Supervised Learning (Decision Tree dan KNN)

Deskripsi Modul

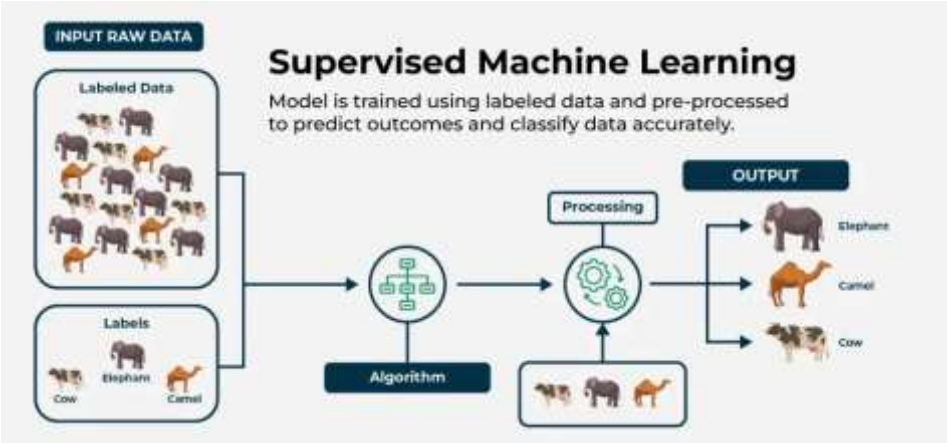
Mata Kuliah	**Machine Learning**
Kode Mata Kuliah / SKS	**RPL / 3**
Semester	**5**
Kelas	**TRPL**
Capaian Pembelajaran	
Deskripsi Singkat Mata Kuliah	
Bahan Kajian Modul	Pengantar Supervised Learning, Metode Decision Tree dan KNN
Bentuk dan Metode Pembelajaran	Praktikum
Waktu pembelajaran	120 Menit
Rekomendasi buku teks:	1. 2.
Petunjuk khusus	-

1. Materi Pembelajaran

1.1 Pengantar Supervised Learning

Pengertian Supervised Learning

Supervised Learning merupakan pendekatan mendasar untuk machine learning dan kecerdasan buatan. Pendekatan ini melibatkan pelatihan model menggunakan data berlabel, di mana setiap masukan (input) disertai keluaran (output) yang benar. Model belajar dengan membandingkan prediksinya dengan jawaban aktual yang diberikan dalam data pelatihan. Seiring berjalannya waktu, model menyesuaikan diri untuk meminimalkan kesalahan dan meningkatkan akurasi. Tujuan pembelajaran terbimbing adalah untuk membuat prediksi yang akurat saat diberikan data baru yang belum pernah dilihat. Misalnya, jika model dilatih untuk mengenali angka tulisan tangan, model akan menggunakan apa yang dipelajarinya untuk mengidentifikasi angka baru yang belum pernah dilihatnya dengan benar.

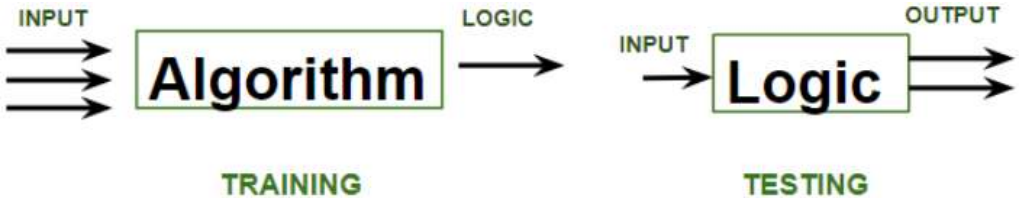


Cara Kerja Supervised Learning

Algoritma supervised learning terdiri dair fitur input dan label output. Proses kerjanya yaitu:

- Training Data: model dirancang dengan training dataset yang terdiri dari input data (fitur) dan data output (label atau variabel target).
- Proses pembelajaran: algoritma memproses training data, mempelajari relasi atau hubungan antara fitur input dan label output. Label output didapatkan dari penyesuaian parameter model untuk mengecilkan perbedaan antara prediksi dan label aktual.

Setelah training, model dievaluasi menggunakan testing dataset untuk menghitung akurasi dan kinerja model. Kemudian kinerja model dioptimasi dengan menyesuaikan parameter dan menggunakan teknik seperti cross validation untuk menyeimbangkan bias dan varians. Penyesuaian ini memastikan model menggeneralisasi secara baik untuk data baru.



- Fase training melibatkan proses algoritma data label, dimana setiap poin data dipasangkan dengan output yang benar. Algoritma belajar untuk mengidentifikasi pola dan relasi antara data input dan output.
- Fase testing melibatkan proses algoritma data baru dan mengevaluasi kemampuan data unutk memprediksi output yang benar berdasarkan pola yang dipelajari.

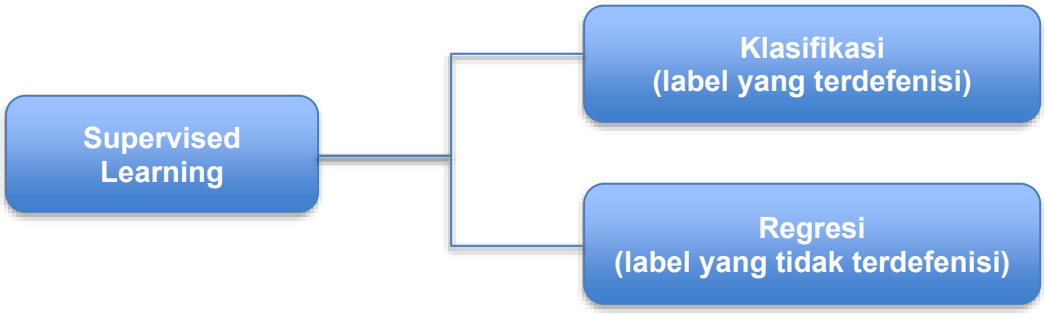
Jenis-Jenis Supervised Learning

1. Klasifikasi

   Dimana output adalah variabel kategorikal, contoh: spam dan non-spam, benar dan salah.

2. Regresi

   Dimana output adalah variabel kontinu, contohnya: prediksi harga rumah, harga saham.



3. Perbedaan klasifikasi dan regresi

   Perbedaan klasifikasi dan regresi dicontohkan oleh data berikut:

User ID	Gender	Age	Salary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	1
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	1
15728773	Male	27	58000	1
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	1
15727311	Female	35	65000	0
15570769	Female	26	80000	1
15606274	Female	26	52000	0
15746139	Male	20	86000	1
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1

Figure A: CLASSIFICATION

Temperature	Pressure	Relative Humidity	Wind Direction	Wind Speed
10.69261758	986.882019	54.19337313	195.7150879	3.278597116
13.59184184	987.8729248	48.0648859	189.2951202	2.909167767
17.70494885	988.1119385	39.11965597	192.9273834	2.973036289
20.95430404	987.8500366	30.66273218	202.0752869	2.965289593
22.9278274	987.2833862	26.06723423	210.6589203	2.798230886
24.04233986	986.2907104	23.46918024	221.1188507	2.627005816
24.41475295	985.2338867	22.25082295	233.7911987	2.448749781
23.93361956	984.8914795	22.35178837	244.3504333	2.454271793
22.68800023	984.8461304	23.7538641	253.0864716	2.418341875
20.56425726	984.8380737	27.07867944	264.5071106	2.318677425
17.76400389	985.4262085	33.54900114	280.7827454	2.343950987
11.25680746	988.9386597	53.74139903	68.15406036	1.650191426
14.37810685	989.6819458	40.70884681	72.62069702	1.553469896
18.45114201	990.2960205	30.85038484	71.70604706	1.005017161
22.54895853	989.9562988	22.81738811	44.66042709	0.264133632
24.23155922	988.796875	19.74790765	318.3214111	0.329656571

Figure B: REGRESSION

- Gambar A: dataset toko yang berguna untuk memprediksi apakah pelanggan akan membeli produk tertentu berdasarkan jenis kelamin, usia dan gaji.
  Input: jenis kelamin, usia, gaji.
  Output: pembelian diwakili oleh 0 dan 1. 1 berarti pelanggan akan membeli dan 0 berarti pelanggan tidak akan membeli.
- Gambar B: dataset meteorologi yang digunakan unutk memprediksi kecepatan angin berdasarkan berbagai parameter.
  Input: titik embun, suhu, tekanan, kelembapan relatif, arah angin
  Output: kecepatan angin

Contoh kasus Supervised Learning

1. Deteksi penipuan (Fraud Detection) dalam perbankan
   ➔ Memanfaatkan algoritma pembelajaran yang diawasi pada data transaksi historis, melatih model dengan dataset berlabel transaksi yang sah dan penipuan untuk memprediksi pola penipuan secara akurat.
2. Prediksi penyakit parkinson
   ➔ Penyakit parkinson adalah gangguan progresif yang mempengaruhi sistem saraf dan bagian tubuh yang dikendalikan oleh saraf. Label pada dataset adalah terdeteksi dan tidak terdeteksi.
3. Prediksi churn pelanggan
   ➔ Menggunakan teknik supervised learning untuk menganalisis data pelanggan historis, mengidentifikasi fitur yang terkait dengan tingkat churn untuk memprediksi retensi pelanggan secara efektif.
4. Klasifikasi sel kanker
   ➔ Menerapkan supervised learning untuk sel kanker berdasarkan fitur-fitur dan mengidentifikasi apakah 'ganas' atau 'jinak'.
5. Prediksi harga saham
   ➔ Menerapkan supervised learning untuk memprediksi sinyal yang menunjukkan apakah membeli saham tertentu akan membantu atau tidak.

Algoritma Populer Supervised Learning

1. Regresi Linier

Regresi linier adalah jenis algoritma supervised learning yang digunakan untuk memprediksi nilai keluaran kontinu. Regresi linier adalah salah satu algoritma paling sederhana dan paling banyak digunakan dalam supervised learning.

2. Regresi logistik

   Regresi logistik adalah jenis algoritma klasifikasi supervised learning yang digunakan untuk memprediksi variabel keluaran biner.

3. Desicion Tree

   Desicion tree adalah sebuah struktur mirip pohon yang digunakan untuk memodelkan keputusan dan kemungkinan konsekuensi. Setiap simpul internal dalam pohon mewakili sebuah keputusan, sementara setiap simpul daun mewakili kemunkinan hasil.

4. Random Forest

   Random forest juga terdiri dari beberapa pohon keputusan yang bekerja sama unutk membuat prediksi. Setiap pohon di hutan dilatih pada subset fitur dan data input yang berbeda. Prediksi akhir dibuat dengan menggabungkan prediksi semua pohon di hutan.

5. Support Vector Machine (SVM)

   Algoritma SVM menciptakan sebuah hiperbidang untuk memisahkan ruang n-dimensi ke dalam kelas-kelas dan mengidentifikasi kategori titik data baru yang tepat. Kasus-kasus ekstrim yang membantu menciptakan hiperbidang disebut vektor pendukung, oleh karena itu dinamakan Support Vector Machine.

6. K-Nearest Neighbors (KNN)

   KNN bekerja dengan menemukan k contoh pelatihan yang paling dekat dengan input yang diberikan, lalu memprediksi kelas atau nilai berdasarkan kelas mayoritas atau nilai rata-rata terdekat. Performa KNN dapat dipengaruhi oleh pilihan k dan metrik jarak yang digunakan untuk mengukur kedekatan.

7. Gradient Boosting

   Gradient boosting menggabungkan pembelajaran yang lemah, seperti desicion tree, untuk menciptakan model yang kuat. Proses ini secara berulang (iteratif) membangun model baru yang mengoreksi kesalahan yang dibuat oleh model sebelumnya.

8. Algoritma Naive-Bayes

   Algoritma Naive-Bayes adalah algoritma supervised learning yang didasarkan pada penerapan Teorema Bayes dengan asumsi "naif" bahwa fitur-fitur bersifat independen satu sama lain mengingat label data.

Algoritma	Regresi, Klasifikasi	Tujuan	Metode	Kasus Penggunaan

Regresi Linier	Regresi	Memprediksi nilai keluaran berkelanjutan	Persamaan linier meminimalkan jumlah kuadrat residual	Memprediksi nilai kontinu
Regresi Logistik	Klasifikasi	Memprediksi variabel keluaran biner	Fungsi logistik mengubah hubungan linier	Tugas klasifikasi biner
Desicion Tree	Keduanya	Keputusan dan hasil model	Struktur seperti pohon dengan keputusan dan hasil	Tugas klasifikasi dan regresi
Random Forest	Keduanya	Meningkatkan akurasi klasifikasi dan regresi	Menggabungkan beberapa pohon keputusan	Mengurangi overfitting, meningkatkan akurasi prediksi
SVM	Keduanya	Membuat hyperplane untuk klasifikasi atau prediksi nilai kontinu	Memaksimalkan margin antar kelas atau memprediksi nilai kontinu	Tugas klasifikasi dan regresi
KNN	Keduanya	Prediksi kelas atau nilai berdasarkan k terdekat	Menemukan k terdekat dan memprediksi berdasarkan mayoritas atau rata-rata	Tugas klasifikasi dan regresi, sensitif terhadap data yang berisik
Gradient Boosting	Keduanya	Gabungkan pelajar yang lemah untuk menciptakan model yang kuat	Memperbaiki kesalahan secara berulang dengan model baru	Tugas klasifikasi dan regresi untuk meningkatkan akurasi prediksi
Naive Bayes	Klasifikasi	Memprediksi kelas berdasarkan asumsi independen fitur	Teorema Bayes dengan asumsi independensi fitur	Klasifikasi teks, penyaringan spam, analisis sentimen, medis

Langkah-Langkah dalam Supervised Learning

1. Pengumpulan dan Pra-pemrosesan Data

   Mengumpulkan dataset berlabel yang terdiri dari fitur masukan dan label keluaran target. Membersihkan data, tangani nilai yang hilang dan skalakan fitur sesuai kebutuhan untuk memastikan kualitas tinggi untuk algoritma supervised learning.
2. Membagi Data

   Membagi data menjadi training set 80% dan testing set 20%.
3. Memilih Model

   Pilih algoritma yang tepat berdasarkan jenis masalah. Langkah ini krusial untuk supervised learning yang efektif dalam AI.
4. Melatih Model

   Memberikan data masukan dan label keluaran pada model, yang memungkinkan mempelajari pola dengan menyesuaikan parameter internal.
5. Evaluasi Model

Menguji model yang dilatih pada testing set yang belum terlihat dan nilai kinerjanya menggunakan berbagai metrik.

6. Penyesuaian Hiperparameter

   Menyesuaikan pengaturan yang mengendalikan proses pelatihan menggunakan teknik seperti pencarian kisi dan validasi silang.

7. Pemilihan dan Pengujian Model Akhir

   Melatih ulang model pada dataset lengkap menggunakan hiperparameter terbaik yang menguji kinerjanya pada testing dataset untuk memastikan kesiapan penerapan.

8. Penerapan Model

   Menerapkan model yang tervalidasi untuk membuat prediksi pada data baru yang belum terlihat.

1.2 Klasifikasi

Klasifikasi mengajarkan mesin untuk mengurutkan benda menjadi kategori. Klasifikasi belajar dengan melihat contoh dengan label (seperti email bertanda "spam" atau "tidak spam"). Setelah belajar, klasifikasi bisa menentukan kategori data baru, misalnya mengidentifikasi apakah email baru meupakan spam atau bukan.

1.3 Regresi

Regresi dalam supervised learning bertujuan untuk memprediksi nilai numerik kontinu berdasarkan satu atau lebih fitur independen. Regresi ini menemukan hubungan antar variabel sehingga prediksi dapat dibuat. Terdapat dua jenis variabel dalam regresi:

- Variabel Dependen (Target) : Variabel yang ingin kita prediksi, misalnya harga rumah.
- Variabel Independen (Fitur) : Variabel input yang memengaruhi prediksi misalnya lokalitas, jumlah kamar.

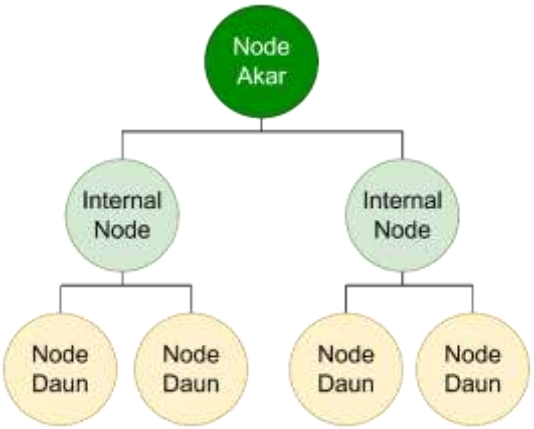
Masalah analisis regresi bekerja jika variabel keluaran berupa nilai riil atau kontinu seperti "gaji" atau "bobot". Banyak model regresi yang dapat digunakan, tetapi model yang paling sederhana adalah regresi linier.

1.4 Decision Tree

Decision tree membantu membuat keputusan dengan memetakan berbagai pilihan dan kemungkinan hasilnya. Metode ini digunakan untuk tugas-tugas seperti klasifikasi dan prediksi. Decision tree memiliki struktur seperti pohon yang dimulai dengan satu pertanyaan utama yang

disebut node akar yang mewakili keseluruhan dataset. Dari sana, pohon tersebut bercabang menjadi berbagai kemungkinan berdasarkan fitur dalam data. Struktur decision tree:

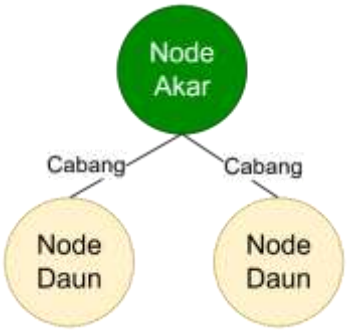
1. Node Akar → Titik awal yang mewakili keseluruhan kumpulan data.
2. Cabang → Garis yang menghubungkan simpul yang menunjukkan aliran dari satu keputusan ke keputusan lainnya.
3. Node Internal → Titik tempat keputusan dibuat berdasarkan fitur data.
4. Node Daun → Titik akhir pohon tempat keputusan atau prediksi akhir dibuat.



Langkah Kerja Decision Tree

1. **Mulai dengan Node Akar**

   ➔ Dimulai dengan pertanyaan utama pada node akar yang diturunkan dari fitur-fitur kumpulan data.



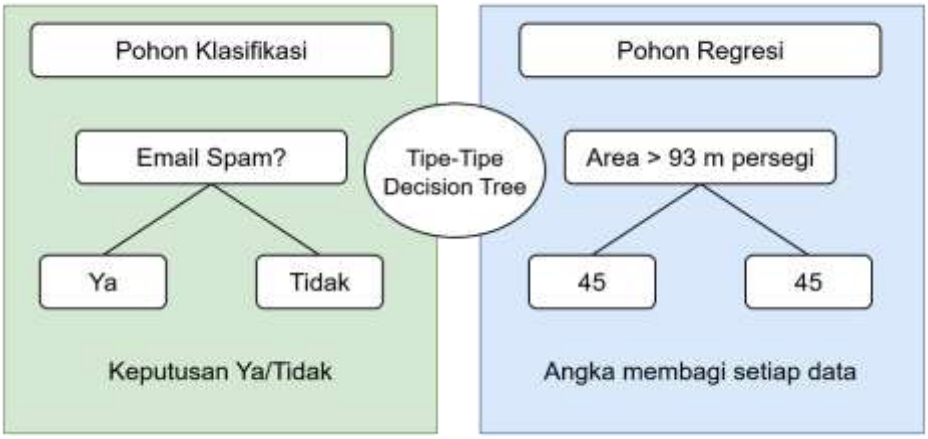
2. **Ajukan pertanyaan Ya/Tidak**

   ➔ Dari akar, pohon mengajukan serangkaian pertanyaan ya/tidak untuk membagi data menjadi subset berdasarkan atribut tersebut.

3. **Percabangan berdasarkan jawaban**

   Setiap pertanyaan mengarah ke cabang yang berbeda:

   - Jika jawabannya ya, pohon mengikutii satu jalur.
   - Jika jawabannya tidak, pohon mengikuti jalur lain.



4. **Lanjutkan pemisahan**

   ➔ Percabangan berlanjut melalui keputusan selanjutnya yang membantu dalam mengurangi data langkah demi langkah.

Kelebihan Decision Tree

1. **Mudah dipahami** → decision tree bersifat visual yang memudahkan unutk mengikuti proses pengambilan keputusan.
2. **Fleksibilitas** → dapat digunakan untuk masalah klasifikasi dan regresi.
3. **Tidak perlu penskalaan fitur** → metode ini tidak mengharuskan melakukan penskalaan atau menormalkan data.
4. **Menangani hubungan non-linier** → menangkap hubungan non-linier yang kompleks antara fitur dan hasil secara efektif.

5. **Kemampuan menafsirkan** → struktur pohon mudah ditafsirkan dan membantu pengguna memahami alasan dibalik setiap keputusan.
6. **Menangani missing value** → dapat menangani missing value dengan menggunakan strategi seperti menetapkan nilai yang paling umu atau mengabaikan data yang hilang selama pemisahan.

Kekurangan Decision Tree

1. **Overfitting** → metode ini rentan dengan overfitting pada data pelatihan jika percabangan pohon terlalu dalam. Artinya setiap cabang pohon hanya menghafal data tanpa mempelajari pola umum data, yang menyebabkan kinerja yang buruk pada data yang tidak terlihat atau data baru.
2. **Ketidakstabilan** → metode ini bida menjadi tidak stabil yang berarti perubahan kecil pada data dapat menyebabkan perbedaan signifikan pada struktur pohon dan prediksi.
3. **Bias terhadap fitur dengan banyak kategori** → dapat menjadi bias terhadap fitur dengan banyak nilai berbeda yang terlalu berfokus pada fitur tersebut dan berpotensi mengilangkan fitur penting lainnya yang dapat mengurangi akurasi prediksi.
4. Komputasi yang besar untuk data besar → merancang dan memangkas decision tree membutuhkan komputasi yang tinggi untuk data dengan jumlah banyak, terutama saat kedalaman pohon meningkat.
5. Tidak bisa diterapkan untuk fitur yang kompleks → decision tree akan kesulitan mengolah data dengan fitur yang kompleks, sehingga membuat metode ini tidak efektif untuk jenis data tertentu.

Contoh Kasus

1. Dataset: dummy dataset yang terdiri dari 8 data.
   Label: positif dan negatif.
   Fitur biner: apakah kalimat mengandung kata bagus, buruk, cepat, mahal.

No	Teks	bagus	buruk	cepat	mahal	Label
1	Produk bagus sekali	1	0	0	0	positif
2	Kualitas buruk	0	1	0	0	negatif
3	Pengiriman cepat dan bagus	1	0	1	0	positif
4	Sangat buruk dan lambat	0	1	0	0	negatif
5	Harga mahal dan kualitas buruk	0	1	0	1	negatif
6	Murah dan bagus	1	0	0	0	positif
7	Cepat tapi buruk	0	1	1	0	negatif
8	Bagus dan murah	1	0	0	0	positif

Ringkasan kelas: 4 positif, 4 negatif

2. Entropi awal $\left(H(S)\right)$

$$H(S) = -\sum_{i=1}^{n} p_i \,.\log_2 p_i$$

Keterangan:

$H(S)$ = entropi dari himpunan data S

n = jumlah kelas

p_i = proporsi elemen dalam kelas ke-i

Jika kelas yang digunakan adalah positif, maka:

$$H(S) = -\sum_{i=positif}^{positif,negatif} p_{positif} \,.\log_2 p_{positif} = -\frac{4}{8}.\log_2\left(\frac{4}{8}\right) = -0.5.\log_2 0.5$$

Karena jumlah kelas positif dan negatif sama-sama 4, maka untuk perhitungan entropi kelas negatif sama dengan perhitungan entropi kelas positif. Pencarian $\log_2$ atau log basis 2, yaitu:

$$\log_2 0.5 = \frac{\log 0.5}{\log 2} = \frac{-0.3010}{0.3010} = -1$$

Sehingga perhitungan nilai entropi awal menjadi:

$$H(S) = -0.5\log_2 0.5 - 0.5\log_2 0.5 = -0.5(-1) - 0.5(-1) = 1.0\ bit$$

3. Hitung Information Gain (untuk memilih akar pohon)

Untuk menentukan nilai positif dan negatif adalah dari frekuensi kemunculan fitur dalam tabel dataset.

Fitur bagus

- bagus=1: 4 positif, 0 negatif → jumlah data untuk bagus=1 adalah 4+0=4

  $p_{positif} = \frac{4}{4} = 1, p_{negatif} = 0$

  $H = -\left(1 \times \log_2 1\right) = -(1 \times 0) = 0$

- bagus=0: 0 positif, 4 negatif → jumlah data untuk bagus=0 adalah 0+4=4

  $p_{positif} = \frac{0}{4} = 0, p_{negatif} = 1$

  $H = -\left(1 \times \log_2 1\right) = -(1 \times 0) = 0$

$$entropi\ bersyarat = \left(\frac{4}{8}\right).0 + \left(\frac{4}{8}\right).0 = 0$$

$$IG(S, bagus) = entropi\ awal - entropi\ bersyarat = 1.0 - 0 = 1.0$$

Fitur buruk

- buruk=1: 4 positif, 0 negatif → H = 0
- buruk=0: 0 positif, 4 negatif → H = 0

$$entropi\ bersyarat = \left(\frac{4}{8}\right).0 + \left(\frac{4}{8}\right).0 = 0$$

$$IG(S, bagus) = 1.0 - 0 = 1.0$$

Fitur cepat

- cepat=1: 1 positif, 1 negatif →

  $$p_{positif} = \frac{1}{2} = 0.5, p_{negatif} = 0.5$$

  $$H = -\left(0.5 \times \log_2 0.5 + 0.5 \times \log_2 0.5\right) = -(-0.5 - 0.5) = 1$$

- cepat=0: 3 positif, 3 negatif → H = 1

  $$entropi\ bersyarat = \left(\frac{2}{8}\right) \times 1 + \left(\frac{6}{8}\right) \times 1 = 1.0$$

  $$IG(S, cepat) = 1.0 - 1.0 = 0$$

Fitur mahal

- mahal=1: 0 positif, 1 negatif → H = 0
- mahal=0: 4 positif, 3 negatif →

  $$p_{positif} = \frac{4}{7} = 0.57, p_{negatif} = \frac{3}{7} = 0.43$$

  $$H = -\left(0.57.\log_2 0.57 + 0.43 \log_2 0.43\right) = -\left(0.57(-0.81) + 0.43(-1.22)\right)$$
  $$= -(-0.4617 - 0.5246) = 0.9863$$

$$entropi\ bersyarat = \left(\frac{1}{8}\right) \times 0 + \left(\frac{7}{8}\right) \times 0.9863 = 0.863$$

$$IG(S, mahal) = 1.0 - 0.863 = 0.137$$

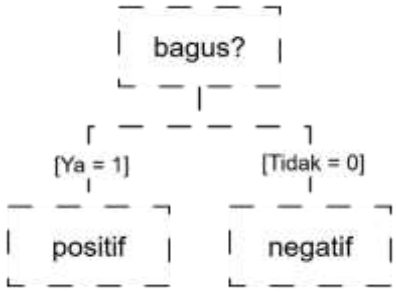
Keputusan akar: bagus atau buruk karena memiliki nilai $IG = 1.0$ (pemisahan sempurna).

Ambil yang intuitif: akar = bagus.

4. Membuat pohon keputusan

   Akara: bagus?

   - Ya (1) → semua conoth positif makan daun = positif
   - Tidak (0) → semua contoh negatif maka daun negatif



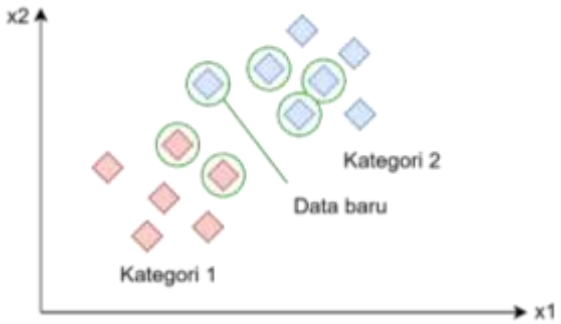
Aturan dari pohon:

- Jika kalimat mengandung kata "bagus" → Sentimen = Positif
- Jika tidak mengandung kata "bagus" → Sentimen = Negatif

Pohon Keputusan dengan data baru:
"Pengiriman lambat dan buruk" → tidak ada bagus → Negatif
"Produk ini bagus sekali" → mengandung bagus → Positif

1.5 K-Nearest Neighbor

K-Nearest Neighbor (KNN) adalah algoritma supervised learning yang umumnya digunakan untuk klasifikasi, tetapi juga dapat digunakan untuk tugas regresi. K-Nearest Neighbors juga disebut sebagai algoritma pembelajar malas (lazy learner) karena tidak langsung belajar dari set pelatihan, melainkan menyimpan seluruh set data dan melakukan perhitungan hanya pada saat klasifikasi.



Titik baru diklasifikasikan sebagai Kategori 2 karena sebagian besar tetangga terdekatnya berupa kotak biru. KNN menetapkan kategori berdasarkan mayoritas titik terdekatnya. Gambar di atas menunjukkan bagaimana KNN memprediksi kategori titik data baru berdasarkan tetangga terdekatnya.

- Berlian merah melambangkan Kategori 1 dan kotak biru melambangkan Kategori 2.

- Titik data baru memeriksa tetangga terdekatnya (titik yang dilingkari).
- Karena mayoritas tetangga terdekatnya adalah kotak biru (Kategori 2), KNN memperkirakan titik data baru tersebut termasuk dalam Kategori 2.

KNN bekerja dengan menggunakan kedekatan dan pemungutan suara mayoritas untuk membuat prediksi.

Nilai 'K' dalam K-Nearest Neighbor

Dalam algoritma k-Nearest Neighbours, k hanyalah angka yang memberi tahu algoritma berapa banyak titik atau tetangga terdekat yang harus dilihat saat membuat keputusan. Algoritma ini bekerja dengan mencari "k" titik data terdekat (tetangga) ke input yang diberikan dan membuat prediksi berdasarkan kelas mayoritas (untuk klasifikasi) atau nilai rata-rata (untuk regresi).

Contoh: Untuk menentukan buah berdasarkan bentuk dan ukurannya.

Jika k = 3, algoritma mencari 3 buah terdekat dengan buah baru.

Jika 2 dari 3 buah tersebut adalah apel dan 1 adalah pisang, algoritma mengatakan buah baru tersebut adalah apel karena sebagian besar buah tetangganya adalah apel.

Bagaimana cara memilih nilai k untuk Algoritma KNN

- Nilai k dalam KNN menentukan berapa banyak tetangga yang dilihat algoritma saat membuat prediksi.
- Memilih k yang tepat penting untuk hasil yang baik.
- Jika data memiliki banyak noise atau outlier, penggunaan k yang lebih besar dapat membuat prediksi lebih stabil.
- Tetapi jika k terlalu besar, modelnya mungkin menjadi terlalu sederhana dan kehilangan pola penting dan ini disebut underfitting.
- Jadi k harus dipilih secara hati-hati berdasarkan data.

Metode Statistik untuk Memilih k

1. **Validasi Silang** → Validasi silang adalah cara yang baik untuk menemukan nilai k terbaik dengan menggunakan validasi silang k-fold. Ini berarti membagi dataset menjadi k bagian. Model dilatih pada beberapa bagian ini dan diuji pada bagian yang tersisa. Proses ini diulang untuk setiap bagian. Nilai k yang memberikan akurasi rata-rata tertinggi selama pengujian ini biasanya merupakan nilai terbaik untuk digunakan.
2. **Metode Elbow** → Dalam Metode Elbow , kita menggambar grafik yang menunjukkan tingkat kesalahan atau akurasi untuk berbagai nilai k. Seiring bertambahnya k, kesalahan biasanya

menurun pada awalnya. Namun, setelah titik tertentu, kesalahan berhenti menurun dengan cepat. Titik di mana kurva berubah arah dan terlihat seperti "siku" biasanya merupakan pilihan terbaik untuk k.

3. **Nilai Ganjil untuk k** → Sebaiknya gunakan angka ganjil untuk k, terutama dalam masalah klasifikasi. Ini membantu menghindari seri saat menentukan kelas mana yang paling umum di antara tetangganya.

Metrik Jarak yang Digunakan dalam Algoritma KNN

KNN menggunakan metrik jarak untuk mengidentifikasi tetangga terdekat. Tetangga-tetangga ini digunakan untuk tugas klasifikasi dan regresi. Untuk mengidentifikasi tetangga terdekat, kami menggunakan metrik jarak berikut:

1. Jarak Euclidean

   Jarak Euklides didefinisikan sebagai jarak garis lurus antara dua titik pada suatu bidang atau ruang. Anda dapat menganggapnya sebagai jalur terpendek yang akan Anda tempuh jika Anda bergerak langsung dari satu titik ke titik lainnya.

$$jarak(x, X_i) = \sqrt{\sum_{j=1}^{D} \left(X_J - X_{iJ}\right)^2}$$

2. Jarak Manhattan

   Ini adalah total jarak yang akan Anda tempuh jika Anda hanya dapat bergerak di sepanjang garis horizontal dan vertikal seperti grid atau jalan-jalan kota. Disebut juga "jarak taksi" karena taksi hanya dapat melaju di sepanjang jalan-jalan kota yang berbentuk grid.

$$D(X, y) = \sum_{i=1}^{N} |x_i - y_i|$$

3. Jarak Minkowski

   Jarak Minkowski seperti keluarga jarak, yang mencakup jarak Euclidean dan Manhattan sebagai kasus khusus.

$$D(X, y) = \left(\sum_{i=1}^{N} (X_i - y_i)^P\right)^{\frac{1}{P}}$$

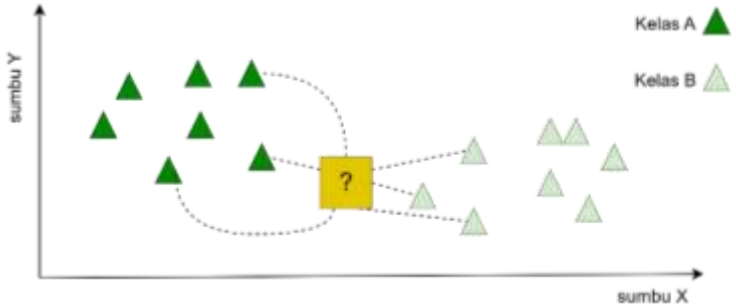
   Dari rumus di atas, ketika p=2, rumus tersebut menjadi sama dengan rumus jarak Euklides, dan ketika p=1, rumus tersebut berubah menjadi rumus jarak Manhattan. Jarak Minkowski pada

dasarnya adalah rumus fleksibel yang dapat merepresentasikan jarak Euklides atau Manhattan, tergantung pada nilai p.

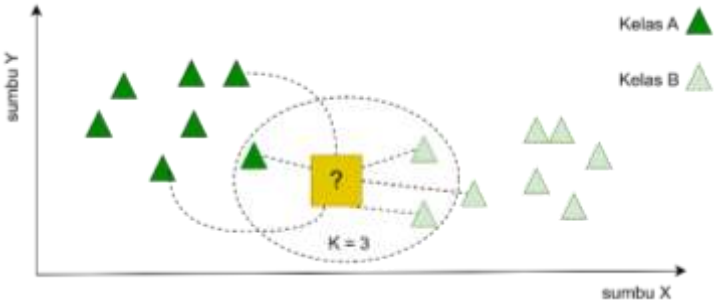
Cara Kerja KNN

1. **Memilih nilai K yang optimal** → K mewakili jumlah tetangga terdekat yang perlu dipertimbangkan saat membuat prediksi.

2. **Menghitung jarak** → Jarak Euclidean banyak digunakan untuk mengukur kesamaan antara titik data target dan titik data latih. Jarak dihitung antara titik data dalam kumpulan data dan titik target.



3. **Menemukan Tetangga Terdekat** → Titik data k dengan jarak terpendek ke titik target merupakan tetangga terdekat.



4. **Memilih Klasifikasi atau Mengambil Rata-rata untuk Regresi**
   - Ketika Anda ingin mengklasifikasikan suatu titik data ke dalam kategori spam atau bukan spam, algoritma KNN akan melihat K titik terdekat dalam kumpulan data. Titik-titik

terdekat ini disebut tetangga. Algoritme kemudian akan melihat kategori tetangga tersebut dan memilih titik yang paling sering muncul. Ini disebut pemungutan suara mayoritas.

- Dalam regresi, algoritma masih mencari K titik terdekat. Alih-alih memilih kelas dalam klasifikasi, algoritma mengambil rata-rata dari nilai K tetangga tersebut. Rata-rata ini adalah nilai prediksi untuk titik baru bagi algoritma.

Kelebihan KNN

- **Mudah digunakan** → mudah dipahami dan diimplementasikan.
- **Tidak membutuhkan tahapan training (pelatihan)**→ metode ini tidak membutuhkan pelatihan data, karena data akan disimpan dan digunakan selama proses prediksi.
- **Parameter yang sedikit** → metode ini hanya butuh mengatur nilai neighbor atau ketetanggaan (k) dan jarak.
- **Serbaguna** → Berfungsi untuk masalah klasifikasi dan regresi.

Kekurangan KNN

- **Lambat dengan data besar** → Perlu membandingkan setiap poin selama prediksi.
- **Berjuang dengan banyak fitur** → Akurasi menurun ketika data memiliki terlalu banyak fitur.
- **Dapat Terjadi Overfitting** → Dapat terjadi overfitting terutama ketika data berdimensi tinggi atau tidak bersih.

Contoh Kasus

1. Dataset: menggunakan dataset pada metode Decision Tree
   vektor biner fitur: [bagus, buruk, cepat, mahal]
   ringkasan data (index = ID):
   [1,0,0,0] → positif
   [0,1,0,0] → negatif
   [1,0,1,0] → positif
   [0,1,0,0] → negatif
   [0,1,0,1] → negatif
   [1,0,0,0] → positif
   [0,1,1,0] → negatif
   [1,0,0,0] → positif

2. Perhitungan jarak per sampel (k = 3 — pilih neighbor terdekat)

Metode: Euclidean

Sampel 1 (ID 1): test = [1,0,0,0] (true = positif)

Hitung jarak ke data lain:

- ke ID6 [1,0,0,0]: d=0 (identik)
- ke ID8 [1,0,0,0]: d=0
- ke ID3 [1,0,1,0]: $d = \sqrt{(1-1)^2 + (0-0)^2 + (0-1)^2 + (0-0)^2} = \sqrt{1} = 1$
- ke ID2 [0,1,0,0]: $d = \sqrt{(1-0)^2 + (0-1)^2 + (0-0)^2 + (0-0)^2} = \sqrt{2} = 1.4142$
- ke ID4 [0,1,0,0]: $d = \sqrt{2} = 1.4142$
- ke ID5 [0,1,0,1]: $d = \sqrt{(1-0)^2 + (0-1)^2 + (0-0)^2 + (0-1)^2} = \sqrt{3} = 1.7321$
- ke ID7 [0,1,1,0]: $d = \sqrt{(1-0)^2 + (0-1)^2 + (0-1)^2 + (0-0)^2} = \sqrt{3} = 1.7321$

3 nearest: ID 6 (pos), 8 (pos), 3 (pos) → prediksi = positif. (benar)

Sampel 2 (ID 2): test = [0,1,0,0] (true = negatif)

Hitung jarak ke data lain:

- ke ID4 [0,1,0,0]: d=0 (identik)
- ke ID5 [0,1,0,1]: $d = \sqrt{(0-0)^2 + (1-1)^2 + (0-0)^2 + (0-1)^2} = \sqrt{1} = 1$
- ke ID7 [0,1,1,0]: $d = \sqrt{(0-0)^2 + (1-1)^2 + (0-1)^2 + (0-0)^2} = \sqrt{1} = 1$
- ke ID1 [1,0,0,0]: $d = \sqrt{(0-1)^2 + (1-0)^2 + (0-0)^2 + (0-0)^2} = \sqrt{2} = 1.4142$
- ke ID6 [1,0,0,0]: $d = \sqrt{(0-1)^2 + (1-0)^2 + (0-0)^2 + (0-0)^2} = \sqrt{2} = 1.4142$
- ke ID8 [1,0,0,0]: $d = \sqrt{(0-1)^2 + (1-0)^2 + (0-0)^2 + (0-0)^2} = \sqrt{2} = 1.4142$
- ke ID3 [1,0,1,0]: $d = \sqrt{(0-1)^2 + (1-0)^2 + (0-1)^2 + (0-0)^2} = \sqrt{3} = 1.7321$

3 nearest: ID 4 (neg), 5 (neg), 7 (neg) → prediksi = negatif. (benar)

Sampel 3 (ID 3): test = [1,0,1,0] (true = positif)

Hitung jarak ke data lain:

- ke ID1 [1,0,0,0]: $d = \sqrt{(1-1)^2 + (0-0)^2 + (1-0)^2 + (0-0)^2} = \sqrt{1} = 1$
- ke ID6 [1,0,0,0]: $d = \sqrt{(1-1)^2 + (0-0)^2 + (1-0)^2 + (0-0)^2} = \sqrt{1} = 1$
- ke ID8 [1,0,0,0]: $d = \sqrt{(1-1)^2 + (0-0)^2 + (1-0)^2 + (0-0)^2} = \sqrt{1} = 1$
- ke ID7 [0,1,1,0]: $d = \sqrt{(1-0)^2 + (0-1)^2 + (1-1)^2 + (0-0)^2} = \sqrt{2} = 1.4142$
- ke ID2 [0,1,0,0]: $d = \sqrt{(1-0)^2 + (0-1)^2 + (1-0)^2 + (0-0)^2} = \sqrt{3} = 1.7321$
- ke ID4 [0,1,0,0]: $d = \sqrt{(0-1)^2 + (1-0)^2 + (0-1)^2 + (0-0)^2} = \sqrt{3} = 1.7321$

- ke ID5 [0,1,0,1]: $d = \sqrt{(1-0)^2 + (0-1)^2 + (1-0)^2 + (0-1)^2} = \sqrt{4} = 2$

3 nearest: ID 1 (pos), 6 (pos), 8 (pos) → prediksi = positif. (benar)

Sampel 4 (ID 4): test = [0,1,0,0] (true = negatif)

(sama pola seperti ID2)

3 nearest: IDs 2 (neg), 5 (neg), 7 (neg) → prediksi = negatif. (benar)

Sampel 5 (ID 5): test = [0,1,0,1] (true = negatif)

Hitung jarak ke data lain:

- ke ID2 [0,1,0,0]: $d = \sqrt{(0-0)^2 + (1-1)^2 + (0-0)^2 + (1-0)^2} = \sqrt{1} = 1$
- ke ID4 [0,1,0,0]: $d = \sqrt{(0-0)^2 + (1-1)^2 + (0-0)^2 + (1-0)^2} = \sqrt{1} = 1$
- ke ID7 [0,1,1,0]: $d = \sqrt{(0-0)^2 + (1-1)^2 + (0-1)^2 + (1-0)^2} = \sqrt{2} = 1.4142$
- ke ID1 [1,0,0,0]: $d = \sqrt{(0-1)^2 + (1-0)^2 + (0-0)^2 + (1-0)^2} = \sqrt{3} = 1.7321$
- ke ID6 [1,0,0,0]: $d = \sqrt{(0-1)^2 + (1-0)^2 + (0-0)^2 + (1-0)^2} = \sqrt{3} = 1.7321$
- ke ID8 [1,0,0,0]: $d = \sqrt{(0-1)^2 + (1-0)^2 + (0-0)^2 + (1-0)^2} = \sqrt{3} = 1.7321$
- ke ID3 [1,0,1,0]: $d = \sqrt{(0-1)^2 + (1-0)^2 + (0-1)^2 + (1-0)^2} = \sqrt{4} = 2$

Sampel 6 (ID 6): test = [1,0,0,0] (true = positif)

(sama pola seperti ID1)

3 nearest: IDs 1 (pos), 8 (pos), 3 (pos) → prediksi = positif. (benar)

Sampel 7 (ID 7) : test = [0,1,1,0] (true = negatif)

Hitung jarak ke data lain:

- ke ID2 [0,1,0,0]: $d = \sqrt{(0-0)^2 + (1-1)^2 + (1-0)^2 + (0-0)^2} = \sqrt{1} = 1$
- ke ID4 [0,1,0,0]: $d = \sqrt{(0-0)^2 + (1-1)^2 + (1-0)^2 + (0-0)^2} = \sqrt{1} = 1$
- ke ID3 [1,0,1,0]: $d = \sqrt{(0-1)^2 + (1-0)^2 + (1-1)^2 + (0-0)^2} = \sqrt{2} = 1.4142$
- ke ID5 [0,1,0,1]: $d = \sqrt{(0-0)^2 + (1-1)^2 + (1-0)^2 + (0-1)^2} = \sqrt{2} = 1.4142$
- ke ID1 [1,0,0,0]: $d = \sqrt{(0-1)^2 + (1-0)^2 + (1-0)^2 + (0-0)^2} = \sqrt{3} = 1.7321$
- ke ID6 [1,0,0,0]: $d = \sqrt{(0-1)^2 + (1-0)^2 + (1-0)^2 + (0-0)^2} = \sqrt{3} = 1.7321$
- ke ID8 [1,0,0,0]: $d = \sqrt{(0-1)^2 + (1-0)^2 + (1-0)^2 + (0-0)^2} = \sqrt{3} = 1.7321$

3 nearest: ID 2 (neg), 4 (neg), 3 (pos) → labels negatif, negatif, positif → mayoritas negatif → prediksi = negatif. (benar)

Sampel 8 (ID 8) : test = [1,0,0,0] (true = positif)
(sama pola seperti ID1/6)
3 nearest: IDs 1 (pos), 6 (pos), 3 (pos) → prediksi = positif. (benar)

Kesimpulan: dari 8 sampel dataset awal, 8 sampel terprediksi benar dengan k = 3 (accuracy = 8/8 = 100%).

2. Praktikum

2.1. Aplikasi Analisa Sentimen dengan Perbandingan Metode KNN dan Desicion Tree

Install Library

```
pip install streamlit
pip install matplotlib
pip install seaborn
pip install scikit-learn
```

Kode Program

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score
```

```
st.set_page_config(page_title="Perbandingan KNN vs Decision Tree",
layout="wide")

st.title("Analisa Sentimen: Perbandingan KNN vs Decision Tree")

# Upload dataset Excel
uploaded_file = st.file_uploader("Unggah dataset Excel", type=["xlsx"])
```

```
if uploaded_file:
    df = pd.read_excel(uploaded_file)

    st.subheader("Data Awal")
    st.write(df.head())

    text_col = st.selectbox("Pilih kolom teks", df.columns)
    label_col = st.selectbox("Pilih kolom label", df.columns)

    if st.button("Jalankan Analisis"):
        X = df[text_col].astype(str)
        y = df[label_col]

        # TF-IDF
        vectorizer = TfidfVectorizer(max_features=5000)
        X_vec = vectorizer.fit_transform(X)

        # Split
        X_train, X_test, y_train, y_test = train_test_split(
            X_vec, y, test_size=0.2, random_state=42)

        # Model KNN
        knn = KNeighborsClassifier(n_neighbors=5, metric="cosine")
        knn.fit(X_train, y_train)
        y_pred_knn = knn.predict(X_test)

        acc_knn = accuracy_score(y_test, y_pred_knn)
        report_knn = classification_report(y_test, y_pred_knn, output_dict=True)

        # Model Decision Tree
        dt = DecisionTreeClassifier(max_depth=10, random_state=42)
        dt.fit(X_train, y_train)
        y_pred_dt = dt.predict(X_test)

        acc_dt = accuracy_score(y_test, y_pred_dt)
        report_dt = classification_report(y_test, y_pred_dt, output_dict=True)

        # Simpan ke session_state untuk dipakai di prediksi manual
        st.session_state.vectorizer = vectorizer
        st.session_state.knn = knn
        st.session_state.dt = dt
        st.session_state.ready = True

        # Tampilkan Hasil
        st.subheader("Hasil Perbandingan")
        result_df = pd.DataFrame({
            "Model": ["KNN", "Decision Tree"],
            "Accuracy": [acc_knn, acc_dt]
        })
        st.write(result_df)

        # Grafik Akurasi
        fig1, ax1 = plt.subplots()
        sns.barplot(data=result_df, x="Model", y="Accuracy", ax=ax1, palette="Set2")
        ax1.set_ylim(0, 1)
        st.pyplot(fig1)

        # Classification Report
        st.subheader("Classification Report KNN")
        st.write(pd.DataFrame(report_knn).transpose())

        st.subheader("Classification Report Decision Tree")
        st.write(pd.DataFrame(report_dt).transpose())
```

```
# Uji Prediksi Manual
st.markdown("---")
st.subheader("Uji Prediksi Teks Baru")

input_text = st.text_area("Masukkan teks untuk diprediksi")

if st.button("Prediksi"):
    if not st.session_state.ready:
        st.warning("Latih model terlebih dahulu (klik 'Jalankan Analisis').")
    elif not input_text.strip():
        st.warning("Teks tidak boleh kosong.")
    else:
        vec = st.session_state.vectorizer.transform([input_text])
        pred_knn = st.session_state.knn.predict(vec)[0]
        pred_dt = st.session_state.dt.predict(vec)[0]

        st.info(f"KNN: **{pred_knn}**")
        st.info(f"Decision Tree: **{pred_dt}**")
```

2.2. Aplikasi Analisa Sentimen Menggunakan Metode Desicion Tree

```
# ====== Fungsi Training Model ======
def train_model(df):
    X = df['Komentar Bersih']
    y = df['Label']
    vectorizer = TfidfVectorizer(max_features=5000)
    vektor_x = vectorizer.fit_transform(X)
    model = DecisionTreeClassifier(max_depth=10, random_state=42)
    model.fit(vektor_x, y)
    return vectorizer, model
```

```
# ====== Load Model jika ada ======
try:
    with open("tfidf_vectorizer.pkl", "rb") as f:
        vectorizer = pickle.load(f)
    with open("decision_tree_model.pkl", "rb") as f:
        model = pickle.load(f)
except:
    vectorizer, model = None, None

st.set_page_config(page_title="Analisis Sentimen", layout="centered")
st.title("Aplikasi Analisis Sentimen Interaktif")
st.write("Menggunakan **Logistic Regression** untuk memprediksi sentimen teks.")
```

```
# ====== Menu Navigasi ======
menu = st.sidebar.radio("Menu", ["Prediksi Sentimen", "Latih Model Baru"])

# ====== Halaman Prediksi ======
if menu == "Prediksi Sentimen":
    if model is None:
        st.warning("Model belum dilatih. Silakan latih model terlebih dahulu di menu
'Latih Model Baru'.")
    else:
        user_input = st.text_area("Masukkan teks atau ulasan:")
        if st.button("Prediksi Sentimen"):
            if user_input.strip() == "":
                st.warning("Harap masukkan teks terlebih dahulu.")
            else:
                # Prediksi
                input_tfidf = vectorizer.fit([user_input])
                prediction = model.predict(input_tfidf)[0]
                proba = model.predict_proba(input_tfidf)[0]

                # Warna indikator
                if prediction.lower() == "positif":
                    st.markdown(f"<h3 style='color:green;'>Sentimen:
                     {prediction}</h3>", unsafe_allow_html=True)
                else:
                    st.markdown(f"<h3 style='color:red;'>Sentimen:
                     {prediction}</h3>", unsafe_allow_html=True)

                st.write(f"Confidence: {max(proba)*100:.2f}%")

                # Grafik Confidence Score
                fig, ax = plt.subplots()
                ax.bar(["Negatif", "Positif"], proba, color=["red", "green"])
                ax.set_ylim(0, 1)
                ax.set_ylabel("Probabilitas")
                ax.set_title("Confidence Score")
                for i, v in enumerate(proba):
                    ax.text(i, v + 0.02, f"{v:.2f}", ha='center', fontsize=10)
                st.pyplot(fig)
```

```
# ====== Halaman Latih Model Baru ======
elif menu == "Latih Model Baru":
    st.subheader("Unggah Dataset Sentimen (Excel)")
    uploaded_file = st.file_uploader("Upload file Excel", type=["xlsx"])

    if uploaded_file:
        df = pd.read_excel(uploaded_file)
        if 'Komentar Bersih' not in df.columns or 'Label' not in df.columns:
            st.error("File Excel harus memiliki kolom 'Komentar Bersih' dan 'Label'")
        else:
            st.write("Preview Dataset:", df.head())

            if st.button("Latih Model"):
                vectorizer = train_model(df)
                with open("tfidf_vectorizer.pkl", "wb") as f:
                    pickle.dump(vectorizer, f)

                model = train_model(df)
                with open("decision_tree_model.pkl", "wb") as f:
                    pickle.dump(model, f)
                st.success("Model dan Vektor berhasil dilatih dan disimpan sebagai
                 'decision_tree_model.pkl', 'tfidf_vectorizer.pkl'.")

                # Download model
                with open("decision_tree_model.pkl", "rb") as f:
                    st.download_button(
                        label="Download Model",
                        data=f,
                        file_name="decision_tree_model.pkl",
                        mime="application/octet-stream")

                with open("tfidf_vectorizer.pkl", "rb") as f:
                    st.download_button(
                        label="Download Vektor",
                        data=f,
                        file_name="tfidf_vectorizer.pkl",
                        mime="application/octet-stream")
```

2.3. Aplikasi Analisa Sentimen Menggunakan Perbandingan Metode KNN

```
import streamlit as st
#import pickle
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
```

```
st.set_page_config(page_title="Analisa Sentimen KNN", layout="wide")

st.title("Aplikasi Analisa Sentimen (KNN)")

# Upload file Excel
uploaded_file = st.file_uploader("Unggah data Excel", type=["xlsx"])
```

```
if uploaded_file:
    df = pd.read_excel(uploaded_file)

    st.subheader("Data Awal")
    st.write(df.head())

    text_col = st.selectbox("Pilih kolom teks", df.columns)
    label_col = st.selectbox("Pilih kolom label", df.columns)

    if st.button("Evaluasi Model"):
        X = df[text_col].astype(str)
        y = df[label_col]

        # TF-IDF
        vectorizer = TfidfVectorizer(max_features=5000)
        X_vec = vectorizer.fit_transform(X)
        X_train, X_test, y_train, y_test = train_test_split(
            X_vec, y, test_size=0.2, random_state=42)

        # Model KNN
        model = KNeighborsClassifier(n_neighbors=3, metric="cosine")
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        # Classification Report
        st.subheader("Laporan Kinerja")
        report = classification_report(y_test, y_pred, output_dict=True)
        st.write(pd.DataFrame(report).transpose())
```

```
st.markdown("---")
st.subheader("Uji Prediksi Teks Baru")
input_text = st.text_area("Masukkan teks")

if st.button("Prediksi Sentimen"):
    if input_text.strip() != "":
        input_vec = vectorizer.transform([input_text])
        pred = model.predict(input_vec)[0]
        st.success(f"Prediksi Sentimen: **{pred}**")
```

3. Rangkuman

Secara sederhana, machine learning dibagi menjadi empat ranah, yaitu supervised learning, unsupervised learning, semi-supervised learning dan re-inforcement learning. Pada modul ini akan dibahas supervised learning atau pembelajaran terawasi. Dikatakan terawasi karena proses pembelajaran pembelajaran akan bergantung pada label data sebagai fitur dependennya. Supervised learning memiliki dua kategori, yaitu klasifikasi dan regresi. Klasifikasi digunakan pada data yang akan dikategorikan sesuai label data. Sedangkan regresi digunakan pada data numerik yang akan diprediksi nilai berikutnya sehingga bersifat kontinu. Pada bagian praktikum pemodelan yang digunakan adalah pemodelan klasifikasi dengan perbandingan metode Decision Tree dan K-Nearest Neighbor. Kasus yang digunakan adalah analisa sentimen review film yang didapatkan dari scrapping data modul sebelumnya. Data dilabeli positif dan negatif untuk melihat dan memprediksi sentimen terhadap sebuah film.

4. Tes Formatif

5. Daftar Pustaka

[1] T. Geeksforgeeks, "Supervised Machine Learning," Geeksforgeeks, 11 07 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/supervised-machine-learning/. [Accessed 24 08 2025].

[2] T. Geeksforgeeks, "Getting started with Classification," Geeksforgeeks, 23 07 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/getting-started-with-classification/. [Accessed 26 08 2025].

[3] T. Geeksforgeeks, "Regression in Machine Learning," Geeksforgeeks, 13 01 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/regression-in-machine-learning/. [Accessed 26 08 2025].

[4] T. Geeksforgeeks, "Decision Tree," Geeksforgeeks, 30 06 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/decision-tree/. [Accessed 31 08 2025].

[5] T. Geeksforgeeks, "K-Nearest Neighbor(KNN) Algorithm," Geeksforgeeks, 23 08 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/k-nearest-neighbours/. [Accessed 02 09 2025].