



MACHINE LEARNING

Penulis :

- Angga Aditya Permana
- Wahyuuddin S
- Leo Willyanto Santoso
- Gentur Wahyu Nyipto Wibowo
- Anindya Khrisna Wardhani
- Rahmaddeni
- Ahmad Jurnaidi Wahidin
- Gusti Eka Yuliasuti
- Elisawati
- Rima Rizqi Wijayanti
- Abdurrasyid



ISBN 978-623-198-072-4



9 786231 980724

MACHINE LEARNING

**Angga Aditya Permana
Wahyuddin S
Leo Willyanto Santoso
Gentur Wahyu Nyipto Wibowo
Anindya Khrisna Wardhani
Rahmadden
Ahmad Jurnaidi Wahidin
Gusti Eka Yuliasuti
Elisawati
Rima Rizqi Wijayanti
Abdurrasyid**



PT GLOBAL EKSEKUTIF TEKNOLOGI

MACHINE LEARNING

Penulis :

Angga Aditya Permana
Wahyuuddin S
Leo Willyanto Santoso
Gentur Wahyu Nyipto Wibowo
Anindya Khrisna Wardhani
Rahmadden
Ahmad Jurnaidi Wahidin
Gusti Eka Yuliasuti
Elisawati
Rima Rizqi Wijayanti
Abdurrasyid

ISBN : 978-623-198-072-4

Editor : Ari Yanto. M.Pd.

Penyunting : Tri Putri Wahyuni, S.Pd

Desain Sampul dan Tata Letak : Atyka Trianisa, S.Pd

Penerbit : PT GLOBAL EKSEKUTIF TEKNOLOGI

Anggota IKAPI No. 033/SBA/2022

Redaksi :

Jl. Pasir Sebelah No. 30 RT 002 RW 001
Kelurahan Pasie Nan Tigo Kecamatan Koto Tengah
Padang Sumatera Barat
Website : www.globaleksekutifteknologi.co.id
Email : globaleksekutifteknologi@gmail.com

Cetakan pertama, 11 Februari 2023

Hak cipta dilindungi undang-undang
Dilarang memperbanyak karya tulis ini dalam bentuk
dan dengan cara apapun tanpa izin tertulis dari penerbit.

KATA PENGANTAR

Segala Puji dan syukur atas kehadiran Allah SWT dalam segala kesempatan. Sholawat beriring salam dan doa kita sampaikan kepada Nabi Muhammad SAW. Alhamdulillah atas Rahmat dan Karunia-Nya penulis telah menyelesaikan Buku Machine Learning ini.

Buku Ini Membahas Machine learning, Algoritma machine learning, Linier regression, K Nearest neighbor, K Means, Naive bayes, K Medoids, Genetic Algorithm, Fuzzy C-Means, Confusion Matrix, K Fold Cross Validation.

Proses penulisan buku ini berhasil diselesaikan atas kerjasama tim penulis. Demi kualitas yang lebih baik dan kepuasan para pembaca, saran dan masukan yang membangun dari pembaca sangat kami harapkan.

Penulis ucapkan terima kasih kepada semua pihak yang telah mendukung dalam penyelesaian buku ini. Terutama pihak yang telah membantu terbitnya buku ini dan telah mempercayakan mendorong, dan menginisiasi terbitnya buku ini. Semoga buku ini dapat bermanfaat bagi masyarakat Indonesia.

Padang, 11 Februari 2023

Penulis

DAFTAR ISI

KATA PENGANTAR	i
DAFTAR ISI	ii
DAFTAR GAMBAR	vi
DAFTAR TABEL	viii
BAB 1 MACHINE LEARNING	1
1.1 Pendahuluan.....	1
1.1.1 Jenis-jenis masalah dan tugas	3
1.1.2 Terminologi Pembelajaran Mesin	8
1.2 Sejarah dan hubungan dengan bidang lainnya	9
1.2.1 Kaitannya dengan statistik.....	12
1.3 Teori	12
1.4 Pendekatan.....	13
1.4.1 Pembelajaran pohon keputusan.....	13
1.4.2 Pembelajaran aturan asosiasi	14
1.4.3 Jaringan syaraf tiruan	14
1.4.4 Pemrograman logika induktif	14
1.4.5 Support Vector Machines	15
1.4.6 Pengelompokan.....	15
1.4.7 Jaringan Bayesian.....	15
1.4.8 Penguatan pembelajaran.....	16
1.4.9 Pembelajaran representasi.....	16
1.4.10 Kemiripan dan pembelajaran metrik	17
1.4.11 Pembelajaran kamus jarang.....	17
1.4.12 Algoritma genetika	18
1.5 Aplikasi	18
1.6 Perangkat Lunak	20
1.6.1 Perangkat lunak open source.....	20
1.6.2 Perangkat lunak komersial dengan edisi Open Source	20
1.6.3 Perangkat lunak komersial.....	21
DAFTAR PUSTAKA.....	22

BAB 2 ALGORITMA MACHINE LEARNING	25
2.1 Algoritma Machine Learning	25
2.1.1 <i>Supervised Learning</i>	25
2.1.2 <i>Unsupervised Learning</i>	26
2.1.3 <i>Semi-supervised Learning</i>	30
2.1.4 <i>Reinforcement Learning</i>	32
2.1.5 <i>Develompmental Learning Algorithm</i>	33
DAFTAR PUSTAKA	34
BAB 3 LINEAR REGRESSION	35
3.1 Pendahuluan	35
3.2 Model Regresi Linier	35
3.2.1 <i>Simple Linear Regression</i> /Regresi Linier Sederhana	36
3.2.2 <i>Multiple Linear Regression</i> /Regresi Linier Berganda	36
3.3 Regresi Linier dengan Python	37
DAFTAR PUSTAKA	44
BAB 4 KNN (K-NEAREST NEIGHBOR)	45
4.1 Pendahuluan	45
4.2 Algoritma K-Nearest Neighbor	45
4.2.1 Kelebihan dan Kekurangan Algoritma KNN	47
4.2.2 Penghitungan Manual Algoritma K-Nearest Neighbor	47
4.2.3 Implementasi K-Nearest Neighbor	51
DAFTAR PUSTAKA	59
BAB 5 K-MEANS	61
5.1 Pendahuluan	61
5.2 Langkah – Langkah K-Means	62
5.3 Kelebihan dan Kelemahan K-Means	64
5.4 Implementasi K-Means	64
DAFTAR PUSTAKA	71
BAB 6 ALGORITMA NAÏVE BAYES	73
6.1 <i>Algoritma Naïve Bayes</i>	73

6.2 Kelebihan dan Kekurangan Naive Bayes	75
6.2.1 Kelebihan Algoritma Naive Bayes	75
6.2.2 Kekurangan Algoritma Naive Bayes.....	75
6.3 Persamaan dan Langkah Naive Bayes.....	76
6.4 Tipe Algoritma Naive Bayes	77
6.4.1 Naive Bayes Classifier	77
6.4.2 Multinomial Naive Bayes	77
6.4.3 Gaussian Naive Bayes Classifier.....	77
6.5 Studi Kasus Algoritma Naive Bayes.....	78
6.5.1 Perhitungan Manual Algoritma Naive Bayes	79
6.5.2 Perhitungan Algoritma Naive Bayes Menggunakan Program Phyton.....	82
DAFTAR PUSTAKA.....	89
BAB 7 K MEDOIDS	93
7.1 Pendahuluan.....	93
7.2 K Medoids.....	93
7.2.1 Langkah - Langkah K-Medoids	97
7.2.2 Contoh Implementasi K Medoids.....	97
DAFTAR PUSTAKA.....	104
BAB 8 GENETIC ALGORITHM.....	105
8.1 Pendahuluan.....	105
8.2 Representasi Kromosom.....	110
8.3 Proses Evaluasi.....	113
8.4 Proses Reproduksi	113
8.4.1 Crossover	113
8.4.2 Mutation	117
8.5 Proses Seleksi.....	119
8.6 Modifikasi pada Genetic Algorithm.....	121
DAFTAR PUSTAKA.....	123
BAB 9 FUZZY C-MEANS (FCM)	127
9.1 Pendahuluan.....	127
9.2 Konsep Fuzzy C-Means.....	128
9.2.1 <i>Algoritma Fuzzy C-Means</i> (FCM).....	128

9.2.2 Contoh Penerapan <i>Fuzzy C-Means</i> (FCM)	130
9.2.3 Hasil <i>Output Fuzzy C-Means</i> dengan Matlab	136
DAFTAR PUSTAKA.....	139
BAB 10 CONFUSION MATRIX	141
10.1 Pendahuluan.....	141
10.2 Permasalahan Klasifikasi Binary	143
10.2.1 Confusion matrix pada binary classification	144
10.3 Contoh Penggunaan Confusion Matrix.....	151
DAFTAR PUSTAKA.....	154
BAB 11 K-FOLD CROSS VALIDATION	155
11.1 Pendahuluan.....	155
11.2 K-Fold Cross Validation.....	157
11.3 Contoh penggunaan k-fold cross-validation	159
DAFTAR PUSTAKA.....	163
BIODATA PENULIS	

DAFTAR GAMBAR

Gambar 1.1 : <i>Perbedaan Supervised dan Unsupervised Learning</i>	4
Gambar 1.2 : <i>Reinforcement Learning</i>	5
Gambar 1.3 : <i>Support Vector Machine</i>	6
Gambar 1.4 : <i>Classification Vs Regression</i>	7
Gambar 1.5 : Terminologi Pembelajaran Mesin	9
Gambar 2.1 : <i>Supervised Learning</i>	26
Gambar 2.2 : Hasil Analisis <i>Clustering</i>	28
Gambar 2.3 : <i>Supervised Learning</i>	29
Gambar 2.4 : <i>Semi-supervised Learning</i>	32
Gambar 2.5 : <i>Reinforcement Learning</i>	33
Gambar 4.1 : Hasil Dataset	52
Gambar 4.2 : Hasil Dataset	53
Gambar 4.3 : Hasil Dataset	58
Gambar 5.1 : Flowchart K-Means.....	63
Gambar 6.1 : Tampilan dataset	t83
Gambar 6.2 : Chart jumlah stroke dan no stroke	84
Gambar 6.3 : Tampilan report splitting data 80:20	86
Gambar 6.4 : Tampilan report splitting data 70:30	86
Gambar 6.5 : Visualisasi dengan splitting data 70:30 dan 80:20	87
Gambar 7.1 : Ilustrasi clustering K-Medoids dan K-means.....	95
Gambar 7.2 : Ilustrasi Euclidean Distance.....	96
Gambar 8.1 : Siklus <i>Genetic Algorithm</i>	107
Gambar 8.2 : Skema <i>Genetic Algorithm</i>	109
Gambar 8.3 : Ilustrasi Teknik <i>N-Point Crossover</i>	114
Gambar 8.4 : Ilustrasi Teknik <i>Uniform Crossover</i>	115
Gambar 8.5 : Ilustrasi Teknik <i>Position Based Crossover</i>	116
Gambar 8.6 : Ilustrasi Teknik <i>Order Based Crossover</i>	116
Gambar 8.7 : Ilustrasi Teknik <i>Binary Code Mutation</i>	117

Gambar 8.8 : Ilustrasi Teknik <i>Position Based Mutation</i>	118
Gambar 8.9 : Ilustrasi Teknik <i>Order Based Mutation</i>	118
Gambar 8.10 : Ilustrasi Teknik <i>Scramble Mutation</i>	119
Gambar 10.1 : Hubungan low recall, low precision.....	146
Gambar 10.2: Hubungan high recall, low precision	147
Gambar 10.3 : Hubungan low recall, high precision	148
Gambar 10.4 : Hubungan high recall, high precision.....	148
Gambar 11.1 : Diagram alur alur kerja <i>cross validation</i>	156
Gambar 11.2 : Cara kerja k-folds <i>cross-validation</i>	158

DAFTAR TABEL

Tabel 4.1 : Data Training (Data Latih).....	48
Tabel 4.2 : Data Testing (Data Uji).....	48
Tabel 4.3 : Hitung Menggunakan <i>Euclidean Distance</i>	49
Tabel 4.4 : Hasil Hitung Menggunakan <i>Euclidean Distance</i>	49
Tabel 4.5 : Penentuan Kategori.....	50
Tabel 4.6 : Hasil Klasifikasi Berdasarkan Kategori Mayoritas.....	51
Tabel 5.1 : Dataset	65
Tabel 5.2 : Centroid Awal	66
Tabel 5.3 : Hasil Euclidian Distance.....	67
Tabel 5.4 : Centroid Baru	68
Tabel 6.1 : Jumlah kelas pertarget dan prior	79
Tabel 6.2 : Jumlah kasus perkelas.....	79
Tabel 6.3 : Perhitungan semua variabel kelas.....	80
Tabel 6.4 : Perbandingan hasil perkelas.....	82
Tabel 6.5 : Hasil perbandingan akurasi berdasarkan splitting data.....	88
Tabel 7.1 : Contoh Kelompok Data.....	98
Tabel 7.2 : Medoid Iterasi Pertama.....	98
Tabel 7.3 : Jarak Objek Ke Medoid Pada Iterasi Pertama.....	101
Tabel 7.4 : Medoid Iterasi Kedua.....	101
Tabel 7.5 : Jarak Objek Ke Medoid Pada Iterasi Kedua	102
Tabel 8.1 : Konversi Proses Alamiah Menjadi Proses Komputasi.....	108
Tabel 8.2 : Pengodean Biner untuk Representasi Bilangan Bulat	110
Tabel 8.3 : Pengodean Biner untuk Representasi Bilangan Riil.....	111
Tabel 8.4 : Pengodean Riil.....	112
Tabel 8.5 : Pengodean Integer	112

Tabel 8.6 : Pengodean Permutasi 112

Tabel 9.1 : Data Pelanggaran Lalu Lintas 130

Tabel 9.2 : Pembagian *Cluster* 136

Tabel 9.3 : Pembagian data berdasarkan *cluster* 136

Tabel 10.1 : Dataset klasifikasi buah 141

Tabel 10.2 : Kelas prediksi dan aktual 142

Tabel 10.3 : Confusion matrix biner 144

Tabel 10.4 : Peta evaluasi model untuk klasifikasi
biner 149

BAB 1

MACHINE LEARNING

Oleh Angga Aditya Permana

1.1 Pendahuluan

Selama dua dekade terakhir Machine Learning telah menjadi salah satu andalan teknologi informasi dan dengan itu, bagian yang agak sentral, meskipun biasanya tersembunyi, dari hidup kita. Dengan semakin banyaknya data yang tersedia, ada pendapat yang sangat baik untuk mempercayai bahwa menganalisa data dengan cerdas akan menjadikan data lebih meresap sebagai bahan baku yang dibutuhkan dalam teknologi yang maju. Tujuan bab ini adalah untuk memberi pembaca gambaran umum tentang berbagai macam aplikasi yang pada intinya memiliki masalah pembelajaran mesin dan untuk membawa beberapa tingkat keteraturan ke kebun binatang masalah. Setelah itu, kita akan membahas beberapa alat dasar dari statistik dan teori probabilitas, karena mereka membentuk bahasa di mana banyak masalah pembelajaran mesin harus diutarakan agar dapat diselesaikan. Akhirnya, kami akan menguraikan serangkaian algoritma yang cukup mendasar namun efektif untuk memecahkan masalah penting, yaitu klasifikasi. Alat yang lebih canggih, diskusi tentang masalah yang lebih umum dan analisis terperinci akan menyusul di bagian selanjutnya dari buku ini.

Pembelajaran mesin ialah cabang ilmu komputer (Dönmez, 2013) yang berkembang dari bidang pengenalan pola dan teori pembelajaran komputer dalam kecerdasan buatan. (Dönmez, 2013) Pembelajaran mesin mempelajari konstruksi dan analisis algoritma pembelajaran dan prediksi data. (Dönmez, 2013) Algoritma semacam itu membangun model berdasarkan input

model untuk mendapatkan sebuah prediksi atau keputusan yang didasarkan dari data (Bharadwaj, Prakash and Kanagachidambaresan, 2021) daripada mengikuti instruksi program statis secara ketat.

Pembelajaran mesin juga berkaitan dan beririsan dengan statistik komputasi; sebuah disiplin ilmu yang membahas dalam pembuatan sebuah prediksi. Pembelajaran mesin memiliki irisan yang kuat dengan pengoptimalisasi matematis, yang membahas teori, metode dan domai aplikasi kedalam impelmentasinya. Pembelajaran mesin dapat digunakan dalam beberapa tugas kumputasi yang melibatkan perancangan dan pemrograman algoritma secara eksplisit tidak dapat dilakukan. Contoh aplikasi termasuk pemfilteran spam, pengenalan karakter optik (OCR), (Shad *et al.*, 2021) mesin telusur, dan *computer vision*. Pembelajaran mesin terkadang digabungkan dengan penambangan data, (Mannila, 1996) meskipun itu lebih berfokus pada analisis data eksplorasi. (Friedman, 1997) Pembelajaran mesin dan pengenalan pola “dapat dilihat sebagai dua sisi dari bidang yang sama.”(Bharadwaj, Prakash and Kanagachidambaresan, 2021)

Ketika digunakan dalam konteks industri, metode pembelajaran mesin dapat disebut sebagai analitik prediktif atau pemodelan prediktif.

Arthur Samuel di tahun 1951 mengartikan pembelajaran mesin sebagai “Bidang studi yang memberi komputer kemampuan untuk belajar tanpa diprogram secara eksplisit”. (Dönmez, 2013)

Tom M. Mitchell juga mendefinisikan pengertian pembelajaran mesin sebagai: “Sebuah program komputer dikatakan belajar dari pengalaman (E) sehubungan dengan beberapa kelas tugas (T) dan kinerja mengukur (P), jika kinerjanya pada tugas di (T), yang diukur dengan (P), meningkat dengan pengalaman (E)”. (Dönmez, 2013) Definisi ini terkenal karena mendefinisikan pembelajaran mesin dalam operasional

fundamental daripada istilah kognitif, sehingga dikutip dari Alan Turing dalam makalahnya "*Computing Machinery and Intelligence*" bahwa "Bisakah mesin berpikir?" dapat diubah dengan sebuah pertanyaan "Dapatkah mesin melakukan apa yang dapat kita (sebagai entitas berpikir) lakukan?"

1.1.1 Jenis-jenis masalah dan tugas

Tugas pembelajaran mesin biasanya diklasifikasikan ke dalam tiga kategori besar, tergantung pada sifat "sinyal" pembelajaran atau "umpan balik" yang tersedia untuk sistem pembelajaran. Ini adalah: (Goel and Davies, 2019)

1. ***Supervised Learning***: Model atau algoritma disajikan dengan contoh masukan dan keluaran yang diinginkan kemudian dicari pola dan hubungan antara masukan dan keluaran tersebut. Tujuannya adalah untuk mempelajari aturan umum yang memetakan input ke output. Proses pelatihan berlanjut hingga model mencapai tingkat akurasi yang diinginkan pada data pelatihan. Beberapa contoh kehidupan nyata adalah:

- **Klasifikasi Gambar**: Anda berlatih dengan gambar/label. Kemudian di masa mendatang Anda memberikan gambar baru dengan harapan komputer akan mengenali objek baru tersebut.
- **Prediksi/Regresi Pasar**: Anda melatih komputer dengan data pasar historis dan meminta komputer memprediksi harga baru di masa mendatang.

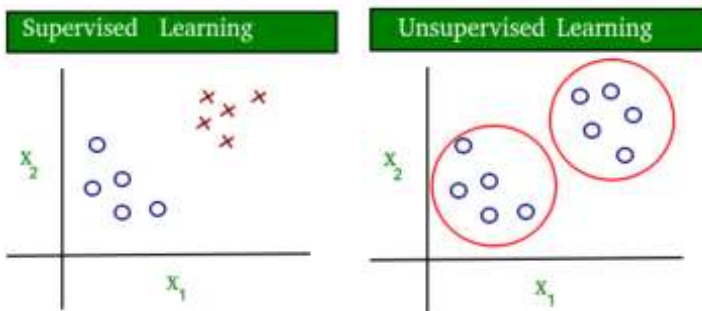
2. ***Unsupervised Learning***: Pada Unsupervised Learning tidak diberikan label pada algoritma pembelajaran, membiarkan algoritma tertentu menemukan pola sebagai masukannya. Dapat dimanfaatkan dalam mengelompokan populasi yang memiliki perbedaan. Unsupervised Learning dapat mengelompokan data tersendiri dan menemukan pola tersembunyi yang berada didalam data.

Pengelompokan: Anda meminta komputer untuk memisahkan data serupa ke dalam kelompok, ini penting dalam penelitian dan sains.

Visualisasi Dimensi Tinggi: Gunakan komputer untuk membantu kami memvisualisasikan data dimensi tinggi.

Model Generatif: Setelah model menangkap distribusi probabilitas dari data input Anda, model tersebut akan dapat menghasilkan lebih banyak data. Ini bisa sangat berguna untuk membuat classifier Anda lebih kuat.

Diagram sederhana yang menjelaskan konsep pembelajaran terawasi dan tidak terawasi ditunjukkan di bawah ini:

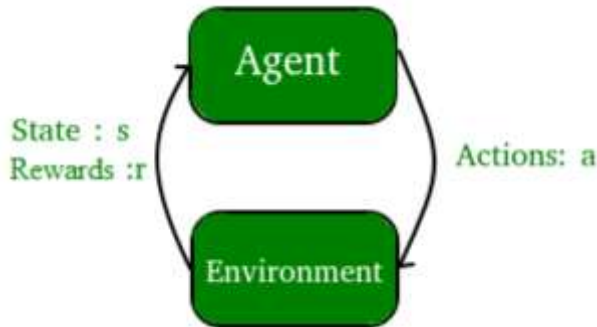


Gambar 1.1 : Perbedaan *Supervised* dan *Unsupervised Learning*

Seperti yang Anda lihat dengan jelas, data dalam *supervised learning* **diberi label**, sedangkan data dalam *unsupervised learning* **tidak diberi label**.

3. **Reinforcement Learning:** Sebuah agent yang dapat beradaptasi dengan lingkungan sekitar yang sangat dinamis di mana agent tersebut dapat memiliki tujuan tertentu (seperti berkendara), tanpa seorang guru secara eksplisit mengatakan apakah ia mendekati tujuannya atau

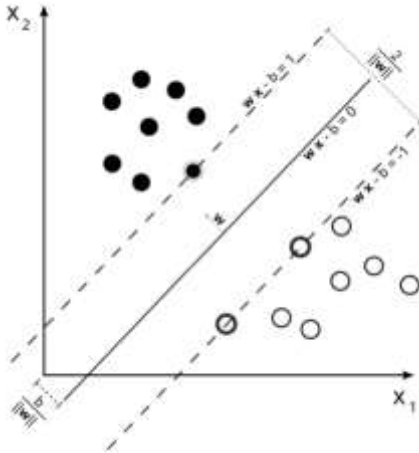
tidak. Contoh lain adalah belajar bermain game dengan bermain melawan lawan. (Bharadwaj, Prakash and Kanagachidambaresan, 2021)



Gambar 1.2 : *Reinforcement Learning*

Antara *Supervised Learning* dan *Unsupervised Learning* adalah ***Semi-supervised learning***: Masalah di mana Anda memiliki sejumlah besar data input dan hanya beberapa data yang diberi label, disebut masalah pembelajaran semi-diawasi. Masalah-masalah ini terdapat di antara *supervised learning* dan *unsupervised learning*. Misalnya, arsip foto yang hanya beberapa gambar yang diberi label, (mis. anjing, kucing, orang) dan sebagian besar tidak diberi label.

Di antara kategori masalah pembelajaran mesin lainnya, belajar untuk belajar mempelajari bias induktifnya sendiri berdasarkan pengalaman sebelumnya. *Reinforcement Learning*, dijabarkan untuk pembelajaran robot, menghasilkan urutannya sendiri (juga disebut kurikulum) dari situasi pembelajaran untuk secara kumulatif mendapatkan repertoar kemampuan yang belum pernah dimiliki melalui berbagai percobaan diri otonom dan interaksi sosial dengan guru manusia, dan menggunakan mekanisme bimbingan seperti pembelajaran aktif, pematangan, sinergi motorik, dan imitasi.



Gambar 1.3 : *Support Vector Machine*

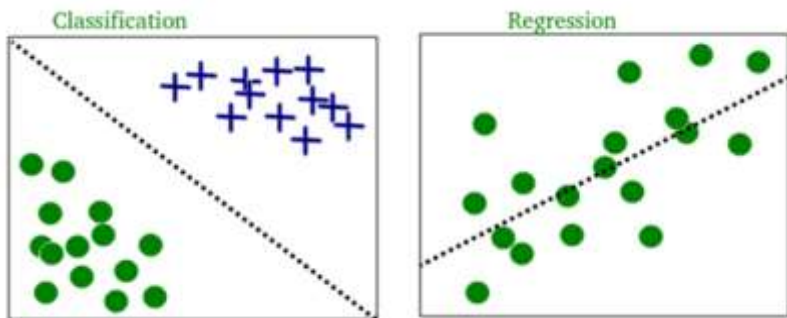
Support Vector Machine adalah pengklasifikasi yang membagi ruang inputnya menjadi dua wilayah, terpisahkan oleh garis linier. Dia belajar membedakan lingkaran putih dan hitam.

Kategorisasi lain dari tugas pembelajaran mesin muncul ketika seseorang mempertimbangkan keluaran yang diinginkan dari sistem pembelajaran mesin: (Bharadwaj, Prakash and Kanagachidambaresan, 2021)

Dua kasus penggunaan paling umum dari *Supervised learning* adalah:

- **Klasifikasi**, didalam klasifikasi masukan dipisah menjadi dua kelas atau lebih, dan pelajar harus membuat sebuah model yang menugaskan masukan tak terlihat ke satu (atau klasifikasi multi-label) atau lebih dari kelas-kelas ini. Ini biasanya ditangani dengan cara yang diawasi. Pemfilteran spam adalah contoh klasifikasi, di mana inputnya adalah pesan email (atau lainnya) dan kelasnya adalah "spam" dan "bukan spam".
- Dalam **regresi**, juga merupakan masalah yang diawasi, keluarannya bersifat kontinyu, bukan diskrit.

Contoh klasifikasi dan regresi pada dua kumpulan data yang berbeda ditunjukkan di bawah ini:



Gambar 1.4 : *Classification Vs Regression*

Pembelajaran *Unsupervised learning* yang paling umum adalah

- **Clustering**, satu set masukan harus dibagi menjadi beberapa kelompok. Tidak seperti dalam klasifikasi, kelompok tidak diketahui sebelumnya, membuat tugas ini biasanya tanpa pengawasan.
- **Density estimation** menemukan distribusi input di beberapa ruang.
- **Dimensionality reduction** menyederhanakan masukan dengan memetakannya ke dalam ruang berdimensi lebih rendah. Pemodelan topik adalah masalah yang terkait, di mana sebuah program diberikan daftar dokumen bahasa manusia dan bertugas untuk mengetahui dokumen mana yang mencakup topik serupa.

Berdasarkan tugas/masalah pembelajaran mesin ini, memiliki sejumlah algoritma yang digunakan untuk menyelesaikan tugas ini. Beberapa algoritma pembelajaran mesin yang umum digunakan adalah Regresi Linier, Regresi Logistik, Pohon

Keputusan, SVM (*Support vector machines*), *Naive Bayes*, KNN (*K nearest neighbors*), K-Means, *Random Forest*, dll.

1.1.2 Terminologi Pembelajaran Mesin

Model adalah representasi spesifik yang dipelajari dari data dengan menerapkan beberapa algoritma pembelajaran mesin. Model disebut juga hipotesis.

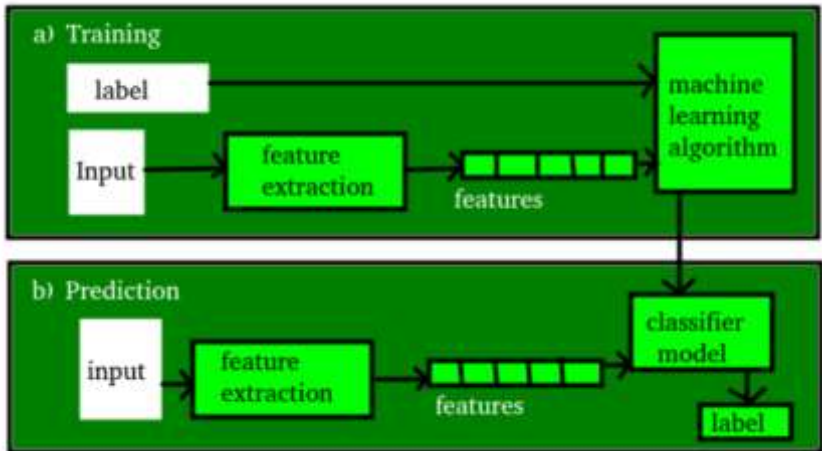
Fitur adalah properti terukur individual dari data kami. Satu set fitur numerik dapat dengan mudah dijelaskan oleh vektor fitur. Vektor fitur dimasukkan sebagai input ke model. Misalnya, untuk memprediksi buah, mungkin ada fitur seperti warna, bau, rasa, dll. Catatan: Memilih fitur informatif, diskriminatif, dan independen merupakan langkah penting untuk algoritma yang efektif. Kami biasanya menggunakan ekstraktor fitur untuk mengekstrak fitur yang relevan dari data mentah.

Target (Label) Variabel atau label target adalah nilai yang akan diprediksi oleh model kita. Untuk contoh buah yang dibahas di bagian fitur, label dengan setiap set masukan adalah nama buah seperti apel, jeruk, pisang, dll.

Pelatihan Idenya adalah untuk memberikan satu set input (fitur) dan output yang diharapkan (label), jadi setelah pelatihan, kita akan memiliki model (hipotesis) yang kemudian akan memetakan data baru ke salah satu kategori yang dilatih.

Prediksi Setelah model kita siap, model tersebut dapat diberi input yang akan memberikan output (label) prediksi. Tapi pastikan jika mesin bekerja dengan baik pada data yang tidak terlihat, maka hanya kita yang bisa mengatakan mesin bekerja dengan baik.

Gambar yang ditunjukkan di bawah ini menjelaskan konsep di atas:



Gambar 1.5 : Terminologi Pembelajaran Mesin

1.2 Sejarah dan hubungan dengan bidang lainnya

Pembelajaran mesin tercipta dari pengembangan kecerdasan buatan. Bahkan pada awal kecerdasan buatan lahir sebagai ilmu baru dalam bidang akademis, banyak peneliti yang tertarik membiarkan mesin belajar dari banyaknya data. Mereka mencoba memecahkan masalah dengan berbagai metode dan metode simbolik yang disebut "jaringan saraf". Perceptron menjadi bagian yang besar serta model lainnya yang kemudian ditemukan menjadi sebuah model statistik linier umum. Penalaran probabilistik juga sebagian besar digunakan dalam diagnosis medis otomatis. (Goel and Davies, 2019)

Namun, meningkatnya penekanan pada pendekatan logis dan berbasis pengetahuan menciptakan kesenjangan antara AI dan pembelajaran mesin. Sistem probabilistik terganggu oleh masalah teoretis dan praktis dari akuisisi dan representasi data. (Goel and Davies, 2019) Sebelum tahun 1980, AI didominasi oleh sistem

pakar dan statistik. Bekerja pada pembelajaran berbasis simbolik/pengetahuan berlanjut di AI, yang mengarah ke pemrograman logika induktif, tetapi lebih banyak arah penelitian statistik sekarang berada di luar cakupan AI yang tepat, dalam pengenalan pola dan pengambilan informasi. (Goel and Davies, 2019) Kecerdasan buatan dan komputer meninggalkan penelitian jaringan saraf pada waktu yang hampir bersamaan. Silsilah ini juga dilanjutkan di luar bidang AI/CS sebagai "koneksionisme" oleh para peneliti di disiplin lain seperti Hopfield, Rumelhart, dan Hinton. Pada pertengahan tahun 1980an, lahirlah sebuah terobosan terbesar ketika hamburan balik ditemukan kembali. (Goel and Davies, 2019)

Direorganisasi ke dalam bidangnya sendiri, pembelajaran mesin mulai muncul pada 1990-an. Industri ini telah mengalihkan fokusnya dari mencapai kecerdasan buatan menjadi memecahkan masalah yang hampir dapat dipecahkan. Ini mengalihkan fokus dari pendekatan simbolik yang diwarisi dari kecerdasan buatan ke metode dan model yang dipinjam dari statistik dan teori probabilitas. (Langley, 2011) Ini juga mendapat manfaat dari meningkatnya ketersediaan informasi digital dan kemampuan untuk menyebarkannya melalui internet.

Pembelajaran mesin dan penambangan data sering menggunakan metode yang sama dan tumpang tindih. Mereka secara kasar dapat dibedakan sebagai berikut:

- Pembelajaran mesin berfokus pada pembuatan prediksi berdasarkan fitur yang diketahui yang dipelajari dari data pelatihan.
- Penambangan data berfokus pada menemukan (sebelumnya) properti yang tidak diketahui dalam data. Ini adalah tahap analisis Penemuan Pengetahuan di Database.

Kedua area ini tumpang tindih dalam banyak hal:

Teknik pembelajaran mesin banyak digunakan untuk penambangan data, tetapi seringkali ditemukan perbedaan dalam tujuannya. Pada pembelajaran *Unsupervised Learning* atau langkah pra-pemrosesan dalam penambangan data dapat menggunakan sebagai upaya untuk meningkatkan akurasi didalam pembuatan model. Terdapat dua pendapat dari dua komunitas ilmiah (yang seringkali memiliki konferensi dan jurnal terpisah, ECML PKDD adalah pengecualian penting) berasal dari asumsi dasar di mana mereka beroperasi:

Pada pembelajaran mesin, keefektifan dapat diukur dengan pengulangan studi. Informasi yang diketahui, sedangkan dalam KDD (*Knowledge Discovery and Data Mining*) tugas utamanya adalah menemukan informasi yang sebelumnya tidak diketahui. Metode uninformed (*unsupervised*) sedikit lebih baik daripada metode supervised ketika dievaluasi terhadap informasi yang diketahui, sedangkan untuk tugas KDD biasa, metode supervised tidak tepat untuk diimplementasikan karena data latih tidak terlalu banyak.

Pembelajaran mesin juga terkait erat dengan pengoptimalan:

Banyak masalah pembelajaran diformulasikan untuk meminimalkan kerugian di antara contoh pelatihan. Fungsi kerugian mengungkapkan perbedaan antara prediksi model yang dilatih dan contoh masalah aktual (misalnya dalam klasifikasi, contoh diinginkan untuk diberi nama, dan model dilatih untuk memprediksi nama pribadi dengan benar dari antara contoh). Generalisasi tersebut menghasilkan data latih dan data uji.

Meskipun algoritma optimalisasi mampu mengurangi kerugian pada saat data dilatih, pembelajaran mesin memiliki tujuan untuk meminimalkan kerugian pada sampel yang tidak terlihat.(Dönmez, 2013)

1.2.1 Kaitannya dengan statistik

Pembelajaran mesin dan statistik terkait erat. Menurut Michael I. Jordan, gagasan pembelajaran mesin, dari prinsip metodologis hingga alat teoretis, memiliki sejarah panjang dalam statistik. (Dönmez, 2013) Ia juga menyarankan untuk mengganti istilah ilmu data untuk menggantikan seluruh bidang. (Dönmez, 2013) Leo Breiman membedakan dua paradigma pemodelan statistik: Model data dan model algoritmik (Dönmez, 2013), di mana "model algoritma " berarti lebih banyak atau lebih sedikit algoritma pembelajaran mesin seperti hutan acak. Beberapa ahli statistik telah menggunakan metode pembelajaran mesin, menghasilkan bidang gabungan yang mereka sebut pembelajaran statistik. (Dönmez, 2013)

1.3 Teori

Tujuan utama siswa adalah menggeneralisasi pengalamannya. (Bharadwaj, Prakash and Kanagachidambaresan, 2021) (Dönmez, 2013) Generalisasi disini merujuk pada kemampuan *machine learning* untuk secara akurat memproses tugas baru yang tidak muncul setelah materi pelatihan. Contoh-contoh praktis biasanya diambil dari distribusi probabilitas yang tidak diketahui (yang harus mewakili ruang kejadian), dan siswa harus menciptakan sebuah model dari ruang tersebut yang dapat menghasilkan prediksi yang akurat dibuat pada kasus baru.

Analisis dan kinerja komputasi algoritma pembelajaran mesin adalah cabang ilmu komputer teoretis yang dapat disebut teori pembelajaran komputasi. Karena rangkaian pelatihan terbatas dan prediksi yang belum pasti, pembelajaran ini tidak menjamin fungsionalitas algoritma. Sebaliknya, batas daya probabilistik banyak digunakan. Dekomposisi bivariat merupakan salah satu cara untuk mengukur standard error.

Terlepas dari keterbatasan efisiensi, pakar teori pembelajaran komputer meneliti kelayakan dan kompleksitas

pembelajaran. Dalam teori belajar komputasi, suatu komputasi memungkinkan untuk dapat dilakukan dalam waktu polinomial. Terdapat dua jenis hasil kompleksitas waktu. Hasil negatif menandakan bahwa didalam waktu polinomial kelas yang diberikan tidak mampu dieksplorasi. Hasil Positif menandakan bahwa didalam waktu polinomial kelas yang diberikan mampu dieksplorasi

Meskipun istilah digunakan secara berbeda, ada banyak kesamaan antara teori pembelajaran mesin dan inferensi statistik.

Kinerja dan analisis komputasi algoritma pembelajaran mesin merupakan cabang ilmu komputer teoretis yang dapat disebut sebagai teori pembelajaran komputasi. Karena set pelatihan terbatas dan masa depan tidak pasti, teori pembelajaran biasanya tidak menjamin fungsionalitas algoritma. Sebaliknya, batas daya probabilistik banyak digunakan. Dekomposisi bivarians adalah salah satu cara untuk mengukur kesalahan standar.

Terlepas dari keterbatasan efisiensi, ahli teori pembelajaran komputer mempelajari kompleksitas dan kelayakan pembelajaran. Dalam teori belajar komputasi, suatu komputasi dianggap mungkin jika dapat dilakukan dalam waktu polinomial. Ada dua jenis hasil kompleksitas waktu. Hasil positif menunjukkan bahwa fungsi kelas tertentu dapat dipelajari dalam waktu polinomial. Hasil negatif menunjukkan bahwa kelas yang diberikan tidak dapat dieksplorasi dalam waktu polinomial.

Meskipun istilah digunakan secara berbeda, ada banyak kesamaan antara teori pembelajaran mesin dan inferensi statistik.

1.4 Pendekatan

1.4.1 Pembelajaran pohon keputusan

Pembelajaran pohon keputusan menggunakan pohon keputusan sebagai model prediktif yang memetakan pengamatan tentang suatu objek ke dalam inferensi tentang nilai target objek tersebut.

1.4.2 Pembelajaran aturan asosiasi

Mempelajari aturan asosiasi adalah salah satu cara untuk menemukan hubungan yang menarik antara variabel dalam database besar.

1.4.3 Jaringan syaraf tiruan

Algoritma pembelajaran jaringan saraf tiruan (JST), sering disebut sebagai "jaringan saraf" (NN), adalah algoritma pembelajaran yang telah dipengaruhi oleh struktur dan aspek fungsional jaringan saraf biologis. Pemrosesan data didasarkan pada kelompok neuron buatan yang saling berhubungan yang memproses informasi menggunakan pendekatan komputer koneksionis. Jaringan saraf modern adalah alat pemodelan data statistik nonlinier. Mereka biasanya digunakan untuk memodelkan hubungan kompleks antara input dan output, untuk mencari pola dalam data, atau untuk menangkap struktur statistik dalam distribusi probabilitas bersama yang tidak diketahui antara variabel yang diamati.

1.4.4 Pemrograman logika induktif

Pemrograman logika induktif (ILP) adalah teknik aturan pembelajaran yang memakai logika pemrograman sebagai representasi terpadu dari contoh input, informasi latar belakang, dan kesimpulan sementara. Mulai dari pengkodean informasi latar belakang yang diketahui dan serangkaian contoh yang disajikan sebagai basis data fakta logis, sistem ILP memperoleh program logika putatif yang berisi semua contoh positif tetapi tidak ada contoh negatif. Pemrograman induktif adalah bidang terkait yang menganggap bahasa pemrograman apa pun untuk mewakili hipotesis (bukan hanya pemrograman logika) sebagai pemrograman fungsional.

1.4.5 Support Vector Machines

Support Vector Machines (SVM) adalah metode pembelajaran yang terawasi (*Supervised Learning*) dapat digunakan pada kasus regresi dan klasifikasi. Berdasarkan data latih, dapat diberikan label dari satu atau beberapa kategori, algoritma ini menghasilkan model yang dapat melakukan sebuah prediksi, apakah data uji termasuk kedalam kategori yang mana.

1.4.6 Pengelompokan

Analisis cluster adalah pembagian sekumpulan observasi menjadi subset (disebut *cluster*) sedemikian rupa sehingga observasi dari kelompok yang memiliki banyak persamaan menurut beberapa kriteria atau kriteria yang memang sudah ditentukan, tetapi observasi dari kelompok yang tidak sama. Teknik clustering yang tidak sama memiliki asumsi yang tidak sama pula tentang struktur datanya, seringkali ditentukan oleh ukuran dan dianalisis berdasarkan : contoh, kohesi internal (kesamaan antara anggota kelompok yang memiliki persamaan kriteria) dan pemisahan antara cluster yang berbeda. Metode kedua didasarkan pada perkiraan kepadatan dan konektivitas grafik. Clustering adalah metode pembelajaran tanpa pengawasan dan teknik umum untuk menganalisis data statistik.

1.4.7 Jaringan Bayesian

Jaringan Bayesian, jaringan yang merepresentasikan *pattern*, atau model grafis asiklik terarah adalah model grafis probabilistik yang mewakili sekumpulan variabel acak dan independensi bersyaratnya melalui grafik asiklik terarah (DAG). Misalnya, jaringan Bayesian dapat merepresentasikan probabilitas antara penyakit dan gejala. Dengan mempertimbangkan gejalanya, jaringan bisa digunakan untuk melakukan perhitungan probabilitas banyak penyakit. Terdapat algoritma yang dapat menyimpulkan dan belajar.

1.4.8 Penguatan pembelajaran

Pembelajaran penguatan berurusan dengan bagaimana seorang agen harus bertindak dalam suatu lingkungan untuk memaksimalkan beberapa imbalan jangka panjang yang dirasakan. algoritma penguatan mencoba menemukan kebijakan yang mengaitkan negara bagian dunia dengan tindakan yang harus dilakukan oleh agen di negara bagian tersebut. Pembelajaran penguatan berbeda dari masalah pembelajaran yang diawasi karena pasangan input/output yang benar tidak pernah disajikan atau tindakan suboptimal dikoreksi secara eksplisit.

1.4.9 Pembelajaran representasi

Beberapa algoritma pembelajaran, sebagian besar algoritma pembelajaran tanpa pengawasan, memiliki tujuan untuk menghasilkan representasi yang lebih baik dari inputan pada saat data *training*. Contoh klasik adalah analisis komponen utama dan analisis kluster. Algoritma pembelajaran representasional sering mencoba untuk menyimpan informasi dalam input mereka tetapi memodifikasinya dengan cara yang membuatnya berguna, seringkali sebagai langkah pra-pemrosesan sebelum melakukan klasifikasi atau prediksi apa pun, sehingga input dapat direkonstruksi dari generator data yang tidak diketahui. Distribusi, meskipun mungkin tidak sesuai dengan konfigurasi, yang tidak masuk akal dalam distribusi ini.

Algoritma pembelajaran serbaguna mencoba ini dengan batasan bahwa representasi yang akan dipelajari adalah dimensi rendah. algoritma pengkodean jarang mencoba ini, dengan peringatan bahwa representasi yang akan dipelajari jarang (mengandung banyak angka nol). Algoritma pembelajaran subruang multilinear bertujuan untuk mempelajari representasi dimensi rendah langsung dari representasi tensor data multidimensi tanpa mengubahnya menjadi vektor (dimensi tinggi). algoritma pembelajaran mendalam menemukan tingkat

representasi yang berbeda, atau menghadirkan hierarki, menggunakan fitur tingkat lebih tinggi yang lebih abstrak yang ditentukan berdasarkan (atau pembuatan) fitur tingkat lebih rendah. Telah dikemukakan bahwa mesin cerdas adalah representasi pembelajaran mesin yang menganalisis faktor-faktor yang mendasari variasi yang menjelaskan data yang diamati. (Bengio, 2009)

1.4.10 Kemiripan dan pembelajaran metrik

Pada soal ini, learning machine diberikan pasangan contoh yang dianggap mirip dan pasangan benda yang kurang mirip. Kemudian perlu mempelajari fungsi kesamaan (atau fungsi metrik jarak) yang dapat memprediksi apakah objek baru serupa. Kadang-kadang digunakan dalam sistem Rekomendasi.

1.4.11 Pembelajaran kamus jarang

Dalam metode ini, sebuah datum direpresentasikan sebagai kombinasi linear dari fungsi basis, dan koefisiennya diasumsikan jarang. Biarkan x menjadi a d -dimensi datum, D menjadi a d by n matrix, di mana setiap kolom D mewakili fungsi basis. r adalah koefisien untuk mewakili x menggunakan D . Secara matematis, pembelajaran kamus jarang berarti mengikuti $x \approx Dr$ di mana r jarang. Secara umum, n diasumsikan lebih besar dari d untuk memungkinkan kebebasan untuk a representasi jarang.

Mempelajari kamus bersama dengan representasi yang jarang adalah NP-hard yang kuat dan juga sangat rumit untuk dipahami. (Tillmann, 2015) Metode heuristik yang banyak digunakan untuk mempelajari kamus dengan sedikit pembaca yaitu K-SVD.

Pembelajaran kamus jarang digunakan dalam berbagai konteks. Masalah dengan klasifikasi adalah menentukan kategori mana yang sebelumnya tidak terlihat milik data. Asumsikan kamus dibuat untuk setiap kelas. Data baru kemudian ditugaskan ke kelas

yang paling mewakili kamus yang sesuai. Pembelajaran kamus jarang juga telah diimplementasikan dalam denoising gambar. Bahwa lokasi gambar yang bersih dapat direpresentasikan secara jarang dalam kamus gambar, tetapi noise tidak bisa ini adalah ide utamanya. (Aharon, Elad and Bruckstein, 2006)

1.4.12 Algoritma genetika

Algoritma genetika ialah heuristik pencarian yang menyerupai langkah-langkah seleksi alam, menggunakan cara seperti mutasi dan persilangan untuk mendapatkan kebaruan genotipe dengan harapan dapat memecahkan masalah tertentu. algoritma genetik menemukan aplikasi dalam pembelajaran mesin pada 1980-an dan 1990-an. (Dönmez, 2013) Sebaliknya, teknik pembelajaran mesin telah digunakan untuk meningkatkan kinerja evolusioner serta algoritma genetika. (Zhang *et al.*, 2011)

1.5 Aplikasi

Aplikasi untuk pembelajaran mesin meliputi:

- Situs web adaptif
- Komputasi afektif
- Bioinformatika
- Antarmuka otak-mesin
- Cheminformatika
- Mengklasifikasi urutan DNA
- Iklan komputasional
- Keuangan komputasi
- Visi komputer, termasuk pengenalan objek
- Mendeteksi penipuan kartu kredit
- Bermain game
- Pencarian informasi
- Deteksi penipuan internet
- Persepsi mesin
- Diagnosa medis

- Pemrosesan bahasa alami
- Optimasi dan metaheuristik
- Sistem pemberi rekomendasi
- Penggerak robot
- Mesin pencari
- Analisis sentimen (atau penambangan opini)
- Penambangan berurutan
- Rekayasa Perangkat Lunak
- Pengenalan ucapan dan tulisan tangan
- Analisis pasar saham
- Pemantauan kesehatan struktural
- Pengenalan pola sintaksis

Pada tahun 2006, perusahaan film online Netflix menyelenggarakan sebuah lomba "Netflix Awards" pertamanya untuk mendapatkan program yang memiliki prediksi preferensi penonton dengan sangat baik dan memiliki nilai akurasi lebih tinggi menggunakan algoritma rekomendasi film Cinematch lebih besar dari 10%. AT&T Labs and Research bergabung dengan tim *Big Chaos dan Pragmatic Theory* sebagai satu tim riset untuk mengembangkan model entitas yang akan memenangkan hadiah utama \$1 juta tahun 2009. Beberapa bulan setelah penghargaan, Netflix memahami bahwa penayangan tidak dapat menjadi indikator terbaik dari kebiasaan menonton ("ini semua tentang rekomendasi"), dan mereka memodifikasi hasil riset mereka.

The Wall Street Journal pada tahun 2010 melaporkan tentang penggunaan pembelajaran mesin oleh perusahaan *Rebellion Research* untuk memprediksi pergerakan ekonomi. Artikel tersebut menjelaskan prediksi Rebellion Research tentang krisis keuangan dan pemulihan ekonomi. Algoritma pembelajaran mesin implementasikan pada studi lukisan seni rupa, dalam sejarah seni ini menemukan hasil yang sangat berdampak dalam

studi lukisan seni rupa yang sebelumnya tidak diketahui di antara para seniman ini dilaporkan pada tahun 2014.

1.6 Perangkat Lunak

Berikut ini adalah perangkat lunak yang memiliki algoritma pembelajaran mesin yang dapat dimanfaatkan:

1.6.1 Perangkat lunak *open source*

- dlib
- ELKI
- Encog
- H2O
- Mahout
- mlpy
- MLPACK
- MOA (Massive Online Analysis)
- ND4J with Deeplearning4j
- OpenCV
- OpenNN
- Orange
- R
- scikit-learn
- Shogun
- Torch (machine learning)
- Spark
- Yooreeka
- Weka

1.6.2 Perangkat lunak komersial dengan edisi Open Source

- KNIME
- RapidMiner

1.6.3 Perangkat lunak komersial

- Amazon Machine Learning
- Angoss KnowledgeSTUDIO
- Databricks
- IBM SPSS Modeler
- KXEN Modeler
- LIONsolver
- Mathematica
- MATLAB
- Microsoft Azure Machine Learning
- Neural Designer
- NeuroSolutions
- Oracle Data Mining
- RCASE
- SAS Enterprise Miner
- STATISTICA Data Miner

DAFTAR PUSTAKA

- Aharon, M., Elad, M. and Bruckstein, A. 2006. 'K-SVD : An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation', 54(11), pp. 4311–4322.
- Bengio, Y. 2009. *Learning deep architectures for AI, Foundations and Trends in Machine Learning*. Available at: <https://doi.org/10.1561/22000000006>.
- Bharadwaj, Prakash, K.B. and Kanagachidambaresan, G.R. (2021) *Pattern Recognition and Machine Learning, EAI/Springer Innovations in Communication and Computing*. Available at: https://doi.org/10.1007/978-3-030-57077-4_11.
- Dönmez, P. 2013. 'Introduction to Machine Learning, 2nd ed., by Ethem Alpaydın. Cambridge, MA: The MIT Press 2010. ISBN: 978-0-262-01243-0. \$54/£ 39.95 + 584 pages.', *Natural Language Engineering*, 19(2), pp. 285–288. Available at: <https://doi.org/10.1017/s1351324912000290>.
- Friedman, J.H. 1997. '2 What is Data Mining? 1 Introduction', *Statistics* [Preprint], (May).
- Goel, A.K. and Davies, J. 2019. *Artificial intelligence, The Cambridge Handbook of Intelligence*. Available at: <https://doi.org/10.1017/9781108770422.026>.
- Langley, P. 2011. 'The changing science of machine learning', *Machine Learning*, 82(3), pp. 275–279. Available at: <https://doi.org/10.1007/s10994-011-5242-y>.
- Mannila, H. 1996. 'Data mining: Machine learning, statistics, and databases', *Proceedings - 8th International Conference on Scientific and Statistical Data Base Management, SSDBM 1996*, pp. 2–8. Available at: <https://doi.org/10.1109/SSDM.1996.505910>.
- Shad, R. et al. 2021. 'Medical Imaging and Machine Learning', (July), pp. 25–38. Available at: <http://arxiv.org/abs/2103.01938>.

- Tillmann, A.M. 2015. 'On the computational intractability of exact and approximate dictionary learning', *IEEE Signal Processing Letters*, 22(1), pp. 45–49. Available at: <https://doi.org/10.1109/LSP.2014.2345761>.
- Zhang, J. *et al.* 2011. 'Evolutionary computation meets machine learning: A survey', *IEEE Computational Intelligence Magazine*, 6(4), pp. 68–75. Available at: <https://doi.org/10.1109/MCI.2011.942584>.

BAB 2

ALGORITMA MACHINE LEARNING

Oleh Wahyuddin S

2.1 Algoritma Machine Learning

Algoritma dalam *machine learning* dapat dikelompokkan berdasarkan masukan dan luaran yang diharapkan dari algoritma. Ada beberapa tipe algoritma dalam *machine learning* antara lain (Wikipedia, 2022a):

2.1.1 Supervised Learning

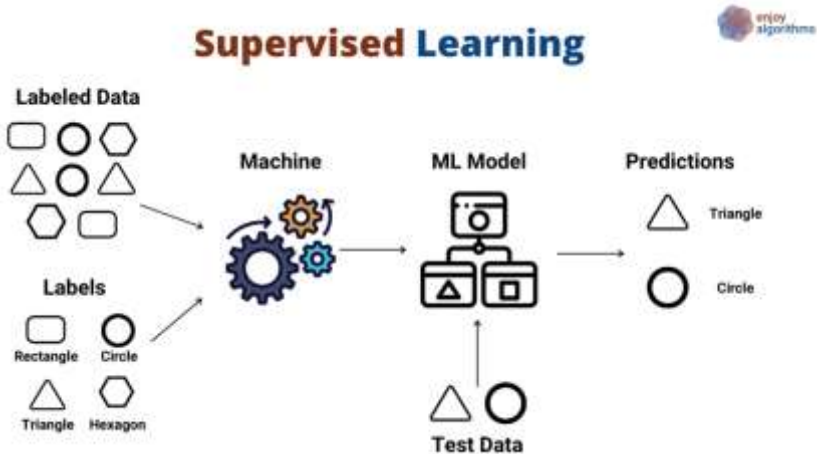
Supervised Learning atau pembelajaran terarah adalah sebuah pendekatan *machine learning* dengan menggunakan data yang telah diberi *tags* atau sebuah *dataset* yang telah diketahui oleh perancangannya. Tujuan dari metode ini adalah agar mesin dapat mengidentifikasi label *input* baru dengan menggunakan fungsi yang ada untuk membuat prediksi dan klasifikasi. Dengan mengamati data tersebut, metode ini dapat membuat model yang dapat memetakan masukan baru menjadi sebuah luaran yang sesuai (Wikipedia, 2022b). Adapun jenis dari *Supervised Learning* terdiri dari:

- Regresi

Teknik regresi dapat memprediksi nilai hasil tunggal dengan menggunakan data pelatihan. Analisis regresi merupakan salah satu analisis yang paling populer dan banyak digunakan. Analisis regresi umumnya digunakan untuk peramalan, dengan penggunaan yang saling melengkapi dengan bidang *machine learning*. Analisis ini juga dapat digunakan untuk memahami *variable* bebas mana saja yang berhubungan dengan *variable* terikat dan untuk mengetahui bentuk dari hubungan tersebut.

- **Klasifikasi**

Klasifikasi dapat mengelompokkan nilai dari luaran ke dalam kelas tertentu. jika algoritma mencoba memberikan *tags* masukan ke dalam dua kelas yang berbeda maka disebut klasifikasi biner. Memilih antara lebih dari dua kelas yang berbeda disebut sebagai klasifikasi *multi* kelas.



Gambar 2.1 : Supervised Learning

Sumber: <https://www.enjoyalgorithms.com/blogs/supervised-unsupervised-and-semisupervised-learning>

2.1.2 Unsupervised Learning

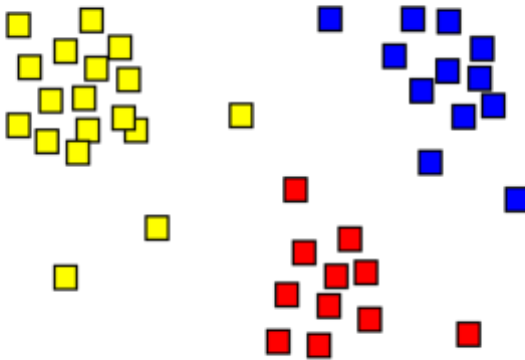
Unsupervised Learning atau pembelajaran tidak terarah dapat memodelkan sebuah himpunan masukan, seperti penggolongan (*clustering*). *Unsupervised Learning* merupakan teknik pembelajaran *machine learning* dimana perlu adanya pengawasan terhadap modelnya serta mengizinkan model tersebut dapat bekerja sendiri guna untuk menemukan informasi yang dibutuhkan. Hal ini terutama yang berkaitan dengan data tanpa label. Algoritma *Unsupervised Learning* dapat memungkinkan melakukan tugas pemrosesan yang lebih kompleks dibandingkan dengan *Supervised Learning*. Pembelajaran tidak terarah bisa lebih

tidak terduga dibandingkan dengan pembelajaran alami lainnya dalam pembelajaran dan metode *Reinforcement Learning*. Algoritma ini tidak memiliki *variable* target. Adapun tujuan dari metode ini adalah untuk mempelajari serta dapat mencari pola yang menarik pada masukan yang telah diberikan, meskipun tidak disediakan luaran yang tepat secara eksplisit tujuan lain dari algoritma ini yaitu untuk mengelompokkan objek yang serupa dalam suatu area tertentu. *Unsupervised Learning* dapat menemukan semua jenis pola yang tidak diketahui dalam sebuah data serta dapat membantu untuk menemukan fitur yang dapat berguna untuk pengelompokan. Pengelompokan ini dapat dilakukan secara *real-time*, sehingga semua data masukan dapat diurai dan diberi *tag* (Abijono, Santoso, & Anggreini, 2021).

Pada umumnya *Unsupervised Learning* digunakan untuk tiga hal utama yaitu:

- *Clustering*

Clustering atau pengelompokan merupakan suatu teknik yang dapat digunakan untuk mengidentifikasi kelompok yang dihasilkan dari pengelompokan item yang lebih kecil berdasarkan kesamaannya. Kemiripan yang menjadi dasar pengelompokan tidak bersifat universal, sehingga peneliti atau peng analisis harus lebih dahulu menggambarkan kesamaan tersebut.



Gambar 2.2 : Hasil Analisis *Clustering*

Sumber: https://id.wikipedia.org/wiki/Analisis_kelompok

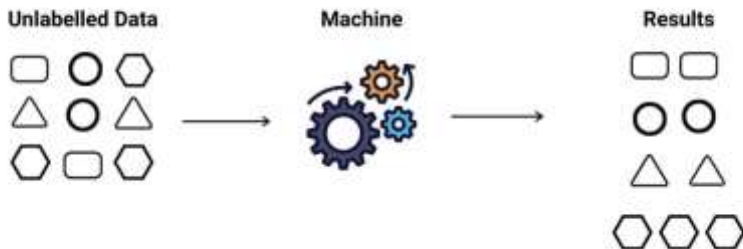
- ***Association Rule***

Algoritma *Association rule* (aturan asosiasi) merupakan algoritma yang menemukan atribut yang muncul secara bersamaan. *Association* di kenal juga dengan istilah analisis keranjang pasar dimana berfungsi untuk mengidentifikasi item-item produk yang kemungkinan dapat dibeli oleh konsumen bersamaan dengan produk lainnya. *Association Rule* adalah teknik yang berguna untuk menemukan hubungan antar objek dalam kumpulan data tertentu, terdapat dua tahapan yaitu menemukan kombinasi yang sering muncul dan menentukan kondisi. Untuk menerapkan metode *association rule* umumnya digunakan saat mencari hubungan antar elemen dalam kumpulan data tertentu. hal ini dapat memudahkan untuk menganalisis proses pencarian aturan dalam suatu objek, karena aturan yang dibentuk berisi banyak elemen dan oleh karena itu harus lebih sesuai dengan sebuah tingkatannya.

- *Dimensionality Reduction*

Dimensionality Reduction atau reduksi dimensi merupakan teknik untuk mereduksi dimensi suatu kumpulan data, dalam hal ini karakteristik data. Biasanya, dataset yang ingin diproses memiliki puluhan bahkan ratusan fitur atau kolom. Reduksi dimensi secara prinsip/konsep sama dengan mengompres *file* berukuran besar menjadi *file* ZIP. Kompresi *file* tidak menghilangkan atau mengurangi informasi di dalam *file*, hanya menyederhanakan *file* untuk memperkecil ukuran *file*, yang dapat mempercepat proses transfer *file*. Reduksi dimensi diperlukan karena banyaknya *variable* input yang dapat menurunkan performa *machine learning*. Dataset yang digunakan pada umumnya diwakili oleh baris dan kolom (Johnson, 2022).

Unsupervised Learning



Gambar 2.3 : Supervised Learning

Sumber: <https://www.enjoyalgorithms.com/blogs/supervised-unsupervised-and-semisupervised-learning>

2.1.3 Semi-supervised Learning

Semi-supervised Learning atau pembelajaran semi terarah merupakan tipe yang menggabungkan antara *supervised* dan *unsupervised* untuk menghasilkan suatu fungsi. *Semi-supervised learning* dapat juga diartikan sebagai salah satu jenis *machine learning* dengan melibatkan data dalam jumlah yang kecil hingga data yang sangat besar, baik data dengan label maupun tanpa label. Berikut contoh dari *semi-supervised learning* antara lain:

- *Speech Recognition*

Speech Recognition atau *speech-to-text* merupakan kemampuan dari mesin atau program untuk mengidentifikasi kata dan frasa yang diucapkan oleh manusia kemudian mengubahnya menjadi teks yang dapat dibaca. Setelah kata telah di olah menjadi format yang dapat dibaca oleh mesin aplikasi, *speech recognition* akan menganalisa perintah yang masuk dan respon apa yang harus di keluarkan oleh mesin atau aplikasi. *Speech recognition* merupakan teknologi perpaduan antara ilmu komputer, *linguistic* dan teknik komputer.

- *Web Content Classification*

Web Content Classification merupakan salah satu bentuk variasi dari *semi-supervised learning* bertujuan untuk dapat mengatasi penyimpanan data yang besar pada sebuah website dan mengurangi waktu proses klasifikasinya secara cepat dan efisien untuk meningkatkan pengalaman dari pengguna. Beberapa mesin pencarian seperti *google*, menggunakan SSL (*Secure Socket Layer*). SSL merupakan salah satu komponen penting yang harus dimiliki sebuah website untuk menciptakan koneksi yang lebih aman antara *website* dan *browser*. SSL digunakan untuk memudahkan pencarian yang berdasarkan pada Bahasa sehari-hari serta relevan. Selain itu, *google search*

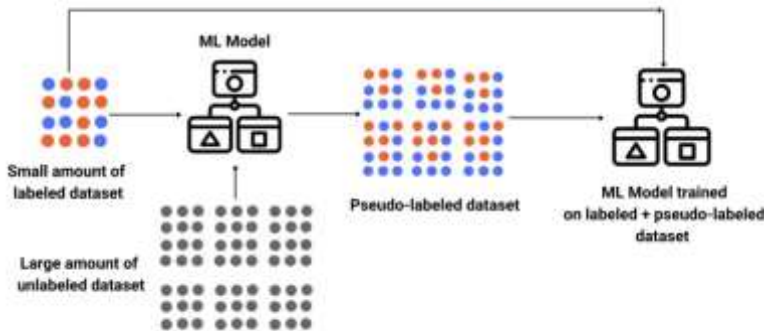
menggunakan SSL untuk menemukan konten yang relevan dengan apa yang dicari oleh pengguna.

▪ *The Document Classification*

The document classification atau klasifikasi dokumen merupakan sebuah proses untuk menangani munculnya masalah sederhana yang berupa bertambahnya jumlah dokumen setiap hari. Dokumen klasifikasi dapat berupa teks, gambar, musik dan lain sebagainya. Dokumen juga dapat dikategorikan berdasarkan dengan subjek atau atribut lain seperti jenis dokumen, pengarang, tahun cetakan dll. Terdapat dua pendekatan klasifikasi dokumen yaitu klasifikasi berbasis konten dan klasifikasi berbasis permintaan (pengindeksan). Adapun teknik klasifikasi telah diterapkan pada:

- Perutean *email* pengiriman *email* yang dikirim ke alamat atau kotak surat berdasarkan identifikasi subjek.
- Identifikasi bahasa, secara otomatis menentukan Bahasa dari teks
- Sentimen analisis, menentukan sikap pembicara atau penulis terhadap berbagai topik atau polaritas kontekstual seluruh dokumen.
- Peringkat keterbacaan, secara otomatis menilai tingkat keterbacaan teks, baik untuk menemukan materi yang sesuai untuk berbagai usia atau jenis pembaca, atau sebagai bagian dari *system* penyederhanaan teks yang lebih luas.

Semi-supervised learning use-case

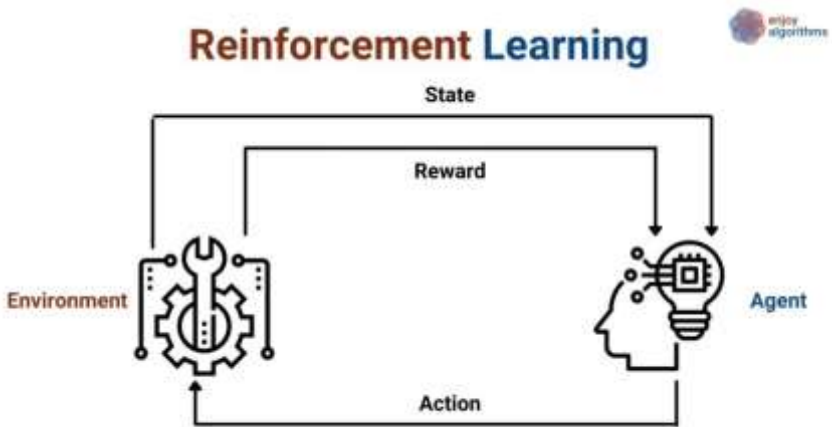


Gambar 2.4 : Semi-supervised Learning

Sumber: <https://www.enjoyalgorithms.com/blogs/supervised-unsupervised-and-semisupervised-learning>

2.1.4 Reinforcement Learning

Reinforcement learning merupakan tipe algoritma *machine learning* yang membuat agent *software* dan mesin dapat bekerja secara otomatis untuk menentukan perilaku yang ideal sehingga dapat memaksimalkan kinerja algoritmanya. Cara kerja dari *reinforcement learning* berbeda dengan *supervised learning* yang memiliki jawaban pasti. Dalam metode ini, mesin direkayasa untuk menjalankan perintah berdasarkan situasi yang dihadapinya disebagikan oleh tidak ada jawaban benar yang pasti, mesin dapat belajar dari pengalaman sebelumnya untuk menghindari kesalahan (Algorithms, 2022).



Gambar 2.5 : Reinforcement Learning

Sumber: <https://www.enjoyalgorithms.com/blogs/supervised-unsupervised-and-semisupervised-learning>

2.1.5 Developmental Learning Algorithm

Developmental learning algorithm atau pembelajaran berkembang merupakan sebuah bidang yang bertujuan untuk mempelajari mekanisme pengembangan, arsitektur, dan batasan yang memungkinkan untuk mengembangkan metode pembelajaran yang dapat diterapkan dalam kehidupan serta bersifat terbuka terhadap kemampuan dan pengetahuan untuk dipasangkan ke mesin (Santoso, 2022).

DAFTAR PUSTAKA

- Abijono, H., Santoso, P., & Anggreini, N. L. 2021. Algoritma Supervised Learning Dan Unsupervised Learning Dalam Pengolahan Data. *Jurnal Teknologi Terapan: G-Tech*, 4(2), 315–318. <https://doi.org/10.33379/gtech.v4i2.635>
- Algorithms, E. 2022. Supervised, Unsupervised and Semi-supervised Learning. Retrieved December 14, 2022, from <https://www.enjoyalgorithms.com/blogs/supervised-unsupervised-and-semisupervised-learning>
- Johnson, D. 2022. Supervised vs Unsupervised Learning: Difference Between Them. Retrieved December 19, 2022, from <https://www.guru99.com/supervised-vs-unsupervised-learning.html>
- Santoso, J. T. 2022. *Algoritma Machine Learning* (1st ed.). Informatika.
- Wikipedia. 2022a. Pembelajaran mesin - Wikipedia bahasa Indonesia, ensiklopedia bebas. Retrieved December 15, 2022, from Wikipedia website: https://id.wikipedia.org/wiki/Pembelajaran_mesin
- Wikipedia. 2022b. Pemelajaran terarah - Wikipedia bahasa Indonesia, ensiklopedia bebas. Retrieved December 11, 2022, from Wikipedia website: https://id.wikipedia.org/wiki/Pemelajaran_terarah

BAB 3

LINEAR REGRESSION

Oleh Leo Willyanto Santoso

3.1 Pendahuluan

Machine learning dapat diimplementasikan di berbagai bidang untuk memecahkan masalah yang kompleks yang tidak dapat diselesaikan dengan mudah berdasarkan pendekatan komputer. Regresi linier/*linear regression* adalah salah satu algoritma *machine learning* yang paling sederhana dan paling umum. Regresi linier tergolong metode pembelajaran tersupervisi (*supervised learning*). Metode ini menggunakan pendekatan matematika untuk melakukan analisis prediktif.

Regresi linier umumnya digunakan dalam metode penelitian, di mana dimungkinkan untuk mengukur dampak yang diprediksi dan memodelkannya terhadap beberapa variabel masukan. Metode ini adalah metode evaluasi dan pemodelan data yang menetapkan hubungan linier antara variabel yang dependen dan independen.

3.2 Model Regresi Linier

Regresi linier adalah metode yang dapat digunakan untuk dua hal. Pertama, analisis regresi biasanya digunakan untuk peramalan dan prediksi, di mana penerapannya terkait *machine learning*. Kedua, analisis regresi dapat digunakan dalam beberapa kasus untuk menentukan hubungan kausal antara variabel independen dan dependen.

Berdasarkan model regresi, variabel independen memprediksi variabel dependen. Analisis regresi memperkirakan nilai variabel 'y' dependen berdasarkan nilai variabel independen

'x'. Regresi linier dapat berupa regresi linier sederhana atau regresi berganda.

3.2.1 *Simple Linear Regression/Regresi Linier Sederhana*

Regresi Linier Sederhana (RLS) adalah metode statistik dengan satu variabel independen, yang berupaya memodelkan hubungan antara dua peubah acak dimana satu peubah acak mempengaruhi peubah acak lainnya. Kata 'sederhana' dalam RLS menunjukkan bahwa dalam model regresi yang terbentuk hanya melibatkan satu variabel bebas (x) dan satu variabel terikat (y).

Regresi Linear Sederhana dimodelkan dengan persamaan:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

Dimana:

- ε = Nilai error/kesalahan pengukuran model regresi
- β_0, β_1 = koefisien regresi linier sederhana
- y = variabel terikat (*dependen*)
- x = variabel bebas (*independen*)

3.2.2 *Multiple Linear Regression/Regresi Linier Berganda*

Regresi Linier Berganda terbentuk apabila jumlah variabel bebas/*independen* dari suatu persamaan regresi lebih dari satu ($x_1, x_2, \dots, x_k$) dan semuanya mempengaruhi variabel terikat/*dependen* (y).

Regresi Linier Berganda dimodelkan dengan persamaan:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k + \varepsilon$$

Dimana:

- ε = Nilai error/kesalahan pengukuran model regresi
- $\beta_0, \beta_1, \dots, \beta_k$ = koefisien regresi linier berganda
- y = variabel terikat (*dependen*)
- $x_1, x_2, \dots, x_k$ = variabel bebas (*independen*)

3.3 Regresi Linier dengan Python

Sekarang Anda akan mulai menerapkan regresi linier dengan Python. Untuk melakukan ini, Anda perlu menggunakan *package* yang tepat beserta fungsi dan *class*-nya.

NumPy adalah *package* Python yang memungkinkan Anda untuk melakukan komputasi berkinerja tinggi pada array satu dimensi dan multidimensi. Selain itu, *package* ini juga menawarkan banyak operasi matematika.

Package *scikit-learn* adalah *library* Python yang banyak digunakan untuk *machine learning*. *Package* ini menyediakan sarana untuk *preprocessing* data, mengurangi dimensi, menerapkan regresi, mengklasifikasikan, *clustering*, dan banyak lagi.

Jika Anda ingin menerapkan regresi linier dan memerlukan fungsionalitas di luar cakupan *scikit-learn*, Anda harus menggunakan *statsmodels*. Ini adalah paket *Python* yang handal untuk estimasi model statistik, melakukan pengujian, dan banyak lagi.

Sekarang, untuk mengikuti tutorial ini, Anda harus menginstal semua *package* yang dibutuhkan.

```
Shell
(venv) $ python -m pip install numpy scikit-learn statsmodels
```

Berikut ini adalah langkah dasar saat Anda menerapkan regresi linier dengan Python:

1. Impor *package* dan *class* yang Anda butuhkan.
2. Sediakan data dan lakukan transformasi yang sesuai.
3. Buat model regresi dan sesuaikan dengan data yang ada.
4. Periksa hasil fitting model untuk mengetahui apakah model tersebut memuaskan.
5. Terapkan model untuk prediksi.

Langkah-langkah tersebut bersifat umum untuk sebagian besar pendekatan dan implementasi metode regresi.

Langkah 1: Impor *package* dan *class*

Langkah pertama adalah mengimpor *package* **numpy** dan *class* `LinearRegression` dari `sklearn.linear_model`:

```
Python
```

```
>>> import numpy as np
>>> from sklearn.linear_model import LinearRegression
```

Sekarang, Anda memiliki semua fungsi yang Anda perlukan untuk mengimplementasikan regresi linier.

Tipe data **NumPy** adalah tipe array yang disebut `numpy.ndarray`. Anda akan menggunakan kelas `sklearn.linear_model.LinearRegression` untuk melakukan regresi linier dan membuat prediksi yang sesuai.

Langkah 2: Menyediakan data

Langkah kedua adalah mempersiapkan data untuk dikerjakan. Input (*regressor*, x) dan output (*respons*, y) harus berupa array atau objek serupa.

```
Python
```

```
>>> x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
>>> y = np.array([5, 20, 14, 32, 22, 38])
```

Sekarang, Anda memiliki dua array: input, x , dan output, y . Anda harus memanggil `.reshape()` pada x karena array ini harus dua dimensi, atau lebih tepatnya, ia harus memiliki satu kolom dan baris sebanyak yang diperlukan. Itulah yang dilakukan oleh argumen `(-1, 1)` dari `.reshape()`. Inilah tampilan x dan y sekarang:

```

Python
>>> x
array([[ 5],
       [15],
       [25],
       [35],
       [45],
       [55]])

>>> y
array([ 5, 20, 14, 32, 22, 38])

```

Seperti yang Anda lihat, x memiliki dua dimensi, dan $x.shape$ adalah (6, 1), sedangkan y memiliki dimensi tunggal, dan $y.shape$ adalah (6,).

Langkah 3: Membuat model dan Menyesuaikan

Langkah selanjutnya adalah membuat model regresi linier dan menyesuaikannya menggunakan data yang ada.

Buat instance *class* `LinearRegression`, yang akan mewakili model regresi:

```

Python
>>> model = LinearRegression()

```

Perintah diatas akan membuat variabel bernama `model` sebagai turunan dari `LinearRegression`. Anda dapat memberikan beberapa parameter opsional ke `LinearRegression`, yaitu:

1. `fit_intercept` bertipe *Boolean*, jika bernilai *True*, memutuskan untuk menghitung *intercept* b_0 atau, jika *False*, menganggapnya sama dengan nol. Nilai *default*-nya adalah *True*.
2. `normalize` bertipe *Boolean*, jika bernilai *True*, memutuskan untuk menormalkan variabel input. Nilai *default*-nya adalah *False*, dalam hal ini tidak menormalkan variabel input.

3. `copy_X` bertipe *Boolean* yang memutuskan apakah akan melakukan *copy* (*True*) atau melakukan *overwrite* variabel masukan (*False*). Nilai *default*-nya adalah *True*.
4. `n_jobs` bertipe *integer* atau *None*. Ini mewakili jumlah pekerjaan yang digunakan dalam komputasi paralel. Nilai *default*-nya adalah *None*, yang biasanya berarti satu pekerjaan. -1 berarti menggunakan semua prosesor yang tersedia.

Model Anda seperti yang didefinisikan di atas menggunakan nilai default dari semua parameter.

Saatnya untuk mulai menggunakan model. Pertama, Anda perlu memanggil `.fit()` pada model:

```
Python
```

```
>>> model.fit(x, y)
LinearRegression()
```

Dengan `.fit()`, Anda menghitung nilai optimal bobot b_0 dan b_1 , menggunakan input dan output yang ada, x dan y , sebagai argumen. Dengan kata lain, `.fit()` cocok dengan modelnya. Anda dapat menyederhanakan program dengan perintah berikut ini, sehingga lebih pendek.

```
Python
```

```
>>> model = LinearRegression().fit(x, y)
```

Pernyataan ini melakukan hal yang sama seperti dua perintah sebelumnya.

Langkah 4: Dapatkan hasil

Setelah model Anda dipasang, Anda bisa mendapatkan hasil untuk memeriksa apakah model bekerja dengan memuaskan dan menafsirkannya.

Anda dapat memperoleh koefisien determinasi, R^2 , dengan `.score()` disebut pada model:

```

Python
>>> r_sq = model.score(x, y)
>>> print(f"coefficient of determination: {r_sq}")
coefficient of determination: 0.7158756137479542

```

Saat Anda menerapkan `.score()`, argumennya juga merupakan prediktor x dan respons y , dan nilai kembaliannya adalah R^2 .

Atribut model adalah `.intercept_` yang merepresentasikan koefisien b_0 , dan `.coef_` yang merepresentasikan b_1 :

```

Python
>>> print(f"intercept: {model.intercept_}")
intercept: 5.633333333333329

>>> print(f"slope: {model.coef_}")
slope: [0.54]

```

Program di atas menjelaskan cara mendapatkan nilai b_0 dan b_1 . Anda dapat melihat bahwa `.intercept_` adalah skalar, sedangkan `.coef_` adalah array.

Nilai b_0 adalah sekitar 5,63. Ini mengilustrasikan bahwa model Anda memprediksi respons 5,63 ketika x adalah nol. Nilai $b_1 = 0,54$ berarti respon prediksi naik 0,54 ketika x dinaikkan satu. Anda juga dapat mengubah y sebagai array dua dimensi. Dalam hal ini, Anda akan mendapatkan hasil yang serupa. Berikut perintah program di Python:

```

Python
>>> new_model = LinearRegression().fit(x, y.reshape((-1, 1)))
>>> print(f"intercept: {new_model.intercept_}")
intercept: [5.63333333]

>>> print(f"slope: {new_model.coef_}")
slope: [[0.54]]

```

Seperti yang Anda lihat, contoh ini sangat mirip dengan contoh sebelumnya, tetapi dalam kasus ini, `.intercept_` adalah array satu dimensi dengan elemen tunggal b_0 , dan `.coef_` adalah array dua dimensi dengan elemen tunggal b_1 .

Langkah 5: Memprediksi respons

Setelah Anda memiliki model yang memuaskan, Anda dapat menggunakannya untuk prediksi dengan data yang ada atau yang baru. Untuk mendapatkan hasil yang diprediksi, gunakan `.predict()`:

```
Python
```

```
>>> y_pred = model.predict(x)
>>> print(f"predicted response:\n{y_pred}")
predicted response:
[ 8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333]
```

Saat menggunakan `.predict()`, Anda akan mengirimkan nilai regressor sebagai argumen dan mendapatkan hasil prediksi yang sesuai. Ini adalah cara yang hampir identik untuk memprediksi hasil:

```
Python
```

```
>>> y_pred = model.intercept_ + model.coef_ * x
>>> print(f"predicted response:\n{y_pred}")
predicted response:
[[ 8.33333333]
 [13.73333333]
 [19.13333333]
 [24.53333333]
 [29.93333333]
 [35.33333333]]
```

Dalam hal ini, Anda mengalikan setiap elemen `x` dengan `model.coef_` dan menambahkan `model.intercept_` ke hasil kali.

Output di sini berbeda dari contoh sebelumnya hanya dalam hal dimensi. Hasil yang diprediksi sekarang menjadi array dua dimensi, sedangkan dalam kasus sebelumnya, ia memiliki satu dimensi.

Jika Anda mengurangi jumlah dimensi `x` menjadi satu, maka kedua pendekatan ini akan menghasilkan hasil yang sama. Anda dapat melakukannya dengan mengganti `x` dengan `x.reshape(-1)`, `x.flatten()`, atau `x.ravel()` saat mengalikannya dengan `model.coef_`.

Dalam praktiknya, model regresi sering diterapkan untuk peramalan. Ini berarti Anda dapat menggunakan model yang sesuai untuk menghitung output berdasarkan input yang baru:

```
Python

>>> x_new = np.arange(5).reshape((-1, 1))
>>> x_new
array([[0],
       [1],
       [2],
       [3],
       [4]])

>>> y_new = model.predict(x_new)
>>> y_new
array([5.63333333, 6.17333333, 6.71333333, 7.25333333, 7.79333333])
```

Di sini `.predict()` diterapkan pada regressor baru `x_new` dan menghasilkan hasil `y_new`. Pada contoh ini menggunakan `arange()` dari **numpy** untuk menghasilkan array dengan elemen dari 0, inklusif, hingga tetapi tidak termasuk 5. Jadi nilainya yaitu: 0, 1, 2, 3, dan 4.

DAFTAR PUSTAKA

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
- Maulud, D.H. dan Abdulazeez, A.M. (2020). A Review on Linear Regression Comprehensive in Machine Learning. Journal of Applied Science and Technology Trends. Vol. 01, No. 04, pp. 140 –147. 2020. doi:10.38094/jastt1457
- Santoso, Leo Willyanto and Yulia. 2020. Predicting student performance in higher education using multi-regression models. TELKOMNIKA (Telecommunication, Computing, Electronics and Control), Vol. 18, No. 3, pp. 14802 – 14808.

BAB 4

KNN (K-Nearest Neighbor)

Oleh Gentur Wahyu Nyipto Wibowo

4.1 Pendahuluan

Algoritma Klasifikasi bertujuan untuk prediksi kelas baru dari kumpulan *dataset* yang telah memiliki kelas (Larose and Larose, 2014). Dua pekerjaan utama yang dilakukan oleh algoritma klasifikasi yaitu (1) membangun pemodelan yang dijadikan sebagai prototipe untuk disimpan sebagai memori dan (2) model tersebut digunakan untuk melakukan pengenalan/klasifikasi/prediksi pada dataset untuk diketahui pada kelas mana objek data tersebut dalam model yang sudah disimpannya (Prasetyo, Alim and Rosyid, 2014).

Pada algoritma klasifikasi dikelompokkan menjadi dua macam, yaitu *eager learner* dan *lazy learner*. *Eager learner* membuat model klasifikasi berdasarkan data latih untuk menghasilkan model yang akan digunakan oleh data uji untuk proses prediksi. Sedangkan *lazy learner* menyimpan data latih dan menunggu sampai data uji muncul. Algoritma *Artificial Neural Network* (ANN), *Support Vector Machine* (SVM), *Decision Tree*, *Naive Bayes* masuk ke dalam kategori *eager learner*. Sedangkan yang termasuk ke dalam *lazy learner* diantaranya, *K-Nearest Neighbor* (KNN), *Fuzzy K-Nearest Neighbor*, *Regresi Linear*. (Prasetyo, Alim and Rosyid, 2014)

4.2 Algoritma K-Nearest Neighbor

Algoritma K-Nearest Neighbor adalah suatu metode algoritma klasifikasi yang bekerja berdasarkan tingkat kemiripan yang dihitung berdasarkan jarak (*distance*) terdekat dari data

pembelajarannya (data latih dan data uji) (Boiculescu, Dimitriu and Moscalu, 2013).

Metode yang paling banyak digunakan peneliti untuk menghitung jarak pada algoritma K-NN adalah metode *Euclidean distance* (Liu, 2012). Metode Euclidean distance secara umum digunakan untuk data numerik. Untuk rumus perhitungan Euclidean distance ada pada perhitungan 4.1.

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Rumus *Euclidean Distance*

Keterangan :

$d(x, y)$ = jarak kedekatan antara data x ke data y

x_i = data *testing* (data uji) ke i

y_i = data *training* (data latih) ke i

n = jumlah atribut 1 sampai n

Langkah-langkah algoritma KNN sebagai berikut :

1. Tentukan nilai parameter K (penentuan nilai k dipilih secara manual)
2. Hitung jarak antara data *training* (data latih) dan data *testing* (data uji)
3. Urutkan data *training* berdasarkan jarak yang terbentuk (dari terkecil ke terbesar)
4. Tetapkan kelas yang bersesuaian, pilihlah kelas dengan jumlah nilai k terbanyak pada data *testing* (data uji)

4.2.1 Kelebihan dan Kekurangan Algoritma KNN

Selain memiliki kelebihan, algoritma KNN juga memiliki kekurangan (Kumar, Chhabra and Garg, 2018).

Kelebihan algoritma KNN :

1. KNN tidak memerlukan *training* sebelum prediksi, sehingga penambahan data baru dapat dilakukan secara mudah tanpa mengurangi keakuratannya.
2. KNN termasuk ke *Lazy Learner* sehingga KNN lebih cepat dibandingkan algoritma lainnya.
3. KNN sangat mudah diimplementasikan, karena hanya menggunakan dua parameter, yaitu nilai K dan fungsi jarak.

Kekurangan algoritma KNN :

1. Tidak mampu menangani data yang besar, karena dengan data yang besar performa kinerja algoritma menurun yang disebabkan penghitungan jarak antara titik baru dengan yang ada lama.
2. Tidak dapat bekerja dengan baik ketika menangani dimensi yang tinggi. Karena dengan dimensi yang tinggi dan besar, algoritma KNN akan sulit untuk menghitung jarak setiap dimensinya.
3. Diperlukannya standarisasi dan normalisasi sebelum penerapan algoritma KNN ke *dataset*. Jika tidak melakukannya algoritma KNN akan menghasilkan prediksi yang salah.
4. Algoritma KNN sangat sensitif terhadap *noise dataset*, sehingga diperlukan penghitungan secara manual terhadap nilai yang hilang secara manual.

4.2.2 Penghitungan Manual Algoritma K-Nearest Neighbor

Sebelumnya dataset dibagi menjadi dua, yaitu sebagai data *training* (data latih) dan data *testing* (data uji). Yang dimaksud dengan data *training* (data latih) adalah data yang sudah memiliki

kelas, sedangkan data *testing* (data uji) merupakan data yang akan dicari kelasnya. Data *training* (data latih) berperan sebagai pembentuk suatu pola/model/pengetahuan, dan data *testing* (data uji) sebagai alat ukur evaluasi algoritma.

Pada penghitungan manual ini data yang digunakan merupakan dataset palsu/*dummy*. Untuk *dataset* dapat dilihat pada tabel 4.1 yang digunakan sebagai data *training* (data latih), dan tabel 4.2 dijadikan sebagai data *testing* (data uji).

Tabel 4.1 : Data Training (Data Latih)

A	B	Kategori
8	7	Buruk
7	7	Buruk
7	6	Buruk
2	4	Baik
3	5	Baik
3	3	Baik

Tabel 4.2 : Data Testing (Data Uji)

A	B	Kategori
4	6	?

Langkah penyelesaiannya

Pertama, tentukan parameter K. Misalnya dibuat jumlah tetangga terdekat $K = 3$

Kedua, hitung jarak antara data baru dengan semua data training. Pada Langkah kedua ini menggunakan metode *Euclidean Distance*.

Tabel 4.3 : Hitung Menggunakan *Euclidean Distance*

A	B	Euclidean Distance
8	7	$\sqrt{(8-4)^2 + (7-6)^2} = \sqrt{(4)^2 + (1)^2} = \sqrt{17} = 4,12$
7	7	$\sqrt{(7-4)^2 + (7-6)^2} = \sqrt{(3)^2 + (1)^2} = \sqrt{10} = 3,16$
7	6	$\sqrt{(7-4)^2 + (6-6)^2} = \sqrt{(3)^2 + (0)^2} = \sqrt{9} = 3$
2	4	$\sqrt{(2-4)^2 + (4-6)^2} = \sqrt{(-2)^2 + (-2)^2} = \sqrt{8} = 2,82$
3	5	$\sqrt{(3-4)^2 + (5-6)^2} = \sqrt{(-1)^2 + (-1)^2} = \sqrt{2} = 1,41$
3	3	$\sqrt{(3-4)^2 + (3-6)^2} = \sqrt{(-1)^2 + (-3)^2} = \sqrt{10} = 3,16$

Langkah ketiga, urutkan jarak dari data baru dengan data training untuk menentukan tetangga terdekat berdasarkan jarak minimum K.

Tabel 4.4 : Hasil Hitung Menggunakan *Euclidean Distance*

A	B	Euclidean Distance	Urutan Jarak	Termasuk
8	7	4,12	5	Tidak ($K > 3$)
7	7	3,16	4	Tidak ($K > 3$)
7	6	3	3	Ya ($K = 3$)
2	4	2,82	2	Ya ($K < 3$)
3	5	1,41	1	Ya ($K < 3$)
3	3	3,16	4	Tidak ($K > 3$)

Berdasarkan kolom urutan jarak dari tabel 4.4 kemudian diurutkan jaraknya antara data baru dengan data *training*, dari yang terdekat ke terjauh. Pada tabel tersebut terdapat data

dengan jarak yang sama dengan nilai 4, terletak pada baris ke 2 dan baris ke 6, sehingga dianggap memiliki urutan yang sama. Dan pada kolom Termasuk disesuaikan dengan nilai $K=3$

Langkah keempat, menentukan kategori berdasarkan dari tetangga terdekat. Perhatikan baris 3, 4, dan 5 pada tabel 4.3. Kategori Ya diambil jika $K \leq 3$. Sehingga baris 3, 4, dan 5 masuk kategori Ya, sedangkan yang tersisa masuk kategori Tidak.

Tabel 4.5 : Penentuan Kategori

A	B	Euclidean Distance	Urutan Jarak	Termasuk	Kategori
8	7	4,12	5	Tidak ($K > 3$)	-
7	7	3,16	4	Tidak ($K > 3$)	-
7	6	3	3	Ya ($K = 3$)	Buruk
2	4	2,82	2	Ya ($K < 3$)	Baik
3	5	1,41	1	Ya ($K < 3$)	Baik
3	3	3,16	4	Tidak ($K > 3$)	-

Berdasarkan tabel 4.5 kategori Ya berada pada baris 3, 4, dan 5 dengan kategori buruk, baik, baik.

Kelima, tentukan dari tetangga terdekat yang memiliki kategori mayoritas untuk nilai prediksi data yang baru. Jika merujuk pada tabel 4.4 data yang dimiliki pada baris 3, 4, dan 5 yaitu 2 kategori bernilai Baik dan 1 kategori bernilai Buruk. Dari hasil tersebut dapat disimpulkan nilai mayoritas pada kategori Baik, sehingga data baru masuk kategori Baik.

Tabel 4.6 : Hasil Klasifikasi Berdasarkan Kategori Mayoritas

A	B	Kategori
8	7	Buruk
7	7	Buruk
7	6	Buruk
2	4	Baik
3	5	Baik
3	3	Baik
4	6	Baik

4.2.3 Implementasi K-Nearest Neighbor

Pada implementasi kali ini akan melakukan pemodelan sederhana dengan menggunakan atribut tinggi badan dan berat badan untuk menentukan jenis kelamin seseorang.

Pemodelan menggunakan bahasa python yang dijalankan ke dalam <https://colab.research.google.com/>

Langkah pertama ditentukan dulu Sample Datasetnya, misalnya data diberi nama sensus

Sample Dataset

```
import pandas as pd
```

```
sensus = {
```

```
    'tinggi': [158, 178, 183, 191, 155, 163, 188, 158, 178],
```

```
    'berat' : [64, 86, 84, 88, 48, 59, 67, 54, 67],
```

```
    'jk' : [ 'pria', 'pria', 'pria', 'pria', 'wanita', 'wanita', 'wanita', 'wanita', 'wanita']
```

```
}
```

```
sensus_df = pd.DataFrame(sensus)
```

```
sensus_df
```


Pada gambar 4.1 menunjukkan hasil ketika *script* dijalankan

	tinggi	berat	jk
0	158	64	pria
1	178	86	pria
2	183	84	pria
3	191	88	pria
4	155	48	wanita
5	163	59	wanita
6	188	67	wanita
7	158	54	wanita
8	178	67	wanita

Gambar 4.1 : Hasil Dataset
(Sumber : SKLearn Indonesia Belajar)

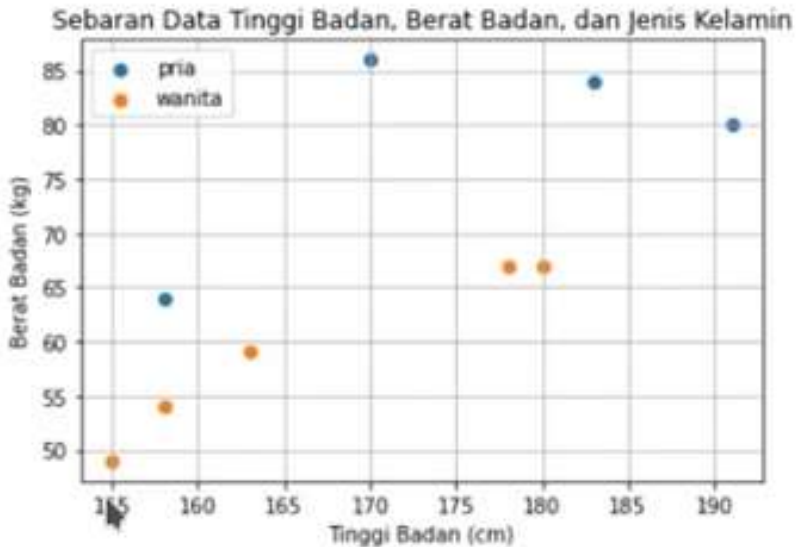
Langkah berikut dicoba untuk membuat tampilan visualisasinya
`import matplotlib.pyplot as plt`

```
fig, ax = plt.subplots()
for jk, d in sensus_df.groupby('jk'):
    ax.scatter(d['tinggi'], d['berat'], label=jk)
```

```
plt.legend(loc='upper left')
plt.title('Sebaran Data Tinggi Badan, Berat Badan, dan Jenis Kelamin')
plt.xlabel('Tinggi Badan (cm)')
plt.ylabel('Berat Badan(kg)')
```

```
plt.grid(True)
plt.show()
```

Tampilan visualisasi akan muncul seperti pada gambar 4.2



Gambar 4.2 : Hasil Dataset
(Sumber : SKLearn Indonesia Belajar)

Langkah berikutnya pengklasifikasian dengan KNN. Sebelum mengimplementasikan algoritma KNN sebelumnya melalui *Preprocessing* Dataset.

```
import numpy as np
x_train = np.array(sensus_df[['tinggi','berat']])
y_train = np.array(sensus_df['jk'])

print(f'x_train:\n{x_train}\n')
print(f'y_train: {y_train}')
```

Hasil ketika *script* dijalankan, dimana *x_train* merupakan nilai dari atribut, sedangkan *y_train* nilai dari kelas/target. Dari hasil untuk nilai *x_train* sudah berbentuk 2 dimensi dan bertipe numerik, namun untuk *y_train* masih dalam tipe *string*. Sehingga perlu diubah dulu untuk *y_trainnya*.

x_train:

```
[[158 64]
 [178 86]
 [183 84]
 [191 88]
 [155 48]
 [163 59]
 [188 67]
 [158 54]
 [178 67]]
```

y_train: ['pria' 'pria' 'pria' 'pria' 'wanita' 'wanita' 'wanita' 'wanita' 'wanita']

Langkah berikutnya mengubah *y_train* dari tipe *string* ke bentuk numerik seperti pada *script* di bawah ini

```
from sklearn.preprocessing import LabelBinarizer
```

```
lb=LabelBinarizer()
y_train = lb.fit_transform(y_train)
print(f'y_train:\n{y_train}')
```

Hasil ketika *script* dijalankan, bisa diperhatikan nilai 0 merupakan representasi dari data pria, dan nilai 1 merupakan representasi dari data wanita. Namun setelah melalui proses transformasi menggunakan *LabelBinarizer* data tersimpan menjadi array 2 dimensi sehingga perlu untuk dikembalikan menjadi *array* 1 dimensi.

y_train:

```
[[0]
 [0]
 [0]
 [0]
 [1]
 [1]
 [1]
 [1]
 [1]]
```

Langkah mengembalikan data menjadi *array* 1 dimensi.

```
y_train=y_train.flatten()
print(f'y_train:{y_train}')
```

Hasil setelah *script* dijalankan

```
y_train:[0 0 0 0 1 1 1 1 1]
```

Setelah dataset sudah siap, maka langkah selanjutnya melakukan training *model* dengan menggunakan algoritma KNN from sklearn.neighbors import KNeighborsClassifier
K=3

```
model = KNeighborsClassifier(n_neighbors=K)
model.fit(x_train, y_train)
```

Pada *script* di atas menggunakan perintah *import* KNN, dengan parameter K = 3, yang digunakan untuk menentukan jumlah tetangga terdekat yang akan dilibatkan untuk prediksi.

Hasil setelah *script* dijalankan, kemungkinan prosesnya terbilang cepat karena data yang *ditraining*kan dalam uji coba jumlahnya yang memang sangat kecil.

```
KNeighborsClassifier(n_neighbors=3)
```

Setelah model klasifikasi KNN *ditraining*, langkah berikutnya penggunaan tren model untuk prediksi jenis kelamin berdasarkan tinggi_badan, dan berat_badan dengan data baru.

```
tinggi_badan = 155
```

```
berat_badan = 70
```

```
x_new = np.array([tinggi_badan, berat_badan]).reshape(1, -1)
```

```
x_new
```

Pada *script* di atas data baru tinggi_badan bernilai 155, dan berat_badan bernilai 70, karena data setnya hanya 1 maka harus dibuat menjadi 2 dimensi dengan menggunakan perintah *reshape*. Hasil setelah dijalankan, dataset sudah menjadi *array* 2 dimensi `array([[155, 70]])`

Setelah data tinggi_badan dan berat_badan siap, langkah selanjutnya memprediksi jenis_kelamin berdasarkan data *training* sebelumnya.

```
y_new = model.predict(x_new)
```

```
y_new
```

Hasil setelah *script* dijalankan, maka akan muncul target bernilai 1, ini artinya data baru dengan tinggi_badan 155 dan berat_badan 70 diprediksi berkelamin 1.

```
array([1])
```

Untuk melihat jenis_kelamin 1 termasuk apa maka dapat dipanggil lagi dengan perintah *script*

```
lb.inverse_transform(y_new)
```

Setelah dijalankan *scriptnya* maka akan muncul hasil

```
array(['wanita'], dtype='<U6')
```

Ini artinya jenis_kelamin bernilai 1 berkorelasi dengan jenis_kelamin wanita. Dengan kata lain data dengan tinggi_badan 155 dan berat_badan 70 diprediksi berjenis kelamin wanita.

Setelah dipelajari KNN secara umum, selanjutnya bagaimana KNN bekerja berdasarkan pengukuran jarak terdekat dari data yang diprediksi dengan sekumpulan data yang dijadikan training.

Pada materi sebelumnya untuk pengukuran jarak terdekat algoritma KNN menggunakan metode *Euclidean Distance*.

Proses kalkulasi jarak menggunakan *Euclidean Distance*.

```
misterius = np.array([tinggi_badan, berat_badan])
```

```
misterius
```

Hasil setelah *script* dijalankan

```
array([155, 70])
```

Selanjutnya ditampilkan nilai atribut dari data *training*

```
x_train
```

Hasil setelah *script* dijalankan

```
array([[158, 64], [178, 86], [183, 84], [191, 88], [155, 48], [163, 59],  
[188, 67], [158, 54], [178, 67]])
```

Kemudian dilanjutkan dengan menghitung jarak antara data testing yang ada pada misterius dengan setiap data pada data training dengan menggunakan metode *Euclidean Distance*.

```
from scipy.spatial.distance import euclidean
```

```
data_jarak = [euclidean(misterius, d) for d in x_train]  
data_jarak
```

Hasil setelah *script* dijalankan menunjukkan jarak antara data testing dengan data *training*

```
6.708203932499369, 28.0178514522438, 31.304951684997057,  
40.24922359499622, 22.0, 13.601470508735444,  
33.13608305156178, 16.278820596099706,  
23.194827009486403]
```

Selanjutnya untuk memudahkan pengamatan maka dibuat kolom baru pada sensus, dengan memanfaatkan `data_jarak`. Dan data sensus nantinya akan tampil dengan urutan dari nilai terkecil ek nilai terbesar.

```
sensus_df['jarak']=data_jarak  
sensus_df.sort_values(['jarak'])
```

Hasil setelah script dijalankan ada pada gambar 4.3.

	tinggi	berat	jk	jarak
0	158	64	pria	6.708204
5	163	59	wanita	13.601471
7	158	54	wanita	16.278821
4	155	48	wanita	22.000000
8	178	67	wanita	23.194827
1	178	86	pria	28.017851
2	183	84	pria	31.304952
6	188	67	wanita	33.136083
3	191	88	pria	40.249224

Gambar 4.3 : Hasil Dataset
(Sumber : SKLearn Indonesia Belajar)

Dari gambar 4.3 dapat diamati bahwasannya data sudah terurut dari terkecil ke terbesar. Karena nilai $K=3$, maka hanya 3 baris tetangga terdekat saja yang menjadi prioritas prediksi, dari ketiga baris tersebut jenis kelamin wanita berjumlah 2 dan jenis kelamin pria berjumlah 1, karena jumlah wanita lebih banyak dibandingkan pria, maka hasil prediksi misterius menggunakan algoritma KNN adalah wanita.

DAFTAR PUSTAKA

- Boiculescu, V.L., Dimitriu, G. and Moscalu, M. 2013. 'Improving recall of k-nearest neighbor algorithm for classes of uneven size', *2013 E-Health and Bioengineering Conference, EHB 2013*, (1), pp. 23–26. doi:10.1109/EHB.2013.6707403.
- Kumar, M., Chhabra, P. and Garg, N.K. 2018. 'An efficient content based image retrieval system using BayesNet and K-NN'.
- Larose, D.T. and Larose, C.D. 2014. *Discovering Knowledge in Data: An Introduction to Data Mining: Second Edition, Discovering Knowledge in Data: An Introduction to Data Mining: Second Edition*. doi:10.1002/9781118874059.
- Liu, B. 2012. *Chapter 13 A SURVEY OF OPINION MINING AND SENTIMENT ANALYSIS*. doi:10.1007/978-1-4614-3223-4.
- Prasetyo, E., Alim, S. and Rosyid, H. 2014. 'Uji Kinerja Dan Analisis K-Support Vector Nearest Neighbor dengan SVM dan ANN Back-Propagation', *Prosiding SENTIA*, 6(November), pp. 173–178.

BAB 5

K-MEANS

Oleh Anindya Khrisna Wardhani

5.1 Pendahuluan

Kemajuan teknologi berjalan sangat cepat belakangan ini hingga mendorong peningkatan penting dalam jumlah data yang dihasilkan dan disimpan di bidang-bidang seperti teknik, keuangan, pendidikan, kedokteran, dan perdagangan, antara lain. Oleh karena itu, ada kepentingan yang dibenarkan untuk memperoleh pengetahuan berguna yang dapat diekstraksi dari sejumlah besar data tersebut, untuk membantu membuat keputusan yang lebih baik dan memahami sifat data. Clustering adalah salah satu teknik dasar untuk mendapatkan wawasan tentang sifat dasar dan struktur data. Tujuan clustering adalah mengorganisasikan sekumpulan data ke dalam cluster-cluster yang elemen-elemennya mirip satu sama lain dan berbeda dengan yang ada di cluster lain (Bimantoro & Wardhani, 2020).

Salah satu *algoritma clustering* yang banyak digunakan hingga saat ini adalah K-means, karena kemudahannya untuk menginterpretasikan hasil dan implementasinya. Faktor lain yang berkontribusi terhadap penggunaannya adalah adanya versi yang diimplementasikan di platform Weka dan SPSS dan bahasa pemrograman *open-source* seperti Python dan R, antara lain (Vishal et al., 2021). Clustering adalah proses membagi populasi atau kumpulan titik data ke dalam kelompok dan menugaskan mereka ke kelompok cluster berdasarkan kesamaan. K-Means Clustering adalah berbasis centroid, partisi clustering dan algoritma pembelajaran tanpa pengawasan (A. K. Wardhani et al., 2018). Dalam algoritma ini, pengukuran berbasis jarak digunakan untuk

menentukan kesamaan antara titik data. Dataset yang tidak berlabel dikelompokkan ke dalam k-jumlah kluster. Pada proses ini nilai K harus ditentukan terlebih dahulu dan proses diulang sampai tidak ditemukan cluster terbaik (A. K. Wardhani, 2017).

K-means clustering adalah algoritma untuk membagi satu set titik data menjadi k cluster yang berbeda, di mana setiap cluster ditentukan oleh rata-rata titik-titik dalam cluster. Titik data ditugaskan ke cluster dengan rata-rata terdekat. Pengelompokan K-means adalah metode populer untuk analisis data ketika tujuannya adalah untuk mengidentifikasi struktur yang mendasari dalam kumpulan data. Harapannya adalah bahwa cluster yang dihasilkan akan selaras dengan beberapa struktur minat yang mendasarinya. Misalnya, kita memiliki kumpulan data medis di mana setiap titik data terdiri dari kumpulan pengukuran yang relevan untuk seseorang. Harapannya kemudian adalah *k-means clustering* akan menghasilkan cluster yang mengidentifikasi sub-populasi orang yang berguna – misalnya, kelompok orang yang mungkin merespons pengobatan secara berbeda. Pengelompokan K-means telah ada sejak lama – dinamai 'k-means' pada tahun 1967 oleh James MacQueen [MacQueen, 1967], tetapi algoritmenya sendiri dikembangkan sepuluh tahun sebelumnya oleh Stuart Lloyd di Bell Labs (Sudirman et al., 2018).

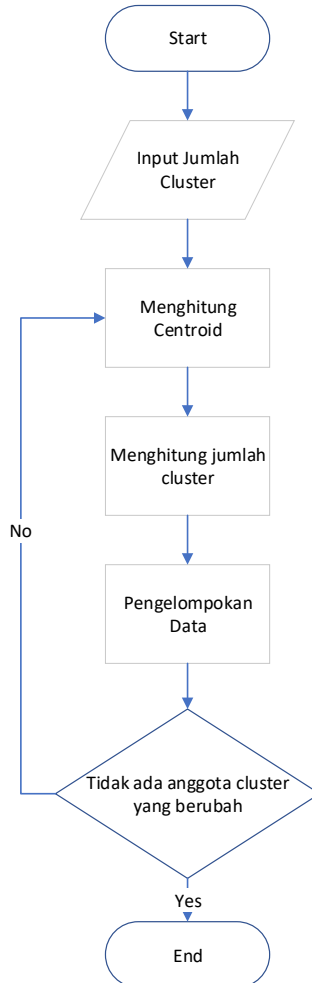
5.2 Langkah – Langkah K-Means

Metode pengelompokan yang diusulkan oleh Jancey terdiri dari empat langkah berikut:

1. Inisialisasi. Pertama, k poin dihasilkan secara acak di dalam ruang, yang digunakan sebagai centroid awal.
2. Klasifikasi. Jarak dari semua objek ke semua centroid dihitung, dan setiap objek ditempatkan ke centroid terdekatnya.
3. Perhitungan pusat massa. Centroid baru dihitung menggunakan nilai rata-rata dari objek yang termasuk dalam setiap cluster.

4. Konvergensi. Algoritme berhenti ketika kesetimbangan tercapai, yaitu ketika tidak ada migrasi objek dari satu cluster ke cluster lainnya. Meskipun tidak tercapai kesetimbangan, proses diulangi dari langkah 2(A. K. Wardhani, 2016).

Berikut merupakan gambaran flowchart metode K-Means :



Gambar 5.1 : Flowchart K-Means

Metode K-Means memiliki input dan 3 proses yaitu proses pertama adalah menghitung centroid, kemudian proses kedua menghitung data yang akan dikelompokkan dengan centroid, kemudian proses ketiga adalah mengelompokkan data berdasarkan jarak terdekat (*minimum distance*). Kemudian dilakukan perulangan dengan kondisi "apakah posisi centroid tetap dan tidak ada perubahan terhadap datanya?" apabila ya, maka pengelompokan selesai. Tapi apabila masih ada perubahan centroid maka kita update kembali nilai centroid melalui proses pertama.

5.3 Kelebihan dan Kelemahan K-Means

Algoritma k-means dinilai cukup efisien, dengan ketentuan n adalah jumlah objek data, k adalah jumlah cluster yang terbentuk, dan t adalah jumlah iterasi. Secara umum nilai k dan t jauh lebih kecil dari nilai n . Selanjutnya, algoritma ini berhenti pada kondisi optimal lokal dalam iterasinya (Ardilla et al., 2021).

Namun, algoritma k-means memiliki beberapa kelemahan algoritma seperti wajib menentukan jumlah cluster yang akan dibentuk, hanya dapat digunakan dalam data yang mean-nya dapat ditentukan, dan tidak mampu menangani data yang mempunyai penyimpangan-penyimpangan (noisy data dan outlier). Selain itu, algoritma k-means sangat bergantung pada pemilihan nilai awal centroid, tidak jelas berapa banyak cluster k yang terbaik dan hanya bekerja pada atribut numerik (A. Wardhani, 2016).

5.4 Implementasi K-Means

Untuk membuat aturan asosiasi, setidaknya diperlukan dua langkah untuk menggunakan nilai dukungan dan kepercayaan di atas:

Tabel 5.1 : Dataset

NO	NAMA MAHASISWA	UTS	TUGAS	UAS
1	Imam	8 9	90	75
2	Clarisa	9 0	71	95
3	Iqbal	7 0	75	80
4	Dilan	4 5	65	59
5	Ratna	6 5	75	53
6	Meilia	8 0	70	75
7	Rudi	9 0	85	81
8	Hafiz	7 0	70	73
9	Tito	9 6	93	85
10	Bima	6 0	55	48
11	Justin	4 5	60	58
12	Jesika	6 0	70	72
13	Ayu	8 5	90	88
14	Siska	5 2	68	55
15	Yusup	4 0	60	70

Langkah pertama adalah menentukan jumlah kelompok (*cluster*) yang akan dibentuk. Adapun cluster yang akan dibentuk antara lain :

- a. Cluster 1 (C1) = Pintar
- b. Cluster 2 (C2) = Sedang
- c. Cluster 3 (C3) = Kurang

Tetapkan pusat kelompok awal secara random dari data, dipilih 3 pusat diantaranya :

Tabel 5.2 : Centroid Awal

Cluster 1	96	93	85
Cluster 2	70	75	80
Cluster 3	60	55	48

Hitung jarak setiap data ke pusat cluster menggunakan metode euclidian distance.

$$d(1,1) = \sqrt{(89 - 96)^2 + (90 - 93)^2 + (75 - 85)^2} = 12,56981$$

$$d(1,2) = \sqrt{(89 - 70)^2 + (90 - 75)^2 + (75 - 80)^2} = 24,71841$$

$$d(1,3) = \sqrt{(89 - 60)^2 + (90 - 55)^2 + (75 - 48)^2} = 52,86776$$

Setelah melakukan perhitungan maka didapat hasil seperti berikut ini:

Tabel 5.3 : Hasil Euclidian Distance

No	Nama Mahasiswa	Jarak Ke Cluster			Hasil
		C1	C2	C3	
1.	Imam	12,56980509	24,71841419	52,86775955	Cluster 1
2.	Clarisa	24,8997992	25,3179778	58,00862005	Cluster 1
3.	Iqbal	32,01562119	0	39,03844259	Cluster 2
4.	Dilan	63,72597587	34,14674216	21,11871208	Cluster 3
5.	Ratna	48,05205511	27,45906044	21,21320344	Cluster 3
6.	Meilia	29,74894956	12,24744871	36,79673899	Cluster 2
7.	Rudi	10,77032961	22,38302929	53,7494186	Cluster 1
8.	Hafiz	36,72873534	8,602325267	30,82207001	Cluster 2
9.	Tito	0	32,01562119	64,10148204	Cluster 1
10.	Bima	64,10148204	39,03844259	0	Cluster 3
11.	Justin	66,47555942	36,52396474	18,70828693	Cluster 3
12.	Jesika	44,65422712	13,74772708	28,3019434	Cluster 2
13.	Ayu	11,78982612	22,6715681	58,73670062	Cluster 1
14.	Siska	58,83026432	31,591138	16,79285562	Cluster 3

No	Nama Mahasiswa	Jarak Ke Cluster			Hasil
		C1	C2	C3	
15.	Yusup	66,70832032	35	30,14962686	Cluster 3

Tentukan kembali titik pusat cluster yang baru

$$\text{Rumus} = \frac{\text{nilai hasil}}{\text{banyak hasil}}$$

Cluster 1

$$(\text{UTS}) = \frac{(89+90+90+90+85)}{5} = 90$$

$$(\text{Tugas}) = \frac{(90+71+85+93+90)}{5} = 85,8$$

$$(\text{UAS}) = \frac{(75+95+81+85+88)}{5} = 84,8$$

Lakukan, perhitungan tersebut untuk cluster 2 dan 3, sehingga didapat nilai cluster baru antara lain :

Tabel 5.4 : Centroid Baru

Kluster 1	90	85,8	84,8
Kluster 2	70	71,25	75
Kluster 3	51,16666667	63,83333333	57,16666667

Lakukan kembali perhitungan menggunakan euclidian distance menggunakan pusat cluster baru hingga anggota cluster tidak berubah sehingga menghasilkan pengelompokan seperti table dibawah ini :

No	Nama Mahasiswa	Jarak Ke Cluster			Hasil
		C1	C2	C3	
1.	Imam	10,70887482	26,69386634	49,33643008	Cluster 1
2.	Clarisa	17,97442628	28,28537608	54,68775	Cluster 1
3.	Iqbal	23,23101375	6,25	31,63463292	Cluster 2
4.	Dilan	55,88631317	30,33253204	6,538348415	Cluster 3
5.	Ratna	41,86740976	22,87055968	18,25970062	Cluster 3
6.	Meilia	21,1111345	10,07782219	34,45891273	Cluster 2
7.	Rudi	3,883297568	25,00124997	50,2402561	Cluster 1
8.	Hafiz	28,08700767	2,358495283	25,3656592	Cluster 2
9.	Tito	9,374433316	35,34207832	60,29441655	Cluster 1
10.	Bima	56,59399261	33,06149573	15,49462272	Cluster 3
11.	Justin	58,38561467	32,25775101	7,30867065	Cluster 3
12.	Jesika	36,24196463	10,51487042	18,33257574	Cluster 2

No	Nama Mahasiswa	Jarak Ke Cluster			Hasil
		C1	C2	C3	
13.	Ayu	7,271863585	27,30499039	52,72649555	Cluster 1
14.	Siska	51,46727115	27,10281351	4,769696007	Cluster 3
15.	Yusup	58,17800272	32,42780443	17,43798536	Cluster 3

DAFTAR PUSTAKA

- Ardilla, Y., Wardhani, A. K., Mustika, Manuhutu, A., Ahmad, N., Hasbi, I., Guntoro, Manuhutu, M. A., Ridwan, M., Hozairi, Alim, S., Romli, I., Religia, Y., Octafian, T., Sufandi, U. U., & Ernawati, I. 2021. *DATA MINING DAN APLIKASINYA* (Rismawati, Ed.; 1st ed.). Widina Bhakti Persada Bandung. <https://repository.penerbitwidina.com/media/publications/351768-data-mining-dan-aplikasinya-7b2a8129.pdf>
- Bimantoro, T., & Wardhani, A. K. 2020. IMPLEMENTASI ALGORITMA PARTITIONING AROUND MEDOIDS DALAM PENGELOMPOKAN RESTORAN. *Indonesian Journal of Technology, Informatics and Science (IJTIS)*, 2(1), 33–36. <https://doi.org/10.24176/ijtis.v2i1.5651>
- Sudirman, Windarto, A. P., & Wanto, A. 2018. Data mining tools | rapidminer: K-means method on clustering of rice crops by province as efforts to stabilize food crops in Indonesia. *IOP Conference Series: Materials Science and Engineering*, 420(1). <https://doi.org/10.1088/1757-899X/420/1/012089>
- Vishal, Kumar, A., & Saini, P. K. 2021. Use of K-Means Clustering Method for Books Data in Acharya Raghuveer Library, Central University of Himachal Pradesh, Dharamshala, India. *Library Philosophy and Practice*.
- Wardhani, A. 2016. IMPLEMENTASI ALGORITMA K-MEANS UNTUK PENGELOMPOKAN PENYAKIT PASIEN PADA PUSKESMAS KAJEN PEKALONGAN. *Jurnal Transformatika*, Volume 14 No 1, 30–37.
- Wardhani, A. K. 2016. Implementasi Algoritma K-Means untuk Pengelompokan Penyakit Pasien pada Puskesmas Kajen Pekalongan. *Jurnal Transformatika*, 14(1), 30–37.

- Wardhani, A. K. 2017. *PENERAPAN ALGORITMA PARTITIONING AROUND MEDOIDS UNTUK MENENTUKAN KELOMPOK PENYAKIT PASIEN (STUDI KASUS: PUSKESMAS KAJEN PEKALONGAN)* (Vol. 6, Issue 1).
- Wardhani, A. K., Widodo, C. E., & Suseno, J. E. 2018. *Information System for Culinary Product Selection Using Clustering K-Means and Weighted Product Method*. 165(ICCSR), 18–22. <https://doi.org/10.2991/iccsr-18.2018.5>

BAB 6

ALGORITMA NAÏVE BAYES

Oleh Rahmaddeni

6.1 Algoritma Naive Bayes

Algoritma Naïve Bayes Termasuk Metode Supervised Learning bagian Klasifikasi. Klasifikasi merupakan cara mendapatkan model atau fungsi yang mampu mendeskripsikan serta menyeleksi kelas data atau konsep, yang bermaksud supaya model tersebut mampu diaplikasikan dalam memproyeksikan kelas yang tidak pasti dari dari sebuah entitas yang diamati. Metode klasifikasi yang kerap dipakai dalam bidang statistika ialah Analisis Diskriminan dan Regresi Logistik. Akan tetapi, bertambah populernya era data berarti ada pertumbuhan cepat volume data yang sangat besar dan menghasilkan kumpulan data besar (big data), untuk itu media kajian yang valid dan fleksibilitas sangat penting dalam menggali wawasan berguna pada kumpulan data besar serta memperbaiki kumpulan data besar tersebut membentuk wawasan yang tertata. Meskipun, lajunya kemajuan teknologi kecerdasan buatan menyebabkan berkembangnya metode machine learning, yakni mesin disempurnakan demi dapat berlatih sendiri tanpa bimbingan pengguna, dimana peningkatannya bertumpu pada bidang lain seperti statistika, matematika dan data mining. Metode klasifikasi pada machine learning yang biasa digunakan dalam metode machine learning adalah Classification and Regression Trees (CART), Random Forest, Naïve Bayes, Support Vector Machines (SVM), dan lain-lain. (Yuliati et al., 2020).

Algoritma Naive Bayes merupakan algoritma yang simpel serta masing-masing atribut bersifat independen, yang

membolehkan masing-masing atribut bisa berpartisipasi pada keluaran akhir. Satu diantara metode algoritma yang tercantum dalam proses klasifikasi merupakan algoritma Naïve Bayes. Proses klasifikasi telah dicoba oleh Naïve Bayes memakai teknik probabilitas serta statistik. Thomas Bayes ialah ilmuwan berketurunan Inggris dan sukses menciptakan Algoritma Naïve Bayes. Rancangan awal dari algoritma Naïve Bayes merupakan bisa melaksanakan prakiraan peluang di masa yang hendak tiba bersumber pada informasi dari masa dulu, serta ide ini diketahui dengan Teorema Bayesian. Rancangan ini digabungkan oleh Naïve yang mempraktikkan nilai atribut yang independen. Algoritma Naïve Bayes mempraktikkan kalau karakteristik spesifik dari suatu kelas tidak berkaitan atas kelas yang lain (Yunita et al., 2018).

Naïve Bayes yaitu pengategorian atau pengelompokkan dengan metode kemungkinan dan perangkaan yang dikemukakan cendekiawan asal Inggris Thomas Bayes, yang meramal suatu peluang di masa depan berdasarkan data di masa lebih dahulu. Naïve Bayes dilandaskan oleh dugaan simplifikasi jika nilai atribut secara sementara saling bebas bila dialokasikan nilai output. Dengan kata lain, didistribusikan nilai output, probabilitas meninjau secara bersama ialah keluaran dari probabilitas individu (Widodo et al., 2021).

Dalam jurnal (Manalu et al., 2017) menerangkan kalau Naïve Bayes ialah suatu pengelompokkan probabilistik simpel yang menilai gabungan probabilitas dengan menambahkan frekuensi serta campuran nilai dari kumpulan data yang dibagikan. Algoritma memanfaatkan teorema Bayes serta memperhitungkan seluruh atribut bebas ataupun tidak saling keterkaitan yang disajikan atas nilai pada variabel kelas. Manfaat pemakaian Naïve Bayes merupakan metode ini cukup memerlukan sejumlah data pelatihan (*Training Data*) dalam menaksir perkiraan skala yang dibutuhkan pada proses pengelompokkan. Naïve Bayes kerap beroperasi jauh

lebih baik dalam mayoritas suasana dunia nyata yang kompleks dari pada yang diperlukan.

6.2 Kelebihan dan Kekurangan Naive Bayes

Mengenai keunggulan dan kelemahan dari Algoritma Naive Bayes dapat diuraikan sebagai berikut :

6.2.1 Kelebihan Algoritma Naive Bayes

Kelebihan penggunaan Algoritma Naive Bayes antara lain

1. Algoritma ini cukup memerlukan sejumlah data pelatihan (Training Data) dalam menaksir perkiraan skala yang dibutuhkan dalam proses pengelompokkan. Naive Bayes kerap beroperasi jauh lebih baik dalam mayoritas suasana dunia nyata yang kompleks dari pada yang diperlukan (Widodo et al., 2021).
2. Cepat dalam perhitungan, algoritma simpel dan presisi. Naive Bayes lebih akurat diaplikasikan dalam data yang besar dan dapat mengerjakan data yang tidak utuh (*missing value*) serta kuat atas atribut yang tidak bermakna dan *noise* pada data (Arifin & Ariesta, 2019).
3. Mudah diimplementasikan dan dalam banyak kasus memberikan hasil yang baik (Suprianto, 2020).

6.2.2 Kekurangan Algoritma Naive Bayes

Kekurangan Algoritma Naive Bayes antara lain

1. Probabilitasnya tidak dapat menilai seberapa presisi suatu pengelompokkan. Algoritma klasifikasi Naive Bayes mempunyai kelemahan pada penetapan atribut yang terdapat di dalam data sehingga dapat mempengaruhi produk akhir berupa tingkat presisi (Arifin & Ariesta, 2019).
2. Amat rentan pada fitur yang berlebih, sehingga membuat tingkat presisi pengelompokkan menjadi rendah.

3. Tidak valid jika probabilitas sementara adalah kosong, jika kosong maka probabilitas prakiraan akan bernilai kosong juga. Serta memperhitungkan variabel independen (Suprianto, 2020).

6.3 Persamaan dan Langkah Naive Bayes

Persamaan dari Teorema Bayes adalah

$$P(H|X) = \frac{P(H|X) \cdot P(X)}{P(X)}$$

Keterangan :

X : Data dengan class yang belum diketahui

H : Hipotesis data X merupakan suatu class spesifik

$P(H|X)$: Probabilitas hipotesis H berdasar kondisi X (posterior probability)

$P(H)$: Probabilitas hipotesis H (prior probability)

$P(X|H)$: Probabilitas X berdasarkan kondisi pada hipotesis H

$P(X)$: Probabilitas X

Adapun langkah atau tahapan dari Algoritma Naïve Bayes tersebut adalah

1. Menghitung jumlah kelas / label " $P(H)$ "
2. Menghitung jumlah kelas per kelas " $P(X|H)$ "
3. Kalikan semua variabel kelas 1 (yes) dan 0 (no) " $P(X|H) * P(H)$ "
4. Bandingkan hasil per kelas 1 (yes) dan 0 (no)

6.4 Tipe Algoritma Naïve Bayes

Mengenai tipe-tipe dari Algoritma Naïve Bayes yang dapat digunakan sebagai berikut:

6.4.1 Naïve Bayes Classifier

Naïve Bayes Classifier merupakan suatu aturan klasifikasi yang menggunakan prinsip kalkulasi peluang (Amrullah et al., 2020) dan memperhitungkan seluruh atribut independen atau tidak saling keterikatan yang dibagikan oleh nilai pada variabel kelas (Dita et al., 2021). Naïve Bayes Classifier memberikan manfaat yakni cukup memerlukan sejumlah data pelatihan (training data) untuk menaksir perkiraan skala yang dibutuhkan dalam proses pengelompokan (Affandi et al., 2020).

6.4.2 Multinomial Naïve Bayes

Multinomial Naïve Bayes memperhitungkan jumlah kemunculan kata dalam dokumen (Purwiantono & Aditya, 2020), sehingga mengasumsikan independensi kehadiran kata pada manuskrip dengan tidak memperkirakan urutan kata ataupun konteks informasi. (Haditira et al., 2022). Manuskrip Multinomial Naïve Bayes dapat juga dikenal sebagai “bag of words” yang rentetan kejadian hadirnya kata dalam manuskrip diabaikan, sehingga setiap kata diproses menerapkan distribusi multinomial (Reddy et al., 2022).

6.4.3 Gaussian Naïve Bayes Classifier

Metode Gaussian naïve bayes ialah satu dari algoritma berbasis nilai berkelanjutan dengan rancangan probabilitas yang mampu diaplikasikan pada penetapan kelas dari manuskrip dan dapat mengerjakan data dalam jumlah besar dengan presisi (Mujahidin et al., 2022).

Gaussian Naive Bayes ialah satu dari Naïve Bayes, di mana metode ini beroperasi dengan probabilitas. Pada metode ini data

yang diaplikasikan pada fitur yang dipakai yakni bertipe angka (Cahyaningrum et al., 2022).

6.5 Studi Kasus Algoritma Naive Bayes

Pada pembahasan ini, Algoritma Naïve Bayes akan diterapkan pada suatu kasus yakni sebuah dataset publik yang bersumber dari dataset kaggle yakni <https://www.kaggle.com/datasets/zgeakyldz/heart-desease-data>. Dataset tersebut terdiri atas 303 data dengan 13 feature dan 1 label atau target.

Adapun 13 feature dan 1 label tersebut yakni

- **age** : usia dalam tahun.
- **sex** : jenis kelamin.
- **cp** : nyeri dada yang dialami (Nilai 1: angina tipikal, Nilai 2: angina atipikal, Nilai 3: nyeri non angina, Nilai 4: asimtomatik)
- **trestbps**: tekanan darah seseorang (mm Hg saat masuk ke rumah sakit).
- **chol**: pengukuran kolesterol dalam mg/dl.
- **fbs** : gula darah (> 120 mg/dl, 1 = benar; 0 = salah).
- **restecg**: pengukuran elektrokardiografi (0 = normal, 1 = memiliki kelainan gelombang ST-T, 2 = menunjukkan kemungkinan atau pasti hipertrofi ventrikel kiri menurut kriteria Estes).
- **thalach**: detak jantung maksimum seseorang tercapai.
- **exang**: latihan menginduksi angina (1 = ya; 0 = tidak).
- **oldpeak**: depresi ST yang diinduksi oleh olahraga relatif terhadap istirahat ('ST' berkaitan dengan posisi pada plot EKG).
- **slope** : kemiringan segmen ST latihan puncak (Nilai 1 : miring ke atas, Nilai 2 : datar, Nilai 3 : miring ke bawah).
- **ca**: jumlah major vessels (0-3)

- **thal:** kelainan darah yang disebut thalassemia (3 = normal; 6 = cacat tetap; 7 = cacat reversibel).
- **target:** heart disease (0 = non stroke, 1 = stroke)

6.5.1 Perhitungan Manual Algoritma Naïve Bayes

Sebelum melakukan langkah Algoritma Naïve Bayes secara manual maka dataset diatas dilakukan pembersihan (cleaning data) yakni data berupa NaN yang mana nilai itu ialah nilai ca = 4 (data ca<0) dan nilai th = 0 (data th=0). Sehingga data yang telah dibersihkan berjumlah 296 data. Selanjutnya dilakukan langkah dari Algoritma Naïve Bayes yang diuraikan berikut ini :

1. Menghitung jumlah kelas / label “P(H)”

Tabel 6.1 : Jumlah kelas pertarget dan prior

Kelas	Jumlah	Prior
non stroke	136	0.506
stroke	160	0.595

2. Menghitung jumlah kelas per kelas “P(X|H)”

Tabel 6.2 : Jumlah kasus perkelas

No	Kelas	Jumlah	Prior
1	P(age 43 non stroke)	3	0.022
	P(age 43 stroke)	5	0.031
2	P(sex 1 non stroke)	114	0.838
	P(sex 1 stroke)	93	0.581
3	P(cp 1 non stroke)	7	0.051
	P(cp 1 stroke)	16	0.100
4	P(trestbps 200 non stroke)	1	0.007
	P(trestbps 200 stroke)	0	0.000
5	P(chol 233 non stroke)	1	0.007
	P(chol 23 stroke)	3	0.019

No	Kelas	Jumlah	Prior
6	P(fbs 1 non stroke)	22	0.162
	P(fbs 1 stroke)	23	0.144
7	P(restecg 0 non stroke)	79	0.581
	P(restecg 0 stroke)	68	0.425
8	P(thalach 103 non stroke)	2	0.015
	P(thalach 103 stroke)	0	0.000
9	P(exang 1 non stroke)	76	0.559
	P(exang 1 stroke)	23	0.144
10	P(oldpeak 0.9 non stroke)	2	0.015
	P(oldpeak 0.9 stroke)	1	0.006
11	P(slope 1 non stroke)	91	0.669
	P(slope 1 stroke)	49	0.306
12	P(ca 2 target 1)	31	0.228
	P(ca 2 target 0)	7	0.044
13	P(thal 2 target 1)	36	0.265
	P(thal 2 target 0)	130	0.813

3. Kalikan semua variabel kelas “ $P(X|H) * P(H)$ ”

Tabel 6.3 : Perhitungan semua variabel kelas

kelas $P(X H)$	$P(\text{non stroke})$	$P(\text{stroke})$
$P(\text{age } 43 \text{ non stroke})$	0.022	
$P(\text{age } 43 \text{ stroke})$		0.031
$P(\text{sex } 1 \text{ non stroke})$	0.838	
$P(\text{sex } 1 \text{ stroke})$		0.581
$P(\text{cp } 1 \text{ non stroke})$	0.051	
$P(\text{cp } 1 \text{ stroke})$		0.100
$P(\text{trestbps } 200 \text{ non stroke})$	0.007	

kelas P(X H)	P(non stroke)	P(stroke)
P(trestbps 200 stroke)		0.000
P(chol 233 non stroke)	0.007	
P(chol 23 stroke)		0.019
P(fbs 1 non stroke)	0.162	
P(fbs 1 stroke)		0.144
P(restecg 0 non stroke)	0.581	
P(restecg 0 stroke)		0.425
P(thalach 103 non stroke)	0.015	
P(thalach 103 stroke)		0.000
P(exang 1 non stroke)	0.559	
P(exang 1 stroke)		0.144
P(oldpeak 0.9 non stroke)	0.015	
P(oldpeak 0.9 stroke)		0.006
P(slope 1 non stroke)	0.669	
P(slope 1 stroke)		0.306
P(ca 2 target 1)	0.228	
P(ca 2 target 0)		0.044
P(thal 2 target 1)	0.265	
P(thal 2 target 0)		0.813
P(X no stroke)	0.000000000000002359	
P(X stroke)		0

Berdasarkan data dari tabel diatas didapatkan nilai $P(X|\text{Stroke}) = 0$ sedangkan $P(X|\text{Non Stroke}) = 0.000000000000002359$

4. Membandingkan hasil perkelas

Tabel 6.4 : Perbandingan hasil perkelas

Target Stroke	Target Non Stroke
$P(X \text{stroke}) * P(\text{stroke})$ $= 0 * (160/269)$ $= 0$	$P(X \text{non stroke}) * P(\text{non stroke})$ $= 0.000000000000002359 * (136/269)$ $= 0.000000000000001193$

Berdasarkan data dari tabel perbandingan di atas maka hasil prediksi dari data uji ialah Non Stroke, karena hasil target Non Stroke lebih besar dari hasil Stroke.

6.5.2 Perhitungan Algoritma Naive Bayes Menggunakan Program Phyton

Dataset yang digunakan dalam perhitungan manual diatas diimplementasikan menggunakan bahasa pemrograman phyton dengan tampilan sebagai berikut

1. Import library

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
```
2. Import data csv

```
data = pd.read_csv('heart_desease_test.csv')
data.head()
```

Setelah dilakukan tahap import untuk memastikan bahwa dokumen telah terimport dengan tepat dapat mengecek 5 data teratas data set dapat dilihat pada gambar

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Gambar 6.1 : Tampilan dataset

3. Melihat jumlah baris dan kolom pada dataset

```
print("Heart data shape is:", data.shape[0], "x", data.shape[1])
```

Hasil :

Heart data shape is : 303 x 14

4. Tahapan preprocessing data

Pada dataset di data #93, 159, 164, 165 dan 252 memiliki ca=4 yang salah. Dalam dataset asli, mereka adalah NaN. data #49 dan 282 memiliki th = 0, juga salah. Mereka juga merupakan NaN dalam kumpulan data asli. Melakukan penghapusan data yang dianggap NaN dalam data asli. Serta menampilkan jumlah data setelah dilakukan pembersihan data sebagai berikut

```
data = data[data['ca'] < 4] #drop the wrong ca values
data = data[data['thal'] > 0] # drop the wong thal value
print("Heart data shape is:", data.shape[0], "x", data.shape[1])
```

Hasil :

Heart data shape is : 296 x 14

5. Menghitung jumlah label/target stroke dan non stroke

```
total = len(data["target"])  
stroke = data["target"].sum()  
non_stroke = len(data["target"]) - stroke  
print("Total label no stroke:", non_stroke)  
print("Total label stroke:", stroke)
```

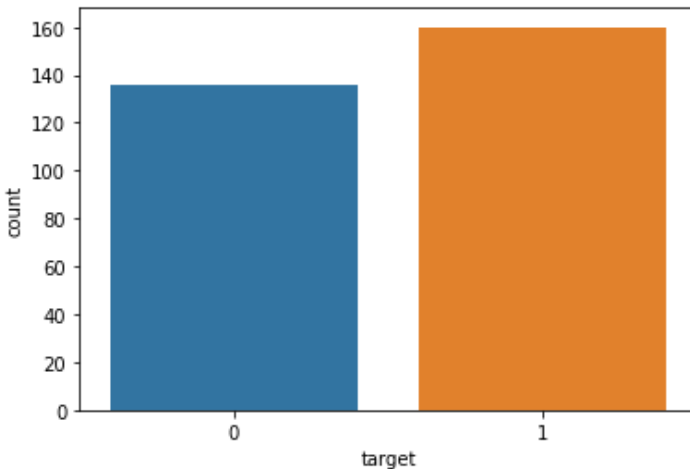
Hasil :

Total label no stroke : 136

Total label stroke : 160

6. Menghitung jumlah label/target stroke dan non stroke dengan bar chart

```
sns.countplot(data["target"])
```



Gambar 6.2 : Chart jumlah stroke dan no stroke

7. Melihat korelasi antar variabel

```
plt.figure(figsize=(10, 10))  
sns.heatmap(data.corr(), annot=True, fmt='.2f')
```

8. Splitting data

```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, plot_confusi
on_matrix, classification_report
from sklearn.metrics import recall_score, accuracy_score, ro
c_curve, auc
```

```
seed = 0
```

```
test_size = 0.2// ubah ke 0.3 untuk ke 70:30
```

```
X = data.drop(["target"], axis=1)
```

```
y = data["target"]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= test_size, random_state=seed)
```

9. Menginisialisasikan algoritma klasifikasi naive bayes serta melakukan pelatihan model.

```
clf = GaussianNB()
```

```
clf.fit(X_train, y_train)
```

10. Melihat nilai score

```
clf.score(X_train, y_train)
```

Hasil :

```
0.847457627118644
```

11. Melihat hasil prediksi dari dataset yang diolah

```
pred = clf.predict(X_test)
```

```
accuracy = accuracy_score(y_test, pred)
```

```
pred_proba = clf.predict_proba(X_test)[:, 1]
```

```
fpr, tpr, thresholds = roc_curve(y_test, pred_proba)
```

```
roc_auc = auc(fpr, tpr)
```

```
print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	0.89	0.78	0.83	32
1	0.78	0.89	0.83	28
accuracy			0.83	60
macro avg	0.84	0.84	0.83	60
weighted avg	0.84	0.83	0.83	60

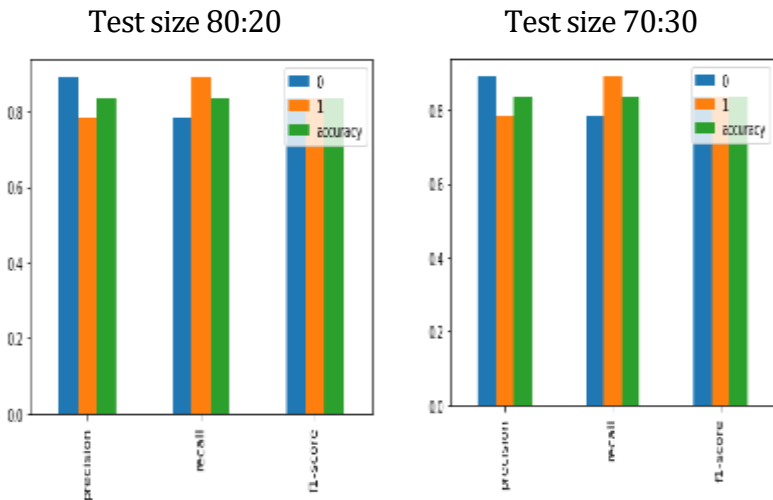
Gambar 6.3 : Tampilan report splitting data 80:20

	precision	recall	f1-score	support
0	0.87	0.76	0.81	45
1	0.78	0.89	0.83	44
accuracy			0.82	89
macro avg	0.83	0.82	0.82	89
weighted avg	0.83	0.82	0.82	89

Gambar 6.4 : Tampilan report splitting data 70:30

12. Visualisasi classification dengan splitting data 80:20 dan 70:30 dengan menggunakan bar chart

```
report=classification_report(y_test,pred, output_dict=True)
reporttb = pd.DataFrame(report).transpose()
reporttb.drop('support', inplace=True, axis=1)
reporttb.iloc[:3, :10].T.plot(kind='bar')
plt.show()
```



Gambar 6.5 : Visualisasi dengan splitting data 70:30 dan 80:20

13. Melakukan proses pengujian dengan data uji

```
X_pred = pd.read_csv("data_uji_heart_disease.csv")
```

```
X_pred
```

```
X_pred["target"] = clf.predict(X_pred)
```

```
X_pred
```

Hasil :

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target	
0	43	1	3	200	233	0	0	103	1	0.9	1	2	2	0

14. Hasil perbandingan akurasi dengan menggunakan splitting data 70:30 dan 80:20

Tabel 6.5 : Hasil perbandingan akurasi berdasarkan splitting data

Spliting data	Accuracy	precision		recall		f1-score	
		0	1	0	1	0	1
80 : 20	83%	89%	78%	78%	89%	83%	83%
70 : 30	82%	87%	78%	76%	89%	81%	83%

Berdasarkan hasil pada tabel diatas dapat di ambil kesimpulan bahwa dengan splitting data menggunakan 80:20 mendapatkan tingkat akurasi prediksi sebesar 83% untuk kasus prediksi heart disease. Pada hasil uji coba menggunakan data uji label yang didapatkan yakni no stroke.

DAFTAR PUSTAKA

- Affandi, L., Pramudhita, A. N., & Sasmita, M. P. 2020. Sistem pakar klasifikasi kecanduan gadget menggunakan teori arthurt t. hovart dengan metode naive bayes classifier untuk anak sekolah dasar. *Seminar Informatika Aplikatif Polinema*, 102–106.
<http://jurnalti.polinema.ac.id/index.php/SIAP/article/view/741>
- Amrullah, A. Z., Sofyan Anas, A., & Hidayat, M. A. J. 2020. Analisis Sentimen Movie Review Menggunakan Naive Bayes Classifier Dengan Seleksi Fitur Chi Square. *Jurnal*, 2(1), 40–44. <https://doi.org/10.30812/bite.v2i1.804>
- Arifin, T., & Ariesta, D. 2019. Prediksi Penyakit Ginjal Kronis Menggunakan Algoritma Naive Bayes Classifier Berbasis Particle Swarm Optimization. *Jurnal Tekno Insentif*, 13(1), 26–30. <https://doi.org/10.36787/jti.v13i1.97>
- Cahyaningrum, H., Arifianto, D., & Abdurrahman, G. 2022. Jurnal Smart Teknologi Analisis Perbandingan Metode K Nearest Neighbor Dan Gaussian Naive Bayes Pada Klasifikasi Jurusan Siswa (Studi Kasus pada Siswa SMA Muhammadiyah 3 Jember) A Comparative Analysis of K Nearest Neighbor and Gaussian Naive Bayes metho. *Jurnal Smart ...*, 3(3), 261–266.
<http://jurnal.unmuhjember.ac.id/index.php/JST/article/view/7385%0Ahttp://jurnal.unmuhjember.ac.id/index.php/JST/article/download/7385/3787>
- Dita, C. A. P., Chairunisyah, P., & ... 2021. Penerapan Naive Bayesian Classifier Dalam Penyeleksian Beasiswa PPA. *Journal of Computer ...*, 2(2), 194–198. <https://ejurnal.seminar-id.com/index.php/josyc/article/view/649>

- Haditira, R., Murdiansyah, D. T., & Astuti, W. 2022. *Analisis Sentimen Pada Steam Review Menggunakan Metode Multinomial Naïve Bayes dengan Seleksi Fitur Gini Index Text*. 9(3), 1793–1799.
- Manalu, E., Sianturi, F. A., & Manalu, M. R. 2017. Penerapan Algoritma Naive Bayes Untuk Memprediksi Jumlah Produksi Barang Berdasarkan Data Persediaan dan Jumlah Pemesanan Pada CV. Papadan Mama Pastries. *Jurnal Mantik Penusa*, 1(2), 16–21. <https://ezp.lib.unimelb.edu.au/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=ffh&AN=2008-10-Aa4022&site=eds-live&scope=site>
- Mujahidin, S., Prasetyo, B., & Utomo, M. C. C. 2022. Implementasi Analisis Sentimen Masyarakat Mengenai Kenaikan Harga BBM Pada Komentar Youtube Dengan Metode Gaussian naïve bayes. *Jurnal Vocational Teknik Elektronika Dan Informatika*, 10(3), 17–24. <http://ejournal.unp.ac.id/index.php/voteknika/index>
- Muthia, D. A. 2017. Analisis Sentimen Pada Review Restoran Dengan Teks Bahasa Indonesia Menggunakan Algoritma Naive Bayes. *Jurnal Ilmu Pengetahuan Dan Teknologi Komputer*, 2(2), 39–45.
- Purwiantono, F. E., & Aditya, A. 2020. Klasifikasi Sentimen Sara, Hoaks Dan Radikal Pada Postingan Media Sosial Menggunakan Algoritma Naive Bayes Multinomial Text. *Jurnal Tekno Kompak*, 14(2), 68. <https://doi.org/10.33365/jtk.v14i2.709>
- Reddy, D., Arifianto, D., & Lusiana, D. 2022. *Jurnal Smart Teknologi Analisis Sentimen Pada Pelayanan Jaringan Internet Indihome Dengan Metode Multinomial Naïve Bayes Masa Pandemi Covid-19 Sentiment Analysis on Indihome Internet Network Services Using The Multinomial Naïve Bayes Method During The Cov*. 3(6), 612–623.

- Suprianto, S. 2020. Implementasi Algoritma Naive Bayes Untuk Menentukan Lokasi Strategis Dalam Membuka Usaha Menengah Ke Bawah di Kota Medan (Studi Kasus: Disperindag Kota Medan). *Jurnal Sistem Komputer Dan Informatika (JSON)*, 1(2), 125. <https://doi.org/10.30865/json.v1i2.1939>
- Widodo, Y. B., Anggraeini, S. A., & Sutabri, T. 2021. Perancangan Sistem Pakar Diagnosis Penyakit Diabetes Berbasis Web Menggunakan Algoritma Naive Bayes. *Jurnal Teknologi Informatika Dan Komputer*, 7(1), 112–123. <https://doi.org/10.37012/jtik.v7i1.507>
- Yunita, D., Rosyani, P., & Amalia, R. 2018. Analisa Prestasi Siswa Berdasarkan Kedisiplinan, Nilai Hasil Belajar, Sosial Ekonomi dan Aktivitas Organisasi Menggunakan Algoritma Naïve Bayes. *Jurnal Informatika Universitas Pamulang*, 3(4), 209. <https://doi.org/10.32493/informatika.v3i4.2032>
- Yuliati, I. F. 2020. Penerapan Metode Machine Learning dalam Klasifikasi Risiko Kejadian Berat Badan Lahir Rendah di Indonesia. *Jurnal Manajemen, Teknik Informatika, dan Rekayasa Komputer*, 20(2), 417-426
- Rifqo, M. H & Wijaya, A. 2017. Implementasi Algoritma Naive Bayes Dalam Penentuan Pemberian Kredit. *Jurnal Pseudocode*, IV(2), (120-128). <http://www.ejournal.unib.ac.id/index.php/pseudocode>

BAB 7

K MEDOIDS

Oleh Ahmad Jurnaidi Wahidin

7.1 Pendahuluan

Terdapat beberapa teknik dalam *unsupervised machine learning*, salah satunya adalah clustering. Clustering merupakan sebuah teknik yang digunakan untuk mengelompokkan data menjadi beberapa kelompok sedemikian rupa sehingga data yang terdapat satu kelompok memiliki kesamaan maksimum dan data antar kelompok memiliki kesamaan minimum.

Machine learning membuat komputer melakukan tugas tanpa harus memprogramnya secara terpisah. Yang diinginkan adalah komputer mempelajari pengetahuan dan menggunakannya dalam berbagai situasi.

Bab ini membahas teknik clustering yaitu K-Medoids, dimana metode K-Medoids sangat mirip dengan K-Means dengan fungsi keduanya yang sama. K-medoids juga merupakan salah satu metode dalam unsupervised learning.

7.2 K Medoids

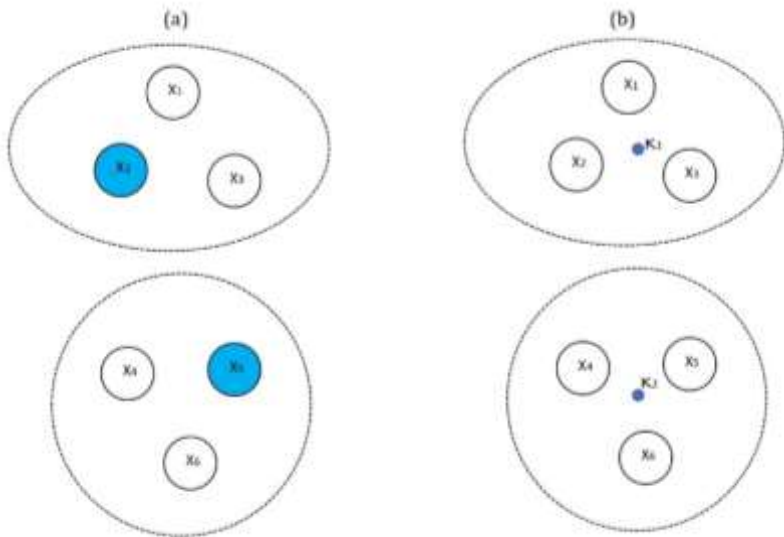
Pada tahun 1987, Leonard Kaufman dan Peter J. Rosseuw mengembangkan algoritma K-medoids, yang sering disebut dengan *Partitioning Around Medoid Algorithm* (PAM). Metode K-Means dan K-Medoid sama-sama termasuk dalam metode partisi (*partitioning*), yaitu metode pengelompokan data menjadi beberapa cluster tanpa struktur hirarki di antara keduanya. Dan proses clustering disetel untuk menemukan centroid cluster (titik tengah) yang meminimalkan jarak antara anggota cluster dan titik tengah. Namun, untuk K-Medoids, fokusnya adalah pada anggota

cluster itu sendiri. Pada metode k-medoids titik pusat cluster disebut dengan medoid.

K-Medoids dibuat dengan tujuan untuk mengatasi kelemahan K-Means yang sensitif terhadap outlier, karena suatu objek dengan nilai yang besar dapat menyimpang secara signifikan dari distribusi data (Han and Kamber, 2006). K Medoid lebih baik dibandingkan dengan K Means dan memiliki kinerja yang lebih baik ketika jumlah data yang digunakan sedikit. dan mengatasi kelemahan k-means yang sensitif terhadap noise ataupun outlier. Keuntungannya adalah hasil dari proses clustering tidak bergantung pada urutan data pada dataset dan K-medoids jarang terjadi perubahan cluster terhadap perubahan nilai pusat cluster, sedangkan untuk k means sangat berbeda jauh dimana jika terjadi perubahan pada nilai cluster sangat rentang terhadap perubahan cluster.

K-medoids menggunakan objek pada kumpulan objek yang mewakili sebuah cluster. Objek yang mewakili sebuah cluster disebut dengan medoid. Medoid adalah objek terletak terpusat di dalam suatu cluster dengan jarak minimum ke titik lain sehingga robust terhadap outlier. Cluster dibangun dengan menghitung kedekatan yang dimiliki antara medoid dengan objek non medoid (Han, Kamber and Pei, 2012). Tahapan pertama K-medoids adalah menemukan titik (medoid) yang paling representatif dalam kumpulan data dengan menghitung jarak cluster di semua kemungkinan kombinasi medoid sedemikian rupa sehingga jarak antar titik dalam suatu cluster kecil sedangkan jarak antar titik dalam kelompok besar.

Untuk menentukan pusat klaster pada setiap klaster metode K-Means menggunakan nilai rata-rata (mean) sedangkan metode K-Medoids menggunakan objek sebagai perwakilan (medoid)



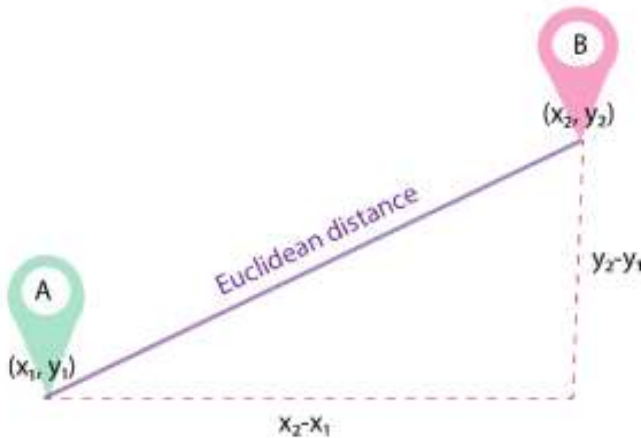
Gambar 7.1 : Ilustrasi clustering K-Medoids dan K-means

Ilustrasi pada gambar 7.1 menjelaskan clustering pada k-medoids (a) dan k means (b), dengan 6 data `yang terbagi kedalam 2 kelompok. Dimana cluster 1 berisikan $\{X1, X2, X3\}$ sedangkan untuk cluster 2 berisikan $\{X4, X5, X6\}$, dengan hasil pengelompokkan yang sama namun pusat clusternya berbeda antara k-medoids dan k-means, dimana k-medoids pusat dari cluster 1 terletak pada $X2$ dan pusat dari cluster 2 terletak pada $X4$. Sedangkan untuk k-means $C1$ adalah pusat dari cluster 1 dan $C2$ adalah pusat dari cluster 2. Pembagian kelompok tidak selalu sama antara k-medoids dan k means seperti pada ilustrasi gambar 7.1, untuk beberapa kasus hasilnya bisa saja berbeda.

Beberapa kemiripan K-medoids dengan k-means yang pertama adalah bersifat non hierarkis, selanjutnya jumlah cluster (k) tidak berubah atau tetap sama dari tahap awal sampai proses pengelompokkan berakhir. Dalam tahapan iterasi titik pusat cluster

(medoid) dan anggota cluster akan berubah-ubah, namun untuk jumlah cluster akan tetap sama.

Euclidean Distance adalah salah satu metode untuk menghitung jarak, digunakan untuk mengukur jarak dari 2 titik yang terdapat pada Euclidean space (Wanto *et al.*, 2020). Contoh euclidean distance antara 2 titik ditampilkan pada gambar 7.2.



Gambar 7.2 : Ilustrasi Euclidean Distance

Jarak A dan B dapat diukur menggunakan rumus euclidean distance dengan persamaan (1).

$$d(x, y) = \sqrt{\sum_{i=1}^n (xi - yi)^2} \quad (1)$$

Keterangan:

d= jarak antara x dan y

x = data pusat cluster

y = data pada atribut

i = setiap data

n = jumlah data

xi = data pada pusat cluster ke i

yi = data pada setiap data ke i

7.2.1 Langkah - Langkah K-Medoids

Terdapat dua tahapan dalam proses K-medoids, inisialisasi merupakan tahapan pertama dimana menentukan posisi-posisi dari medoid yang akan digunakan pertama kalinya. Proses dari K-Medoids adalah sebagai berikut:

1. Tentukan jumlah *cluster* (k), dimana jumlah k tidak boleh melebihi jumlah data (n).
2. Tentukan medoid awal secara acak sebanyak jumlah k , dimana posisi medoids harus terletak pada salah satu data dan jika terdapat lebih dari satu medoid tidak berada pada satu data yang sama.
3. Lakukan klasterisasi awal dengan cara menghitung jarak setiap objek dengan medoid awal, kemudian tandai jarak terdekat objek ke medoid dan hitung totalnya.
4. Lakukan iterasi medoid menggunakan persamaan, tahapan iterasi adalah yang kedua dilakukan yaitu reposisi medoid dan perhitungan jarak total.
5. Hitung total simpangan (S)
Jika a merupakan jumlah jarak terdekat antara objek ke medoid awal, dan b merupakan jumlah jarak terdekat antara objek ke medoid baru, maka total simpangan dihitung menggunakan persamaan (2)
$$S = b - a \quad (2)$$
Jika $S < 0$, maka tukar objek dengan data untuk membentuk sekumpulan k baru sebagai medoid.
6. Ulangi langkah 3 sampai 5 dan hentikan jika sudah tidak terjadi perubahan anggota medoid.

7.2.2 Contoh Implementasi K Medoids

Implementasi dari k-medoids yaitu dalam suatu populasi memiliki data sejumlah n . Kemudian data pada populasi tersebut dikelompokkan kedalam cluster dengan jumlah cluster (k). Dimana nilai k harus lebih kecil dari nilai n . Pada clustering ini data pada

populasi akan dikelompokkan berdasarkan kedekatan antara satu sama lain jadi rata-rata jarak data dalam cluster minimal. Contoh implementasi k-medoids dengan n adalah 10 dan nilai k adalah 2, dengan data yang ditampilkan pada tabel 7.1.

Tabel 7.1 : Contoh Kelompok Data

Objek	X1	X2
A	4	8
B	5	7
C	2	9
D	1	5
E	8	2
F	7	4
G	8	3
H	6	1
I	9	1
J	8	2

Tahap awal yang dilakukan adalah menentukan jumlah *kyaitu* sebanyak 2.

Selanjutnya tahapan kedua, pilih C (2, 9) sebagai medoid pertama dan G (8, 3) sebagai medoid kedua.

Tabel 7.2 : Medoid Iterasi Pertama

Objek	X1	X2
A	4	8
B	5	7
C	2	9
D	1	5
E	8	2
F	7	4
G	8	3
H	6	1
I	9	1
J	8	2

Tahapan ketiga yaitu menghitung jarak objek ke setiap medoid yang telah dipilih. Dengan menggunakan rumus jarak yaitu Euclidean yang ditunjukkan pada persamaan (1)

Jarak objek ke medoid pertama

$$\begin{aligned}dA,M_1 &= \sqrt{(4-2)^2 + (8-9)^2} = \sqrt{(5)} = 2,236 \\dB,M_1 &= \sqrt{(5-2)^2 + (7-9)^2} = \sqrt{(13)} = 3,606 \\dC,M_1 &= \sqrt{(2-2)^2 + (9-9)^2} = \sqrt{(0)} = 0 \\dD,M_1 &= \sqrt{(1-2)^2 + (5-9)^2} = \sqrt{(17)} = 4,123 \\dE,M_1 &= \sqrt{(8-2)^2 + (2-9)^2} = \sqrt{(85)} = 9,220 \\dF,M_1 &= \sqrt{(7-2)^2 + (4-9)^2} = \sqrt{(50)} = 7,071 \\dG,M_1 &= \sqrt{(8-2)^2 + (3-9)^2} = \sqrt{(72)} = 8,485 \\dH,M_1 &= \sqrt{(6-2)^2 + (1-9)^2} = \sqrt{(80)} = 9,944 \\dI,M_1 &= \sqrt{(9-2)^2 + (1-9)^2} = \sqrt{(113)} = 10,630 \\dJ,M_1 &= \sqrt{(8-2)^2 + (2-9)^2} = \sqrt{(85)} = 9,220\end{aligned}$$

Jarak objek ke medoid kedua

$$\begin{aligned}dA,M_2 &= \sqrt{(4-8)^2 + (8-3)^2} = \sqrt{(41)} = 6,403 \\dB,M_2 &= \sqrt{(5-8)^2 + (7-3)^2} = \sqrt{(25)} = 5 \\dC,M_2 &= \sqrt{(2-8)^2 + (9-3)^2} = \sqrt{(72)} = 8,485 \\dD,M_2 &= \sqrt{(1-8)^2 + (5-3)^2} = \sqrt{(53)} = 7,280 \\dE,M_2 &= \sqrt{(8-8)^2 + (2-3)^2} = \sqrt{(1)} = 1 \\dF,M_2 &= \sqrt{(7-8)^2 + (4-3)^2} = \sqrt{(2)} = 1,414 \\dG,M_2 &= \sqrt{(8-8)^2 + (3-3)^2} = \sqrt{(0)} = 0 \\dH,M_2 &= \sqrt{(6-8)^2 + (1-3)^2} = \sqrt{(8)} = 2,828 \\dI,M_2 &= \sqrt{(9-8)^2 + (1-3)^2} = \sqrt{(5)} = 2,236 \\dJ,M_2 &= \sqrt{(8-8)^2 + (2-3)^2} = \sqrt{(1)} = 1\end{aligned}$$

Perhitungan jarak objek terhadap medoid pertama dan kedua dirangkum dalam tabel 7.3

Tabel 7.3 : Jarak Objek Ke Medoid Pada Iterasi Pertama

Objek	X_1	X_2	M_1	M_2
A	4	8	2,236	6,403
B	5	7	3,606	5
C	2	9	0	8,485
D	1	5	4,123	7,280
E	8	2	9,220	1
F	7	4	7,071	1,414
G	8	3	8,485	0
H	6	1	8,944	2,828
I	9	1	10,630	2,236
J	8	2	9,220	1

Jumlah jarak terdekat dengan medoid pertama sebesar $2,236 + 3,606 + 0 + 4,123 + 1 + 1,414 + 0 + 2,828 + 2,236 + 1 = 18,443$

Dimana anggota cluster pertama adalah A, B, C, dan D dan anggota cluster kedua adalah E, F, G, H, I, dan J

Tahapan keempat yaitu melakukan iterasi kembali terhadap medoid yang akan berhenti jika anggota cluster yang terbentuk dari medoid tidak mengalami perubahan lagi.

Menentukan medoid pertama pada iterasi kedua yaitu D (1, 5), dan medoid kedua pada iterasi kedua yaitu H (6, 1).

Tabel 7.4 : Medoid Iterasi Kedua

Objek	X1	X2
A	4	8
B	5	7
C	2	9
D	1	5
E	8	2
F	7	4
G	8	3
H	6	1
I	9	1
J	8	2

Kembali melakukan perhitungan jarak objek ke setiap medoid yang dipilih kembali.

Jarak objek ke medoid pertama iterasi kedua

$$dA, M_2 = \sqrt{(4 - 1)^2 + (8 - 5)^2} = \sqrt{(18)} = 4,243$$

$$dB, M_2 = \sqrt{(5 - 1)^2 + (7 - 5)^2} = \sqrt{(20)} = 4,472$$

$$dC, M_2 = \sqrt{(2 - 1)^2 + (9 - 5)^2} = \sqrt{(17)} = 4,123$$

$$dD, M_2 = \sqrt{(1 - 1)^2 + (5 - 5)^2} = \sqrt{(0)} = 0$$

$$dE, M_2 = \sqrt{(8 - 1)^2 + (2 - 5)^2} = \sqrt{(58)} = 7,616$$

$$dF, M_2 = \sqrt{(7 - 1)^2 + (4 - 5)^2} = \sqrt{(37)} = 6,083$$

$$dG, M_2 = \sqrt{(8 - 1)^2 + (3 - 5)^2} = \sqrt{(53)} = 7,280$$

$$dH, M_2 = \sqrt{(6 - 1)^2 + (1 - 5)^2} = \sqrt{(41)} = 6,403$$

$$dI, M_2 = \sqrt{(9 - 1)^2 + (1 - 5)^2} = \sqrt{(80)} = 8,944$$

$$dJ, M_2 = \sqrt{(8 - 1)^2 + (2 - 5)^2} = \sqrt{(58)} = 7,616$$

Jarak objek ke medoid kedua iterasi kedua

$$dA, M_2 = \sqrt{(4 - 6)^2 + (8 - 1)^2} = \sqrt{(53)} = 7,280$$

$$dB, M_2 = \sqrt{(5 - 6)^2 + (7 - 1)^2} = \sqrt{(37)} = 6,083$$

$$dC,M_2 = \sqrt{(2-6)^2 + (9-1)^2} = \sqrt{(80)} = 8,944$$

$$dD,M_2 = \sqrt{(1-6)^2 + (5-1)^2} = \sqrt{(41)} = 6,403$$

$$dE,M_2 = \sqrt{(8-6)^2 + (2-1)^2} = \sqrt{(5)} = 2,236$$

$$dF,M_2 = \sqrt{(7-6)^2 + (4-1)^2} = \sqrt{(10)} = 3,162$$

$$dG,M_2 = \sqrt{(8-6)^2 + (3-1)^2} = \sqrt{(8)} = 2,828$$

$$dH,M_2 = \sqrt{(6-6)^2 + (1-1)^2} = \sqrt{(0)} = 0$$

$$dI,M_2 = \sqrt{(9-6)^2 + (1-1)^2} = \sqrt{9} = 3$$

$$dJ,M_2 = \sqrt{(8-6)^2 + (2-1)^2} = \sqrt{(5)} = 2,236$$

Perhitungan jarak objek terhadap medoid pertama dan kedua dalam iterasi kedua dirangkum dalam tabel 7.5.

Tabel 7.5 : Jarak Objek Ke Medoid Pada Iterasi Kedua

Objek	X ₁	X ₂	M ₁	M ₂
A	4	8	4,243	7,280
B	5	7	4,472	6,083
C	2	9	4,123	8,944
D	1	5	0	6,403
E	8	2	7,616	2,236
F	7	4	6,083	3,162
G	8	3	7,280	2,828
H	6	1	6,403	0
I	9	1	8,944	3
J	8	2	7,616	2,236

Jumlah jarak terdekat dengan medoid pertama sebesar
 $4,243 + 4,472 + 4,123 + 0 + 2,236 + 3,162 + 2,828 + 0 + 3 + 2,236 = 26,301$

Dan didapat anggota cluster pada iterasi kedua adalah A, B, C, D untuk cluster pertama sedangkan E, F, G, H, I, J untuk cluster kedua.

Tahapan terakhir yaitu menghitung total simpangan menggunakan persamaan (2), dimana $S = b - a$.

$$S = 26,301 - 18,443$$

$$S = 7.857$$

Dengan didapat nilai $b > a$, maka iterasi dihentikan. Sehingga, anggota cluster yang terbentuk pada masing-masing medoid adalah: anggota cluster pertama yaitu objek A, B, C, dan D sedangkan anggota cluster kedua yaitu objek E, F, G, H, I, dan J.

DAFTAR PUSTAKA

- Han, J. and Kamber, M. 2006. *Data Mining: Concepts and Techniques, Soft Computing*. doi: 10.1007/978-3-642-19721-5.
- Han, J. W., Kamber, M. and Pei, J. 2012. 'Clustering analysis', *Data Mining: Concept and Technique, MK imprint of Elsevier, New York*, pp. 478–490.
- Wanto, A. *et al.* 2020. *Data Mining : Algoritma dan Implementasi*. Yayasan Kita Menulis.

BAB 8

GENETIC ALGORITHM

Oleh Gusti Eka Yuliasuti

8.1 Pendahuluan

Kehidupan di alam semesta telah memberikan dampak positif bagi para peneliti, salah satunya adalah banyak terciptanya algoritma komputasi untuk menyelesaikan permasalahan sehari-hari. Algoritma yang terinspirasi dari kehidupan di alam semesta ini disebut dengan Algoritma Evolusi. Adapun salah satu contoh algoritma dari Algoritma Evolusi adalah Algoritma Genetika.

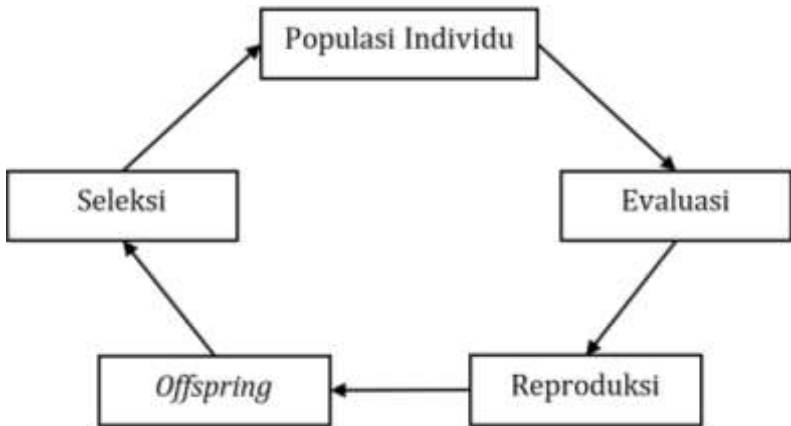
Algoritma Genetika atau *Genetic Algorithm* (GA) merupakan suatu algoritma yang terinspirasi secara biologis dengan konsep “*Survival of the Fittest*”. GA dikembangkan dengan dasar prinsip alamiah teori evolusi Darwin (Zukhri, 2014). Inti dari konsep teori evolusi adalah beberapa organisme yang disebut populasi memiliki kemampuan untuk bereproduksi dan menghasilkan keturunan yang lebih baik.

Pada awalnya, GA hanya digunakan untuk memahami bagaimana proses seleksi alam berlangsung. GA tidak dirancang untuk menjadi sebuah algoritma komputasi, namun seiring berjalannya waktu GA menjadi salah satu algoritma komputasi paling populer pada lingkup Algoritma Evolusi.

GA termasuk ke dalam algoritma pencarian heuristik yang adaptif dan termasuk yang terbesar dari Algoritma Evolusi. Hal tersebut merupakan eksploitasi dari pencarian acak yang dilengkapi dengan data historis untuk mengarahkan pencarian ke wilayah kinerja yang lebih baik di ruang solusi. Biasanya GA digunakan untuk menghasilkan solusi berkualitas tinggi dalam permasalahan optimasi dan pencarian.

Populasi awal dalam GA dibentuk dengan cara membangkitkan beberapa individu secara acak, yang kemudian dapat disebut dengan kromosom. Kromosom merupakan representasi solusi untuk permasalahan yang akan diselesaikan (Yuliastuti, Mahmudy and Rizki, 2017b). Semua informasi genetik disimpan dalam kromosom. Kromosom tersebut dibangun dari *Dioxy Ribo Nucleic Acid* (DNA). Kromosom dibagi menjadi beberapa bagian yang disebut dengan gen. Gen akan menentukan sifat-sifat spesies tersebut yang nantinya membentuk karakteristik individu. Kemungkinan gen untuk satu sifat disebut dengan alel dan setiap gen dapat mengambil alel yang berbeda.

Setiap kromosom ini memiliki nilai fungsi kebugaran (*fitness*) tersendiri. Nilai *fitness* merupakan suatu titik referensi untuk mengukur kualitas dari individu tersebut menjadi calon solusi yang baru (Mahmudy, Marian, & Luong, 2012). Sebagian individu yang terpilih akan melakukan proses reproduksi untuk menghasilkan individu baru melalui dua cara yakni menggunakan operator tukar silang (*crossover*) dan operator mutasi (*mutation*). Individu-individu baru yang dihasilkan dari proses reproduksi akan membentuk populasi baru pada generasi berikutnya. Individu-individu baru ini juga dapat disebut sebagai anak (*offspring*). Pada saat yang sama, individu yang terpilih pada generasi sebelumnya melakukan reproduksi dan bertindak sebagai induk (*parent*). Adapun siklus dari GA seperti ditunjukkan pada Gambar 8.1.



Gambar 8.3 : Siklus *Genetic Algorithm*

Pada GA, terdapat beberapa parameter yang menjadi kendala yaitu ukuran populasi, probabilitas *crossover*, probabilitas *mutation* dan berlangsungnya proses komputasi hingga kondisi terpenuhi. Ukuran populasi pada setiap generasi akan selalu tetap atau tidak berubah. Populasi pada generasi berikutnya terbentuk dengan cara menyeleksi individu-individu pada generasi sebelumnya berdasarkan nilai *fitness* yang dihasilkan. Setelah melalui evolusi beberapa generasi, tentu akan solusi yang optimal (Mahmudy, 2015a). Proses komputasi pada GA akan berhenti jika telah memenuhi kondisi tertentu. Terdapat beberapa kriteria yang dapat digunakan untuk menentukan kondisi berhenti tersebut, antara lain:

1. Iterasi berhenti sampai generasi ke- n . Nilai n tersebut ditentukan terlebih dahulu berdasarkan penelitian sebelumnya atau referensi yang menjadi rujukan (Yogeswaran, Ponnambalam and K., 2009).
2. Iterasi berhenti jika tidak ditemukan solusi yang optimal dan lebih baik untuk n generasi berturut-turut. Nilai n tersebut tidak dapat diprediksi (Mahmudy et al., 2012).

- Iterasi berhenti ketika t satuan waktu telah tercapai. Nilai t tersebut awalnya ditentukan berdasarkan penelitian sebelumnya atau referensi yang menjadi rujukan. Selain itu, nilai t dapat ditentukan berdasarkan kebutuhan dan tujuan pengujian. Pada umumnya penentuan t satuan waktu ini digunakan untuk membandingkan performa antar algoritma (Mahmudy, 2014).

Secara singkat, konversi dari proses alamiah evolusi menjadi proses komputasi pada GA dapat dilihat pada Tabel 8.1.

Tabel 8.2 : Konversi Proses Alamiah Menjadi Proses Komputasi

Proses Alamiah	Proses Komputasi
Individu	Solusi suatu permasalahan
Populasi	Seperangkat solusi suatu permasalahan
Nilai <i>Fitness</i>	Kualitas solusi
Kromosom	Representasi (pengodean) dari solusi
Gen	Bagian kecil dari representasi solusi
Pertumbuhan	Pendekodean dari solusi
Penyilangan	Operator genetika <i>crossover</i>
Mutasi	Operator genetika <i>mutation</i>
Seleksi Alam	Memilih solusi sementara berdasarkan kualitasnya

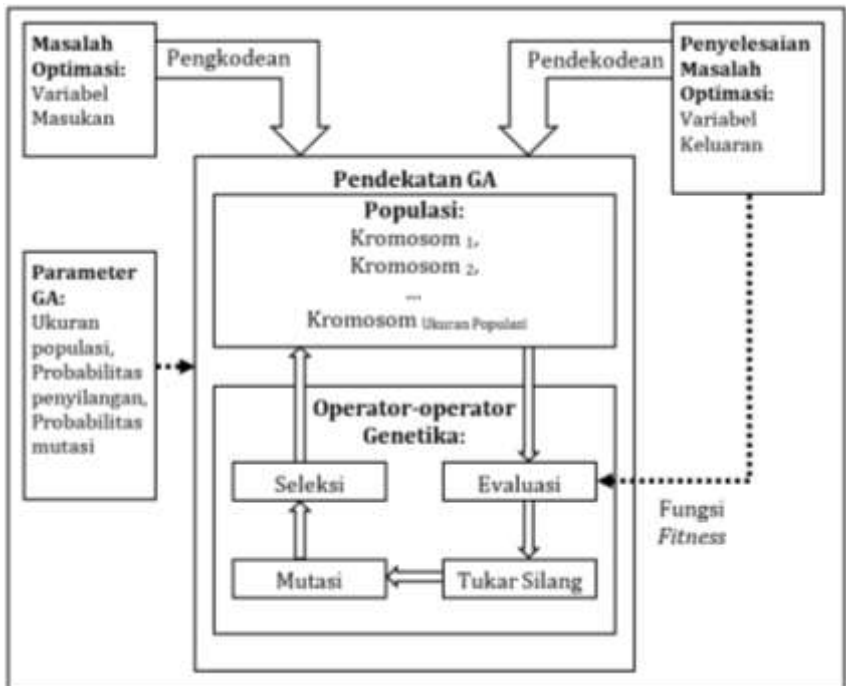
Sumber: (Zukhri, 2014)

Menurut Gen dan Cheng (Gen and Cheng, 1997) terdapat beberapa kelebihan dari GA dibandingkan dengan algoritma lainnya, antara lain:

- Sederhana, karena hanya perlu sedikit perhitungan matematis untuk menghasilkan solusi dalam menyelesaikan suatu permasalahan.
- Efektif, karena terdapat operator-operator genetika yang digunakan untuk melakukan pencarian solusi secara global.

3. Fleksibel, karena dapat dengan mudah digabungkan dengan algoritma lain untuk meningkatkan efektivitas pencarian solusinya.

Keberhasilan dalam menyelesaikan suatu permasalahan dengan menggunakan GA tersebut ditentukan oleh representasi solusi di awal dan juga operator-operator genetika yang digunakan (Michalewicz, 1996). Adapun struktur penerapan GA dalam menyelesaikan suatu permasalahan seperti ditunjukkan pada Gambar 8.2.



Gambar 8.2 : Skema Genetic Algorithm

8.2 Representasi Kromosom

Representasi kromosom merupakan hal yang utama dalam menyelesaikan suatu permasalahan dengan menggunakan GA. Semua jenis permasalahan tidak dapat diselesaikan dengan menggunakan representasi yang sama, karena pada dasarnya representasi bergantung pada jenis permasalahan serta solusi yang diharapkan. Terdapat beberapa bentuk umum representasi solusi untuk suatu permasalahan, antara lain:

a. Pengodean Biner untuk Representasi Bilangan Bulat

Bilangan bulat dapat direpresentasikan dalam kode biner jika jumlah bilangan bulat yang mungkin sama dengan pangkat angka yang diberikan misalnya 2, 4, 8, 16, dst. Setiap kode biner masing-masing dapat dipetakan ke dalam semua bilangan bulat. Untuk lebih jelasnya dapat melihat contoh pada Tabel 8.2.

Tabel 8.2 : Pengodean Biner untuk Representasi Bilangan Bulat

Kode Biner	Bilangan Bulat
0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 1	1
0 0 0 0 0 0 1 1	2
0 0 0 0 0 0 1 1	3
...	...
1 1 1 1 1 1 1 1	255

Sumber: (Zukhri, 2014)

b. Pengodean Biner untuk Representasi Bilangan Riil

Pengodean biner juga dapat digunakan untuk mewakili bilangan riil. Jika suatu bilangan riil x berada diantara x_{min} dan x_{max} , maka pengodean biner yang mungkin dapat dilakukan adalah dengan memetakan biner terkecil (0 0 0 ... 0) untuk x_{min} dan biner terbesar (1 1 1 ... 1) untuk x_{max} . Setelah itu memetakan semua kode biner di antara biner terendah dan biner tertinggi ke bilangan riil antara x_{min} dan x_{max} dengan interval sebesar $(x_{max} - x_{min}) / (2^n - 1)$. Sebagai contoh, konversi bilangan riil dalam interval 10 sampai 30 menggunakan 3 bit dan 4 bit kode biner dapat dilihat pada Tabel 8.3. Kekurangan pada pengodean ini yakni lamanya waktu komputasi untuk melakukan konversi.

Tabel 8.3 : Pengodean Biner untuk Representasi Bilangan Riil

3 Bit		4 Bit	
Kode Biner	Bilangan Riil	Kode Biner	Bilangan Riil
0 0 0	10,0000	0 0 0 0	10,0000
0 0 1	12,8571	0 0 0 1	11,3333
...	...	...	...
1 1 1	30,0000	1 1 1 1	30,0000

Sumber: (Zukhri, 2014)

c. Pengodean Riil

Pengodean riil ini mengatasi keterbatasan pengodean biner dimana pengodean ini tidak membutuhkan waktu banyak dalam hal konversi dari biner ke riil maupun sebaliknya (Mahmudy, 2015a). Pengodean riil biasanya digunakan untuk menyelesaikan permasalahan optimasi fungsi. Selain itu, bisa juga digunakan untuk memecahkan permasalahan *job shop scheduling* (Mahmudy, Marian, & Luong, 2013). Contoh pengodean riil dapat dilihat pada Tabel 8.4.

Tabel 8.4 : Pengodean Riil

Kromosom			
Gen ₁	Gen ₂	Gen ₃	Gen ₄
1,945	5,892	4,213	8,527

Sumber: (Mahmudy, 2015a)

d. Pengodean Integer

Pengodean *integer* merupakan pengodean dengan bilangan bulat dimana bilangan yang sama dapat muncul lebih dari satu kali (Mahmudy, 2015a). Pengodean *integer* dapat digunakan dalam menyelesaikan permasalahan *job shop scheduling* (Mahmudy, 2015b). Tabel 8.5 merupakan contoh pengodean *integer*.

Tabel 8.5 : Pengodean Integer

Kromosom					
Gen ₁	Gen ₂	Gen ₃	Gen ₄	Gen ₅	Gen ₆
2	1	2	3	1	4

Sumber: (Mahmudy, 2015a)

e. Pengodean Permutasi

Pengodean permutasi hampir sama dengan pengodean *integer* yakni dengan menggunakan bilangan bulat, namun bedanya pengodean permutasi ini setiap bilangan hanya dapat muncul satu kali saja karena menunjukkan suatu urutan (Mahmudy, 2015a). Pengodean permutasi sangat cocok untuk menyelesaikan permasalahan *Travelling Salesman problem* (TSP) (Yuliasuti, Mahmudy and Rizki, 2017a) dan juga dapat menyelesaikan permasalahan distribusi (Lesmawati, Rahmi and Mahmudy, 2016). Contoh pengodean permutasi ditunjukkan pada Tabel 8.6.

Tabel 8.6 : Pengodean Permutasi

Kromosom					
Gen ₁	Gen ₂	Gen ₃	Gen ₄	Gen ₅	Gen ₆
3	5	1	2	4	6

Sumber: (Mahmudy, 2015a)

8.3 Proses Evaluasi

Proses evaluasi ini dilakukan untuk mengetahui kualitas individu tersebut dengan menghitung nilai *fitness*nya (Mahmudy, 2015a). Nilai *fitness* ini diberikan pada setiap individu yang menunjukkan kemampuannya dalam berkompetisi. Individu yang memiliki nilai *fitness* optimal akan dipertahankan. Semakin besar nilai *fitness*nya, maka semakin besar pula peluang individu tersebut menjadi calon solusi yang baru. Perhitungan nilai *fitness* ini tidak memiliki formula tertentu, karena pada dasarnya formula untuk nilai *fitness* bergantung dengan permasalahan yang akan diselesaikan (Yulianti *et al.*, 2020).

8.4 Proses Reproduksi

Setelah populasi awal pada generasi pertama dibangkitkan, GA mengembangkan generasi-generasi berikutnya dengan proses reproduksi. Proses reproduksi ini dilakukan untuk menghasilkan keturunan (*offspring*) (Yulianti, Mahmudy and Rizki, 2017b). Adapun proses reproduksi mencakup dua jenis operator, yakni metode pindah silang (*crossover*) dan mutasi (*mutation*) (Yulianti *et al.*, 2019). Envji

8.4.1 Crossover

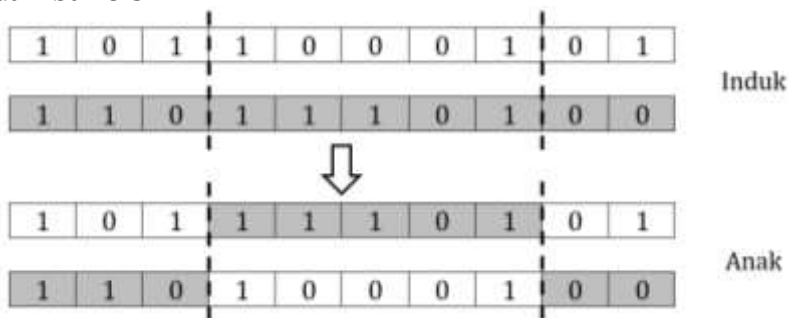
Crossover merupakan salah satu operator dalam GA yang bertujuan untuk menghasilkan individu baru yang mewarisi sifat dari induknya. Pada proses *crossover* memerlukan dua individu yang dipilih secara acak menggunakan operator seleksi untuk bertindak sebagai induk (Mahmudy, 2015a). Operator *crossover* mengacu pada pertukaran beberapa gen antara dua individu yang bertindak sebagai induk untuk menghasilkan dua anak.

Seperti di kehidupan nyata, materi genetik dari kedua induk sebagai orang tua bercampur dalam proses reproduksi seksual. Biasanya kromosom dipisah dan digabungkan secara acak, dengan konsekuensi bahwa beberapa gen anak berasal dari satu induk sementara anak yang lain berasal dari induk lainnya.

Terdapat berbagai macam teknik dalam *crossover*, antara lain (Zukhri, 2014):

a. Penyilangan N-Titik (*N-Point Crossover*)

Teknik ini merupakan teknik yang paling sederhana dimana kromosom induk akan dipotong menjadi $N+1$ bagian. Nilai N ini harus ditentukan terlebih dahulu atau dapat dibangkitkan secara acak. Sebelumnya, jumlah dan letak titik-titik yang digunakan sebagai syarat batas harus ditentukan terlebih dahulu. Hasil dari teknik ini adalah kromosom anak pertama akan mewariskan bagian potongan dalam urutan ganjil dari induk yang pertama dan mewariskan bagian potongan dalam urutan genap dari induk yang kedua. Hal tersebut juga berlaku seterusnya. Dengan representasi kromosom nilai biner, akan cocok menggunakan teknik ini sebagai operator *crossover*. Ilustrasi Teknik *N-Point Crossover* seperti ditunjukkan pada Gambar 8.3.

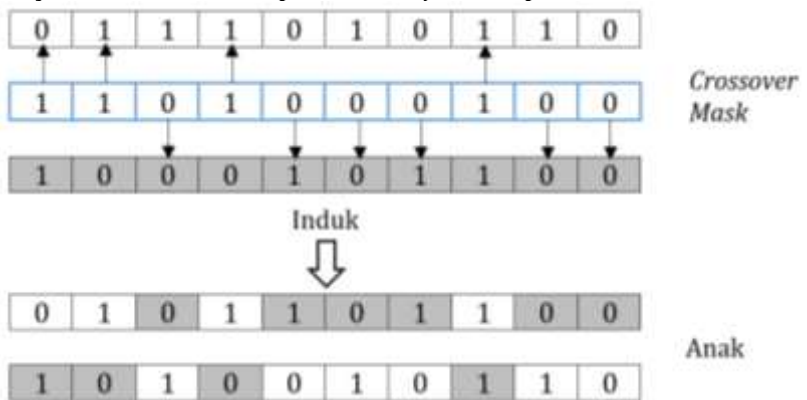


Gambar 8.3 : Ilustrasi Teknik *N-Point Crossover*

b. Penyilangan Seragam (*Uniform Crossover*)

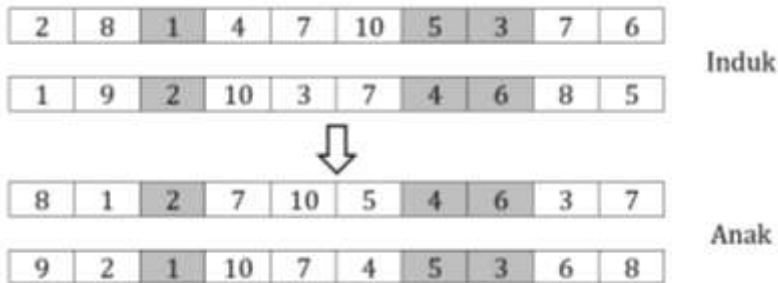
Dalam teknik ini, perlu membangkitkan biner lain secara acak sebagai topeng penyilangan (*crossover mask*) guna menentukan pewarisan induk pada setiap gen anaknya. Kromosom induk tidak perlu didekomposisi menjadi segmen-segmen untuk dilakukan penyilangan. Biasanya

dengan representasi kromosom nilai biner, akan cocok menggunakan teknik ini sebagai operator *crossover*. Sebagai contoh misalnya biner acak menghasilkan bilangan 1, maka gen tersebut diwariskan dari induk pertama. Sedangkan jika biner acak menghasilkan bilangan 0, maka gen tersebut diwariskan dari induk kedua. Ilustrasi Teknik *Uniform Crossover* seperti ditunjukkan pada Gambar 8.4.



Gambar 8.4 : Ilustrasi Teknik *Uniform Crossover*

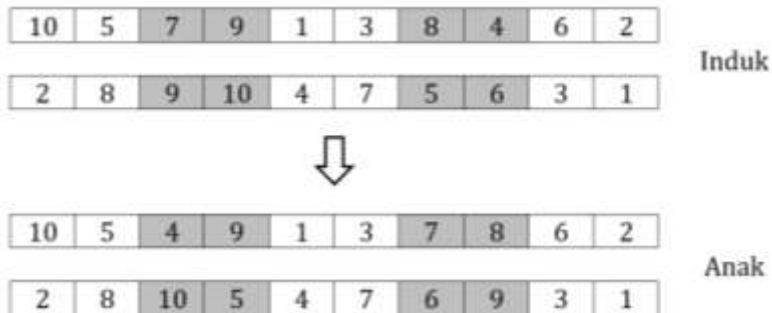
- c. **Penyilangan Berbasis Posisi (*Position Based Crossover*)**
 Dalam teknik ini, dipilih beberapa gen secara acak kemudian gen-gen posisi terpilih pada induk pertama tersebut diwariskan pada kromosom anak yang kedua. Sedangkan untuk kromosom anak pertama diambil dari gen-gen induk kedua pada posisi terpilih yang sama seperti induk pertama tadi. Dengan representasi kromosom berupa permutasi, akan cocok menggunakan teknik ini sebagai operator *crossover*. Ilustrasi Teknik *Position Based Crossover* seperti ditunjukkan pada Gambar 8.5.



Gambar 8.5 : Ilustrasi Teknik *Position Based Crossover*

d. Penyilangan Berbasis Urutan (*Order Based Crossover*)

Dalam teknik ini, dipilih posisi gen-gen secara acak kemudian untuk membentuk kromosom anak pertama diwariskan langsung dari induk pertama namun untuk posisi gen-gen terpilih tadi dilakukan pengacakan sehingga urutannya berubah. Begitu juga berlaku untuk membentuk kromosom anak kedua. Dengan representasi kromosom berupa permutasi, akan cocok menggunakan teknik ini sebagai operator *crossover*. Ilustrasi Teknik *Order Based Crossover* seperti ditunjukkan pada Gambar 8.6.



Gambar 8.6 : Ilustrasi Teknik *Order Based Crossover*

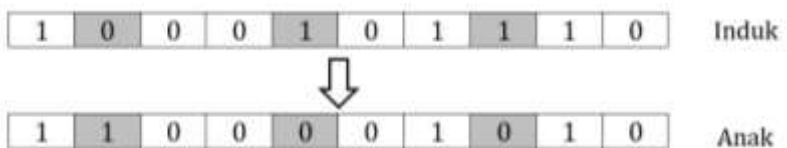
8.4.2 Mutation

Operator lain dalam GA yakni mutasi (*mutation*). Berbeda dengan *crossover* yang membutuhkan dua individu sebagai induk, pada mutasi hanya dibutuhkan satu individu yang dipilih secara acak sebagai induk (Mahmudy, 2015a). Ide utama dari *mutation* yakni menyisipkan gen acak pada keturunan untuk mempertahankan keragaman dalam populasi. Selain itu, tujuan dari *mutation* juga sebagai upaya menghindari konvergensi dini.

Operator mutasi mengacu pada perubahan nilai gen-gen tertentu dari sebuah kromosom. Operator mutasi menghasilkan keberagaman dalam populasi. Beberapa hal yang perlu diperhatikan terkait operator mutasi yakni probabilitas operator mutasi dalam setiap generasi dan juga *outlier* yang dihasilkan setelah melakukan operator mutasi. Terdapat berbagai macam teknik dalam mutasi, antara lain (Zukhri, 2014):

a. Mutasi untuk Kode Biner (*Binary Code Mutation*)

Teknik ini merupakan teknik dalam mutasi yang paling sederhana dimana setiap kromosom yang melakukan mutasi hanya perlu mengubah nilai gen pada posisi tertentu menjadi nilai inversi atau nilai kebalikannya. Ilustrasi Teknik *Binary Code Mutation* seperti ditunjukkan pada Gambar 8.7.

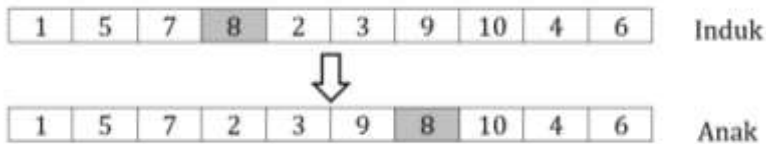


Gambar 8.7 : Ilustrasi Teknik *Binary Code Mutation*

b. Mutasi Berbasis Posisi (*Position Based Mutation*)

Dalam teknik ini, proses mutasi dilakukan dengan cara memilih posisi sebuah gen secara acak kemudian gen yang terpilih tersebut akan dipindahkan pada posisi lain secara acak juga. Dengan representasi kromosom berupa permutasi, akan cocok menggunakan teknik ini sebagai

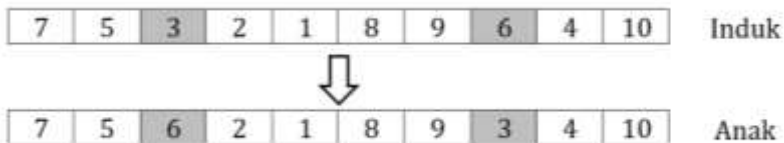
operator *mutation*. Ilustrasi Teknik *Position Based Mutation* seperti ditunjukkan pada Gambar 8.8.



Gambar 8.8 : Ilustrasi Teknik *Position Based Mutation*

c. Mutasi Berbasis Urutan (*Order Based Mutation*)

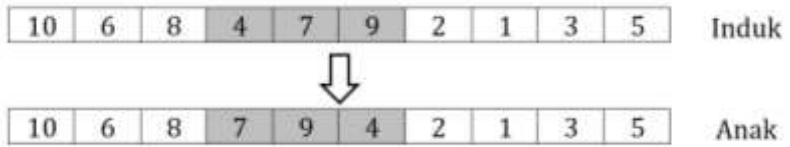
Dalam teknik ini, proses mutasi dilakukan dengan cara memilih dua posisi gen secara acak kemudian kedua gen tersebut saling bertukar posisi. Dengan representasi kromosom berupa permutasi, akan cocok menggunakan teknik ini sebagai operator *mutation*. Ilustrasi Teknik *Order Based Mutation* seperti ditunjukkan pada Gambar 8.9.



Gambar 8.9 : Ilustrasi Teknik *Order Based Mutation*

d. Mutasi Campur Aduk (*Scramble Mutation*)

Dalam teknik ini, proses mutasi dilakukan dengan cara memilih beberapa posisi gen secara acak kemudian urutan gen-gen yang terpilih tersebut akan saling ditukar secara acak pula. Dengan representasi kromosom berupa permutasi, akan cocok juga menggunakan teknik ini sebagai operator *mutation*. Ilustrasi Teknik *Scramble Mutation* seperti ditunjukkan pada Gambar 8.10.



Gambar 8.10 : Ilustrasi Teknik *Scramble Mutation*

8.5 Proses Seleksi

Proses seleksi merupakan salah satu komponen penting dalam GA yang nantinya dapat menentukan individu seperti apa yang terbentuk pada generasi berikutnya. Proses seleksi akan memandu GA untuk memilih individu sebagai solusi berdasarkan kualitasnya. Individu dengan kualitas terbaik akan bertahan dalam populasi dan melanjutkan siklus hidup pada generasi berikutnya. Tingkat konvergensi pada GA biasanya dipengaruhi oleh teknik seleksi yang digunakan. Terdapat berbagai macam teknik seleksi, antara lain (Zukhri, 2014):

a. Seleksi Sebanding dengan Nilai *Fitness* (*Roulette Wheel*)

Teknik seleksi ini merupakan teknik paling sederhana dan juga paling sering digunakan. Konsep dasar dari teknik ini yakni jika f_i merupakan nilai *fitness* untuk kromosom ke- i , maka nilai rata-rata *fitness* dalam suatu populasi berukuran N adalah sebagai berikut:

Persamaan 8.1 Nilai *Fitness* untuk Seleksi Sebanding

$$\bar{f} = \frac{\sum_{i=1}^N f_i}{N} \quad (8.1)$$

Maka probabilitas suatu kromosom untuk dipertahankan pada populasi di generasi tersebut yakni sebesar:

Persamaan 8.2 Probabilitas Kromosom untuk Seleksi Sebanding

$$P_i = \frac{f_i}{\sum_{i=1}^N f_i} = \frac{f_i}{\bar{f} N} \quad (8.2)$$

Teknik ini biasanya diterapkan dengan model roda *roulette* (*roulette wheel*). Teknik seleksi ini mengacu pada pemetaan semua kemungkinan kromosom yang ada ke dalam roda, dimana sebagian roda telah dialokasikan untuk kromosom-kromosom lain sesuai dengan nilai *fitness* masing-masing.

Sebagai ilustrasi, keliling lingkaran roda *roulette* diberi busur-busur sebanyak N . Dimana perbandingan busur sama dengan perbandingan nilai *fitness* setiap kromosom. Pada roda *roulette*, jarum roda diputar sebanyak N kali secara acak, sehingga terdapat hasil seleksi berdasarkan posisi jarum sebanyak N . Karena kemungkinan sebuah kromosom untuk terpilih bergantung pada nilai *fitness*-nya, maka sangatlah mungkin sebuah kromosom dengan nilai *fitness* paling tinggi akan terpilih lebih dari satu kali. Hal tersebut menunjukkan bahwa kromosom itu yang akan dominan pada populasi di generasi tersebut.

b. Seleksi Peringkat (*Ranking Selection*)

Dalam teknik seleksi ini, pengurutan berdasarkan nilai *fitness*-nya dilakukan untuk semua kromosom dalam suatu populasi termasuk anak hasil reproduksi (*offspring*). Kromosom dengan nilai *fitness* yang tinggi akan berada pada urutan awal dan berlaku sebaliknya. Probabilitas terpilihnya kromosom (P_i) ditentukan berdasarkan suatu fungsi distribusi, sehingga jumlah probabilitas semua kromosom sama dengan satu.

Persamaan 8.3 Nilai Fitness untuk Seleksi Peringkat

$$\sum_{i=1}^N P_i = 1 \quad (8.3)$$

c. Seleksi Turnamen (*Tournament Selection*)

Dalam metode ini, kromosom-kromosom dalam suatu populasi termasuk anak hasil reproduksi (*offspring*) dibagi ke dalam beberapa grup secara acak. Setiap grup harus beranggotakan minimal dua kromosom. Seleksi dilakukan

dengan cara mempertahankan kromosom dengan nilai *fitness* tertinggi pada setiap grup. Dengan melakukan pembagian populasi ke dalam beberapa grup dan jumlah anggota yang lebih sedikit dapat membuat proses komputasi lebih ringan.

d. Seleksi Elitisme (*Elitism Selection*)

Dalam metode seleksi ini, diperlukan minimal sebuah kromosom yang memiliki nilai *fitness* tertinggi nantinya akan dipertahankan untuk menjadi populasi pada generasi berikutnya. Pada metode ini sangat dimungkinkan terjadinya konvergensi dini karena terjebak pada nilai optimum lokal.

8.6 Modifikasi pada Genetic Algorithm

Seiring dengan perkembangan ilmu pengetahuan, banyak peneliti melakukan modifikasi pada GA bertujuan untuk mendapatkan hasil yang lebih optimal. Adapun beberapa modifikasi yang dilakukan, antara lain:

a. *Adaptive Parameter*

Modifikasi yang terjadi yakni nilai parameter dapat berubah menyesuaikan diri dimana tujuan dari perubahan tersebut agar kombinasi nilai probabilitas *crossover* dan *mutation* berganti (Rajkumar and Balamurgan, 2016). Dalam setiap iterasi, kedua nilai probabilitas dapat bertambah atau berkurang sesuai dengan kondisi pada iterasi tersebut.

b. *Random Injection*

Modifikasi yang dilakukan yakni dengan memasukkan angka acak sebagai suntikan (*injection*) di tengah proses perhitungan. *Random injection* dilakukan bila setelah beberapa generasi tidak didapatkan peningkatan hasil (Seisarrina, Cholissodin and Nurwarsito, 2018). *Injection*

disini bertujuan untuk mendiversifikasi kromosom dengan harapan menemukan solusi akhir yang optimal.

c. *Improved Reproduction*

Modifikasi yang dilakukan yakni dengan penggunaan operator reproduksi secara berganti-ganti. Sebelum dilakukan pergantian operator tersebut, perlu ditentukan terlebih dahulu terkait kombinasi probabilitas antara kedua operator baik *crossover* maupun *mutation* (Wang *et al.*, 2020). Kemudian menentukan batas atau interval untuk dilakukannya pergantian dari kedua operator tersebut.

d. *Guided Mutation*

Perubahan yang dilakukan merupakan pedoman untuk melakukan *mutation*. Jika *mutation* biasanya dipilih secara acak, maka pada model ini ditentukan berdasarkan seperangkat aturan yang telah dibuat (Sapru, Reddy and Sivaselvan, 2010). Modifikasi ini akan melakukan percobaan untuk semua kemungkinan yang ada, kemudian dari sekian percobaan tersebut akan diambil yang terbaik untuk masuk *offspring* (Ray *et al.*, 2010). Namun, kelemahan pada modifikasi ini yakni proses komputasi yang cukup lama.

e. *Parallel Population*

Modifikasi yang dilakukan yakni membagi populasi menjadi bagian-bagian yang lebih kecil atau selanjutnya dapat disebut sebagai sub-populasi (Cochran, Horng and Fowler, 2003). Proses pembentukan populasi terjadi seperti pada umumnya, setiap sub-populasi dijalankan pada waktu yang sama kemudian pada generasi tertentu dilakukan migrasi atau perpindahan satu kromosom ke sub-populasi lainnya (Chang, Chen and Lin, 2005).

DAFTAR PUSTAKA

- Chang, P. C., Chen, S. H. and Lin, K. L. 2005. 'Two-Phase Sub Population Genetic Algorithm for Parallel Machine-Scheduling Problem', *Expert Systems with Applications*, 29(3), pp. 705–712. doi: 10.1016/j.eswa.2005.04.033.
- Cochran, J. K., Horng, S. M. and Fowler, J. W. 2003. 'A Multi-Population Genetic Algorithm to Solve Multi-Objective Scheduling Problems for Parallel Machines', *Computers and Operations Research*, 30(7), pp. 1087–1102. doi: 10.1016/S0305-0548(02)00059-X.
- Gen, M. and Cheng, R. 1997. *Genetic Algorithm and Engineering Design*. New York: John Wiley & Sons, Inc.
- Lesmawati, W., Rahmi, A. and Mahmudy, W. F. 2016. 'Optimization of Frozen Food Distribution using Genetic Algorithms', *Journal of Environmental Engineering & Sustainable Technology*, 3(1), pp. 51–58.
- Mahmudy, W. F. 2014. *Optimisation of Integrated Multi-Period Production Planning and Scheduling Problems in Flexible Manufacturing Systems (FMS) Using Hybrid Genetic Algorithms*. University of South Australia.
- Mahmudy, W. F. 2015a. *Dasar-Dasar Algoritma Evolusi*, Universitas Brawijaya. Malang: Universitas Brawijaya.
- Mahmudy, W. F. 2015b. 'Optimization of Part Type Selection and Machine Loading Problems in Flexible Manufacturing System Using Variable Neighborhood Search', *IAENG International Journal of Computer Science*, (July).
- Mahmudy, W. F., Marian, R. M. and Luong, L. H. S. 2012. 'Solving Part Type Selection and Loading Problem in Flexible Manufacturing System Using Real Coded Genetic Algorithms – Part II: Optimization', in *International Conference on Control, Automation and Robotics*. Singapore.

- Mahmudy, W. F., Marian, R. M. and Luong, L. H. S. 2013. 'Real Coded Genetic Algorithms for Solving Flexible Job-Shop Scheduling Problem – Part II: Optimization', *Advanced Materials Research*, 701, pp. 364–369. doi: 10.4028/www.scientific.net/AMR.701.364.
- Michalewicz, Z. 1996. *Genetic Algorithms + Data Structures = Evolution Programs*. Third (Ext. New York: Springer.
- Rajkumar, P. and Balamurgan, V. 2016. 'Adaptive Genetic Algorithm for a Real Time Medical images', in *International Conference on Green Engineering and Technologies (IC-GET)*.
- Ray, B. *et al.* 2010. 'An Efficient GA with Multipoint Guided Mutation for Graph Coloring Problems', *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 3(2), pp. 51–58.
- Sapru, V., Reddy, K. and Sivaselvan, B. 2010. 'Time Table Scheduling using Genetic Algorithms Employing Guided Mutation', *2010 IEEE International Conference on Computational Intelligence and Computing Research, ICCIC 2010*, pp. 335–339. doi: 10.1109/ICCIC.2010.5705788.
- Seisarrina, M. L., Cholissodin, I. and Nurwarsito, H. 2018. 'Invigilator Examination Scheduling using Partial Random Injection and Adaptive Time Variant Genetic Algorithm', *Journal of Information Technology and Computer Science*, 3(2), pp. 113–119. doi: 10.25126/jitecs.20183250.
- Wang, Z. *et al.* 2020. 'An Improved Partheno-Genetic Algorithm with Reproduction Mechanism for the Multiple Traveling Salesperson Problem', *IEEE Access*, 8, pp. 102607–102615. doi: 10.1109/ACCESS.2020.2998539.
- Yogeswaran, M., Ponnambalam, S. G. and K., T. M. 2009. 'An Efficient Hybrid Evolutionary Heuristic Using Genetic Algorithm and Simulated Annealing Algorithm to Solve Machine Loading Problem in FMS', *International Journal of Production Research*, 47(19), pp. 5421–5448.

- Yuliastuti, G. E. *et al.* 2019. 'Optimization of Multi-Product Aggregate Production Planning using Hybrid Simulated Annealing and Adaptive Genetic Algorithm', *International Journal of Advanced Computer Science and Applications*, 10(11). doi: 10.14569/IJACSA.2019.0101167.
- Yuliastuti, G. E. *et al.* 2020. 'Optimasi Rute Jaringan Mikrotik dengan Algoritme Genetika', in *Prosiding Seminar Nasional Sains dan Teknologi Terapan VIII*, pp. 209–216.
- Yuliastuti, G. E., Mahmudy, W. F. and Rizki, A. M. 2017a. 'Implementation of Genetic Algorithm to Solving Travelling Salesman Problem with Time Window (TSP-TW) for Scheduling Tourist Destinations in Malang City', *Journal of Information Technology and Computer Science*, 2(1), pp. 1–10.
- Yuliastuti, G. E., Mahmudy, W. F. and Rizki, A. M. 2017b. 'Penanganan Fuzzy Time Window pada Travelling Salesman Problem (TSP) dengan Penerapan Algoritma Genetika', *MATICS Jurnal Ilmu Komputer dan Teknologi Informasi*, 9(1), pp. 38–43.
- Zukhri, Z. 2014. *Algoritma Genetika*. Yogyakarta: Andi Publisher.

BAB 9

FUZZY C-MEANS (FCM)

Oleh Elisawati

9.1 Pendahuluan

Fuzzy clustering adalah suatu teknik dimana untuk menentukan *cluster* optimal dalam suatu ruang vektor yang di dasarkan pada bentuk normal *Euclidian* untuk jarak antar vektor. *Fuzzy clustering* sangat berguna bagi pemodelan *fuzzy* terutama dalam mengidentifikasi atura-aturan *fuzzy* (Kusumadewi and Purnomo, 2010).

Metode *Clustering* pada dasarnya mengoptimumkan pusat *cluster* (*Centroid*). Beberapa metode *clustering* yang sering digunakan antara lain yaitu :

1. Berbasis metode statistik seperti *Hirarchical Clustering Method* dan *Non Hirarchical Clustering Method*.
2. Berbasis *Fuzzy* yaitu *Fuzzy C-Means (FCM)*
3. Berbasis *Neural Network* seperti Kohonen SOM, LVQ, dan
4. Metode lain untuk optimasi *centroid* atau lebar *cluster* seperti Genetik Algoritma (GA)(Sanusi et al., 2018).

Fuzzy C-Means (FCM) adalah suatu teknik pengclusteran data yang mana keberadaan tiap-tiap titik data dalam suatu cluster di tentukan oleh derajat keanggotaan. Teknik ini pertama kali di perkenalkan oleh Jim Bezdek pada tahun 1981(Kusumadewi, 2002)

9.2 Konsep Fuzzy C-Means

Fuzzy C-Means (FCM) dikenal juga sebagai *Isodata* yang merupakan metode *clustering* atau bagian dari metode *Hard K-Means*. FCM menggunakan pengelompokan *Fuzzy* sehingga data dapat menjadi anggota dari semua kelas atau *cluster* terbentuk dengan derajat atau tingkat keanggotaan yang berbeda antara 0 hingga 1. (D. L. Rahakbauw, v. Y. I. Ilwaru and M. H. Hahury, 2017).

Konsep dasar Fuzzy C-Means (FCM) adalah menentukan pusat *cluster* untuk sebagai patokan lokasi rata-rata pada tiap-tiap *cluster*. Tiap titik data dapat memiliki derajat keanggotaan masing-masing untuk setiap *cluster*. Kemudian FCM melakukan perbaikan pusat *cluster* dan derajat keanggotaan secara berulang-ulang sesuai dengan jumlah iterasi yang ditetapkan sampai mendapatkan lokasi *cluster* yang tepat atau paling mendekati (Irwansyah and Faisal, 2002).

Output FCM merupakan deretan pusat *cluster* dan beberapa derajat keanggotaan untuk tiap titik data.

9.2.1 Algoritma Fuzzy C-Means (FCM)

Algoritma ini dimulai dengan menentukan jumlah *cluster* yang di inginkan serta menginisialisasikan nilai keanggotaan yang berisikan semua data kemudian akan di kelompokkan berdasarkan *clusternya*. Pusat-pusat *cluster* di hitung dari jarak terdekat ke titik-titik yang memiliki nilai keanggotaan lebih besar. Dengan kata lain nilai-nilai keanggotaan tersebut akan bertindak sebagai nilai bobot sementara pada suatu *cluster*.

Algoritma *Fuzzy C-Means* (FCM) adalah sebagai berikut:

1. *Input* data yang akan di*cluster* X , berupa matriks berukuran $n \times m$ (n = jumlah sampel data, m = atribut setiap data). X_{ij} = data sampel ke- i ($i = 1, 2, \dots, n$), atribut ke- j ($j = 1, 2, \dots, m$).
2. Tentukan :
 - Jumlah *cluster* $= c$;
 - Pangkat $= w$;

- Maksimum Iterasi = MaxIter;
- *Error* terkecil yang di harapkan = ξ ;
- Fungsi Objektif awal = $P_0 = 0$;
- Iterasi awal = $t = 1$;

3. Bangkitkan bilangan random μ_{ik} , $i=1,2,\dots,n$; $k=1,2,\dots,c$; sebagai elemen-elemen matriks partisi awal U.

Hitung jumlah setiap kolom :

$$Q_i = \sum_{k=1}^c \mu_{ik} = 1 \quad (9.1)$$

Persamaan (9.2) menjelaskan matrik awal yang terbentuk dari setiap data yang akan di masukkan kedalam perhitungan. Jumlah *cluster* yang akan di bentuk digambarkan oleh $\mu_{11}(X_1)$ sampai dengan $\mu_{1c}(X_1)$, sedangkan jumlah data yang akan di kelompokkan digambarkan oleh $\mu_{11}(X_1)$ sampai dengan $\mu_{n1}(X_1)$.

$$U = \begin{bmatrix} \mu_{11}(X_1) & \mu_{12}(X_1) & \cdots & \mu_{1c}(X_1) \\ \mu_{21}(X_2) & \mu_{22}(X_2) & \cdots & \mu_{2c}(X_2) \\ \vdots & \vdots & \vdots & \vdots \\ \mu_{n1}(X_n) & \mu_{n2}(X_n) & \cdots & \mu_{nc}(X_n) \end{bmatrix} \quad (9.2)$$

4. Menghitung matriks V sebagai pusat *cluster*, untuk setiap *cluster* di tunjukkan oleh persamaan (9.3).

$$V_{kj} = \frac{\sum_{i=1}^n ((\mu_{ik})^w * X_{ij})}{\sum_{i=1}^n (\mu_{ik})^w} \quad (9.3)$$

5. Menghitung fungsi objektif pada iterasi ke- t , P_t ditunjukkan persamaan (9.4).

$$P_t = \sum_{i=1}^n \sum_{k=1}^c \left(\left[\sum_{j=1}^m (X_{ik} - V_{kj})^2 \right] (\mu_{ik})^w \right) \quad (9.4)$$

6. Persamaan (9.5) digunakan untuk melakukan perhitungan matrik partisi U , yang berfungsi untuk membuat derajat keanggotaan baru.

$$\mu_{ik} = \frac{(\sum_{j=1}^m (X_{ij} - V_{kj})^2)^{\frac{-1}{w-1}}}{\sum_{k=1}^c (\sum_{j=1}^m (X_{ij} - V_{kj})^2)^{\frac{-1}{w-1}}} \quad (9.5)$$

7. Memeriksa kriteria pemberhentian yaitu jika $|P_t - P_{t-1}| < \xi$ atau $t > \text{maxIter}$ maka iterasi di hentikan. Jika tidak memenuhi syarat di atas, maka iterasi di lanjutkan dan ulangi langkah ke 4 (Kusumadewi and Purnomo, 2010)

9.2.2 Contoh Penerapan Fuzzy C-Means (FCM)

Pada kasus ini, terdapat data di peroleh dari Pengadilan Negeri Dumai mengenai data pelanggaran lalu lintas di kota Dumai bulan Desember tahun 2017.

Tabel 9.1 : Data Pelanggaran Lalu Lintas

No	Pasal	Sepeda Motor	Mobil Pribadi	Mobil Umum	Mobil Pickup	Truk	Lain - Lain
1	280	8	2	1	0	1	2
2	281	74	1	0	3	5	0
3	283	0	1	0	0	0	0
4	284	1	0	0	0	0	0

No	Pasal	Sepeda Motor	Mobil Pribadi	Mobil Umum	Mobil Pickup	Truk	Lain - Lain
5	285	57	0	0	0	0	0
6	287	37	16	0	1	23	4
7	288	92	4	0	5	8	4
8	289	0	8	0	7	4	0
9	291	233	0	0	0	0	0
10	298	0	0	0	0	4	2
11	303	0	0	0	1	1	0
12	305	1	0	0	0	1	4

Sumber : (Elisawati, Wahyuni and Arianto, 2019)

Keterangan Tabel 9.1 :

- Pasal 280 tentang tidak dipasang tanda nomor kendaraan
- Pasal 281 tentang tidak memiliki SIM
- Pasal 283 tentang kegiatan yang mengganggu konsentrasi saat berkendara
- Pasal 284 tentang kendaraan bermotor tidak mengutamakan pejalan kaki atau pesepeda
- Pasal 285 tentang Pengemudi Sepeda Motor tidak memenuhi persyaratan teknis dan laik jalan
- Pasal 287 tentang melanggar rambu lalu lintas
- Pasal 288 tentang tidak memiliki STNK
- Pasal 289 tentang Setiap orang yang mengemudikan kendaraan bermotor atau Penumpang yang duduk disamping pengemudi yang tidak menggunakan sabuk keselamatan.
- Pasal 291 tentang tidak mengenakan helm standar nasional.
- Pasal 298 tentang kendaraan bermotor yang tidak menggunakan isyarat saat berhenti atau parkir darurat.

- Pasal 303 tentang pengemudi mobil barang untuk mengangkut orang.
- Pasal 305 tentang kendaraan bermotor yang mengangkut barang khusus yang tidak memenuhi persyaratan keselamatan.

Data yang di peroleh dibentuk 3 *cluster* dengan menggunakan *Fuzzy C-Means* (FCM) sebagai berikut :

1. Dibentuk matrik berdasarkan data pada Tabel 9.1 yang berukuran 12 x 6 sebagai berikut :

$$X_{12 \times 6} = \begin{bmatrix} 8 & 2 & 1 & 0 & 1 & 2 \\ 74 & 1 & 0 & 3 & 5 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 57 & 0 & 0 & 0 & 0 & 0 \\ 37 & 16 & 0 & 1 & 23 & 4 \\ 92 & 4 & 0 & 5 & 8 & 4 \\ 0 & 8 & 0 & 7 & 4 & 0 \\ 233 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 2 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 4 \end{bmatrix}$$

Dari data tersedia diperoleh parameter sebagai berikut :

- n : data Jenis pelanggaran yang di *cluster* sebanyak 12 pasal.
- m : data Jumlah kendaraan yang melakukan pelanggaran sebanyak 6 kendaraan.
- X : Matriks dibuat dari data di atas adalah berukuran 12x6.
- c : Banyak *cluster* adalah 3 *cluster*.
- w : Pangkat = 2
- t : jumlah Iterasi, dimulai t = 1

ξ : *Error* terkecil = 0.0001 = (10^{-4})

P_t : Fungsi Objektif dengan nilai awal $P_0=0$

MaxIter : 100

2. Membentuk matrik random/acak sebagai partisi awal U berukuran 12x6 menggunakan matlab sesuai persamaan (9.1) dan (9.2).

$$U = \begin{bmatrix} 0.3243 & 0.4595 & 0.2162 \\ 0.2500 & 0.1944 & 0.5556 \\ 0.2143 & 0.6071 & 0.1786 \\ 0.3478 & 0.3478 & 0.3043 \\ 0.1389 & 0.5000 & 0.3611 \\ 0.0952 & 0.5238 & 0.3810 \\ 0.5000 & 0.4063 & 0.0938 \\ 0.4000 & 0.5714 & 0.0286 \\ 0.4545 & 0.2727 & 0.2727 \\ 0.6842 & 0.1053 & 0.2105 \\ 0.2143 & 0.4286 & 0.3571 \\ 0.2759 & 0.4483 & 0.2759 \end{bmatrix}$$

Masing-masing baris di hitung dengan jumlah adalah satu.

Baris Ke-1 : 0.3243+0.4595+0.2162 = 1

Baris Ke-2 : 0.2500+0.1944+0.5556=1

$\vdots$

$\vdots$

Baris ke-12 : 0.2759+0.4483+0.2759=1

3. Menghitung pusat *cluster*

Pusat *cluster* dihitung berdasarkan dari hasil pemangkatan dikalikan setiap data, Kemudian dibagi dengan jumlah dari hasil pemangkatan *cluster* yang dihitung. Berdasarkan persamaan (9.3) di hitung matrik V sebagai berikut :

$$V_{11} = \frac{(0.3243)^2 * 8 + (0.2500)^2 * 74 + \dots + (0.2759)^2 * 1}{(0.3243)^2 + (0.2500)^2 + \dots + (0.2759)^2}$$

$$V_{12} = \frac{(0.3243)^2 * 2 + (0.2500)^2 * 1 + \dots + (0.2759)^2 * 0}{(0.3243)^2 + (0.2500)^2 + \dots + (0.2759)^2}$$

Perhitungan diatas dilakukan sampai selesai untuk semua elemen, maka di peroleh matrik V sebagai berikut :

$$V = \begin{bmatrix} 49,8384 & 1,7480 & 0,0670 & 1,6644 & 3,3515 & 1,5846 \\ 27,7468 & 3,8169 & 0,0949 & 1,6553 & 4,3902 & 1,3514 \\ 49,9585 & 2,5734 & 0,0430 & 1,1481 & 4,9507 & 1,0138 \end{bmatrix}$$

4. Menghitung fungsi objectif pada iterasi ke-t.

$$P_{11} = ((8 - 49,8384)^2 + (2 - 1,7480)^2 + \dots + (2 - 1,5846)^2) * 0,10517$$

$$P_{12} = ((74 - 49,8384)^2 + (1 - 1,7480)^2 + \dots + (0 - 1,5846)^2) * 0,06250$$

Kemudian dilakukan penjumlahan dari setiap elemen, sehingga mendapatkan hasil $P_t = 19040,849$. Selisih $P_t > \xi$, maka perhitungan iterasi di lanjutkan. Gunakan persamaan (9.5) untuk mencari matrik partisi U .

$$U_{11} = ((8 - 49,8384)^2 + (2 - 1,7480)^2 + \dots + (2 - 1,5846)^2)^{\frac{-1}{2-1}}$$

$$U_{12} = ((74 - 49,8384)^2 + (1 - 1,7480)^2 + \dots + (0 - 1,5846)^2)^{\frac{-1}{2-1}}$$

Dari perhitungan di atas dilakukan penjumlahan Matrik partisi U pada data tiap *cluster* kemudian untuk mencari keanggotaan baru maka hasil matrik partisi U di bagi dengan hasil total tiap data pada *cluster*.

Pada iterasi terakhir dimana iterasi di hentikan karena $t > \maxIter$ atau $|P_1 - P_2| < \xi$.

Perhitungan ini dilakukan dengan menggunakan matlab dan iterasi berhenti pada iterasi ke-17 dengan hasil matrik V sebagai berikut :

$$V = \begin{bmatrix} 71.0519 & 2.9467 & 0.0000 & 2.5501 & 6.0846 & 1.5237 \\ 2.5653 & 1.9752 & 0.1376 & 1.1182 & 2.2024 & 1.2307 \\ 232.8362 & 0.0092 & 0.0000 & 0.0027 & 0.0142 & 0.0036 \end{bmatrix}$$

Dan di peroleh matrik partisi U sebagai berikut:

$$U = \begin{bmatrix} 0.0083 & 0.9910 & 0.0007 \\ 0.9962 & 0.0032 & 0.0006 \\ 0.0030 & 0.9968 & 0.0003 \\ 0.0028 & 0.9969 & 0.0003 \\ 0.9150 & 0.0775 & 0.0075 \\ 0.5174 & 0.4611 & 0.0215 \\ 0.9265 & 0.0524 & 0.0212 \\ 0.0158 & 0.9827 & 0.0015 \\ 0.0000 & 0.0000 & 1.0000 \\ 0.0031 & 0.9967 & 0.0003 \\ 0.0026 & 0.9971 & 0.0002 \\ 0.0034 & 0.9963 & 0.0003 \end{bmatrix}$$

Pembagian *cluster* yaitu mencari nilai terbesar pada data yang di C1, C2, dan C3, terdapat pada Tabel 9.2.

Tabel 9.2 : Pembagian *Cluster*

No	Pasal	C1	C2	C3	<i>Cluster</i> dipilih	<i>Cluster</i>
1	280	0.0083	0.9910	0.0007	0.9910	2
2	281	0.9962	0.0032	0.0006	0.9962	1
3	283	0.0030	0.9968	0.0003	0.9968	2
4	284	0.0028	0.9969	0.0003	0.9969	2
5	285	0.9150	0.0775	0.0075	0.9150	1
6	287	0.5174	0.4611	0.0215	0.5174	1
7	288	0.9265	0.0524	0.0212	0.9265	1
8	289	0.0158	0.9827	0.0015	0.9827	2
9	291	0.0000	0.0000	1.0000	1.0000	3
10	298	0.0031	0.9967	0.0003	0.9967	2
11	303	0.0026	0.9971	0.0002	0.9971	2
12	305	0.0034	0.9963	0.0003	0.9963	2

Pembagian data sesuai *cluster* yang di pilih pada Tabel 9.2 sebagai berikut :

Tabel 9.3 : Pembagian data berdasarkan *cluster*

Cluster 1	Cluster 2	Cluster 3
Pasal 281	Pasal 280	Pasal 291
Pasal 285	Pasal 283	
Pasal 287	Pasal 284	
Pasal 288	Pasal 289	
	Pasal 298	
	Pasal 303	
	Pasal 305	

9.2.3 Hasil Output Fuzzy C-Means dengan Matlab

X =

```
8  2  1  0  1  2
74 1  0  3  5  0
```

0	1	0	0	0	0
1	0	0	0	0	0
57	0	0	0	0	0
37	16	0	1	23	4
92	4	0	5	8	4
0	8	0	7	4	0
233	0	0	0	0	0
0	0	0	0	4	2
0	0	0	1	1	0
1	0	0	0	1	4

U =

0.3243	0.4595	0.2162
0.2500	0.1944	0.5556
0.2143	0.6071	0.1786
0.3478	0.3478	0.3043
0.1389	0.5000	0.3611
0.0952	0.5238	0.3810
0.5000	0.4063	0.0938
0.4000	0.5714	0.0286
0.4545	0.2727	0.2727
0.6842	0.1053	0.2105
0.2143	0.4286	0.3571
0.2759	0.4483	0.2759

err =

1.0000e-04

U =

0.0083	0.9910	0.0007
0.9962	0.0032	0.0006
0.0030	0.9968	0.0003
0.0028	0.9969	0.0003

0.9150	0.0775	0.0075
0.5174	0.4611	0.0215
0.9265	0.0524	0.0212
0.0158	0.9827	0.0015
0.0000	0.0000	1.0000
0.0031	0.9967	0.0003
0.0026	0.9971	0.0002
0.0034	0.9963	0.0003

V =

71.0519	2.9467	0.0000	2.5501	6.0846	1.5237
2.5653	1.9752	0.1376	1.1182	2.2024	1.2307
232.8362	0.0092	0.0000	0.0027	0.0142	0.0036

CL =

0	1	0
1	0	0
0	1	0
0	1	0
1	0	0
1	0	0
1	0	0
0	1	0
0	0	1
0	1	0
0	1	0
0	1	0

DAFTAR PUSTAKA

- D. L. Rahakbauw, v. Y. I. Ilwaru and M. H. Hahury. 2017. *Implementasi Fuzzy C-Means Clustering Dalam Penentuan Beasiswa*.
- Elisawati, Wahyuni, D. and Arianto, A. 2019. *Analisa Clustering Pada Data Pelanggaran Lalu Lintas Di Pengadilan Negeri Dumai Dengan Menggunakan Metode K-Means, JISKa*.
- Irwansyah, E. and Faisal, M. 2002. *Advanced Clustering Teori dan Aplikasi*. Yogyakarta: Deepublish.
- Kusumadewi, S. 2002. *Analisis Desain Sistem Fuzzy Menggunakan Toolbox Matlab*. Yogyakarta: Graha Ilmu.
- Kusumadewi, S. and Purnomo, H. 2010. *Aplikasi Logika Fuzzy Untuk Mendukung Keputusan*. Yogyakarta: Graha Ilmu.
- Sanusi, W. et al. 2018. *Analisis Fuzzy C-Means dan Penerapannya Dalam Pengelompokan Kabupaten/Kota di Provinsi Sulawesi Selatan Berdasarkan Faktor-faktor Penyebab Gizi Buruk*.

BAB 10

CONFUSION MATRIX

Oleh Rima Rizqi Wijayanti

10.1 Pendahuluan

Confusion matrix adalah tabel yang digunakan untuk menggambarkan kinerja model klasifikasi (Swamynathan, 2017). Ide umumnya adalah menghitung berapa kali contoh kelas A diklasifikasikan sebagai kelas B (Géron, 2017, 2019). Misalnya, untuk mengetahui berapa kali pengklasifikasi mengukur suatu gambar buah anggur dibandingkan dengan gambar buah lainnya, confusion matrix bermanfaat untuk melihat **sebaran validitas percobaan**.

Tabel 10.1 : Dataset klasifikasi buah

Data aktual	Output model (prediksi)	Kesimpulan
Anggur	Anggur	valid
jeruk	Apel	invalid
apel	jeruk	invalid
Anggur	apel	invalid
jeruk	jeruk	valid

Tabel 10.1 merupakan contoh data yang akan dicoba untuk melihat bagaimana sebaran validitas dari percobaan yang dilakukan.

Tabel 10.2 : Kelas prediksi dan aktual

		Kelas Prediksi			
		Anggur	Apel	Jambu	Jeruk
Kelas Aktual	Anggur	20	2	1	2
	Apel	0	23	1	1
	Jambu	1	1	22	1
	Jeruk	1	0	0	24

Pada contoh tabel 10.2 diatas, menunjukkan pengujian terhadap citra buah anggur yang dilakukan sebanyak 25 kali pada dataset citra buah anggur, di mana pada tabel tersebut satu citra buah anggur tepat diuji satu kali. Bila diperhatikan pada area kotak berwarna hijau, pada area hijau menunjukan 25 citra buah anggur yang diujikan pada model algoritma klasifikasi mampu mengidentifikasi 20 citra anggur dengan benar, dengan detail dimana 20 citra buah anggur benar diidentifikasi sebagai citra anggur. Disisi lain terdapat kesalahan dalam melakukan klasifikasi dimana 2 citra anggur diidentifikasi sebagai citra buah apel, 1 citra anggur diidentifikasi sebagai citra jambu, dan 2 buah citra anggur diidentifikasi sebagai citra jeruk. Total percobaan yang dilakukan sebanyak 25 kali percobaan dengan 20 menghasilkan nilai yang benar sedangkan 5 percobaan menghasilkan nilai yang salah. Total uji pada kasus diatas berjumlah 100 kali pengujian, dengan proporsi pengujian pengujian citra masing-masing buah sebanyak 25 kali.

Sehingga hasilnya :

Nilai akurasi untuk pengujian citra anggur adalah :

$$20/25 \times 100\% = 80\%$$

Adapun akurasi total pengujian adalah :

$$(20+23+22+24)/(25+25+25+25) \times 100\% = 89\%$$

Penggunaan pengujian menggunakan confusion matrix mampu memberikan akurasi yang lebih baik, meskipun hasil akurasi akhir masih berdasarkan jumlah total uji coba yang dilakukan.

10.2 Permasalahan Klasifikasi Binary

Dalam permasalahan klasifikasi, tidak jarang kita dijumpai permasalahan yang hanya terdapat dua kelas. Hal ini disebut dengan permasalahan klasifikasi biner. Misalnya, klasifikasi gambar CT yang diambil dari paru-paru pasien saat mengidentifikasi penyakit Covid-19. Kasus ini memprediksi hanya dua kategori, yaitu pasien yang diprediksi mengidap Covid-19 (positif Covid-19) atau tidak terjangkit Covid-19 (negatif Covid-19). Klasifikasi semacam itu hanya mengklasifikasikan data sampel menjadi salah satu dari berikut ini: kasus positif atau negatif, atau bisa juga digunakan dalam bentuk klasifikasi seperti Baik/Tidak Baik, Baik/Buruk, Ya/Tidak, Sesuai/Tidak Sesuai.

Mengklasifikasikan gambar hewan sebagai termasuk gambar kucing atau bukan adalah contoh lain dari model klasifikasi biner. Dalam kasus terakhir, pemilihan model memaksa gambar diklasifikasikan dalam kelas kucing atau kelas bukan kucing. Oleh karena itu, dengan model seperti itu, harus memperhatikan asumsi awal, mis. gambar khusus hanya untuk anjing dan kucing, dengan tujuan utama klasifikasi untuk mengidentifikasi kucing. Saat memperhatikan kucing, prediksi positif berarti benar kucing, sedangkan prediksi negatif berarti bukan kucing.

Pada klasifikasi problem binary, ada empat parameter yang dapat digunakan untuk melakukan evaluasi dari hasil prediksi pada sebuah model yaitu:

- **Benar Positif(TP):** dimana nilai sebenarnya positif namun nilai prediksinya menunjukkan positif

- **Salah Positif (FP):** dimana nilai sebenarnya negatif namun nilai prediksinya menunjukkan positif
- **Benar Negatif (TN):** dimana nilai sebenarnya negatif namun nilai prediksinya menunjukkan negatif
- **Salah Negatif (FN):** dimana nilai sebenarnya positif namun nilai prediksinya menunjukkan negatif.

Mengenai klasifikasi gambar kucing dan bukan kucing (yang paling utama adalah benar kucing, di mana nilai positifnya adalah kucing dan nilai negatifnya adalah anjing):

True Positive (TP) adalah bahwa gambar yang diuji adalah gambar seekor kucing, maka hasil prediksi dari model tersebut juga seekor kucing. Hasil *false positive* (FP) terjadi ketika gambar uji adalah foto seekor anjing, tetapi hasil prediksi model menunjukkan kucing. *True Negative* (TN) adalah, jika gambar yang diuji adalah foto seekor anjing, maka hasil prediksi dari model tersebut juga menunjukkan seekor anjing. *False Negative* (FN) adalah, jika gambar yang diuji adalah foto seekor anjing, maka hasil prediksi dari model tersebut juga adalah seekor anjing.

10.2.1 Confusion matrix pada binary classification

mewakili prediksi data dari model yang telah dilatih dalam kondisi dunia nyata. Parameter yang digunakan pada Confusion matrix adalah TP, FP, TN dan FN yang secara sederhana disajikan dalam tabel 10.3 dibawah ini

Tabel 10.3 : Confusion matrix biner

		Nilai Prediksi	
		Positive	Negative
Nilai Aktual	Positive	True Positive (TN)	False Negatif (FN)
	Negative	False Positive (FP)	True Negatif (TN)

Pada kasus mendeteksi pasien positif Covid, dibuat detektor baru dengan nama N, diuji dengan menggunakan 100 sampel. Sampel diuji dengan alat yang hasil deteksinya dijadikan acuan validitas (ground truth), yaitu dengan menggunakan antigen. Dari pemeriksaan antigen sebelumnya diketahui 90 sampel bernilai negatif dan 10 sampel bernilai positif. Dengan menggunakan detektor N, 90 sampel negatif dapat dikenali sebagai negatif. Namun hasil yang diperoleh dari 10 sampel positif adalah 5 sampel dinyatakan positif dan 5 sampel sisanya dinyatakan negatif. Pertanyaannya berapakah tingkat akurasi detektor N?

$(90+5)/(90+10) \times 100\% = 95\%$!

Lalu dengan nilai 95% apakah merupakan hasil yang baik?

Kesalahan detektor X yang hanya pada 5% saja bisa berdampak besar. Apakah ada metrik lain yang menjelaskan kasus seperti ini?

Kasus memprediksi seseorang dengan Covid-19, kasus negatif palsu, adalah kasus yang paling tidak diharapkan. Hasil negatif palsu berarti seseorang benar-benar mengidap Covid-19 tetapi hasil tesnya negatif. Dalam memprediksi penyakit dan bencana pada umumnya, kejadian yang paling dihindari (sebagian besar ditekan seminimal mungkin) adalah kejadian negatif palsu, karena biasanya menimbulkan kerugian yang besar. Berdasarkan Confusion Matrix, kita dapat menghitung presisi, daya ingat, dan beberapa ukuran lainnya sebagai berikut.

Precision

Precision merupakan rasio dari data positif yang dapat diklasifikasikan dengan nilai benar terhadap jumlah total hasil prediksi bernilai positif. Dibawah ini merupakan rumus untuk menghitung nilai precision.

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

Recall

Penggunaan istilah Recall biasa juga dikenal dengan *Sensitivity* atau *True Positive Rate*. Selain itu juga didefinisikan sebagai rasio jumlah data positif (TP) yang diprediksi dengan benar terhadap jumlah data positif aktual. Apabila didapatkan nilai Recall tinggi maka menunjukkan bahwa kategori tersebut dapat dikenali dengan baik (FN rendah), dibawah ini menunjukkan rumus untuk mendapatkan nilai Recall

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

Nilai *Precision* dan *Recall* memiliki hubungan yang berimplikasi pada bagaimana prediktabilitas model yang dibuat, mengingat data latih, adalah data perkiraan awal yang digunakan untuk menguji suatu model agar kemampuannya dapat meningkat. Berikut adalah contoh istilah hubungannya:

A. Low Recall, Low Precision

Kondisi dengan nilai low recall dan low precision adalah model yang dianggap memiliki kinerja kurang baik. Dimana False Negative (FN) juga False Positive (FP), keduanya menunjukan prediksi dengan hasil yang salah, dan nilai yang besar. Model ini kita dapat digambarkan seperti pada gambar 10.1 dibawah ini.



Gambar 10.1 : Hubungan low recall, low precision

Pada contoh kasus klasifikasi kucing sebelumnya, model seperti ini menunjukan bahwa banyak citra yang

seharusnya diidentifikasi kucing sebaliknya diidentifikasi sebagai anjing juga sebaliknya.

B. *High Recall, Low Precision*

Kondisi ini bermakna dimana sebagian besar data positif dapat dikenali dengan baik sebagai (FN Rendah), namun terdapat banyak data yang negatif yang dikenali sebagai positive (FP) dan model ini kita dapat digambarkan seperti pada gambar 10.2 dibawah ini.



Gambar 10.2: Hubungan high recall, low precision

Contohnya Model untuk klasifikasi citra kucing dan bukan kucing. Model mampu mengklasifikasikan banyak gambar kucing dengan benar sebagai kucing namun tidak sedikit juga yang mengklasifikasikan gambar anjing dianggap sebagai kucing.

C. *Low Recall, High Precision*

Kondisi ini dapat mengklasifikasi lebih sedikit data yang bernilai positif (FN tinggi), namun model mengklasifikasikan data positif yang bernilai positif benar-benar bernilai positif (FP rendah).

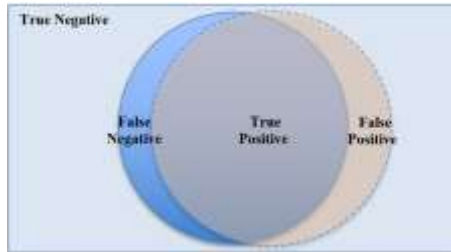


Gambar 10.3 : Hubungan low recall, high precision

Misalnya, model yang digunakan untuk mengklasifikasikan gambar bukan kucing dan kucing. Hasil prediksi model berupa citra kucing yang sebagian besar benar sebagai citra kucing. Sayangnya, banyak juga gambar yang sebenarnya gambar kucing, tapi seharusnya gambar anjing.

D. *High Recall, High Precision*

Kondisi yang diharapkan dimana model memiliki nilai Recall serta nilai Precision yang tinggi. Prediksi yang salah, baik kondisi *False Positive* juga *False Negative* harus dapat diminimalkan.



Gambar 10.4 : Hubungan high recall, high precision

F1-Score

Biasanya F1-Score digunakan untuk menghitung hubungan antara nilai Precision dan nilai Recall yang dinyatakan pada persamaan dibawah ini.

$$F1 - Score = \frac{Precision * Recall}{Precision + Recall}$$

Specificity

Specificity atau *True Negative Rate*. *Specificity* adalah rasio data yang diprediksi hasilnya benar negatif dibandingkan dengan data sebenarnya yang bernilai negatif dapat dilihat pada rumus dibawah ini.

$$Specificity = \frac{TN}{TN + FP}$$

Negative Predictive Value(NPV)

Nilai dari NPV adalah rasio data yang dengan benar diprediksi bernilai negatif dibandingkan dengan data sebenarnya yang diprediksi bernilai negatif.

$$Negative Predictive Value = \frac{TN}{TN + FN}$$

Accuracy

Akurasi merupakan model pengukuran paling mendasar dimana untuk kasus klasifikasi biner dapat dirumuskan dibawah ini:

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

Tabel 10.4 : Peta evaluasi model untuk klasifikasi biner

		Nilai Prediksi		
		Positif	Negatif	
Nilai Aktual	Positif	True Positive (TP)	False Negatif (FN)	Sensitivity, Recall, TP Rate $\frac{TP}{TP + FN}$
	Negatif	False Positive (FP)	True Negatif (TN)	Specificity, TN Rate $\frac{TN}{FP + TN}$ FP Rate

		Nilai Prediksi		
		Positif	Negatif	
				$\frac{FP}{FP + TN}$
		Precision $\frac{TP}{TP + FP}$	NPV $\frac{TN}{TN + FN}$	Accuracy $\frac{TP + TN}{TN + TN + FP + FN}$

Untuk menghitung confusion matrix, Anda harus terlebih dahulu memiliki serangkaian prediksi, sehingga dapat dibandingkan dengan target yang sebenarnya. Kita dapat membuat prediksi pada set pengujian, yang perlu diingat adalah kita menggunakan set pengujian hanya di akhir proyek setelah kita memiliki pengklasifikasi yang siap. kita dapat menggunakan fungsi `cross_val_predict()`:

```
from sklearn.model_selection import cross_val_predict

y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3)
```

Sama seperti fungsi `cross_val_score()`, `cross_val_predict()` melakukan *K-fold cross-validation*, tetapi alih-alih mengembalikan skor evaluasi, fungsi ini mengembalikan prediksi yang dibuat pada setiap lipatan tes. Ini berarti Anda mendapatkan prediksi bersih untuk setiap instans dalam set pelatihan.

Pada *confusion matrix* terdapat 2 parameter pertama adalah kelas target (`y_true`) dan kelas yang diprediksi (`y_pred`) seperti pada contoh dibawah ini:

```
From sklearn.metrics import confusion_matrix
y_true = [2, 0, 2, 2, 0, 1]
y_pred = [0, 0, 2, 2, 0, 2]
confusion_matrix(y_true, y_pred)
array([[2, 0, 0],
```

```
[0, 0, 1],  
[1, 0, 2]]])
```

10.3 Contoh Penggunaan Confusion Matrix

Jumlah titik dimana label diprediksi sama dengan label sebenarnya diwakili oleh elemen diagonal, adapun elemen yang salah label oleh pengklasifikasi disebut elemen off-diagonal. Confusion matrix semakin baik apabila nilai diagonal semakin tinggi, hal ini menunjukkan semakin banyak prediksi yang benar.

pada source dibawah ini menunjukkan confusion matrix dengan dan tanpa normalisasi berdasarkan ukuran dukungan kelas (jumlah elemen di setiap kelas). Normalisasi semacam ini bisa menarik jika terjadi ketidakseimbangan kelas untuk memiliki interpretasi yang lebih visual tentang kelas mana yang salah diklasifikasikan.

Pada contoh hasilnya terlalu baik ini dapat diakibatkan karena pilihan parameter regularisasi C bukan yang terbaik. Dalam aplikasi kehidupan nyata parameter ini biasanya dipilih menggunakan `grid_search`.

```
import matplotlib.pyplot as plt  
import numpy as np  
  
from sklearn.metrics import ConfusionMatrixDisplay  
from sklearn.model_selection import train_test_split  
from sklearn import svm, datasets  
  
# mengimport dataset iris  
iris = datasets.load_iris()  
A = iris.data  
b = iris.target  
class_names = iris.target_names
```

```
# Pisahkan data menjadi set pelatihan dan set pengujian
X_train, X_test, y_train, y_test = train_test_split(A, b,
random_state=0)
```

```
# Jalankan pengklasifikasi, menggunakan model too
(C too low) to see
```

```
# Dampaknya pada hasil
```

```
classifier = svm.SVC(kernel="linear", C=0.01).fit(X_train, y_train)
```

```
np.set_printoptions(precision=2)
```

```
# Plot confusion matrix yang tidak dinormalisasi
```

```
titles_options = [
```

```
    ("Confusion matrix, tanpa normalisasi", None),
```

```
    ("Confusion matrix dinormalisasi ", "true"),]
```

```
for title, normalize in titles_options:
```

```
    disp = ConfusionMatrixDisplay.from_estimator(
        classifier,
```

```
        X_test,
```

```
        y_test,
```

```
        display_labels=class_names,
```

```
        cmap=plt.cm.Blues,
```

```
        normalize=normalize, )
```

```
    disp.ax_.set_title(title)
```

```
    print(title)
```

```
    print(disp.confusion_matrix)
```

```
plt.show()
```

```
(Learn, 2022)
```

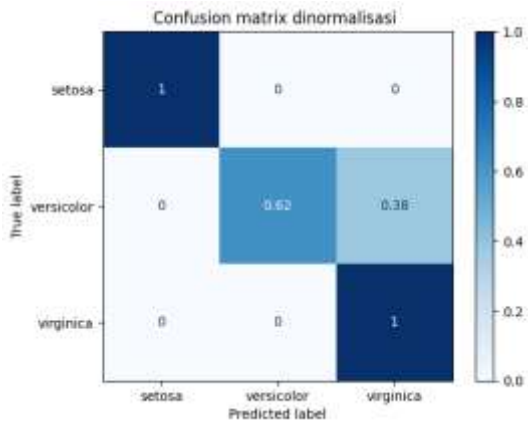
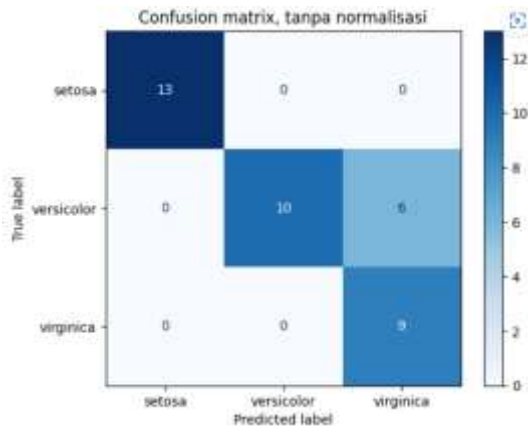
Hasilnya akan terlihat seperti dibawah ini

Confusion matrix, tanpa normalisasi

```
[[13  0  0]
 [ 0 10  6]
 [ 0  0  9]]
```

Confusion matrix dinormalisasi

```
[[1.  0.  0. ]
 [0.  0.62 0.38]
 [0.  0.  1. ]]
```



DAFTAR PUSTAKA

- Géron, A. 2017. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. 1st edn. Edited by N. Tache. United States of America: O'Reilly Media.
- Géron, A. 2019. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 2nd edn. United States of America: O'Reilly Media.
- Learn, S. 2022. *Metrics and scoring: quantifying the quality of predictions*, *scikit-learn.org*. Available at: https://scikit-learn.org/stable/modules/model_evaluation.html#confusion-matrix (Accessed: 12 December 2022).
- Swamynathan, M. 2017. *Mastering Machine Learning with Python in Six Steps*. New York: Apress. doi: 10.1007/978-1-4842-2866-1.

BAB 11

K-FOLD CROSS VALIDATION

Oleh Abdurrasyid

11.1 Pendahuluan

Mempelajari parameter fungsi prediksi dan mengujinya pada data yang sama adalah kesalahan metodologis: model yang hanya akan mengulangi label sampel yang sudah ada akan memiliki skor sempurna tetapi akan gagal memprediksi apa pun yang berguna pada data yang belum pernah diuji. Situasi ini disebut **overfitting**. Untuk menghindarinya, hal yang biasa dilakukan saat melakukan eksperimen pembelajaran mesin (*supervised*) dengan menahan sebagian dari data yang tersedia sebagai **set pengujian** **X_{test}**, **y_{test}**. Perhatikan bahwa kata "eksperimen" tidak dimaksudkan untuk menunjukkan penggunaan akademis saja, karena bahkan dalam pengaturan komersial pembelajaran mesin biasanya dimulai secara eksperimental.

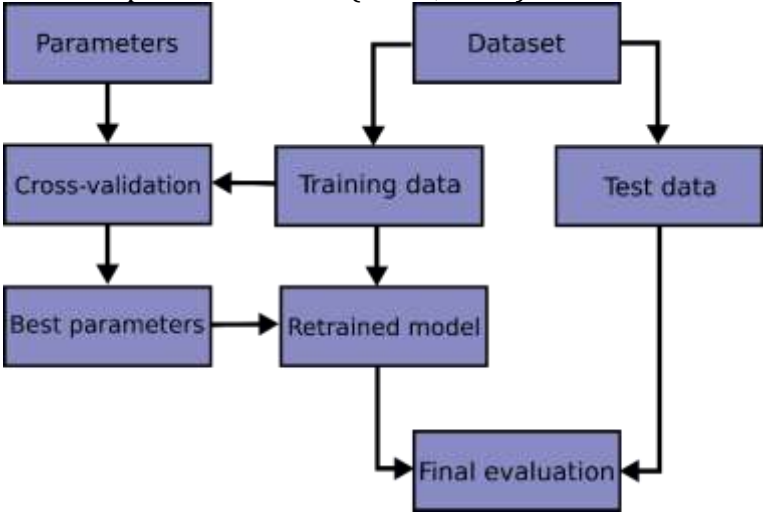
Cross-validation, kadang-kadang disebut estimasi rotasi atau pengujian di luar sampel, merupakan salah satu dari berbagai teknik validasi model serupa untuk menilai bagaimana hasil analisis statistik akan menggeneralisasi dataset yang independen. Cross-validation adalah metode resampling yang menggunakan bagian data yang berbeda untuk menguji dan melatih model pada iterasi yang berbeda. Terutama digunakan dalam pengaturan tujuannya adalah prediksi seberapa akurat kinerja model prediktif dalam praktiknya.

Dalam masalah prediksi, model biasanya diberikan himpunan data dari data yang diketahui di mana pelatihan dijalankan (training dataset), dan himpunan data dari data yang

tidak diketahui (atau data yang pertama kali dilihat) di mana model diuji (disebut dataset validasi atau dataset pengujian).

Tujuan cross-validation adalah untuk menguji kemampuan model untuk memprediksi data baru yang tidak digunakan dalam estimasinya, menandai masalah seperti overfitting atau bias pemilihan, dan memberikan insight tentang generalisasi model pada dataset independen.

Berikut adalah diagram alur alur kerja *cross validation* umum dalam pelatihan model. (Learn, 2022).



Gambar 11.1 : Diagram alur alur kerja *cross validation*

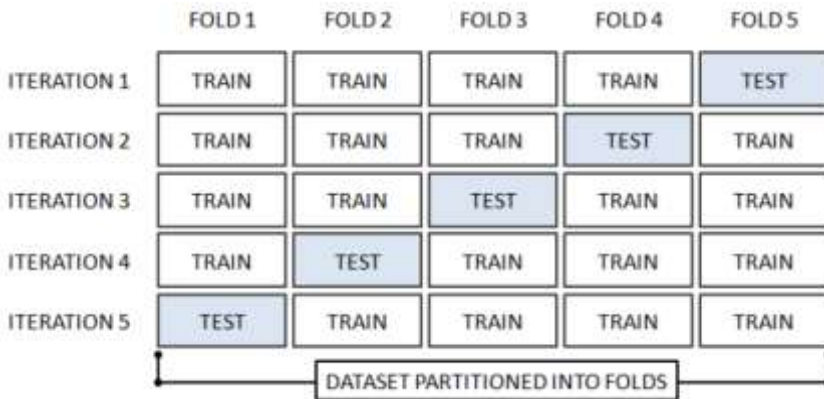
Masalah yang terlihat dengan pemisahan set pelatihan/pengujian adalah bahwa terdapat bias dalam pengujian karena pemisahan tentu akan mengurangi ukuran data pelatihan dalam sampel. Ketika membagi data, mungkin benar-benar menyimpan beberapa contoh yang berguna dari pelatihan. Selain itu, terkadang data yang dimiliki sangat kompleks sehingga set pengujian, meskipun tampaknya mirip dengan set pelatihan, tidak benar-benar mirip karena kombinasi nilai berbeda (yang merupakan tipikal dari kumpulan data yang sangat dimensi).

Masalah ini menambah ketidakstabilan hasil pengambilan sampel ketika Anda tidak memiliki banyak contoh. Risiko membagi data Anda dengan cara yang tidak menguntungkan juga menjelaskan mengapa pemisahan pelatihan/uji bukanlah solusi yang disukai oleh praktisi pembelajaran mesin ketika Anda harus mengevaluasi dan menyetel solusi pembelajaran mesin (Mueller and Massaron, 2016) jawaban dari masalah ini adalah dengan menggunakan **k-folds cross validation**.

11.2 K-Fold Cross Validation

Biasanya *k-fold cross-validation* digunakan ketika ukuran dataset tidak terlalu besar dimana cara kerja dari K-folds cross-validation adalah dengan membagi dataset Anda menjadi sejumlah k lipatan (bagian dari data Anda) dengan ukuran yang sama. Kemudian, setiap lipatan diadakan secara bergantian sebagai training data dan yang lainnya digunakan untuk testing data. Setiap iterasi menggunakan lipatan yang berbeda sebagai tes, yang akan menghasilkan perkiraan kesalahan.

Bahkan, setelah menyelesaikan tes pada satu lipatan terhadap yang lain yang digunakan sebagai pelatihan, lipatan berturut-turut, berbeda dari sebelumnya, diadakan dan prosedur diulangi untuk menghasilkan perkiraan kesalahan lain. Proses berlanjut sampai semua k-fold digunakan satu kali sebagai set pengujian dan Anda memiliki sejumlah perkiraan kesalahan yang dapat Anda hitung menjadi perkiraan kesalahan rata-rata (skor CV) dan standar kesalahan perkiraan (Mueller and Massaron, 2016; Swamynathan, 2017; Géron, 2019).



Gambar 11.2 : Cara kerja k-folds *cross-validation*

Keuntungan k-folds *cross-validation*

- Dapat bekerja dengan baik tidak tergantung dari jumlah contoh, karena dengan meningkatkan jumlah lipatan yang digunakan, Anda sebenarnya meningkatkan ukuran data training Anda (k lebih besar, data pelatihan lebih besar, bias berkurang) dan mengurangi ukuran testing data.
- Perbedaan distribusi untuk lipatan individu tidak terlalu penting. Ketika lipatan memiliki distribusi yang berbeda dibandingkan dengan yang lain, itu digunakan hanya sekali sebagai set tes dan dicampur dengan yang lain sebagai bagian dari set pelatihan.
- Menguji semua pengamatan, jadi Anda sepenuhnya menguji hipotesis pembelajaran mesin Anda menggunakan semua data yang ada.

Menggunakan *k-fold cross-validation* merupakan pilihan optimal kecuali data yang digunakan memiliki semacam urutan yang penting. Misalnya, jika data melibatkan deret waktu, seperti penjualan. Dalam hal ini, Anda tidak boleh menggunakan metode

pengambilan sampel acak tetapi mengandalkan pemisahan data latih/uji berdasarkan urutan asli sehingga urutannya dipertahankan.

Sebagai contoh : terdapat sebanyak 100.000 data dan ditentukan nilai **K** nya adalah 5 ($K=5$), maka index pembagian data untuk setiap iterasi adalah sebagai berikut :

1. Iterasi 1:
 - a. Index training data = 20.001 hingga 100.000
 - b. Index testing data = 1 hingga 20.000
2. Iterasi 2:
 - a. Index training data = 1 hingga 20.001 + 40.001 hingga 100.000
 - b. Index testing data = 20.001 hingga 40.000
3. Iterasi 3:
 - a. Index training data = 1 hingga 40.000 + 60.001 hingga 100.000
 - b. Index testing data = 40.001 hingga 60.000
4. Iterasi 4:
 - a. Index training data = 1 hingga 60.000 + 80.001 hingga 100.000
 - b. Index testing data = 60.001 hingga 80.000
5. Iterasi 5:
Index training data = 1 hingga 80.000
Index testing data = 80.001 hingga 100.000
Index akan berubah disesuaikan dengan nilai **K** yang digunakan.

11.3 Contoh penggunaan *k-fold cross-validation*

```
from sklearn.cross_validation import cross_val_score
# membaca dataset diabetes
df = pd.read_csv("Data/Diabetes.csv")
X = df.ix[:,8].values # variable independent
y = df['class'].values # variable dependent
```

```

# Normalisasi data
Sc=StandardScaler()
sc.fit(X)
X = sc.transform(X)
# Melakukan evaluasi model dengan memecah dataset menjadi
data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=2017)
# membuat decision tree classifier
clf = tree.DecisionTreeClassifier(random_state=2017)
# melakukan evaluasi model dengan menggunakan 10-fold cross
validation
train_scores=cross_val_score(clf, X_train, y_train,
scoring='accuracy', cv=5)
test_scores=cross_val_score(clf, X_test, y_test, scoring='accuracy',
cv=5)
print "Train Fold AUC Scores: ", train_scores
print "Train CV AUC Score: ", train_scores.mean()
print "\nTest Fold AUC Scores: ", test_scores
print "Test CV AUC Score: ", test_scores.mean()
(Swamynathan, 2017)

```

Hasilnya akan terlihat seperti dibawah ini

```

Train Fold AUC Scores: [0.80 0.73 0.81 0.76 0.71]
Train CV AUC Score: 0.76

Test Fold AUC Scores: [0.80 0.78 0.78 0.77 0.8]
Test CV AUC Score: 0.79

```

Stratified K-fold cross-validation

Sebelum mempelajari stratified cross-validation, kita perlu mengetahui mengenai pengambilan stratified sampling. Stratified sampling adalah teknik pengambilan sampel di mana sampel

dipilih dalam proporsi yang sama (dengan membagi populasi menjadi kelompok-kelompok yang disebut "strata" berdasarkan karakteristik) seperti yang terjadi dalam populasi. Misalnya, jika populasi yang diinginkan adalah 30% laki-laki dan 70% perempuan, kami membagi populasi menjadi dua kelompok (laki-laki dan perempuan) dan memilih 30% sampel sebagai laki-laki serta 70% sampel acak dari kelompok perempuan.

Menerapkan konsep pengambilan stratified sampling dalam *cross-validation* memastikan training *dataset* dan *testing dataset* memiliki proporsi fitur yang sama seperti pada dataset asli. Melakukan ini dengan variabel target memastikan bahwa hasil *cross-validation* dapat meminimalisir perkiraan kesalahan generalisasi.

Stratified K-fold cross-validation adalah perluasan dari K-fold cross-validation biasa tetapi khusus untuk masalah klasifikasi di mana pada kondisi pemisahan benar-benar acak, rasio atau proporsi antara kelas target akan tetap sama di setiap lipatan seperti dalam himpunan data lengkap, yang mengarah ke perkiraan bias dan varians yang lebih baik.

Contoh penggunaan stratified k-fold cross-validation

```
# contoh stratified kfold cross-validation
```

```
#nilai random_state dapat diubah
```

```
k_fold=cross_validation.StratifiedKFold(y=y_train,          n_folds=5,  
random_state=2017)
```

```
train_scores=[]
```

```
test_scores=[]
```

```
for k, (train, test) in enumerate(k_fold):
```

```
clf.fit(X_train[train], y_train[train])
```

```
train_score=clf.score(X_train[train],y_train[train])
```

```
train_scores.append(train_score)
```

```
# skor untuk test set
```

```
test_score=clf.score(X_train[test], y_train[test])
```

```
test_scores.append(test_score)
```

```
print('Fold: %s, Class dist.: %s, Train Acc: %.3f, Test Acc: %.3f %'  
(k+1, np.bincount(y_train[train]), train_score, test_score))  
print('\nTrain CV accuracy: %.3f % (np.mean(train_scores)))  
print("Test CV accuracy: %.3f % (np.mean(test_scores)))
```

Hasilnya akan terlihat seperti dibawah ini :

```
Fold: 1, Class dist.: [277 152], Train Acc: 0.758, Test Acc: 0.806  
Fold: 2, Class dist.: [277 152], Train Acc: 0.779, Test Acc: 0.731  
Fold: 3, Class dist.: [278 152], Train Acc: 0.767, Test Acc: 0.813  
Fold: 4, Class dist.: [278 152], Train Acc: 0.781, Test Acc: 0.766  
Fold: 5, Class dist.: [278 152], Train Acc: 0.781, Test Acc: 0.710
```

```
Train CV accuracy: 0.773
```

```
Test CV accuracv: 0.765
```

DAFTAR PUSTAKA

- Géron, A. 2019. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 2nd edn. United States of America: O'Reilly Media.
- Learn, S. 2022. *Cross-validation: evaluating estimator performance*, *scikit-learn.org*. Available at: https://scikit-learn.org/stable/modules/cross_validation.html#k-fold (Accessed: 25 December 2022).
- Mueller, J. paul and Massaron, L. 2016. *Machine Learning for dummies*. Canada: John Wiley & Sons, Inc.
- Swamynathan, M. 2017. *Mastering Machine Learning with Python in Six Steps*. New York: Apress. doi: 10.1007/978-1-4842-2866-1.

BIODATA PENULIS



Angga Aditya Permana
Newcomer Writer

Angga Aditya Permana (lahir pada Desember 1989 di Jakarta) seorang dosen full time di bidang Computer Science, fokus penelitian pada kriptografi, steganografi serta keamanan computer, namun pada tahun 2021 sedang mendalami topik bioinformatika dan network science. Angga juga memiliki hobi yaitu touring menggunakan motor dan juga bermain bulutangkis, saat ini sedang menjadi mahasiswa program doctoral pada IPB University. Memulai karir pertama kali sebagai Network Enginner tahun 2011 dan memulai profesinya sebagai dosen pada 2013 di kampus swasta yang ada di Jakarta, pernah juga mengajar di kampus negeri yang berada di Jakarta dan Tangerang, juga pernah di undang menjadi dosen tamu untuk mengenalkan bioinformatika di kampus negeri di Jakarta. Terimakasih..

BIODATA PENULIS



Wahyuddin S, S.Kom., M.Kom
Dosen STMIK Amika Soppeng

Wahyuddin S. was born at Malaka-Bone-Sulawesi Selatan in 1992. In 2011 he attended STMIK Dipanegara Makassar and was completed in 2015. He was completed after attending 7 semesters and active on an XPcom (Extreme Programmer Computer) campus organization. He was also active as a lecturer assistant for three semesters and taught several courses on programming. He continued his Master of Information systems at UNIKOM Bandung in 2016 and was completed in April 2019. He worked as a lecturer at a campus (STMIK AMIKA Soppeng) 2019 to present and also a Freelance Web Programmer. Has competence in the field of software engineer, application developer, multimedia, web developer, network security, and data analyst.

BIODATA PENULIS



Leo Willyanto Santoso

Dosen Program Studi Informatika
Fakultas Teknologi Industri Universitas Kristen Petra

Penulis merupakan dosen tetap di Program Studi Informatika Universitas Kristen Petra Surabaya. Penulis telah menyelesaikan pendidikan S2 di University of Melbourne, Australia

BIODATA PENULIS



Gentur Wahyu Nyipto Wibowo, S.Kom., M.Kom.

Dosen Program Studi Teknik Informatika
Fakultas Sains dan Teknologi Universitas Islam Nahdlatul Ulama
Jepara

Penulis lahir di Blora tanggal 23 Agustus 1979. Penulis adalah dosen tetap pada Program Studi Teknik Informatika Fakultas Sains dan Teknologi, Universitas Islam Nahdlatul Ulama Jepara. Menyelesaikan pendidikan S1 pada Jurusan Teknik Informatika dan melanjutkan S2 pada Jurusan Teknik Informatika. Penulis mulai belajar menekuni bidang Data Mining, Interaksi Manusia dan Komputer, serta Mikrokontroler.

BIODATA PENULIS



Anindya Khrisna Wardhani, S.Kom., M.Kom.,
Dosen Program Studi Rekam Medis dan Informasi Kesehatan
Politeknik Rukun Abdi Luhur, Kudus, Jawa Tengah

Penulis lahir di Kudus, Jawa Tengah pada tanggal 15 Agustus 1994. Saat ini penulis merupakan pengajar dan peneliti pada Program Studi Rekam Medis dan Informasi Kesehatan pada Politeknik Rukun Abdi Luhur, Kudus, Jawa Tengah. Penulis menyelesaikan Program Sarjana Teknik Informatika di Fakultas Ilmu Komputer Universitas Dian Nuswantoro (Udinus) Semarang pada tahun 2016. Semangat untuk terus belajar membuat penulis terpilih sebagai penerima Beasiswa Unggulan Masyarakat Berprestasi oleh Kementerian Pendidikan dan kebudayaan RI pada Program Magister Sistem Informasi di Universitas Diponegoro (Undip) Semarang dan selesai pada tahun 2018.

BIODATA PENULIS



Rahmaddeni, S.Kom., M.Kom.

Dosen Program Studi Teknik Informatika
STMIK Amik Riau

Penulis lahir di Padang tanggal 7 Desember 1983. Penulis adalah dosen tetap pada Program Studi Teknik Informatika, STMIK Amik Riau. Menyelesaikan pendidikan S1 di Jurusan Teknik Informatika STMIK Amik Riau pada tahun 2008 dan melanjutkan S2 di Jurusan Teknologi Informasi Universitas Putra Indonesia “YPTK” Padang pada tahun 2011. Bidang ilmu yang ditekuni penulis adalah Data Science, Machine Learning dan Data Mining. Penelitian yang dilakukan sesuai bidang ilmu telah dipublikasikan dalam berbagai Jurnal penelitian bidang informatika baik yang dikerjakan mandiri maupun yang berkolaborasi bersama dosen lain dan mahasiswa. Penulis juga merupakan reviewer jurnal bidang informatika.

BIODATA PENULIS



Ahmad Jurnaid Wahidin, S.Kom., M.Kom.

Dosen Program Studi Teknologi Informasi
Fakultas Teknik dan Informatika

Penulis bertempat tinggal di Tangerang dan menjadi dosen di Universitas Bina Sarana Informatika. Penulis lahir di Kalibata pada 15 Mei 1993. Tahun 2016 menyelesaikan pendidikan S1 pada Jurusan Teknik Informatika dan melanjutkan S2 pada Jurusan Ilmu Komputer yang selesai pada tahun 2018. Ikut tergabung dalam penulisan buku dengan judul Teknologi Jaringan Nirkabel, Aplikasi Pembelajaran Digital, Manajemen Proyek TI, Data Mining dan Inovasi Platform Digital Menuju Indonesia Emas.

BIODATA PENULIS



Gusti Eka Yulastuti, S.Kom, M.Kom

Dosen Program Studi Teknik Informatika
Fakultas Teknik Elektro dan Teknologi Informasi
Institut Teknologi Adhi Tama Surabaya (ITATS)

Penulis lahir di Surabaya pada tahun 1994. Penulis memperoleh gelar Sarjana Program Studi Informatika pada tahun 2016 dan gelar Magister Program Studi Ilmu Komputer pada tahun 2018 dari Fakultas Ilmu Komputer Universitas Brawijaya Malang.

Penulis merupakan dosen tetap Jurusan Teknik Informatika Institut Teknologi Adhi Tama Surabaya (ITATS) sejak 2019. Selain aktif dalam bidang pendidikan, penulis juga aktif dalam bidang penelitian dan pengabdian. Beberapa hasil penelitian penulis dimuat pada jurnal nasional terakreditasi dan jurnal internasional bereputasi. Bidang penelitian penulis yakni pada Algoritma Evolusi.

BIODATA PENULIS



Elisawati, S.Kom., M.Kom.

Dosen Program Studi Sistem Informasi
STMIK Dumai

Penulis lulusan STIKI Malang, Program Studi Teknik Informatika pada tahun 2003, dan memperoleh gelar Master Ilmu Komputer di Universitas Putra Indonesia (UPI) Padang pada tahun 2010 dengan konsentrasi Teknologi Informasi. Saat ini penulis bekerja sebagai Dosen Tetap di STMIK Dumai dan aktif mengajar di beberapa perguruan Tinggi yang ada di Dumai.

BIODATA PENULIS



Rima Rizqi Wijayanti, S.ST., MMSI.

Dosen Program Studi Teknik Informatika
Fakultas Teknik, Universitas Muhammadiyah Tangerang

Penulis lahir di Jakar tanggal 22 Mei 1986. Saat ini penulis merupakan dosen tetap pada Program Studi Teknik Informatika Fakultas Teknik, Universitas Muhammadiyah Tangerang. Penulis menyelesaikan pendidikan S1 di STMI Jakarta dan melanjutkan S2 di Universitas Gunadarma pada Tahun 2017, serta memiliki ketertarikan pada bidang sistem informasi.

BIODATA PENULIS



Abdurrasyid, S.Kom., MMSI.

Dosen Program Studi Teknik Informatika
Fakultas Telematika Energi Institut Teknologi PLN

Penulis lahir di Jakarta tanggal 10 Juni 1987. Saat ini penulis menjadi dosen tetap pada Program Studi Teknik Informatika, Fakultas Telematika Energi, Institut Teknologi PLN Jakarta. Penulis aktif melakukan publikasi dalam berbagai jurnal ilmiah baik nasional maupun internasional, memiliki ketertarikan pada bidang software engineering dan machine learning.