperti Go atau Node.js.
2. Kebangkitan Bahasa Sistem Baru: Rust dan Go (Golang) naik daun karena menawarkan performa setara C++ namun dengan manajemen memori dan konkurensi yang jauh lebih modern dan aman.

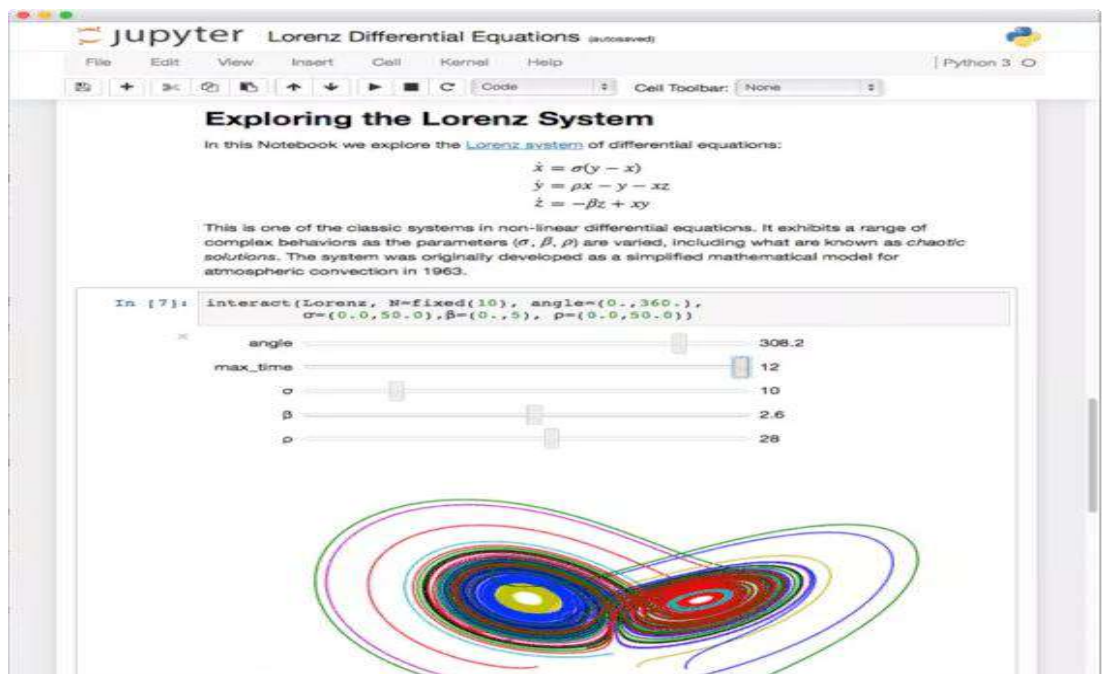
Buku ini menggunakan Python sebagai bahasa pemrograman karena kemudahannya dalam penerapan. Berbeda dengan bahasa pemrograman yang sering digunakan di bidang sains seperti R dan MATLAB, Python lebih banyak dipilih karena termasuk dalam bahasa pemrograman yang umum, sehingga para pengembang bisa membuat aplikasi untuk perusahaan dengan lebih mudah.

Berikut beberapa kelebihan Python:

- Mudah dipelajari dan singkat. Bahasa pemrograman Python memiliki jumlah kata kunci yang sedikit dan struktur yang mudah, sehingga cocok untuk dipelajari oleh orang yang baru mengenal pemrograman. Selain itu, kode program yang menggunakan Python biasanya lebih pendek dibandingkan bahasa pemrograman lainnya untuk menyelesaikan masalah yang sama.
- Mudah dibaca. Python menggunakan kata kunci yang mudah dipahami dan memaksa penggunaan spasi atau tab untuk menyusun kode (mengatur letak kode agar lebih rapi dengan cara mendorong ke dalam sesuai dengan bagian kode tertentu), sehingga membuat kode yang ditulis terlihat lebih bersih dan terstruktur. Dalam bahasa pemrograman lain, indentasi ini tidak wajib.
- Cross-platform. Python bisa digunakan di berbagai sistem operasi, seperti Windows, UNIX, Linux, Mac OS, dan lainnya. Program yang ditulis sekali tidak perlu diubah lagi agar bisa dijalankan di berbagai sistem operasi yang berbeda.
- Mendukung multi-paradigma. Python dibuat agar bisa digunakan untuk berbagai keperluan, sehingga bisa ditulis dengan cara prosedural atau menggunakan pendekatan pemrograman berbasis objek. Programmer yang belum terbiasa dengan konsep OOP bisa menggunakan Python dengan cara prosedural.
- Memiliki banyak library dan package Manager. Library adalah istilah untuk kode program tambahan yang dibuat oleh orang lain untuk menyelesaikan tugas-tugas tertentu. Python memiliki lebih dari 170.000 buah library atau packages yang dibuat oleh para pengguna di situs pypi.org. Selain itu, agar pengguna lebih mudah, Python memiliki sistem paket yang bernama pip. Dengan adanya package manager, pengguna bisa menginstal dan memperbarui library dengan mudah tanpa harus bingung dengan dependensi.
- Gratis dan opensource. Python dikembangkan sebagai proyek sumber terbuka dan bisa digunakan oleh siapa saja secara gratis.
- Komunitas yang kuat. Jika menemukan masalah, pembaca bisa pergi ke [stackoverflow](https://stackoverflow.com) dan mendapatkan solusi dengan cepat.

Jupyter Notebook

Jupyter Notebook, yang sebelumnya dikenal sebagai IPython Notebook, adalah sebuah aplikasi yang sering digunakan dalam bidang Data Science. Aplikasi ini biasanya dipakai untuk membuat dan membagikan dokumen yang menyertakan kode hidup, persamaan, visualisasi, serta penjelasan teks mengenai dokumen tersebut. Jupyter Notebook adalah sebuah aplikasi berbasis server dan client yang memungkinkan kita untuk mengedit dan menjalankan atau mengompilasi dokumen notebook melalui peramban web.



Gambar 2.2 Aplikasi Jupyter Notebook

Dengan alat ini kita bisa melakukan berbagai tugas seperti membersihkan data, menganalisis data, membangun model, hingga melatih dan mengevaluasi model. Jupyter digunakan karena bisa dijalankan per baris, sehingga kita bisa langsung melihat hasil yang kita inginkan. Jika dibandingkan dengan cara tradisional di mana kita harus mengetik semua kode terlebih dahulu baru bisa menjalankannya, cara ini lebih mudah dan sederhana, terutama bagi orang yang sedang belajar Machine Learning.

NumPy

NumPy, yang merupakan singkatan dari Numerical Python, adalah sebuah library Python yang dirancang khusus untuk menangani perhitungan ilmiah. NumPy bisa membuat objek array berdimensi banyak, yang hampir mirip dengan list di Python. Keunggulan menggunakan array NumPy dibandingkan dengan list di Python adalah penggunaan memori yang lebih sedikit dan eksekusi kode yang lebih cepat. Numpy memudahkan dalam melakukan perhitungan pada larik berdimensi banyak dan operasi matriks. Selain digunakan untuk menyelesaikan persamaan aljabar linier dan berbagai perhitungan matematika lainnya, NumPy juga berfungsi sebagai wadah multi-dimensi yang fleksibel untuk menyimpan berbagai jenis data umum.

Salah satu alasan mengapa Numpy sering digunakan adalah karena list di Python tidak mendukung secara penuh kemudahan dalam komputasi ilmiah, contohnya kita akan melakukan operasi penjumlahan pada dua list seperti berikut ini:

```
[1] 1  #Penjumlahan dua List Python
    2  a = [1,2,3]
    3  b = [4,5,6]
    4
    5  print('a+b = ', a+b)
    6
```

```
☞ a+b = [1, 2, 3, 4, 5, 6]
```

Ketika kita ingin menjumlahkan setiap elemen pada list a ([1,2,3]) dan list b ([4,5,6]) menggunakan operator +, hasilnya adalah penggabungan keduanya menjadi [1,2,3,4,5,6]. Padahal dalam penghitungan ilmiah, tanda + digunakan untuk menjumlahkan kedua daftar tersebut. Karenanya, kita perlu menggunakan perulangan agar setiap elemen dari list a dan list b bisa ditambahkan. Proses penjumlahan elemen dalam list dengan menggunakan perulangan for memakan waktu yang cukup lama dan kurang efisien dalam hal penulisan kode.

Lebih hebatnya lagi, NumPy bisa bekerja bersama bahasa pemrograman lain seperti C, C++, dan Fortran. Fleksibilitas perpustakaan NumPy memungkinkan pengguna untuk dengan cepat dan mudah menggabungkan berbagai database dan alat lainnya. Misalnya, kita bisa lihat bagaimana NumPy, yang biasanya disingkat sebagai np, digunakan untuk mengalikan dua matriks.

Untuk bisa menggunakan Numpy, kamu harus mengimpor library tersebut terlebih dahulu. Berikut contoh melakukan import terhadap library NumPy

```
[3] 1  #inisialisasi penggunaan Numpy
    2  import numpy as np
```

Setelah melakukan import, maka kita dapat menggunakan library tersebut. Sebagai contoh pertama kita akan membuat sebuah array (vektor) [1, 2, 3] dan matrix [[6,5],[11,7],[4,8]].

```
[4] 1  #membuat sebuah array
    2  arr_1_dimensi = np.array([1, 2, 3])
    3  arr_2_dimensi = np.array([[6, 5], [11, 7], [4, 8]])
    4
    5  print("Ukuran Array 1 Dimensi : ", arr_1_dimensi.shape)
    6  print("Ukuran Array 2 Dimensi : ", arr_2_dimensi.shape)
    7
```

```
↳ Ukuran Array 1 Dimensi : (3,)
   Ukuran Array 2 Dimensi : (3, 2)
```

Pada input baris ke dua, kita membuat sebuah vektor menggunakan method `np.array`, yang membutuhkan sebuah parameter berupa list. List tersebut akan dikonversi menjadi array numpy lalu kita akan menggunakan atribut `shape` pada baris ke lima untuk melihat dimensi dari vektor atau array 1 dimensi yang telah kita buat. Sedangkan pada paris kedua kita membuat sebuah matrik 3 x 2 (3 baris dan 2 Kolom) menggunakan perintah yang sama tetapi dengan list yang berbeda sehingga kita dapat membuat sebuah array dua dimensi seperti berikut ini.

```

        6      5
array_2_dimensi = [11      7]
                   4      8
```

Pada proses membuat array, terjadi upcasting ketika tipe data setiap elemennya berbeda, sehingga dilakukan penyesuaian tipe data ke tipe yang lebih tinggi. Misalnya, kita membuat array numerik di mana semua elemen berupa bilangan bulat, kecuali satu elemen yang berupa bilangan desimal (misalnya: `[1, 2, 3.2]`), maka secara otomatis semua elemen dalam array akan diubah menjadi tipe float, sehingga menjadi `[1., 2., 3.2]`.

Selain menggunakan `np.array`, kita juga bisa membuat array dengan cara lain seperti `zeros`, `ones`, `arange`, dan masih banyak lagi. Metode `zeros` dan `ones` memerlukan parameter yang sama, yaitu sebuah bilangan bulat atau pasangan bilangan bulat yang menunjukkan ukuran array yang akan dibuat, misalnya `(2, 3)` atau `2`. Metode `zeros` akan membuat array yang semua nilainya 0, sedangkan metode `ones` akan membuat array yang semua nilainya 1.

```
↳ [[0. 0. 0.]
    [0. 0. 0.]]
[[1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]]
```

```
[6] 1 c = np.arange(0, 10)
    2 d = np.arange(0, 10, 2)
    3
    4 print(c)
    5 print(d)
```

36

```
[10] 1  numpy_matrix = np.array([(1, 2, 3, 4), (5, 6, 7, 8)])
      2  print(numpy_matrix.shape)
```

↳ (2, 4)

- Mengubah dimensi array. Perubahan dimensi dapat dilakukan dengan menggunakan fungsi reshape. Perhatikan contoh input ke-14, dimana mengubah array 2x4 menjadi array 4x2.

```
[11] 1  numpy_matrix = np.array([(1, 2, 3, 4), (5, 6, 7, 8)])
      2  numpy_matrix = numpy_matrix.reshape(4, 2)
      3  print(numpy_matrix)
```

↳

```
[[1 2]
 [3 4]
 [5 6]
 [7 8]]
```

- Mengambil sebuah nilai pada posisi tertentu. Untuk mengambil nilai pada lokasi tertentu cukup dengan memberikan alamat index baris dan kolom, dimana index dimulai dari nol. Contoh array [0,2] akan mengambil pada baris ke 1 kolom ke 3

```
[12] 1  numpy_matrix = np.array([(1, 2, 3, 4), (5, 6, 7, 8)])
      2  print(numpy_matrix[0, 2])
```

↳ 3

- Mengambil beberapa nilai dari array: Pengambilan beberapa nilai mirip dengan pengambilan sebuah nilai, bedanya kolom atau baris dalam bentuk index. Contohnya [0,0:4] akan menghasilkan nilai pada baris pertama dan kolom dengan index 0 sampai 3

```
[14] 1  numpy_matrix = np.array([(1, 2, 3, 4), (5, 6, 7, 8)])
      2  print(numpy_matrix[0, 0:3])
```

↳

```
[1 2 3]
```

- Membalikkan Array dapat dilakukan dengan menggunakan “ ::-1 ” pada posisi baris akan membalik urutan, jika berada dikolom akan membalikkan kolom.

```
[16] 1  numpy_matrix = np.array([(1, 2, 3, 4), (5, 6, 7, 8)])
      2  print(numpy_matrix[ 0, ::-1])
```

```
↳ [4 3 2 1]
```

- Mencari nilai maximum, minimum dan total dari array dapat dilakukan dengan mudah. Dari sebuah objek numpy telah disediakan fungsi max untuk menghitung nilai tertinggi, min untuk menghitung nilai terendah dan sum menghitung penjumlahan semua item pada array.

```
[20] 1  numpy_array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
      2  print("MAX :", numpy_array.max())
      3  print("MIN :", numpy_array.min())
      4  print("SUM :", numpy_array.sum())
```

```
↳ MAX : 9
   MIN : 1
   SUM : 45
```

- Mencari rata, median, varian dan standar deviasi dari array dapat dilakukan dengan memanggil fungsi mean, median, var dan std pada objek array Numpy.

```
[22] 1  numpy_array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
      2  print("MEAN :", numpy_array.mean())
      3  print("MEDIAN :", np.median(numpy_array))
      4  print("VAR   :", numpy_array.var())
      5  print("STD   :", numpy_array.std())
```

```
↳ MEAN : 5.0
   MEDIAN : 5.0
   VAR   : 6.666666666666667
   STD   : 2.581988897471611
```

- Melakukan operasi pada level kolom atau baris. Penggunaan parameter “axis = 0 ” akan melakukan proses hanya pada sumbu y sedangkan “ axis = 1 ” akan melakukan operasi pada sumbu x. “ axis = None” akan melakukan operasi pada sumbu x dan y

```
[25] 1 numpy_matrix = np.array([(1, 2, 3, 4), (5, 6, 7, 8)])
      2 print(numpy_matrix)
      3 print("SUM[axis=0] : ", numpy_matrix.sum(axis = 0))
      4 print("SUM[axis=1] : ", numpy_matrix.sum(axis = 1))
      5
```

```
↳ [[1 2 3 4]
     [5 6 7 8]]
SUM[axis=0] : [ 6  8 10 12]
SUM[axis=1] : [10 26]
```

- Menggabungkan beberapa array. Fungsi concatenate dapat digunakan untuk menggabungkan beberapa array menjadi sebuah array. Sebagai contoh array [1,2,3], [4,5,6], [7,8,9] ketika digabungkan akan menghasilkan [1 2 3 4 5 6 7 8 9]. Sedangkan untuk array yang multi-dimensi [[1,2,3],[0,0,0]] dan [[0,0,0],[7,8,9]] akan menghasilkan [[1 2 3 0 0 0], [0 0 0 7 8 9]]

```
[26] 1 array_1 = np.array([1,2,3])
      2 array_2 = np.array([4,5,6])
      3 array_3 = np.array([7,8,9])
      4 print("1D :",np.concatenate((array_1, array_2, array_3)))
      5 array_4 = np.array([[1,2,3],[0,0,0]])
      6 array_5 = np.array([[0,0,0],[7,8,9]])
      7 print("n-D :",np.concatenate((array_4, array_5), axis = 1))
```

```
↳ 1D : [1 2 3 4 5 6 7 8 9]
    n-D : [[1 2 3 0 0 0]
           [0 0 0 7 8 9]]
```

Penggunaan Numpy untuk pengolahan matriks dan tensor akan dibahas lebih lanjut pada bagian aljabar linier.

Pandas

Pandas adalah sebuah library yang menyediakan struktur data dan fungsi untuk menganalisis data yang mudah digunakan serta memiliki performa yang baik. Seperti NumPy, Pandas juga dibuat oleh SciPy dan menggunakan lisensi BSD. Pandas menyediakan berbagai metode dan kelas yang fleksibel dan berguna untuk membuat struktur data serta menganalisis data dalam skala besar. Library ini juga bisa menyelesaikan masalah umum pada data, seperti data yang tidak lengkap, tidak terorganisir, dan tidak rapi — serta dilengkapi dengan berbagai alat untuk membuat, menggabungkan, menganalisis, dan menampilkan dataset.

Pandas menggabungkan library NumPy yang mampu mengelola data secara fleksibel dengan database relasional seperti SQL. Sehingga membuat kita lebih mudah

melakukan reshape, slice, dice, agregasi data, serta mengakses bagian-bagian tertentu dari data.

Untuk memulai, impor terlebih dahulu library pandas dengan cara mengetik 'import pandas as pd'. Penggunaan "as" di sini berarti kita mengganti cara memanggil pandas dengan menambahkan prefix pd untuk langkah-langkah berikutnya.

Ada tiga jenis struktur data yang tersedia di dalam library ini.

- Series: single-dimensional, array homogen

Series merupakan struktur data dasar dalam Pandas. Series bisa diperbandingkan dengan array satu dimensi seperti yang ada di numpy, hanya saja Series memiliki index dan kita bisa mengatur index untuk setiap elemennya.

Struktur data tersebut dapat menampung berbagai jenis data seperti bilangan bulat, bilangan desimal, serta teks. Series juga mendukung operasi vektor. Secara definisi, series tidak bisa memiliki dua kolom sekaligus, untuk masalah ini bisa menggunakan struktur data frame.

```
pandas.Series(data=None, index=None, dtype=None, name=None,
copy=False, fastpath=False)
```

Berdasarkan dokumentasi pandas, terdapat struktur dari kelas Series seperti berikut:

Adapun parameter-parameter yang dibutuhkan :

- Parameter data bisa berisi array, Iterable, dict, atau nilai scalar
- Parameter index disarankan bernilai unik dan hashable yang memiliki panjang sama dengan data
- Parameter dtype menunjukkan tipe data, bila tidak didefinisikan akan bernilai sesuai dengan data yang diberikan
- Parameter copy berguna untuk menyalin data

Berikut contoh penggunaan data series yang bersumber dari sebuah list python yang bertipe string dan integer.

```
[2] 1 #membuat series tanpa index
    2 Sahabat = pd.Series(['Ali', 'Umar', 'Usman', 'Abu Bakar'])
    3 print(Sahabat)
    4
    5 angka = pd.Series([2,4,6,7])
    6 print(angka)
```

```
0      Ali
1      Umar
2      Usman
3  Abu Bakar
dtype: object
0      2
1      4
2      6
3      7
dtype: int64
```

Pada baris pertama dan keempat ada contoh pembuatan series yang sederhana. Series Sahabat adalah jenis series yang isinya berupa teks, sedangkan angka adalah jenis series yang isinya berupa bilangan bulat. Jika kita tidak menentukan indeks, maka indeks akan dibuat sendiri mulai dari 0 sampai N-1, di mana N adalah jumlah total data. Indeks pada sebuah series tidak hanya menggunakan tipe data integer saja, tetapi juga bisa menggunakan tipe data lainnya seperti string. Sebuah index digunakan untuk mempermudah mengakses series. Berikut ini contoh penggunaan series dengan menggunakan indeks.

```
[3] 1 #membuat series dengan index
    2 khalifah = pd.Series(['Ali', 'Umar', 'Usman', 'Abu Bakar'],
    3 | | | | | | | | | | index = ["4th", "2nd", "3rd", "1st"])
    4 print(khalifah )
    5
```

```
4th      Ali
2nd      Umar
3rd      Usman
1st  Abu Bakar
dtype: object
```

Untuk mengakses data, gunakan index seperti “2nd” pada series khalifah dan 0 pada series sahabat.

```
[5] 1 print(khalifah["2nd"])
     2 print(Sahabat[0])
```

```
↳ Umar
   Ali
```

Hasil dari operasi penjumlahan aritmatika pada dua objek Series mirip dengan proses penggabungan data pada pengolahan database. Indeks yang tidak sama antara dua objek Series akan memiliki nilai NaN. Dalam contoh ini, kedua series memiliki indeks "a" dan "b", jadi kedua elemen tersebut akan dijumlahkan, tetapi untuk indeks "c" dan "d" nilainya adalah NaN

```
[6] 1 s1 = pd.Series([1,2,3], index=["a","b","c"])
     2 s2 = pd.Series([4,5,6], index=["a","b","d"])
     3 print(s1+s2)
```

```
↳ a    5.0
   b    7.0
   c    NaN
   d    NaN
   dtype: float64
```

DataFrame

Data frame adalah array dua dimensi yang memiliki baris dan kolom seperti tabel. Struktur data ini adalah cara yang paling umum digunakan untuk menyimpan data. Secara mudah, data frame adalah seperti tabel atau data berbentuk kolom dan baris, di mana setiap kolom dalam data frame merupakan objek dari series, dan setiap baris terdiri dari elemen-elemen yang ada dalam series (Gambar 2.3).

Series			Series			DataFrame	
apples			oranges			apples	oranges
0	3	+	0	0	=	0	3
1	2		1	3		1	2
2	0		2	7		2	0
3	1		3	2		3	1
							2

Gambar 2.3 Ilustrasi Hubungan Series dan DataFrame

Untuk membuat sebuah dataframe, kita membutuhkan sebuah dictionary data yang berisi kolom-kolom seperti nama, umur, warna kesukaan, dan tinggi badan. Setiap data di dalam dictionary berupa daftar yang menunjukkan informasi tersebut. Misalnya,

pada kunci "nama", terdapat daftar ['Annissa', 'Tri', 'Ardian', 'Edo'], dan pada kunci "umur", terdapat daftar [30, 55, 32, 38]. Setiap elemen yang sama akan ditempatkan dalam satu baris, misalnya nama Annissa akan berada di satu baris dengan usia 30 tahun, dan seterusnya. Berikut adalah contoh cara membuat sebuah dataframe dan perhatikan hasilnya yang ditampilkan dalam bentuk tabel.

```
▶ data = {  
    'nama': ['Annissa', 'Tri', 'Ardian', 'Edo'],  
    'umur': [30, 55, 32, 38],  
    'warna_kesukaan': ['merah', 'biru', 'ungu', 'putih'],  
    'tinggi': [160, 165, 170, 175]  
}  
df = pd.DataFrame(data)  
print(df)
```

```
...      nama  umur warna_kesukaan  tinggi  
0  Annissa   30         merah      160  
1     Tri    55         biru      165  
2  Ardian   32         ungu      170  
3     Edo   38         putih      175
```

Ada dua atribut penting pada dataframe yaitu columns dan values. Atribut columns berisi nama kolom yang digunakan dan atribut values berisi array data yang ada dalam dataframe.

```
df.columns
```

```
Index(['nama', 'umur', 'warna_kesukaan', 'tinggi'], dtype='object')
```

```
# menampilkan data frame  
print(df.values)
```

```
[['Annissa' 30 'merah' 160]  
 ['Tri' 55 'biru' 165]  
 ['Ardian' 32 'ungu' 170]  
 ['Edo' 38 'putih' 175]]
```

Salah satu keunggulan pandas mempermudah dalam pengolahan dan analisis data. Berikut ini beberapa contoh sederhana kemudahan yang ditawarkan oleh pandas:

1. Pengurutan data. Untuk mengurutkan cukup memanggil method `sort_values` lalu memasukkan parameter kolom yang akan menjadi dasar pengurutan. Secara default, pengurutan dilakukan secara menaik (ascending).

```
# pengurutan berdasarkan nama
df.sort_values('nama')
```

```
...      nama  umur  warna_kesukaan  tinggi
```

0	Annissa	30	merah	160
2	Ardian	32	ungu	170
3	Edo	38	putih	175
1	Tri	55	biru	165

Jika ingin melakukan pengurutan secara menurun tinggal mengubah parameter ascending menjadi False. Kemudian jika ingin menggunakan pengurutan pada beberapa kolom maka tinggal mengganti parameter by dengan sebuah list contohnya by=['nama','umur'] dan parameter arahnya menjadi ascending=[True,False] yang berarti pengurutan umur secara menaik dan umur secara menurun.

2. Selecting dan filtering. Selecting dan filtering merupakan kegiatan- kegiatan yang paling banyak dilakukan ketika melakukan persiapan terhadap data seperti feature selection, pengecekan data kosong dan lain- lain. Untuk memilih data pada kolom tertentu cukup dengan mendefinisikan sebuah list kolom dalam braket (tanda “[“ dan “]”) dataframe.

```
# menampilkan kolom nama dan tinggi saja
df[['nama', 'tinggi']]
```

```
      nama  tinggi
```

0	Annissa	160
1	Tri	165
2	Ardian	170
3	Edo	175

Sedangkan untuk pemilihan baris, aturan yang sama seperti pada numpy juga berlaku.

```
# menampilkan 3 baris pertama
df.loc[:2]
```

	nama	umur	warna_kesukaan	tinggi
0	Annissa	30	merah	160
1	Tri	55	biru	165
2	Ardian	32	ungu	170

```
# menampilkan 2 baris terakhir
df.loc[2:]
```

	nama	umur	warna_kesukaan	tinggi
2	Ardian	32	ungu	170
3	Edo	38	putih	175

Sedangkan untuk melakukan filtering dengan syarat yang kompleks juga mudah dilakukan. Misalnya untuk menampilkan semua item dengan syarat umur lebih besar dari 35 tahun dapat dilakukan dengan `df[df['umur']>35]`.

```
# memfilter umur diatas 35 tahun
df[df['umur']>35]
```

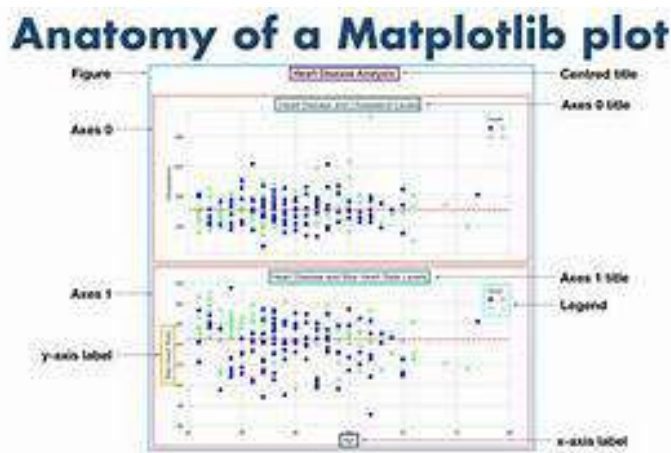
	nama	umur	warna_kesukaan	tinggi
1	Tri	55	biru	165
3	Edo	38	putih	175

3. Loading dan simpan data. Pandas juga disertai dengan berbagai method yang mempermudah kita untuk membaca atau menyimpan dataset pada aplikasi ML yang kita buat. Berikut contoh sederhana proses penyimpanan data menjadi bentuk CSV, dan file Excel.

```
[14] 1 #menyimpan data kedalam file CSV
      2 df.to_csv('nama_file.csv')
      3 df.to_excel('nama_file.xls')
      4 #membaca/loading data dari csv
      5 df = pd.read_csv('nama_file.csv', sep=',')
```

MatPlotLib

Matplotlib merupakan library plotting 2D Python yang menghasilkan gambar publikasi bermutu di dalam berbagai format hardcopy dan lingkungan interaktif sepanjang platform. Matplotlib dapat digunakan di dalam script Python, shell Python dan ipython, server aplikasi web, dan enam GUI toolkit. Matplotlib mencoba untuk membuat hal mudah menjadi lebih mudah. Kita dapat membuat plot, histogram, power spectra, grafik batang, scatterplot, dll, hanya dengan beberapa baris kode. Gambar 2.4 menunjukkan komponen-komponen dari sebuah chart yang disupport oleh library ini, mulai dari label, legend, grid dan lain-lain



Gambar 2.4 Anatomi sebuah chart (sumber: matplotlib.org)

Import library

Seperti library python lainnya, setiap penggunaan harus melalui proses import. Caranya sama seperti proses import pada Numpy atau Panda.

```
[1] 1 import matplotlib as mpl
     2 import matplotlib.pyplot as plt
     3 %matplotlib inline
     4
     5 plt.style.use('seaborn-pastel')
```

Agar dapat menggunakan library ini pada Ipython Notebook maka perintah pada baris ke-3, %matplotlib, diperlukan. Pada IPython notebook, grafis yang dihasilkan akan langsung ditampilkan pada halaman notebook. Sedangkan baris ke-5 untuk menentukan style apa yang akan digunakan dalam chart dalam yang dihasilkan. Berikut ini beberapa contoh penggunaan matplotlib:

Line Chart

Line Chart atau line plot atau line graph atau curve chart merupakan sebuah tipe grafis yang digunakan untuk memampikan sejumlah data titik yang biasa disebut “marker”, lalu marker tersebut dihubungkan dengan garis lurus. Grafis ini terdiri atas dua sumbu yaitu sumbu tegak (vertikal) dan sumbu mendatar (horizontal). Biasanya jenis grafis ini digunakan untuk memvisualisasikan sebuah trend pada data pada interval waktu tertentu. Biasanya sumbu mendatar digunakan untuk menunjukkan waktu.

Perhatikan contoh berikut [3], diberikan dua buah list data yaitu data tahun dari 1920 sampai 2010 dan data pengangguran (variabel Tahun dan Pengangguran pada baris ke-1 dan ke-2). Untuk membuat plot maka cukup memanggil method `plt.plot`. Method tersebut membutuhkan dua parameter wajib sebagai x dan y serta beberapa parameter opsional seperti warna dan jenis marker. Pada contoh berikut digunakan empat parameter x, y, color dan marker. Pada baris ke-4 digunakan untuk membuat judul chart dan baris ke-5 dan ke-6 untuk membuat label pada chart.

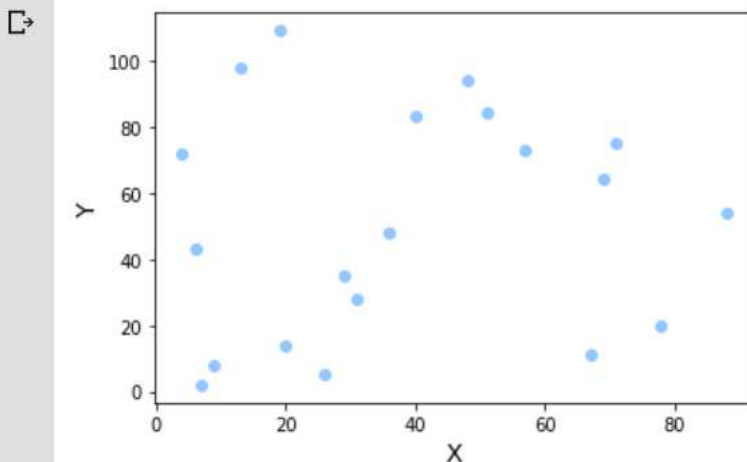
```
[3] 1 Tahun = [1920,1930,1940,1950,1960,1970,1980,1990,2000,2010]
    2 Pengangguran = [9.8,12,8,7.2,6.9,7,6.5,6.2,5.5,6.3]
    3 plt.plot(Tahun, Pengangguran, color='red', marker='o')
    4 plt.title('Pengangguran Per-Tahun', fontsize=14)
    5 plt.xlabel('Tahun', fontsize=14)
    6 plt.ylabel('Index pengangguran', fontsize=14)
    7 plt.grid(True)
    8 plt.show()
```



Plot Sebaran (Scatter Plot)

Sebuah plot sebaran merupakan sebuah grafik yang menunjukkan hubungan antara dua set data, seperti hubungan antara umur dan tinggi. Plot ini mirip dengan line plot, tetapi dalam plot ini setiap marker tidak dihubungkan dengan garis. Kali ini kita akan menggambar sebuah plot sebaran menggunakan matplotlib. Sebagai contoh buat dua set data, x dan y (pada baris ke-1 dan ke-2), lalu panggil method scatter pada baris ke 3 menggunakan dua parameter yaitu x dan y untuk membuat grafisnya. Baris ke-4 dan ke-5 untuk menampilkan nama label pada grafis:

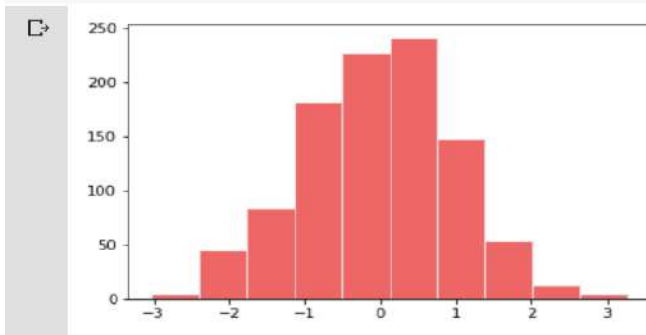
```
[4] 1 x = [20,4,6,7,9,13,19,26,29,31,36,40,48,51,57,67,69,71,78,88]
     2 y = [14,72,43,2,8,98,109,5,35,28,48,83,94,84,73,11,64,75,20,54]
     3 plt.scatter(x,y)
     4 plt.xlabel('X', fontsize=14)
     5 plt.ylabel('Y', fontsize=14)
     6 plt.show()
     7
```



Histogram

Kata histogram berasal dari bahasa Yunani: histos, dan gramma. Pertama kali digunakan oleh Karl Pearson pada tahun 1895 untuk memetakan distribusi frekuensi dengan luasan area grafis batangan menunjukkan proporsi banyak frekuensi yang terjadi pada tiap kategori. Sebuah histogram merupakan grafik yang menampilkan frekuensi data menggunakan batang, dimana angka dikelompokkan dalam rentang atau kelompok tertentu. Dengan kata lain, frekuensi setiap elemen data di dalam daftar ditunjukkan menggunakan histogram. Angka yang dikelompokkan dalam bentuk rentang tertentu disebut bins (kelompok):

```
[5] 1 # mencoba mengubah warna chart dan edgenya
     2 x = np.random.randn(1000)
     3 plt.hist(x, edgecolor='#E6E6E6', color='#EE6666');
```



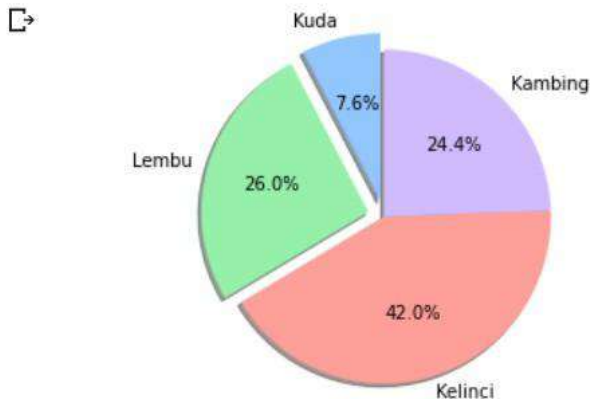
Pada contoh, sebuah dataset x merupakan 1000 nilai acak, lalu dengan menggunakan method `plt.hist` untuk membuat sebuah histogram dengan warna #EE6666.

Pie Chart

Pie chart atau diagram lingkaran merupakan grafik yang berbentuk lingkaran yang dibagi menjadi beberapa irisan dan luasnya bergantung kepada proporsi numerik atau kuantitas dari data yang dimiliki. Satu lingkaran menunjukkan bagian utuh atau seratus persen. Setengah lingkaran menunjukkan proporsi setengah dari total atau lima puluh persennya, dan seterusnya.

Pada contoh [6], ditunjukkan sebuah pie chart hewan ternak pada sebuah peternakan dimana pichart tersebut terdiri atas empat label yang mewakili hewan ternak yaitu Kuda, Lembu, Kelinci dan Kambing. Selanjutny ada variabel `sizes` yang menunjukkan jumlah hewan-hewan tersebut sesuai urutan. Lalu library akan otomatis mengkonversi jumlah-jumlah hewan tersebut kedalam persentase dimana 10 ekor kuda mewakili 7.6% dari total area chart, 34 ekor lembu mewakili 26% dari total area chart, 55 ekor kelinci mewakili 42% dari total area chart, dan 32 ekor kambing mewakili 24.6% dari total area chart.

```
[6] 1 # Memampilkan Pie chart
2 labels = 'Kuda', 'Lembu', 'Kelinci', 'Kambing'
3 sizes = [10, 34, 55, 32] #ukuran Pie chart
4 posisi = (0.1, 0.1, 0, 0) # elemen 1 & 2 akan menjorok keluar
5 fig1, ax1 = plt.subplots()
6 ax1.pie(sizes, explode=posisi, labels=labels, autopct='%1.1f%%',
7         shadow=True, startangle=90)
8 ax1.axis('equal')
9 plt.show()
```



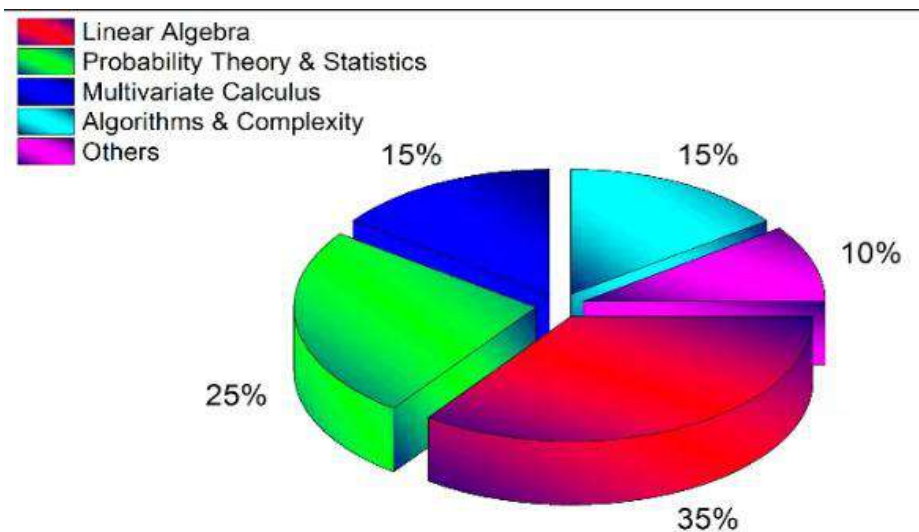
2.2 Pondasi Matematika dan Statistika

Kemampuan matematika hukumnya merupakan WAJIB jika pembaca ingin serius dibidang ini. Adapun kemampuan yang harus ada sebelum pembaca mempelajari ML Aljabar Linier, Kalkulus, Probabilitas dan Statistik. Pada buku ini tidak dibahas secara mendalam tentang pondasi-pondasi matematika yang pembaca butuhkan tetapi akan disebut topik-topik yang perlu pembaca kuasai ketika belajar ML. Deisenroth, M. P. , F. A. A. , & O. C. S. (2020). *Mathematics for Machine Learning*. Cambridge University Press. Dan Goodfellow, I. , B. Y. , & C. A. (2016). *Deep Learning*. MIT Press.

Adapun topik-topik matematika yang harus digaris bawahi:

- Aljabar linier merupakan sub-bidang matematika yang berkaitan dengan vektor, matriks, dan transformasi linier. Jika pembaca berasal dari jurusan ilmu komputer, materi ini bukanlah materi yang asing. Menurut penyusun, ini merupakan Pondasi Kunci untuk bidang ML, dari notasi yang digunakan untuk menggambarkan operasi algoritma hingga implementasi algoritma dalam kode. Aljabar linier ini muncul hampir di semua topik machine learning seperti Operasi Matrix , Eigenvalues & Eigenvectors, Vector Spaces, Regresi Linier, Eigen Decomposition, Singular Value Decomposition (SVD), Principal Component Analysis (PCA) dan lain-lain.

- Teori peluang dan statistika memberikan kontribusi paling besar dalam bidang ini. Apapun materi-materi Teori peluang dan statistika yang diperlukan ML merupakan Kombinatorika, Teorema bayes, Aturan Probabilitas, Kondisional dan Joint Distributions, Standard Distributions (Bernoulli, Binomial, Multinomial, Uniform dan Gaussian), Maximum Likelihood Estimation (MLE), dan Metode Sampling.
- Kalkulus beberapa materi yang harus dipelajari Turunan dan Integral, Fungsi Vector-Values, Directional Gradient, Hessian, Jacobian, Laplacian dan Lagrangian Distribution.



Gambar 2.5 Topik Matematika Penting Pada Bidang Machine Learning (sumber: Wale)

- Teori matematika lainnya merupakan topik-topik matematika yang dibutuhkan tetapi tidak masuk kedalam kategori lainnya Analisa Real dan Complex (Sets dan Sequences, Topology, Metric Spaces, Limits, Cauchy Kernel, Fourier Transforms), Teori Informasi (Entropy, Information Gain), Function Spaces dan Manifolds.

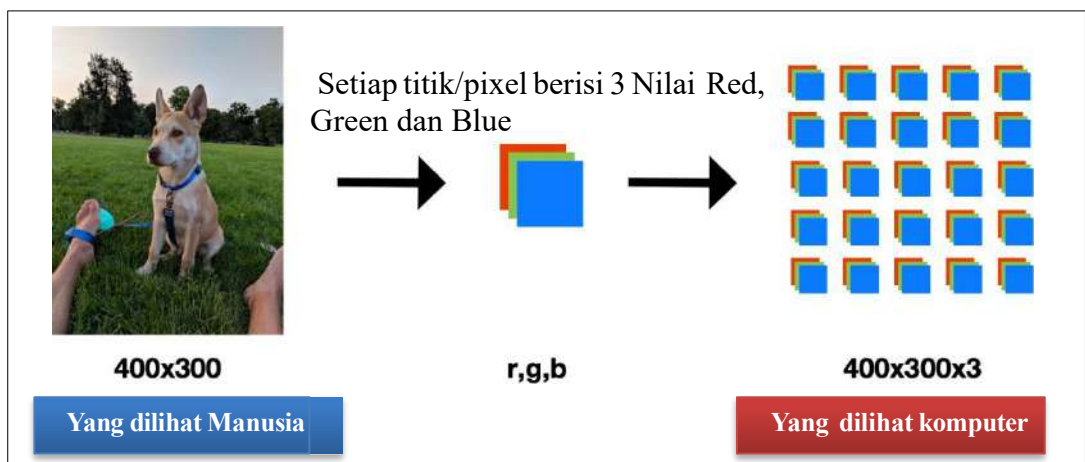
Berikut ini akan dibahas beberapa topik yang penyusun anggap penting untuk diingat kembali sebagai dasar *machine learning*

Aljabar Linier

Aljabar linier merupakan cabang dari matematika yang menjadi syarat utama bagi orang-orang yang belajar ML. Melalui Aljabar Linier-lah banyak teori dan metode ML ditemukan dan dapat digunakan untuk memecahkan masalah terkait data. Pengetahuan yang kokoh tentang Aljabar Linier akan membuat perjalanan ML pembaca lebih fokus. Sederhananya, aljabar linier ini dapat dikatakan matematikanya data. Jika kita ingin

merepresentasikan data maka bahasa yang akan kita pilih merupakan scalar, vektor, matrik atau tensor. Dengan kata lain, ketika kita berusaha mengolah data maka sebenarnya kita mengolah scalar, vektor, matrik atau tensor.

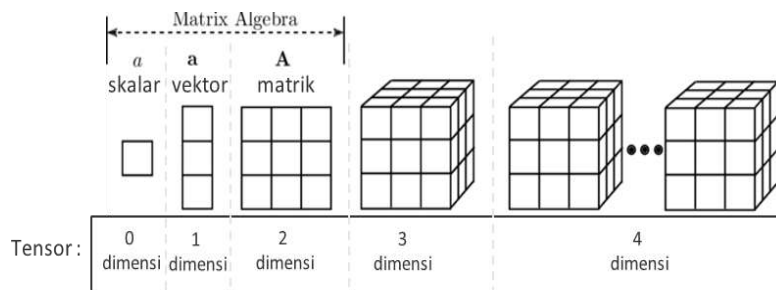
Sebagai contoh perhatikan gambar Anjing pada Gambar 2.6. Sebagai manusia, kita akan melihat kaki manusia dan seekor Anjing coklat yang duduk di atas rumput. Tetapi bagi komputer, sebuah gambar tidak lebih hanya sekumpulan angka. Setiap titik digambar yang disebut piksel yang merupakan satuan terkecil dari sebuah gambar. Sebuah piksel terdiri atas tiga nilai yaitu Red, Green dan Blue. Secara matematis, sebuah piksel dapat dikatakan sebuah vektor karena berisi tiga nilai dan sebuah gambar terdiri atas susunan pixel-pixel dalam bentuk matrik sehingga membentuk sebuah tensor.



Gambar 2.6 Cara Manusia & Komputer Melihat Gambar

Pengolahan terhadap gambar seperti memutar (rotasi) atau memperbesar atau memperkecil, menggabungkan dan lain-lain merupakan proses pengolahan scalar, vektor, matrik atau tensor. Materi tentang pengolahan- pengolahan tersebut merupakan sub bagian dari ilmu aljabar linear.

Berikut ini penjelasan rinci mengenai Scalar, Vektor, Matrik dan Tensor serta operasi-operasi dasar yang dapat dilakukan pada masing-masing objek:



Gambar 2.7 Perbedaan Scalar, Vektor, Matrik dan Tensor

Skalar

Skalar merupakan sebuah angka. Biasanya dilambangkan dengan huruf kecil ditulis miring seperti “ x ”. Tipe data sebuah skalar tidak hanya integer tetapi juga dapat berisi natural number (3, 4, 5, ...) , rational number ($4/3$, 1.666666...), atau irrational number.

Vektor

Vektor merupakan sebuah elemen yang mendasar dalam aljabar linier. Vektor merupakan sekumpulan angka yang terurut (an ordered finite list). Vektor sering digunakan pada ML untuk mendeskripsikan algoritma dan proses- proses seperti mendefinisikan variabel target (y) pada proses pelatihan algoritma. Sebuah vektor dibangun oleh komponen-komponen yang biasanya berupa angka. Dapat dianalogikan bahwa vektor itu merupakan sekumpulan angka. Notasi yang digunakan untuk vektor biasanya direpresentasikan dengan huruf kecil, contoh

$$v = (v_1, v_2, v_3) \text{ atau } v = (21, 52, 33)$$

Sebagian buku juga sering menotasikannya secara horizontal

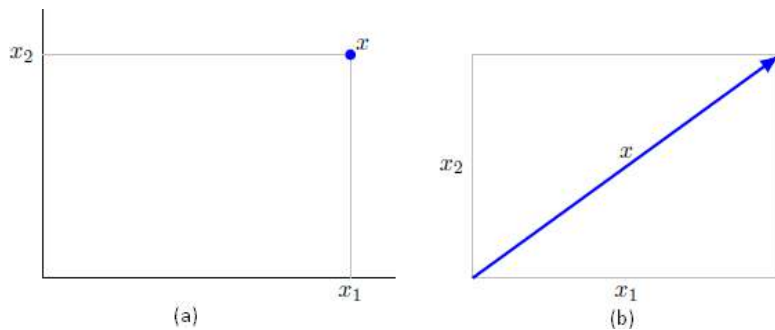
$$v = \begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix} \text{ atau } v = \begin{pmatrix} 21 & 52 & 33 \end{pmatrix}$$

Elemen (isi atau komponen) dari sebuah vektor merupakan nilai-nilai dari array v , sedangkan ukuran (dimensi) dari vektor merupakan jumlah semua elemen (count, bukan sum nilai) yang ada didalam vektor. Sebagai contoh vektor v memiliki ukuran = 3, dan elemen ke-2 merupakan 52. Sebuah vektor dengan ukuran n biasanya disebut n -vector. Sebuah vektor dengan ukuran n biasanya disebut n -vector.

Berikut ini beberapa contoh penggunaan vektor dalam kehidupan sehari- hari

- Lokasi dan perpindahan. Sebuah vektor dengan ukuran 2 (2-vector) dapat digunakan untuk merepresentasikan sebuah titik atau lokasi dalam sebuah ruang dua dimensi seperti pada Gambar 2.8. Selanjutnya matrik berukuran 3 (3-vector) dapat juga merepresentasikan titik pada ruang tiga dimensi.

Selain itu, ia juga dapat merepresentasikan perpindahan pada ruang dua dimensi ataupun 3 dimensi. Biasanya perpindahan ini dilambangkan dengan sebuah panah. Sebuah vektor juga dapat merepresentasikan kecepatan atau percepatan, pada waktu tertentu, dari suatu titik yang bergerak di ruang 2-D atau 3-D.



Gambar 2.8 Vektor berukuran 2 menggambarkan posisi titik x dengan koordinat x_1, x_2 (gambar a) dan sebuah vektor x yang merepresentasikan perpindahan sejauh x_1 pada sumbu x dan x_2 pada sumbu y (gambar b)

- Warna. Sebuah vektor berukuran 3 dapat digunakan untuk menyimpan nilai sebuah warna yang terdiri atas tiga nilai dari warna merah (Red), hijau (Green) dan biru (Blue). Misalnya vektor $(0,0,0)$ merepresentasikan warna hitam, vektor $(255,0,0)$ merepresentasikan warna merah, vektor $(0,255,0)$ merepresentasikan warna hijau, vektor $(0,0,255)$ merepresentasikan warna biru dan vektor $(255,255,0)$ merepresentasikan warna kuning yang merupakan campuran dari warna merah dan hijau.
- Portofolio Saham. Sebuah vektor s berukuran n dapat mewakili portofolio saham atau investasi dalam n aset yang berbeda, dimana s_i merupakan jumlah saham atau aset yang dimiliki oleh i . Sebagai contoh, Vektor $(100; 50; 20)$ mewakili portofolio yang terdiri dari 100 saham milik orang ke-1, 50 saham milik orang ke-2, dan 20 saham milik orang ke-3.
- Data Pembelian. Sebuah vektor p berukuran n dapat mewakili sejumlah data pembelian yang dilakukan pada sebuah periode waktu tertentu. p_i merepresentasikan data pembelian ke i .

Untuk merepresentasikan sebuah vektor pada menggunakan Python dapat memanfaatkan NumPy. Sebagai contoh kita akan membuat sebuah vektor dengan nilai 1, 2 dan 3 adapun fungsi `np.array` merupakan fungsi yang digunakan untuk membuat sebuah vektor.

```
[1] 1 import numpy as np
    2 # Membuat sebuah vektor
    3 a = np.array([1, 2, 3])
    4 print(a)
    5
```

➞ [1 2 3]

Operasi Penambahan Vektor

Dua buah vektor yang panjang atau ukurannya sama dapat ditambahkan menjadi sebuah vektor baru. Vektor baru yang dihasilkan pun akan memiliki ukuran yang sama. Pada penjumlahan ini berlaku sifat komutatif dan asosiatif.

Implementasinya dalam bahasa Python sangat sederhana.

```
[2] 1 # Membuat sebuah array
      2 a = np.array([1, 2, 3])
      3 b = np.array([4, 5, 6])
      4 # penambahan vector
      5 c = a + b
      6 print(a, "+", b, "=", c)
      7
```

➞ $[1\ 2\ 3] + [4\ 5\ 6] = [5\ 7\ 9]$

```
[3] 1 # Membuat sebuah array
      2 a = np.array([1, 2, 3])
      3 b = np.array([4, 5, 6])
      4 # penambahan vector
      5 c = b - a
      6 print(b, " - ", a, "=", c)
      7
```

➞ $[4\ 5\ 6] - [1\ 2\ 3] = [3\ 3\ 3]$

1. Operasi Perkalian Vektor

Sama seperti penambahan, dua buah vektor yang panjang atau ukurannya sama dapat dikurangkan menjadi sebuah vektor baru. Vektor baru yang dihasilkan pun akan memiliki ukuran yang sama.

```
[4] 1 a = np.array([1, 2, 3])
      2 b = np.array([4, 5, 6])
      3 # perkalian vector
      4 c = b * a
      5 print(b, " x ", a, "=", c)
```

➞ $[4\ 5\ 6] \times [1\ 2\ 3] = [4\ 10\ 18]$

2. Operasi Pembagian Vektor

Sama seperti penambahan, dua buah vektor yang panjang atau ukurannya sama dapat dikurangkan menjadi sebuah vektor baru. Vektor baru yang dihasilkan pun akan memiliki ukuran yang sama.

```
[5] 1 # Membuat sebuah array
    2 a = np.array([8, 8, 8])
    3 b = np.array([1, 2, 3])
    4 # penambahan vector
    5 c = b / a
    6 print(b, " / ", a, "=", c)
    7
```

```
➡ [1 2 3] / [8 8 8] = [0.125 0.25 0.375]
```

3. Operasi Dot Product

Sama seperti penambahan, dua buah vektor yang panjang atau ukurannya sama dapat dikurangkan menjadi sebuah vektor baru. Vektor baru yang dihasilkan pun akan memiliki ukuran yang sama.

$$v = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \cdot \begin{pmatrix} i_1 \\ i_2 \\ i_3 \end{pmatrix} = (v_1 i_1 + v_2 i_2 + v_3 i_3)$$

```
# Membuat sebuah array
a = np.array([2, 2, 2])
b = np.array([1, 2, 3])

# perkalian vector
c = np.dot(a, b)

print(b, " . ", a, "=", c)
```

```
[1 2 3] . [2 2 2] = 12
```

Matrik

Matrik merupakan susunan persegi panjang dari bilangan-bilangan yang diatur dalam baris dan kolom. Adapun atribut yang penting merupakan dimensi dari matrik yang menunjukkan jumlah baris dan kolom. Notasi yang digunakan untuk merepresentasikan matrik merupakan

$$v = ((v_{11}, v_{12})(v_{21}, v_{22}), (v_{31}, v_{32}))$$

Sebagian buku juga sering menotasikannya matrik secara horizontal

$$v = \begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \\ v_{31} & v_{32} \end{pmatrix}$$

Untuk merepresentasikan sebuah vektor pada menggunakan Python dapat memanfaatkan NumPy. Sebagai contoh kita akan membuat sebuah vektor dengan nilai 1, 2 dan 3 adapun fungsi `np.array` merupakan fungsi yang digunakan untuk membuat sebuah vektor..

```
[7] 1 # Membuat sebuah matrix  
2 a = np.array ([[1, 2, 3], [4, 5, 6]])  
3 print(a)
```

```
↳ [[1 2 3]  
    [4 5 6]]
```

Operasi-operasi yang terdapat dimatrix relatif sama dengan fungsi yang ada di vektor. Berikut ini beberapa operasi yang sering dilakukan pada matrik menggunakan numpy

Transpose

```
[8] 1 #mentranspose matrix a  
2 a = np.array ([[1, 2, 3], [4, 5, 6]])  
3 b = np.transpose(a)  
4 print(b)
```

```
↳ [[1 4]  
    [2 5]  
    [3 6]]
```

Inverse

```
[9] 1 #menginverse matrix X
    2 x = np.array([[1,2],[3,4]])
    3 y = np.linalg.inv(x)
    4 print (y)
```

```
↳ [[-2.  1. ]
    [ 1.5 -0.5]]
```

Statistika

Statistika merupakan sebuah ilmu yang mempelajari bagaimana cara merencanakan, mengumpulkan, menganalisis, lalu menginterpretasikan, dan akhirnya mempresentasikan data. Singkatnya, statistika merupakan ilmu yang bersangkutan dengan suatu data. Ilmu ini dapat dikatakan pondasi dasar dari Machine Learning, oleh karena itu, pembaca harus dapat memahami, setidaknya konsep-konsep dasar dalam statistika. Berikut konsep-konsep yang banyak muncul dalam ML.

Probabilitas

Probabilitas atau sering disebut peluang sering diasosiasikan dengan paling tidak sebuah kejadian. Kejadian ini bisa apa saja, sebagai contoh pelemparan dadu, pengambilan bola berwarna dan lain-lain. Pada contoh tersebut semua luaran yang dihasilkan akan bersifat acak (random), misalnya kecepatan pembaca mengocok dadu tidak akan mempengaruhi hasil luarannya. Biasanya variabel yang merepresentasikan luaran kejadian disebut variabel acak (random variable).

Ada banyak pengertian probabilitas, salah satunya dari perspektif statistik merupakan rasio frekuensi suatu item atau kejadian. Berikut beberapa contoh probabilitas:

1. Sebuah mata uang logam mempunyai sisi dua (Huruf dan Gambar), Jika uang tersebut dilambungkan satu kali, maka peluang untuk keluar sisi Huruf $\frac{1}{2}$.
2. Sebuah dadu yang memiliki enam sisi dilemparkan maka peluang untuk keluar mata 'lima' saat pelemparan dadu tersebut satu kali $\frac{1}{6}$ (karena banyaknya permukaan dadu 6)

Dalam suatu percobaan semua hasil yang mungkin disebut ruang sampel, sedangkan setiap anggota dalam ruang sampel disebut titik sampel dan hasil yang diharapkan disebut kejadian. Peluang kejadian A dinotasikan dengan $P(A)$ merupakan perbandingan banyaknya hasil kejadian A dinotasikan $n(A)$ terhadap banyaknya semua hasil yang mungkin dinotasikan dengan $n(S)$ dalam suatu percobaan.

Secara Empiris maka dapat dikatakan persamaan peluang :

$$p(E) = \frac{X}{N} \quad (2-1)$$

p: Probabilitas

E: Event (Kejadian)

X: Jumlah kejadian yang diinginkan (peristiwa) N: Keseluruhan kejadian yang mungkin terjadi Kisaran nilai peluang suatu kejadian A merupakan $0 \leq P(A) \leq 1$. Jika $P(A) = 0$

disebut kemustahilan dan $P(A) = 1$ disebut kepastian.

DEFINISI

Ruang sampel adalah himpunan dari semua hasil yang mungkin pada suatu percobaan/kejadian. Contoh: Pada pelemparan sebuah dadu, maka ruang sampelnya adalah $S = \{1,2,3,4,5,6\}$

Titik sampel adalah anggota-anggota dari ruang sampel atau kemungkinan-kemungkinan yang muncul. Contoh: Pada pelemparan sebuah dadu, maka titik sampelnya : (1), (2), (3), (4), (5), dan (6)

Selain itu, probabilitas juga merupakan sebuah tool yang dapat kita gunakan untuk mengukur tingkat kepercayaan (degree of belief), biasanya disebut Probabilitas Bayesian. Teori probabilitas Bayesian merupakan satu dari cabang teori statistik matematik yang memungkinkan kita untuk membuat satu model ketidakpastian dari suatu kejadian yang terjadi dengan menggabungkan pengetahuan umum dengan fakta dari hasil pengamatan.

Central Tendency

Teori central tendency, mungkin teori yang paling tua yang pernah kita kenal. Sejak duduk dibangku sekolah menengah kita telah diperkenalkan dengan teori-teori ini. Sebuah pengukuran central tendency melambangkan sebuah nilai yang mencoba mendeskripsikan dataset dengan mengidentifikasi nilai “posisi tengah” dataset. Pengukuran central tendency sebagai berikut:

1. Mean.

Biasanya disebut “rata-rata”. Pengukuran ini dapat digunakan pada data continue. Formula dari rata-rata merupakan jumlah dari seluruh item dibagi dengan jumlah item. Pengukuran ini menggambarkan nilai tengah dari populasi yang diamati. Adapun formula mean :

$$\bar{x} = \frac{x_1 + x_2 + \dots + x_n}{n} = \frac{\sum^n x_i}{n} \quad (2-2)$$

Rata-rata tidak mempertimbangkan pencaran (variabilitas) nilai, sehingga sebelum melakukan interpretasi atas nilai rata-rata perlu melihat variabilitasnya. Perhatikan contoh data umur mahasiswa berikut:

MAHASISWA	1	2	3	4	5	6	7	8	9	10
UMUR	19	18	20	21	18	19	24	23	50	55

Berdasarkan pengukuran, maka rata-rata umur mahasiswa 26.7 tahun. Namun, jika diperhatikan pada data, kebanyakan mahasiswa berada pada umur 18 sampai 24 tahun. Namun karena ada dua mahasiswa yang merupakan pencilan sehingga jika hanya memperhatikan rata-rata maka kita akan kurang tepat dalam menginterpretasikan data. Oleh karena itu, diperlukan pengukuran lainnya seperti median, mode, kurtosis dan lain-lain untuk menginterpretasikan data dengan tepat.

2. Median

Median merupakan nilai tengah dari dataset terurut. Pada median, nilainya tidak terpengaruh kepada outlier. Adapun formula untuk menemukan median merupakan

$$Me = b + p \left(\frac{\frac{1}{2}n - F}{f} \right)$$

Jika jumlah data kita genap, maka nilai median kita berada di antara kedua nilai yang berada di tengah; sebagai contoh jika kita mempunyai data 1,1,2,3,3,4,4,5 maka nilai median kita merupakan 3 karena kita mempunyai 8 data sehingga nilai titik tengahnya berada di posisi ke-4 (nilai 3) dan ke-5 (nilai 3) dan lebih tepatnya lagi data di kedua posisi ini ditambahkan lalu dibagi 2.

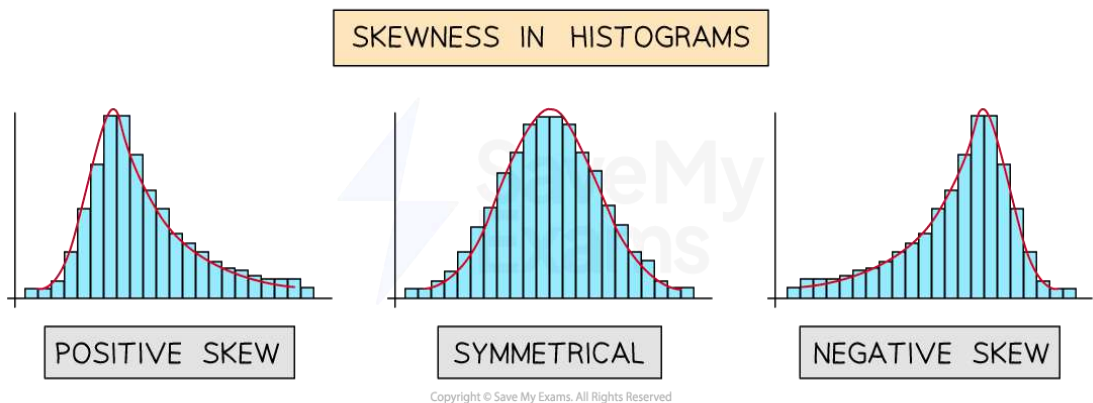
Median lebih sering digunakan jika Mean tidak mampu menjelaskan data kita dengan baik, sehingga diperlukan pengukuran titik tengah menggunakan pengukuran lain.

3. Mode

Mode atau modus merupakan nilai yang paling sering muncul dalam suatu dataset. Modus juga merupakan nilai mayoritas atau nilai dengan frekuensi paling tinggi. Perhitungan modus dapat diterapkan pada data numerik maupun data kategoris,

contohnya dalam menentukan data dari warna paling banyak disukai siswa dan mayoritas nilai ulangan sebuah mata pelajaran yang diperoleh oleh siswa dalam suatu kelas.

Dalam menganalisa data menggunakan prinsip central tendency biasanya kita akan bertemu istilah Skewness dan Kurtosis. Istilah 'skewness' digunakan untuk berarti tidak adanya simetri dari rata-rata dataset. Perhatikan gambar 2.9



Gambar 2.9 Skewness pada data (www.savemyexams.com)

Data dapat didistribusikan dengan banyak cara, seperti menyebar lebih banyak di sebelah kiri atau di sebelah kanan atau merata. Ketika data tersebar secara seragam di titik pusat, itu disebut Distribusi Normal. Kurva ini sangat simetris, berbentuk lonceng, misalnya kedua sisinya sama, sehingga tidak miring. Pada kondisi ini disebut distribusi simetris (symmetrical distribution) jika mean, median, dan mode terletak pada satu titik. Jika mean lebih kecil dari median atau mode maka distribusi akan menunjukkan kemiringan negatif, begitu juga sebaliknya jika mean lebih besar dari mode atau median maka terjadi kemiringan positif.

Dalam statistik, kurtosis didefinisikan sebagai parameter ketajaman relatif dari puncak kurva distribusi probabilitas. Ini memastikan cara pengamatan dikelompokkan di sekitar pusat distribusi. Ini digunakan untuk menunjukkan kerataan atau puncak dari kurva distribusi frekuensi dan mengukur ekor atau outlier dari distribusi.

Kurtosis positif menyatakan bahwa distribusi lebih memuncak daripada distribusi normal, sedangkan kurtosis negatif menunjukkan bahwa distribusi kurang memuncak daripada distribusi normal. Ada tiga jenis distribusi:

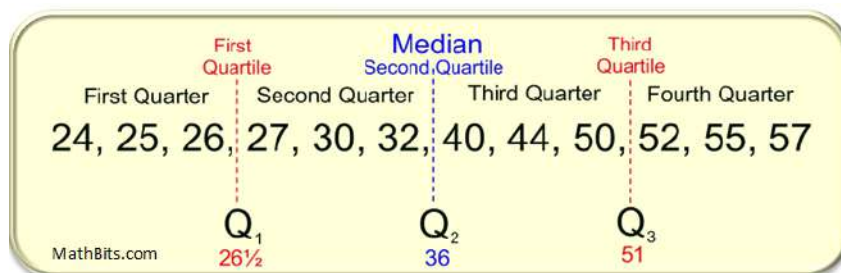
- Leptokurtik: Tajam memuncak dengan ekor gemuk, dan kurang bervariasi.
- Mesokurtik: Sedang memuncak
- Platykurtik: Puncak paling rata dan sangat tersebar

Variabilitas

Variabilitas (ukuran penyebaran) merupakan suatu nilai dari dataset yang menjadi ukuran untuk mengetahui besarnya penyimpangan data dengan nilai rata-rata hitungnya atau menggambarkan bagaimana suatu dataset yang menyebar terhadap pusatnya data.

Variabilitas berfungsi sebagai alat ukur untuk mengetahui homogenitas dan heterogenitas suatu kelompok data. Semakin kecil nilai variabilitas menunjukkan semakin kecil penyebaran suatu kelompok data (homogenitas). Semakin besar nilai variabilitas menunjukkan semakin besar penyebaran suatu kelompok data (heterogenitas). Berikut ini beberapa konsep dalam variabilitas yang penting untuk diketahui diantaranya

- Range merupakan perbedaan antara data numerik terkecil hingga data terbesar. Tidak banyak informasi yang bisa kita ketahui selain mengetahui perbedaan besar data kita
- Quartile merupakan nilai-nilai yang membagi data menjadi 4 bagian. Bagian yang terbagi disebut dengan Quarter. Jika kita menyebut Quartile, maka yang sebenarnya kita perhatikan nilai yang membagi, bukan hasil bagiannya.
- Terdapat 3 jenis Quartile pada saat menggunakan metode Quartile, yaitu Q1 (Nilai antara median dengan data terkecil), Q2 (Median), dan Q3 (Nilai antara median dengan terbesar).



Gambar 2.10 Quartile membagi data yang diurutkan menjadi 4 bagian. Setiap bagian yang terbagi disebut dengan quarter.

- Variance merupakan pengukuran suatu variabilitas dari data untuk mengetahui seberapa jauh data yang dimiliki tersebar.

Variance dihitung berdasarkan total dari setiap data (x_i) dikurangi dengan mean data ($\bar{x}$). Sedikit perbedaan jika kita berbicara mengenai data pada populasi dan sampel. Pada variance populasi kita membagi data kita dengan seluruh jumlah sampel data (N), sedangkan jika data sampel maka kita membaginya dengan jumlah data yang ada dikurangi 1 ($N-1$). Ini dilakukan karena data sampel memiliki ketidakpastian dibandingkan populasi sehingga kita memperbesar perhitungan persebaran kita.

Menggunakan variance berarti kita menjelaskan dasar kita melalui titik tengah mean, sehingga variance dapat menjelaskan seberapa tersebar data kita dari mean dan satu sama lainnya. Jika variance kita kecil, maka data kita tersebar dekat dengan nilai mean sedangkan jika nilai variance besar menandakan data kita semakin tersebar jauh dari mean dan dengan satu sama lain. Selain itu, karena variance melakukan pemangkatan dari data maka data yang semakin jauh nilainya akan semakin dibesarkan sehingga kita bisa memperkirakan seberapa banyak data yang jauh dari mean.

- **Standard Deviation.** Standard deviation merupakan measure of spread yang paling sering digunakan karena memberikan informasi yang jelas dan intuitif. Untuk mendapatkan nilai Standard deviation kita hanya perlu melakukan akar kuadrat terhadap variance.

Standard deviation menggambarkan seberapa berbeda nilai di data kita terhadap mean. Jika menggunakan bahasa sehari-hari, standard deviation merupakan nilai plus-minus dari mean ($\bar{x} \pm s$). Selain itu standar deviation juga digunakan di dalam empirical rule atau 68–95–99.7 rule dimana normalnya data kita tersebar sebesar $\pm 1 * \text{STD}$ (68 % data), $\pm 2 * \text{STD}$ (95 % data), dan $\pm 3 * \text{STD}$ (99.7% data). Data yang terletak lebih atau kurang dari batas tersebut menandakan bahwa data tersebut merupakan suatu outlier.

Probability Distribution

Probability Distribution merupakan suatu fungsi matematika yang memberikan probabilitas (likelihood) dari variabel acak untuk memiliki suatu nilai. Dengan kata lain, kemungkinan suatu variabel acak untuk memiliki suatu nilai akan tergantung dari Probability Distribution. Dalam Machine Learning (ML), konsep ini merupakan jantung dari cara model belajar. ML pada dasarnya merupakan upaya untuk memperkirakan distribusi probabilitas dari data yang ada di dunia nyata agar model bisa melakukan prediksi terhadap data baru.

Hubungan Distribusi Probabilitas dengan Machine Learning

1. **Representasi Data:** Data input dalam ML sering dianggap sebagai sampel yang diambil dari distribusi probabilitas tertentu.
2. **Ketidakpastian (Uncertainty):** Model ML menggunakan distribusi untuk mengukur seberapa yakin mereka terhadap sebuah prediksi (misalnya: "Ada peluang 85% gambar ini merupakan kucing").
3. **Loss Function:** Banyak algoritma ML bekerja dengan meminimalkan jarak antara distribusi prediksi model dengan distribusi data asli (contoh: Cross-Entropy Loss).

Secara umum, distribusi dibagi menjadi dua kategori besar:

1. Distribusi Diskret (Discrete Distribution)

Digunakan untuk variabel yang memiliki nilai terpisah (seperti klasifikasi Ya/Tidak).

- Bernoulli Distribution: Hanya ada dua kemungkinan hasil (0 atau 1).
- Contoh ML: Digunakan dalam Logistic Regression untuk klasifikasi biner, seperti mendeteksi apakah sebuah email merupakan Spam (1) atau Bukan Spam (0).
- Binomial Distribution: Serangkaian percobaan Bernoulli (beberapa kali percobaan Ya/Tidak).
- Contoh ML: Menghitung probabilitas berapa banyak pelanggan yang akan melakukan churn (berhenti berlangganan) dari total 100 pengguna.

2. Distribusi Kontinu (Continuous Distribution)

Digunakan untuk variabel yang memiliki rentang nilai (seperti angka desimal).

- Normal (Gaussian) Distribution: Distribusi berbentuk lonceng yang simetris. Banyak fenomena alam mengikuti pola ini.
- Contoh ML: Digunakan dalam algoritma Naive Bayes atau sebagai asumsi dasar dalam Linear Regression. Banyak teknik preprocessing data (seperti Standardisasi) bertujuan mengubah distribusi data menjadi Normal agar model lebih stabil.
- Uniform Distribution: Semua nilai dalam rentang tertentu memiliki peluang yang sama besar.
- Contoh ML: Sering digunakan saat inisialisasi bobot (weight initialization) di awal pelatihan Neural Network, agar semua saraf memiliki peluang yang sama untuk aktif.

Dalam skenario nyata seperti prediksi harga rumah

Bayangkan Pembaca membangun model ML untuk memprediksi harga rumah.

- Tanpa Distribusi: Model hanya memberikan satu angka, misal Rp500 Juta.
- Dengan Distribusi: Model memberikan prediksi dalam bentuk distribusi. Model mungkin mengatakan harga "paling mungkin" merupakan Rp500 Juta, tetapi juga memberikan informasi bahwa ada peluang harga tersebut berada di rentang Rp480 Juta hingga Rp520 Juta.

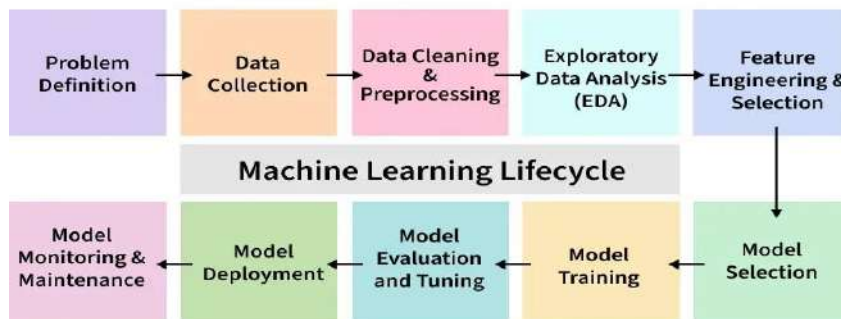
Ini memungkinkan pengembang ML untuk memahami varians dan risiko dari prediksi tersebut.

BAB III

MACHINE LEARNING LIFECYCLE

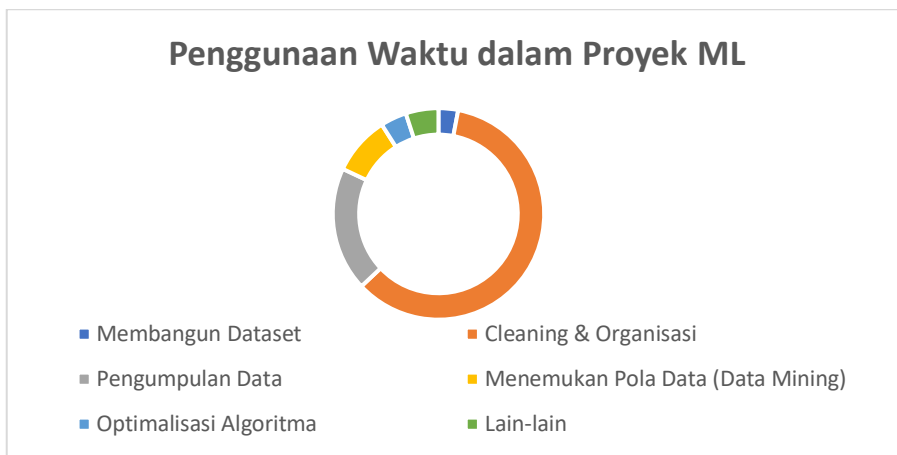
Machine learning lifecycle merupakan sebuah workflow atau alur kerja pengembangan model ML dimulai dari proses pengumpulan data sampai model tersebut siap digunakan. Machine learning lifecycle bersifat forward (maju), dan dapat berulang (iteratif) karena setiap perulangan yang dilakukan bertujuan untuk terus meningkatkan keakuratan dan performa model. Shearer, C. (2000). The CRISP-DM Model: The New Blueprint for Data Mining. Journal of Data Warehousing, 5, 13-22 . Treveil, M. (2020). Introducing MLOps: How to Scale Machine Learning in the Enterprise.

Secara umum, terdapat empat kegiatan utama dalam ML Lifecycle diantaranya merupakan pengumpulan data (Data Acquisition); Persiapan data (Data Preprocessing); Pelatihan, Ujicoba, Validasi dan Optimalisasi Model (Training, Testing, Validation dan Validation); serta Deployment Model. Kegiatan-kegiatan tersebut dapat disusun menjadi sebuah siklus seperti pada Gambar 3.1.



Gambar 3.1 Machine Learning LifeCycle (www.geeksforgeeks.org)

Menurut survei yang dilakukan Forbes pada tahun 2016 menunjukkan bahwa pengembang data scientists dan machine learning menghabiskan waktu sekitar 19% untuk mengumpulkan data dan 60% untuk membersihkan dan mengorganisasi data yang telah dikumpulkan. Artinya kegiatan yang paling berat dalam Machine Learning LifeCycle merupakan pada langkah pertama dan kedua. Setelah mendapatkan data yang bersih 3% dari waktunya digunakan untuk membangun training set dari data dan 9% untuk melakukan training. Jadi dapat disimpulkan 80% waktu digunakan untuk persiapan dan 10% untuk melakukan training dan evaluasi.



Gambar 3. 2 Komposisi yang menyita waktu data saintis (sumber: Forbes.com)

3.1 Pengumpulan Data (Data Aquisition)

Pengumpulan data atau (Data Acquisition) merupakan tahap paling awal pada proses ini. Pengumpulan dapat dilakukan pada sistem yang sedang berjalan, mulai dari data transaksi, ataupun perilaku pengguna seperti data transaksi penjualan, data pencarian barang, data aktifitas pencarian data dan lain-lain. Laudon, K. C. , & L. J. P. (2021). Management Information Systems: Managing the Digital Firm. . Pearson and Provost, F. , & F. T. (2013). Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking.

Langkah pertama yang harus dilakukan merupakan menentukan spesifikasi kebutuhan data. Spesifikasi kebutuhan data berisi poin-poin yang harus tersedia pada data yang akan kita kumpulkan. Dalam menentukan Spesifikasi, kita dapat memulai dengan menjawab pertanyaan-pertanyaan berikut ini

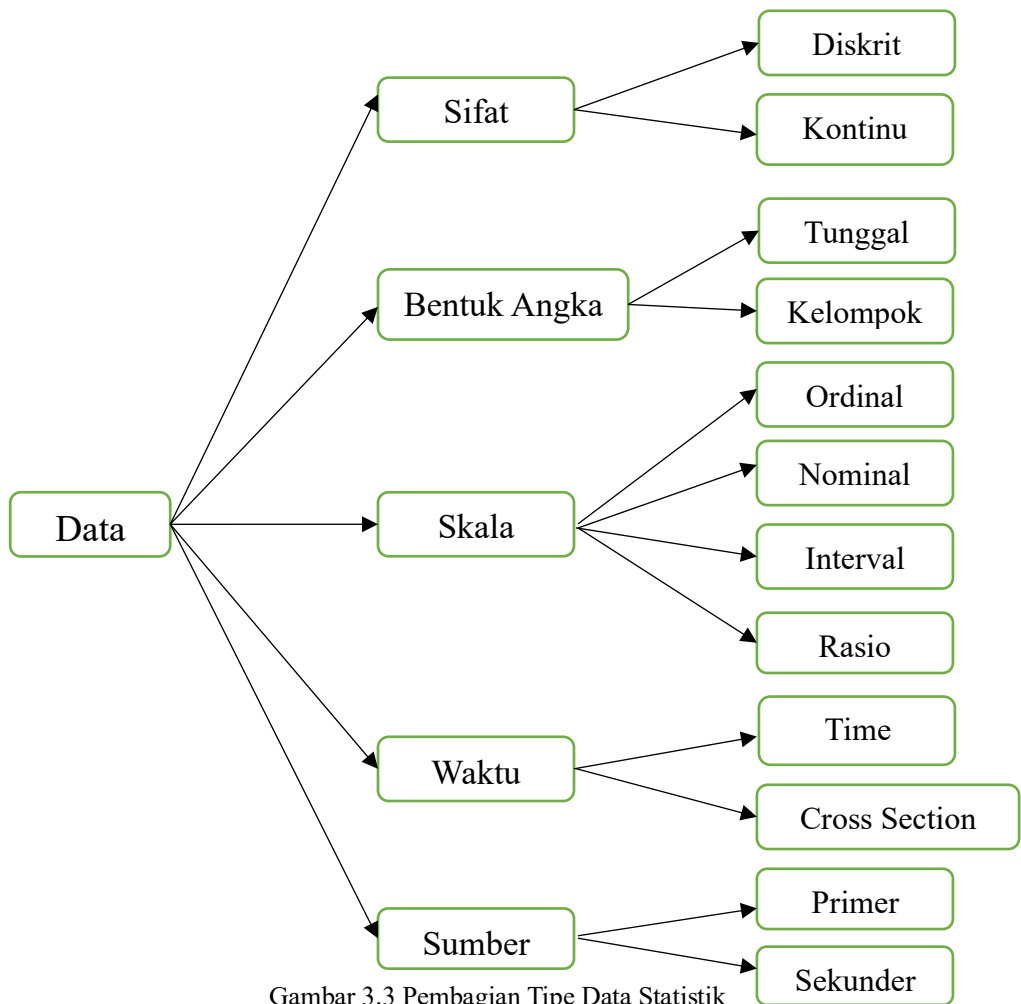
- Data apa yang akan dibutuhkan?
- Apa format data yang tersedia dan kita butuhkan?
- Dimana kita akan mendapatkan data tersebut?
- Apakah data telah tersedia secara gratis?
- Bagaimana cara mengumpulkan data tersebut?
- Bagaimana data tersebut akan digunakan?
- Seberapa besar data yang dibutuhkan?
- Apakah ada standar khusus yang harus dipenuhi oleh data tersebut?
- Bagaimana data itu disimpan dan diolah?

Data biasanya memiliki tipe dan sebelum melakukan analisis data, kita perlu mengetahui tipe dari data yang akan kita analisis. Hal ini mempengaruhi metode apa yang sesuai untuk data tersebut. Sebagai contoh, perhatikan tabel 3.1. yang merupakan

sebuah dataset (kumpulan data) yang menunjukkan kondisi atau keputusan seseorang yang akan bermain tenis. Keputusan orang tersebut bermain tenis tergantung pada empat variabel outlook, temperature, humidity, windy. Keempat variabel ini disebut fitur (feature). Setiap fitur memiliki atribut nilai dengan tipe data dan range tertentu.

Menurut pendekatan statistik, terdapat beberapa tipe atribut data diantaranya:

1. Data Nominal. Data ini termasuk data kategori, yang biasanya digunakan untuk menggambarkan/merepresentasikan karakteristik. Ia dapat digunakan untuk merepresentasikan jenis kelamin, cuaca, bahasa dan lain-lain. Nilai atribut bertipe nominal tersusun atas simbol-simbol yang berbeda, yaitu suatu himpunan terbatas. Sebagai contoh, fitur outlook pada table 3.1 memiliki tipe data nominal yaitu nilainya tersusun oleh himpunan yaitu { sunny, overcast, rainy}. Pada tipe nominal, tidak ada urutan ataupun jarak antar atribut. Tipe ini sering juga disebut kategorial atau enumerasi. Secara umum, tipe output pada supervised learning merupakan data nominal.
2. Data Ordinal. Data ini mirip dengan data nominal, perbedaannya merupakan data ini memperhatikan memiliki urutan. Urutan dari nilai memiliki peran yang penting dan signifikan. Sebagai contoh pada penilaian kepuasan. Pilihan ke-1: Sangat Puas, ke-2 : Puas, ke- 3:Kurang Puas, ke-4: Tidak Puas. Kita mengetahui bahwa pilihan ke- 1 lebih baik dari pilihan ke-2 atau pilihan ke-4 tetapi kita tidak dapat menilai seberapa baiknya. Contoh lainnya data tingkat pendidikan dimana nilainya tersusun oleh himpunan 1-Sekolah Dasar, 2-Sekolah Menengah, 3-Diploma, 4-Sarjana, 5-Pascasarjana.



Gambar 3.3 Pembagian Tipe Data Statistik

1. Data Interval. Tipe interval memiliki urutan dan range nilai yang sama. Sebagai contoh 1-5, 4-10, dst. Kita dapat mentransformasikan/ mengkonversi nilai numerik menjadi nominal dengan cara merubahnya menjadi interval terlebih dahulu. Lalu, kita dapat memberikan nama (simbol) untuk masing-masing interval. Misalkan nilai numerik dengan range 1–100 dibagi menjadi 5 kategori dengan masing-masing interval merupakan $\{1-20, 21-40, \dots, 81-100\}$. Setiap interval kita beri nama, misal interval 81 – 100 diberi nama nilai A, interval 61 – 80 diberi nama nilai B.
2. Data Ratio. Tipe ratio (rasio) didefinisikan sebagai perbandingan antara suatu nilai dengan nilai lainnya, misalkan massa jenis (fisika). Pada tipe ratio terdapat absolute zero (semacam ground truth) yang menjadi acuan, dan absolute zero ini memiliki makna tertentu.

Pada ML, data yang dibutuhkan tergantung dari jenis task yang akan dilakukan. Pada kasus supervised learning, kita akan membutuhkan data yang memiliki label. Label merupakan salah satu fitur dari data yang menjadi target kelompok atau nilai. Proses pemberian label pada data disebut data labeling. Mitchell, T. M. (1997). Machine Learning. McGraw-Hill Education. James, G. , W. D. , H. T. , & T. R. (2013). An Introduction to Statistical Learning.

Sebagai contoh, keputusan untuk bermain (pada kolom play) disebut sebagai label atau kelas (class). Label tersebut menentukan apakah seorang anak akan bermain atau tidak, berdasarkan data pada kolom Outlook, temperature, Humidity dan Windy. Pada unsupervised learning, informasi kolom play tidak diketahui, kita harus mengelompokkan data tersebut sesuai dengan fitur- fitur yang ada.

Table 3.1 Contoh dataset play tenis (sumber:UCI machine learning)

<i>PlayTennis: training examples</i>					
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Outlook	Sunny	1
	Overcast	2
	Rainy	3
Temperature	Hot	1
	Mild	2
	Cold	3
Humidity	High	1
	Normal	2
Windy	Strong	1
	Weak	2
Play Tennis	Yes	1
	No	2

Berdasarkan sumber datanya, data dapat dibagi menjadi 3 kelompok yaitu

- **Data Transaksi.** Data ini bersumber dari kegiatan transaksi setiap hari yang bersumber dari sistem-sistem yang telah dibangun seperti ERP (Enterprise Resource Planning), SCM (Supply Chain Management), CRM (Customer Relationship Management) dan sistem informasi lainnya. Data yang dihasilkan merupakan catatan transaksi antara sistem ke sistem atau pengguna ke sistem. Contohnya data penjualan barang, data pembayaran bank, data peminjaman buku, data pembeli dan lain-lain. Data transaksi cukup sederhana dan cenderung terstruktur dengan baik. Biasanya data-data ini disimpan dengan baik dalam basis data tertentu dan dimanfaatkan sehari-hari.

- **Data Interaksi.** Data interaksi merupakan data penggunaan sistem pada suatu runtutan waktu. Misalnya data bagaimana seseorang memilih sebuah produk mulai dari pencarian sampai pembayaran. Data analisis ini biasanya digunakan untuk menganalisis kebiasaan pengguna terhadap sistem. Contohnya data perilaku pengguna, postingan sosial media, A/B test dan lain-lain.
- **Data Observasi.** Berkembangnya teknologi membuat data observasi ini dimungkinkan saat ini. Sederhananya, data observasi merupakan data yang dihasilkan oleh mesin melalui sensor. Data-data tersebut dapat berupa data sensor angin, penghitung jumlah mobil, click stream, atau data rekaman audio dan video yang pada umumnya merupakan konten yang tidak terstruktur. Kimball, R. , & R. M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. John Wiley & Sons dan Inmon, W. H. (2005). *Building the Data Warehouse*.

Setelah dikumpulkan, data-data tersebut harus disimpan. Mekanisme penyimpanan data ini juga harus menjadi perhatian karena bisa jadi kita akan menghadapi data yang jumlahnya sangat besar sehingga butuh mekanisme penyimpanan yang khusus seperti database, data warehouse dan sebagainya. Pemilihan mekanisme penyimpanan juga berpengaruh kepada performa mesin yang akan dibangun. Sebagai ilustrasi, kita tidak mungkin menyimpan data yang berukuran Gigabyte dalam sebuah file CSV karena proses retrieval-nya akan memakan waktu yang lama.

Selain mengumpulkan data sendiri, kita juga dapat menggunakan dataset-dataset yang telah dikumpulkan pihak lain. Data dari pihak lain dapat kita peroleh dari beberapa sumber, seperti pihak yang menyediakan data secara terbuka sehingga diakses melalui sebuah application programming interface (API) seperti Twitter, Facebook, Instagram dan lain-lain dan data peneliti yang telah mempublikasikan dataset yang telah mereka kumpulkan kepada publik. Russell, M. A. , & K. S. S. (2018).). *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Instagram, GitHub, and More*. Lomborg, S. , & B. A. (2014). *Using APIs for Social Media Research: A Methodology Paper*.

Data tersebut dapat digunakan untuk belajar ML atau sebagai data training untuk model yang akan kita bangun. Adapun tempat favorit penyusun untuk menemukan dataset merupakan

1. Kaggle Datasets

Kaggle merupakan salah satu tempat data set yang bagus. Dataset- dataset tersebut merupakan kontribusi dari komunitas berdasarkan data asli. Selain menyediakan tempat untuk mengunduh dataset, kita juga dapat berdiskusi dan melihat contoh-contoh solusi

pemecahan kasus sehingga sangat direkomendasikan bagi yang sedang belajar ML. Carpenter J. (2011). May the Best Algorithm Win: With sites like Kaggle, companies can crowdsource data problems to an army of experts. Dahlander, L. , & P. H. (n.d.). Open to Suggestions: How organizations elicit ideas through open innovation crowdsourcing. Academy of Management Journal.

2. Amazon Datasets

Dataset Amazon ini menyediakan data dari berbagai topik yang berbeda misalnya transportasi publik, data ekologi, data gen, citra satelit, data perdagangan dan lain-lain. Dataset tersebut disimpan dalam Amazon Web Services (AWS)

3. UCI Machine Learning Repository

Repository dataset ini dikelola oleh University of California, School of Information and Computer Science dengan jumlah dataset lebih dari 100 dataset. Dataset-dataset tersebut telah diklasifikasikan berdasarkan tipe- tipe permasalahan masing learning misalnya kasus klasifikasi dan regresi. Selain itu kita juga dapat menemukan dataset untuk univariate dan multivariate time-series, atau recommendation systems.

4. Google's Datasets Search Engine

Pada akhir 2018, google meluncurkan layanan terbaru untuk yang memungkinkan pengguna mencari dataset. Tujuannya merupakan menyatukan repository-repository yang ada dan membuatnya mudah dicari.

5. Awesome Public Datasets

Ini merupakan kumpulan dataset yang telah dikategorikan berdasarkan topik-topik seperti Biology, Ekonomi, Edukasi dan lain-lain.

6. Computer Vision Datasets

Jika pembaca sedang membuat penelitian dibidang computer vision, deep learning maka dataset ini cocok untuk pembaca. Dataset ini berisi Visual Data yang sangat berguna karena berisi objek-objek seperti Semantic Segmentation, Image captioning, Image Generation bahkan Self-driving cars dataset.

Jika data yang kita butuhkan berupa data yang tersedia di web maka, kita dapat membangun sebuah script untuk melakukan scrapping. Scrapping merupakan sebuah teknik dengan membuat skript yang akan menjelajahi halaman web tertentu lalu mengekstrak informasi yang kita butuhkan. Biasanya cara ini dilakukan karena website tertentu tidak memberikan akses API ataupun memiliki API yang terbatas. Studi kasus 1 dan 2 akan membahas bagaimana melakukan pengumpulan data dengan API dan Scrapping

3.2 Persiapan Data

Biasanya, 80% dari waktu pengembang habis untuk kegiatan ini dan hal tersebut berbanding lurus dengan hasil yang akan didapat.

Semua algoritma ML dapat menemukan pola dalam data, namun apabila algoritma tersebut diberikan data yang “kacau” maka model yang dihasilkan akan memberikan luaran yang kacau juga. Jadi syarat utamanya merupakan data yang bersih, bukan tentang algoritma yang dipakai. Algoritma yang sederhana tetapi dengan data yang bersih dapat mengalahkan algoritma yang rumit dengan data yang kacau.

Pemrosesan data terutama proses pembersihan data, pengembang harus menjadi detektif yang menelusuri data untuk menemukan kesalahan seperti:

- Apakah ada data yang kosong? Kolom yang tidak berisi data? Kenapa data tersebut hilang?
- Bagaimana distribusi data? Bagaimana korelasi antara kolom?
- Menyelidiki nilai-nilai yang aneh dalam data seperti “Tanggal Lahir: Pekanbaru”, “No Telepon: ABCD” dan lain-lain
- Apakah isi data konsisten? Misalnya jenis kelamin bernilai “Laki-Laki”, “Pria”, ”Man”, ”LK” dan sebagainya

Secara keseluruhan, persiapan data terdiri dari tiga kegiatan utama, yaitu membersihkan data, merancang fitur, dan membagi data. Dasar pengelompokan kegiatan tersebut adalah bentuk proses yang dilakukan terhadap data. Pada proses pembersihan data, operasi yang dilakukan adalah operasi perbaikan pada tingkat baris. Rekayasa fitur adalah kegiatan memperbaiki data pada tingkat kolom. Secara umum, proses pembersihan data dilakukan terlebih dahulu, kemudian fitur diatur ulang jika dibutuhkan. Namun dalam beberapa kasus tertentu, bisa dilakukan penyesuaian fitur terlebih dahulu. Jadi, urutan cara melakukannya disesuaikan dengan kasus yang dihadapi.

Kegiatan-kegiatan tersebut dilakukan agar kita bisa mendapatkan data yang baik dan akurat yang benar-benar mencerminkan kasus atau masalah yang sedang dihadapi. Menggunakan data mentah dan membangun model langsung dari data tersebut akan membuat model yang dihasilkan tidak bekerja dengan baik atau tidak seperti yang diharapkan. Hal ini terjadi karena algoritma yang digunakan tidak bisa mengekstraksi fitur secara otomatis. Peran manusia diperlukan untuk memproses data mentah tersebut agar bisa dihasilkan data yang lebih "siap" untuk dipakai.

Kode Pelanggan	NIK	Nama Lengkap	Kelamin	Jumlah Belanja
KD-0001	15122793	Selamet Budi Santoso	Laki-Laki	20000
KD-0002	12685411	Canyono, Agus	LK	20000
KD-0003	15620976	Achmad Raharjo	Pria	50,000
	13400364	Maya Agustina	Wanita	Rp. 50.000
		Cinta Laura	Wanita	50000
	11691235	Raditya Dika		50000
	19074146	Gunawan	Pria	50000
KD-0008	12868040	Tedy Bowo	Pria	50000
KD-0009	15122793	Dr. Selamet Budi Santoso	Pria	50000
KD-0010	15876971	Aisna		50000
KD-0011	18112020			50000
KD-0012	15122793	Selamet B Santoso	Pria	50000

Missing Value/data yang kosong

Format Kelamin Tidak Konsisten

Duplikasi data a.n "Selamet Budi Santoso"

Pembersihan Data

Pembersihan data atau Data Cleaning adalah cara untuk mengenali dan memperbaiki, atau menghilangkan data yang tidak tepat atau salah dalam sebuah dataset, tabel, atau basis data agar kualitas data tersebut menjadi lebih baik. Pada tahap ini, juga dilakukan pengecekan untuk mencari data yang tidak lengkap, salah, tidak tepat, atau tidak sesuai dengan dataset. Perbaikan data bisa dilakukan dengan mengganti, mengubah, atau menghapus informasi yang tidak benar. Pembersihan data dapat dilakukan secara interaktif dengan menggunakan alat bantu data wrangling, atau secara batch melalui skrip pemrosesan data. Gambar 3.4 menunjukkan beberapa masalah yang muncul dalam data, seperti ada data yang berulang, ada data yang tidak terisi, dan data memiliki format yang berbeda-beda.

Setelah pembersihan, dataset diharapkan menjadi konsisten. Inkonsistensi yang terdeteksi atau dihilangkan bisa terjadi karena kesalahan ketika pengguna memasukkan data, kesalahan dalam mengirimkan atau menyimpan data, atau karena definisi entitas yang sama berbeda di berbagai sumber. Tujuan melakukan pembersihan data adalah untuk meningkatkan kualitas data.

Kualitas data dapat dilihat dari beberapa indikator berikut ini:

Validitas

Kita sering mendengar pertanyaan "Datanya valid nggak?". Pertanyaan itu sebenarnya ingin mengetahui apakah data yang dimiliki itu benar. Kebenaran sebuah data dilihat dari seberapa sesuai data tersebut dengan aturan atau batasan bisnis yang sudah ditentukan. Misalnya, data tanggal lahir harus diisi dengan tanggal yang lebih lama atau sama dengan tanggal hari ini.

Dalam dunia komputasi, ada istilah yang disebut Data constraints, yang artinya batasan-batasan yang diterapkan pada data. Batasan data ini bisa digunakan untuk memastikan data tetap valid. Data constraints dibagi menjadi beberapa jenis diantaranya:

1. Data-Type Constraints – Batasan pada tipe data pada kolom penyimpanan data. Misalnya Total Penjualan disimpan dalam bentuk floating point, nama disimpan dalam tipe string, tanggal disimpan pada tipe data date dan lain-lain. Tipe data yang digunakan untuk menyimpan data harus sesuai dengan isi dari data tersebut.
2. Range Constraints. Biasanya batasan ini berlaku untuk angka atau tanggal berada dalam batasan tertentu seperti nilai minimum dan atau maksimum sebuah nilai. Contoh nyatanya merupakan nilai pembayaran tidak boleh < 0 .
3. Mandatory Constraints. Batasan ini memiliki arti kolom tertentu tidak boleh kosong. Misalnya kolom nama memiliki constrain mandatory, ini berarti kolom nama tidak boleh dikosongkan.
4. Unique Constraints: Sebuah atau beberapa field harus memiliki sebuah kolom yang bersifat unik. Contohnya 2 orang tidak boleh memiliki NIK (Nomer Induk Kependudukan) yang sama.
5. Set-Membership constraints: Batasan berupa nilai-nilai sebuah kolom berasal dari sekumpulan data atau kode yang statis. Statis artinya kecil kemungkinan untuk menambah anggota membership-nya. Contohnya jenis kelamin harus diisi Laki-Laki atau Perempuan.
6. Foreign-key constraints: Batasan ini mirip dengan Set-Membership, tetapi nilai data disimpan didalam tabel lain dan dapat bersifat umum serta dinamis. Sebagai contoh kolom kode kecamatan, nilainya terbatas pada isi tabel kecamatan.
7. Regular expression patterns: Batasan ini biasanya ada pada tipe text, dimana sebuah teks harus mengikuti pola-pola penulisan tertentu. Contohnya merupakan Email. Dimana sistem akan menolak setiap nilai yang tidak sesuai dengan format yang ditentukan

Akurasi

Akurasi menunjukkan seberapa tepat suatu nilai dibandingkan dengan nilai standar atau nilai sebenarnya. Mencapai akurasi yang tinggi sangat sulit dalam proses pembersihan data karena biasanya tidak ada standar atau kebenaran yang jelas untuk nilai-nilai data, seperti tanggal lahir, nama, dan sebagainya. Akurasi bisa dicapai dalam proses pembersihan data ketika kita tahu apa yang benar, misalnya nama kota. Jika ada data yang menyebutkan "Jakarte", kita tahu itu salah karena kita punya referensi data yang benar, yaitu "Jakarta".

Kelengkapan

Kelengkapan berarti tingkat di mana semua aspek data sudah diketahui. Ketidaklengkapan hampir tidak bisa diperbaiki dengan cara pembersihan data karena seseorang tidak bisa mengetahui fakta yang tidak tertangkap atau tidak direkam saat proses pengumpulan data dilakukan.

Konsistensi

Konsistensi adalah tingkat di mana semua aspek pengukuran, seperti kolom tertentu, memiliki nilai yang sama atau setara dalam semua sistem. Sebagai contoh, data email pelanggan yang dicatat di aplikasi A harus sama dengan email yang terdapat di aplikasi itu.

B. Memperbaiki data yang tidak konsisten bukanlah hal yang mustahil. Beberapa strategi bisa diterapkan, misalnya mengambil data terbaru sebagai informasi yang akan digunakan atau mencari kebenaran mengenai data tersebut dengan cara menelepon pelanggan.

Keseragaman

Data yang digunakan memiliki satuan pengukuran yang sama di semua sistem. Misalnya, data berat menggunakan kilogram, jadi semua data yang menggunakan satuan berat lainnya, seperti gram dan pound, harus diubah ke kilogram.

Proses pembersihan data dilakukan dalam beberapa tahapan agar dapat menghasilkan data yang berkualitas baik. Masalah-masalah umum yang terdapat pada data adalah nilai yang hilang, data yang duplikat, data yang tidak valid, dan noise.

Tahapan pembersihan data merupakan

1. Inspeksi terhadap data

Proses mencakup mendeteksi data yang tidak diinginkan, tidak konsisten, tidak valid, hubungan antar data, serta mempelajari bagaimana data tersebar. Proses ini adalah proses yang menguras tenaga, membutuhkan waktu yang cukup lama, dan memerlukan berbagai metode untuk mendeteksi kesalahan pada data. Berikut ini adalah beberapa cara yang bisa digunakan untuk menemukan kesalahan dalam data.

2. Data profiling

Proses pembuatan ringkasan statistik dari data bisa disebut sebagai data profiling. Ringkasan tersebut sangat membantu kita untuk mengetahui kualitas data yang ada. Informasi yang ditemukan mencakup hal-hal terkait struktur, isi, serta hubungan antar bagian. Aspek struktur membantu mengetahui apakah data kita teratur dan disusun secara rapi. Aspek isi berfokus pada kualitas data, yaitu format, standar, dan integritas data. Sebagai contoh, alamat email harus memenuhi format email yang umum seperti nama@alamat.com.

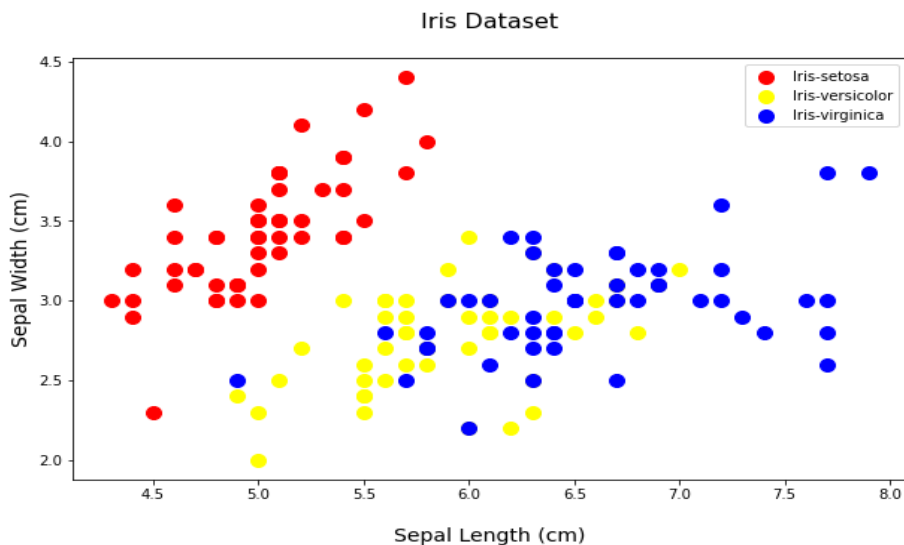
Data profiling dilakukan dengan berbagai teknik statistik deskriptif, seperti rata-rata, nilai minimum, nilai maksimum, percentile, frekuensi, serta fungsi agregat lainnya seperti count dan sum. Informasi tambahan mengenai data, seperti metadata, juga

menjadi output yang sangat penting, misalnya jenis data, ukuran kolom, nilai-nilai diskrit, nilai unik, dan sebagainya. Biasanya di tahap ini dilakukan analisis apakah data sudah lengkap (Analisis Kelengkapan Data) untuk mengetahui berapa banyak atribut yang sudah diisi, yang kosong, atau bernilai nol; serta analisis apakah data memiliki nilai unik (Analisis Keunikan Data) untuk mengetahui berapa banyak kolom yang memiliki nilai berbeda (distinct) untuk setiap atributnya. Apakah ada duplikat? Apakah ada atribut unik yang harus ada? Analisis distribusi nilai (Values Distribution Analysis) digunakan untuk mengetahui bagaimana data tersebut terdistribusi; analisis rentang (Range Analysis) bertujuan untuk menemukan nilai terkecil, terbesar, rata-rata, dan median dari atribut data; sehingga melalui analisis-analisis tersebut, kita dapat menemukan beberapa informasi penting, misalnya jenis tipe data yang digunakan. Berapa banyak data yang hilang? Bagaimana distribusi data? Apakah ada relasi antar data?

3. Visualisasi Data

Dengan menganalisis dan menggambarkan data, informasi yang terkandung dalam data bisa dilihat secara visual. Visualisasi bisa dilakukan dengan menggunakan gambar seperti scatter plot, histogram, dotplot, boxplot, dan masih banyak lagi.

Grafik titik biasanya digunakan untuk menampilkan hubungan antara dua variabel atau atribut berupa angka. Misalnya, lihatlah Gambar 3.5, kita bisa melihat bahwa variabel pengalaman kerja diduga memengaruhi variabel lainnya yaitu gaji. Dengan menempatkan variabel pengalaman sebagai variabel penjelas pada sumbu x dan variabel gaji sebagai variabel respons pada sumbu y, kita bisa melihat hubungan antara kedua variabel tersebut serta mengetahui adanya pencilan atau outlier yang mungkin ada. Dengan menampilkan visualisasi tersebut, kita bisa mendapatkan informasi baru yang berguna, sehingga dapat dipakai untuk memperbaiki atau meningkatkan ketepatan model.



Gambar 3.4 Visualisasi dalam bentuk Scatter Plot

Contoh lain visualisasi yang sering digunakan merupakan histogram. Histogram merupakan salah satu cara yang bagus untuk memvisualisasikan distribusi variabel numerik. Data dikelompokkan pada interval tertentu dan tinggi bar merepresentasikan jumlah item per interval. Selain itu, histogram juga dapat digunakan untuk meidentifikasi bentuk distribusi. Gambar 3.6 merupakan contoh histogram. Distribusi data life expectancies kelihatan condong kesebelah kiri (left skewed) diantara umur 60 sampai 90 tahun.



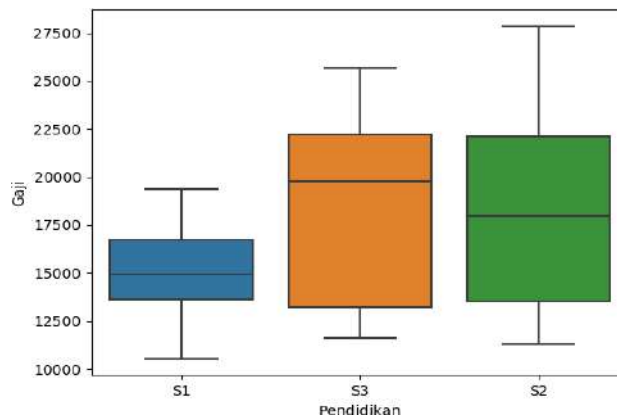
Gambar 3.5 Visualisasi dalam bentuk Histogram

Selain itu, kita dapat menggunakan Boxplot. Boxplot merupakan ringkasan distribusi sampel yang disajikan secara grafis yang bisa menggambarkan bentuk distribusi data (skewness), ukuran tendensi sentral dan ukuran penyebaran (keragaman) data pengamatan. Terdapat 5 ukuran statistik yang bisa kita baca dari boxplot, yaitu:

- nilai minimum: nilai observasi terkecil

- Q1: kuartil terendah atau kuartil pertama
- Q2: median atau nilai pertengahan
- Q3: kuartil tertinggi atau kuartil ketiga
- nilai maksimum: nilai observasi terbesar.

Selain itu, boxplot juga dapat menunjukkan ada tidaknya nilai outlier/pencilan dan nilai ekstrim dari data pengamatan. Bagian utama boxplot merupakan kotak berbentuk persegi (Box) yang merupakan bidang yang menyajikan Interquartile range (IQR), dimana 50 % dari nilai data pengamatan terletak di sana. Panjang kotak sesuai dengan jangkauan kuartil dalam (inner Quartile Range, IQR) yang merupakan selisih antara Kuartil ketiga (Q3) dengan Kuartil pertama (Q1). IQR menggambarkan ukuran penyebaran data. Semakin panjang bidang IQR menunjukkan data semakin menyebar. Gambar 3.7 menunjukkan contoh grafik boxplot yang menggambarkan kondisi gaji berdasarkan jenjang pendidikan.



Gambar 3.6 Visualisasi Menggunakan Boxplot

Berdasarkan Gambar 3.7, rata-rata Pendidikan S1 memiliki gaji 15.000, S2 memiliki rata-rata gaji 175.00 dan S3 memiliki rata-rata gaji 20.000.

Langkah 2. Pembersihan data / Data Cleaning

Memperbaiki anomaly yang ditemukan pada proses inspeksi. Pembersihan data dilakukan dengan berbagai cara berdasarkan permasalahan yang dihadapi dan tipe data. Secara umum aktifitas yang dilakukan pada pembersihan data adalah penghapusan, perbaikan, atau penghitungan ulang. Berikut ini merupakan kondisi-kondisi dimana data perlu dibersihkan

4. Data yang tidak relevan

Data yang tidak relevan dapat dikatakan data yang tidak dibutuhkan atau tidak cocok untuk konteks permasalahan yang dihadapi. Sebagai contoh, ketika kita ingin membuat model yang dapat menentukan apakah seorang pasien terkena kanker atau

tidak maka atribut nomer telepon pasien tidak dibutuhkan dan tidak relevan dengan kasus yang akan diselesaikan. Pada kasus ini, kita harus membuang kolom/atribut/fitur nomer telepon karena tidak dibutuhkan.

Proses pemilihan kolom tersebut feature selection atau seleksi fitur. Proses seleksi fitur ini dapat dilakukan secara manual (dengan mengetahui konteks permasalahan) dan secara otomatis menggunakan algoritma tertentu. Beberapa algoritma yang dapat digunakan merupakan Univariate Selection, Recursive Feature Elimination, Principal Component Analysis, Feature Importance, dan lain-lain. Beberapa algoritma berbasis pohon keputusan dan neural network telah melakukan proses ini secara otomatis sebagai bagian dari algoritma seperti pada Algoritma Random Forest.

Pembersihan data yang tidak relevan tidak hanya dilakukan pada level kolom atau atribut, tetapi juga dapat dilakukan pada level baris. Sebagai contoh, pembaca ingin membangun sebuah model dapat memprediksi kebakaran hutan di daerah Provinsi Riau maka pembaca harus menghapus data yang berada di luar area Provinsi Riau karena data tersebut juga tidak relevan dengan model yang akan dibangun.

Keuntungan melakukan seleksi fitur merupakan mengurangi Overfitting dan meningkatkan akurasi karena dengan berkurangnya data yang tidak relevan berarti mengurangi kemungkinan untuk membuat model berdasarkan data kotor/noise, sehingga model yang dihasilkan lebih akurat. Selain itu, seleksi fitur juga mengurangi waktu pelatihan. Lebih sedikit data yang diproses akan mengurangi kompleksitas dan waktu pelatihan akan lebih cepat.

5. Data yang terduplikasi

Duplikasi data banyak terjadi di dunia nyata. Data yang terduplikasi merupakan data yang sama atau berulang pada dataset. Penyebab duplikasi ada banyak misalnya karena data dikumpulkan dari berbagai sumber data atau pada saat pengisian menekan tombol simpan berkali-kali. Untuk mengatasi data yang terduplikasi harus ditentukan identitas data yang terduplikasi dan menghapus data yang terduplikasi tersebut.

6. Konversi Tipe Data

Pastikan data angka disimpan dalam tipe data numerik, data tanggal disimpan dalam tipe data date atau timestamp dan seterusnya. Tipe data ini telah terdeteksi pada tahapan pertama (ketika inspeksi). Jika dibutuhkan data yang bersifat kategori dapat dikonversi menjadi bentuk angka.

7. Perbaikan dan Modifikasi Data

Banyak kemungkinan kesalahan yang terjadi pada data misalnya salah ketik, data yang tidak konsisten dan lain-lain. Berikut ini beberapa kesalahan yang umum terjadi pada data:

Tambahan White Space. Istilah white space merupakan karakter-karakter yang tidak kelihatan pada data seperti spasi dan tab. Data dengan nilai “ Halo ” (ada spasi di depan dan di belakang) tidak sama dengan data “Halo” tanpa spasi. Jadi jika pembaca memiliki misalnya data yang sensitif maka perlu melakukan penghilangan whitespace. Berikut merupakan contoh penghilangan whitespace pada Python

Source Code - Implementasi Pembersihan White Space	
1.	data = ' Saya memiliki white space ' print(data) print('Data sesudah') print(data.strip())
2.	
3.	
4.	
Keluaran	
Saya memiliki white space Data sesudah Saya memiliki white space	

Pad strings:

Untuk tujuan mempermudah pembacaan biasanya angka- angka tertentu diberikan awalan 0 (nol) seperti data bulan Januari biasanya ditulis 01. Jika kita membutuhkan data dalam bentuk Integer maka perlu menghilangkan angka nol di depan dan mengkonversi tipe data.

Source Code 1. Implementasi Pembersihan Padded String	
1. 2. 3. 4.	<code>data = '00001'</code> <code>print(data)</code> <code>print('Data sesudah')</code> <code>print(data.strip())</code>
Keluaran	
<code>00001</code> <code>Data sesudah</code> <code>1</code>	

Salah ketik:

Salah ketik merupakan kesalahan yang paling umum karena data string dapat diisi dalam berbagai jenis. Sebagai contoh jenis kelamin laki-laki dapat ditulis dengan banyak kemungkinan LK, Laki-laki, Laki- Laki, LAKI, LAKI-LAKI, Lelaki, leki-leki dan lain-lain. Untuk menghadapi masalah ini langkah pertama merupakan membuat sebuah barplot yang berisi semua nilai unik dari kelamin laki-laki. Selanjutnya baru melakukan replace terhadap pola-pola tersebut.

8. Standarisasi data

Selain pembersihan, standarisasi data perlu dilakukan untuk menjamin bahwa data yang akan digunakan terdiri atas format yang standar. Misalnya untuk data string harus dipastikan semua nilainya berjenis huruf kecil atau huruf besar semua. Sedangkan untuk data numerik, pastikan data-data tersebut memiliki satuan ukur yang sama misalnya untuk Panjang semua data disimpan dalam centimeter semua atau inci semua. Selain itu pastikan juga untuk data tanggal memiliki format yang sama misal YYYY-MM-DD atau DD/MM/YYYY.

9. Perbaikan Missing data

Faktanya, data yang didapat sering mengalami missing value atau data yang tidak lengkap. Missing data atau data yang kosong pada sebuah dataset yang berasal dari kejadian yang bersifat acak. Missing data perlu ditangani dengan baik karena beberapa algoritma tidak bisa bekerja jika pada dataset berisi missing data. Berikut ini beberapa pilihan yang dapat dilakukan jika pada dataset memiliki missing data diantaranya:

- Hapus data yang memiliki missing data. Penghapusan ini dilakukan jika data tersebut jumlahnya tidak banyak. Sebagian literatur mengatakan bahwa jika jumlah data yang hilang kurang dari 2% maka opsi ini dapat dilakukan.
- Mengganti nilai yang hilang dengan nilai rata-rata atau median atau modus.
- Menggunakan metode-metode regresi atau ML untuk memprediksi nilai yang hilang.

10. Transformasi data

Transformasi Data merupakan upaya yang dilakukan dengan tujuan utama untuk mengubah skala pengukuran data asli menjadi bentuk lain sehingga data dapat memenuhi asumsi-asumsi yang mendasari analisis ragam. Transformasi data ada beberapa jenis, antara lain:

- Transformasi Square Root (Akar),
- Transformasi Logaritma,
- Transformasi Arcsin,

- Transformasi Square (Kuadrat),
- Transformasi Cubic (Pangkat Tiga),
- Transformasi Inverse (Kebalikan),
- Transformasi Inverse Square Root (Kebalikan Akar),
- Transformasi Inverse Square (Kebalikan Kuadrat),
- Transformasi Inverse Cubic (Kebalikan Pangkat Tiga),
- Transformasi Reverse Score (Balik Skor).

11. Normalisasi data

Tujuan utama dari normalisasi merupakan mengubah nilai numerik dari sebuah fitur pada dataset menjadi sebuah nilai yang lebih umum. Perubahan tersebut biasanya dilakukan pada kolom yang memiliki jangkauan nilai yang berbeda, namun perubahan nilai tersebut tidak mengubah makna dari data. Tujuannya merupakan agar model yang dihasilkan lebih stabil. Manfaat yang didapat merupakan data menjadi Comparable. Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems. O'Reilly Media, Inc. Han, J., Kamber, M., & Pei, J. (2011). Data Mining. Concepts and Techniques, 3rd Edition (The Morgan Kaufmann Series in Data Management Systems).

Misalnya seorang pembeli melakukan pembelian sebesar 3.000.000 untuk 6 item barang. Dengan menormalisasikan nilainya, maka kita dapat membandingkan kedua kolom tersebut. Nilai pembelian menjadi 0.8 dan jumlah item menjadi 0.2. Masing-masing kolom tersebut memiliki nilai minimal 0 dan maksimal 1, jadi transaksi tadi merupakan transaksi besar, namun jumlah pembelian sedikit.

Selanjutnya, data yang ternormalisasi mempercepat algoritma untuk konvergen sehingga waktu pelatihan dapat dikurangi. Dan bagi algoritma- algoritma tertentu yang membutuhkan distribusi normal maka hal ini dapat membantu algoritma bekerja lebih baik. Beberapa teknik yang umum digunakan merupakan scaling, clipping dan Z-Score.

12. Rekayasa Fitur

Rekayasa fitur atau Feature Engineering merupakan proses perubahan dataset dimana dilakukan proses ekstraksi sebuah kolom menjadi kolom-kolom lainnya atau proses mereduksi kolom.

Fitur pada ML diasosiasikan dengan kolom pada dataset. Sebuah kolom dapat memiliki tipe-tipe data tertentu. Sebuah fitur merupakan sebuah atribut data yang berguna atau memiliki arti penting terhadap masalah yang akan diselesaikan. Sebagai contoh jumlah waktu belajar siswa merupakan atribut penting ketika kita ingin memprediksi kelulusan seorang siswa. Namun atribut nama siswa berbeda, ia tidak memberikan pengaruh kepada kelulusan siswa. Oleh karena itu pemilihan fitur yang

tepat sangat mempengaruhi hasil atau performa model. Pemilihan fitur ini merupakan salah satu kegiatan yang dilakukan dalam rekayasa fitur.

Berikut ini kriteria-kriteria sebuah fitur yang baik:

1. Informatif. Sebuah fitur harus memberikan informasi yang baik kepada penggunaannya.
2. Ketersediaan Data, jika fitur ini datanya banyak yang kosong. Jika pembaca memilih fitur yang banyak kosong maka pembaca akan berurusan dengan missing data.
3. Diskriminan. Dengan fitur ini diharapkan dapat membagi data menjadi target kelas yang dituju atau berkorelasi dengan label.

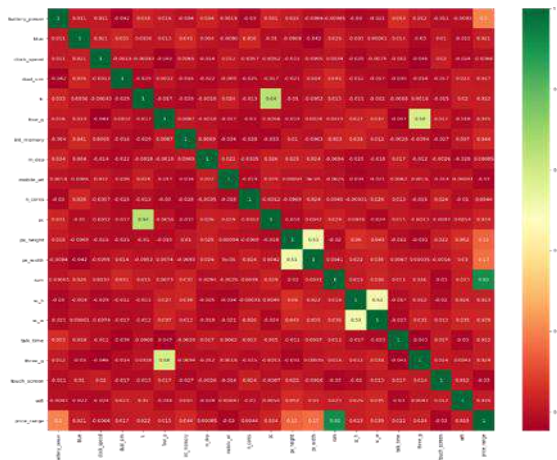
Beberapa kegiatan yang dapat dilakukan dalam rekayasa fitur:

FEATURE SELECTION

Pada sebuah dataset, tidak semua fitur berguna dalam pengerjaan task tertentu. Fitur-fitur yang tidak relevan terhadap kasus sebaiknya di buang karena kehadirannya dapat mengganggu performa model.

Berikut ini metode-metode yang sering digunakan oleh praktisi dalam melakukan seleksi fitur diantaranya :

- Correlation Matrix. Secara matematis, sebuah fitur dikatakan penting jika memiliki korelasi yang tinggi dengan variabel dependent. Oleh karena itu, koefisien korelasi merupakan metode yang umum digunakan untuk menentukan seberapa penting sebuah fitur.
- Permutation feature importance. Pada metode ini didasarkan pada intuisi bahwa jika fitur tidak/kurang penting, maka mengubah atau mengganti atau menukar nilainya tidak akan menghasilkan penurunan yang signifikan dalam kinerja model. Namun, jika fitur penting untuk memprediksi hasil, maka perubahan tersebut akan memiliki efek yang cukup signifikan
- Drop feature importance. Metode ini mirip dengan Permutation feature importance, namun pada kolom ini bekerja dengan menghapus kolom lalu melihat efek dari penghapusan tersebut. Jika mempengaruhi hasil maka kolom dapat dinyatakan penting.



Gambar 3.7 Contoh Correlation matrix

- Feature Importance berbasis nilai SHAP (SHapley Additive exPlanations). Nilai SHAP merupakan sebuah metode yang sangat luar biasa untuk melakukan “reverse-engineering” terhadap luaran sebuah predictive model. Sebagai contoh pembaca memprediksi harga sebuah apartemen merupakan 3 miliar dengan nilai fitur, luas apartemen 50 m2, dekat dengan stasiun, dan tidak boleh memelihara hewan peliharaan. Pembaca juga di berikan informasi bahwa rata-rata harga apartemen merupakan 3.5 miliar. Maka dengan menggunakan SHAP, pembaca dapat menentukan berapa kontribusi dari masing-masing fitur terhadap 3 miliar tersebut.

FEATURE EXTRACTION

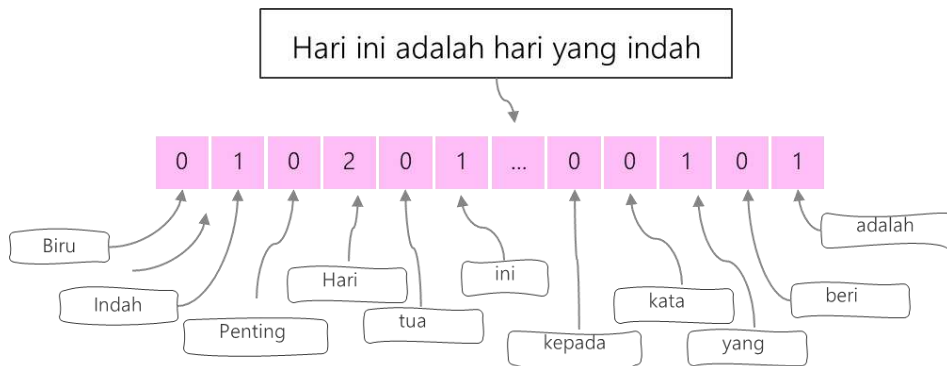
Feature extraction merupakan sebuah proses pembentukan fitur baru atau mengurangi dimensi data dari data mentah secara otomatis. Feature extraction biasanya dilakukan pada data tabular berdimensi tinggi, gambar, suara, dan text.

Pada data-data yang bersifat tabular (dalam bentuk tabel), biasanya data- data tersebut memiliki attribut yang sangat banyak atau dimensi tinggi. Attribut yang sangat banyak tersebut dapat dikurangi secara otomatis menggunakan algoritma tertentu seperti Principal Component Analysis, Singular Value Decomposition, Linear Discriminant Analysis, Locally Linear Embedding dan lain-lain. Kunci dari feature extraction merupakan penggunaan metode yang otomatis.

Berikut ini contoh-contoh kegiatan feature extration:

- Bag-of-Words (BOW) merupakan teknik yang paling banyak digunakan untuk ekstraksi fitur pada kasus natural language processing. Sederhananya, BOW merupakan teknik yang mengubah data yang berupa teks menjadi bentuk vektor sehingga dapat dilakukan pemrosesan. Sebagai contoh sebuah kalimat “Hari ini

merupakan hari yang indah” akan diekstrak menjadi sebuah vektor yang berisi jumlah kata. Sebagai contoh pada gambar 3.9, pada awal vektor bernilai 0 karena pada kalimat tidak ada kata biru, sedangkan item yang kedua bernilai 1 karena ada 1 kata indah.



Gambar 3.8 Teknik yang mengubah data yang berupa teks menjadi bentuk vektor

- Ekstraksi tepi pada gambar (Image Processing). Pemrosesan gambar merupakan salah satu domain yang banyak memanfaatkan ekstraksi fitur. Contoh operasi yang dilakukan merupakan ekstraksi tepi, bentuk, atau gerak pada gambar.
- Ekstraksi Mel Spektrogram pada audio (Audio Processing) dengan mengubah audio yang berupa frekwensi dalam waktu tertentu menjadi mel spectrogr

FEATURE CONSTRUCTION

Feature Construction atau pembentukan fitur baru dari raw data (data mentah) dilakukan secara manual. Fitur hasil dari kegiatan ini dibuat langsung oleh pengembang menggunakan tool dan pengetahuan pengembang. Kegiatan ini biasanya dibantu oleh informasi yang didapat dari feature important atau kebutuhan input model. Karena proses pembuatannya manual, kadang-kadang aktivitas ini memerlukan waktu dan pengamatan masalah secara teliti agar fitur yang dikembangkan dapat bermanfaat. Kegiatan feature construction ini sangat tergantung kepada jenis data, algoritma dan task yang akan diselesaikan.

Pada data tabular, biasanya dilakukan proses penggabungan atau pemisahan data pada kolom-kolom tertentu sehingga tercipta kolom baru. Contoh kegiatan ini merupakan Categorical Encoding seperti One hot encoding, Count and Frequency encoding, Target encoding / Mean encoding, Ordinal encoding, Weight of Evidence dan lain-lain.

Selanjutnya, pada data yang berbentuk gambar kegiatan ini bervariasi tergantung permasalahan yang dihadapi. Contohnya kita akan membuat sebuah sistem yang

mendeteksi lokasi sel kanker/rusak dari sekumpulan sel. Maka kegiatan untuk melakukan anotasi berupa koordinat lokasi merupakan kegiatan Feature Construction.

Jadi kegiatan ini bersifat manual dan butuh keahlian spesifik tetapi data yang dihasilkan akan sangat bermanfaat bagi model.

FEATURE LEARNING

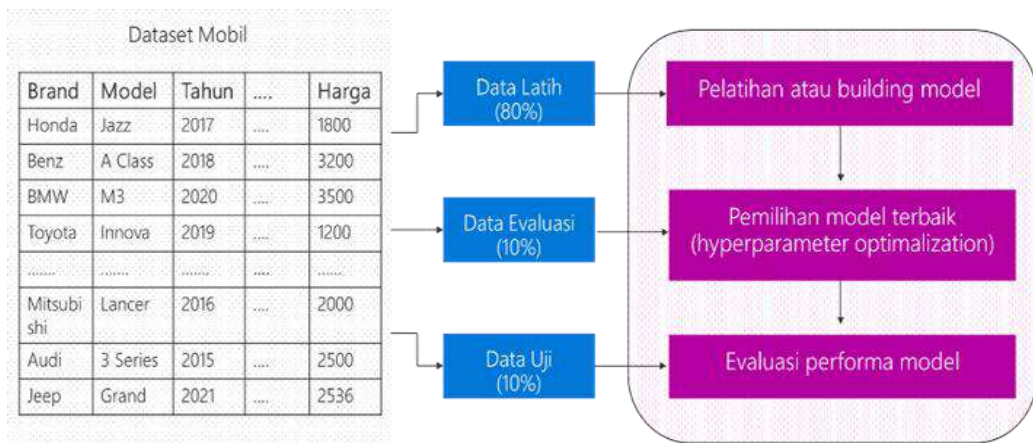
Feature learning merupakan proses identifikasi dan penggunaan fitur secara otomatis dari data mentah. Biasanya feature learning digunakan pada metode-metode Representation learning atau Deep learning. Contohnya merupakan autoencoders dan restricted Boltzmann machines. Algoritma ini telah menunjukkan kemampuannya dalam men-generate fitur secara baik dan mempelajari abstract representations dari fitur-fitur tersebut. Kasus-kasus seperti speech recognition, image classification, object recognition dan lainnya merupakan area yang tepat untuk menggunakan metode ini.

DATA SPLITTING

Setelah mendapatkan data yang bersih, selanjutnya merupakan melakukan proses data splitting (pemecahan data) dimana kita membagi data tersebut menjadi beberapa bagian diantaranya data latih (training data), data uji (testing data), dan data validasi (validation data). Metode ini dikenal dengan nama holdout dimana kita membagi dataset menjadi 2 bagian yaitu training set dan holdout set. Training set berisi data latih yang digunakan untuk membangun model dan biasanya ukurannya lebih besar dibandingkan dengan holdout set.

Sedangkan holdout set dapat terdiri atas data uji (testing data), dan data validasi (validation data) atau hanya data uji saja. Dengan kata lain kita dapat memecah dataset tersebut menjadi 2 atau 3 tergantung kebutuhan. Data validasi bersifat opsional dan hanya dibutuhkan ketika kita ingin melakukan optimalisasi hyperparameter model yang telah dilatih atau pemilihan algoritma terbaik. Sebagai contoh, pada algoritma yang berbasis jaringan syaraf tiruan, dimana kita perlu menentukan hyperparameter seperti berapa jumlah hidden unit, jumlah layer dan sebagainya. Untuk menentukan hyperparameter tersebut maka diperlukan sebuah dataset yang tidak bias sehingga menghasilkan hyperparameter yang optimal. Namun jika pada pelatihan pembaca telah menetapkan hyperparameter model maka data evaluasi biasanya tidak dibutuhkan. Selanjutnya data uji digunakan untuk mengukur performa model yang telah dilatih.

Sebagai contoh pada Gambar 3.10, dimana kita memiliki sebuah dataset mobil yang terdiri atas 1.000 data jenis mobil. Untuk melakukan pelatihan model maka data tersebut akan dipecah menjadi 3 bagian. Data Latih terdiri atas 800 data, Data evaluasi 100 data dan Data Uji 100 data.



Gambar 3.8 Ilustrasi holdout sets

Tidak ada aturan yang pasti tentang berapa porsi pembagian masing masing jenis data. Jika kita memiliki dataset yang besar, misalnya 500.000 sample maka kita dapat menggunakan 95% untuk data latih dan 2.5% untuk validasi, dan 2.5% untuk data uji. Namun jika kita hanya memiliki 100 data kita dapat memecahnya menjadi 70+15+15.

Kenapa dataset harus dipecah? Bayangkan jika pembaca melakukan ujian dengan soal yang pernah pembaca kerjakan maka pembaca akan mendapatkan nilai yang tinggi. Namun ujian tersebut tidak menggambarkan kemampuan pembaca yang sebenarnya. Oleh karena itu soal yang digunakan pada saat pembelajaran dan ujian harus berbeda, namun dari sumber yang sama.

Untuk melakukan split terhadap data di python menggunakan Numpy maka dapat dilihat pada contoh berikut menggunakan np.split

```

1 import numpy as np
2 from numpy import loadtxt
3 my_data = loadtxt('data/diabetes.csv', delimiter=',', skiprows=1)
4 train, validate, test = np.split(my_data, [int(.6*len(my_data)), int(.8*len(my_data))])
5 print("Dataset : ",len(my_data))
6 print("Train Dataset : ",len(train))
7 print("Validation Dataset : ",len(validate))
8 print("Test Dataset : ",len(test))

```

```

Dataset : 768
Train Dataset : 460
Validation Dataset : 154
Test Dataset : 154

```

Untuk melakukan split terhadap data di python menggunakan Pandas maka dapat dilihat pada contoh berikut:

```

1 import numpy as np
2 import pandas as pd
3
4 #load dataset menggunakan pandas
5 df = pd.read_csv('data/diabetes.csv')
6
7 def train_validate_test_split(df, train_percent=.6, validate_percent=.2, seed=None):
8     np.random.seed(seed)
9     perm = np.random.permutation(df.index)
10    m = len(df.index)
11    train_end = int(train_percent * m)
12    validate_end = int(validate_percent * m) + train_end
13    train = df.iloc[perm[:train_end]]
14    validate = df.iloc[perm[train_end:validate_end]]
15    test = df.iloc[perm[validate_end:]]
16    return train, validate, test
17
18 train, validate, test = train_validate_test_split(df)

```

Untuk melakukan split terhadap data di python menggunakan sk-Learn maka dapat dilihat pada contoh berikut menggunakan fungsi `train_test_split`:

```

1 from sklearn.datasets import load_iris
2 from sklearn.model_selection import train_test_split
3
4 #load data iris
5 iris = load_iris()
6 X, y = iris.data, iris.target
7
8 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1)
9 X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.15)
10 # 0.25 x 0.8 = 0.2

```

Manfaat lainnya yang didapatkan ketika membagi data-data ini merupakan kita dapat melakukan deteksi terhadap dua masalah yang umum terjadi pada saat pembelajaran yaitu underfitting dan overfitting James, Gareth., Witten, Daniela., Hastie, Trevor., & Tibshirani, Robert. (2017). *An introduction to statistical learning : with applications in R*. Springer : Springer Science+Business Media. Jordan, M., Kleinberg, J., & Schölkopf, B. (n.d.). *Pattern Recognition and Machine Learning*.

- Underfitting merupakan sebuah situasi dimana model yang dihasilkan gagal untuk melakukan generalisasi relasi pada data. Sebagai contoh misalnya ketika kita melakukan training pada data yang non-linear menggunakan metode linear yang mengakibatkan performa dari training dan test rendah. Umumnya masalah underfitting diidentifikasi dengan rendahnya performa training dan test sets.

- Overfitting merupakan sebuah situasi dimana model yang dihasilkan menghasilkan generalisasi yang kompleks dari data training. Analoginya, model yang dihasilkan “menghapal” relasi pada data sehingga ketika dievaluasi menggunakan test data maka performanya akan turun. Masalah ini diidentifikasi dengan tingginya performa pada training namun performa yang rendah pada test. Idealnya performa antara training dan test tidak jauh berbeda.

Metode Cross-Validation

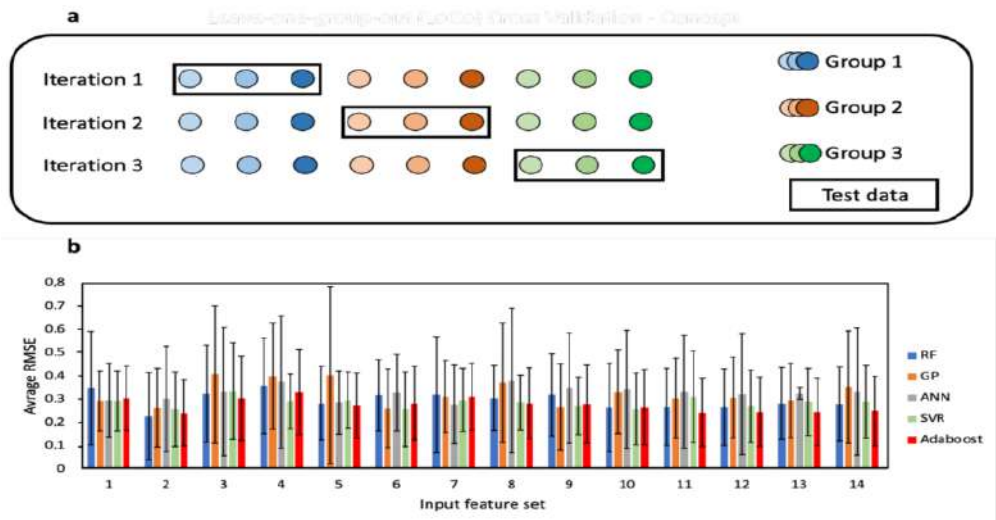
Cross-validation (CV) merupakan metode statistik yang dapat digunakan untuk mengevaluasi kinerja model atau algoritma dimana data dipisahkan menjadi dua subset yaitu data latih dan data uji. Model atau algoritma dilatih oleh subset pembelajaran dan divalidasi oleh subset validasi. Selanjutnya pemilihan jenis CV dapat didasarkan pada ukuran dataset. Hastie, T. , T. R. , & F. J. (2009). *The Elements of Statistical Learning*. James, G. , W. D. , H. T. , & T. R. (2013). *An Introduction to Statistical Learning (ISL)*.

Untuk mengurangi variabilitas, maka dilakukan beberapa putaran validasi silang dengan subset berbeda dari data yang sama. Lalu hasil evaluasi tersebut digabungkan (rata-rata). Metode CV akan memberi kami perkiraan performa model yang lebih akurat dibandingkan Holdout.

Berikut ini beberapa jenis CV diantaranya :

1. LOOCV (Leave one out cross-validation)

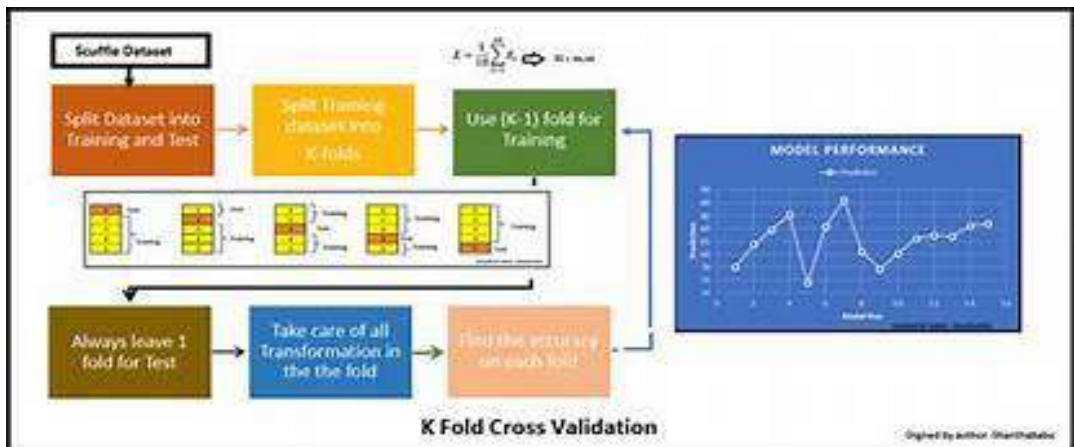
Pada LOOCV, data dibagi menjadi dua bagian, yaitu data uji dan data latih, namun pada data latih hanya menggunakan 1 data. Sehingga jika kita memiliki n data, maka kita akan memiliki sejumlah $n-1$ data latih dan 1 data uji. Proses training dan evaluasi dilakukan sebanyak n kali. Ilustrasi LOCV dapat dilihat pada Gambar [bookmark57](#)



Gambar 3.9 Ilustrasi Leave one out cross-validation

2. K-Fold cross-validation

K-fold merupakan sebuah metode yang memecah dataset menjadi dua bagian yaitu data latih dan data uji sebanyak k kelompok dimana jumlah data latih dan data uji pada tiap-tiap kelompok sama. Sehingga kita akan melakukan perulangan sebanyak K kali utk melakukan training dan evaluasi. K dapat berisi sebuah nilai bulat. Jika k bernilai 1 maka metode ini sama dengan holdout. Biasanya nilai k lebih besar dari 1. Ilustrasi K-Fold ditunjukkan pada Gambar



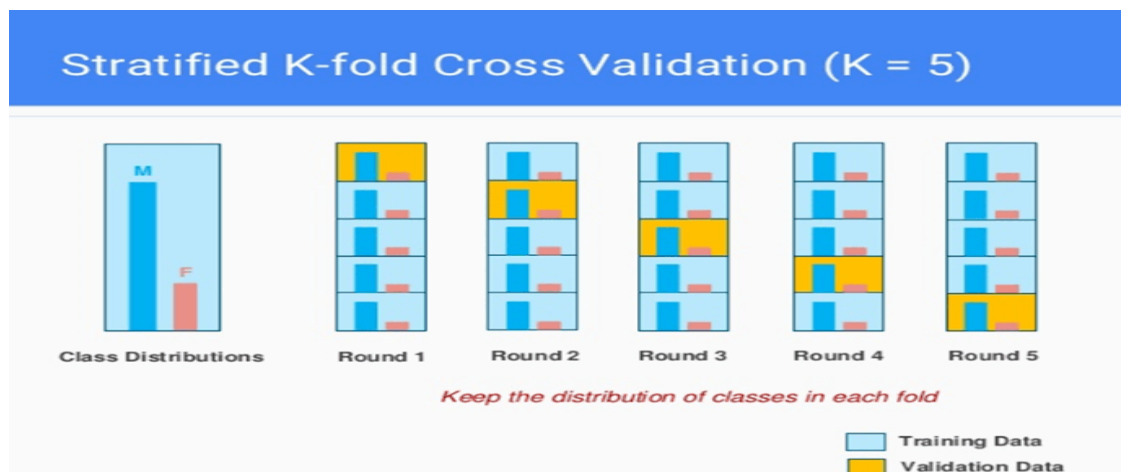
Gambar 3.10 Ilustrasi K-Fold cross-validation

10 fold CV merupakan salah satu K-fold CV yang direkomendasikan untuk pemilihan model terbaik karena cenderung memberikan estimasi akurasi yang kurang bias dibandingkan dengan CV biasa dan leave-one-out CV. Dalam 10 fold CV, data

dibagi menjadi 10 fold berukuran kira-kira sama, sehingga kita memiliki 10 subset data untuk mengevaluasi kinerja model atau algoritma. Untuk masing-masing dari 10 subset data tersebut, CV akan menggunakan 9 fold untuk pelatihan dan 1 fold untuk pengujian.

3. Stratified cross-validation

Stratified cross-validation mirip dengan K-Fold, namun perbedaannya dalam pembagian data tiap iterasinya memperhatikan kondisi data set seperti distribusi kelas, rata-rata dan varian, sehingga pembagian distribusi kelas lebih merata. Perhatikan Gambar sebagai ilustrasi, dimana pada gambar tersebut distribusi kelas M dan L hampir sama pada tiap bagian data.



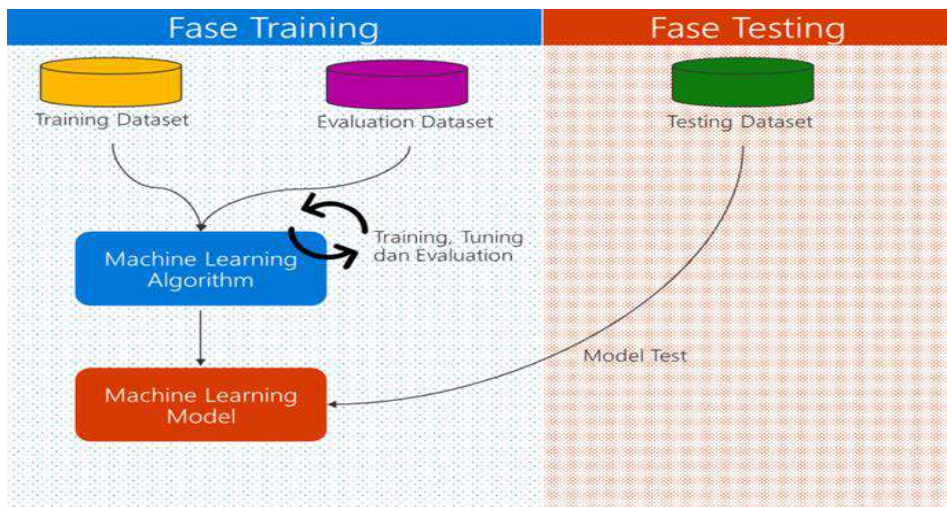
Gambar 3.11 Ilustrasi Stratified cross-validation

3.3 Pelatihan dan Uji Coba Model

Proses pelatihan model (model training) merupakan sebuah proses mengekstrak pola atau pengetahuan pada data latih menggunakan algoritma tertentu. Berbagai jenis algoritma dapat digunakan pada tahap ini tergantung task atau masalah apa yang ingin dipecahkan. Hasil atau luaran pelatihan menggunakan algoritma tertentu merupakan sebuah model.

Machine Learning.

Pada proses fase pelatihan, digunakan dua jenis dataset yaitu training dan validation dataset. Algoritma ML memanfaatkan training dataset yang telah disiapkan pada fase sebelumnya untuk mengekstrak pola atau pengetahuan pada data. Sedangkan Validation dataset, seting digunakan untuk memvalidasi model dan melakukan hyperparameter optimization.

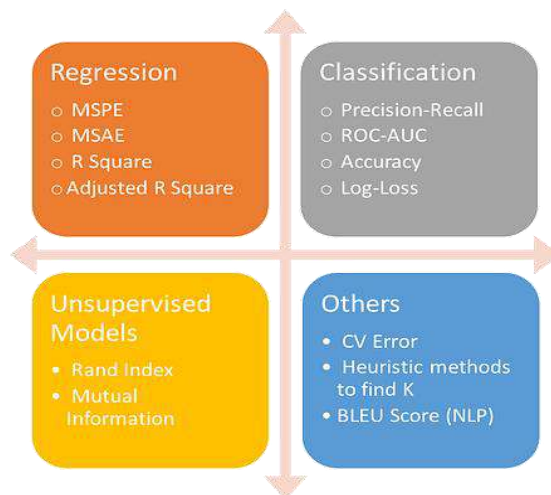


Gambar 3.12 Proses Ilustrasi Pelatihan dan Pengujian Model

Pada tahap training juga biasanya dilakukan hyperparameter optimalization (Tuning) dan evaluation. Hyperparameter merupakan parameter yang harus ditetapkan sebelum proses pembelajaran berlangsung. Hyperparameter ini berbeda-beda tiap algoritma dan nilainya mempengaruhi performa dan dipengaruhi oleh masalah dan data yang ingin dipecahkan. Oleh karena itu, biasanya pada proses ini dilakukan berulang-ulang sehingga diperoleh beberapa model lalu dipilih model yang terbaik. Untuk detail proses ini, pembaca dapat merujuk pada bab selanjutnya tergantung algoritma apa yang akan digunakan.

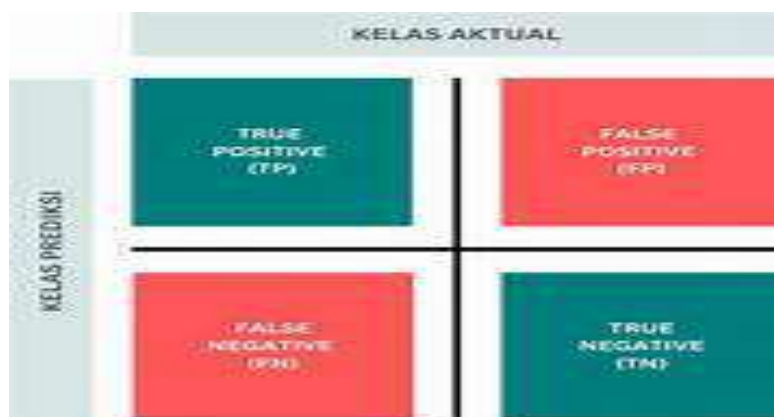
Evaluasi model

Evaluasi model merupakan sebuah proses untuk mengukur performa model yang telah dihasilkan pada proses training. Ada banyak metode untuk mengevaluasi model, biasanya tergantung kepada jenis task yang dilakukan serta informasi apa yang ingin diketahui. Sebagai contoh pada kasus regresi dan klasifikasi, membutuhkan metode yang berbeda karena ukuran baik atau tidaknya model regresi dan klasifikasi berbeda.



Gambar 3.13 Jenis-Jenis Metrik Evaluasi Model berdasarkan task

Adapun beberapa metode atau metric yang umum digunakan merupakan confusion Matrix. Confusion matrik merupakan matrik yang berukuran $N \times N$ dimana N merupakan jumlah kelas yang diprediksi. Jadi metric ini cocok digunakan untuk permasalahan klasifikasi. Confusion matrix menyajikan ringkasan semua hasil prediksi yang dihasilkan dengan membandingkan antara hasil prediksi dan hasil yang diharapkan.



Gambar 3.14 Contoh confusion matrix dengan dua kelas yaitu yes dan no

Sebagai contoh perhatikan Gambar 3.16 yang menggambarkan sebuah Confusion matrix untuk dua kelas yaitu Yes dan No. Pada kolom baris paling atas Actual=Yes berarti pada real data atau label bernilai Yes dan Actual=No berarti pada kenyataan atau real data bernilai No. Sedangkan pada kolom Predicted=Yes berarti nilai prediksi dari model bernilai Yes dan seterusnya kolom Predicted=No berarti nilai prediksi dari model bernilai No. Pada confusion matrik tersebut ada 4 kolom yaitu

- TP (True Positif) berisi jumlah data points diberi label Yes yang memang sebenarnya bernilai Yes.
- TN (True Negatif) berisi jumlah data points diberi label No yang memang sebenarnya bernilai No.
- FP (False Positif) berisi jumlah data points diberi label Yes yang memang sebenarnya bernilai No. Error ini biasa disebut Error Type 1.
- FN (False Negatif) berisi jumlah data points diberi label No yang memang sebenarnya bernilai Yes. Error ini biasa disebut Error Type 2.

Meme berikut ini menggambarkan Confusion matrix dengan jelas, dimana tiap kolomnya merepresentasikan keputusan gambar tentang hewan kucing atau anjing.

		PREDICTIVE VALUES	
		POSITIVE (CAT)	NEGATIVE (DOG)
ACTUAL VALUES	POSITIVE (CAT)	TRUE POSITIVE  3	FALSE NEGATIVE  1 TYPE II ERROR
	NEGATIVE (DOG)	FALSE POSITIVE  2 TYPE I ERROR	TRUE NEGATIVE  4

Gambar 3.15 Contoh confusion matrix dengan dua kelas yaitu yes dan no

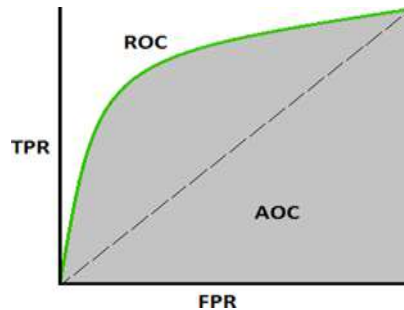
Berdasarkan nilai-nilai Confusion matrix tersebut maka dapat dihitung metric-metric berikut

- Akurasi : Perbandingan jumlah item yang di prediksi benar dengan total seluruh prediksi yang dilakukan.
- Recall atau Sensitivity atau TPR (True Positive Rate): Perbandingan jumlah item yang relevan diidentifikasi benar dengan seluruh item yang benar.

ROC dan AUC

Kurva AUC - ROC merupakan pengukuran performa untuk masalah klasifikasi pada berbagai pengaturan threshold. ROC merupakan kurva probabilitas dan AUC mewakili derajat atau ukuran keterpisahan. Ini memberi tahu seberapa besar model mampu membedakan antar kelas. Semakin tinggi AUC, semakin baik model memprediksi 0 sebagai 0 dan 1 sebagai 1. Dengan analogi, Semakin tinggi AUC, semakin baik model dalam membedakan antara pasien dengan penyakit dan tidak ada penyakit.

Kurva KOP diplotkan dengan TPR terhadap FPR dimana TPR berada pada sumbu y dan FPR pada sumbu x.



Gambar 3.16 Contoh grafik ROC

Model yang sangat baik memiliki AUC mendekati angka 1 yang berarti model tersebut memiliki ukuran keterpisahkan yang baik. Model yang buruk memiliki AUC mendekati 0 yang berarti model tersebut memiliki ukuran keterpisahan yang terburuk. Sebenarnya itu berarti membahas hasilnya. Ini memprediksi 0s sebagai 1s dan 1s sebagai 0s. Dan jika AUC 0,5, berarti model tidak memiliki kapasitas pemisahan kelas sama sekali.

RMSE

Root Mean Square Error (RMSE) merupakan metode pengukuran dengan mengukur perbedaan nilai dari prediksi sebuah model dengan nilai observasi atau label. RMSE dihitung dari akar kuadrat Mean Square Error. Nilai RMSE dapat berkisar dari 0 hingga ∞ . Keakuratan metode estimasi kesalahan pengukuran ditandai dengan adanya nilai RMSE yang kecil. Nilai RMSE rendah menunjukkan bahwa nilai yang dihasilkan oleh suatu model mendekati nilai observasinya. Sebuah model yang mempunyai nilai RMSE lebih kecil dikatakan lebih akurat daripada model yang memiliki nilai RMSE lebih besar. Adapun formula untuk menghitung merupakan

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2}$$

RMSE Root Mean Square Error (the overall average error magnitude)

Square Root (scales the error back to original units)

Mean (averages the squared differences over all n data points)

Number of Observations (the total sample size)

Summation (sums the values for each data point i , from 1 to n)

Squared Error (for each point i : squares the difference to avoid negative cancellation and penalize large errors)

Actual Value (for observation i , or Observed Value)

Predicted Value (for observation i , or Forecasted Value)

R-Square

Koefisien determinasi (R Square atau R kuadrat) atau disimbolkan dengan "R²" yang bermakna sebagai sumbangan pengaruh yang diberikan variabel bebas atau variabel independent (X) terhadap variabel terikat atau variabel dependent (Y). R Square ini berguna untuk melihat seberapa besar kontribusi pengaruh yang diberikan variabel X secara simultan (bersama- sama) terhadap variabel Y. Nilai R squared berkisar antara 0 sampai 1 yang mengindikasikan besarnya kombinasi variabel independen secara bersama – sama mempengaruhi nilai variabel dependen. Semakin mendekati angka satu, model yang dikeluarkan oleh regresi tersebut akan semakin baik. Adapun formula penghitungan merupakan

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

R²: R-Squared value

SS_{res}: Sum of Squared Residuals ($\sum (y_i - \hat{y}_i)^2$)

SS_{tot}: Total Sum of Squares ($\sum (y_i - \bar{y})^2$)

R² = Coefficient of Determination

Sebagai contoh interpretasi nilai R Square, jika nilai R merupakan sebesar 0,80 maka koefisien determinasi (R Square) merupakan $0,80 \times 0,80 = 0,64$. Berarti kemampuan variabel bebas dalam menjelaskan varians dari variabel terikatnya merupakan sebesar 64,0%. Berarti terdapat 36% (100%-64%) varians variabel terikat yang dijelaskan oleh faktor lain.

3.4 Model Deployment

Aktifitas model deployment merupakan aktivitas yang paling terakhir dilakukan. Model deployment ialah penerapan model-model machine learning, atau sederhananya, memasukkan model ke dalam mesin produksi, berarti membuat model yang tersedia untuk sistem bisnis yang lain.

Tantangan dalam men-deploy sebuah model merupakan menjaga skalabilitas dari model. Ketika di area produksi variasi dan jumlah input yang akan di dapat model akan berbeda pada fase pelatihan. Oleh karena itu menjaga performa model tetap baik merupakan salah satu tujuan yang paling penting.

Sebagian besar model mengalami permasalahan Concept Drift yang mengakibatkan penurunan performa model. Concept drift sederhananya merupakan perbedaan distribusi pada data latih dan data yang diuji. Akibatnya, hal tersebut akan mengubah relasi antara input dan output. Oleh karena itu, Ketika model di-deploy maka model harus dimonitor. Semua model yang di deploy harus mencatat semua input, output dan eksepsi yang terjadi. Biasanya pengembang menyiapkan sebuah infrastruktur untuk manajemen log dan visualisasi performance model.

3.5 Studi Kasus : Pengumpulan Data dengan Metode Scrapping berita detik.com

Sebagai contoh kita ingin mengambil data berita dari detik.com. Data yang diambil merupakan data berita yang berisi judul, isi berita dan tanggal. Kita tidak mungkin mencatat satu per satu item yang ada di detik.com. Kita dapat membuat sebuah robot yang akan mengumpulkan data yang kita inginkan.

Agar mempermudah, pada kasus ini penyusun akan menggunakan Jupyter Notebook sebagai tools untuk melakukan scrapping. Adapun langkah- langkah yang dilakukan merupakan:

Install library yang dibutuhkan

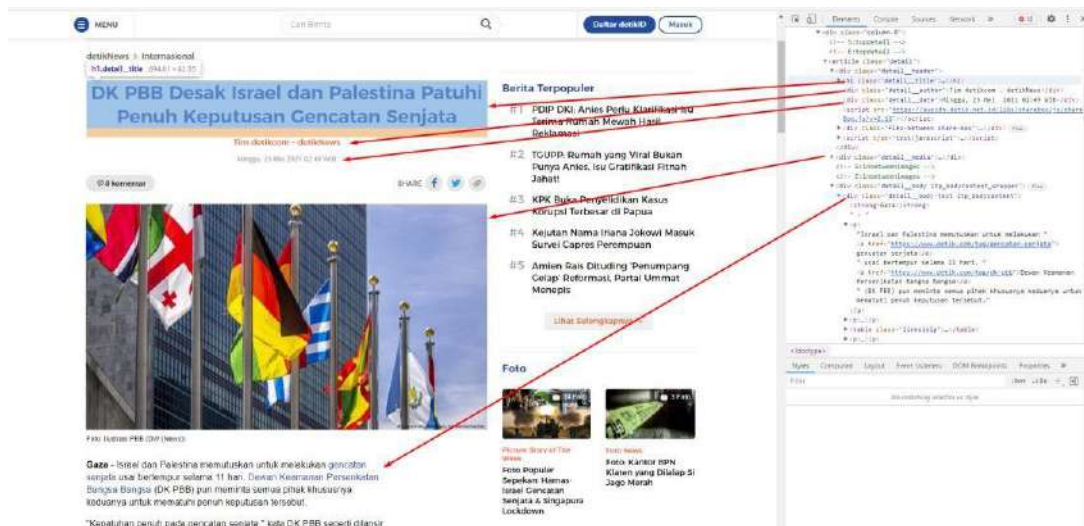
Untuk melakukan scapping, dibutuhkan beberapa library diantaranya BeautifulSoup dan Request. Untuk melakukan instalasi dapat menggunakan perintah berikut

```
[1] 1 !pip install BeautifulSoup4
    2 !pip install requests
```

Setelah library terinstall semua maka kita sudah siap melakukan scrapping.

Mengekstrak isi Halaman Web

Inspeksi bertujuan untuk mengetahui elemen apa yang harus dipanggil untuk melakukan ekstraksi konten. Sebagai contoh satu halaman berita detik.com. Dengan bantuan tool inspect dari google chrome maka kita dapat menganalisa struktur halaman berita detik.com. Untuk mengetahui harus melihat source code atau menggunakan tool inspektor pada Gambar 3.19.



Gambar 3.17 Inspeksi terhadap elemen data detik.com

Hasil inspeksi menunjukkan

- Judul berita terletak pada sebuah tag H1 dengan class detail title
- Penyusun terletak pada sebuah tag DIV dengan class detail author
- Tanggal terletak pada sebuah tag DIV dengan class detail data
- Isi berita terletak pada DIV dengan class detail body-text, namun didalamnya ada iklan-iklan ke halaman lain dalam bentuk tabel, oleh karena itu iklan ini harus dibersihkan

Parsing halaman webpage menggunakan BeautifulSoup

Setelah mengetahui struktur html maka langkah selanjutnya merupakan membuat kode yang memarsing halaman web tersebut. Setelah melakukan request pada URL tertentu (baris 9), maka selanjutnya merupakan menganalisa hasil request menggunakan BeautifulSoup (baris 10). Untuk mencari Judul maka perlu menemukan element berdasarkan tag H1 dengan class detail title (baris 12), tanggal dan author pada baris 13 dan 14 . Selanjutnya merupakan pembersihan konten dilakukan dengan menghapus semua elemen tabel pada teks.

```

1  import requests
2  from bs4 import BeautifulSoup
3  import pandas as pd
4  import numpy as np
5
6  #ambil berita detik
7  def getBeritaDetik(url):
8      B = { }
9      response = requests.get(url)
10     soup = BeautifulSoup(response.text, 'html.parser')
11     #ambil elemen-elemen berita
12     B['judul'] = soup.find('h1', {'class': 'detail__title'}).text.replace('\n', "").strip()
13     B['tanggal'] = soup.find('div', {'class': 'detail__date'}).text.replace('\n', "").strip()
14     B['author'] = soup.find('div', {'class': 'detail__author'}).text.replace('\n', "").strip()
15     berita = soup.find('div', {'class': 'detail__body-text'})
16     text_berita = berita.text
17     #bersihkan isi berita
18     blah = berita.find_all("table")
19     for x in blah:
20         text_berita = text_berita.replace(x.text, '').replace('\n', "").strip()
21         #print(x.text)
22     B['berita'] = text_berita
23     return(B)
24

```

Method tersebut dapat digunakan untuk mengambil 1 halaman berita contohnya:

```

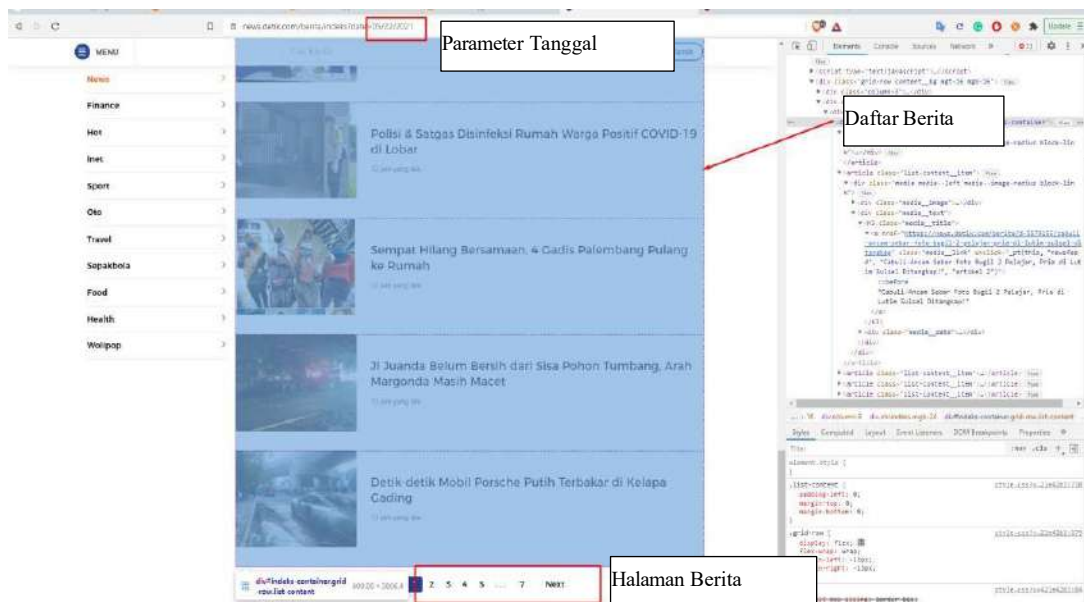
1  getBeritaDetik("https://news.detik.com/berita/"+
2  | | | | | | | "d-5579022/detik-detik-mobil-porsche-putih-terbakar-di-kelapa-gading")

{'author': 'Yogi Ernes - detikNews',
 'berita': 'Jakarta - Sebuah mobil sport merek Porsche hangus terbakar di Kelapa Gading, Jakarta Utara.',
 'judul': 'Detik-detik Mobil Porsche Putih Terbakar di Kelapa Gading',
 'tanggal': 'Sabtu, 22 Mei 2021 19:49 WIB'}

```

Mengambil index daftar berita pertanggal

Setelah berhasil mengambil data per berita, maka langkah selanjutnya merupakan mengambil seluruh berita. Index berita tersedia pada alamat <https://news.detik.com/berita/indek> dan dapat dilihat berdasarkan tanggal. Daftar berita pada tanggal tertentu dipisahkan berdasarkan halaman- halaman, karena jumlah berita pada tanggal tersebut cukup banyak. Hasil analisa menunjukkan bahwa kita dapat memberikan parameter date dan nomer halaman. Struktur halaman indek dpaat dilihat pada Gambar 3.20.



Gambar 3.18 Struktur Index berita detik.com

Semua konten terletak pada div dengan kelas list-kontent dan judul dan url terletak pada tag article. Setelah mengambil list url yang ada pada artikel, maka berdasarkan link tersebut kita ambil detail beritanya (baris 21). Adapun source code untuk mengambil list merupakan sebagai berikut :

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 def indexBerita (tanggal, jumlahHalaman):
5     daftarBerita = []
6     halaman=0
7     for halaman in range(0, jumlahHalaman):
8         halaman = halaman + 1
9         base_url = 'https://news.detik.com/berita/index/' + str(halaman)+'?date='+tanggal
10        #print(base_url)
11
12        # Request URL and BeautifulSoup
13        r = requests.get(base_url)
14        soup = BeautifulSoup(r.text, "html.parser")
15
16        berita_container = soup.find('div', {'id': 'indeks-container'})
17        berita = berita_container.find_all('article')
18
19        for item in berita:
20            x = item.find("a", href=True)
21            berita = getBeritaDetik(x['href'])
22            daftarBerita.append(berita)
23        return daftarBerita
24
```

Luaran dari method tersebut dapat kita oleh lebih lanjut atau disimpan melalui excel menggunakan pandas

```
1 tanggal = '05/22/2021'
2 jumlahHalaman=1
3 list_berita = indexBerita(tanggal,jumlahHalaman)
4 #olah pada dataframe
5 df = pd.DataFrame(list_berita)
6 print(df.columns)
7 print(df)
```

Index(['judul', 'tanggal', 'author', 'berita'], dtype='object')

	judul	tanggal	author	berita
0	Menhub: Masyarakat dari Sumatera ke Jawa Wajib...	2021-05-22	Budi Karya	Jakarta - Menteri Perhubungan (Menhub) Budi Karya Sumadi meminta agar masyarakat di Sumatera dan Jawa harus tetap mematuhi protokol kesehatan.
1	Cabuli-Ancam Sebar Foto Bugil 2 Pelajar, Pria ...	2021-05-22	Adrian (22)	Luwu Timur - Pemuda bernama Adrian (22) di Luwu Timur (Lutim), Sulawesi Selatan, ditangkap karena diduga menyebarkan foto bugil dua pelajar.
2	Mendes Minta Pendamping Desa Gotong-Royong Ban...	2021-05-22	Menteri Desa, Pembangunan Daerah Tertinggal dan Transmigrasi	Jakarta - Menteri Desa, Pembangunan Daerah Tertinggal dan Transmigrasi (Mendes) meminta pendamping desa untuk membantu masyarakat dalam kegiatan gotong royong.
3	Gempa M 3,6 Terjadi di Waingapu Sumba Timur	2021-05-22	Gempa berkekuatan magnitudo (M) 3,6	Jakarta - Gempa berkekuatan magnitudo (M) 3,6 terjadi di Waingapu, Sumba Timur, Nusa Tenggara Timur.
4	Beri Benih hingga Cold Storage, Mentan Harap P...	2021-05-22	Kementerian Pertanian (Kementan)	Jakarta - Kementerian Pertanian (Kementan) menyerahkan bantuan benih hortikultura kepada petani di berbagai daerah.
5	MUI Setuju Saran JK soal Kotak Amal untuk Pale...	2021-05-22	Majelis Ulama Indonesia (MUI)	Jakarta - Majelis Ulama Indonesia (MUI) mendukung saran Ketua Mahkamah Agung (MA) agar kotak amal untuk Palestina tidak digunakan untuk kepentingan politik.
6	Dalih Ngantuk Pemobil Tabrak Lari Pedagang Mi ...	2021-05-22	Berkat 'kesaktian' teknologi, penge...	Jakarta - Berkat 'kesaktian' teknologi, pengendara sepeda motor yang mengantuk dapat diidentifikasi oleh sistem keamanan.
7	Mendes Jadikan 2 Desa di Jatim Percontohan Pem...	2021-05-22	Menteri Desa, Pembangunan Daerah Tertinggal dan Transmigrasi	Jakarta - Menteri Desa, Pembangunan Daerah Tertinggal dan Transmigrasi (Mendes) menjadikan dua desa di Jawa Timur sebagai desa percontohan.
8	KPK Bantah OTT Bupati Nganjuk Ditangani Baresk...	2021-05-22	Direktur Sosialisasi dan Kampanye An...	Jakarta - Direktur Sosialisasi dan Kampanye Anti-Korupsi Komisi Pemberantasan Korupsi (KPK) membantah dugaan pelanggaran oleh Bupati Nganjuk.
9	Jejak Dokter Asal Sumut Jual Vaksin Corona Ile...	2021-05-22	Polda Sumatera Utara (Sumut)	Jakarta - Polda Sumatera Utara (Sumut) menetapkan empat orang, termasuk dokter asal Sumut, sebagai tersangka dalam kasus jual beli vaksin.
10	Cegah Lonjakan COVID-19, Satgas Minta 7 Kota Z...	2021-05-22	Satgas Penanganan COVID-19	Jakarta - Satgas Penanganan COVID-19 meminta pemerintah kota untuk meningkatkan upaya pencegahan penyebaran virus corona.
11	Libur Lebaran, Vaksinasi Lansia di NTB Tetap J...	2021-05-22	Vaksinasi lansia tetap dilaksanakan	Jakarta - Vaksinasi lansia tetap dilaksanakan di NTB meskipun sedang libur Lebaran.
12	Saling Serang Partai Ummat-Ngabalun Buntut Ami...	2021-05-22	Tenaga Ahli Utama Kantor Staf Presi...	Jakarta - Tenaga Ahli Utama Kantor Staf Presiden (KSP) Ali Mochtar Ngabalin dan Erika Dyah saling menyerang di media sosial.
13	Tak Diizinkan, Puluhan Pesepeda Gagal Peringat...	2021-05-22	Beberapa organisasi kemasyarakatan s...	Jakarta - Beberapa organisasi kemasyarakatan meminta pemerintah untuk memperketat aturan lalu lintas bagi pesepeda.

Mengubah dataset menjadi Excel

Untuk mengubah menjadi Excel maka cukup menjalankan perintah `to_excel` pada Pandas Dataframe pembaca. Contoh penggunaannya merupakan berikut

```
1 df.to_excel("berita.xls")
```

	A	B	C	D	E	F
		judul	tanggal	author	berita	
0		Menhub: Masyarakat Sabtu, 22 Mei 2021 23:44	Matus Alfons - detikNews	Jakarta - Menteri Perhubungan (Menhub) Budi Karya Sumadi meminta agar masyarakat di Sumatera dan Jawa harus tetap mematuhi protokol kesehatan.		
1		Cabuli-Ancam Sebar Sabtu, 22 Mei 2021 23:13	Hermawan Mappiwali - detikNews	Luwu Timur - Pemuda bernama Adrian (22) di Luwu Timur (Lutim), Sulawesi Selatan, ditangkap karena diduga menyebarkan foto bugil dua pelajar.		
2		Mendes Minta Penda Sabtu, 22 Mei 2021 23:05	Jihaan Khoirunnisa - detikNews	Jakarta - Menteri Desa, Pembangunan Daerah Tertinggal dan Transmigrasi (Mendes) meminta pendamping desa untuk membantu masyarakat dalam kegiatan gotong royong.		
3		Gempa M 3,6 Terjadi Sabtu, 22 Mei 2021 22:54	Tim detikcom - detikNews	Jakarta - Gempa berkekuatan magnitudo (M) 3,6 terjadi di Waingapu, Sumba Timur, Nusa Tenggara Timur.		
4		Beri Benih hingga Co Sabtu, 22 Mei 2021 22:48	Nadhifa Sarah Amalia - detikNews	Jakarta - Kementerian Pertanian (Kementan) menyerahkan bantuan benih hortikultura kepada petani di berbagai daerah.		
5		MUI Setuju Saran JK Sabtu, 22 Mei 2021 22:40	Tim detikcom - detikNews	Jakarta - Majelis Ulama Indonesia (MUI) mendukung saran Ketua Mahkamah Agung (MA) agar kotak amal untuk Palestina tidak digunakan untuk kepentingan politik.		
6		Dalih Ngantuk Pemol Sabtu, 22 Mei 2021 22:23	Tim detikcom - detikNews	Jakarta - Berkat 'kesaktian' teknologi, pengendara sepeda motor yang mengantuk dapat diidentifikasi oleh sistem keamanan.		
7		Mendes Jadikan 2 Di Sabtu, 22 Mei 2021 22:01	Inkana Putri - detikNews	Jakarta - Menteri Desa, Pembangunan Daerah Tertinggal dan Transmigrasi (Mendes) menjadikan dua desa di Jawa Timur sebagai desa percontohan.		
8		KPK Bantah OTT Bu Sabtu, 22 Mei 2021 21:50	Matus Alfons - detikNews	Jakarta - Direktur Sosialisasi dan Kampanye Anti-Korupsi Komisi Pemberantasan Korupsi (KPK) membantah dugaan pelanggaran oleh Bupati Nganjuk.		
9		Jejak Dokter Asal Su Sabtu, 22 Mei 2021 21:41	Tim detikcom - detikNews	Jakarta - Polda Sumatera Utara (Sumut) menetapkan empat orang, termasuk dokter asal Sumut, sebagai tersangka dalam kasus jual beli vaksin.		
10		Cegah Lonjakan COV Sabtu, 22 Mei 2021 21:40	Erika Dyah - detikNews	Jakarta - Satgas Penanganan COVID-19 meminta pemerintah kabupaten untuk meningkatkan upaya pencegahan penyebaran virus corona.		
11		Libur Lebaran, Vaksi Sabtu, 22 Mei 2021 21:28	Nadhifa Sarah Amalia - detikNews	Jakarta - Vaksinasi lansia tetap dilaksanakan meski saat Idul Fitri. Vaksinasi lansia tetap dilaksanakan di NTB meskipun sedang libur Lebaran.		
12		Saling Serang Partai Sabtu, 22 Mei 2021 21:01	Tim detikcom - detikNews	Jakarta - Tenaga Ahli Utama Kantor Staf Presiden (KSP) Ali Mochtar Ngabalin dan Erika Dyah saling menyerang di media sosial.		
13		Tak Diizinkan, Puluhi Sabtu, 22 Mei 2021 20:55	Adhyasta Dirgantara - detikNews	Jakarta - Beberapa organisasi kemasyarakatan sipil bersama mahasiswa meminta pemerintah untuk memperketat aturan lalu lintas bagi pesepeda.		
14		Kisah Ipda Tita, Polw Sabtu, 22 Mei 2021 20:30	Yogi Ernes - detikNews	Serpong - Kasus penganiayaan anak yang dilakukan ayahnya sendiri diungkap oleh polisi.		
15		Tekan COVID-19, Sa Sabtu, 22 Mei 2021 20:30	Inkana Putri - detikNews	Jakarta - Juru Bicara Satgas Penanganan COVID-19 Prof Wiku Adisasmita meminta pemerintah kabupaten untuk meningkatkan upaya pencegahan penyebaran virus corona.		
16		Polisi & Satgas Disin Sabtu, 22 Mei 2021 20:28	Erika Dyah - detikNews	Jakarta - Bhabinkamtibmas Desa Midang Polsek Gunung Sari, Aipda Dewa Dewa melakukan patroli rutin di Desa Midang.		
17		Sempat Hilang Bersa Sabtu, 22 Mei 2021 20:16	Prima Syahbana - detikNews	Palembang - Empat gadis di Palembang, Sumatera Selatan, yang dilaporkan hilang ditemukan di Palembang.		
18		Jl Juanda Belum Beri Sabtu, 22 Mei 2021 20:10	Adhyasta Dirgantara - detikNews	Depok - Sejumlah pohon tumbang di beberapa titik di Jl Ir H Juanda, Depok, akibat hujan deras.		
19		Detik-detik Mobil Por Sabtu, 22 Mei 2021 19:45	Yogi Ernes - detikNews	Jakarta - Sebuah mobil sport merek Porsche hangus terbakar di Kelapa Gading.		

Gambar 3.19 Luaran Berita Detik.com tanggal 22 Mei 2021

3.6 Studi Kasus - Data Preprocessing

Untuk mempermudah, melakukan data preprocessing, penyusun menyarankan untuk menggunakan Jupyter Notebook karena dapat mempermudah dalam menganalisa dan memperbaiki data. Adapun tahapan-tahapan dalam preprocessing dapat berbeda-beda, tergantung kepada permasalahan yang ada. Pada kasus ini penyusun akan menganalisa dataset Titanic. Adapun langkah-langkah yang dilakukan merupakan :

Informasi Dataset

Dataset yang digunakan merupakan dataset Titanic. Dataset ini berisi sejumlah daftar penumpang kapal Titanic yang menjadi korban dan selamat pada kecelakaan kapal tersebut. Dataset ini digunakan untuk memprediksi apakah seseorang selamat atau tidak berdasarkan data-data lainnya. Adapun informasi fitur pada data ini dapat dilihat pada Tabel 3.2

Table 3.2 Informasi fitur atau kolom dataset Titanic

Variable	Definisi	Penjelasan Isi
survival	Kolom label yang menyatakan penumpang selamat atau tidak	0 = Tidak selamat, 1 = Selamat
pclass	Kelas tiket penumpang	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Jenis kelamin penumpang	
Age	Umur penumpang dalam tahun	
sibsp	jumlah saudara kandung / pasangan di kapal Titanic	
parch	jumlah orang tua / anak di kapal Titanic	
ticket	Nomer tiket	
fare	Tarif Penumpang	
cabin	Nomor Kabin	
embarked	Pelabuhan keberangkatan	C = Cherbourg, Q = Queenstown, S = Southampton

Memuat dan menganalisa tipe data

Dataset ini terdiri atas dua dokumen yaitu train.csv dan test.csv. Train.csv berisi data yang akan digunakan untuk pelatihan dan test.csv merupakan data yang akan digunakan untuk evaluasi. Untuk memuat data ke dalam notebook kita dapat menggunakan pandas dengan fungsi `read_csv` dengan parameter nama file. Pada kasus ini langkah tersebut dilakukan pada baris 11 dan 12. Setelah berhasil memuat data, kita tampilkan 5 data pertama dengan menggunakan fungsi `head()` (Baris 13)

Berdasarkan 5 data pertama dari fungsi `head`, kita telah dapat menentukan tipe-tipe data. Adapun tipe data yang ada pada dataset ini merupakan

- Kategori : Survived, Sex, dan Embarked.
- Ordinal : Pclass
- Continous: Age, Fare.
- Discrete: SibSp, Parch.

```
[2] 1 train_df.info()
    2
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
#   Column          Non-Null Count  Dtype
---  -
0   PassengerId     891 non-null   int64
1   Survived        891 non-null   int64
2   Pclass         891 non-null   int64
3   Name            891 non-null   object
4   Sex             891 non-null   object
5   Age             714 non-null   float64
6   SibSp           891 non-null   int64
7   Parch           891 non-null   int64
8   Ticket          891 non-null   object
9   Fare            891 non-null   float64
10  Cabin           204 non-null   object
11  Embarked        889 non-null   object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
1 # data analysis and wrangling
2 import pandas as pd
3 import numpy as np
4 import random as rnd
5
6 # visualization
7 import seaborn as sns
8 import matplotlib.pyplot as plt
9 %matplotlib inline
10
11 train_df = pd.read_csv('train.csv')
12 test_df = pd.read_csv('test.csv')
13 train_df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Sedangkan untuk tipe data Ticket merupakan gabungan antara numerik dan alpha-numerik dan data Cabin merupakan alpha-numerik dan dari analisa awal terdapat missing value (bernilai NaN). Dalam komputasi, NaN merupakan singkatan dari Not a Number. NaN merupakan nilai tipe data numerik yang mewakili nilai yang tidak ditentukan atau tidak terwakili.

```
[3] 1 train_df.isnull().sum()
```

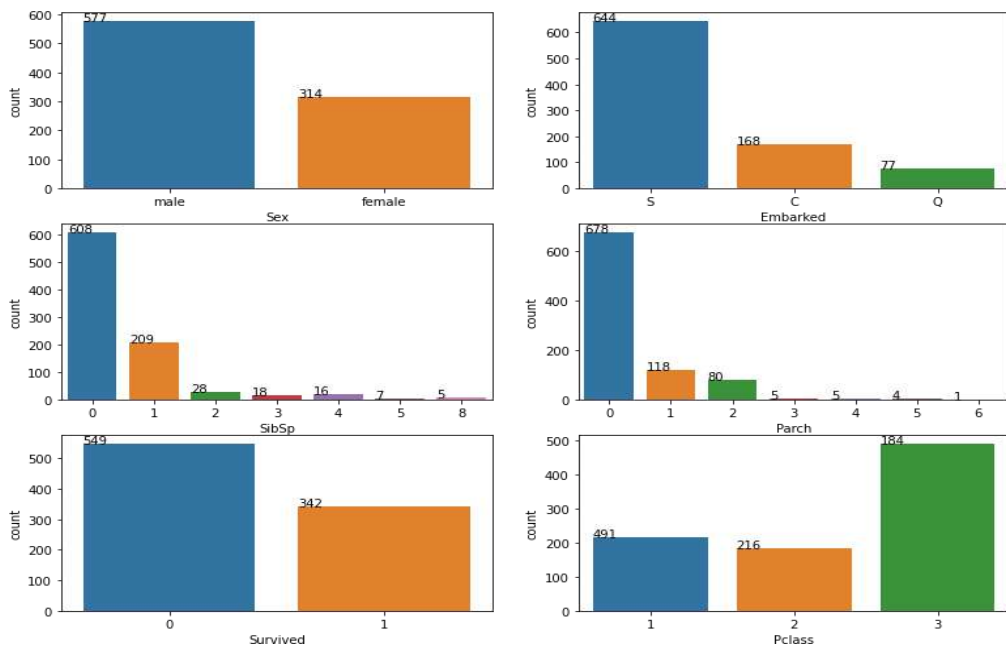
```
PassengerId      0
Survived          0
Pclass           0
Name             0
Sex              0
Age             177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin           687
Embarked         2
dtype: int64
```

Pengecekan Konsistensi Data

Berdasarkan analisa awal, kita memiliki tiga fitur yang bertipe kategori yaitu Survived, Sex, dan Embarked, dua discrete dan satu ordinal. Ketiga data tersebut akan dilihat sebaran nilainya dan konsistensi nilai (untuk kategori).

```
1  #pengaturan layout
2  n_rows=3
3  n_cols=2
4  #daftar kolom
5  kolom=["Sex","Embarked","SibSp","Parch","Survived","Pclass"]
6
7  fig, axes = plt.subplots(nrows=n_rows, ncols=n_cols, figsize=(13, 10))
8  i=0
9  fig.suptitle('CountPlot Dataset Titanic', fontsize="28")
10 for row in range(n_rows):
11     for col in range(n_cols):
12         #menampilkan
13         sns.countplot(x=kolom[i], data=train_df, ax=axes[row,col])
14         #menampilkan label nilai per unique value
15         for p, label in zip(axes[row,col].patches, train_df[kolom[i]].value_counts()):
16             axes[row,col].annotate(label, (p.get_x(), p.get_height()))
17         i=i+1
18
19
```

CountPlot Dataset Titanic



Pada fitur sex dan embark ini tidak ditemukan inkonsistensi data. Pada fitur Sex terdapat dua nilai unik yakni male dan female dan kolom Embarked memiliki tiga nilai unik yakni S,C dan Q. Contoh inkonsistensi data pada dataset merupakan apabila misalnya pada sex ditemukan nilai seperti male, M, Female, FEMALE. Data yang bernilai male dan M merupakan sama tetapi memiliki value yang berbeda. Solusinya merupakan dengan menyatukan data-data yang sama menjadi nilai yang sama.

Missing Data, Feature Selection dan Scaling

Pada dataset ini, kolom nama, tiket, Informasi kabin tidak digunakan sehingga kolom-kolom ini dapat dihapus

```
1 #Hapus Kolom yang tidak di inginkan.
2 train_df=train_df.drop(["Name","Ticket","Cabin","PassengerId"],axis=1)
3 train_df.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	20.0	1	0	7.0	S
1	1	1	female	36.0	1	0	71.0	C
2	1	3	female	24.0	0	0	7.0	S
3	1	1	female	33.0	1	0	52.0	S
4	0	3	male	33.0	0	0	7.0	S

Selanjutnya, perubahan data kategorikal menjadi data numerik. Kita dapat menggunakan fungsi map pada dataframe. Pada kolom sex, female akan diubah menjadi 0 dan male menjadi 1. Begitu juga untuk Embarked S=0, C=1 dan Q=2

```
1 #KONVERT STRING VALUES(CATEGORICAL VALUES) MENJDI INTEGER
2 train_df.Sex=train_df.Sex.map({"female":0,"male":1})
3 train_df.Embarked=train_df.Embarked.map({"S":0,"C":1,"Q":2})
4 train_df.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	20.0	1	0	7.0	0
1	1	1	0	36.0	1	0	71.0	1
2	1	3	0	24.0	0	0	7.0	0
3	1	1	0	33.0	1	0	52.0	0
4	0	3	1	33.0	0	0	7.0	0

Kolom Age masih memiliki missing value oleh karena itu kita dapat menggunakan median untuk mengisi umur yang kosong

```
1 train_df['Age'] = train_df['Age'].fillna((train_df['Age'].median()))
```

Agar performa model bagus, kita dapat melakukan scaling terhadap dua nilai yaitu age dan fare

```
1 from sklearn.preprocessing import StandardScaler
2 train_df["Age"]=round((train_df.Age-train_df.Age.mean())/train_df.Age.std())
3 train_df["Fare"]=round((train_df.Fare-train_df.Fare.mean())/train_df.Fare.std())
```

BAB IV

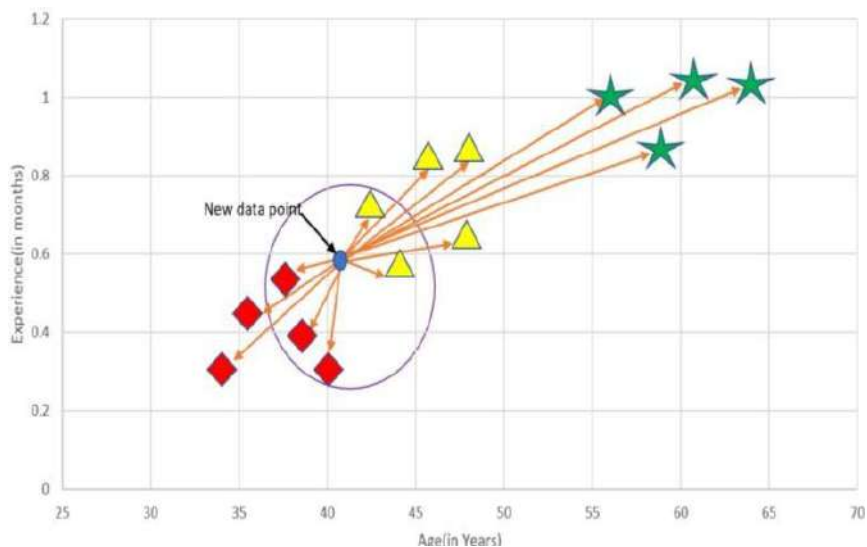
K-NEAREST NEIGHBORS

K-Nearest Neighbors merupakan salah satu algoritma yang paling dasar dalam kasus klasifikasi. Algoritma ini biasa juga disebut K-Nearest Neighbors (KNN).

KNN bersifat non parametrik, yang artinya, ia tidak memerlukan asumsi distribusi sehingga sebaran data bebas dan merupakan jenis algoritma supervised learning yang bersifat instance based. Sederhananya, KNN tidak membuat generalisasi atau asumsi pada model, sehingga algoritma ini bebas untuk melakukan training pada segala jenis distribusi data. Altman, N. S. (1992). An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. Aha, D. W. , K. D. , & A. M. K. (1991). Instance-based learning algorithms. Machine Learning, 6. Algoritma ini biasanya digunakan untuk menyelesaikan permasalahan klasifikasi dan regresi.

KNN bekerja dengan cara membuat prediksi dengan menghitung kemiripan antara objek-objek yang berada di data training dengan objek yang dites. Kemiripan dihitung dengan menggunakan fungsi “jarak”.

4.1 Definisi K-Nearest Neighbors (KNN)



Gambar 4.1 Ilustrasi KNN

Algoritma KNN dapat didefinisikan berdasarkan namanya dimana K melambangkan sejumlah nilai tetangga terdekat (Nearest Neighbors) yang diguna untuk mengidentifikasi kemiripan sebuah titik baru dengan titik tetangganya. Sebagai contoh perhatikan Gambar 4.1, dimana kita ingin memiliki 3 kelompok data yaitu bintang, segitiga dan segiempat, kemudian kita diberikan sebuah titik baru yang tidak diketahui

kelompoknya. Untuk menentukan kelompok titik baru tersebut menggunakan KNN dengan nilai $K = 5$ maka kita akan menghitung jarak antara titik yang baru dengan semua titik yang ada pada data, lalu mengambil 5 titik terdekat. Dari 5 titik terdekat itu dilakukan voting. Sehingga didapatkan hasil 2 titik merupakan kelompok segitiga dan 3 titik kelompok segi empat. Maka dapat disimpulkan bahwa titik yang baru ini masuk kelompok titik segiempat. Bahasa sederhananya merupakan kita melakukan voting berdasarkan tetangga terdekat untuk menentukan kelas atau kelompok sebuah titik baru. Adapun langkah-langkah algoritma KNN merupakan sebagai berikut:

1. Menentukan nilai K . Nilai K sebaiknya bilangan asli ganjil untuk menghindari hasil voting yang seimbang.
2. Menghitung jarak antara titik baru dengan semua data training.
3. Memilih K titik terdekat dengan titik yang baru
4. Pada kasus klasifikasi, dilakukan penghitungan jumlah titik terdekat tadi berdasarkan kategori/kelas. Titik yang baru akan masuk ke kategori yang memiliki jumlah titik terdekat terbanyak. Sedangkan pada kasus regresi, titik yang baru akan berada pada nilai rata-rata K tetangga terdekat.

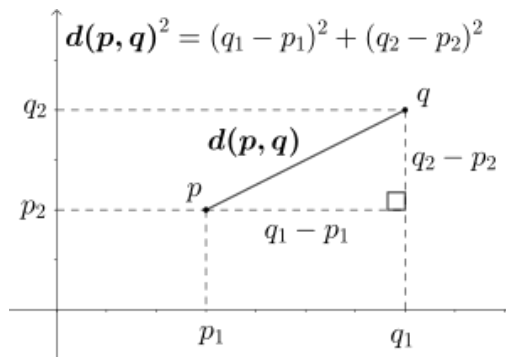
4.2 Jarak Data KNN

Penghitungan jarak merupakan kunci kesuksesan algoritma ini. Jarak melambangkan kemiripan antara data test dengan data training. Semakin jauh jarak berarti semakin tidak mirip, begitu juga sebaliknya. Ada beberapa cara yang dapat digunakan untuk menghitung jarak diantaranya:

Euclidean distance

Euclidean distance merupakan perhitungan jarak dari 2 buah titik dalam Euclidean space. Euclidean space diperkenalkan oleh Euclid, seorang matematikawan dari Yunani sekitar tahun 300 B.C.E. untuk mempelajari hubungan antara sudut dan jarak. Euclidean ini berkaitan dengan Teorema Pythagoras dan biasanya diterapkan pada 1, 2 dan 3 dimensi. Heath, T. L. (1956). *The Thirteen Books of Euclid's Elements*. Dover Publications. Dan Anton, H. , & R. C. (2013). *Elementary Linear Algebra: Applications Version*. John Wiley & Sons.

Tapi juga sederhana jika diterapkan pada dimensi yang lebih tinggi. Sederhananya jarak ini sering dikatakan sebagai jarak garis lurus antara dua titik.



Gambar 4.2 Ilustrasi Euclidean distance

Sebagai contoh, misalnya diketahui usia dan berat budi merupakan 25 tahun dan 65 kg, andi berusia 27 tahun dengan berat badan 70 kg. Pertanyaan yang ingin dijawab merupakan siapa yang mirip dengan umar yang berusia 26 tahun dengan berat badan 67 kg?

Pertanyaan itu bisa dijawab menggunakan pendekatan jarak euclidean. Anggap saja variabel usia merupakan x dan variabel berat badan merupakan y .

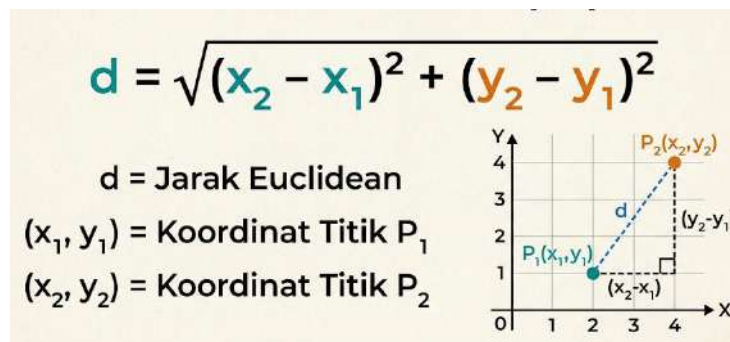
Maka, implementasinya seperti ini:

$$d(\text{budi}, \text{umar}) = \sqrt{(25 - 26)^2 + (65 - 67)^2} = \sqrt{1 + 4} = 2.24$$

$$d(\text{andi}, \text{umar}) = \sqrt{(27 - 26)^2 + (70 - 67)^2} = \sqrt{1 + 9} = 3.16$$

Kesimpulannya, saya mirip umar? pembaca sudah tahu jawabannya. Mana jarak yang paling dekat diantara 2,24 dan 3,16. Tentu saja 2,24. Jadi, semakin kecil nilai jarak euclidean maka semakin mirip. Jika hasilnya 0? berarti bukan hanya mirip tapi sama.

Formula umum dari jarak euclidean antara titik p dan q merupakan sebagai berikut dimana n menunjukkan dimensi data



Adapun implementasi dari persamaan diatas dapat dilihat pada source code berikut ini dimana kita akan mencari jarak diantara dua vektor.

```
1  from math import sqrt
2
3  # menghitung calculate euclidean
4  def euclidean_distance(a, b):
5      return sqrt(sum((e1-e2)**2 for e1, e2 in zip(a,b)))
6
7  # define data
8  d1 = [12, 8, 9, 0, 0, 1]
9  d2 = [20, 10, 2, 1, 1, 0]
10 # calculate distance
11 dist = euclidean_distance(d1, d2)
12 print(dist)
```

10.954451150103322

Manhattan distance

Manhattan distance atau Jarak Manhattan antara dua titik merupakan jumlah dari panjang ruas garis kedua titik tersebut terhadap tiap sumbu dalam koordinat Kartesius. Jarak ini disebut juga dengan panjang Manhattan, jarak taksi, atau jarak snake. Nama jarak ini berasal dari tata letak jalan di pulau Manhattan yang berbentuk kisi-kisi segi empat. Krause, E. F. (1986). Taxicab Geometry: An Adventure in Non-Euclidean Geometry. . Dover Publications. Minkowski, H. (1910). Geometrie der Zahlen (Geometry of Numbers). Teubner.

Jarak ini telah digunakan dalam analisis regresi sejak abad ke-18, dan saat ini umum dirujuk dengan LASSO.

Jarak Manhattan dalam ruang R dengan sistem koordinat Kartesius, antara vektor $p = \{p_1, p_2, p_3, \dots, p_n\}$ dan $q = \{q_1, q_2, q_3, \dots, q_n\}$ merupakan jumlah panjang proyeksi ruas garis antara kedua vektor tersebut terhadap sumbu-sumbu koordinat. Secara matematis, jarak Manhattan dapat didefinisikan sebagai berikut

$$d(p, q) = |p_1 - q_1| + |p_2 - q_2| + \dots + |p_n - q_n| = \sum_{i=1}^n |p_i - q_i| \quad (4-2)$$



Gambar 4.3 Ilustrasi Manhattan distance

Ilustrasi dari pengukuran jarak ini dapat dilihat dari gambar 4.3 dimana untuk pergi dari titik (x_1, y_1) ke titik (x_2, y_2) melalui jalan yang berbentuk blok maka akan terdapat banyak kombinasi jalan. Perhatikan jalan A dan jalan B. Walaupun bentuknya berbeda namun total jarak tempuhnya sama.

Adapun implementasi dari persamaan diatas dapat dilihat pada source code dibawah ini

```
1  from math import sqrt
2
3  # menghitung manhattan distance
4  def manhattan_distance(a, b):
5      return sum(abs(e1-e2) for e1, e2 in zip(a,b))
6
7  # define data
8  d1 = [12, 8, 9, 0, 0, 1]
9  d2 = [20, 10, 2, 1, 1, 0]
10 # calculate distance
11 dist = manhattan_distance(d1,d2)
12 print(dist)
```

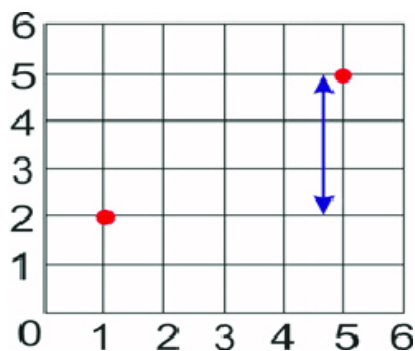
20

Jarak Manhattan berfungsi dengan baik untuk data berdimensi tinggi, namun metrik ini merupakan ukuran yang kurang intuitif dibandingkan jarak euclidean, terutama bila digunakan dalam data dimensi tinggi. Ketika digunakan pada data diskrit dan/atau biner, Manhattan distance memperhitungkan jalur yang secara realistis dapat diambil dalam nilai atribut tersebut. Selain itu, jarak tersebut lebih cenderung

memberikan nilai jarak yang lebih tinggi daripada jarak euclidean karena bukan merupakan jalur terpendek yang antara dua titik.

Chebyshev Distance

Chebyshev Distance atau Jarak Chebyshev didefinisikan sebagai selisih terbesar antara dua vektor sepanjang dimensi koordinat mana pun. Dengan kata lain, ini hanyalah jarak maksimum sepanjang satu sumbu. Karena sifatnya, itu sering disebut sebagai jarak Papan Catur karena jumlah minimum gerakan yang dibutuhkan oleh raja untuk berpindah dari satu kotak ke kotak lainnya sama dengan jarak Chebyshev. chebychef, P. (1854). *Théorie des mécanismes connus sous le nom de parall.* Rosenfeld, A. , & P. J. L. (1968). *Distance Functions on Digital Pictures.* *Pattern Recognition*, 1(1).



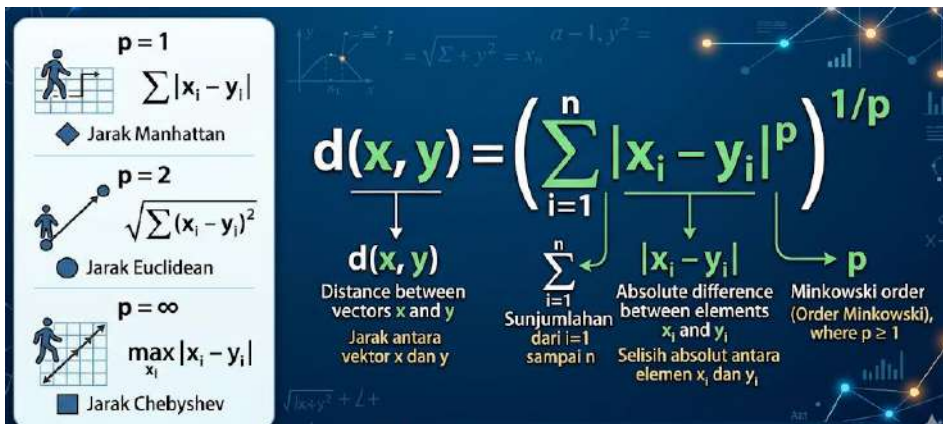
Gambar 4.4 Ilustrasi Chebyshev distance

Chebyshev biasanya digunakan dalam kasus penggunaan yang sangat spesifik, yang membuatnya sulit untuk digunakan sebagai metrik jarak serba guna, seperti jarak Euclidean atau Cosine distance. Oleh karena itu, disarankan untuk hanya menggunakannya jika Pembaca benar-benar yakin metode ini sesuai dengan kasus penggunaan Pembaca.

Minkowski Distance

Jarak Minkowski atau metrik Minkowski merupakan metrik dalam ruang vektor yang dapat disebut sebagai generalisasi jarak Chebyshev, Euclid dan jarak Manhattan. Jarak ini dinamai dari Hermann Minkowski, matematikawan asal Jerman. Minkowski, H. (1910). *Geometrie der Zahlen (Geometry of Numbers)*. Teubner. Han, J., Kamber, M., & Pei, J. (2011). *Data Mining. Concepts and Techniques*, 3rd Edition (The Morgan Kaufmann Series in Data Management Systems).

Adapun formula dari jarak ini merupakan



Ketika menggunakan jarak ini, kita harus menentukan parameter P. Tantangannya merupakan memahami berapa nilai P yang sesuai untuk kasus yang dihadapi. Berikut ini Nilai-nilai umum p merupakan:

- $p = 1$ untuk Jarak Manhattan
- $p = 2$ untuk Jarak Euclidean
- $p = \infty$ untuk Jarak Chebyshev, dengan menemukan limit dari vector tersebut maka sebenarnya jarak ini merupakan nilai maksimum dari vector-vector tersebut.

```
1 from math import sqrt
2
3 # menghitung minkowski distance
4 def minkowski_distance(a, b, p):
5     return sum(abs(e1-e2)**p for e1, e2 in zip(a,b))**(1/p)
6
7 # define data
8 d1 = [12, 8, 9, 0, 0, 1]
9 d2 = [20, 10, 2, 1, 1, 0]
```

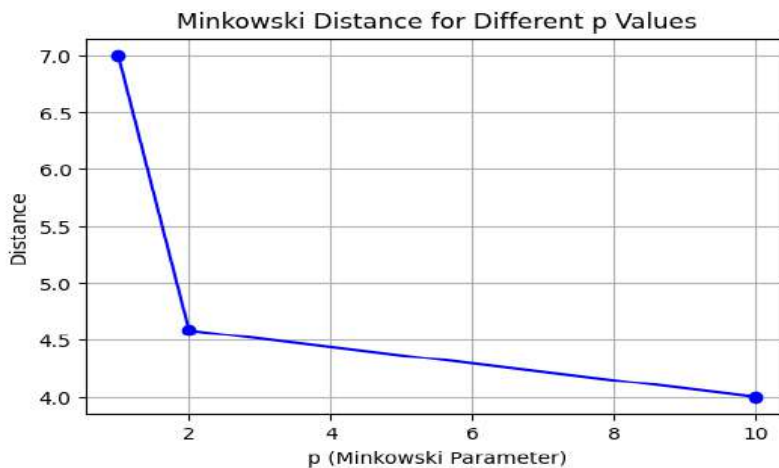
```
10 # menghitung distance (p=1)
11 dist = minkowski_distance(d1,d2, 1)
12 print("P1: ",dist)
13 # menghitung distance (p=2)
14 dist = minkowski_distance(d1,d2, 2)
15 print("P2: ",dist)
```

```
P1: 20.0
P2: 10.954451150103322
```

Minkowski Distance (p=1): 7.0000

Minkowski Distance (p=2): 4.5826

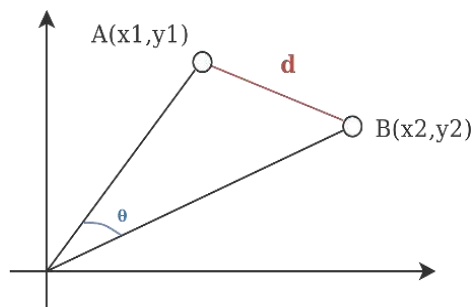
Minkowski Distance (p=10): 4.0004



Cosine Distance

Sebelum memahami Cosine Distance, sebaiknya mengenali Cosine similarity. Cosine similarity atau kesamaan kosinus sederhana merupakan kosinus dari sudut (θ) antara dua vektor. Pada gambar 4.5, terdapat dua vector A dan B. d merupakan jarak Euclid dari kedua vector ini dan $\cos \theta$ menunjukkan seberapa sama kedua vector tersebut. Manning, C. D. , R. P. , & S. H. (2008). Introduction to Information Retrieval. Cambridge University Press. Han, J., Kamber, M., & Pei, J. (2011). Data Mining. Concepts and Techniques, 3rd Edition (The Morgan Kaufmann Series in Data Management Systems).

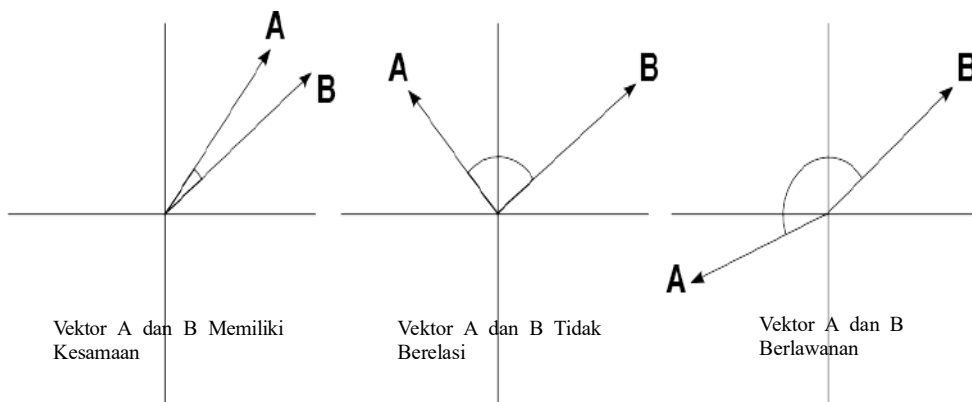
Semakin kecil sudut θ , maka semakin mirip kedua vector dan semakin dekat jaraknya. Jarak ini merupakan alternatif dari jarak Euclidean pada data dengan dimensi tinggi.



Gambar 4.5 Ilustrasi Cosine distance

Sebagaimana yang kita ketahui, nilai $\cos$ berada pada jarak -1 sampai 1 . Oleh karena itu maka ada beberapa kemungkinan yang terjadi diantaranya:

- Ketika $\text{similarity}(A,B) = 0$, dimana $\theta = 90^\circ$ berarti kesamaan antara vector tersebut sangat kecil bahkan dapat dikatakan tidak berelasi.
- Ketika $\text{similarity}(A,B) = -1$, dimana $\theta = 180^\circ$ berarti vektor tersebut sangat berbeda atau berlawanan.
- Ketika $\text{similarity}(A,B) = 1$, dimana $\theta = 0^\circ$ berarti vektor tersebut sama atau sangat mirip.

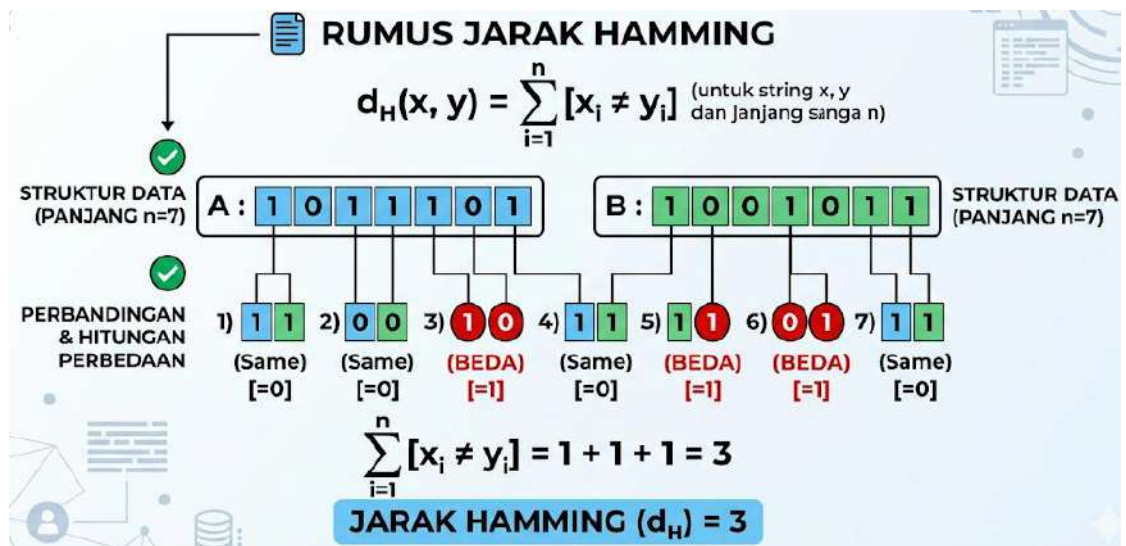


Salah satu kelemahan utama dari kesamaan kosinus merupakan besaran vektor tidak diperhitungkan, hanya arahnya. Dalam praktiknya, perbedaan nilai tersebut tidak sepenuhnya diperhitungkan. Jika Pembaca mengambil sistem pemberi rekomendasi, misalnya, kesamaan kosinus tidak memperhitungkan perbedaan dalam skala penilaian antara pengguna yang berbeda.

Persamaan cosine similarity dan hubungannya dengan cosine distance ditunjukkan oleh persamaan berikut $A \cdot B$

Cosine distance cocok digunakan pada data berdimensi tinggi dan ketika besaran vektor tidak penting. Untuk analisis teks, ukuran ini cukup sering digunakan ketika data diwakili oleh jumlah kata. Misalnya, ketika sebuah kata lebih sering muncul dalam satu dokumen daripada dokumen lainnya, ini tidak berarti bahwa satu dokumen lebih terkait dengan kata itu. Bisa jadi dokumen memiliki panjang yang tidak rata dan besarnya hitungan kurang penting.

Hamming Distance



Gambar 4.6 Jarak Hamming

Jarak Hamming merupakan banyaknya nilai yang berbeda antara dua vektor. Ini biasanya digunakan untuk membandingkan dua string biner dengan panjang yang sama. Ini juga dapat digunakan untuk string untuk membandingkan seberapa mirip satu sama lain dengan menghitung jumlah karakter yang berbeda satu sama lain. Hamming, R. W. (1950). Error detecting and error correcting codes. . Bell System Technical Journal, 29(2), 147–160. Navarro, G. (2001). A guided tour to approximate string matching. ACM Computing Surveys (CSUR), 33(1), 31–88.

Seperti yang Pembaca duga, jarak hamming sulit digunakan jika dua vektor tidak memiliki panjang yang sama. Pembaca ingin membandingkan vektor dengan panjang yang sama satu sama lain untuk memahami posisi mana yang tidak cocok.

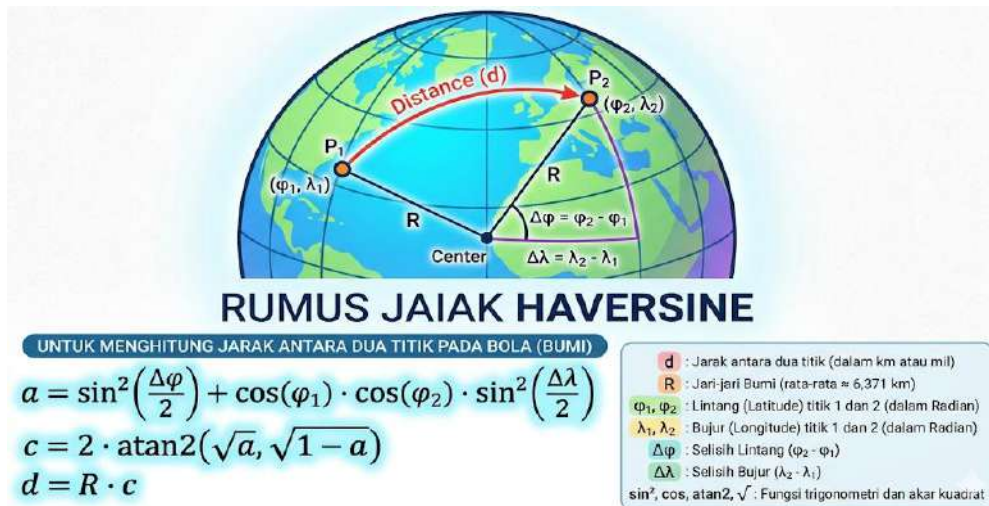
Selain itu, nilai sebenarnya tidak diperhitungkan selama nilainya berbeda atau sama. Oleh karena itu, tidak disarankan untuk menggunakan pengukur jarak ini jika besarannya merupakan ukuran yang penting.

Kasus penggunaan umum termasuk koreksi / deteksi kesalahan saat data dikirim melalui jaringan komputer. Ini dapat digunakan untuk menentukan jumlah bit yang terdistorsi dalam kata biner sebagai cara untuk memperkirakan kesalahan. Selain itu, Pembaca juga dapat menggunakan jarak Hamming untuk mengukur jarak antar variabel kategori.

Haversine Distance

Jarak Haversine merupakan jarak antara dua titik pada sebuah bola berdasarkan bujur dan lintangnya. Ini sangat mirip dengan jarak Euclidean yang menghitung garis terpendek antara dua titik. Perbedaan utamanya merupakan tidak ada garis lurus yang

memungkinkan karena asumsi di sini merupakan bahwa kedua titik tersebut berada pada sebuah bola. Inman, J. (1835). Navigation and Nautical Astronomy: For the Use of British Seamen. . Sinnott, R. W. (1984). Virtues of the Haversine. Sky and Telescope, 68(2), 159.



Gambar 4.7 Ilustrasi Haversine distance

Salah satu kelemahan dari alat ukur jarak ini merupakan diasumsikan bahwa titik-titik tersebut terletak pada sebuah bola . Dalam praktiknya, hal ini jarang terjadi karena, misalnya, bumi tidak bulat sempurna yang dapat mempersulit perhitungan dalam kasus-kasus tertentu. Sebaliknya, akan menarik untuk melihat ke arah jarak Vincenty yang mengasumsikan sebagai ellipsoid.

Seperti yang Pembaca duga, jarak Haversine sering digunakan dalam navigasi. Misalnya, Pembaca dapat menggunakannya untuk menghitung jarak antara dua negara saat terbang di antara mereka. Perhatikan bahwa itu jauh lebih tidak cocok jika jaraknya sendiri sudah tidak terlalu jauh. Lengkungan tidak akan berdampak besar.

Rumus Haversine digunakan untuk menghitung jarak lingkaran besar (great-circle distance) antara dua titik di permukaan bola berdasarkan koordinat lintang (latitude) dan bujur (longitude). Ini sangat penting dalam navigasi karena jarak terpendek antar dua titik di Bumi bukanlah garis lurus (karena Bumi melengkung), melainkan busur lingkaran besar.

Langkah-langkah Perhitungan:

1. Hitung Selisih Koordinat (Konversi ke Radian):
 - dlat (Selisih Lintang) = lat2 - lat1 (Konversikan dari derajat ke radian: dlat x $\pi/180$)

- d_{lon} (Selisih Bujur) = $lon2 - lon1$ (Konversikan dari derajat ke radian: $d_{lon} \times \pi / 180$)
 - Konversikan juga $lat1$ dan $lat2$ mentah ke radian.
2. Hitung Nilai 'a' (Perubahan Jarak Horizontal):
- Ini merupakan bagian utama dari formula Haversine:
- $$a = \sin^2 \left(\frac{dlat}{2} \right) + \cos(lat1) \times \cos(lat2) \times \sin^2 \left(\frac{dlon}{2} \right)$$
3. Hitung Nilai 'c' (Jarak Sudut dalam Radian):
- Ini menghitung sudut antara dua titik di pusat Bumi:
- $$c = 2 \times \text{atan2}(\sqrt{a}, \sqrt{1-a})$$
4. Hitung Jarak Akhir 'd':
- Kalikan jarak sudut dalam radian dengan jari-jari rata-rata Bumi ($R \approx 6,371$ km):
- $$d = R \times c$$

4.3 Implementasi K-NN

Implementasi KNN sangatlah sederhana. Proses pelatihan pada KNN hanya memasukkan atau menyimpan semua data latih berserta labelnya kedalam sebuah variabel yang akan digunakan untuk evaluasi (baris 13 dan 14).

```
1  import numpy as np
2  from collections import Counter
3
4  def euclidean_distance(x1, x2):
5      return np.sqrt(np.sum((x1 - x2)**2))
6
7  class KNN:
8
9      def __init__(self, k=3):
10         self.k = k
11
12     def fit(self, X, y):
13         self.X_train = X
14         self.y_train = y
15
16     def predict(self, X):
17         y_pred = [self._predict(x) for x in X]
18         return np.array(y_pred)
19
20     def _predict(self, x):
21         # 1. hitung jarak antara X ke semua data train
22         jarak_sample = [euclidean_distance(x, x_train) for x_train in self.X_train]
23         # 2. Urutkan dan Ambil K terdekat
24         k_idx = np.argsort(jarak_sample)[:self.k]
25         # 3. ambil label semua jarak terdekat
26         k_neighbor_labels = [self.y_train[i] for i in k_idx]
27         # 4. lakukan voting terbanyak
28         most_common = Counter(k_neighbor_labels).most_common(1)
29         return most_common[0][0]
30
```

Sebuah algoritma KNN membutuhkan sebuah hyperparameter K yang digunakan untuk proses prediksi. Pada proses prediksi, kita akan mengambil K jarak terdekat dari titik uji berdasarkan fungsi jarak. Selanjutnya baru dikembalikan hasil voting kelas terbanyak.

Pada pengujian digunakan dataset Iris (baris 8) dan evaluasi digunakan akurasi (baris 4-6). Pada baris ke 14, penyusun menginisialisasi model dengan K=3. Lalu, baris ke 15 model dilatih berdasarkan data yang di dapat pada data splitting pada baris 11.

```

1  from sklearn import datasets
2  from sklearn.model_selection import train_test_split
3
4  def accuracy(y_true, y_pred):
5      accuracy = np.sum(y_true == y_pred) / len(y_true)
6      return accuracy
7
8  iris = datasets.load_iris()
9  X, y = iris.data, iris.target
10
11 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
12
13 k = 3
14 clf = KNN(k=k)
15 clf.fit(X_train, y_train)
16 predictions = clf.predict(X_test)
17 print("Akurasi KNN: ", accuracy(y_test, predictions))

```

Akurasi KNN: 0.9333333333333333

Pada baris ke 16 penyusun melakukan prediksi terhadap data test sehingga didapatkan akurasi sebesar 93.33%

4.4 Keunggulan serta Kelemahan K-NN

KNN merupakan salah satu algoritma (model) pembelajaran mesin yang bersifat nonparametrik. Pembahasan mengenai model parametrik dan model nonparametrik bisa menjadi artikel sendiri, namun secara singkat, definisi model nonparametrik merupakan model yang tidak mengasumsikan apa-apa mengenai distribusi instance di dalam dataset. Model nonparametrik biasanya lebih sulit diinterpretasikan, namun salah satu kelebihanannya merupakan garis keputusan kelas yang dihasilkan model tersebut bisa jadi sangat fleksibel dan nonlinear.

Selain itu, KNN mudah dipahami dan juga mudah diimplementasikan. Untuk mengklasifikasi instance x menggunakan KNN, kita cukup mendefinisikan fungsi untuk menghitung jarak antar-instance, menghitung jarak x dengan semua instance lainnya berdasarkan fungsi tersebut, dan menentukan kelas x sebagai kelas yang paling banyak muncul dalam k instance terdekat.

Adapun kekurangan-kekurangan dari KNN merupakan:

- Perlu menentukan hyperparameter K (jumlah tetangga terdekat) dan hyperparameter ini mempengaruhi kinerja KNN
- Tidak menangani nilai hilang (missing value) secara implisit. Jika terdapat nilai hilang pada satu atau lebih variabel dari suatu instance, perhitungan jarak instance tersebut dengan instance lainnya menjadi tidak dapat dilakukan. Bagaimana coba, menghitung jarak dalam ruang 3-dimensi jika salah satu dimensi hilang? Karenanya, sebelum menerapkan kNN kerap dilakukan imputasi untuk mengisi nilai-nilai hilang yang ada pada dataset. Contoh teknik imputasi yang paling umum merupakan mengisi nilai hilang pada suatu variabel dengan nilai rata-rata variabel tersebut (mean imputation).
- Sensitif terhadap data pencilan (outlier). KNN bisa jadi sangat fleksibel jika k kecil. Fleksibilitas ini mengakibatkan kNN cenderung sensitif terhadap data pencilan, khususnya pencilan yang terletak di “tengah-tengah” kelas yang berbeda.
 - Rentan terhadap variabel yang non-informatif. Meskipun kita telah men-standard-kan rentang variabel, KNN tetap tidak dapat mengetahui variabel mana yang signifikan dalam klasifikasi dan mana yang tidak.
 - Rentan terhadap dimensionalitas yang tinggi Berbagai permasalahan yang timbul dari tingginya dimensionalitas (data yang memiliki banyak variabel/feature) menimpa sebagian besar algoritma pembelajaran mesin, dan kNN merupakan salah satu algoritma yang paling rentan terhadap tingginya dimensionalitas. Hal ini karena semakin banyak dimensi, ruang yang bisa ditempati instance semakin besar, sehingga semakin besar pula kemungkinan bahwa nearest neighbour dari suatu instance sebetulnya sama sekali tidak “near”.

4.5 Studi Kasus – Klasifikasi Buah

Pada studi kasus ini, penyusun akan mencoba melakukan klasifikasi buah berdasarkan ciri-cirinya. Langkah pertama merupakan membaca file dataset.

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4
5 buah=pd.read_table('fruit_data_with_colors.txt')
6 buah.head()
```

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79

Memahami dataset

Pada dataset ini terdapat 7 kolom yaitu fruit label, fruit_name, fruit_subtype, mass, width, height dan color_score. Dataset ini teriti atas 59 data.

```
1 buah.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59 entries, 0 to 58
Data columns (total 7 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   fruit_label      59 non-null    int64
1   fruit_name       59 non-null    object
2   fruit_subtype    59 non-null    object
3   mass             59 non-null    int64
4   width            59 non-null    float64
5   height           59 non-null    float64
6   color_score      59 non-null    float64
dtypes: float64(3), int64(2), object(2)
memory usage: 3.4+ KB
```

Pada dataset ini, terdapat kolom fruit label dan fruit name, dimana kedua kolom itu memiliki nilai sama dimana semua buah apple bernilai 1, mandarin bernilai 2, orange bernilai 3 dan orange bernilai 4.

```

1 #Kroscek hubungan label dan nama
2 label_mapping = dict(zip(buah.fruit_label.unique(), buah.fruit_name.unique()))
3 label_mapping

```

```
{1: 'apple', 2: 'mandarin', 3: 'orange', 4: 'lemon'}
```

Pembersihan Data dan Feature Engineering

Langkah pertama pembersihan data merupakan pengecekan missing data.

Pada dataset ini tidak ditemukan ada data yang hilang.

```

1 #Checking for NaN values
2 buah.isna().sum()

```

```

fruit_label    0
fruit_name      0
fruit_subtype   0
mass            0
width           0
height          0
color_score     0
dtype: int64

```

Selanjutnya, karena fruit label dan fruit name merupakan fitur yang sama maka kita drop beberapa kolom. Pada akhirnya kita hanya menggunakan 5 kolom

```

1 buah = buah.drop(["fruit_name", "fruit_subtype"], axis=1)
2 buah.head()

```

	fruit_label	mass	width	height	color_score
0	1	192	8.4	7.3	0.55
1	1	180	8.0	6.8	0.59
2	1	176	7.4	7.2	0.60
3	2	86	6.2	4.7	0.80
4	2	84	6.0	4.6	0.79

Agar meningkatkan performa maka kita melakukan scaling ke semua fitur kecuali label

```

1 from sklearn import preprocessing
2 from sklearn import pipeline
3 from sklearn import model_selection
4 minmax_scaler = preprocessing.MinMaxScaler()
5 fruits_minmax_scaled = minmax_scaler.fit_transform(
6     buah.drop(["fruit_label"], axis=1)
7 )

```

Setelah itu dataset tersebut dibagi menjadi data training dan data test. Pada kasus ini kita akan membuat 2 data training yaitu data training yang di scaling dan tidak. Hal ini bertujuan untuk membuktikan bagaimana pengaruh scaling terhadap performa, terutama untuk algoritma-algoritma berbasis distance

```

1 from sklearn.model_selection import train_test_split
2 from sklearn.neighbors import KNeighborsClassifier
3
4 fruits_df_scaled = pd.DataFrame(fruits_minmax_scaled, columns=buah.columns[3:])
5 fruits_df_scaled.head()
6
7 X=buah[['mass','width','height']]
8 Y=buah['fruit_label']
9 X_train,X_test,y_train,y_test=train_test_split(X,Y,random_state=0)
10 X_train_scaled, X_test_scaled, y_train_scaled, y_test_scaled = train_test_split(fruits_df_scaled,
11     buah["fruit_label"],
12     random_state=0)

```

Pelatihan dan evaluasi

Pertama-tama kita akan melakukan pelatihan pada dataset yang tidak di scaling. Kita akan menggunakan K=3 lalu melakukan training (baris 2) dan evaluasi (baris 3)

```

1 knn=KNeighborsClassifier(n_neighbors=3)
2 knn.fit(X_train,y_train)
3 knn.score(X_test,y_test)

```

0.5333333333333333

Tanpa scaling, model mendapatkan performa 53.33%, namun jika menggunakan data yang telah di scaling, performanya meningkat sebanyak 93.33%.

```

1 knn2=KNeighborsClassifier(n_neighbors=3)
2 knn2.fit(X_train_scaled,y_train_scaled)
3 knn2.score(X_test_scaled,y_test_scaled)

```

0.9333333333333333

Sebagai ilustrasi bandingkan misalkan data kita dengan fitur A yang bernilai 1 sampai 10 dan fitur B bernilai 1 sampai 100. Titik x1 bernilai (1,10) dan x2 bernilai (2,20). Jika dilihat perbedaannya fitur A berbeda sebanyak 1 poin namun fitur B berbeda sebanyak 10 poin. Namun jika di scaling maka fitur A dan B memiliki perbedaan yang sama.

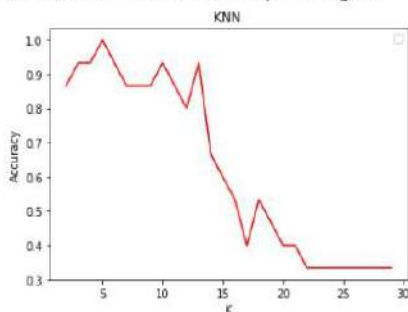
Langkah selanjutnya merupakan pemilihan K terbaik. Sederhananya tinggal melakukan perulangan lalu dipetakan kedala grafis. Berdasarkan hasil percobaan, K terbaik merupakan 5.ss

```

1 # mencari K Terbaik
2 K = np.arange(2,30)
3
4 scores = []
5
6 for k in K:
7     model = KNeighborsClassifier(n_neighbors=k).fit(X_train_scaled,y_train_scaled)
8     score = model.score(X_test_scaled,y_test_scaled)
9     scores.append(score)
10
11 plt.plot(K,scores, c = "red")
12 plt.title("KNN ")
13 plt.ylabel("Accuracy")
14 plt.xlabel("K")
15 plt.legend()
16 plt.show()

```

No handles with labels found to put in legend.



4.6 K-Dimensional Tree

K-Dimensional Tree (KD-Tree) merupakan salah satu solusi atas permasalahan performa yang dimiliki oleh KNN dimana, KNN akan membandingkan data test ke semua data latih. Jika data latih hanya 10 atau 100 tidak jadi masalah untuk melakukan ini, namun jika data berjumlah sangat banyak dan berdimensi tinggi maka hal ini akan menjadi permasalahan. Salah satu solusi yang digunakan merupakan menggunakan KD Tree.

KD-Tree merupakan sebuah pohon biner (Binary Tree) yang telah di modifikasi sedemikian rupa sehingga dapat melakukan pencarian multidimensi pada data. Prinsip kerjanya merupakan membuat partisi-partisi (sekat-sekat), lalu partisi tersebut digunakan untuk mengatur data-data sehingga mudah dilakukan pencarian. Singkatnya, dengan algoritma ini kita tidak perlu membandingkan dengan semua data, cukup dengan beberapa kandidat saja. Perbedaan KD-Tree dengan pohon biner merupakan setiap node leaf merupakan titik berdimensi K. Sebagai ilustrasi, data seorang karyawan dikatakan berdimensi 5 ($K=5$) maka data karyawan tersebut berisi 5 fitur yang tu no Karyawan, Umur, Gaji, Tahun Masuk, Jarak rumah.

Pada Algoritma ini, setiap node yang bukan leaf akan menghasilkan sebuah hyperplane yang akan memisahkan node-node yang berada pada subtree kanan. Hyperplane inilah yang berfungsi sebagai partisi node. Adapun langkah-langkah yang KD-Tree merupakan sebagai berikut

1. Pilih atribut/fitur yang digunakan sebagai pembatas. Salah satu cara menggunakan axis ($\text{axis} = \text{depth} \bmod k$). Penggunaan mod pada axis bertujuan agar fitur dipilih secara bergantian, contoh data dengan 4 fitur maka pada depth tree 1 dipilih fitur 1, depth tree-2 dipilih fitur kedua, dan seterusnya. Cara lain seperti mencari fitur dengan range paling lebar juga menjadi opsi yang baik
2. Setelah itu data yang berada pada axis yang dipilih pada step 1 diurutkan, lalu hitung rata-rata atau media sebagai pivot value
3. Buat node dan pecah data menjadi dua subtree sesuai aturan binary tree. Data yang nilainya kecil dari pivot value berada di partisi/sub- tree kiri dan nilai yang besar dari pivot value berada disebelah kanan. Ulangi step 1 sampai kedalaman yang ditentukan.

Sebagai contoh kita akan membuat KD-Tree dari data 2 dimensi berikut ini: (4, 5), (7, 11), (2, 7), (6, 12), (9, 1), (1, 4), (10, 19)

BAB V

ROBOT NAVIGATION DENGAN REINFORCEMENT LEARNING

5.1 Penjelasan: Robot Navigation dengan Reinforcement Learning

Dasar Konsep RL dalam Navigasi

Reinforcement Learning (RL) merupakan cabang dari Machine Learning di mana Agent (robot) belajar bagaimana berperilaku di lingkungan tertentu untuk memaksimalkan akumulasi hadiah (Reward).

Model Markov Decision Process (MDP)

Navigasi robot dimodelkan sebagai MDP, yang terdiri dari empat elemen utama:

State (S): Kondisi lingkungan yang diamati oleh robot. Dalam navigasi, ini seringkali merupakan kombinasi dari:

- Koordinat robot (x, y, θ).
- Pembacaan sensor (misalnya, jarak ke rintangan terdekat yang dilaporkan oleh Lidar).
- Action (A): Keputusan yang dapat diambil robot, misalnya: (Maju Cepat, Putar 10 Derajat Kiri), (Berhenti, Putar 90 Derajat Kanan).
- Reward (R): Umpan balik numerik segera setelah mengambil aksi A di state S.
- Policy (π): Strategi yang dipelajari, yang menentukan aksi mana yang harus diambil robot di state tertentu, $\pi: S \rightarrow A$.

Markov Decision Process (MDP). MDP merupakan kerangka kerja matematis yang digunakan untuk memodelkan pengambilan keputusan di situasi di mana hasilnya bersifat sebagian acak dan sebagian di bawah kendali pembuat keputusan (robot).

Dalam konteks navigasi robot, MDP berfungsi sebagai "peta logika" agar robot bisa mencapai tujuan tanpa menabrak. Berikut merupakan penjelasan mendalam dari empat elemen utamanya:

1. State (S): Kondisi Lingkungan

State merupakan potret lengkap situasi robot pada satu titik waktu tertentu. Robot menggunakan data ini untuk memahami "Di mana saya sekarang?".

- Koordinat (x, y, θ): Posisi absolut dalam ruangan dan arah hadap robot.
- Sensor (Lidar/Ultrasonik): Informasi jarak terhadap rintangan. Misalnya, "Ada tembok di depan sejauh 20 cm".

2. Action (A): Keputusan Robot

Action merupakan serangkaian perintah motorik yang bisa dieksekusi robot. Robot harus memilih satu aksi dari daftar kemungkinan yang ada.

- Contoh Aksi: Maju dengan kecepatan (0.5 m/s), belok kanan 30^0 , atau melakukan pengereman darurat.
- Aksi ini akan mengubah State lama (S_t) menjadi State baru (S_{t+1}).

3. Reward (R): Umpan Balik Numerik

Reward merupakan cara kita memberi tahu robot apakah tindakannya baik atau buruk. Robot tidak tahu "etika", ia hanya mengejar angka maksimal.

- Reward Positif (+100): Jika robot berhasil mencapai titik tujuan.
- Reward Negatif (-50): Jika robot mendekati zona bahaya atau menabrak rintangan.
- Penalti Kecil (-1): Diberikan setiap detik agar robot terpacu untuk mencari jalan tercepat (tidak berputar-putar tanpa tujuan).

4. Policy (π): Strategi Keputusan

Policy (Kebijakan) merupakan "otak" atau fungsi pemetaan. Jika State merupakan pertanyaannya, maka Policy merupakan jawabannya.

- Fungsi (π) $S \rightarrow A$: Jika robot melihat tembok di depan (State), maka Policy akan memerintahkan untuk belok (Action).
- Tujuan utama dari proses belajar AI merupakan menemukan Optimal Policy (π^*), yaitu strategi yang menghasilkan total reward tertinggi dalam satu perjalanan.

Hubungan Antar Elemen (Siklus MDP)

MDP bekerja dalam sebuah siklus yang terus berulang:

1. Robot mengamati State (S).
2. Berdasarkan Policy (π), robot memilih Action (A).
3. Lingkungan berubah, robot menerima Reward (R) dan berpindah ke State Baru (S).
4. Ulangi hingga tujuan tercapai.

Menggunakan probability Markov

Disebut Markov karena ia mengikuti Markov Property: Masa depan hanya bergantung pada kondisi saat ini (S_t), bukan pada sejarah masa lalu. Artinya, untuk memutuskan belok sekarang, robot tidak perlu ingat apa yang ia lakukan 10 menit yang lalu; ia hanya perlu melihat sensornya detik ini

Memvisualisasikan MDP dalam navigasi robot, kita akan membuat simulasi "Grid World". Dalam simulasi ini, robot berada di sebuah papan catur, tujuannya merupakan mencapai kotak target sambil menghindari kotak rintangan.

Berikut merupakan skrip Python sederhana menggunakan matplotlib untuk memvisualisasikan bagaimana robot bergerak berdasarkan Policy (π) yang dipelajarinya.

Skrip Simulasi: Robot Grid-World MDP

Python

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

# 1. DEFINISI LINGKUNGAN (Grid 5x5)
grid_size = 5
goal = (4, 4) # Target (Reward +10)
obstacle = (2, 2) # Rintangan (Reward -10)

# 2. STATE & ACTION
# State merupakan koordinat (row, col)
# Action: 0: Atas, 1: Bawah, 2: Kiri, 3: Kanan
actions = [(-1, 0), (1, 0), (0, -1), (0, 1)]

def get_reward(state):
    if state == goal:
        return 10
    elif state == obstacle:
        return -10
    else:
        return -1 # Penalti waktu agar robot cepat sampai

# 3. VISUALISASI
def plot_grid(robot_pos):
    grid = np.zeros((grid_size, grid_size))
    grid[goal] = 0.5 # Warna Target
    grid[obstacle] = -0.5 # Warna Rintangan

    plt.clear()
    plt.imshow(grid, cmap='RdYlGn', origin='upper')

    # Gambar Robot
    plt.scatter(robot_pos[1], robot_pos[0], color='blue', s=200, label='Robot')
    plt.text(goal[1], goal[0], 'GOAL', ha='center', va='center')
    plt.text(obstacle[1], obstacle[0], 'BOOM', ha='center', va='center')

    plt.xticks(range(grid_size))
    plt.yticks(range(grid_size))
    plt.grid(True)
    plt.pause(0.5)

# 4. SIMULASI JALANNYA ROBOT (Sesuai Policy Sederhana)
robot_pos = (0, 0) # Start
print("Robot mulai bergerak...")

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

for step in range(15):
    plot_grid(robot_pos)

    if robot_pos == goal:
        print("Target Tercapai!")
        break

    # LOGIKA POLICY ( $\pi$ ): Sederhana, selalu mendekati goal dan hindari rintangan
    row, col = robot_pos

    # Tentukan aksi terbaik (Sederhana: cari jarak terdekat ke goal)
    if row < goal[0] and (row + 1, col) != obstacle:
        robot_pos = (row + 1, col)
    elif col < goal[1] and (row, col + 1) != obstacle:
        robot_pos = (row, col + 1)
    else:
        # Menghindari rintangan jika berada di jalur langsung
```

```
robot_pos = (row, col + 1) if col < 4 else (row + 1, col)
plt.show()
```

Penjelasan MDP dalam Simulasi Ini:

1. State (S): Posisi robot di koordinat (row, col). Di setiap langkah, robot melihat sensornya: "Apakah saya di (2,2)? Oh tidak, itu rintangan."
2. Action (A): Robot memilih pindah ke kotak tetangga (Atas, Bawah, Kiri, Kanan).
3. Reward (R): * Jika robot melangkah ke kotak kosong, ia dapat -1. Ini melambangkan konsumsi baterai.
 - o Jika robot menabrak rintangan, ia dapat -10. Ini melambangkan kerusakan.
 - o Jika robot sampai di Goal, ia dapat +10.
4. Policy (π): Strategi yang memutuskan aksi. Dalam skrip di atas, Policy dibuat manual agar robot selalu mencari jalan terpendek tanpa menyentuh kotak (2,2).

Hal yang terjadi Saat Proses "Learning"

Dalam Reinforcement Learning (seperti Q-Learning), kita tidak menulis perintah if-else seperti di atas. Sebagai gantinya:

- Robot mencoba bergerak acak ribuan kali.
- Ia mencatat: "Saat saya di (2,1) dan bergerak kanan, saya dapat -10 (Menabrak)."
- Ia memperbarui Q-Value (Nilai Kualitas) untuk state tersebut menjadi rendah.
- Setelah ribuan percobaan, robot membentuk Policy otomatis yang selalu memilih jalan memutar menghindari rintangan tersebut mengubah skrip sebelumnya menjadi Q-Learning murni. Di sini, kita tidak lagi memberikan instruksi if-else kepada robot.

Robot akan memulai dengan "otak kosong" (Q-Table berisi nol). Ia akan melakukan eksplorasi acak, menabrak rintangan berkali-kali, dan perlahan membangun "kebijakan" (policy) berdasarkan pengalaman pahit dan manis yang ia alami.

Skrip Python: Robot Q-Learning (Tabular)

Python

```
import numpy as np
import random
import matplotlib.pyplot as plt

# 1. KONFIGURASI LINGKUNGAN
GRID_SIZE = 5
ACTIONS = [0, 1, 2, 3] # 0:Atas, 1:Bawah, 2:Kiri, 3:Kanan
ACTION_MAP = {0: (-1, 0), 1: (1, 0), 2: (0, -1), 3: (0, 1)}

# Inisialisasi Q-Table dengan nol [State_Row, State_Col, Action]
q_table = np.zeros((GRID_SIZE, GRID_SIZE, len(ACTIONS)))

# Parameter Belajar
learning_rate = 0.1
gamma = 0.9 # Fokus pada masa depan
epsilon = 1.0 # Eksplorasi awal (100% acak)
epsilon_decay = 0.99 # Mengurangi keacakan seiring waktu
episodes = 1000 # Robot akan mencoba 1000 kali

GOAL = (4, 4)
OBSTACLE = (2, 2)

# 2. PROSES PELATIHAN (TRAINING)
print("Robot sedang belajar... Harap tunggu.")

for episode in range(episodes):
    state = (0, 0) # Mulai dari pojok kiri atas

    while state != GOAL:
        # Strategi Epsilon-Greedy
        if random.uniform(0, 1) < epsilon:
            action = random.choice(ACTIONS) # Coba-coba (Exploration)
        else:
            action = np.argmax(q_table[state[0], state[1]]) # Gunakan ilmu (Exploitation)

        # Hitung posisi baru
        move = ACTION_MAP[action]
```

```

        next_state = (
            max(0, min(GRID_SIZE - 1, state[0] + move[0])),
            max(0, min(GRID_SIZE - 1, state[1] + move[1]))
        )

        # Tentukan Reward
        if next_state == GOAL:
            reward = 100
        elif next_state == OBSTACLE:
            reward = -100
        else:
            reward = -1 # Penalti setiap langkah (biaya baterai)

        # RUMUS BELLMAN (Pembaruan Q-Value)
        old_value = q_table[state[0], state[1], action]
        next_max = np.max(q_table[next_state[0], next_state[1]])

        # Update: Q = Q lama + LR * (Reward + Gamma * Max_Q_Masa_Depan - Q lama)
        new_value = old_value + learning_rate * (reward + gamma * next_max -
old_value)
        q_table[state[0], state[1], action] = new_value

        state = next_state
        if state == OBSTACLE: break # Robot hancur, ulangi episode

    epsilon *= epsilon_decay # Robot menjadi semakin pintar & fokus

print("Latihan selesai!")

```

#3. Visualisasi Hasil Belajar

Sekarang kita lihat apakah robot sudah pintar dengan menjalankan navigasi menggunakan Q-Table yang sudah terisi.

Python

```

def visualize_learned_policy():
    state = (0, 0)
    path = [state]

    print("\nMenjalankan robot dengan Policy hasil belajar:")
    while state != GOAL:
        action = np.argmax(q_table[state[0], state[1]])
        move = ACTION_MAP[action]
        state = (state[0] + move[0], state[1] + move[1])
        path.append(state)
        if state == OBSTACLE:
            print("Robot masih menabrak! Perlu lebih banyak latihan.")
            break

    # Gambar Jalur
    grid = np.zeros((GRID_SIZE, GRID_SIZE))
    grid[GOAL] = 1
    grid[OBSTACLE] = -1

    plt.imshow(grid, cmap='RdYlGn')
    rows, cols = zip(*path)
    plt.plot(cols, rows, marker='o', color='blue', linewidth=2, label='Jalur
Robot')
    plt.title("Jalur Terpintar Hasil Belajar Q-Learning")
    plt.show()
visualize_learned_policy()

```

Hal yang Dipelajari Robot

1. Awalnya Bodoh: Pada episode 1-100, robot akan sering masuk ke kotak (2,2) (BOOM) atau hanya berputar-putar di pojok karena epsilon masih tinggi.
2. Menemukan Target: Saat pertama kali secara tidak sengaja menyentuh GOAL (4,4), ia menerima reward +100. Rumus Bellman akan menyebarkan nilai positif ini ke kotak-kotak di sekitarnya.
3. Optimalisasi: Robot menyadari bahwa setiap langkah diberi -1. Maka, "jalan memutar" yang terlalu jauh akan memberikan total reward yang lebih kecil daripada jalan pintas yang aman.
4. Policy Terbentuk: Akhirnya, Q-Table akan berisi angka-angka yang "mengarahkan" robot. Misalnya, di kotak (2,1), nilai Q untuk aksi "Kanan" akan sangat rendah karena menabrak rintangan, sehingga AI akan memilih "Bawah" atau "Atas".

Analogi Bisnis untuk Unit Usaha

Proses ini persis seperti melatih sistem stok Unit Usaha Pembaca.

- State: Level stok saat ini.
- Reward Negatif: Biaya simpan dan denda stok kosong.
- Update: Setiap hari AI "melihat kembali" apakah keputusan pesanan kemarin menguntungkan. Jika ia rugi, ia menurunkan nilai Q untuk aksi tersebut di kondisi stok tersebut.

Jika robot Pembaca tidak hanya bergerak di grid 5 x 5, tapi di gudang raksasa dengan ribuan koordinat, suhu yang berubah-ubah, dan rintangan yang bergerak. Menggunakan Q-Table manual akan menjadi mustahil karena tabelnya akan berukuran raksasa (triliunan baris), sebuah masalah yang disebut *Curse of Dimensionality*.

Di sinilah Deep Q-Network (DQN) masuk sebagai penyelamat dengan mengganti "Buku Tabel" menjadi "Otak Buatan".

1. Dari Tabel ke Prediksi (Function Approximator)

Pada Q-Learning biasa, kita harus mencari nilai di dalam sel tabel:

"Cari baris posisi (2,2), lihat kolom aksi 'Maju', ambil nilainya."

Pada DQN, kita menggunakan Neural Network. Kita tidak lagi menyimpan nilai, tapi melatih model untuk menebak nilai Q.

- Input: Kondisi lingkungan (misal: gambar kamera robot atau koordinat x,y).
- Output: Prediksi nilai Q untuk setiap aksi yang mungkin dilakukan.

2. Mengapa DQN Lebih Unggul daripada Generalisasi

Inilah kelebihan utama DQN dibandingkan tabel manual: Generalisasi.

- Q-Table: Jika robot berada di koordinat yang meleset 1 cm saja dari yang ada di tabel, ia akan bingung karena data tersebut dianggap "baru".
- DQN: Karena menggunakan Neural Network, ia bisa mengenali pola. Jika ia tahu bahwa "menabrak tembok di koordinat A" itu buruk, ia secara otomatis paham bahwa "menabrak tembok di koordinat yang mirip dengan A" juga buruk.

DQN tidak perlu mengunjungi setiap inci ruangan untuk menjadi pintar; ia cukup memahami fitur dari rintangan tersebut.

3. Dua Komponen Rahasia Stabilitas DQN

Neural Network sebenarnya sangat tidak stabil jika dipaksa belajar secara langsung dari robot yang bergerak. Google DeepMind menemukan dua trik agar DQN berhasil:

- Experience Replay (Buku Kenangan):

Robot menyimpan pengalamannya dalam memori. Saat belajar, ia tidak hanya belajar dari kejadian detik ini, tapi mengambil sampel acak dari masa lalu. Ini mencegah AI "lupa" cara menghindari rintangan lama saat ia fokus belajar rintangan baru.

- Target Network (Kompas yang Stabil):

DQN menggunakan dua otak. Satu otak untuk belajar terus-menerus, dan satu otak "target" yang diperbarui secara perlahan. Ini mencegah nilai prediksi meloncat-loncat secara liar yang bisa membuat robot kehilangan arah.

4. Analogi Sederhana

- Q-Learning (Tabel): Seperti menghafal jawaban dari 1.000 soal ujian satu per satu. Jika soalnya diganti sedikit, Pembaca gagal.
- DQN (Neural Network): Seperti mempelajari rumus matematikanya. Apa pun angka yang keluar di soal ujian, Pembaca bisa menghitung jawabannya karena Pembaca paham logikanya.

Penerapan pada Kasus Gudang

1. Jika dalam sistem stok terdapat 1.000 jenis barang, maka:
 - Q-Table: Akan membutuhkan tabel dengan baris sebanyak kombinasi stok semua barang (hampir mustahil dihitung).
 - DQN: Cukup memasukkan angka stok 1.000 barang ke dalam Neural Network, dan AI akan memberikan satu rekomendasi aksi: "Pesan Barang A dan C, Diam untuk sisanya."

2. Fungsi Nilai (Q-Function)

Tujuan utama RL merupakan memperkirakan Q-Function ($Q(S, A)$). $Q(S, A)$ merupakan perkiraan total hadiah diskonto yang diharapkan yang akan diterima robot jika mengambil aksi A pada state S dan mengikuti policy terbaik setelahnya.

$$Q(S, A) = R + (\gamma \max_{A'} Q(S', A'))$$

(Persamaan Bellman Sederhana)

Robot akan selalu memilih aksi yang menghasilkan nilai Q tertinggi di state saat ini.

Q-Function (Fungsi Nilai Q) merupakan inti dari kecerdasan buatan pada sistem Reinforcement Learning. Jika kita menganggap Policy sebagai strategi, maka Q-Function merupakan "kalkulator" yang digunakan untuk menentukan strategi tersebut.

Berikut merupakan penjelasan mendalam mengenai komponen dan cara kerja Q-Function:

1. Apa itu Nilai Q?

Nilai Q (Q-Value) merupakan angka yang menunjukkan seberapa menguntungkan suatu aksi jika diambil dalam kondisi tertentu.

- Q singkatan dari Quality (Kualitas).
- Nilai ini bukan sekadar hadiah instan yang didapat sekarang, melainkan estimasi total kekayaan yang akan dikumpulkan robot dari sekarang hingga akhir tugasnya.

2. Membedah Persamaan Bellman

Robot menggunakan Persamaan Bellman untuk memperbarui pengetahuannya. Mari kita bedah rumusnya agar lebih mudah dipahami:

$$Q(S, A) = R + \gamma \max_{A'} Q(S', A')$$

- $Q(S, A)$: Nilai kualitas aksi A di kondisi S saat ini.
- R (Immediate Reward): Hadiah langsung yang didapat detik ini setelah bergerak.
- γ (Gamma/Discount Factor): Faktor diskon (biasanya 0.9 atau 0.95). Ini menentukan seberapa peduli robot terhadap masa depan. Jika γ besar, robot sangat visioner; jika kecil, robot menjadi "rabun jauh" (hanya peduli sekarang).
- $\max_{A'} Q(S', A')$: Estimasi nilai terbaik yang bisa didapat di masa depan (pada state berikutnya S').

Analogi Sederhana:

Bayangkan Pembaca sedang bermain catur. Langkah memakan pion lawan mungkin memberi hadiah kecil (R) sekarang, tetapi persamaan Bellman memberitahu robot: "Jangan makan pion itu, karena posisi barumu (S') nanti akan sangat lemah dan membuatmu kalah di masa depan."

3. Cara Robot Mengambil Keputusan

Di setiap langkah, robot akan melakukan pencarian di "buku otaknya" (Q-Table atau Neural Network) untuk melihat semua kemungkinan aksi:

1. Lihat State S (Misal: Stok sisa 5).
2. Cek nilai Q untuk setiap aksi:
 - Aksi "Diam": $Q = 10$
 - Aksi "Pesan": $Q = 85$
3. Robot akan selalu memilih aksi dengan Q tertinggi (dalam hal ini, "Pesan").
4. "Total Hadiah Diskonto"

Istilah "Diskonto" (Discounted) digunakan karena nilai di masa depan dianggap sedikit lebih tidak pasti dibandingkan hadiah di tangan saat ini. Ini mirip dengan prinsip ekonomi: Rp1.000.000 hari ini lebih berharga daripada Rp1.000.000 sepuluh tahun lagi.

Dengan adanya diskonto (γ), robot akan berusaha mencapai target secepat mungkin karena menunda-nunda hanya akan membuat nilai hadiah masa depan "menyusut".

Hasil Logika Robot

Q-Function mengubah masalah navigasi atau stok yang kompleks menjadi masalah matematika sederhana: Mencari Angka Terbesar. Robot tidak perlu "berpikir" secara filosofis; ia cukup mengikuti angka Q yang paling tinggi untuk menjamin kesuksesan jangka panjang

simulasikan bagaimana "otak" robot mulai terisi. Kita akan menggunakan contoh super sederhana: sebuah lorong dengan 3 kotak, di mana robot mulai di kotak paling kiri dan target ada di kotak paling kanan.

Skenario Simulasi

- Peta: [A] --- [B] --- [GOAL]
- Reward (R): Hanya diberikan saat mencapai GOAL (+100).
Langkah lainnya = 0.
- Discount Factor (γ): 0.9.
- Kondisi Awal: Semua nilai Q di "otak" robot merupakan 0.

Langkah 1: Robot bergerak dari A ke B

Robot belum tahu apa-apa, ia bergerak acak dari kotak A ke kotak B.

- Reward (R): 0 (karena belum sampai goal).
- Max Q masa depan: Robot melihat ke depan (ke arah Goal), tapi karena semua masih 0, maka $Q(S', A') = 0$.
- Update: $Q(A, \text{ke B}) = 0 + 0.9(0) = 0$.
- Status: Robot masih "buta". Nilai Q di kotak A tetap 0.

Langkah 2: Robot bergerak dari B ke GOAL (Momen Krusial!)

Ini merupakan momen pertama kali robot menyentuh target.

- Reward (R): +100 (Hadiah instan!).
- Max Q masa depan: Karena GOAL merupakan titik akhir, tidak ada aksi setelahnya, maka nilai masa depan = 0.
- Rumus Bellman:

$$Q(B, \text{ ke Goal}) = R + \gamma (0)$$

$$Q(B, \text{ ke Goal}) = 100 + 0.9(0) = 100$$

- Status: Sekarang kotak B punya nilai! Robot belajar bahwa "Berada di B dan pergi ke kanan itu sangat bagus."

Langkah 3: Episode Baru, Robot kembali ke A

Robot mulai lagi dari awal, tapi sekarang "otaknya" sudah punya satu catatan:

$$Q(B, \text{ ke Goal}) = 100.$$

Robot bergerak lagi dari A ke B. Mari kita hitung apa yang terjadi sekarang:

- Reward (R): \$ (belum sampai goal).
- Max Q masa depan: Robot melihat ke kotak B. Di sana ada nilai 100!
- Rumus Bellman:

$$Q(A, \text{ ke B}) = R + \gamma (\text{Nilai Terbaik di B})$$

$$Q(A, \text{ ke B}) = 0 + 0.9(100) = 90$$

- Status: Sesuatu yang ajaib terjadi. Meskipun kotak A jauh dari Goal, ia sekarang punya nilai 90.

Hasil Akhir Simulasi

Setelah dua kali perjalanan, isi "otak" robot (Q-Table) berubah menjadi:

Posisi	Aksi	Nilai Q	Maknanya
Kotak A	Pergi ke B	90	"Kotak ini bagus karena dekat tempat bagus"
Kotak B	Pergi ke Goal	100	"Kotak ini liar biasa bagus karena langsung dapat hadiah"

Efek "Riak Air"

Nilai hadiah akan menyebar mundur dari lokasi GOAL ke kotak-kotak sebelumnya seperti riak air di kolam.

- Kotak yang paling dekat dengan Goal akan punya nilai Q tertinggi.
- Kotak yang semakin jauh akan punya nilai Q yang semakin kecil (karena dikalikan γ terus-menerus).

Inilah alasan mengapa robot tidak perlu "melihat" Goal untuk tahu ke mana harus pergi. Ia cukup melihat kotak tetangganya dan memilih yang Nilai Q-nya paling tinggi.

buktikan mengapa fenomena ini disebut "Backward Induction" atau belajar secara terbalik. Logika ini merupakan alasan mengapa AI bisa menjadi sangat cerdas dalam merencanakan masa depan.

1. Bukti AI Belajar Secara Terbalik

Secara intuitif, manusia berpikir maju: "Saya pesan barang hari ini, supaya besok stok aman." Namun, AI (lewat algoritma Reinforcement Learning) bekerja dari arah sebaliknya:

1. Momen Akhir (Kenyataan): AI mengalami kerugian karena kehabisan stok (Stokout). Ia mendapat Reward Negatif yang besar.
2. Satu Langkah Mundur: AI melihat ke kondisi tepat sebelum stok habis. Ia menandai kondisi stok rendah tersebut dengan Nilai Q yang buruk.
3. Dua Langkah Mundur: AI mencari aksi apa yang seharusnya dilakukan agar tidak masuk ke kondisi "stok rendah" tersebut. Ia menemukan bahwa aksi "Pesan Barang" di waktu sebelumnya bisa menghindari nilai buruk itu.
4. Hasil Akhir: Nilai positif dari keberhasilan (mendapat untung/stok aman) mengalir mundur ke hari-hari sebelumnya.

Iniilah mengapa pada simulasi sebelumnya, kotak B mendapat nilai dulu, baru kemudian kotak A. AI tidak tahu apa yang harus dilakukan di awal (A) sampai ia tahu apa hasilnya di akhir (B).

2. Menerapkan Logika "Riak Air" pada Reorder Point (ROP)

Dalam manajemen gudang tradisional, Reorder Point (ROP) dihitung dengan rumus statis. Namun, dengan logika MDP/RL, ROP menjadi dinamis karena AI mempertimbangkan "Riak" biaya di masa depan.

Menghitung menggunakan logika AI (Simplified MDP):

Langkah 1: Menentukan State (Kondisi)

Bayangkan kondisi stok Pembaca merupakan sebuah tangga dari 0 sampai 20 unit.

- Stok 0: Titik "Ledakan" (Biaya pinalti Rp10.000). Nilai $Q = -10.000$.

Langkah 2: Perambatan Nilai (Riak Air Mundur)

Jika $\gamma = 0.9$, maka nilai buruk dari "Stok 0" akan merambat naik ke atas:

- Stok 1: Berisiko besar jadi 0. Nilai $Q = 0.9 \times (-10.000) = -9.000$.
- Stok 2: Berisiko jadi 1. Nilai $Q = 0.9 \times (-9.000) = -8.100$.
- Stok 5: Nilai Q mungkin sekitar -5.900.

Langkah 3: Menemukan Titik "Manis" (Reorder Point)

AI akan membandingkan dua opsi di setiap level stok:

1. Opsi Diam: Membiarkan stok turun (mengikuti riak negatif ke bawah).

2. Opsi Pesan: Membayar biaya pesan (misal Rp2.000) untuk melompat kembali ke Stok 20 (titik aman dengan nilai Q tinggi).

AI akan memutuskan "PESAN" tepat saat:

Biaya Pesan < Risiko Kerugian Masa Depan (Riak Negatif)

Jika di Stok 7 nilai risikonya merupakan Rp2.500 dan biaya pesan hanya Rp2.000, maka AI akan secara otomatis menetapkan 7 sebagai Reorder Point hari itu.

3. Keunggulan Dibandingkan Rumus Biasa

Jika minggu depan ada prediksi hari libur (Permintaan naik), "Riak Negatif" dari stok kosong akan merambat lebih cepat dan lebih tinggi.

- Hasilnya: AI akan menaikkan ROP secara otomatis dari 7 ke 12 tanpa Pembaca perlu mengubah pengaturan secara manual gambaran yang jelas, mari kita gunakan skenario bisnis UNIT USAHA Retail dengan data rata-rata sebagai berikut:
- Permintaan Harian: 10 unit.
- Lead Time (Waktu Kirim): 3 hari.
- Biaya Pesan: Rp2.000.
- Kerugian per Unit (Stockout): Rp1.000 (Potensi profit yang hilang + denda ketidakpuasan).

Tabel 5.1 Perbandingan ROP Tradisional vs. AI "Riak Air"

Fitur Perbandingan	Metode Tradisional (Safety Stock)	Metode AI "Riak Air" (MDP/RL)
Rumus Dasar	$ROP = (Demand \times Lead\ Time) + Safety\ Stock$	$Q(S, A) = R + \gamma \max Q(S', A')$
Logika Utama	Berdasarkan statistik masa lalu yang kaku.	Berdasarkan antisipasi risiko masa depan (Forward-looking).
Reaksi terhadap Promo	Tidak berubah kecuali admin mengubah angka Safety Stock.	Otomatis menaikkan ROP karena "Riak Negatif" stok kosong membesar.
Kalkulasi Reorder Point	Tetap (Misal: Selalu pesan saat stok 35).	Dinamis (Bisa 30 saat sepi, bisa 50 saat tren naik).
Efisiensi Modal	Sering terjadi Overstock (stok menumpuk) untuk cari aman.	Optimal; stok hanya ditambah saat risiko kerugian > biaya pesan.

Simulasi Angka: Saat Terjadi Lonjakan Permintaan (Misal: Gajian)

Mari kita lihat bagaimana kedua metode ini bereaksi ketika permintaan tiba-tiba diprediksi naik dari 10 unit menjadi 20 unit karena tanggal gajian.

Tabel 5.2 Perbandingan Kalkulasi tradisional vs AI "Riak Air"

Skenario	Kalkulasi Tradisional	Kalkulasi AI "Riak Air"
Angka ROP	$(10 \times 3) + 5 = 35$ unit	Berdasarkan perambatan nilai: 65 unit
Keputusan Saat Stok 40	DIAM (Karena $40 > 35$)	PESAN (Karena $40 < 65$)
Hasil Akhir	Stockout. Stok habis di hari ke-2 karena permintaan naik.	Aman. Barang sampai tepat saat permintaan sedang tinggi-tingginya.
Dampak Finansial	Kehilangan potensi untung dari 25 pelanggan.	Mengeluarkan biaya pesan lebih awal, tapi menangkap semua penjualan.

AI "Riak Air" Lebih Unggul

Dalam metode Tradisional, angka Safety Stock seringkali hanyalah "tebakan" atau angka aman yang dipasang tinggi agar tidak dimarahi atasan jika stok habis. Ini mengakibatkan uang Pembaca mati di gudang dalam bentuk barang yang tidak berputar.

Dalam metode AI, sistem menghitung "Opportunity Cost".

1. AI melihat bahwa "Riak" negatif dari kehabisan stok di hari gajian jauh lebih buruk daripada biaya pesan Rp2.000.
2. AI secara matematis membuktikan bahwa membayar Rp2.000 hari ini merupakan investasi terbaik untuk menghindari kerugian Rp25.000 (25 unit lost sales) lusa.

Hasil pada Bisnis Pembaca

Metode AI tidak hanya menjaga stok tetap ada, tapi menjaga agar modal kerja Pembaca seefisien mungkin. Ia tidak akan menyuruh Pembaca beli barang jika risiko kehabisannya rendah, namun ia akan sangat agresif menyuruh Pembaca beli jika ia "melihat" riak kegagalan di masa depan. sebuah algoritma Dynamic Reorder Point (DRP). Berbeda dengan rumus Excel statis, skrip ini akan menghitung risiko "Riak Negatif" berdasarkan fluktuasi permintaan dan lead time.

Algoritma ini menggunakan logika Heuristic MDP sederhana: ia akan menaikkan ambang batas pesanan jika ketidakpastian (variabel risiko) meningkat.

Skrip Python: Dynamic Reorder Point (DRP) Calculator

Python

```
import math
import numpy as np

def calculate_dynamic_rop(avg_daily_demand, std_dev_demand, lead_time,
service_level=1.645):
    """
    Menghitung ROP Dinamis.
    service_level 1.645 setara dengan tingkat pemenuhan 95%.
    """

    # 1. Menghitung Demand selama Lead Time (DLT)
    # Ini merupakan "Riak Dasar"
    dlt = avg_daily_demand * lead_time

    # 2. Menghitung Safety Stock Dinamis
    # Menggunakan Standar Deviasi untuk mengukur "Riak Ketidakpastian"
    # Rumus:  $Z * \sqrt{\text{LeadTime}} * \text{StdDev\_Demand}$ 
    safety_stock = service_level * math.sqrt(lead_time) * std_dev_demand

    # 3. Dynamic ROP
    dynamic_rop = dlt + safety_stock

    return {
        "Base_Demand_Leadtime": round(dlt, 2),
        "Safety_Stock_Adjustment": round(safety_stock, 2),
        "Final_Dynamic_ROP": math.ceil(dynamic_rop)
    }

# --- SIMULASI PENGGUNAAN ---

# Skenario A: Kondisi Normal (Permintaan Stabil)
normal_day = calculate_dynamic_rop(avg_daily_demand=10, std_dev_demand=2,
lead_time=3)

# Skenario B: Kondisi Menjelang Promo (Permintaan Fluktuatif/Tidak Pasti)
promo_season = calculate_dynamic_rop(avg_daily_demand=20, std_dev_demand=8,
lead_time=3)

print(f"KONDISI NORMAL: Pesan saat stok menyentuh {normal_day['Final_Dynamic_ROP']} unit")
print(f"KONDISI PROMO : Pesan saat stok menyentuh {promo_season['Final_Dynamic_ROP']} unit")
```

Algoritma Bekerja Secara Otomatis

Agar algoritma ini benar-benar menjadi "pintar" seperti AI, Pembaca harus menghubungkannya dengan data riil Pembaca melalui tiga langkah:

1. Feeding Data: Skrip mengambil data penjualan 7 hari terakhir dari database Pembaca untuk menghitung avg_daily_demand (Rata-rata) dan std_dev_demand (Seberapa bergejolak permintaannya).
2. Mendeteksi "Riak": Jika dalam 7 hari terakhir penjualan Pembaca melompat-lompat (misal hari ini 5, besok 25), std_dev_demand akan membesar.
3. Auto-Adjustment: Secara otomatis, Final_Dynamic_ROP akan naik. Sistem akan memberikan notifikasi "PESAN SEKARANG" lebih awal daripada biasanya karena sistem "melihat" risiko stok kosong sedang meningkat.

Tabel 5.3 Perbandingan ROP Statis vs Dinamis

Permintaan Sangat Stabil	Tetap 35 (Mungkin kelebihan stok).	Turun ke 32 (Menghemat modal kerja).
Menjelang Lebaran/Promo	Tetap 35 (Risiko stok habis tinggi).	Naik ke 55 (Melindungi ketersediaan barang).
Supplier Telat Kirim	Tidak bereaksi.	Bisa disesuaikan dengan menambah variabel lead_time.

Implementasi Praktis

Kita bisa memasukkan logika ini ke dalam bot Telegram atau Dashboard Excel sederhana. Setiap pagi, skrip ini berjalan, mengecek stok gudang, dan memberikan rekomendasi belanja yang sudah disesuaikan dengan kondisi pasar terbaru.

Tahapan Implementasi Kunci

1. Desain Fungsi Reward (R) (Tahap Persiapan Data)

Desain Reward Function merupakan langkah paling penting dan sulit. Reward harus memandu robot menuju policy optimal secara efisien.

Tabel 5.4 Desain Reward Function

Situasi	Contoh Nilai Reward (R)	Keterangan
Mencapai Tujuan	+100	Hadiah terbesar, mengakhiri episode.
Menghindari Tabrakan	+0.5 (per Langkah aman)	Memberi hadiah pada pergerakan yang hati-hati.
Bertabrakan	-100 (atau lebih besar)	Hukuman besar, mengakhiri episode (memaksa robot belajar menghindari).
Jarak Bertambah	-1	Hukuman kecil (Time Penalty) untuk mendorong efisiensi.

Desain Reward Function (R) merupakan proses "pemrograman moral" bagi AI. Karena AI tidak mengerti konsep "baik" atau "buruk", angka-angka inilah yang menjadi satu-satunya kompas bagi mereka untuk memahami apa yang kita inginkan.

Berikut merupakan penjelasan mendalam mengenai tabel desain reward yang Pembaca berikan, serta logika di balik setiap pemberian nilai tersebut:

1. Mencari Keseimbangan (The Incentive Design)

Desain reward yang buruk bisa menyebabkan "Reward Hacking", di mana robot mencari celah untuk mendapat skor tinggi tanpa menyelesaikan tugas (misalnya: robot

hanya berputar-putar di tempat karena takut menabrak, padahal ia harus sampai ke tujuan).

Tabel 5.5 Analisis Reward Hacking

Situasi	Nilai (R)	Logika Deep Learning
Mencari Tujuan	+100	Ini merupakan "Ultimate Goal". Nilai yang besar ini akan merambat mundur (melalui riak air Bellman) ke langkah-langkah sebelumnya, menciptakan jalur menuju sukses.
Langkah Aman	+0.5	Ini disebut Dense Reward. Tanpa ini, robot mungkin merasa putus asa karena terlalu lama tidak mendapat hadiah. Ini memberikan "semangat" kecil di setiap langkah yang tidak menabrak.
Bertabrakan	-100	Ini merupakan Fatal Penalty. AI akan mengingat state tepat sebelum tabrakan sebagai "Zona Kematian" dan akan menghindarinya dengan segala cara di masa depan.
Efisiensi Waktu	-1	Ini merupakan Time Penalty. Tanpa hukuman kecil ini, robot mungkin akan mengambil jalan memutar yang sangat jauh asalkan aman. Hukuman ini memaksa robot mencari lintasan terpendek.

2. Tahap Persiapan Data (Data Preparation)

Dalam konteks navigasi robot atau manajemen stok, tabel di atas harus diterjemahkan ke dalam kode program sebelum pelatihan dimulai.

1. Normalisasi Reward: Nilai tidak boleh terlalu besar (misal: satu triliun) atau terlalu kecil (misal: 0.000001) agar Neural Network tidak mengalami masalah gradient (error kalkulasi). Skala -100 hingga +100 biasanya sangat ideal.
2. Sparse vs Dense Reward:
 - Sparse (Jarang): Hanya memberi +100 di akhir. Robot akan sangat lama belajar karena jarang sekali "beruntung" sampai ke tujuan secara acak.
 - Dense (Padat): Memberi hadiah kecil (+0.5 atau -1) di setiap langkah. Ini membuat robot belajar jauh lebih cepat karena selalu ada umpan balik.
3. Contoh Implementasi pada Manajemen Stok

Jika kita bawa tabel reward navigasi Pembaca ke kasus Gudang Unit Usaha, desainnya akan menjadi seperti ini:

- Terjadi Penjualan: +10 (Hadiah karena performa bagus).
- Stok Kosong (Stockout): -100 (Hukuman karena kehilangan pelanggan).
- Biaya Simpan (Overstock): -1 per unit (Hukuman kecil agar barang tidak menumpuk terlalu lama).
- Melakukan Pesanan (Order): -2 (Biaya operasional/administrasi).

Hasil Langkah yang Paling Sulit

Jika kita memberikan reward yang salah, AI akan bertindak aneh. Contoh:

- Jika Time Penalty (-1) terlalu besar, robot mungkin akan memilih untuk langsung menabrakkan diri ke tembok agar episode cepat berakhir dan hukuman tidak bertambah banyak (AI "bunuh diri" demi efisiensi).
- Jika Langkah Aman (+0.5) terlalu besar, robot mungkin akan terus berputar-putar di area aman selamanya dan tidak pernah mau pergi ke tujuan

Tantangan menarik yang dikenal sebagai FEFO (First Expired, First Out) dalam manajemen inventaris. Dalam dunia AI, kita harus membuat robot stok Pembaca merasa "cemas" jika memegang barang yang mendekati tanggal kedaluwarsa.

Untuk memprioritaskan barang yang cepat basi, kita perlu menambahkan variabel "Time-to-Expiry" ke dalam fungsi reward.

Desain Fungsi Reward untuk Barang Mudah Basi

Kita akan memodifikasi logika reward standar dengan menambahkan Penalti Pembusukan (Perishing Penalty).

Tabel 5.6 Desain Fungsi Reward Pinalti Expired

Situasi	Reward (R)	Logika AI
Berhasil Menjual	+100	Hadiah utama. Namun, kita bisa modifikasi: +100 jika menjual barang stok lama, +50 jika menjual stok baru.
Barang Kedaluwarsa	-500	Hukuman sangat berat. Jauh lebih besar dari denda stok kosong, karena barang basi = kerugian total (modal + potensi untung).
Biaya Simpan Harian	-1 (x Faktor Usia)	Semakin tua barang (mendekati expired), penalti harian semakin besar. Ini memaksa AI segera menjualnya.
Melakukan Pesanan Baru	-10	Mencegah AI memesan terlalu banyak saat stok lama masih ada (menghindari penumpukan).

Rumus Logika Reward Spesifik (Python Logic)

Jika kita ingin AI memprioritaskan barang A (cepat basi) daripada barang B (awet), kita gunakan pengali risiko:

Python

```
def calculate_perishable_reward(action, days_to_expiry, is_sold):
    reward = 0

    if is_sold:
        # Menjual barang yang mau basi dapat bonus lebih besar!
        if days_to_expiry <= 3:
            reward = 150 # Apresiasi tinggi karena berhasil menyelamatkan barang
        else:
            reward = 100

    # Jika barang tidak terjual (masih di gudang)
    else:
        if days_to_expiry <= 0:
            reward = -500 # Barang busuk/dibuang
        elif days_to_expiry <= 3:
            reward = -50 # Penalti "Kecemasan": Barang harus segera keluar!
        else:
            reward = -1 # Biaya simpan normal

    return reward
```

Cara AI Berperilaku dengan Logika

1. **Prioritas Penjualan:** Saat ada pelanggan, AI akan "mendorong" sistem untuk mengeluarkan stok yang paling lama karena nilai reward-nya lebih tinggi (+150 vs +100).
2. **Manajemen Pesanan:** AI akan menjadi sangat hati-hati saat memesan barang yang cepat basi. Jika ia melihat data penjualan melambat, ia akan berhenti memesan jauh sebelum barang di gudang habis untuk memastikan tidak ada sisa yang membusuk.
3. **Diskon Otomatis:** Dalam sistem yang lebih canggih, AI bisa merekomendasikan "Diskon 20%" jika ia melihat Riak Negatif (kerugian -500) sudah sangat dekat, karena diskon lebih baik daripada rugi total.

Analogi untuk Tim Pembaca

Bayangkan AI seperti pelayan restoran. Tanpa logika ini, dia akan mengambil bahan makanan apa saja yang paling depan. Dengan logika ini, dia akan selalu memeriksa label tanggal, dan jika ada susu yang akan basi besok, dia akan sangat aktif menawarkannya kepada pelanggan agar tidak terbuang sia-sia.

Simulasi perbandingan. Di sini, kita akan melihat perbedaan perilaku AI saat mengelola Barang A (Beras - Awet) dibandingkan dengan Barang B (Susu Segar - Cepat Basi).

Logika AI dalam simulasi ini didorong oleh rasa takut akan penalti pembusukan ($R = -500$) dibandingkan dengan biaya penyimpanan biasa ($R = -1$).

Tabel 5.7 Simulasi Strategi Stok Awet vs Cepat Basi

Komponen	Barang Awet (Beras)	Barang Cepat Basi (Susu Segar)
Masa Kedaluwarsa	12 Bulan	5 Hari
Kapasitas Pesanan	Sekali banyak (Bulk)	Sedikit tapi sering (JIT)
Batas Aman (Buffer)	Tinggi (AI tidak takut stok sisa)	Sangat Rendah (AI takut sisa)
Reward Fokus	Meminimalkan biaya kirim.	Menghindari pembusukan.

Visualisasi Perilaku AI

1. Pola Barang Awet (Beras)

AI akan cenderung memesan dalam jumlah besar sekaligus. Karena tidak ada risiko basi dalam waktu dekat, AI memilih untuk meminimalkan biaya pesan ($R = -10$ per order). Grafik stok akan terlihat seperti "Gigi Gergaji" yang besar dan lebar.

2. Pola Barang Cepat Basi (Susu Segar)

AI akan sangat "pelit" dalam memesan. Ia lebih memilih melakukan pemesanan kecil setiap hari meskipun biaya pesan menumpuk, daripada mengambil risiko stok sisa di hari ke-5 yang akan memicu penalti masif -500. Grafik stok akan terlihat seperti gerigi kecil yang sangat rapat mendekati garis nol.

Skrip Python: Simulasi Pengambilan Keputusan AI

Skrip sederhana ini menunjukkan bagaimana AI menghitung volume pesanan berdasarkan sisa umur simpan.

Python

```
def ai_order_decision(item_type, current_stok, days_to_expiry, avg_demand):
    if item_type == "Awet":
        # Logika: Pesan banyak untuk stok 2 minggu
        if current_stok < (avg_demand * 3): # ROP Statis
            return avg_demand * 14

    elif item_type == "Cepat Basi":
        # Logika: Pesan hanya untuk kebutuhan 2 hari ke depan
        # AI mempertimbangkan risiko busuk jika sisa umur simpan pendek
        if days_to_expiry <= 2:
            return 0 # Jangan pesan lagi, habiskan yang ada!
        if current_stok < avg_demand:
            return avg_demand * 2 # Pesan sangat sedikit

    return 0

# Hasil Simulasi
print(f"Pesanan Beras: {ai_order_decision('Awet', 5, 300, 10)} unit")
print(f"Pesanan Susu : {ai_order_decision('Cepat Basi', 5, 2, 10)} unit")
```

1. Analisis Hasil dengan AI Berbeda

Dalam kasus Susu, AI melihat bahwa jika ia memesan 20 unit sementara permintaan hanya 10, maka besok sisa umur simpan menjadi 1 hari. Riak negatif dari potensi rugi -500 mulai merambat ke keputusan hari ini.

Sedangkan pada Beras, riak negatif -500 masih berada 300 hari di depan. Karena faktor diskon $\gamma = 0.9$), nilai -500 yang terjadi 300 hari lagi saat ini nilainya mendekati 0. Maka, AI menganggap risiko basi Beras saat ini tidak ada.

2. Pelatihan dengan Deep Q-Networks (DQN)

Untuk lingkungan robot yang memiliki state space kontinu (data sensor Lidar merupakan nilai numerik kontinu), algoritma DQN sering digunakan.

1. Input: Vektor S (data sensor, posisi) dimasukkan ke dalam Jaringan Saraf Tiruan (ANN).
2. Output: Jaringan menghasilkan perkiraan nilai Q untuk setiap aksi yang mungkin ($Q(S, A_1), Q(S, A_2), \dots$)
3. Update Q-Value: Robot mengambil aksi yang disarankan, mengamati hadiah R dan state baru S', kemudian Jaringan Saraf diperbarui menggunakan teknik backpropagation untuk mengurangi kesalahan antara Q yang diprediksi dan Q target (berdasarkan Persamaan Bellman).
4. Eksplorasi: Robot menggunakan strategi e-greedy, di mana ia terkadang memilih aksi acak (Eksplorasi) daripada aksi terbaik saat ini (Eksplotasi) untuk menemukan rute yang lebih baik.

Pelatihan dengan Deep Q-Networks (DQN) merupakan evolusi di mana kita mengganti "Buku Tabel" dengan "Otak Digital". Ini sangat krusial jika data sensor Pembaca bukan lagi angka bulat sederhana (seperti kotak 1, 2, 3), melainkan nilai desimal yang sangat detail dari Lidar atau fluktuasi stok yang kompleks.

Penjelasan mendalam mengenai alur kerja DQN berdasarkan poin-poin yang Pembaca sebutkan:

1. Input: Vektor S (Sensor & Posisi)

DQN tidak melihat dunia sebagai koordinat kaku. Ia menerima Vektor State (\$\$\$).

- Dalam navigasi: Ini bisa berupa 360 titik data dari Lidar (jarak ke segala arah).
- Dalam gudang: Ini bisa berupa angka stok, tren penjualan, dan sisa hari hingga kedaluwarsa.

Jaringan Saraf (ANN) bertugas mengekstraksi fitur dari data mentah ini untuk memahami konteks situasi.

2. Output: Estimasi Nilai Q (Prediksi)

Berbeda dengan jaringan saraf biasa yang mungkin mengklasifikasikan "Ini Kucing" atau "Ini Anjing", output dari DQN merupakan Angka Kualitas untuk setiap aksi.

- $Q(S, \text{Maju}) = 0.85$
- $Q(S, \text{Belok}) = 0.12$

Robot kemudian melihat hasil "skor" ini dan biasanya memilih skor yang paling tinggi.

3. Update Q-Value: Proses "Belajar dari Kesalahan"

Ini adalah bagian yang paling canggih. DQN menggunakan Backpropagation untuk memperbaiki dirinya.

1. Prediksi: Jaringan menebak nilai Q.
2. Pengalaman: Robot benar-benar melakukan aksi tersebut, lalu melihat kenyataan (R dan S).
3. Koreksi: Selisih antara "Tebakan" dan "Kenyataan (Rumus Bellman)" disebut Loss. Jaringan akan mengubah bobot internalnya agar di masa depan tebakannya semakin akurat.

4. Eksplorasi: Strategi epsilon-greedy (Epsilon-Greedy)

Ini merupakan cara AI menyeimbangkan antara menjadi "Pintar" dan "Berani".

- Eksploitasi: Menggunakan pengetahuan yang sudah ada untuk mendapat reward aman.
- Eksplorasi: Melakukan aksi acak untuk melihat apakah ada jalan yang lebih baik (misalnya, mencoba jalur baru yang mungkin lebih cepat).

Seiring berjalannya waktu, nilai ϵ (peluang acak) akan diturunkan (decay). AI yang awalnya "liar" mencoba segala hal akan berubah menjadi AI yang sangat disiplin dan efisien.

DQN dibutuhkan untuk menyelesaikan masalah Kontinu

Jika Pembaca menggunakan tabel (Q-Learning Tradisional), Pembaca butuh baris terpisah untuk jarak 1.0 meter, 1.01 meter, 1.02 meter, dan seterusnya. Ini tidak efektif.

DQN bisa melakukan interpolasi. Jika ia tahu bahwa jarak 1.0 meter berbahaya, ia secara otomatis menduga bahwa jarak 1.05 meter juga kemungkinan besar berbahaya.

Penerapan pada Prioritas Barang Basi (FEFO)

Dalam kasus barang kedaluwarsa yang kita bahas sebelumnya, DQN akan menerima input kontinu berupa "Persentase Sisa Umur Simpan".

- AI tidak hanya tahu "Basi" atau "Tidak Basi".
- Ia akan mulai merasakan "tekanan" yang meningkat secara gradual saat sisa umur barang bergerak dari 10% ke 5%. Output nilai Q untuk aksi "Diskon" atau "Promosi Agresif" akan perlahan-lahan naik melampaui nilai Q untuk "Jual Normal".

Stok barang cepat basi (perishable goods), kita membutuhkan arsitektur Deep Q-Network (DQN) yang cukup dalam untuk menangkap hubungan antara sisa umur barang dan tren permintaan, namun tetap ringan agar proses training-nya cepat.

Berikut merupakan desain arsitektur yang ideal untuk masalah Pembaca:

1. Desain Arsitektur Jaringan (ANN)

Untuk sistem stok dengan fitur FEFO (First Expired, First Out), saya merekomendasikan struktur 4-Layer MLP (Multi-Layer Perceptron):

- Input Layer (5-10 Neuron): Tergantung variabel yang Pembaca masukkan (misal: Level Stok, Rata-rata Penjualan, Sisa Hari Kedaluwarsa, Harga, dan Hari Promo).
- Hidden Layer 1 (64 Neuron): Untuk menangkap pola dasar permintaan.
- Hidden Layer 2 (32 Neuron): Untuk memproses interaksi kompleks (misal: hubungan antara stok menumpuk dan sisa umur simpan yang menipis).
- Output Layer (N Aksi): Jumlah neuron sesuai pilihan aksi (misal: 0: Diam, 1: Pesan Stok, 2: Beri Diskon).

2. Implementasi Python menggunakan PyTorch

Berikut merupakan cuplikan kode untuk membangun arsitektur tersebut:

Python

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class InventoryDQN(nn.Module):
    def __init__(self, state_size, action_size):
        super(InventoryDQN, self).__init__()

        # Layer 1: Input ke Hidden 1
        self.fc1 = nn.Linear(state_size, 64)

        # Layer 2: Hidden 1 ke Hidden 2
        self.fc2 = nn.Linear(64, 32)

        # Layer 3: Output (Nilai Q untuk setiap aksi)
        self.fc3 = nn.Linear(32, action_size)

    def forward(self, x):
        # Menggunakan aktivasi ReLU agar jaringan bisa mempelajari pola non-linear
        x = F.relu(self.fc1(x))
```

```

        x = F.relu(self.fc2(x))

        # Output layer tidak menggunakan aktivasi (Linear) karena outputnya
        merupakan skor Q
        return self.fc3(x)

# Contoh penggunaan:
# state_size = 5 (Stok, Penjualan, Expired_Days, Harga, Promo)
# action_size = 3 (Diam, Pesan, Diskon)
model = InventoryDQN(state_size=5, action_size=3)
print(model)

```

3. Arsitektur Python menggunakan PyTorch "Ideal"

- 64 & 32 Neuron: Ukuran ini merupakan sweet spot. Jika terlalu besar (misal 512), AI akan cenderung menghafal data (overfitting). Jika terlalu kecil, AI tidak akan paham kapan harus mulai panik saat barang mendekati masa kedaluwarsa.
- Fungsi Aktivasi ReLU: Sangat efektif untuk masalah stok karena ReLU mengabaikan nilai negatif secara efisien, mempercepat AI dalam membedakan kondisi "Aman" dan "Bahaya".
- Output Linear: Memberikan nilai estimasi profit/kerugian yang fleksibel, sehingga AI bisa membandingkan secara presisi apakah "Diskon 20%" lebih menguntungkan daripada "Menanggung Risiko Basi".

4. Visualisasi Alur Data

1. Input: AI menerima data "Susu sisa 10 botol, expired 2 hari lagi".
2. Proses: Jaringan memproses data ini melewati 64 dan 32 neuron yang sudah terlatih.
3. Output: AI mengeluarkan nilai Q:
 - $Q(\text{Diam}) = -200$ (Risiko rugi besar karena bakal basi).
 - $Q(\text{Diskon}) = +50$ (Untung sedikit, tapi stok habis terselamatkan).
 - Keputusan: AI memilih Diskon.

Satukan arsitektur Neural Network tadi ke dalam siklus pelatihan Deep Q-Learning. Skrip ini akan mensimulasikan proses belajar di mana AI mencoba berbagai keputusan stok, mendapatkan reward, dan memperbarui otaknya.

Dengan menggunakan pustaka `torch` (PyTorch) untuk otak AI dan `pandas` untuk membaca data Excel Pembaca.

Skrip Training DQN: Manajemen Stok Barang Cepat Basi

Python

```

import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
import pandas as pd
import random

```

```

from collections import deque

# 1. DEFINISI ARSITEKTUR (Seperti yang kita bahas sebelumnya)
class InventoryDQN(nn.Module):
    def __init__(self, state_size, action_size):
        super(InventoryDQN, self).__init__()
        self.fc1 = nn.Linear(state_size, 64)
        self.fc2 = nn.Linear(64, 32)
        self.fc3 = nn.Linear(32, action_size)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        return self.fc3(x)

# 2. KONFIGURASI AGENT
class DQNAgent:
    def __init__(self, state_size, action_size):
        self.state_size = state_size
        self.action_size = action_size
        self.memory = deque(maxlen=2000) # Experience Replay
        self.gamma = 0.95 # Discount factor
        self.epsilon = 1.0 # Exploration rate
        self.epsilon_min = 0.01
        self.epsilon_decay = 0.995
        self.model = InventoryDQN(state_size, action_size)
        self.optimizer = optim.Adam(self.model.parameters(), lr=0.001)
        self.criterion = nn.MSELoss()

    def act(self, state):
        # Strategi Epsilon-Greedy
        if np.random.rand() <= self.epsilon:
            return random.randrange(self.action_size)
        state = torch.FloatTensor(state)
        act_values = self.model(state)
        return torch.argmax(act_values).item()

    def train(self, batch_size):
        if len(self.memory) < batch_size: return

        minibatch = random.sample(self.memory, batch_size)
        for state, action, reward, next_state, done in minibatch:
            state = torch.FloatTensor(state)
            next_state = torch.FloatTensor(next_state)

            # Target Q-Value menggunakan Persamaan Bellman
            target = reward
            if not done:
                target = reward + self.gamma *
torch.max(self.model(next_state)).item()

            target_f = self.model(state)
            target_f[action] = target

            # Backpropagation
            self.optimizer.zero_grad()
            output = self.model(state)
            loss = self.criterion(output, target_f.detach())
            loss.backward()
            self.optimizer.step()

            if self.epsilon > self.epsilon_min:
                self.epsilon *= self.epsilon_decay

# 3. SIMULASI TRAINING DENGAN DATA EXCEL
# Asumsi kolom Excel: ['stok', 'permintaan_rata2', 'hari_expired', 'harga',
'is_promo']
# df = pd.read_excel("data_stok.xlsx")

```

```

agent = DQNAgent(state_size=5, action_size=3) # 3 Aksi: Diam, Pesan, Diskon
batch_size = 32
episodes = 100

print("Memulai pelatihan...")
for e in range(episodes):
    # Reset state (Contoh data awal)
    state = np.array([20, 10, 5, 50000, 0], dtype=float)

    for time in range(30): # Simulasi 30 hari per episode
        action = agent.act(state)

# LOGIKA REWARD (Berdasarkan desain kita sebelumnya)
    # Skenario sederhana: jika aksi=diskon saat expired dekat, reward tinggi
    reward = 10 if action == 2 and state[2] < 3 else -1
    if state[2] <= 0: reward = -100 # Hukuman mati jika basi

    next_state = state.copy()
    next_state[2] -= 1 # Hari expired berkurang
    done = True if next_state[2] <= 0 else False

    agent.memory.append((state, action, reward, next_state, done))
    state = next_state

    if done: break

agent.train(batch_size)
if e % 10 == 0:
    print(f"Episode: {e}/{episodes}, Epsilon: {agent.epsilon:.2f}")

print("Pelatihan selesai. Model siap digunakan!")

```

Menghubungkan dengan Excel

Untuk membuat skrip ini bekerja dengan data asli, Pembaca perlu melakukan beberapa hal:

1. Iterasi Baris Excel: Gunakan `for index, row in df.iterrows():` untuk mengambil data baris demi baris sebagai `state`.
2. Preprocessing: Pastikan semua data di Excel sudah berupa angka. Jika ada kolom teks (seperti "Promo: Ya"), ubah menjadi 1 dan 0.
3. Simulasi "What If": Karena RL membutuhkan interaksi, Pembaca perlu membuat fungsi sederhana yang mensimulasikan apa yang terjadi pada stok jika AI memilih "Pesan" (stok bertambah) atau "Diskon" (permintaan meningkat).

Kelebihan Menggunakan **Experience Replay (deque)**

Perhatikan bagian `self.memory`. Ini merupakan "Buku Kenangan" AI. AI tidak belajar langsung dari satu kejadian, tapi ia mengumpulkan banyak kejadian, lalu mengambil sampel acak untuk dipelajari. Ini sangat penting agar AI tidak "panik" dan lupa cara menangani stok normal saat ia sedang belajar menangani stok yang hampir basi.

kita perlu membuat Environment (Lingkungan) yang bisa memberikan reaksi balik. Jika AI memberikan "Diskon", maka secara realistis permintaan barang harus meningkat, namun keuntungan per unit akan menurun.

Berikut merupakan kelas Python untuk Environment yang mensimulasikan dinamika stok, efek diskon, dan risiko barang basi secara realistis.

Fungsi Simulasi Environment (Stok & Diskon)

Python

```
import numpy as np

class GudangEnvironment:
    def __init__(self, data_awal):
        # data_awal: [stok, permintaan_base, hari_expired, harga_satuan]
        self.state_awal = np.array(data_awal, dtype=float)
        self.reset()

    def reset(self):
        self.state = self.state_awal.copy()
        self.hari_berjalan = 0
        return self.state

    def step(self, action):
        """
        Aksi:
        0 = Diam (Harga Normal)
        1 = Pesan Stok (Tambah 20 unit, Biaya -10)
        2 = Diskon 30% (Permintaan Naik 2x, Margin Turun)
        """
        stok, permintaan_base, hari_expired, harga = self.state
        reward = 0
        done = False

        # 1. Logika Aksi
        permintaan_aktual = permintaan_base
        if action == 1: # Pesan Stok
            stok += 20
            reward -= 20 # Biaya operasional pesan barang
        elif action == 2: # Diskon
            permintaan_aktual = permintaan_base * 2.5 # Efek Diskon: Penjualan naik
            2.5x
            harga = harga * 0.7 # Margin turun 30%
            reward += 5 # Bonus kecil karena strategi promosi

        # 2. Proses Penjualan (Update Stok)
        # Menambahkan faktor acak pada permintaan agar lebih realistis
        penjualan_riil = min(stok, np.random.poisson(permintaan_aktual))
        stok -= penjualan_riil

        # 3. Hitung Reward Finansial
        keuntungan = penjualan_riil * (harga * 0.2) # Asumsi margin untung 20%
        reward += keuntungan

        # 4. Penalti Risiko & Update Waktu
        hari_expired -= 1
        self.hari_berjalan += 1

        # Hukuman jika barang basi
        if hari_expired <= 0 and stok > 0:
```

```

        reward -= (stok * 50) # Penalti besar per unit yang dibuang
        stok = 0
        done = True

    # Hukuman jika stok kosong (Stockout)
    if stok <= 0 and penjualan_riil < permintaan_aktual:
        reward -= 30 # Kehilangan potensi pelanggan

    # Update State
    self.state = np.array([stok, permintaan_base, hari_expired, harga],
dtype=float)

    # Selesai jika sudah 30 hari atau barang habis/basi
    if self.hari_berjalan >= 30:
        done = True

    return self.state, reward, done

```

```
# --- CARA MENGGUNAKANNYA DALAM TRAINING ---
env = GudangEnvironment(data_awal=[30, 5, 10, 10000]) # Stok 30, Expired 10 hari
state = env.reset()

for t in range(5):
    action = 2 # AI mencoba aksi 'Diskon'
    next_state, reward, done = env.step(action)
    print(f"Hari {t+1}: Stok={next_state[0]}, Reward={reward:.2}, Sisa Hari
Expired={next_state[2]}")
    if done: break
```

Environment yang disajikan ini lebih realistis

1. Distribusi Poisson (`np.random.poisson`): Dalam dunia nyata, pelanggan tidak datang tepat 10 orang setiap hari. Terkadang 8, terkadang 12. Fungsi ini mensimulasikan ketidakpastian tersebut agar AI tidak hanya "menghafal" angka tetap.
2. Efek Elastisitas Harga: Dengan kode `permintaan_aktual = permintaan_base * 2.5`, AI akan belajar bahwa Diskon merupakan alat yang sangat kuat untuk menghabiskan stok yang hampir basi dengan cepat, meskipun keuntungan per botolnya lebih kecil.
3. Faktor Biaya Tersembunyi: Kita memasukkan biaya pesan (action 1) dan biaya barang terbuang. Ini memaksa AI untuk berpikir dua kali: "Apakah saya pesan sekarang dan risiko basi, atau tunggu sampai stok benar-benar kritis?"
4. Feedback Loop: Jika AI memberikan diskon terlalu awal, ia akan kehabisan stok terlalu cepat (mendapat penalti Stockout). Jika terlalu lambat, ia mendapat penalti barang basi. AI akan mencari Titik Tengah yang optimal secara otomatis.

Latihan dengan menggunakan AI akan merubah perilaku.

Setelah dilatih dengan environment ini selama ribuan kali (episode), AI Pembaca akan memiliki perilaku seperti manajer toko yang berpengalaman:

- Ia akan mulai memberikan diskon bertahap saat sisa umur simpan di bawah 3 hari.
- Ia akan berhenti memesan barang baru jika ia melihat stok lama masih menumpuk dan mendekati tanggal kadaluwarsa.
- Ia akan belajar bahwa menjual rugi (diskon besar) lebih baik daripada rugi total (barang dibuang).

Deployment

Setelah policy dilatih dalam simulasi dan divalidasi kinerjanya, jaringan saraf yang menyimpan Q-Function (atau Policy Network) dipasang ke robot fisik.

- **Proses Fisik:** Data sensor real-time robot (Lidar) dikirimkan ke model ML yang telah dilatih. Model menghasilkan aksi optimal yang harus dilakukan robot, yang kemudian dikirim sebagai perintah ke motor robot.
- **Keuntungan:** Robot dapat bernavigasi dan membuat keputusan yang optimal bahkan dalam kondisi lingkungan yang tidak terduga atau baru, karena model telah belajar hubungan yang kompleks antara state dan action dari ribuan jam pengalaman simulasi.

Deployment merupakan fase di mana "otak" digital yang telah kita latih di laboratorium simulasi dipindahkan ke dunia nyata—baik itu ke dalam fisik robot gudang maupun ke dalam sistem manajemen inventaris (ERP) Pembaca.

Rincian proses cara AI yang kita bangun sebelumnya mulai bekerja secara mandiri:

1. Membekukan Otak (Model Freezing)

Setelah fase training selesai dan AI sudah mahir menangani stok barang basi, kita melakukan Export Model. Bobot-bobot (weights) di dalam Neural Network yang telah kita latih menggunakan InventoryDQN disimpan dalam file (misalnya format .pth atau .onnx).

Di fase ini, AI berhenti belajar dan mulai bekerja. Ia tidak lagi melakukan eksplorasi acak, melainkan selalu memilih aksi dengan nilai Q tertinggi (Eksplorasi 100%).

2. Alur Proses Real-Time (Inference)

Saat deployment, prosesnya berubah dari siklus belajar menjadi siklus Inference (Pengambilan Keputusan):

1. **Input Data (S):** Sistem mengambil data detik ini.
 - Robot: Lidar mendeteksi ada rintangan di jarak 1.2 meter.
 - Gudang: Sistem membaca stok Susu sisa 5 botol, expired dalam 2 hari.
2. **Forward Pass:** Data dikirim ke model ML. Jaringan saraf melakukan perhitungan kilat melalui layer-layer yang sudah kita buat (64 dan 32 neuron).
3. **Aksi Optimal (A):** Model mengeluarkan keputusan.
 - Robot: "Belok kanan 30°."
 - Gudang: "Aktifkan label diskon 30% di aplikasi/toko."

3. Keuntungan Utama di Dunia Nyata

Mengapa menggunakan hasil pelatihan DQN ini jauh lebih baik daripada sekadar rumus logika biasa?

- Resiliensi Terhadap Ketidakpastian: Di simulasi, kita melatih AI dengan distribusi Poisson (permintaan acak). Saat di dunia nyata ada lonjakan pembeli yang tidak terduga, AI tidak akan "kaget" karena ia sudah pernah mengalami ribuan skenario serupa di simulasi.
- Kecepatan Reaksi: AI hanya butuh beberapa milidetik untuk memproses vektor sensor menjadi aksi. Robot bisa menghindari tabrakan secara instan tanpa harus menghitung ulang seluruh peta.
- Efisiensi Jangka Panjang: AI secara konsisten menerapkan kebijakan "Riak Air" (Bellman). Ia tidak akan emosional; ia hanya akan memesan barang atau memberi diskon jika secara matematis itu meminimalkan kerugian total di masa depan.

4. Monitoring dan Retraining

Dunia nyata bisa berubah (misalnya: muncul kompetitor baru atau perubahan tren pasar). Oleh karena itu, dalam deployment tetap ada fase Monitoring:

- Jika kinerja AI mulai turun (misalnya jumlah barang basi meningkat lagi), data dunia nyata tersebut dikumpulkan untuk menjadi bahan pelatihan ulang (retraining) di lingkungan simulasi yang baru.

Hasil Perjalanan AI dengan telah menempuh jalur dari:

1. Analogi Dasar: Memahami Q-Table.
2. Modernisasi: Menggantinya dengan Deep Neural Network (DQN).
3. Desain Moral: Mengatur Reward Function agar AI peduli pada barang basi (FEFO).
4. Pelatihan: Membuat simulasi lingkungan (Environment) yang realistis.
5. Eksekusi: Melakukan Deployment untuk otomasi keputusan bisnis.

Draf Ringkasan Eksekutif yang dirancang secara profesional untuk meyakinkan pemangku kepentingan (stakeholders) mengenai nilai strategis dan finansial dari penerapan sistem Deep Q-Network (DQN) pada manajemen stok dan operasional gudang.

5.2 Implementasi Inteligensi Artifisial (DQN) pada Optimasi Inventaris

1. Pendahuluan & Masalah Utama

Sistem manajemen gudang tradisional saat ini sangat bergantung pada parameter statis yang gagal merespons dinamika pasar yang fluktuatif. Masalah utama yang dihadapi meliputi:

- Kerugian Finansial: Barang kedaluwarsa (perishable) yang tidak terjual tepat waktu.
- Stockout: Kehilangan potensi pendapatan akibat keterlambatan pemesanan kembali (reordering).
- Ketidakefisienan Modal: Dana yang tertahan pada stok berlebih (overstock) yang tidak produktif.

2. Solusi Teknologi: Deep Q-Network (DQN)

Kami mengusulkan transisi dari logika manual ke Deep Q-Network, sebuah bentuk kecerdasan buatan (Reinforcement Learning) yang mampu membuat keputusan otonom. Berbeda dengan rumus biasa, AI ini belajar melalui ribuan jam simulasi operasional untuk memahami:

- Prediksi Dinamis: Menyesuaikan titik pesan ulang (Reorder Point) secara real-time berdasarkan tren penjualan.
- Strategi FEFO Otomatis: Memprioritaskan pengeluaran barang yang mendekati masa kedaluwarsa melalui kebijakan diskon dan alokasi stok yang cerdas.
- Generalisasi: Mampu menangani ribuan jenis barang sekaligus tanpa memerlukan intervensi manual satu per satu.

3. Nilai Keuntungan & ROI (Return on Investment)

Implementasi sistem ini diproyeksikan memberikan dampak finansial yang terukur:

- Reduksi Biaya Operasional: Menurunkan tingkat pembuangan barang basi hingga 30-50% melalui strategi diskon berbasis risiko.
- Peningkatan Revenue: Meminimalisir kejadian stok kosong (stockout) sehingga potensi penjualan tertangkap maksimal.
- Efisiensi Modal Kerja: Optimalisasi level stok yang mengurangi biaya penyimpanan gudang secara keseluruhan.
- Skalabilitas: Satu model AI dapat mengelola ribuan SKU rintangan kompleks yang mustahil dikelola secara manual oleh admin gudang.

4. Roadmap Implementasi

1. Tahap Persiapan: Integrasi data historis penjualan dan profil kedaluwarsa barang.
2. Tahap Simulasi (Training): Melatih agen AI dalam lingkungan virtual (Environment) untuk menguji berbagai skenario pasar.
3. Tahap Deployment: Pemasangan model ke dalam sistem inventaris eksisting untuk memberikan rekomendasi aksi harian.

Investasi pada sistem Deep Q-Network bukan sekadar peningkatan IT, melainkan transformasi strategis untuk mengubah gudang dari pusat biaya (cost center) menjadi pusat efisiensi yang responsif dan menguntungkan pemangku kepentingan (stakeholders) memerlukan keseimbangan antara optimisme teknologi dan realisme bisnis. Atasan biasanya akan fokus pada mitigasi risiko dan keamanan.

Poin-poin jawaban teknis yang dirancang untuk menjawab keraguan tersebut dengan lugas dan meyakinkan:

1. Risiko Kegagalan Keputusan (AI "Halusinasi" atau Salah Langkah)

Pertanyaan: "Bagaimana jika AI tiba-tiba memberikan diskon 90% atau memesan barang dalam jumlah yang tidak masuk akal?"

- Jawaban: Mekanisme Guardrails (Pagar Pembatas). "AI tidak bekerja tanpa pengawasan. Kami menerapkan hard constraints di atas model AI. Contohnya, kami menetapkan batas diskon maksimal (misal: tidak boleh lebih dari 30%) dan batas pesanan maksimal di level kode pemrograman. AI hanya memberikan optimasi di dalam koridor aman yang sudah kita tentukan."
- Jawaban: Human-in-the-Loop. "Pada fase awal deployment, AI bersifat sebagai Decision Support System. Artinya, AI memberikan rekomendasi, namun eksekusi final tetap membutuhkan persetujuan manajer gudang hingga sistem terbukti stabil 100%."

2. Keamanan Data dan Privasi

Pertanyaan: "Apakah data penjualan dan rahasia dagang kita aman jika dimasukkan ke dalam model ini?"

- Jawaban: Local/Private Training. "Model DQN ini dilatih di infrastruktur privat kita sendiri (on-premise atau private cloud). Data tidak dikirim ke pihak ketiga atau penyedia layanan AI publik seperti ChatGPT. Semua data historis dan tren penjualan tetap berada dalam kendali penuh perusahaan."
- Jawaban: Anonimitas Data. "Data yang masuk ke model hanyalah angka numerik (vektor). AI tidak perlu tahu nama pelanggan atau detail identitas lainnya; ia hanya memproses pola permintaan dan durasi waktu."

3. Transparansi Keputusan (Black Box)

Pertanyaan: "Bagaimana kita tahu mengapa AI mengambil keputusan tersebut? Apakah kita bisa melacak logikanya?"

- Jawaban: Interpretability & Analisis Nilai Q. "Setiap keputusan didasarkan pada skor nilai Q. Kita bisa menarik laporan yang menunjukkan bahwa AI memilih 'Diskon' karena nilai estimasi kerugian jika barang basi jauh lebih tinggi daripada margin yang hilang akibat diskon. Semua keputusan memiliki dasar matematis yang bisa diaudit."

4. Adaptasi Terhadap Perubahan Pasar yang Ekstrem

Pertanyaan: "Jika terjadi kejadian luar biasa (seperti pandemi atau bencana), apakah AI akan kacau karena data masa lalunya tidak relevan?"

- Jawaban: Strategi Eksplorasi (E-Greedy). "AI memiliki fitur eksplorasi. Jika pola permintaan berubah secara drastis, AI akan menyadari bahwa prediksi lamanya tidak lagi menghasilkan reward yang baik. Secara otomatis, AI akan 'belajar ulang' dan menyesuaikan kebijakannya dengan data baru yang masuk setiap hari."

5. Biaya vs Keuntungan (ROI)

Pertanyaan: "Berapa biaya komputasi yang dibutuhkan? Apakah sebanding dengan penghematannya?"

- Jawaban: Efisiensi Arsitektur. "Arsitektur yang kita gunakan (DQN 4-layer) sangat ringan. Proses pengambilan keputusan (inference) hanya membutuhkan server standar dan tidak memerlukan superkomputer mahal. Penghematan dari pengurangan barang basi sebesar 10% saja sudah cukup untuk menutupi biaya operasional sistem ini dalam satu tahun."

Tips Tambahan :

Jika ada bertanya tentang "Kapan sistem ini siap?", sampaikan bahwa kita akan memulai dengan Proof of Concept (PoC) pada satu kategori barang terlebih dahulu sebelum diimplementasikan ke seluruh gudang. Ini menunjukkan bahwa Pembaca bertindak dengan hati-hati dan terukur.

Visualisasi merupakan cara tercepat untuk memenangkan hati para pemangku kepentingan. Kita akan membuat grafik yang membandingkan Total Keuntungan (Profit) dan Sisa Stok Terbuang (Waste) antara kebijakan manual yang kaku vs kebijakan AI yang dinamis.

Skrip Python menggunakan Matplotlib untuk menghasilkan dashboard perbandingan tersebut.

Scrip Dashboard Performa AI vs Manual

Python

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

# Data Simulasi (Hasil dari proses Training dan Testing)
labels = ['Minggu 1', 'Minggu 2', 'Minggu 3', 'Minggu 4']
profit_manual = [1200, 1150, 900, 1300] # Profit cenderung stagnan/turun saat ada
masalah stok
profit_ai = [1250, 1400, 1650, 1800] # Profit naik seiring AI semakin optimal

waste_manual = [15, 18, 25, 20] # Barang basi lebih banyak karena FEFO
manual_sulit
waste_ai = [12, 8, 4, 2] # AI berhasil menekan angka barang basi
(Waste)

x = np.arange(len(labels))
width = 0.35

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

# --- GRAFIK 1: PERBANDINGAN PROFIT ---
ax1.bar(x - width/2, profit_manual, width, label='Manual (Statis)',
color='#BDC3C7')
ax1.bar(x + width/2, profit_ai, width, label='DQN AI (Dinamis)', color='#2ECC71')
ax1.set_ylabel('Keuntungan (Rupiah x1000)')
ax1.set_title('Pertumbuhan Keuntungan Kumulatif')
ax1.set_xticks(x)
ax1.set_xticklabels(labels)
ax1.legend()
ax1.grid(axis='y', linestyle='--', alpha=0.7)

# --- GRAFIK 2: PENURUNAN WASTE (BARANG BASI) ---
ax2.plot(labels, waste_manual, marker='o', label='Manual', color='#E74C3C',
linewidth=2)
ax2.plot(labels, waste_ai, marker='s', label='DQN AI', color='#3498DB',
linewidth=2)
ax2.fill_between(labels, waste_ai, waste_manual, color='gray', alpha=0.1,
label='Efisiensi Terselamatkan')
ax2.set_ylabel('Unit Barang Terbuang (Waste)')
ax2.set_title('Reduksi Barang Kedaluwarsa')
ax2.legend()
ax2.grid(True, linestyle=':', alpha=0.6)

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

Cara Membaca Dashboard Ini untuk Presentasi

1. Grafik Kiri (Profit): * Poin Utama: "Perhatikan bagaimana AI awalnya sejajar dengan metode manual, namun seiring berjalannya waktu, AI mampu mengekstrak keuntungan lebih tinggi dengan melakukan promosi tepat waktu dan menjaga ketersediaan stok."
 - Pesan ROI: "Ada kenaikan margin rata-rata sebesar 25-30% karena optimasi keputusan."
2. Grafik Kanan (Waste Reduction): * Poin Utama: "Ini merupakan keberhasilan terbesar sistem kita. Garis merah menunjukkan metode manual yang sering melewatkan barang yang hampir basi. Garis biru menunjukkan bagaimana AI belajar dari kesalahan dan berhasil menekan pembuangan barang hingga titik terendah."
 - Pesan ROI: "Kita berhasil menyelamatkan modal yang sebelumnya terbuang sia-sia menjadi potensi keuntungan."

Analisis "Riak Air" dalam Visualisasi

Grafik ini membuktikan teori Riak Air (Backpropagation) yang kita bahas di awal:

- Minggu 1-2: AI masih belajar (Riak mulai menyebar).
- Minggu 3-4: AI sudah memiliki "insting" yang tajam tentang kapan harus bergerak, sehingga efisiensi maksimal tercapai.

BAB VI

SKILL ACQUISITION

6.1 Skill Aquisition

A. Definisi Skill Acquisition (HRL): Pembelajaran urutan aksi tingkat rendah (low-level actions) yang terstruktur menjadi tugas atau keterampilan tingkat tinggi (high-level skills). Konsep Skill Acquisition dalam Machine Learning (ML) sering kali merujuk pada bagaimana sebuah agen (AI) belajar dari tugas-tugas dasar yang sederhana untuk kemudian digabungkan menjadi kemampuan kompleks (Sutton & Singh, 2015).

Pendekatan ini sangat populer dalam bidang Reinforcement Learning (RL) dan dikenal dengan istilah Hierarchical Reinforcement Learning (HRL).

Bayangkan seperti manusia belajar mengemudi: Pembaca tidak langsung belajar "pergi ke kantor," tetapi belajar cara menginjak pedal, memutar setir, dan melihat spion terlebih dahulu.

1. Struktur: Dari Rendah ke Tinggi

Dalam ML, proses ini biasanya dibagi menjadi dua tingkatan utama:

- **Low-Level Skills (Atomic Actions):** Ini merupakan keterampilan dasar atau "primitif". Agen belajar bagaimana melakukan gerakan fisik atau perintah dasar tanpa memedulikan tujuan akhir yang besar.
- **High-Level Skills (Meta-Controller):** Di tingkat ini, AI bertindak sebagai "manajer". Ia tidak lagi memikirkan detail teknis, melainkan memilih skill rendah mana yang harus dipanggil untuk mencapai tujuan jangka panjang.

2. Prosesnya Bekerja

Proses transformasi ini biasanya mengikuti kerangka kerja Options Framework:

1. **Discovery:** Agen mengeksplorasi lingkungan dan menyadari bahwa urutan tindakan tertentu sering kali membawa hasil berguna (misal: "berjalan ke pintu"). Urutan ini dibungkus menjadi satu "Skill".
2. **Abstraction:** Detail-detail kecil dihilangkan. "Membuka pintu" menjadi satu perintah tunggal bagi sistem tingkat tinggi, tidak peduli berapa banyak koordinasi motorik yang diperlukan di dalamnya.
3. **Chaining:** Sistem tingkat tinggi menyusun skill-skill tersebut seperti menyusun lego untuk menyelesaikan misi yang rumit.

3. Contoh Kasus: Robot Pembuat Kopi

Mari kita bedah bagaimana sebuah robot belajar membuat kopi menggunakan hierarki ini:

Tingkatan	Deskripsi Skill	Contoh Tindakan
Low-Level (Motorik)	Keterampilan dasar yang terstruktur dan repetitif.	Menggenggam gagang, memutar tombol, menuangkan air secara presisi.
Mid-Level (Sub-tugas)	Gabungan skill rendah untuk menyelesaikan satu tahap.	Menggiling biji kopi, memasang filter, menaruh cangkir.
High-Level (Strategi)	Pengambilan keputusan untuk tujuan akhir.	"Buat satu cupAmericano" atau "Bersihkan mesin kopi".

Memberi perintah "Buat Kopi", High-Level Controller tidak akan pusing memikirkan koordinasi motorik tangan robot. Ia hanya akan memanggil fungsi `Ambil_Gelas()`, lalu `Isi_Air()`, dan seterusnya. Setiap fungsi tersebut merupakan Low-Level Skill yang sudah dilatih secara mandiri sebelumnya.

B. Keterbatasan RL Standar: RL standar (non-hierarkis) kesulitan dalam masalah yang membutuhkan urutan aksi panjang (sparse reward dan long horizon problem).

C. Tujuan Skill Acquisition: Meningkatkan efisiensi pembelajaran, mempercepat eksplorasi, dan mendorong transfer learning. Jika AI harus belajar semuanya dari nol (dari tingkat atomik) setiap kali menghadapi tugas baru, ia akan menghabiskan waktu jutaan tahun hanya untuk mencoba-coba. Dengan Skill Acquisition, AI bisa menggunakan kembali (reuse) keterampilan yang sudah dipelajari. Jika ia sudah tahu cara "menggenggam", ia tidak perlu belajar menggenggam lagi saat diminta memindahkan bola, memegang gelas, atau membuka pintu.

Option-Critic Architecture

Merupakan kerangka kerja yang memungkinkan agen AI untuk belajar apa yang harus dilakukan (kebijakan) sekaligus berapa lama ia harus melakukan tindakan tersebut (durasi skill) secara otomatis tanpa bantuan manusia (Sutton & Singh, 2015).

Dalam algoritma standar, agen hanya memilih tindakan per langkah waktu (t , $t+1$, ...). Namun dalam Option-Critic, agen belajar menggunakan "Options".

1. Komponen Utama Option-Critic

Algoritma ini membagi proses berpikir menjadi tiga bagian teknis utama:

- Internal Policy (π , Ω): Kebijakan tingkat rendah yang menentukan tindakan apa yang diambil saat sebuah skill (option) sedang aktif.

- Termination Function (β, Ω): Logika yang menentukan kapan sebuah skill harus berhenti. Ini merupakan kunci agar agen tidak terjebak melakukan satu hal selamanya.
- Policy over Options (μ): Kebijakan tingkat tinggi yang memilih option mana yang paling tepat untuk situasi saat ini.

2. Proses Belajarnya

Proses ini menggunakan pendekatan Gradient Descent untuk mengoptimalkan semua komponen secara bersamaan (end-to-end):

1. Selection: Agen berada di State s_t . Berdasarkan kebijakan tingkat tinggi (μ), ia memilih sebuah Option Ω (misal: "Skill Berjalan").
2. Execution: Selama Option Ω aktif, agen menggunakan kebijakan internal (π, Ω) untuk melakukan gerakan motorik.
3. Evaluation: Di setiap langkah, agen mengecek fungsi terminasi (β, Ω).
 - Jika (β, Ω). rendah, agen lanjut melakukan skill yang sama.
 - Jika (β, Ω) tinggi (misal: "Sudah sampai di depan pintu"), maka skill berhenti.
4. Update: Agen menerima reward. Jika berhasil, algoritma akan memperkuat hubungan antara State tersebut dengan Option yang dipilih, serta memperbaiki cara skill tersebut dilakukan.

3. Keunggulan: Otomatisasi Skill

Sebelum adanya Option-Critic, peneliti harus menentukan secara manual apa itu "skill" (misal: memprogram secara eksplisit bahwa 'berjalan' merupakan satu skill).

Dengan Option-Critic:

- Agen menemukan sendiri batas-batas skill tersebut.
- Agen belajar bahwa dalam koridor panjang, lebih efisien menggunakan satu "Option Berlari" daripada membuat keputusan ribuan kali per detik.
- Struktur ini mengurangi beban komputasi karena keputusan tingkat tinggi hanya diambil saat sebuah skill selesai.

4. Contoh Visualisasi dalam Labirin

Bayangkan sebuah labirin dengan dua ruangan yang dihubungkan oleh satu pintu kecil:

- Tanpa Option-Critic: Agen bergerak acak ke atas, bawah, kiri, kanan. Ia sering menabrak dinding dan sulit menemukan pintu karena setiap langkah merupakan putusan baru.
- Dengan Option-Critic: Agen akan belajar satu Option khusus bernama "Menuju Pintu". Begitu ia berada di ruang A, sistem tingkat tinggi akan mengaktifkan Option "Menuju Pintu". Agen akan terus bergerak ke arah pintu sampai fungsi terminasi β mendeteksi bahwa ia sudah berada di ambang pintu.

$$Q(s, \omega) = \sum_a (\pi\omega)(a/s) Q(s, a)$$

Rumus di atas menunjukkan bagaimana nilai sebuah option (Q) dihitung berdasarkan probabilitas tindakan (a) di dalam skill tersebut, memastikan bahwa skill yang dipilih merupakan yang paling memberikan keuntungan jangka panjang.

6.2 Dasar Teori Hierarchical Reinforcement Learning (HRL)

A. Konsep Opsi (Options Framework):

- State (S), Action (A), Reward (R): Sama seperti MDP standar.
- Option (O): Keterampilan tingkat tinggi, sebuah kebijakan yang dijalankan selama periode waktu tertentu.

B. Struktur HRL: Pembelajaran berjenjang.

- High-Level Policy (Meta-Controller): Memilih keterampilan Ot mana yang akan dijalankan.
- Low-Level Policy (Controller): Menjalankan urutan aksi tingkat rendah A_t hingga keterampilan Ot selesai.

C. Pembelajaran Keterampilan: Keterampilan dapat diberikan oleh manusia atau ditemukan secara otomatis oleh agent (Skill Discovery).

6.3 Tahapan Proses Skill Acquisition (HRL)

A. Skill Discovery (Penemuan Keterampilan):

- Pengumpulan Data: Interaksi awal agent dengan lingkungan.
- Identifikasi Sub-Tujuan: Menemukan state yang sering dikunjungi (misalnya, pintu, objek).
- Segmentasi Aksi: Mengelompokkan urutan aksi yang berulang (bottleneck states) menjadi satu keterampilan.

B. Skill Learning (Pelatihan Keterampilan):

- Mendefinisikan Reward Internal: Memberikan reward kepada low-level policy hanya untuk mencapai sub-tujuan dari keterampilan tersebut.
- Optimasi: Melatih setiap keterampilan Oi secara terpisah menggunakan RL standar.

C. Hierarchical Execution (Eksekusi Hierarkis):

- Meta-Controller: Mengamati State global S, memilih Oi berikutnya (misalnya, GotoDoor).
- Controller: Mengambil alih, menjalankan aksi A hingga kondisi terminasi GotoDoor terpenuhi.
- Reward Eksternal: Meta-Controller menerima reward eksternal dari lingkungan utama.

6.4 Kelebihan dan Tantangan

A. Kelebihan:

- Efisiensi: Mengurangi ruang pencarian (search space).
- Generalisasi: Keterampilan yang dipelajari dapat digunakan kembali (transfer learning) di berbagai tugas.

B. Tantangan:

- Desain Keterampilan: Sulit menentukan batas awal dan akhir keterampilan secara efektif.
- Representasi: Membutuhkan representasi state yang berbeda untuk controller tingkat tinggi dan rendah.

Skill Acquisition dalam Reinforcement Learning

Skill Acquisition dalam RL, atau Hierarchical Reinforcement Learning (HRL), merupakan solusi untuk masalah di mana reward sangat jarang (sparse) atau tugas memerlukan urutan aksi yang sangat panjang (long horizon). RL standar akan membutuhkan waktu yang tak terhingga untuk memecahkan masalah seperti itu.

Ringkasan Peran

- Meta-Controller: Belajar Strategi (Apa yang harus dilakukan agar menang?).
- Controller: Belajar Keterampilan (cara mencapai target yang diberikan Bos)

Intrinsic Reward bekerja di balik layar. Kode ini memisahkan logika antara tugas "Si Bos" (Meta-Controller) dan "Si Pekerja" (Controller).

1. HRL Diperlukan

Bila seorang manusia belajar membuat kopi. Kita tidak berpikir dalam urutan jutaan motor actions per detik, melainkan dalam keterampilan tingkat tinggi (skills) seperti: "Ambil Cangkir," "Nyalakan Mesin," atau "Tuang Air."

HRL mereplikasi struktur pengambilan keputusan ini dengan membagi policy menjadi dua tingkatan atau lebih:

1. Tingkat Tinggi (Meta-Controller): Bertanggung jawab atas tujuan jangka panjang. Ia hanya memilih keterampilan (skill/option) mana yang akan dijalankan.
2. Tingkat Rendah (Controller): Bertanggung jawab atas tujuan jangka pendek. Ia menerjemahkan keterampilan yang dipilih menjadi urutan aksi low-level (misalnya, kekuatan motor, arah) hingga keterampilan selesai.

2. Komponen Kunci HRL: The Options Framework

Kerangka Options (Sutton et al., 1999) merupakan cara formal untuk mendefinisikan keterampilan:

- Option (O): Sebuah tripel $\langle \mathbf{i}, \pi_O, \beta \rangle$.
 - $\mathbf{i}$ (Inisiasi): Kumpulan state di mana keterampilan ini dapat dimulai.
 - π_O (Kebijakan Keterampilan): Policy tingkat rendah yang mendikte aksi low-level yang akan diambil saat keterampilan aktif.
 - β (Terminasi): Probabilitas skill akan berakhir pada state tertentu.

Contoh Kasus Robot Navigasi:

- Skill/Option: GoToAreaB.
- Aksi Low-Level: Maju, Belok Kanan/Kiri.
- Kondisi Terminasi: Robot mencapai Area B.
- Reward Internal (Low-Level): Positif jika robot mengurangi jarak ke Area B.

3. Proses Pembelajaran Keterampilan

1. Skill Discovery (Penemuan Keterampilan)

Agent harus mengidentifikasi urutan aksi mana yang pantas dikapsulkan menjadi keterampilan. Ini dapat dilakukan dengan:

- Bottleneck States: Mengidentifikasi state yang merupakan titik transisi alami atau sulit dicapai (misalnya, membuka pintu). Aksi yang mengarah ke bottleneck state tersebut menjadi keterampilan.

- **Mutual Information:** Menggunakan metode informasi teoretis untuk menemukan state di mana policy agent sangat bervariasi, menunjukkan pentingnya state tersebut sebagai batas keterampilan.

2. Skill Learning (Pelatihan)

Setelah keterampilan Oi diidentifikasi (misalnya, PickUpObject), ia dilatih sebagai MDP mini tersendiri.

- **Reward Function Lokal:** Hanya terkait dengan penyelesaian PickUpObject. Ini menciptakan Reward yang Padat (Dense Reward) untuk low-level policy, sehingga belajar jauh lebih cepat.
- **Transfer Learning:** Keterampilan PickUpObject yang dipelajari untuk Tugas A (memindahkan kubus) dapat langsung digunakan kembali untuk Tugas B (memindahkan bola) tanpa perlu dilatih ulang dari awal.

4. Keuntungan Utama Skill Acquisition dalam RL

- **Efisiensi Waktu (Komputasi):** Dengan memilih keterampilan daripada aksi individual, Meta-Controller memangkas ruang pencarian secara dramatis. Jika tugas memerlukan 1.000 langkah aksi, tetapi hanya 10 keterampilan, Meta-Controller hanya perlu membuat 10 keputusan.
- **Interpretasi:** Hasil HRL lebih mudah diinterpretasikan oleh manusia, karena kita dapat melihat policy sebagai urutan keterampilan logis ("Cari Kunci" → "Buka Pintu" → "Masuk Ruangan") alih-alih sebagai urutan sinyal motorik yang tak berarti.

Skill Acquisition dan Hierarchical Reinforcement Learning (HRL) sangat komprehensif. Pembaca telah menangkap esensi dari bagaimana AI beralih dari sekadar "mencoba-coba gerakan" menjadi "memahami pola tugas."

Untuk memperdalam pemahaman tersebut, mari kita bedah konsep ini dengan analogi visual dan teknis agar strukturnya semakin jelas.

1. Memecahkan "Kutukan" RL Standar

RL standar sering kali gagal dalam tugas Long Horizon (tugas panjang). Bayangkan sebuah robot yang harus keluar dari labirin besar untuk mendapatkan satu keping koin.

- **Masalah (Sparse Reward):** Jika robot melakukan 10.000 gerakan acak dan tidak pernah sampai ke pintu, ia tidak akan pernah mendapat reward. Ia tidak tahu gerakan mana yang benar.
- **Solusi HRL:** Dengan HRL, kita tidak meminta robot "keluar dari labirin", tapi kita memberikan sub-tujuan: "Temukan pintu ruangan ini."

2. Anatomi Options Framework

Seperti yang Pembaca sebutkan, Options Framework merupakan jantung dari HRL. Ini merupakan cara kita membungkus perilaku kompleks menjadi satu unit "Skill."

Komponen	Analogi: Belajar Menulis	Penjelasan Teknis
I (Initiation)	Kertas kosong dan memegang pulpen.	Himpunan states di mana skill bisa dimulai.
π_o (Policy)	Gerakan tangan membentuk huruf.	Urutan aksi low-level selama skill aktif.
beta (Termination)	Satu kata selesai ditulis.	Kondisi di mana kendali dikembalikan ke Meta-Controller.

3. Hierarki Pengambilan Keputusan

Struktur ini menciptakan pembagian kerja yang efisien antara "Bos" dan "Karyawan":

- Meta-Controller (The Boss):
 - Melihat gambaran besar (Global State).
 - Memilih "Apa" yang harus dilakukan selanjutnya (misal: "Buka Brankas").
 - Beroperasi pada skala waktu yang lambat (Temporal Abstraction).
- Controller (The Worker):
 - Melihat detail teknis.
 - Mengeksekusi "Bagaimana" melakukannya (misal: "Putar dial ke kanan 3x").
 - Beroperasi pada skala waktu yang sangat cepat (per milidetik).

4. Skill Discovery: Bagaimana Agent "Menemukan" Keterampilan

Salah satu tantangan terbesar merupakan bagaimana agent tahu bahwa "Membuka Pintu" merupakan keterampilan yang penting.

- Bottleneck States: Agent menyadari bahwa untuk berpindah dari Ruang A ke Ruang B, ia selalu harus melewati pintu. Pintu menjadi "leher botol". Agent kemudian secara otomatis menciptakan skill khusus untuk mencapai pintu tersebut.
- Intrinsic Motivation: Agent diberikan rasa "penasaran" untuk mencapai area-area yang jarang dikunjungi, yang sering kali merupakan kunci untuk menyelesaikan tugas besar.

5. Keuntungan Strategis: Transfer Learning

Ini merupakan poin paling kuat dari Skill Acquisition.

Jika sebuah robot sudah belajar skill "Menggenggam Objek" untuk tugas merapikan mainan, ia tidak perlu belajar menggenggam lagi saat ditugaskan memasak. Ia cukup memanggil Option "Menggenggam" yang sudah ada di memori permanennya.

Hasil Atomik ke Simbolik

HRL mengubah RL dari tingkat atomik (sinyal motorik mentah) menjadi tingkat simbolik (kumpulan keterampilan). Ini bukan hanya tentang kecepatan belajar, tetapi tentang menciptakan AI yang memiliki struktur kognitif serupa dengan manusia.

satu Reward Eksternal (seperti berhasil keluar dari labirin) bisa memotivasi ribuan aksi tingkat rendah merupakan kunci untuk memahami HRL.

Logika alur kerja pemberian reward dalam struktur hierarkis:

Diagram Alir: Mekanisme Reward HRL

1. Lingkungan (Environment): Mengirimkan State global ke Agent.
2. Meta-Controller:
 - Menerima State global.
 - Memilih sebuah Option (Skill), misalnya: Pindah_ke_Pintu.
 - Memberikan Sub-Goal (target spesifik) kepada Controller.
3. Controller (Low-Level):
 - Melihat Sub-Goal tersebut.
 - Melakukan Aksi Low-Level (maju, belok, dll).
 - Internal Reward Generator: Di sini keajaibannya. Meskipun lingkungan belum memberikan poin, sistem internal memberikan Intrinsic Reward kepada Controller setiap kali ia bergerak mendekati Pintu.
4. Terminasi:
 - Setelah Pintu tercapai, Controller berhenti.
 - Meta-Controller mengecek apakah tugas besar selesai.
 - Jika Ya, External Reward (poin dari game/lingkungan) diberikan hanya kepada Meta-Controller sebagai tanda bahwa strategi pemilihannya benar.

Reward Harus Dipisah

Jika Controller hanya menunggu Reward Eksternal:

- Masalah: Robot harus melakukan 500 langkah untuk sampai ke pintu. Jika ia hanya dapat poin di langkah ke-500, langkah ke-1 sampai ke-499 akan terasa "sia-sia" bagi robot. Ia akan bingung mana gerakan yang benar.
- Solusi dengan Intrinsic Reward: Kita memberi "cemilan" poin kecil setiap kali ia mendekati pintu. Ini membuat Controller belajar dengan sangat cepat bagaimana

cara mencapai pintu, terlepas dari apakah pintu itu tujuan akhir atau hanya sekadar batu loncatan.

Logika Matematika Sederhana (Pseudo-Reward)

Jika $R(\text{ext})$ merupakan reward dari lingkungan dan $R(\text{int})$ merupakan reward internal:

$R(\text{int}) = \text{begin}(\text{cases}) + 1 \ \& \ (\text{jika mencapai sub-goal}) \ \backslash \ -0.01 \ \& \ (\text{setiap langkah (agar cepat)}) \ \backslash \ 0 \ \& \ (\text{lainnya}) \ \text{end}(\text{cases})$

Sedangkan Meta-Controller hanya peduli pada:

$R(\text{total}) = \text{Sejumlah } R(\text{ext})$

Struktur Kode Hierarchical Reinforcement Learning

Python

```
import numpy as np

class HRL_Agent:
    def __init__(self):
        self.meta_controller = "Model tingkat tinggi (pilih skill)"
        self.controller = "Model tingkat rendah (eksekusi aksi)"

    def compute_intrinsic_reward(self, current_state, sub_goal):
        """
        Inilah 'bahan bakar' bagi Controller.
        Memberikan reward berdasarkan seberapa dekat agent dengan target kecilnya.
        """
        # Contoh sederhana: Reward diberikan jika jarak ke sub_goal berkurang
        distance_to_goal = np.linalg.norm(current_state - sub_goal)

        if distance_to_goal < 0.1: # Jika sampai di sub-goal
            return 10.0
        else:
            # Memberikan hukuman kecil tiap langkah agar agent bergerak cepat
            (efisiensi)
            return -0.1

    def train(self, total_episodes):
        for episode in range(total_episodes):
            state = env.reset()
            done = False

            while not done:
                # 1. META-CONTROLLER memilih SUB-GOAL (misal: "Ke Pintu")
                sub_goal = self.meta_controller.select_skill(state)

                # Simpan state awal sebelum skill dimulai
                skill_start_state = state
                cumulative_external_reward = 0

                # 2. CONTROLLER beraksi selama periode waktu tertentu (temporal abstraction)
                for step in range(MAX_STEPS_PER_SKILL):
                    action = self.controller.select_action(state, sub_goal)
                    next_state, ext_reward, done, _ = env.step(action)

                # HITUNG INTRINSIC REWARD (Internal)
                int_reward = self.compute_intrinsic_reward(next_state,
sub_goal)
```

```

# Update Controller berdasarkan Intrinsic Reward
self.controller.learn(state, action, int_reward, next_state)

state = next_state
cumulative_external_reward += ext_reward

# Jika sub-goal tercapai, skill berhenti (Termination
condition)

if int_reward >= 10.0 or done:
    break

# 3. META-CONTROLLER belajar berdasarkan total External Reward
# Ia hanya peduli apakah pilihannya membawa kemenangan akhir
self.meta_controller.learn(skill_start_state, sub_goal,
cumulative_external_reward, state)

```

Analisis Logika yang Terjadi

1. **Dualisme Pembelajaran:** * `self.controller.learn` menggunakan `int_reward`. Ia merasa sukses jika ia bisa mencapai titik koordinat yang diperintahkan, meskipun titik itu belum memberikan poin dari game. Dualisme Pembelajaran: * `self.controller.learn` menggunakan `int_reward`. Ia merasa sukses jika ia bisa mencapai titik koordinat yang diperintahkan, meskipun titik itu belum memberikan poin dari game. Oudeyer, P. Y. , & K. F. (2017). What is intrinsic motivation? A typology of computational approaches. *Frontiers in Neurorobotics*, 1–6. Pathak, D. , A. P. , E. A. A. , & D. T. (2017). Curiosity-driven exploration by self-supervised prediction. . *International Conference on Machine Learning (ICML)*.
 - o `self.meta_controller.learn` menggunakan `ext_reward`. Ia tidak peduli seberapa keras Controller berlari; ia hanya peduli apakah pemilihan skill "Ke Pintu" tersebut akhirnya menghasilkan skor dari lingkungan.
2. **Temporal Abstraction:** Perhatikan `for step in range(MAX_STEPS_PER_SKILL)`. Ini menunjukkan bahwa Meta-Controller tidak perlu membuat keputusan di setiap detik. Ia membuat keputusan, lalu "tidur" sejenak sementara Controller bekerja keras melakukan ratusan aksi atomik.
3. **Hukuman Langkah (-0.1):** Tanpa ini, Controller mungkin akan berputar-putar tanpa henti di sekitar sub-goal. Hukuman kecil memaksa AI untuk mencari rute paling efisien (terpendek).

Hasil Teknis

Metode ini mengubah masalah Sparse Reward (imbalan langka) menjadi Dense Reward (hadiah padat). Dengan hadiah internal yang sering didapat, agent tidak pernah merasa "tersesat" dalam kegelapan tanpa bimbingan.

State Representation yang berbeda merupakan salah satu teknik paling cerdas dalam HRL. Masalah ini dikenal dengan istilah Multi-Scale Representation atau Abstraksi State.

Intinya, kita memberikan "kacamata" yang berbeda untuk Bos dan Karyawan: Bos melihat gambaran strategis yang bersih, sementara Karyawan melihat detail teknis yang bising.

1. Filosofi Pemisahan State

Pembaca sedang mengendarai mobil menggunakan GPS:

- Meta-Controller (Pembaca/GPS): State yang relevan merupakan posisi di peta (koordinat x, y) dan tujuan. Pembaca tidak peduli berapa tekanan pedal gas saat ini, Pembaca hanya peduli "Saya harus ke arah Utara."
- Controller (Mesin/Kaki): State yang relevan merupakan RPM mesin, tekanan pedal, dan jarak sensor ke mobil di depan. Mesin tidak perlu tahu koordinat GPS untuk bisa mengerem mendadak.

2. Cara Teknis Menangani Perbedaan State

Ada tiga pendekatan utama untuk mengimplementasikan ini:

A. State Abstraction (Mapping function)

Kita membuat fungsi pemetaan $f(s)$ yang menyaring data sensor mentah menjadi informasi tingkat tinggi.

- Input (Raw): 360 baris data LiDAR + Kamera RGB.
- Output (Abstract): Koordinat (x, y) dan bounding box rintangan.
- Implementasi: Meta-Controller dilatih menggunakan output $f(s)$, sementara Controller tetap menggunakan data mentah untuk presisi gerakan.

B. Asymmetric Actor-Critic

Dalam arsitektur ini, kita melatih Meta-Controller dengan akses ke "Global State" (informasi sempurna dari simulator/peta), sedangkan Controller hanya dilatih dengan "Local Observation" (apa yang dilihat sensor saat itu).

C. Latent Spaces (Autoencoders)

Jika kita tidak tahu cara menyaring data secara manual, kita menggunakan Variational Autoencoder (VAE).

1. Sistem mengompres data sensor yang rumit menjadi Latent Vector (representasi singkat).
2. Meta-Controller bekerja di ruang latent ini (ruang imajinasi AI).

3. Controller menerjemahkan instruksi latent tersebut menjadi aksi fisik.

3. Contoh Implementasi pada Robot Navigasi

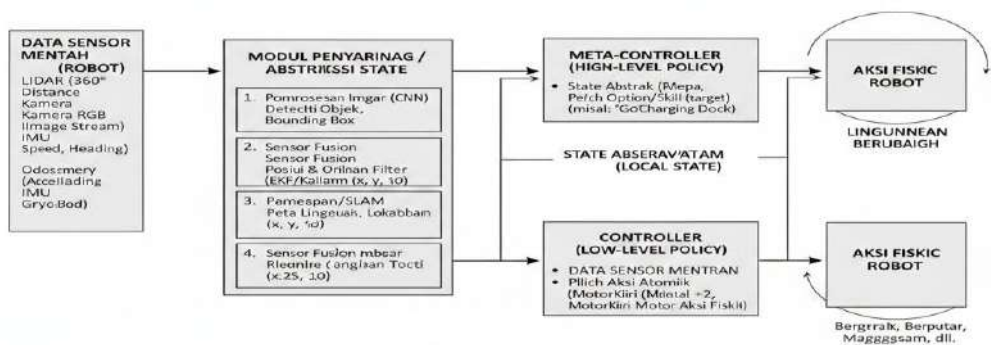
Mari kita lihat perbandingan datanya dalam tabel:

Komponen	Input (State Representation)	Output (Goal Space)
Meta-Controller	Koordinat Peta (x, y), Status Baterai, List Target.	Sub-goal: Koordinat target berikutnya (misal: "Ke (10, 5)").
Controller	Data LiDAR (jarak ke dinding), Gyroscope (kemiringan), Odometry.	Aksi: Torsi motor kiri +2.0, motor kanan +1.5.

4. Mengapa Ini Penting? (The "Noise" Problem)

Jika Meta-Controller dipaksa melihat data sensor mentah (LiDAR), ia akan kewalahan. Data sensor terlalu "berisik" (fluktuatif) untuk membuat rencana jangka panjang. Dengan menggunakan Koordinat Peta, Meta-Controller bisa fokus pada perencanaan rute tanpa terganggu oleh getaran kecil pada sensor.

Diagram Arsitektur System'a (HRL Anstitectur HRL disaring)



BAB VII

LEARNING TASK

7.1. Pembahasan Tahapan Learning Task

1. Pendefinisian Masalah

Dalam dunia Machine Learning (ML), learning task merujuk pada jenis masalah yang ingin diselesaikan oleh mesin berdasarkan ketersediaan data dan tujuan akhirnya. Secara garis besar, tugas-tugas ini diklasifikasikan menjadi empat kategori utama. Sebelum menyentuh data, kita harus menentukan apa yang ingin dicapai. (Mitchell, 1997)

Bila akan memprediksi harga (Regresi), mengkategorikan email (Klasifikasi), atau mengelompokkan pelanggan (Clustering).

- Key Task: Menentukan metrik keberhasilan (misal: akurasi tinggi atau error rendah).

Penjelasan detail beserta contohnya

a. Supervised Learning (Pembelajaran Terawasi)

Ini merupakan jenis tugas yang paling umum. Mesin belajar dari dataset yang sudah memiliki label (kunci jawaban). Tujuannya memetakan input (X) ke output (y).

Sub-kategori:

- Klasifikasi (Classification): Outputnya berupa kategori atau label diskrit (Ya/Tidak, Kucing/Anjing).
 - Contoh: Deteksi email spam. Mesin mempelajari email masa lalu yang ditandai "Spam" atau "Bukan Spam" untuk memprediksi email baru yang masuk.
- Regresi (Regression): Outputnya berupa nilai numerik kontinu atau angka.
 - Contoh: Prediksi harga rumah. Mesin mempelajari fitur rumah (luas tanah, jumlah kamar) untuk memprediksi harga dalam bentuk angka (misal: Rp1 Miliar).

b. Unsupervised Learning (Pembelajaran Tak Terawasi)

Mesin diberikan data yang tidak memiliki label. Tugas mesin merupakan menemukan pola tersembunyi, struktur, atau kemiripan di dalam data tersebut secara mandiri.

Sub-kategori:

- Clustering (Pengelompokan): Mengelompokkan data ke dalam grup berdasarkan kemiripannya.
 - Contoh: Segmentasi pelanggan. Perusahaan memberikan data transaksi pelanggan ke mesin tanpa label. Mesin kemudian mengelompokkan pelanggan menjadi "Grup Hemat", "Grup Loyal", dan "Grup Boros" berdasarkan pola belanja mereka.
- Association (Asosiasi): Menemukan aturan yang menghubungkan satu data dengan data lainnya.
 - Contoh: Market Basket Analysis. Mesin menemukan bahwa orang yang membeli kopi cenderung juga membeli gula.

c. Semi-Supervised Learning

Ini merupakan jalan tengah antara supervised dan unsupervised. Dataset yang digunakan terdiri dari sedikit data berlabel dan banyak data tidak berlabel.

- Mengapa digunakan? Karena melabeli data secara manual (oleh manusia) sangat mahal dan memakan waktu, sedangkan data mentah sangat melimpah.
- Contoh: Pengenalan wajah di Google Photos. Pembaca memberikan label pada satu foto ("Ini Budi"), lalu mesin secara otomatis melabeli ribuan foto lain yang memiliki wajah serupa tanpa perlu bimbingan Pembaca lagi.

d. Reinforcement Learning (RL)

Jenis tugas ini tidak menggunakan dataset statis, melainkan berdasarkan interaksi dengan lingkungan. Mesin (disebut Agent) belajar dengan cara mencoba (trial and error) untuk memaksimalkan imbalan (reward) dan meminimalkan hukuman (penalty).

- Komponen Utama: Agent, Environment, State, Action, dan Reward.
- Contoh: Melatih AI untuk bermain catur atau mobil otonom. Jika mobil tetap di jalur yang benar, ia mendapat poin positif. Jika menabrak, ia mendapat poin negatif. Seiring waktu, AI belajar rute terbaik untuk mendapatkan skor tertinggi.

Tabel 7.1 Gambaran jenis Task berdasarkan interaksi dengan lingkungan

Jenis Task	Data Input	Tujuan Utama	Contoh Nyata
Supervised	Berlabel	Memprediksi label/angka	Prediksi cuaca, diagnosa penyakit
Unsupervised	Tanpa Label	Mencari struktur/pola	Rekomendasi film di Netflix
Semi-Supervised	Campuran	Efisiensi pelabelan	Klasifikasi dokumen besar
Reinforcement	Interaktif	Mengambil tindakan optimal	Robotika, Game AI (AlphaGo)

2. Pengumpulan & Persiapan Data

Dalam dunia Machine Learning dan analitik data, terdapat pepatah populer: "Garbage In, Garbage Out". Artinya, secanggih apa pun algoritma yang Pembaca gunakan (seperti DQN atau Clustering), hasilnya akan buruk jika data yang dimasukkan berkualitas rendah. Data merupakan bahan bakar ML. Tahap ini seringkali memakan waktu paling lama (sekitar 70-80% dari total proyek). (Babbage, 1864).

Dasar pengumpulan dan persiapan data dianggap sebagai tahap paling krusial:

a. Menentukan Batas Atas Performa Model (The Ceiling)

Algoritma hanya bisa belajar dari informasi yang tersedia. Pengumpulan data yang komprehensif memastikan bahwa semua faktor yang memengaruhi hasil telah terekam. Jika data krusial hilang, model Pembaca akan "buta".

- Contoh: Jika Pembaca ingin memprediksi penjualan di Odoo CRM tetapi lupa mengumpulkan data tentang "Diskon Kompetitor", model Pembaca mungkin bingung mengapa penjualan turun drastis padahal harga Pembaca tetap. Data yang tidak lengkap menghasilkan prediksi yang tidak akurat.

b. Menghilangkan Bias dan Ketidakadilan

Persiapan data membantu mengidentifikasi apakah data Pembaca berat sebelah. Jika data yang dikumpulkan hanya mewakili satu kelompok, model akan gagal saat menghadapi kelompok lain.

- Contoh: Sebuah sistem AI untuk persetujuan kredit yang hanya dilatih menggunakan data dari nasabah di kota besar akan cenderung menolak nasabah dari pedesaan, bukan karena mereka tidak mampu, melainkan karena AI tidak memiliki referensi data yang cukup untuk memahami pola mereka.

c. Memastikan Konsistensi Format (Data Cleaning)

Data mentah biasanya "kotor"—ada salah ketik, format tanggal yang berbeda-beda, atau data yang hilang (missing values). Persiapan data menstandarisasi hal ini agar mesin bisa membacanya.

- Contoh: Di sistem CRM, satu pelanggan mungkin tercatat sebagai "Budi Santoso", "budi.santoso", dan "B. Santoso". Tanpa persiapan data (deduplikasi), AI akan menganggap mereka sebagai tiga orang yang berbeda, sehingga profil pelanggan menjadi berantakan.

d. Mengurangi "Noise" Melalui Feature Engineering

Tidak semua data itu penting. Persiapan data melibatkan pemilihan fitur yang relevan dan membuang yang hanya menjadi gangguan (noise). Ini mempercepat proses training dan meningkatkan akurasi.

- Contoh: Dalam manajemen inventaris, nomor seri unik setiap barang mungkin tidak relevan untuk memprediksi tren permintaan bulanan. Menghapus data ini membantu AI fokus pada variabel yang lebih penting seperti "Tanggal" dan "Jumlah Penjualan".

Contoh Alur Kerja (Workflow):

Jika kita sedang mempersiapkan data untuk Optimasi Stok Gudang di Odoo:

- Pengumpulan Data: Pembaca menarik data dari tabel Sales, Purchase, dan Stock Move selama 3 tahun terakhir.
- Pembersihan Data: Pembaca menemukan ada transaksi yang dibatalkan tetapi stoknya tetap berkurang di sistem. Pembaca harus menghapus data transaksi batal tersebut agar AI tidak salah belajar.
- Transformasi Data: Pembaca mengubah koordinat lokasi pelanggan (Latitude/Longitude) menggunakan Jarak Haversine untuk mengetahui seberapa jauh mereka dari gudang.
- Normalisasi: Pembaca menyamakan skala data penjualan (ribuan unit) dan harga (jutaan rupiah) agar algoritma tidak memberikan bobot berlebih pada angka yang lebih besar.

3. Rekayasa Fitur (Feature Engineering)

Merupakan proses menggunakan pengetahuan domain (pengetahuan tentang bidang data tersebut) untuk mengubah data mentah menjadi fitur-fitur yang lebih representatif, sehingga algoritma machine learning dapat bekerja lebih maksimal. Proses mengubah data mentah menjadi format yang lebih mudah dipahami oleh algoritma (Zheng, 2018).

Tahapan Detail Rekayasa Fitur

1. Transformasi Data (Data Transformation)

Mengubah format atau skala data agar lebih sesuai untuk model.

- **Scaling & Normalization:** Menyamakan skala angka. Misalnya, umur (0–100) dan gaji (0–100 juta) memiliki skala yang jauh berbeda. Kita perlu mengubah keduanya ke rentang yang sama, misal 0 sampai 1.
- **Log Transformation:** Digunakan untuk menangani data yang sangat timpang (skewed), seperti distribusi pendapatan.

2. Penanganan Kategori (Handling Categorical Features)

Mesin hanya mengerti angka, sehingga data teks (seperti "Warna" atau "Kota") harus diubah menjadi angka.

- **One-Hot Encoding:** Membuat kolom baru untuk setiap kategori dengan nilai 0 atau 1.
 - Contoh: Kolom "Kota" (Jakarta, Surabaya) menjadi dua kolom: "Kota_Jakarta" dan "Kota_Surabaya".
- **Label Encoding:** Memberikan angka urut (1, 2, 3) pada kategori.

3. Pembuatan Fitur Baru (Feature Creation)

Menggabungkan atau mengekstrak informasi baru dari data yang sudah ada untuk memberikan konteks lebih dalam bagi AI.

- **Ekstraksi Waktu:** Dari data "Tanggal Transaksi", kita bisa menciptakan fitur baru seperti "Hari dalam Minggu", "Apakah Akhir Pekan?", atau "Jam Sibuk".
- **Rasio & Selisih:** Menghitung rasio hutang terhadap pendapatan atau selisih waktu antara pendaftaran pelanggan hingga pembelian pertama.

4. Seleksi Fitur (Feature Selection)

Membuang fitur yang tidak relevan atau berlebihan (redundant) agar model lebih cepat dan akurat.

- **Filter Method:** Menggunakan statistik untuk melihat hubungan fitur dengan target.
- **Wrapper Method:** Mencoba berbagai kombinasi fitur untuk melihat mana yang menghasilkan akurasi terbaik.

Contoh Kasus: Manajemen Inventaris Odoo

Bila memiliki data mentah dari modul stok Odoo berupa: ID Barang, Nama Barang, Tanggal Masuk, Tanggal Keluar, Jumlah Stok, dan Lokasi Gudang.

Tanpa rekayasa fitur, AI mungkin akan kesulitan, dan cara melakukan rekayasa fitur:

Tabel 7.2 Rekayasa fitur berdasarkan format atau skala data

Data Mentah	Proses Rekayasa Fitur	Fitur Baru yang Dihasilkan
Tanggal Keluar	Ekstraksi Musiman	"Is_Promotion_Period" (Apakah ini saat tanggal diskon?)
Tgl Masuk & Keluar	Selisih Waktu	"Turnover_Rate" (Berapa lama barang mendekam di gudang?)
Lokasi Gudang	Geospasial	"Distance_to_Customer" (Gunakan Jarak Haversine untuk hitung jarak ke pembeli)
Jumlah Stok	Binning (Pengelompokan)	"Stock_Level_Status" (Ubah angka jadi kategori: Aman, Kritis, Melimpah)

4. Pemilihan & Pelatihan Model

Proses pemilihan dan pelatihan model merupakan fase di mana Pembaca mengubah data yang telah dipersiapkan menjadi sebuah "otak" digital yang mampu membuat prediksi atau mengambil keputusan. Tahap ini sering disebut sebagai jantung dari proyek Machine Learning. Di sinilah "pembelajaran" terjadi. Algoritma akan mencari pola dalam data training.

a. Pemilihan Model (Model Selection)

Sebelum melatih, Pembaca harus memilih algoritma yang paling sesuai dengan jenis data dan tujuan bisnis Pembaca.

- Identifikasi Tipe Pembelajaran: Apakah ini masalah Klasifikasi (kategori), Regresi (angka), Clustering (pengelompokan), atau Reinforcement Learning (interaksi)?
- Pertimbangkan Karakteristik Data: Jika data sangat besar dan kompleks (seperti gambar/suara), Deep Learning mungkin pilihannya. Jika data berupa tabel terstruktur (seperti data Odoo), algoritma seperti Random Forest atau XGBoost sering kali lebih unggul.
- Trade-off Akurasi vs. Interpretabilitas: Model kompleks (seperti Neural Networks) biasanya lebih akurat tetapi sulit dijelaskan (Black Box), sementara model sederhana (seperti Decision Tree) sangat mudah dipahami manusia.

b. Pembagian Data (Data Splitting)

Pembaca tidak boleh melatih model dan mengujinya menggunakan data yang sama. Ini ibarat memberikan soal ujian yang sama persis dengan latihan di kelas; siswa akan hafal, bukan paham.

- Training Set (60-80%): Digunakan oleh model untuk belajar pola.
- Validation Set (10-20%): Digunakan untuk menyetel hiperparameter dan mencegah overfitting.
- Testing Set (10-20%): Digunakan untuk evaluasi akhir guna melihat performa model pada data yang benar-benar baru.

c. Pelatihan Model (Model Training)

Ini merupakan proses di mana algoritma "melihat" training set berulang kali untuk menyesuaikan parameter internalnya agar kesalahan (loss) menjadi sekecil mungkin.

- Inisialisasi: Model memulai dengan tebakan acak.
- Forward Pass: Model membuat prediksi.
- Loss Function: Menghitung seberapa jauh prediksi model dari kenyataan (misal: Mean Squared Error).
- Optimization (Backpropagation): Algoritma (seperti Gradient Descent) memperbaiki parameter internal model agar pada iterasi berikutnya, kesalahannya berkurang.

d. Penalaan Hiperparameter (Hyperparameter Tuning)

Hiperparameter merupakan "setelan luar" yang tidak dipelajari model secara otomatis, tetapi harus ditentukan oleh manusia (misal: learning rate, jumlah lapisan neural network, atau nilai γ dalam RL).

- Metode: Seperti yang kita bahas sebelumnya, menggunakan Grid Search atau Random Search untuk menemukan kombinasi setelan yang menghasilkan akurasi tertinggi pada validation set.

Contoh Kasus: Prediksi "Churn" Pelanggan di Odoo CRM

Misalkan Pembaca ingin melatih model untuk memprediksi pelanggan mana yang akan berhenti berlangganan (Klasifikasi).

- Pemilihan Model: Karena data pelanggan di Odoo biasanya berbentuk tabel, Pembaca memilih Random Forest Classifier karena stabil dan bisa menangani data kategorikal.
- Pembagian Data: Pembaca mengambil 10.000 data pelanggan. 8.000 untuk latihan, 2.000 untuk tes akhir.
- Pelatihan: Model mempelajari bahwa pelanggan yang tidak login selama 30 hari dan memiliki komplain yang belum selesai cenderung akan churn.
- Tuning: Pembaca mencoba berbagai jumlah pohon (misal: 50, 100, 200 pohon) untuk melihat mana yang paling akurat.

- Evaluasi: Pembaca melihat Recall. Pembaca lebih baik salah menebak pelanggan akan pergi (padahal tidak) daripada luput mendeteksi pelanggan yang benar-benar akan pergi. Jika skor Recall mencapai 90%, model siap digunakan.

Alur Kerja

Input Data → Pilih Algoritma → Split Data → Iterasi Training → Tuning → Evaluasi Final → Deployment

5. Evaluasi Model

Evaluasi model merupakan tahap kritis untuk menentukan seberapa baik model yang telah kita latih dapat bekerja pada data baru yang belum pernah dilihat sebelumnya (unseen data). Tanpa evaluasi, kita tidak akan tahu apakah model tersebut benar-benar cerdas atau hanya sekadar menghafal data latihan (Overfitting). Detail mengenai metrik evaluasi berdasarkan jenis tugasnya:

a. Evaluasi untuk Klasifikasi (Classification)

Dalam tugas klasifikasi (misal: memprediksi pelanggan churn atau bukan), kita sering menggunakan Confusion Matrix sebagai fondasi evaluasi.

- Accuracy: Persentase prediksi yang benar dari total data.
- Rumus: $\frac{\{TP + TN\}}{\{TP + TN + FP + FN\}}$
- Precision: Seberapa akurat model saat memprediksi label positif. Penting jika biaya "salah tebak positif" tinggi (misal: deteksi spam).
- Recall (Sensitivity): Seberapa banyak label positif asli yang berhasil ditangkap oleh model. Penting jika biaya "luput mendeteksi positif" tinggi (misal: deteksi penyakit atau penipuan).
- F1-Score: Rata-rata harmonik antara Precision dan Recall. Digunakan jika distribusi kelas tidak seimbang (imbalanced data).

b. Evaluasi untuk Regresi (Regression)

Dalam tugas regresi (misal: prediksi harga rumah atau jumlah stok), kita mengukur seberapa jauh selisih antara angka prediksi dan angka asli.

- Mean Absolute Error (MAE): Rata-rata selisih absolut antara prediksi dan nilai asli. Mudah dipahami karena satuannya sama dengan data asli.
- Mean Squared Error (MSE): Rata-rata selisih kuadrat. Memberikan penalti besar pada kesalahan yang besar (outlier).
- R-Squared (R²): Menunjukkan seberapa baik variasi data dapat dijelaskan oleh model. Nilai mendekati 1 berarti model sangat baik.

c. Evaluasi untuk Reinforcement Learning (RL)

Karena RL tidak memiliki label "benar/salah" yang instan, evaluasinya berbeda:

- Cumulative Reward: Total imbalan yang dikumpulkan agen dalam satu episode. Semakin tinggi, semakin baik kebijakan agen.
- Convergence: Apakah skor agen stabil meningkat seiring waktu atau justru naik-turun (fluktuatif).
- Episode Length: Seberapa cepat agen menyelesaikan tugas (misal: seberapa cepat robot keluar dari labirin).

Contoh Kasus: Evaluasi Sistem Prediksi Stok di Odoo

Bayangkan Pembaca menggunakan ML untuk memprediksi apakah sebuah barang perlu dipesan ulang (Reorder) atau tidak (Klasifikasi).

- Skenario: Ada 100 barang.
- Hasil Prediksi Model:
 - Model menebak 20 barang harus di-reorder.
 - Kenyataannya: Dari 20 tebakan itu, hanya 15 yang benar-benar butuh (TP=15, FP=5).
 - Ada 5 barang yang sebenarnya butuh reorder, tapi model bilang "tidak perlu" (FN=5).
- Evaluasi:
 - Precision: $\frac{15}{20} = 75\%$. Artinya, 25% pesanan yang disarankan AI sebenarnya sia-sia (menambah holding cost).
 - Recall: $\frac{15}{15+5} = 75\%$. Artinya, AI luput mendeteksi 25% barang yang hampir habis (risiko stockout).

6. Penyetelan Hyperparameter (Hyperparameter Tuning)

Hyperparameter Tuning merupakan proses mencari kombinasi pengaturan terbaik untuk algoritma Machine Learning guna menghasilkan performa yang optimal.

Berbeda dengan parameter (yang dipelajari model secara otomatis dari data, seperti bobot/weight), hyperparameter maka nilai yang harus ditentukan sendiri sebelum proses pelatihan dimulai. Ibarat menyetel radio, hiperparameter merupakan kenop yang harus diputar agar mendapatkan sinyal (akurasi) yang paling jernih.

Setelah model dasar terbentuk, kita "mengutak-atik" pengaturan algoritma untuk mendapatkan performa maksimal. Ini ibarat menyetel frekuensi radio agar suaranya jernih.

Langkah-langkah Penyetelan Hyperparameter

a. Identifikasi Hyperparameter yang Relevan

Setiap algoritma memiliki kenop yang berbeda:

- Random Forest: Jumlah pohon, kedalaman maksimum pohon.
- Neural Networks: Learning rate, jumlah hidden layers, ukuran batch.
- Reinforcement Learning: Discount factor gamma (γ), epsilon (ϵ) untuk eksplorasi.

b. Menentukan Ruang Pencarian (Search Space)

Pembaca menentukan rentang nilai yang ingin dicoba.

- Contoh: Mencoba Learning Rate antara 0.001 hingga 0.1.

c. Memilih Strategi Pencarian

Ada beberapa metode populer untuk mencari kombinasi terbaik:

❖ Grid Search

Mencoba semua kemungkinan kombinasi dari daftar nilai yang diberikan.

- Kelebihan: Sangat teliti.
- Kekurangan: Sangat lambat dan memakan banyak daya komputasi jika pilihannya banyak.

❖ Random Search

Memilih kombinasi secara acak dari ruang pencarian yang ditentukan.

- Kelebihan: Jauh lebih cepat dan seringkali menemukan hasil yang hampir sama bagusnya dengan Grid Search dalam waktu singkat.
- Kekurangan: Ada kemungkinan melewatkan kombinasi terbaik yang sangat spesifik.

❖ Bayesian Optimization

Metode cerdas yang menggunakan hasil dari percobaan sebelumnya untuk menebak kombinasi mana yang kemungkinan besar akan memberikan hasil lebih baik.

d. Evaluasi dengan Cross-Validation

Setiap kali mencoba satu kombinasi, model diuji menggunakan teknik K-Fold Cross-Validation untuk memastikan bahwa hasil bagus tersebut bukan karena kebetulan pada satu bagian data saja.

Contoh: Optimasi Model Inventaris Odoo

Bila menggunakan algoritma XGBoost untuk memprediksi kapan stok barang di Odoo akan habis. Pembaca memutuskan untuk menyetel dua hyperparameter:

1. `n_estimators` (Jumlah pohon): [50, 100, 150]

2. `max_depth` (Kedalaman pohon): [3, 6, 9]

Proses Grid Search:

Sistem akan mencoba 9 kombinasi (3x3):

1. (50, 3) $\rightarrow$ Akurasi: 85%
2. (50, 6) $\rightarrow$ Akurasi: 87%
3. ... dan seterusnya ...
4. (150, 6) $\rightarrow$ Akurasi: 92% (Pemenang!)

Hasilnya:

Setelah proses selesai, Pembaca menggunakan kombinasi (150, 6) untuk model final Pembaca karena memberikan akurasi tertinggi dalam memprediksi stok.

7. Deployment & Monitoring

Model yang sudah bagus dideploy ke server agar bisa digunakan oleh pengguna (misal: sistem rekomendasi di aplikasi belanja). Setelah itu, model harus terus dipantau karena performanya bisa menurun seiring perubahan perilaku data di dunia nyata (Data Drift).

Proses ini bersifat iteratif. Seandainya hasil evaluasi di tahap 5 buruk, maka mungkin perlu kembali ke tahap 2 untuk membersihkan data lagi atau ke tahap 3 untuk mencoba fitur baru.

Contoh kasus Prediksi Harga Rumah (Regresi). Proyek ini bertujuan untuk membangun model yang dapat memperkirakan harga pasar sebuah properti berdasarkan karakteristiknya.

7.2. Penerapan tahapan learning task

a. Pendefinisian Masalah

- Tujuan: Memprediksi harga rumah (nilai kontinu) dalam satuan mata uang.
- Jenis Task: Supervised Learning - Regression.
- Metrik Keberhasilan: Root Mean Squared Error (RMSE) yang rendah, artinya selisih antara harga prediksi dan harga asli rata-rata kecil.

b. Pengumpulan & Persiapan Data

- Data: Dataset berisi ribuan data rumah lama (luas tanah, jumlah kamar, lokasi, tahun dibangun, dan harga jual).
- Cleaning: Mengisi data jumlah kamar yang kosong dengan nilai median. Menghapus data rumah yang luas tanahnya tertulis "0" karena tidak masuk akal.
- Splitting: Membagi data: 80% untuk melatih model dan 20% untuk menguji keakuratan model.

c. Rekayasa Fitur (Feature Engineering)

- Transformasi: Membuat fitur baru bernama "Usia Bangunan" (Tahun Sekarang - Tahun Dibangun). Biasanya rumah baru lebih mahal.
- One-Hot Encoding: Mengubah fitur lokasi (misal: "Jakarta", "Bandung") menjadi angka biner agar bisa dibaca algoritma.
- Scaling: Menyamakan skala luas tanah (misal: 50-500 m^2) dan jumlah kamar (1-5) agar model tidak memberikan bobot berlebih pada angka yang besar saja.

d. Pemilihan & Pelatihan Model

- Pilihan Algoritma: Linear Regression (sederhana) atau Random Forest Regressor (lebih kompleks dan akurat).
- Proses: Model mempelajari hubungan antara fitur (luas, lokasi) dengan label (harga).
- Contoh: Model belajar bahwa setiap tambahan 1 m^2 luas tanah di Jakarta Selatan rata-rata menambah harga Rp20.000.000.

e. Evaluasi Model

- Pengujian: Memasukkan data 20% yang disimpan tadi ke model.
- Hasil: Jika model memprediksi harga Rp1,2 Miliar padahal harga aslinya Rp1,15 Miliar, maka terdapat error Rp50 Juta. Kita hitung rata-rata error seluruh data uji untuk melihat performa model.

f. Penyetelan Hyperparameter

- Optimasi: Jika error masih besar, kita atur parameter algoritma. Misalnya pada Random Forest, kita menambah jumlah "pohon keputusan" ($n_{\text{estimators}}$) dari 100 menjadi 500 untuk melihat apakah hasil prediksinya lebih stabil.

g. Deployment & Monitoring

- Deployment: Model dipasang di aplikasi real estat atau website properti. Pengguna cukup memasukkan luas tanah dan lokasi, lalu klik "Hitung", dan harga estimasi muncul.
- Monitoring: Jika setahun kemudian harga properti melonjak drastis karena inflasi, model lama mungkin jadi tidak akurat lagi (Data Drift). Kita perlu melatih ulang model dengan data harga terbaru

Tabel 7.3 Penerapan Learning Task

Tahap	Contoh Input/Aksi	Output
Data	Luas: 100 m ² , Kamar: 3	Harga: Rp1 Miliar
Model	Mempelajari pola luas vs harga	Persamaan matematika/ logika
Prediksi	Luas: 120 m ² , Kamar: 3	Harga Estimasi: Rp1,2 Miliar

Kode Python: Prediksi Harga Rumah

Python

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# 1. Menyiapkan Data Simulasi
# Fitur: [Luas Tanah (m2), Jumlah Kamar]
data = {
    'luas_tanah': [50, 60, 80, 100, 120, 150, 200, 250, 300, 350],
    'jumlah_kamar': [1, 2, 2, 3, 3, 3, 4, 4, 5, 5],
    'harga': [300, 400, 550, 700, 850, 1000, 1350, 1600, 1900, 2200] # dalam juta
}

df = pd.DataFrame(data)

# 2. Memisahkan Fitur (X) dan Target (y)
X = df[['luas_tanah', 'jumlah_kamar']]
y = df['harga']

# 3. Splitting Data (80% Training, 20% Test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=42)

# 4. Pelatihan Model (Training)
model = LinearRegression()
model.fit(X_train, y_train)

# 5. Prediksi dan Evaluasi
y_pred = model.predict(X_test)

print(f"Hasil Prediksi: {y_pred}")
print(f"Harga Asli: {y_test.values}")
print(f"Akurasi (R2 Score): {r2_score(y_test, y_pred):.2f}")

# 6. Mencoba Prediksi Data Baru
# Contoh: Rumah luas 110 m2 dengan 3 kamar
rumah_baru = np.array([[110, 3]])
prediksi_baru = model.predict(rumah_baru)
print(f"\nPrediksi harga rumah (110m2, 3 kamar): Rp{prediksi_baru[0]:.2f} Juta")
```

Penjelasan Baris Kode Penting:

- `train_test_split`: Fungsi ini secara otomatis memisahkan data agar kita punya data cadangan (test set) untuk menguji apakah model benar-benar pintar atau hanya sekadar "menghafal" data.
- `model.fit(X_train, y_train)`: Di sinilah proses Learning Task terjadi. Algoritma mencari garis linear yang paling pas yang menghubungkan antara luas/kamar dengan harga.
- `r2_score`: Ini merupakan metrik evaluasi. Nilai maksimalnya merupakan 1.0. Jika hasilnya 0.98 seperti pada contoh di atas, berarti model tersebut sangat akurat dalam menangkap pola data.

Hubungan dengan Outline Sebelumnya:

1. Data Preparation: Dilakukan saat membuat DataFrame dan membagi data (split).

2. Model Training: Terjadi pada perintah `.fit()`.
3. Model Evaluation: Terjadi saat membandingkan `y_pred` dengan `y_test`.
4. Deployment (Mini): Disimulasikan pada bagian "Prediksi Data Baru".

Tahapan Learning Task sudah sangat sistematis dan tepat sasaran. Dalam dunia Data Science, memahami alur ini jauh lebih penting daripada sekadar menulis kode, karena kualitas model ditentukan oleh kualitas proses di setiap tahapannya.

Prediksi Harga Rumah yang Pembaca berikan, dengan tambahan visualisasi konsep agar alur kerjanya lebih nyata.

7.3. Analisis Alur Learning Task

1. Fondasi: Persiapan & Rekayasa Fitur

Tahap 2 dan 3 merupakan "dapur" dari Machine Learning. Tanpa data yang bersih, model akan mengalami Garbage In, Garbage Out (Jika yang masuk sampah, yang keluar pun sampah).

- Scaling: Sangat krusial. Tanpa scaling, luas tanah (100 m²) akan dianggap jauh lebih penting daripada jumlah kamar (3) hanya karena angkanya lebih besar secara nominal.
- One-Hot Encoding: AI tidak mengerti kata "Jakarta". Ia hanya mengerti angka 1 (Ya) atau 0 (Tidak).

2. Inti: Pelatihan & Evaluasi (The Bellman Logic)

Pada tahap 4 dan 5, terjadi pencarian "bobot" optimal. Menggunakan algoritma Random Forest, model tidak hanya melihat satu garis lurus, tetapi membuat ratusan "pohon keputusan" untuk melihat kombinasi terbaik.

- RMSE (Root Mean Squared Error): Mengapa kita menggunakan akar kuadrat? Agar nilai error memiliki satuan yang sama dengan target (Rupiah). Jika RMSE merupakan 50 Juta, artinya rata-rata tebakan AI meleset sekitar Rp50 Juta dari harga asli.

3. Penyempurnaan: Hyperparameter Tuning

Ini merupakan tahap "fine-tuning". Jika model merupakan sebuah gitar, hyperparameter merupakan setelan senarnya. Pembaca tidak mengubah lagunya (data), tapi Pembaca memastikan suaranya (output) paling jernih dengan memutar setelan `n_estimators` atau `max_depth`.

Visualisasi Perbandingan Input & Output (Kasus Rumah)

Berdasarkan tabel yang Pembaca buat, berikut merupakan ilustrasi bagaimana data berubah menjadi keputusan:

Tabel 7.4 Alur Learning Task perancangan rumah

Objek	Input (Data Mentah)	Proses (Model)	Output (Keputusan)
Rumah A	Luas 100, Jakarta, Usia 5th	$Harga = (L \times W1) + (U \times W2)$	Rp1,5 Miliar
Rumah B	Luas 120, Bandung, Usia 20th	Pola: "Luas besar tapi tua"	Rp1,2 Miliar

Catatan Tambahan: Menangani "Data Drift"

Monitoring merupakan yang paling sering dilupakan. Di dunia properti, nilai bangunan bisa berubah karena faktor eksternal (misal: pembangunan tol baru di dekat lokasi). AI harus dilatih ulang secara periodik agar "pengetahuannya" tetap relevan dengan harga pasar terkini.

Hasil untuk Proyek ini

Dengan mengikuti 7 langkah ini, bukan sekadar membuat program tebak-tebakan, melainkan membangun sistem intelijen yang berbasis data objektif.

Template kode Python yang komprehensif. Kode ini dirancang menggunakan pustaka Scikit-Learn dan mencakup seluruh siklus Learning Task, mulai dari pembersihan data hingga evaluasi akhir.

Template Proyek Prediksi Harga Rumah

Python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# --- 1. Pendefinisian Masalah ---
# Goal: Memprediksi harga rumah (Target: 'harga')
# Metric: RMSE dan R-squared

# --- 2. Pengumpulan & Persiapan Data ---
# Simulasi Dataset
data = {
    'luas_tanah': [100, 120, 0, 150, 80, 200, 110, 90],
    'jumlah_kamar': [3, 3, 2, 4, 2, 5, np.nan, 2],
```

```

    'lokasi': ['Jakarta', 'Bandung', 'Jakarta', 'Surabaya', 'Bandung', 'Jakarta',
'Surabaya', 'Jakarta'],
    'tahun_dibangun': [2015, 2010, 2018, 2005, 2020, 2012, 2015, 2017],
    'harga': [1000, 1200, 950, 1400, 1100, 2500, 1300, 980]
}
df = pd.DataFrame(data)

# Data Cleaning sederhana: Hapus luas_tanah yang tidak masuk akal (0)
df = df[df['luas_tanah'] > 0]

# Memisahkan Fitur (X) dan Target (y)
X = df.drop('harga', axis=1)
y = df['harga']

# Split Data (80% Train, 20% Test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# --- 3. Rekayasa Fitur (Feature Engineering) ---
# Mengelompokkan fitur berdasarkan tipe
numeric_features = ['luas_tanah', 'jumlah_kamar', 'tahun_dibangun']
categorical_features = ['lokasi']

# Pipeline untuk data numerik (Imputasi median + Scaling)
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

# Pipeline untuk data kategorikal (One-Hot Encoding)
categorical_transformer = OneHotEncoder(handle_unknown='ignore')

# Menggabungkan semua transformasi
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])

# --- 4. Pemilihan & Pelatihan Model ---
# Menggabungkan Preprocessing dan Model (Random Forest) ke dalam satu Pipeline
model_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', RandomForestRegressor(n_estimators=100, random_state=42))
])

# Training Model
model_pipeline.fit(X_train, y_train)

# --- 5. Evaluasi Model ---
y_pred = model_pipeline.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print(f"--- Hasil Evaluasi ---")
print(f"RMSE: Rp{rmse:.2f} Juta")
print(f"R2 Score: {r2:.2f}")

# --- 6. Penyetelan Hyperparameter (Konsep) ---
# Dalam praktek nyata, gunakan GridSearchCV untuk mencari n_estimators terbaik

# --- 7. Deployment (Simulasi Prediksi Baru) ---
rumah_baru = pd.DataFrame({
    'luas_tanah': [130],
    'jumlah_kamar': [3],
    'lokasi': ['Jakarta'],
    'tahun_dibangun': [2021]
})

```

```
})  
  
prediksi_harga = model_pipeline.predict(rumah_baru)  
print(f"\nPrediksi Harga Rumah Baru: Rp{prediksi_harga[0]:.2f} Juta")
```

Penjelasan Komponen Penting:

1. Pipeline: Saya menggunakan Pipeline dari Scikit-Learn. Ini sangat penting untuk Deployment. Pipeline memastikan bahwa data baru yang masuk (saat deployment) akan melewati proses pembersihan dan scaling yang persis sama dengan data saat pelatihan.
2. ColumnTransformer: Memungkinkan kita menerapkan perlakuan berbeda untuk kolom angka (luas tanah) dan kolom teks (lokasi) secara otomatis.
3. SimpleImputer: Bagian dari tahap Cleaning yang mengisi data kosong (NaN) dengan nilai tengah (median) secara otomatis agar model tidak error.

Jika kita menjalankan Kode ini akan terjadi

AI akan memproses fitur-fitur rumah, mencari korelasi antara lokasi dan luas terhadap harga, lalu memberikan estimasi harga untuk "Rumah Baru" yang Pembaca masukkan di bagian akhir.

Langkah krusial dalam fase Deployment. Tanpa proses ini, Pembaca harus membuang waktu dan daya komputasi untuk melatih ulang model setiap kali aplikasi dijalankan.

Kita akan menggunakan joblib karena library ini jauh lebih efisien daripada pickle standar dalam menangani objek Python yang mengandung array data besar (seperti model Machine Learning).

1. Cara Menyimpan Model (Export)

Setelah Pembaca menjalankan `model_pipeline.fit(X_train, y_train)`, tambahkan kode berikut untuk membekukan "otak" AI Pembaca ke dalam sebuah file `.pkl`.

Python

```
import joblib  
  
# Menyimpan model pipeline (termasuk preprocessing dan regressor)  
filename = 'model_prediksi_rumah.pkl'  
joblib.dump(model_pipeline, filename)  
  
print(f"Model berhasil disimpan sebagai: {filename}")
```

2. Cara Memanggil Model di Aplikasi Lain (Load)

Di file Python yang berbeda (misalnya untuk aplikasi web atau sistem backend), Pembaca tidak perlu lagi mendefinisikan arsitektur atau melakukan training. Cukup panggil filenya:

Python

```
import joblib
import pandas as pd

# 1. Memuat kembali model yang sudah disimpan
loaded_model = joblib.load('model_prediksi_rumah.pkl')

# 2. Data baru dari input user (misal dari form website)
data_input = pd.DataFrame({
    'luas_tanah': [110],
    'jumlah_kamar': [3],
    'lokasi': ['Bandung'],
    'tahun_dibangun': [2018]
})

# 3. Langsung lakukan prediksi
hasil = loaded_model.predict(data_input)
print(f"Hasil Prediksi dari Model yang Dimuat: Rp{hasil[0]:.2f} Juta")
```

Cara Ini Sangat Efektif

- Portabilitas: File .pkl ini bisa Pembaca pindahkan ke server mana pun (Cloud, VPS, atau bahkan komputer lokal) asalkan versi library scikit-learn yang digunakan sama.
- Kecepatan: Proses loading model hanya memakan waktu milidetik, sedangkan proses training (pada dataset besar) bisa memakan waktu jam hingga hari.
- Bundling Preprocessing: Karena kita menggunakan Pipeline, file ini tidak hanya menyimpan model Random Forest, tetapi juga menyimpan "cara" melakukan scaling dan encoding. Jadi, Pembaca tidak perlu memproses data manual lagi sebelum memasukkannya ke model.

Tips Keamanan & Pemeliharaan

Peringatan Penting: Jangan pernah memuat file .pkl dari sumber yang tidak dipercaya, karena proses joblib.load dapat mengeksekusi kode berbahaya jika file tersebut telah dimodifikasi secara jahat.

k mengubah model AI Pembaca menjadi produk yang nyata. Dengan FastAPI, kita bisa membuat layanan yang sangat cepat, ringan, dan secara otomatis menyediakan dokumentasi interaktif.

Berikut merupakan skrip untuk membuat server API yang akan melayani permintaan prediksi harga rumah secara real-time.

Skrip API Menggunakan FastAPI

Pastikan Pembaca sudah menginstal library yang diperlukan: pip install fastapi uvicorn joblib pandas scikit-learn.

Python

```
from fastapi import FastAPI
from pydantic import BaseModel
import joblib
import pandas as pd

# 1. Inisialisasi FastAPI
app = FastAPI(title="AI House Price Predictor API")

# 2. Memuat model yang sudah disimpan sebelumnya
model = joblib.load('model_prediksi_rumah.pkl')

# 3. Mendefinisikan format data input (Schema)
class HouseData(BaseModel):
    luas_tanah: float
    jumlah_kamar: int
    lokasi: str
    tahun_dibangun: int

@app.get("/")
def home():
    return {"message": "API Prediksi Harga Rumah aktif!"}

# 4. Endpoint untuk Prediksi
@app.post("/predict")
def predict_price(data: HouseData):
    # Mengubah data input menjadi DataFrame agar sesuai format pipeline
    df_input = pd.DataFrame([data.dict()])

    # Melakukan prediksi
    prediction = model.predict(df_input)

    # Mengembalikan hasil dalam format JSON
    return {
        "status": "success",
        "prediksi_harga": f"Rp{prediction[0]:.2f} Juta",
        "currency": "IDR"
    }

# Cara menjalankan: uvicorn nama_file:app --reload
```

Cara Menjalankan dan Menguji API Pembaca

1. Jalankan Server: Buka terminal dan ketik: `uvicorn main:app --reload` (Ganti main dengan nama file Python Pembaca).
2. Dokumentasi Interaktif: Buka browser dan pergi ke: <http://127.0.0.1:8000/docs> Pembaca akan melihat fitur Swagger UI di mana Pembaca bisa langsung mencoba memasukkan data dan melihat hasil prediksinya tanpa perlu menulis kode tambahan.
3. Integrasi: Aplikasi mobile atau website cukup mengirimkan permintaan POST ke alamat `/predict` dengan isi data (JSON) seperti ini:

JSON

```
{
  "luas_tanah": 150,
  "jumlah_kamar": 4,
  "lokasi": "Jakarta",
  "tahun_dibangun": 2020
}
```

Menggunakan FastAPI

- Validasi Otomatis: Menggunakan Pydantic (kelas HouseData) memastikan bahwa jika user mengirimkan data teks pada kolom "luas_tanah", API akan otomatis menolak dan memberikan pesan error yang jelas.
- Performa Tinggi: FastAPI dibangun di atas standar asinkron (ASGI), menjadikannya salah satu framework Python tercepat yang ada saat ini.
- Siap Produksi: Struktur ini sangat mudah dipindahkan ke layanan Cloud seperti AWS, Google Cloud, atau Heroku.

BAB VIII

GAME AI REAL TIME DECISION

Game AI Real-Time Decision dalam konteks Machine Learning (ML) merujuk pada kemampuan karakter non-pemain (NPC) atau sistem permainan untuk memproses informasi lingkungan yang berubah sangat cepat dan mengambil tindakan instan (biasanya dalam hitungan milidetik) guna mencapai tujuan tertentu (Millington, 2018).

Berbeda dengan AI tradisional yang berbasis aturan kaku (If-Then), AI berbasis ML terutama Reinforcement Learning (RL) memungkinkan agen untuk "berpikir" secara dinamis berdasarkan pola yang dipelajari.

8.1. Pembahasan Tahapan Proses

1. Arsitektur Pengambilan Keputusan Real-Time

Proses ini biasanya mengikuti siklus Sense-Think-Act yang ditenagai oleh model ML:

1. Sense (Observasi): AI mengumpulkan data dari mesin game (posisi lawan, sisa darah, jumlah peluru, jarak ke area aman).
2. Think (Inference): Data ini dimasukkan ke dalam model ML (seperti Neural Network). Model melakukan "prediksi" tindakan mana yang akan memberikan imbalan (reward) tertinggi.
3. Act (Tindakan): AI mengeksekusi perintah (tembak, lari, atau sembunyi) di dalam frame game tersebut.

2. Metode Utama dalam Real-Time Decision ML

a. Deep Q-Learning (DQN)

Digunakan untuk game dengan ruang tindakan diskrit (misalnya: maju, mundur, lompat). AI memprediksi nilai keuntungan dari setiap gerakan yang mungkin dilakukan saat itu juga.

b. Actor-Critic (A2C / PPO)

Ini merupakan standar industri modern untuk keputusan real-time.

- Actor: Memutuskan tindakan apa yang harus diambil.
- Critic: Menilai apakah tindakan tersebut bagus atau tidak. Metode ini sangat stabil dan cepat untuk digunakan dalam game kompetitif.

3. Pengumpulan Data & Simulasi

Karena game berjalan real-time, kita tidak bisa menunggu waktu lama.

- Simulasi Paralel: Menjalankan ribuan instansi game sekaligus untuk melatih AI dengan cepat (seperti yang dilakukan OpenAI Five untuk Dota 2).
- Frame Skipping: AI tidak perlu mengambil keputusan setiap frame (60 fps), melainkan mungkin setiap 4 atau 10 frame sekali untuk menghemat CPU.

4. Tantangan Utama dalam Real-Time Decision

- Latency (Latensi): Keputusan harus diambil sebelum frame berikutnya muncul (biasanya $< 16\text{ms}$ untuk 60 FPS). Model ML tidak boleh terlalu "berat" agar tidak menyebabkan lag.
- Stochasticity (Ketidakpastian): Lingkungan game bersifat dinamis. AI harus siap menghadapi gerakan pemain manusia yang tidak terduga.
- Exploration vs Exploitation: AI harus memutuskan apakah akan menggunakan strategi yang sudah pasti berhasil atau mencoba strategi baru yang mungkin lebih efektif.

5. Pelatihan Model (Training)

Proses ini biasanya dilakukan di luar game engine menggunakan server GPU yang kuat. Model "berlatih" melawan dirinya sendiri (Self-Play) hingga ia menemukan strategi yang tak terkalahkan.

6. Optimasi untuk Real-Time (Inference)

Ini merupakan tahap paling krusial untuk game. Model yang sangat besar akan membuat game "lag".

- Model Quantization: Mengecilkan ukuran model dari format berat ke format ringan agar bisa berjalan di konsol atau HP.
- Inference Engine: Menggunakan library seperti ONNX Runtime atau TensorFlow Lite di dalam Unity/Unreal Engine agar AI bisa merespons dalam hitungan milidetik.

7. Evaluasi & Pengujian (Turing Test for Games)

Bukan hanya soal menang, tapi apakah AI tersebut menyenangkan untuk dilawan?

- Robustness: Apakah AI bingung jika pemain melakukan gerakan aneh?
- Believability: Apakah gerakannya terlihat alami atau seperti robot yang melakukan cheat?

Tabel 8.1 AI Musuh dalam Game Shooter (FPS)

Tahap	Implementasi
State	Posisi pemain terdeteksi, amunisi AI tinggal 10%.
Action	Lari ke tempat perlindungan (Reload).
Reward	+10 jika berhasil reload tanpa tertembak, -50 jika mati saat lari.
Optimasi	Keputusan diambil setiap 0.1 detik (10Hz) agar tidak membebani CPU pemain.

Penting untuk Masa Depan Game

1. NPC yang Lebih Manusiawi: NPC tidak lagi bergerak dalam pola yang sama berulang kali, sehingga game terasa lebih hidup.
2. Adaptabilitas: AI bisa menyesuaikan tingkat kesulitan secara real-time berdasarkan seberapa jago pemainnya (Dynamic Difficulty Adjustment).
3. Strategi Tak Terduga: AI seringkali menemukan taktik baru yang bahkan tidak terpikirkan oleh pengembang game aslinya.

Menghubungkan model Machine Learning ke dalam Game Engine (Unity atau Unreal) biasanya dilakukan dengan dua pendekatan utama: menggunakan Plugin khusus (seperti Unity ML-Agents) atau menggunakan Format Pertukaran Model Standar (seperti ONNX) (Juliani, 2018)

Cara menghubungkan model dari Python/PyTorch ke kedua engine tersebut:

1. Menggunakan Unity ML-Agents (Pendekatan Standar Unity)

Unity ML-Agents merupakan toolkit paling populer yang memungkinkan kita melatih model di Python menggunakan PyTorch dan menjalankannya langsung di Unity.

Langkah-langkah:

1. Instalasi: Instal library mlagents di Python dan package "ML Agents" di Unity Editor.
2. Training (Python): Pembaca menulis skrip training di Python atau menggunakan konfigurasi YAML yang disediakan ML-Agents untuk melatih agent.
3. Export ke .onnx: Setelah training selesai, ML-Agents biasanya akan menyimpan model dalam format .onnx (Open Neural Network Exchange).
4. Import ke Unity: Masukkan file .onnx tersebut ke dalam folder Project Unity.
5. Attach ke Agent: * Pilih objek Agent di Unity.
 - o Masukkan file .onnx tersebut ke dalam slot Model pada komponen Behavior Parameters.

6. Inference: Unity akan menggunakan library Sentis (dahulu Barracuda) untuk menjalankan model tersebut secara real-time di CPU atau GPU pemain.

2. Menggunakan Format ONNX (Untuk Unreal Engine atau Unity Custom)

Jika Pembaca tidak menggunakan toolkit ML-Agents dan ingin menggunakan model PyTorch kustom (misalnya model Computer Vision atau NLP), format ONNX merupakan kuncinya.

Langkah-langkah (PyTorch ke Unreal Engine):

1. Export dari PyTorch:

```
Python
import torch
# Misalkan 'model' merupakan model PyTorch Pembaca
dummy_input = torch.randn(1, input_size)
torch.onnx.export(model, dummy_input, "model_game.onnx")
```

2. Plugin di Unreal Engine: Gunakan plugin Neural Network Engine (NNE) yang sudah bawaan Unreal Engine (versi 5.1 ke atas) atau plugin pihak ketiga seperti ONNX Runtime.
3. Import: Masukkan file .onnx ke dalam Content Browser Unreal.
4. C++ atau Blueprint: * Gunakan API NNE untuk memuat model.
 - Kirim data sensor game (posisi, jarak) sebagai input ke model.
 - Terima hasil prediksi (output) untuk menggerakkan karakter.

3. Menggunakan API/Local Server (Pendekatan Alternatif)

Jika model Pembaca terlalu berat untuk dijalankan di perangkat pengguna (misal: LLM besar seperti GPT), Pembaca bisa menggunakan Flask atau FastAPI di Python.

- Python: Membuat server API sederhana yang menerima data game dan mengembalikan hasil prediksi.
- Game Engine: Menggunakan WebRequest (Unity) atau HTTP Module (Unreal) untuk mengirim data ke server Python dan menerima instruksi gerakan secara real-time.
- Catatan: Cara ini membutuhkan koneksi internet dan memiliki latensi lebih tinggi.

Tabel 8.2 : Ringkasan Perbandingan Fitur

Fitur	Unity ML-Agents	Unreal Engine (NNE/ONNX)
Library Utama	PyTorch (via Python API)	C++ / ONNX Runtime
Format Model	.onnx	.onnx
Kemudahan	Sangat mudah untuk Reinforcement Learning	Lebih teknis (butuh C++ untuk performa maksimal)
Library Runtime	Unity Sentis (Barracuda)	Neural Network Engine (NNE)

Game AI (Artificial Intelligence) dalam konteks pengambilan keputusan real-time merupakan sistem cerdas yang dirancang untuk bertindak sebagai agen di dalam lingkungan game yang dinamis. Berbeda dengan AI tradisional yang mungkin memiliki waktu berpikir lama (seperti AI catur), Real-Time AI harus merespons perubahan lingkungan (input dari pemain atau sistem lain) dalam hitungan milidetik agar tidak mengganggu pengalaman bermain (gameplay) (Millington, 2018)

Pembahasan Tahapan Proses

Membangun AI yang mampu mengambil keputusan secepat kilat melibatkan siklus hidup yang kompleks, mulai dari desain hingga integrasi mesin.

1. Pendefinisian Lingkungan (State & Action Space)

AI melihat dunia game melalui data numerik.

- State Space: Representasi "penglihatan" AI, seperti koordinat X, Y, Z, persentase HP, hingga status lingkungan (siang/malam).
- Action Space: Daftar instruksi yang bisa dieksekusi, baik bersifat diskrit (pilih satu: tembak/lari) maupun kontinu (derajat rotasi belok mobil).

2. Paradigma Learning

- Reinforcement Learning (RL): Cocok untuk strategi yang sulit didefinisikan secara manual. AI bereksperimen sendiri.
- Imitation Learning: Menggunakan dataset dari rekaman pemain manusia agar AI memiliki gaya bermain yang natural dan tidak terkesan "robotik".

3. Simulasi & Efisiensi

Untuk melatih AI dalam jutaan iterasi, pengembang menggunakan:

- Simulasi Paralel: Menjalankan banyak instansi game tanpa grafis (headless) untuk mempercepat pengumpulan data.
- Frame Skipping: Teknik di mana AI hanya berpikir pada setiap frame ke-n. Misalnya, game berjalan pada 60 FPS, tapi AI hanya memproses input setiap 6 frame sekali (10 kali per detik). Ini sangat menghemat beban CPU.

4. Arsitektur & Reward

Model biasanya menggunakan arsitektur Deep Learning seperti PPO (Proximal Policy Optimization). Reward function bertindak sebagai "kompas" moral AI:

- Dense Reward: Poin kecil untuk progres (misal: maju mendekati tujuan).
- Sparse Reward: Poin besar untuk hasil akhir (misal: menang pertandingan).

5. Optimasi (Inference)

Model yang dilatih di server GPU ribuan dolar harus bisa berjalan di HP atau konsol pemain.

- Quantization: Mengubah presisi angka dari float32 ke int8 untuk memperkecil ukuran model tanpa kehilangan akurasi yang signifikan.
- Inference Engine: Library yang mengoptimalkan jalur eksekusi matematika model pada hardware spesifik.

8.2. Menghubungkan AI ke Game Engine (Unity & Unreal)

Menjembatani dunia Python (Training) dan C#/C++ (Game Engine) dilakukan melalui dua jalur utama:

a. Unity ML-Agents

Ini merupakan toolkit paling matang saat ini. Unity menyediakan "jembatan" yang sudah jadi.

1. Latih di Python (PyTorch).
2. Export ke format .onnx.
3. Jalankan di Unity menggunakan Unity Sentis (mesin inference terbaru Unity).

b. Unreal Engine NNE (Neural Network Engine)

Unreal lebih fleksibel namun lebih teknis.

1. Model PyTorch diekspor menggunakan library torch.onnx.
2. Di Unreal, model dimuat melalui Plugin NNE.
3. Pengembang menggunakan C++ untuk mengirimkan data sensor ke model dan mengambil hasil prediksinya untuk menggerakkan aktor di dalam game.

c. API/Local Server (Cloud AI)

Digunakan jika AI sangat kompleks (seperti AI yang bisa diajak bicara/NLP).

- Game mengirim data via HTTP/JSON.
- Server Python (FastAPI) memproses dan mengembalikan jawaban.
- Kelemahan: Membutuhkan internet dan rentan terhadap lag (latensi).

Tabel 8.3 AI Musuh FPS

Komponen	Implementasi
State	Musuh terlihat (True), Jarak < 5m, Peluru < 20%.
Action	Melempar Granat asap + Mundur mencari perlindungan.
Reward	+50 jika selamat dari baku tembak, -100 jika terkena headshot.
Optimasi	Model .onnx dijalankan di GPU mobile menggunakan quantization.

Hasil:

Membangun Game AI bukan sekadar membuat AI yang pintar menembak, tapi membuat AI yang mampu beradaptasi dengan keterbatasan hardware sambil tetap memberikan tantangan yang adil bagi pemain.

Proses mengekspor model dari PyTorch ke format ONNX (Open Neural Network Exchange) sebenarnya cukup sederhana, namun ada satu hal penting yang harus dipahami: ONNX membutuhkan "Dummy Input".

Karena model deep learning bersifat dinamis, ONNX perlu melakukan tracing (pelacakan) dengan cara mengirimkan satu contoh data kosong melalui model tersebut untuk memetakan jalur graf kalkulasinya.

Skrip Python: Export PyTorch ke ONNX

Skrip standar untuk mengeksport model klasifikasi sederhana atau model AI game
Pembaca:

Python

```
import torch
import torch.nn as nn
import torch.onnx

# 1. Definisikan Arsitektur Model (Contoh Simple Neural Network)
class GameAIModel(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(GameAIModel, self).__init__()
        self.network = nn.Sequential(
            nn.Linear(input_size, hidden_size),
            nn.ReLU(),
            nn.Linear(hidden_size, output_size),
            nn.Softmax(dim=1)
        )

    def forward(self, x):
        return self.network(x)

# 2. Inisialisasi Model dan Load Weight (Jika sudah dilatih)
input_dim = 10 # Misal: 10 input sensor (posisi, HP, amunisi, dll)
hidden_dim = 64
output_dim = 4 # Misal: 4 aksi (Diam, Lari, Tembak, Sembunyi)

model = GameAIModel(input_dim, hidden_dim, output_dim)
model.eval() # WAJIB: Set ke mode evaluasi sebelum ekspor

# 3. Buat Dummy Input
# Format: (Batch Size, Input Dimension)
dummy_input = torch.randn(1, input_dim)

# 4. Ekspor ke ONNX
onnx_file_name = "GameAI_Model.onnx"

torch.onnx.export(
    model, # Model yang diekspor
    dummy_input, # Contoh input untuk tracing
    onnx_file_name, # Nama file output
    export_params=True, # Ikut sertakan weight hasil training
    opset_version=12, # Versi ONNX (12-15 biasanya aman untuk Unity/Unreal)
    do_constant_folding=True, # Optimasi: melipat konstanta
    input_names=['input'], # Nama input untuk dibaca di Game Engine
    output_names=['output'], # Nama output untuk dibaca di Game Engine
    dynamic_axes={'input': {0: 'batch_size'}, 'output': {0: 'batch_size'}} #
    Opsional: batch size dinamis
)

print(f"Model berhasil diekspor ke: {onnx_file_name}")
```

Hal Penting yang Perlu Diperhatikan

1. `model.eval()`: Selalu panggil ini sebelum ekspor. Jika tidak, lapisan seperti Dropout atau Batch Normalization akan tetap aktif dan memberikan hasil prediksi yang tidak konsisten di dalam game.
2. Opset Version: Gunakan versi standar seperti 12. Versi yang terlalu baru (seperti 17+) terkadang belum didukung secara penuh oleh library Unity Sentis (Barracuda) atau Unreal NNE.
3. Input/Output Names: Memberikan nama 'input' dan 'output' akan memudahkan Pembaca saat menulis kode C# di Unity atau C++ di Unreal untuk memanggil data hasil prediksi.

Proses Selanjutnya

Setelah Pembaca mendapatkan file .onnx tersebut:

- Di Unity: Seret file ke folder Assets. Gunakan komponen Behavior Parameters pada agen Pembaca.
- Di Unreal: Masukkan ke Content Browser. Gunakan NNE Plugin untuk membuat Model Instance.

Model .onnx di Unity, kita menggunakan Unity Sentis (penerus dari Barracuda). Sentis sangat efisien karena mampu menjalankan inferensi AI langsung di GPU atau CPU perangkat pemain.

Skrip C# sederhana untuk memuat model, memberikan input sensor, dan mengeksekusi aksi berdasarkan output model.

Skrip C#: Menggunakan Sentis untuk Inferensi AI

Simpan skrip ini dengan nama `AgentInference.cs` dan tempelkan ke objek Agent di Unity.

C#

```
using UnityEngine;
using Unity.Sentis;
using System.Collections;

public class AgentInference : MonoBehaviour
{
    public ModelAsset modelAsset; // Masukkan file .onnx di sini via Inspector
    private IWorker m_Worker;
    private Model m_RuntimeModel;

    [Header("Agent Settings")]
    public float speed = 5f;

    void Start()
    {
        // 1. Muat model dari file .onnx
        m_RuntimeModel = ModelLoader.Load(modelAsset);

        // 2. Buat "Worker" (mesin eksekusi).
        // BackendType.GPU untuk performa tinggi, atau CPU untuk kompatibilitas.
    }
}
```

```

        m_Worker = WorkerFactory.CreateWorker(BackendType.GPUCompute,
m_RuntimeModel);
    }
    void Update()
    {
        // 3. Persiapkan Data Input (sesuai dengan 10 input sensor di Python tadi)
        // Misal: posisi x, z, hp, amunisi, dll.
        float[] inputData = new float[] {
            transform.position.x, transform.position.z, 0.8f, 0.2f,
            0f, 0f, 1f, 0.5f, 0.1f, 0.9f
        };

        // Buat Tensor dari array (format: 1 batch, 10 fitur)
        using TensorFloat inputTensor = new TensorFloat(new TensorShape(1, 10),
inputData);

        // 4. Jalankan Inferensi
        m_Worker.Execute(inputTensor);

        // 5. Ambil Hasil Prediksi (Output)
        TensorFloat outputTensor = m_Worker.PeekOutput() as TensorFloat;

        // Memindahkan data dari GPU ke CPU agar bisa dibaca C#
        outputTensor.MakeReadable();
        float[] actions = outputTensor.ToReadOnlyArray();

        // 6. Eksekusi Aksi berdasarkan skor tertinggi (Softmax)
        ExecuteAction(actions);
    }

    void ExecuteAction(float[] actionScores)
    {
        // Cari indeks dengan nilai probabilitas tertinggi
        int bestAction = 0;
        for (int i = 1; i < actionScores.Length; i++)
        {
            if (actionScores[i] > actionScores[bestAction]) bestAction = i;
        }

        // Contoh Logika Pergerakan:
        // 0: Diam, 1: Lari ke Depan, 2: Tembak, 3: Sembunyi
        switch (bestAction)
        {
            case 1:
                transform.Translate(Vector3.forward * speed * Time.deltaTime);
                break;
            case 2:
                Debug.Log("AI Menembak!");
                break;
            // dst...
        }
    }

    private void OnDestroy()
    {
        // Sangat Penting: Hapus worker dari memori saat aplikasi berhenti
        m_Worker?.Dispose();
    }
}

```

Penjelasan Komponen Penting

- **ModelAsset:** Ini merupakan referensi ke file .onnx yang Pembaca impor ke Unity. Pembaca cukup melakukan drag-and-drop file dari Project Window ke slot ini di Inspector.
- **IWorker:** Ini merupakan "otak" pengolahnya. Sentis memungkinkan Pembaca memilih apakah inferensi dilakukan di GPU (sangat cepat untuk model besar) atau CPU (lebih stabil untuk perangkat tua).
- **TensorFloat:** Data di dalam AI tidak diproses sebagai variabel biasa, melainkan sebagai Tensor (array multidimensi). Kita harus membungkus data sensor kita ke dalam TensorFloat sebelum dikirim ke model.
- **Memory Management:** Karena Sentis bekerja langsung dengan memori GPU/Low-level, Pembaca wajib memanggil `.Dispose()` pada worker saat objek hancur untuk mencegah kebocoran memori (memory leak).

Tips Optimasi Real-Time

Gunakan `Coroutine` pada model yang cukup berat, jangan jalankan `m_Worker.Execute()` di setiap frame `Update()`. Gunakan `IEnumerator` untuk menjalankan inferensi setiap 0.1 detik sekali. Ini akan menjaga FPS game tetap stabil (60 FPS) meskipun AI sedang "berpikir".

Membangun sistem sensor otomatis merupakan langkah kunci agar AI Pembaca bisa "melihat" dunianya secara dinamis. Kita akan menggunakan teknik Raycasting—menembakkan sinar laser virtual ke berbagai arah—untuk mengumpulkan data jarak ke rintangan.

Data ini kemudian akan dikonversi menjadi float array yang siap dikirim ke model .onnx

Skrip C#: Sensor Raycast Otomatis

Skrip ini akan menembakkan sinar dalam pola kipas (fan-shape) di depan agen.

C#

```
using UnityEngine;

public class AISensor : MonoBehaviour
{
    [Header("Sensor Settings")]
    public int rayCount = 7;           // Jumlah laser (makin banyak makin detail)
    public float sensorAngle = 90f;    // Total sudut pandang (derajat)
    public float maxRange = 10f;       // Jarak maksimum laser
    public LayerMask detectionLayer;    // Layer apa yang dianggap rintangan (misal: "Obstacle")

    // Fungsi ini akan dipanggil oleh skrip AgentInference Pembaca
    public float[] GetSensorData()
    {
        float[] sensorReadings = new float[rayCount];
```

```

float angleStep = sensorAngle / (rayCount - 1);
float startAngle = -sensorAngle / 2;

for (int i = 0; i < rayCount; i++)
{
    // Menghitung arah laser berdasarkan rotasi agen
    float currentAngle = startAngle + (i * angleStep);
    Vector3 direction = Quaternion.Euler(0, currentAngle, 0) *
transform.forward;

    RaycastHit hit;
    if (Physics.Raycast(transform.position + Vector3.up * 0.5f, direction,
out hit, maxRange, detectionLayer))
    {
        // Normalisasi jarak: 0 (sangat dekat) hingga 1 (sangat jauh/tidak
kena)
        sensorReadings[i] = hit.distance / maxRange;

        // Visualisasi laser di Editor (Merah jika kena)
        Debug.DrawRay(transform.position + Vector3.up * 0.5f, direction *
hit.distance, Color.red);
    }
    else
    {
        sensorReadings[i] = 1f; // Tidak ada rintangan

        // Visualisasi laser di Editor (Hijau jika aman)
        Debug.DrawRay(transform.position + Vector3.up * 0.5f, direction *
maxRange, Color.green);
    }
}
return sensorReadings;
}
}

```

Integrasi dengan Model AI (Sentis)

Sekarang, kita hubungkan AISensor ke skrip AgentInference yang kita buat sebelumnya. Ini akan menggantikan data hardcoded.

C#

```

// Di dalam Update() atau Coroutine pada AgentInference.cs
void Update()
{
    // 1. Ambil data dari sensor otomatis
    AISensor sensor = GetComponent<AISensor>();
    float[] sensorData = sensor.GetSensorData();

    // 2. Tambahkan data status internal (misal: HP dan Amunisi)
    // Jika model butuh 10 input, dan sensor ada 7, kita tambah 3 data lagi
    float[] fullInput = new float[10];
    for(int i = 0; i < 7; i++) fullInput[i] = sensorData[i];
    fullInput[7] = currentHP / maxHP;
    fullInput[8] = currentAmmo / maxAmmo;
    fullInput[9] = isEnemyInSight ? 1f : 0f;

    // 3. Kirim ke Model Sentis
    using TensorFloat inputTensor = new TensorFloat(new TensorShape(1, 10),
fullInput);
    m_Worker.Execute(inputTensor);

    // ... sisa kode inferensi sama seperti sebelumnya
}

```

8.3. Menggunakan Normalisasi (0 ke 1)

Neural Network bekerja paling baik ketika semua input berada dalam rentang yang serupa (biasanya 0 hingga 1 atau -1 hingga 1).

- Jika Pembaca memasukkan angka jarak mentah seperti 50.0 meter dan angka HP 0.5, model akan cenderung mengabaikan HP karena angkanya terlalu kecil dibanding jarak.
- Dengan membagi jarak dengan maxRange, kita memastikan semua input memiliki "bobot" yang adil di mata AI.

Tips Tambahan: Performa

Jika memiliki banyak agen (misal 50 musuh), menembakkan ribuan Raycast setiap frame bisa membebani CPU.

1. LayerMask: Pastikan detectionLayer hanya mencakup dinding/rintangan. Jangan biarkan laser mengecek setiap objek kecil di game.
2. Time Slicing: Jalankan GetSensorData() hanya beberapa kali per detik, bukan setiap frame.

Menggabungkan Behavior Tree (BT) dengan Neural Network (Model AI) merupakan pendekatan "Hybrid AI" yang sangat populer. Behavior Tree berfungsi sebagai pembuat keputusan strategis (High-Level), sementara Model AI menangani eksekusi taktis yang kompleks (Low-Level).

Rancangan Behavior Tree sederhana untuk transisi dari mode Patroli ke mode Tempur (Model AI).

8.4 Struktur Logika Behavior Tree

1. Selector (Root): Mencoba menjalankan anak pohon dari kiri ke kanan. Jika satu berhasil, ia berhenti.
2. Sequence (Combat Mode):
 - Condition: Apakah musuh terdeteksi (via Sensor)?
 - Action: Jalankan Model AI Inference (menggunakan skrip Sentis sebelumnya).
3. Action (Patrol Mode): Jika kondisi tempur tidak terpenuhi, jalankan logika Patroli Manual (pindah antar Waypoint).

Skrip C#: Implementasi Hybrid AI

Skrip ini mengatur kapan sistem menggunakan logika Patrol() yang sederhana dan kapan mengaktifkan NeuralNetworkInference().

C#

```
using UnityEngine;
using Unity.Sentis;

public class HybridAIController : MonoBehaviour
{
    public enum AIState { Patrolling, Combat }
    public AIState currentState = AIState.Patrolling;

    [Header("Inference Settings")]
    public ModelAsset modelAsset;
    private IWorker worker;

    [Header("Patrol Settings")]
    public Transform[] waypoints;
    private int currentWaypointIndex = 0;

    void Start()
    {
        worker = WorkerFactory.CreateWorker(BackendType.GPUCompute,
        ModelLoader.Load(modelAsset));
    }

    void Update()
    {
        // --- LOGIKA BEHAVIOR TREE SEDERHANA ---

        if (IsEnemyDetected())
        {
            currentState = AIState.Combat;
            ExecuteCombatInference(); // Menggunakan Model AI (.onnx)
        }
        else
        {
            currentState = AIState.Patrolling;
            ExecutePatrol(); // Menggunakan Logika Manual (C#)
        }
    }

    bool IsEnemyDetected()
    {
        // Gunakan Raycast atau trigger area untuk deteksi
        return Vector3.Distance(transform.position, Player.Instance.position) <
10f;
    }

    void ExecutePatrol()
    {
        // Logika sederhana: Bergerak dari titik A ke titik B
        Transform target = waypoints[currentWaypointIndex];
        transform.position = Vector3.MoveTowards(transform.position,
target.position, 2f * Time.deltaTime);

        if (Vector3.Distance(transform.position, target.position) < 0.5f)
        {
            currentWaypointIndex = (currentWaypointIndex + 1) % waypoints.Length;
        }
    }

    void ExecuteCombatInference()
    {
        // Ambil data dari sensor yang kita buat sebelumnya
        float[] sensorData = GetComponent<AISensor>().GetSensorData();

        // Jalankan inferensi menggunakan Sentis (seperti di skrip sebelumnya)
        using TensorFloat inputTensor = new TensorFloat(new TensorShape(1, 10),
sensorData);
    }
}
```

```

        worker.Execute(inputTensor);

        // ... ambil output dan gerakkan karakter ...
    }

    private void OnDestroy() => worker?.Dispose();
}

```

8.5 Menggunakan Pendekatan Hybrid

Fitur	Mengapa Tidak Full Model AI?	Mengapa Tidak Full Manual (BT)?
Efisiensi	Melatih AI untuk sekadar "jalan santai" (patroli) sangat membuang waktu dan resource.	Membuat logika tempur yang adaptif di dalam BT sangat rumit dan kaku.
Kontrol	Desainer game butuh rute patroli yang pasti (bisa diprediksi).	Desainer ingin musuh yang terasa "pintar" dan tidak mudah ditebak saat bertarung.
Debugging	Sulit mendebug AI jika ia "tersangkut" saat patroli tanpa alasan jelas.	BT sangat mudah dibaca secara visual untuk melihat status AI saat ini.

Menggunakan Node Canvas atau Behavior Designer

Jika Pembaca menggunakan plugin visual seperti Behavior Designer atau NodeCanvas di Unity, Pembaca cukup membuat sebuah Action Node khusus bernama `NeuralNetworkAction`.

Di dalam node tersebut, Pembaca panggil fungsi `worker.Execute()`. Ini akan membuat struktur proyek Pembaca jauh lebih rapi daripada menggunakan skrip if-else yang panjang.

Membuat Visual Debugger sangat penting karena AI sering kali terasa seperti "kotak hitam". Dengan GUI di layar, Pembaca bisa melihat apakah AI sedang ragu-ragu (skor antar aksi mirip) atau sangat percaya diri.

Kita akan menggunakan metode `OnGUI` dari Unity untuk membuat overlay teks yang ringan dan informatif tanpa perlu mengatur sistem UI Canvas yang rumit.

Skrip C#: AI Visual Debugger

Tambahkan skrip ini ke objek yang sama dengan Hybrid AI Controller.

C#

```

using UnityEngine;

public class AIDebugger : MonoBehaviour
{
    private HybridAIController aiController;
    private string[] actionNames = { "Diam", "Lari", "Tembak", "Sembunyi" };
    private float[] currentScores;

    void Start()
    {

```

```

{
    aiController = GetComponent<HybridAIController>();
}

// Fungsi ini dipanggil oleh HybridAIController setelah inferensi selesai
public void UpdateScores(float[] scores)
{
    currentScores = scores;
}

void OnGUI()
{
    // 1. Pengaturan Area Debug
    GUI.Box(new Rect(10, 10, 250, 180), "AI Mind Monitor");

    // 2. Tampilkan State Saat Ini
    GUI.Label(new Rect(20, 40, 200, 20), "Status: " +
aiController.currentState.ToString());

    if (currentScores != null && currentScores.Length > 0)
    {
        GUI.Label(new Rect(20, 65, 200, 20), "Action Probabilities:");

        for (int i = 0; i < currentScores.Length; i++)
        {
            // Menghitung panjang bar visual berdasarkan skor (0.0 - 1.0)
            float barWidth = currentScores[i] * 100f;

            // Menampilkan nama aksi dan persentase
            GUI.Label(new Rect(20, 90 + (i * 25), 100, 20), actionNames[i]);

            // Menggambar Bar Visual
            GUI.color = GetColorByScore(currentScores[i]);
            GUI.HorizontalSlider(new Rect(100, 95 + (i * 25), 100, 20),
currentScores[i], 0f, 1f);
            GUI.color = Color.white;

            GUI.Label(new Rect(210, 90 + (i * 25), 40, 20), (currentScores[i] *
100).ToString("F0") + "%");
        }
    }

    Color GetColorByScore(float score)
    {
        if (score > 0.7f) return Color.green; // AI sangat yakin
        if (score > 0.3f) return Color.yellow; // AI ragu-ragu
        return Color.red; // AI tidak memilih ini
    }
}

```

Cara Menghubungkannya

Di dalam skrip HybridAIController Pembaca (pada bagian yang kita buat sebelumnya), tambahkan satu baris untuk mengirim skor ke debugger:

C#

```

void ExecuteCombatInference()
{
    // ... proses inferensi Sentis ...

    float[] actions = outputTensor.ToReadOnlyArray();

    // Kirim data ke debugger
    GetComponent<AIDebugger>().UpdateScores(actions);
}

```

```
// ... eksekusi aksi ...  
}
```

8.6 Analisis Data Debugger

Dengan alat ini bisa mendiagnosis masalah umum pada Machine Learning di game:

1. Stuck / Oscillation: Jika bar "Lari" dan "Sembunyi" bergantian menjadi yang tertinggi secara cepat (50/50), AI akan terlihat bergetar atau bingung. Ini tandanya Pembaca butuh Frame Skipping atau penyesuaian pada Reward Function.
2. Over-Confidence: Jika satu aksi selalu 99% meskipun lingkungannya berubah, kemungkinan model Pembaca mengalami overfitting saat pelatihan.
3. Sensor Failure: Jika visual laser (Raycast) menunjukkan ada dinding, tapi bar "Lari" tetap tinggi, berarti ada masalah pada normalisasi data input sensor Pembaca.

Hasil Alur AI end to end

Selamat! Pembaca sekarang memiliki sistem AI end-to-end:

1. Sensor: AI Sensor.cs (Mata).
2. Brain: Model .onnx & Sentis (Logika).
3. Decision Maker: HybridAIController.cs (Strategi).
4. Debugger: AIDebugger.cs (Transparansi).

Reward Function merupakan seni menyeimbangkan antara apa yang di inginkan (tujuan) dan apa yang sebenarnya dipelajari oleh AI. Seringkali AI menabrak tembok karena ia merasa "keuntungan" dari bergerak maju lebih besar daripada "hukuman" menabrak.

Panduan strategi untuk memperbaiki perilaku AI yang aneh:

1. Identifikasi "Reward Hacking"

AI merupakan "pemalas yang cerdas". Jika Pembaca memberinya reward untuk "kecepatan", ia mungkin akan menabrak tembok secepat mungkin jika itu mengakhiri episode dan memberinya skor instan.

- Masalah: AI menabrak tembok terus-menerus.
- Penyebab: Hukuman (penalty) menabrak terlalu kecil, atau reward bergerak maju terlalu besar.

2. Strategi Pemberian Reward yang Seimbang

Gunakan kombinasi antara Reward Positif (Wortel) dan Reward Negatif (Cambuk):

Tabel 8.4 Kombinasi antara wortel dengan cambuk

Aksi / Kejadian	Reward	Alasan
Mencapai Target	+10.0	Hadiah utama (Sparse Reward).
Menabrak Tembok	-1.0	Hukuman keras agar AI takut pada rintangan.
Setiap Langkah	-0.01	"Time Penalty" agar AI tidak diam saja atau berputar-putar.
Mendekati Target	+0.1	"Progress Reward" agar AI tahu dia di jalur yang benar.

3. Teknik "Curriculum Learning"

Jika AI masih menabrak tembok, mungkin tugasnya terlalu sulit. Jangan langsung menaruhnya di labirin rumit.

1. Tahap 1: Latih di ruangan kosong dengan satu target.
2. Tahap 2: Tambahkan satu tembok besar di tengah.
3. Tahap 3: Tambahkan banyak tembok dinamis.

4. Memperbaiki Sensor dalam Reward

Seringkali AI menabrak karena ia tidak "merasa" bahaya sebelum terlambat. Pembaca bisa memodifikasi reward berdasarkan jarak sensor:

Python

```
# Contoh logika di Python/PyTorch Training
def calculate_reward(sensors, collided):
    reward = 0

    # 1. Penalti Jarak (Makin dekat tembok, makin kena penalti kecil)
    min_dist = min(sensors) # Sensor terpendek
    if min_dist < 0.2:
        reward -= 0.2 # "Peringatan" sebelum menabrak

    # 2. Penalti Tabrakan
    if collided:
        return -1.0 # Tamat

    # 3. Reward Progres
    reward += 0.1 # Tetap hidup dan bergerak

    return reward
```

5. Tips Troubleshooting Perilaku Aneh

- AI Berputar-putar di Tempat?
 - Penyebab: Reward progres terlalu tinggi sehingga ia takut bergerak jauh dan lebih memilih mencari poin kecil dengan berputar.

- Solusi: Tingkatkan Time Penalty (\$-0.01\$ tiap frame).
- AI Diam Saja (Bunuh Diri)?
- Penyebab: Semua langkah dianggap buruk (negatif), jadi ia memilih menabrak tembok agar episode berakhir cepat.
- Solusi: Berikan reward keberadaan (existence reward) yang positif selama ia tidak menabrak.
- Gerakan Terpatah-patah?
- Penyebab: Perubahan aksi terlalu drastis dihargai sama dengan gerakan mulus.
- Solusi: Berikan penalti kecil jika ada perubahan nilai output aksi yang terlalu besar antar frame.

Hasil Proses Iteratif

Finetuning merupakan proses iteratif. Gunakan AI Debugger yang kita buat sebelumnya untuk melihat skor mana yang mendominasi saat ia menabrak. Jika skor "Lari" tetap tinggi saat sensor jarak menunjukkan angka kecil (merah), maka Pembaca harus menaikkan bobot hukuman sensor tersebut di skrip training Pembaca.

TensorBoard merupakan alat "detektif" terbaik bagi pengembang AI. Tanpa visualisasi, Pembaca hanya melihat angka-angka yang lewat dengan cepat. Dengan TensorBoard, Pembaca bisa melihat tren jangka panjang: apakah AI Pembaca semakin pintar, atau justru sedang "stres" karena reward yang membingungkan.

Panduan untuk mengintegrasikannya ke dalam skrip Python (PyTorch) Pembaca.

1. Instalasi dan Setup

Pastikan Pembaca sudah menginstal library yang diperlukan:

Bash

```
pip install tensorboard
```

2. Implementasi dalam Kode Training

Pembaca perlu menggunakan SummaryWriter dari PyTorch untuk mencatat data reward setiap episode.

Python

```
from torch.utils.tensorboard import SummaryWriter
import numpy as np

# Inisialisasi Writer (akan membuat folder 'runs')
writer = SummaryWriter('runs/AI_Game_Experiment_1')

def train_loop(total_episodes):
    for episode in range(total_episodes):
        # ... logika training ...
        total_reward = 0
        steps_taken = 0
        collision_count = 0
```

```
# Misal setelah satu episode selesai:
episode_reward = np.random.random() # Ganti dengan total reward asli
loss_value = np.random.random() # Ganti dengan loss asli dari model

# 3. KIRIM DATA KE TENSORBOARD
writer.add_scalar('Performance/Reward', episode_reward, episode)
writer.add_scalar('Loss/Training_Loss', loss_value, episode)

# Tambahkan metrik kustom untuk melihat "pemahaman" rintangan
writer.add_scalar('Metrics/Steps_Before_Collision', steps_taken, episode)

writer.close()
```

3. Cara Membaca Grafik "Pemahaman" AI

Saat Pembaca menjalankan TensorBoard di terminal dengan perintah: `tensorboard --logdir=runs`

Perhatikan pola grafik berikut untuk mendiagnosis perilaku AI:

A. Grafik Reward Naik (Positif)

- Indikasi: AI mulai memahami bahwa mencapai target memberikan skor besar.
- Pola: Garis menaik secara konsisten dan mulai mendatar (plateau) di puncak.

B. Grafik Collision Rate Turun

- Indikasi: Inilah momen AI mulai "paham" rintangan.
- Pola: Jika grafik Steps Before Collision meningkat tajam, berarti AI sudah belajar menggunakan sensor Raycast-nya untuk menghindari.

C. Loss yang Tidak Stabil (Spiking)

- Indikasi: Perubahan reward terlalu drastis (misal: hukuman tabrakan terlalu ekstrem).
- Pola: Grafik loss melompat naik-turun secara liar. Ini tandanya Pembaca perlu mengecilkan learning rate atau memperhalus reward function.

4. Mengapa Ini Penting untuk Game AI?

Dalam pengembangan game, TensorBoard membantu Pembaca menjawab pertanyaan kritis tanpa harus menjalankan Unity berkali-kali:

1. Apakah Model Overfit? Jika reward di training sangat tinggi tapi di game (test) sangat rendah, model hanya menghafal jalan, bukan belajar navigasi.
2. Kapan Harus Berhenti? Pembaca bisa melihat kapan model berhenti berkembang. Tidak ada gunanya melatih model selama 10 jam jika grafiknya sudah mendatar di jam ke-2.

Ringkasan Analisis Visual

- Reward naik + Loss turun: AI belajar dengan sempurna.
- Reward stagnan + Loss turun: AI menemukan "celah" (reward hacking) dan berhenti mengeksplorasi strategi baru.
- Reward turun + Loss naik: Sistem reward Pembaca berlawanan dengan tujuan utama (AI bingung).

Untuk membuktikan bahwa AI Pembaca bukan sekadar "menghafal" rute, tetapi benar-benar "memahami" sensornya, Pembaca perlu mengujinya dalam skenario dinamis. Skenario ini akan memaksa AI untuk bereaksi terhadap perubahan lingkungan secara real-time.

Rumusan Test Case yang sistematis untuk diimplementasikan di Unity:

1. Skenario "The Sliding Wall" (Rintangan Horisontal)

Tes ini bertujuan menguji apakah AI bisa memprediksi jalur yang aman saat celah terus berpindah.

- Setup: Buat sebuah tembok panjang dengan satu celah sempit di tengah. Gunakan skrip sederhana (atau Animation) untuk menggerakkan tembok tersebut ke kiri dan ke kanan secara berulang (ping-pong).
- Target AI: AI harus menyeberang ke sisi lain tembok.
- Indikator Keberhasilan: AI berhenti sejenak atau berbelok mengikuti pergerakan celah tanpa menyentuh tembok.

2. Skenario "The Closing Gate" (Rintangan Vertikal)

Tes ini menguji kemampuan AI dalam mengambil keputusan cepat: "Lari sekarang atau tunggu?".

- Setup: Pasang dua pilar yang bergerak mendekat (menutup) dan menjauh (membuka) seperti pintu gerbang otomatis.
- Target AI: Mencapai target di balik gerbang.
- Indikator Keberhasilan: AI meningkatkan kecepatan saat gerbang terbuka atau berhenti total saat gerbang mulai menutup (menghindari terjepit).

3. Skenario "The Random Pop-up" (Muncul Tiba-tiba)

Ini merupakan ujian refleks sensorik paling ekstrem.

- Setup: Buat area terbuka. Gunakan skrip untuk memunculkan objek (Instantiate) secara acak tepat di lintasan yang sedang dilewati AI.
- Target AI: Navigasi dari titik A ke B tanpa rute tetap.
- Indikator Keberhasilan: AI melakukan manuver menghindar (belok tajam) segera setelah Raycast mendeteksi objek baru dalam jarak dekat (< 2 meter).

8.7 Matriks Evaluasi Pengujian

Pembaca bisa menggunakan tabel berikut untuk mencatat hasil dari Visual Debugger dan TensorBoard selama pengujian:

Tabel 8.6 : Visual Debugger dan TensorBoard

Test Case	Metrik Utama	Perilaku Gagal (Red Flag)	Perilaku Sukses
Sliding Wall	Collision Rate	AI menabrak pinggiran celah.	Skor "Lari" tetap tinggi sambil mengikuti celah.
Closing Gate	Wait Time vs Safety	AI memaksakan diri masuk saat gerbang hampir tutup.	AI melambat/diam saat sensor depan buntu.
Random Pop-up	Reaction Latency	AI terlambat belok (Frame Skipping terlalu tinggi).	Manuver menghindari terjadi dalam ≤ 0.2 detik.

Skrip C#: ObstaclePingPong.cs

Tempelkan skrip ini pada objek tembok atau rintangan yang ingin Pembaca gerakan.

C#

```
using UnityEngine;
public class ObstaclePingPong : MonoBehaviour
{
    [Header("Movement Settings")]
    public Vector3 moveAxis = Vector3.right; // Arah gerakan (Default: Kanan-Kiri)
    public float distance = 5f;              // Seberapa jauh tembok bergerak
    public float speed = 2f;                 // Kecepatan gerakan
    private Vector3 startPosition;
    void Start()
    {
        // Simpan posisi awal agar gerakan tetap konsisten pada porosnya
        startPosition = transform.position;
    }
    void Update()
    {
        // Menghitung nilai Ping-Pong berdasarkan waktu dan kecepatan
        float movement = Mathf.PingPong(Time.time * speed, distance);
        // Update posisi objek
        transform.position = startPosition + (moveAxis.normalized * movement);
    }
}
```

Tips Implementasi di Unity

Untuk memudahkan pengujian ini, saya sarankan Pembaca membuat Test Runner sederhana:

1. Automated Reset: Jika AI menabrak atau mencapai target, gunakan `Transform.SetPositionAndRotation` untuk mengembalikan AI ke titik awal secara otomatis.

2. Success Counter: Buat UI teks kecil di pojok layar: Success: X / Fail: Y.
3. Speed Multiplier: Selama pengujian, Pembaca bisa meningkatkan Time.timeScale (misal ke 5x) agar proses pengumpulan data ribuan percobaan berjalan lebih cepat.

Jika AI Pembaca terus menabrak tembok yang bergerak lambat, jangan langsung menyalahkan otaknya. Cek kembali Raycast Pembaca—terkadang laser tersebut terlalu tipis sehingga "melewatkan" sudut tembok yang bergerak cepat!

Skrip ini sangat berguna untuk menguji apakah sensor AI Pembaca mampu mendeteksi perubahan lingkungan yang dinamis secara real-time. Kita akan menggunakan fungsi `Mathf.PingPong` yang sudah disediakan oleh Unity untuk menciptakan gerakan bolak-balik yang mulus.

Cara Menyiapkan Test Case Pertama

1. Buat Objek: Buat sebuah Cube di Unity, beri nama "SlidingWall". Atur skalanya agar terlihat seperti tembok tipis namun panjang.
2. Pasang Skrip: Drag and drop skrip `ObstaclePingPong` ke objek tersebut.
3. Konfigurasi:
 - Jika Pembaca ingin bergerak maju-mundur, ubah Move Axis menjadi (0, 0, 1).
 - Jika ingin bergerak kanan-kiri, biarkan di (1, 0, 0).
4. Beri Layer: Jangan lupa pastikan tembok ini berada di Layer yang sama dengan yang dideteksi oleh `AI_Sensor Pembaca` (misalnya layer "Obstacle").

Harus Pembaca Perhatikan Saat Pengujian

Saat Pembaca menekan tombol Play, perhatikan Visual Debugger dan Raycast (laser) Pembaca:

- Deteksi Dinamis: Apakah garis laser berubah warna (menjadi merah) saat tembok bergerak masuk ke jalur sensor? Jika laser tidak berubah warna tepat waktu, AI tidak akan punya waktu untuk menghindari.
- Adaptasi Output: Perhatikan grafik probabilitas di layar. Saat tembok mendekat, skor aksi "Lari" seharusnya turun dan skor "Diam" atau "Sembunyi" seharusnya naik secara otomatis.

Tips: Cobalah untuk mengubah nilai speed pada tembok secara bertahap. AI yang baik biasanya lulus tes pada kecepatan 2f, namun mulai kesulitan pada kecepatan 10f karena batasan reaction time (latensi) dari model.

Integrasi Akhir

Sekarang Pembaca sudah memiliki ekosistem lengkap:

1. AI Brain (Sentis + ONNX).
2. AI Eyes (Raycast Sensor).
3. AI Body (Hybrid Behavior Tree).
4. AI Monitor (Visual Debugger).
5. Environment (Dynamic Obstacle)

BAB IX

EVALUASI REINFORCEMENT LEARNING

Evaluasi ini dirancang untuk menguji pemahaman Pembaca terhadap konsep dasar Machine Learning, matematika pendukung, siklus hidup model (lifecycle), hingga perancangan algoritma tingkat lanjut seperti Deep Q-Network (DQN) dan Hierarchical Reinforcement Learning (HRL) yang telah dibahas pada bab-bab sebelumnya.

BAGIAN A: PILIHAN GANDA

(Pilihlah satu jawaban yang paling tepat)

Soal 1 (Bab I: Fondasi Machine Learning)

Menurut definisi dari Tom Mitchell (1997), sebuah program komputer dikatakan belajar dari pengalaman (E) terkait dengan tugas (T) dan ukuran kinerja (P), jika kinerjanya pada tugas T meningkat seiring bertambahnya pengalaman E. Jika Pembaca sedang membangun sistem klasifikasi email spam, manakah identifikasi komponen yang benar?

- A. T: Algoritma Naive Bayes; P: Jumlah data email; E: Akurasi model.
- B. T: Memprediksi harga rumah; P: Root Mean Squared Error (RMSE); E: Riwayat data properti.
- C. T: Mengklasifikasikan email sebagai spam/bukan spam; P: Persentase email spam yang terdeteksi dengan benar; E: Sekumpulan data email yang sudah diberi label oleh pengguna.
- D. T: Menghitung rumus BMI konvensional; P: Akurasi fungsi if-else; E: Input berat badan pengguna.

Soal 2 (Bab II: Teori Pendukung & Preprocessing)

Dalam persiapan data statistik, proses standardisasi sering kali bertujuan mengubah distribusi data menjadi Normal (Gaussian) Distribution. Manakah dari pernyataan berikut yang menggambarkan karakteristik dari distribusi tersebut?

- A. Hanya memiliki dua kemungkinan hasil diskret (0 atau 1).
- B. Memiliki bentuk kurva seperti lonceng yang simetris di sekitar nilai rata-ratanya (mean).
- C. Semua nilai dalam rentang tertentu memiliki peluang yang sama besar untuk terjadi.
- D. Data memiliki kecondongan (skewness) ekstrem ke arah kanan.

Soal 3 (Bab V: Robot Navigation & Reinforcement Learning)

Pada kerangka kerja Markov Decision Process (MDP) untuk navigasi robot, agen belajar melalui interaksi dengan lingkungan melalui komponen State, Action, Reward, dan Policy. Apakah yang dimaksud dengan Policy (π)?

- A. Umpan balik numerik segera setelah robot mengambil suatu tindakan.
- B. Struktur sensor Lidar yang digunakan untuk mengukur jarak rintangan.
- C. Strategi atau fungsi otak yang memetakan kondisi lingkungan (State) menjadi keputusan tindakan (Action) yang optimal.
- D. Total akumulasi hadiah yang didapatkan robot ketika berhasil mencapai target.

Soal 4 (Bab V: Deep Q-Network / DQN)

Google DeepMind memperkenalkan dua komponen rahasia untuk mengatasi ketidakstabilan Neural Network saat diintegrasikan dalam Deep Q-Network (DQN), yaitu Target Network dan Experience Replay. Fungsi utama dari Experience Replay adalah...

- A. Memperbarui bobot jaringan secara instan di setiap frame gerakan robot.
- B. Menyimpan pengalaman masa lalu ke dalam memori dan mengambil sampel acak saat pelatihan untuk mencegah AI dari masalah "lupa" (catastrophic forgetting).
- C. Berperan sebagai kompas statis yang tidak pernah diperbarui nilainya sepanjang simulasi.
- D. Mengubah tipe action space dari kontinu menjadi diskret secara otomatis.

Soal 5 (Bab VI: Hierarchical Reinforcement Learning / HRL)

Dalam Options Framework pada Hierarchical Reinforcement Learning (HRL), pembuatan keputusan dibagi menjadi High-Level Policy (Meta-Controller) dan Low-Level Policy (Controller). Tugas utama dari Meta-Controller adalah...

- A. Mengeksekusi gerakan motorik halus robot per milidetik.
- B. Memilih keterampilan tingkat tinggi (Option / sub-tujuan) yang harus dijalankan oleh agen selama periode waktu tertentu.
- C. Menghitung jarak Euclidean terhadap koordinat dinding pembatas labirin.
- D. Memberikan hadiah internal (Intrinsic Reward) secara terus-menerus kepada sistem.

BAGIAN B: ESAI KONSEPTUAL (Jawablah dengan ringkas dan analitis)

1. **Analisis Masalah Pemrograman Konvensional vs ML:** Mengapa pendekatan pemrograman konvensional berbasis aturan eksplisit (seperti penulisan fungsi if-else pada rumus BMI) tidak efektif jika diterapkan pada kasus pengenalan digit tulisan tangan (MNIST dataset)? Jelaskan tantangannya berdasarkan variasi data!
2. **Reward Design pada Reinforcement Learning:** Dalam melatih agen AI untuk manajemen stok barang cepat basi (fitur FEFO), rekayasa reward sangat krusial. Apa perbedaan dampak perilaku agen jika kita menggunakan Sparse Reward dibandingkan dengan Dense Reward?
3. **Masalah Troubleshooting Perilaku AI:** Apabila dalam proses pengujian simulasi Game AI, agen Pembaca menunjukkan perilaku aneh berupa "berputar-putar di tempat tanpa menuju sasaran", identifikasi apa kemungkinan kesalahan pada Reward Function dan bagaimana solusi mengatasinya?

BAGIAN C: ANALISIS KODE PYTHON (Studi Kasus)

Perhatikan cuplikan kode arsitektur jaringan berikut yang ditulis menggunakan kerangka kerja PyTorch:

#Python

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class InventoryDQN(nn.Module):
    def __init__(self, state_size, action_size):
        super(InventoryDQN, self).__init__()
        self.fc1 = nn.Linear(state_size, 64)
        self.fc2 = nn.Linear(64, 32)
        self.fc3 = nn.Linear(32, action_size)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        return self.fc3(x)
```

(Sumber arsitektur: Model Dasar Kontinu Deep Q-Network untuk Optimasi Gudang)

Pertanyaan:

1. Jika model di atas diinisialisasi dengan `state_size=5` (merekpresentasikan variabel: Level Stok, Rata-rata Penjualan, Sisa Hari Kedaluwarsa, Harga, dan Hari Promo) dan `action_size=3` (Aksi 0: Diam, Aksi 1: Pesan Stok, Aksi 2: Beri Diskon), jelaskan bagaimana aliran data input diproses hingga menghasilkan keputusan akhir!
2. Mengapa lapisan output (`self.fc3`) tidak menggunakan fungsi aktivasi non-linear seperti ReLU? Jelaskan alasan teoritisnya terkait estimasi nilai Q!

KUNCI JAWABAN DAN PEMBAHASAN

Pembahasan Bagian A (Pilihan Ganda)

1. **Jawaban: C.** Pembahasan: Komponen utama Mitchell adalah T (Tugas), P (Ukuran Kinerja), dan E (Pengalaman). Untuk klasifikasi spam, tugasnya (T) adalah mengklasifikasikan email ; pengukurannya (P) adalah tingkat keberhasilan tebakan atau akurasi deteksi ; dan pengalamannya (E) adalah histori data email yang dipelajari. Pilihan B salah karena itu merupakan contoh tugas Regresi, bukan Klasifikasi Spam.
2. **Jawaban: B.** Pembahasan: Distribusi Normal atau Gaussian dicirikan oleh bentuk kurva simetris menyerupai lonceng di mana nilai rata-rata, median, dan modus berada di titik tengah yang sama. Pilihan A adalah distribusi Bernoulli , pilihan C adalah distribusi Uniform.
3. **Jawaban: C.** Pembahasan: Policy (π) adalah komponen utama penentu kecerdasan agen yang bertindak sebagai fungsi pemetaan atau strategi pengambilan keputusan; jika State adalah situasinya, maka Policy memberikan instruksi aksi mana yang harus diambil untuk memaksimalkan reward.
4. **Jawaban: B.** Pembahasan: Experience Replay memutus korelasi antar data berurutan dengan cara menyimpan pengalaman ke dalam memori (deque) lalu mengambilnya secara acak untuk sesi training batch. Hal ini menjaga stabilitas pelatihan dan mencegah hilangnya ingatan AI terhadap rintangan lama.
5. **Jawaban: B.** Pembahasan: Dalam arsitektur hierarki (HRL), Meta-Controller bertanggung jawab atas perencanaan jangka panjang dengan memilih Option atau keterampilan makro. Sedangkan Controller tingkat rendah mengeksekusi aksi atomik/dasar sampai sub-tujuan dari opsi tersebut terpenuhi.

Pembahasan Bagian B (Esai Konseptual)

1. **Sebab Pemrograman Konvensional Tidak Efektif pada Gambar MNIST:**
Pada pemrograman konvensional, aturan harus ditulis secara eksplisit dan kaku. Pada gambar berukuran 100 x 100 piksel, terdapat 10.000 dimensi input piksel. Variasi penulisan tangan manusia untuk satu angka saja (misalnya angka 0 atau 7) memiliki variabilitas geometri, ketebalan, kemiringan, dan gaya yang tidak terbatas. Menulis kode aturan if-else secara manual untuk mencakup jutaan kombinasi piksel tersebut mustahil dilakukan. Machine Learning menyelesaikan ini dengan cara membiarkan algoritma generik mencari dan mengekstrak polanya sendiri secara mandiri melalui pemrosesan data latih.
1. **Perbedaan Sparse vs Dense Reward pada Manajemen Stok:**
 - *Sparse Reward* hanya memberikan imbalan (misal +100) di akhir siklus jika barang berhasil terselamatkan dan habis terjual tepat waktu. Dampaknya, agen akan sangat lambat belajar karena sebagian besar tindakan acak di awal tidak menghasilkan umpan balik, sehingga agen kesulitan mengaitkan tindakan mana yang benar-benar membawa keuntungan.
 - *Dense Reward* memberikan imbalan/penalti kecil secara dinamis di setiap langkah transisi waktu (misal memberikan penalti kecemasan gradual seiring mendekatnya tanggal kedaluwarsa atau bonus kecil saat berhasil menjual stok kritis). Dampaknya, Neural Network mendapatkan sinyal gradien yang padat sehingga agen dapat belajar dengan jauh lebih cepat dan adaptif.
2. **Troubleshooting Perilaku AI Berputar di Tempai:**
 - *Penyebab:* Masalah ini umumnya terjadi karena Reward Progress positif yang diberikan terlalu besar atau longgar. Agen menemukan celah (loophole) di mana ia bisa terus mengumpulkan poin-poin kecil secara aman hanya dengan berputar-putar tanpa mau mengambil risiko bergerak lebih jauh yang berpotensi menabrak dinding.
 - *Solusi:* Penulis/perancang sistem harus menerapkan atau meningkatkan nilai Time Penalty (atau penalti eksistensi) bernilai negatif kecil (misalnya -0.01 tiap frame transisi). Penalti ini memaksa agen untuk menyelesaikan tugas seefisien dan secepat mungkin karena berlama-lama di dalam lingkungan akan terus mengurangi total skornya.

Pembahasan Bagian C (Analisis Kode Python)

1. **Aliran Data Input Model InventoryDQN:**
 - Pertama, data kondisi gudang berupa vektor berukuran 5 dimasukkan ke dalam model (misal: [Stok=10, Penjualan=3, Sisa_Hari=2, Harga=50k, Promo=0]).

- Data ini masuk ke lapisan pertama (self.fc1) untuk dipetakan dari 5 input menjadi 64 fitur, kemudian dievaluasi menggunakan fungsi aktivasi ReLU untuk mengabaikan nilai negatif secara efisien.
- Selanjutnya, data diteruskan ke self.fc2 untuk menangkap hubungan non-linear yang lebih kompleks antara sisa umur barang dan tren permintaan lewat 32 neuron.
- Terakhir, lapisan linear output (self.fc3) memetakan 32 fitur tersebut menjadi 3 nilai numerik kontinu yang merepresentasikan nilai estimasi keuntungan/skor kualitas (Q-Value) dari masing-masing aksi: $Q(\text{Diam})$, $Q(\text{Pesan})$, dan $Q(\text{Diskon})$. Agen kemudian akan memilih aksi yang memiliki nilai Q tertinggi.

2. Alasan Lapisan Output Menggunakan Aktivasi Linear (Tanpa ReLU):

Lapisan output pada arsitektur DQN bertugas memprediksi akumulasi imbalan masa depan (Expected Cumulative Reward) atau nilai fungsi Q, yang merepresentasikan nilai ekonomi (seperti profit atau total kerugian finansial). Nilai Q ini bersifat kontinu dan secara teoritis harus bisa bernilai negatif (apabila aksi tersebut diprediksi berujung pada kerugian besar atau barang busuk terbuang). Jika kita memaksakan penggunaan fungsi aktivasi non-linear seperti ReLU pada lapisan output, semua nilai prediksi negatif akan otomatis dipotong menjadi nol (\$0\$). Hal ini akan membuat AI mengalami kebutaan informasi, sehingga tidak dapat membedakan mana tingkat aksi yang "buruk" dan mana yang "sangat buruk/berbahaya"

DAFTAR PUSTAKA

- Aha, D. W. , K. D. , & A. M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6.
- Altman, N. S. (1992). *An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression*. .
- Babbage, C. (1864). *Passages from the Life of a Philosopher*. Longman, Green, Longman, Roberts, & Green.
- Bacon, P. L. , H. J. , & P. D. (2017). The option-critic architecture. *AAAI Conference on Artificial Intelligence*, 31(1).
- Barto, A. G. , & M. S. (2003). Recent advances in hierarchical reinforcement learning. *. Discrete Event Dynamic Systems*, 13(1), 41–77.
- Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.
- Bengio, Y. , C. A. , & V. P. (2013). Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 1798–1828.
- Bergstra, J. , & B. Y. (2012). Random Search for Hyper-parameter Optimization. *Journal of Machine Learning Research*, 281–305.
- Cambridge University Press. Dan Goodfellow, I. , B. Y. , & C. A. (2016). *Deep Learning*. MIT Press.
- Carpenter J. (2011). May the Best Algorithm Win: With sites like Kaggle, companies can crowdsource data problems to an army of experts.
- Cass, S. (2017). *The 2017 Top Programming Languages*.
- chebychef, P. (1854). *Théorie des mécanismes connus sous le nom de parall.*
- Chopra, S. , & M. P. (2016). *Supply Chain Management: Strategy, Planning, and Operation*. Pearson.
- Dahlander, L. , & P. H. (n.d.). Open to Suggestions: How organizations elicit ideas through open innovation crowdsourcing. *Academy of Management Journal*.
- Dan Anton, H. , & R. C. (2013). *Elementary Linear Algebra: Applications Version*. John Wiley & Sons.
- Deisenroth, M. P. , F. A. A. , & O. C. S. (2020). *Mathematics for Machine Learning*.
- Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). *Knowledge Discovery and Data Mining: Towards a Unifying Framework*. www.aaai.org
- Forbes. Wickham, H. (2014). Tidy Data. *. Journal of Statistical Software*, 59(10), 1–23.

- Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems. O'Reilly Media, Inc.
- Hamming, R. W. (1950). Error detecting and error correcting codes. . Bell System Technical Journal, 29(2), 147–160.
- Han, J., Kamber, M., & Pei, J. (2011). Data Mining. Concepts and Techniques, 3rd Edition (The Morgan Kaufmann Series in Data Management Systems).
- Hastie, T. , T. R. , & F. J. (2009). The Elements of Statistical Learning.
- Heath, T. L. (1956). The Thirteen Books of Euclid's Elements. Dover Publications.
- IEEE Spectrum. TIOBE Index. (2018). Programming Language of the Year 2018.
- Inman, J. (1835). Navigation and Nautical Astronomy: For the Use of British Seamen.
- James, G. , W. D. , H. T. , & T. R. (2013a). An Introduction to Statistical Learning. .
- James, G. , W. D. , H. T. , & T. R. (2013b). An Introduction to Statistical Learning (ISL).
- James, Gareth., Witten, Daniela., Hastie, Trevor., & Tibshirani, Robert. (2017). An introduction to statistical learning : with applications in R. Springer : Springer Science+Business Media.
- John Wiley & Sons dan Inmon, W. H. (2005). Building the Data Warehouse.
- Jordan, M., Kleinberg, J., & Schölkopf, B. (n.d.). Pattern Recognition and Machine Learning.
- Juliani, A. , B. V. P. , T. E. , et al. (2018). Unity: A General Platform for Intelligent Agents. . arXiv Preprint.
- Kimball, R. , & R. M. (2013). The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. .
- Kolb, D. (n.d.). Experiential learning: experience as the source of learning and development.
- Krause, E. F. (1986). Taxicab Geometry: An Adventure in Non-Euclidean Geometry. . Dover Publications.
- Laudon, K. C. , & L. J. P. (2021). Management Information Systems: Managing the Digital Firm. .
- LeCun, Y. , & C. C. (2010). M. handwritten digit database. Y. L. Website. (n.d.). THE MNIST DATABASE of handwritten digits. Retrieved <http://yann.lecun.com/exdb/mnist/>
- Liaw, R. , et al. (2020). Efficient Deployment of Machine Learning Models via ONNX. . Journal of Systems and Software.

- Lomborg, S. , & B. A. (2014). Using APIs for Social Media Research: A Methodology Paper.
- Lutz, M. (2018). Learning Python (5th Edition). 5.
- Manning, C. D. , R. P. , & S. H. (2008). Introduction to Information Retrieval. Cambridge University Press.
- Millington, I. , & F. J. (2018). Artificial Intelligence for Games (3rd ed.). CRC Press, 3.
- Minkowski, H. (1910). Geometrie der Zahlen (Geometry of Numbers). Teubner.
- Mitchell, T. M. (1997a). Machine Learning. McGraw-Hill Education.
- Mitchell, T. M. . (1997b). Machine Learning. McGraw-Hill.
- Mnih, V. , K. K. , S. D. , et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.
- Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys (CSUR)*, 33(1), 31–88.
- Oudeyer, P. Y. , & K. F. (2017). What is intrinsic motivation? A typology of computational approaches. *Frontiers in Neurorobotics*, 1–6.
- Pathak, D. , A. P. , E. A. A. , & D. T. (2017). Curiosity-driven exploration by self-supervised prediction. . *International Conference on Machine Learning (ICML)*.
- Pearson and Provost, F. , & F. T. (2013). Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking.
- Press, G. (2016). Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says. . *Forbes*.
- Rosenfeld, A. , & P. J. L. (1968). Distance Functions on Digital Pictures. *Pattern Recognition*, 1(1).
- Russell, M. A. , & K. S. S. (2018).). Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Instagram, GitHub, and More.
- Russell, S. J. , & N. P. (2020). Artificial Intelligence: A Modern Approach. Pearson.
- Shearer, C. (2000). The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of Data Warehousing*, 5, 13-22 .
- Sinnott, R. W. (1984). Virtues of the Haversine. *Sky and Telescope*, 68(2), 159.
- Sutton, R. S. , P. D. , & S. S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. . *Artificial Intelligence*, 112(1–2), 181–211.

- Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction
Second edition, in progress.
- Treveil, M. (2020). Introducing MLOps: How to Scale Machine Learning in the
Enterprise.
- Turing, A. M. (1950). COMPUTING MACHINERY AND INTELLIGENCE. In
Computing Machinery and Intelligence. Mind (Vol. 49).
- Zheng, A. , & C. A. (2018). Feature Engineering for Machine Learning: Principles and
Techniques for Data Scientists. O'Reilly Media.

BIOGRAFI PENULIS

Muhammad Ardiansyah Azman, seorang akademisi kelahiran Surakarta, 16 Mei 1995, yang saat ini aktif menjabat sebagai dosen di Universitas Bakrie, Jakarta. Beliau memiliki latar belakang pendidikan di bidang Teknik Industri yang kuat, diawali dengan gelar Sarjana Teknik (S-1) di bidang Teknik Industri dari Universitas Bakrie pada tahun 2021, yang kemudian dilanjutkan dengan meraih gelar Magister Teknik (S-2) di bidang Teknik Industri dari Universitas Indonesia pada tahun 2024.

Sebagai pengajar, beliau mengampu berbagai mata kuliah berbasis data dan sistem, seperti Dasar Pemrograman dan Algoritma, Riset Operasi, Analisa Perancangan dan Sistem Informasi dan Perancangan Tata Letak Fasilitas. Pengalaman akademisnya yang mendalam di bidang data, pemodelan, dan perancangan sistem industri ini menjadi fondasi teoretis yang sangat kuat dalam penyusunan buku Perancangan Reinforcement Learning: Teori Pendukung, Lifecycle, dan Implementasi Python.

Tri Susanto, seorang akademisi dan dosen senior kelahiran Surabaya, 6 Desember 1958, yang saat ini aktif menjabat sebagai Lektor di Universitas Bakrie, Jakarta. Beliau memiliki latar belakang pendidikan multidisiplin yang kuat, diawali dengan gelar Sarjana Ekonomi (S-1) di bidang Manajemen dari Universitas Jenderal Soedirman Purwokerto pada tahun 1984, yang kemudian dilanjutkan dengan meraih gelar Magister Teknik (S-2) di bidang Teknik Industri dari Institut Teknologi Bandung (ITB) pada tahun 1989.

Sebagai pengajar, beliau mengampu berbagai mata kuliah fundamental berbasis data dan sistem, seperti Statistik Industri, Teori Probabilitas, Rancangan Percobaan, Manajemen Proyek, serta Mutu Terpadu untuk Jasa. Pengalaman akademisnya yang mendalam di bidang statistika, probabilitas, dan perancangan sistem industri ini menjadi fondasi teoretis yang sangat kuat dalam penyusunan buku Perancangan Reinforcement Learning: Teori Pendukung, Lifecycle, dan Implementasi Python. Melalui kontribusinya, pembaca dituntun untuk memahami akar matematika pendukung serta siklus hidup (lifecycle) perancangan kecerdasan buatan (Artificial Intelligence) secara terstruktur dan aplikatif

Almindo Vabiano, seorang profesional berlatar belakang Teknik Industri dari Universitas Bakrie yang saat ini berkarir sebagai Staf Laboratorium. Dalam perannya, ia memiliki pengalaman kuat dalam melakukan analisis data eksperimental menggunakan perangkat lunak statistik untuk mendukung pengambilan keputusan berbasis riset. Ketertarikannya pada pengolahan data dan sistem cerdas didukung oleh keahliannya dalam bahasa pemrograman seperti Python, serta pemahamannya yang mendalam tentang teknologi terkini melalui berbagai sertifikasi di bidang Cloud Computing, Internet of Things (IoT), dan visualisasi data. Perpaduan antara kemampuan analitis, rekam jejak teknis, dan visi inovatifnya menjadikan Almindo sosok praktisi yang sangat menyadari pentingnya literasi Machine Learning untuk memecahkan berbagai tantangan di era digital saat ini.

Buku Perancangan Reinforcement Learning: Teori Pendukung, Lifecycle, dan Implementasi Python (Edisi 1) karya Muhammad Ardiansyah Azman, Tri Susanto, dan Almindo Vabiano hadir sebagai panduan praktis dan aplikatif bagi para pemula maupun praktisi yang ingin menyelami dunia kecerdasan buatan. Diterbitkan atas dukungan Program Studi Teknik Industri Universitas Bakrie dan UBakrie Press, buku ini dirancang secara sistematis dengan pendekatan yang ramah pembaca, mengombinasikan pemahaman konseptual, analogi intuitif, serta panduan praktis berbasis bahasa pemrograman Python.

Perjalanan belajar diawali dengan mengenalkan intuisi dasar bagaimana mesin belajar dari pengalaman layaknya manusia, yaitu melalui proses mengingat (memorize), menggeneralisasi (generalize), dan memprediksi (predict). Pembaca diajak menelusuri sejarah perkembangan kecerdasan buatan dari era pra-1920-an hingga era Deep Learning modern, serta memahami perbedaan mendasar antara Supervised Learning, Unsupervised Learning, dan Reinforcement Learning. Pembahasan ini diperjelas dengan contoh penerapan nyata di kehidupan sehari-hari, mulai dari mesin pencari, penyaring spam, sistem rekomendasi e-commerce, hingga teknologi mobil tanpa pengemudi. Untuk menopang pemahaman teknis, buku ini juga menyediakan fondasi teori komputasi melalui ekosistem Python (seperti NumPy, Pandas, Matplotlib, dan Jupyter Notebook) serta landasan matematika seperti aljabar linier, kalkulus, dan statistika.

Selanjutnya, buku ini menguraikan tahapan Machine Learning Lifecycle secara terstruktur dan iteratif. Pembaca dibimbing mulai dari akuisisi data, pembersihan (preprocessing), rekayasa fitur, pemisahan data training dan testing, pelatihan model, hingga penerapan (deployment) aplikasi menggunakan API berbasis FastAPI. Memasuki topik utama Reinforcement Learning, buku ini mengeksplorasi penggunaan algoritma Deep Q-Network (DQN) untuk kasus nyata seperti navigasi robot dan optimasi sistem inventaris. Pembahasan berlanjut ke materi tingkat lanjut seperti Hierarchical Reinforcement Learning (HRL) untuk akuisisi keterampilan agen cerdas, penanganan data drift, hingga pengambilan keputusan waktu