Pengantar Data Mining #5: Klasifikasi [1]

Isnan Mulia, S.Komp, M.Kom

Apa Itu Klasifikasi?

- Tugas analisis data, dalam bentuk membangun model/*classifier* untuk memprediksi label kelas:
  - ➢ "aman" atau "beresiko", untuk data pengajuan kredit
  - ➢ "perlakuan A", "perlakuan B", atau "perlakuan C", untuk data medis
  - ➢ "ya" atau "tidak", untuk bermain tenis
- Menggunakan data yang sudah memiliki label kelas
  - ➔ Disebut *supervised learning*

Relation: weather.symbolic

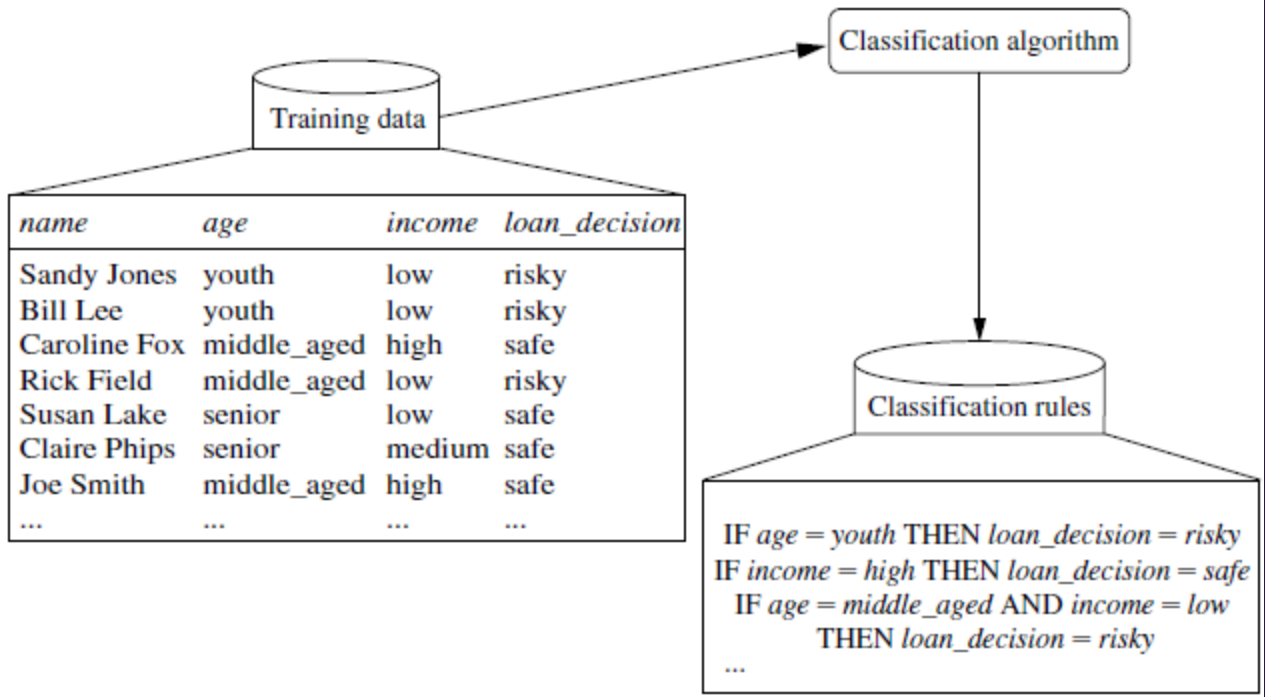
No.	1: outlook Nominal	2: temperature Nominal	3: humidity Nominal	4: windy Nominal	5: **play** Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes

Langkah Klasifikasi

- Fase pembelajaran (fase *training*)
  - Membangun model pengklasifikasi berdasarkan data *training*, yang terdiri dari *n* atribut & sudah memiliki label kelas

    Tuple X = (x_1, x_2, ..., x_n) → label kelas = *y*
  - Mempelajari fungsi *y* = *f*(*X*) yang dapat memprediksi label kelas yang berasosiasi *y* dari *tuple X* yang diberikan

Langkah Klasifikasi - *Training*

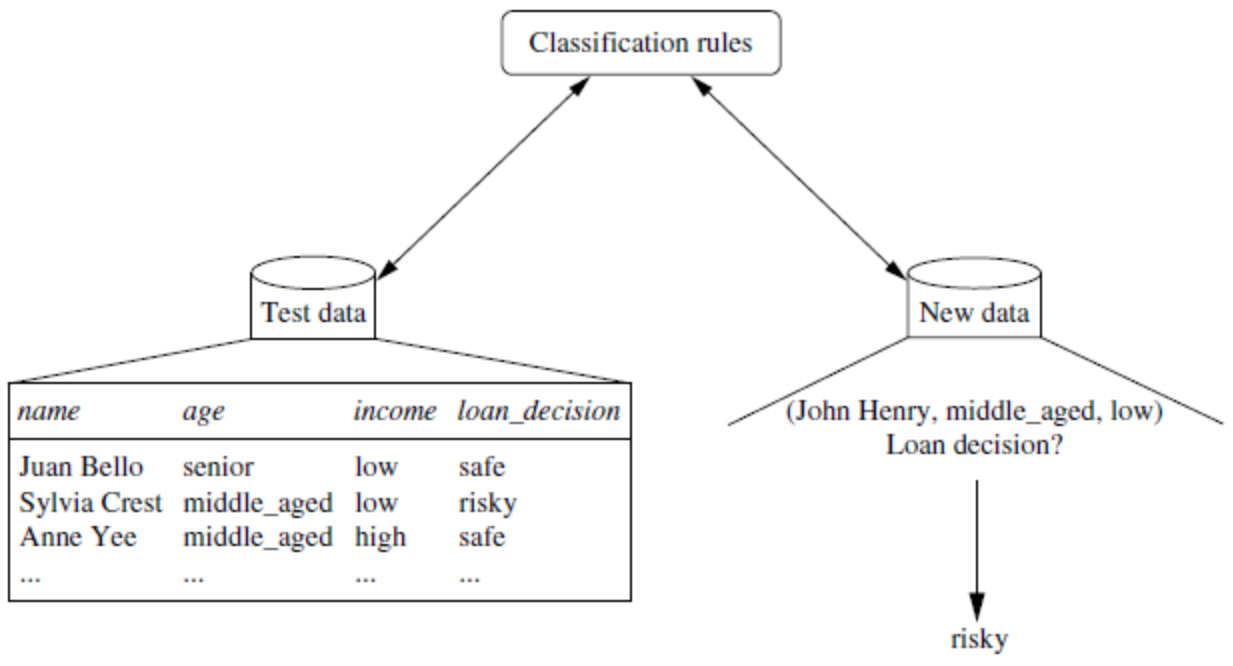


name	*age*	*income*	*loan_decision*
Sandy Jones	youth	low	risky
Bill Lee	youth	low	risky
Caroline Fox	middle_aged	high	safe
Rick Field	middle_aged	low	risky
Susan Lake	senior	low	safe
Claire Phips	senior	medium	safe
Joe Smith	middle_aged	high	safe
...	...	...	...

Langkah Klasifikasi

- Fase klasifikasi (fase *testing*)
  - Menerapkan model klasifikasi untuk memprediksi label kelas untuk data yang diberikan
  - Menggunakan data khusus untuk *testing*
  - Usahakan untuk tidak menggunakan data yang digunakan untuk *training*, karena ada potensi untuk *overfit*
  - Akurasi = $\frac{\text{jumlah } \textit{testing tuple} \text{ yang terklasifikasi benar}}{\text{jumlah } \textit{testing tuple}} \times 100\%$

Langkah Klasifikasi - *Testing*



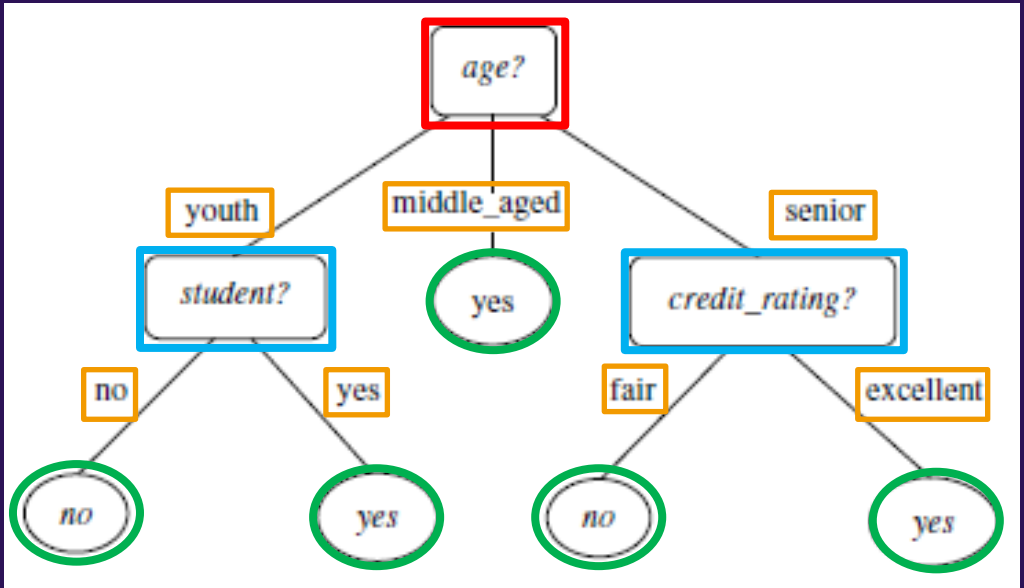
Variasi Metode Klasifikasi

1. *Decision Tree*
2. *Neural Network*
3. *Support Vector Machine*
4. *k Nearest Neighbours*
5. *Fuzzy Inference System*
6. *Deep Learning*

Decision Tree

- Membangun sebuah pohon yang membaca beberapa nilai atribut dari data untuk menentukan kelas label untuk data tersebut
- Pembuatan *decision tree* tidak memerlukan pengetahuan domain atau *setting* parameter
- Representasi pengetahuan yang disajikan intuitif & mudah dicerna
- Mengadopsi pendekatan *greedy* → ambil bagian berikutnya yang menawarkan keuntungan langsung
- Varian:
  - ID3
  - C4.5
  - CART

Decision Tree – Anatomi



- *Root node*: node paling atas
- *Node* internal/*non-leaf node*: pengujian pada suatu atribut
- Nilai atribut yang diperiksa
- *Leaf node*: kelas label

Decision Tree – Algoritma Umum

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition, *D*.

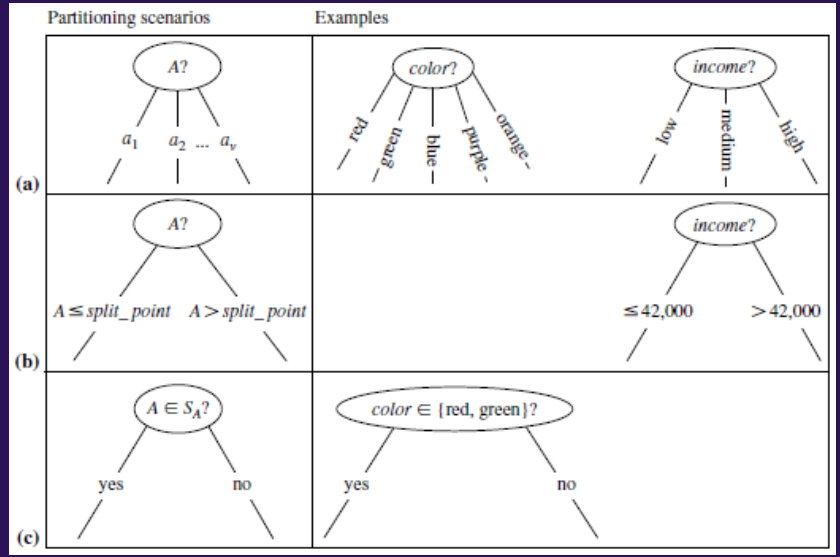
Input:

- Data partition, *D*, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split-point* or *splitting subset*.

Output: A decision tree.

Method:

(1) create a node *N*;
(2) **if** tuples in *D* are all of the same class, *C*, **then**
(3) return *N* as a leaf node labeled with the class *C*;
(4) **if** *attribute_list* is empty **then**
(5) return *N* as a leaf node labeled with the majority class in *D*; // majority voting
(6) apply **Attribute_selection_method**(*D*, *attribute_list*) to **find** the "best" *splitting_criterion*;
(7) label node *N* with *splitting_criterion*;
(8) **if** *splitting_attribute* is discrete-valued **and**
multiway splits allowed **then** // not restricted to binary trees
(9) *attribute_list* ← *attribute_list* − *splitting_attribute*; // remove *splitting_attribute*
(10) **for each** outcome *j* of *splitting_criterion*
// partition the tuples and grow subtrees for each partition
(11) let D_j be the set of data tuples in *D* satisfying outcome *j*; // a partition
(12) **if** D_j is empty **then**
(13) attach a leaf labeled with the majority class in *D* to node *N*;
(14) **else** attach the node returned by **Generate_decision_tree**(D_j, *attribute_list*) to node *N*;
endfor
(15) return *N*;



Decision Tree – Ukuran Seleksi Atribut

- Heuristik untuk memilih kriteria pemisah (*splitting criteria*) terbaik yang dapat memisahkan partisi data yang diberikan menjadi kelas-kelas tunggal
- Hasil partisi diharapkan "semurni mungkin" → setiap *tuple* yang terdapat pada suatu partisi merupakan anggota dari kelas label yang sama
- Varian:
  - *Information gain* → digunakan pada ID3

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

$$Gain(A) = Info(D) - Info_A(D)$$

→ Atribut *A* dengan *Gain*(*A*) tertinggi akan dipilih sebagai atribut pemisah

Decision Tree – Ukuran Seleksi Atribut

- Varian:
  - *Gain Ratio* → digunakan pada C4.5, untuk menghindari bias

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

  - *Gini Index* → digunakan pada CART, untuk menghitung "ketidakmurnian"

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

$$Gini_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} Gini(D_j)$$

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

Decision Tree – Contoh Penerapan ID3

RID	Age	Income	Student	Credit_Rating	Buys_Computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

youth: yes = 2, no = 3; middle_aged: yes = 4, no = 0; senior: yes = 3, no = 2

Menentukan *root node*:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$$= -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right)$$

$$= 0{,}940\ bits$$

$$Info_{age}(D) = \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right)$$

$$+ \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4}\right)$$

$$+ \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right)$$

$$= 0{,}694\ bits$$

Decision Tree – Contoh Penerapan ID3

RID	Age	Income	Student	Credit_Rating	Buys_Computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

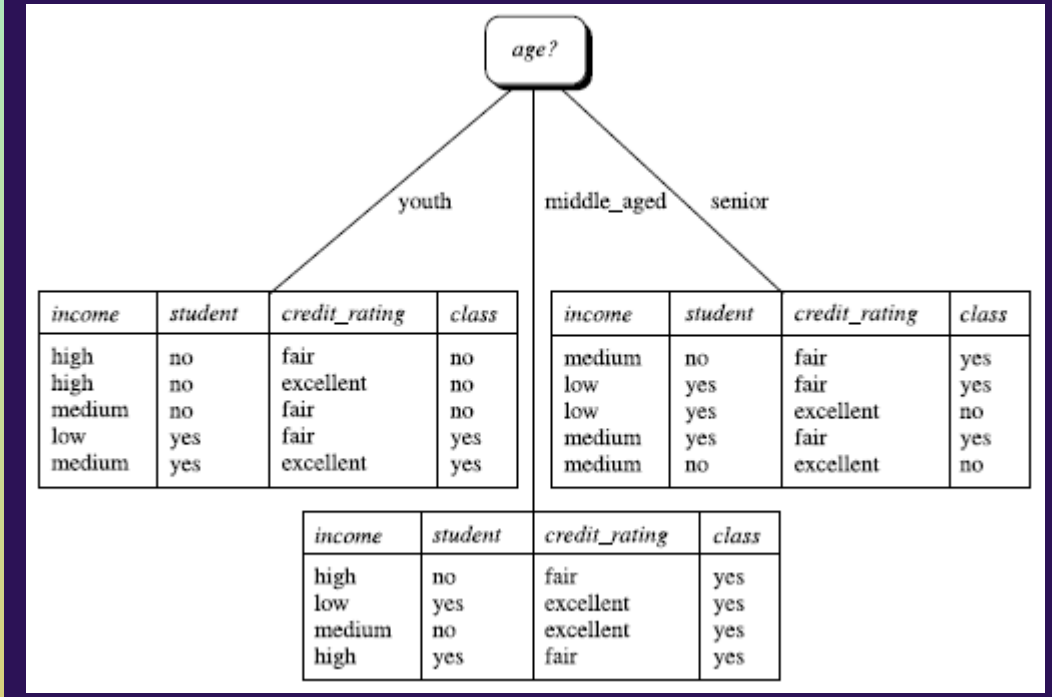
$$Info(D) = 0{,}940\ bits$$

$$\begin{aligned} Gain(age) &= Info(D) - Info_{age}(D) \\ &= 0{,}940 - 0{,}694 \\ &= 0{,}246\ bits \end{aligned}$$

$$Gain(income) = 0{,}029\ bits$$
$$Gain(student) = 0{,}151\ bits$$
$$Gain(credit\_rating) = 0{,}048\ bits$$

➔ Atribut *age* terpilih sebagai *splitting criteria* & akan ditempatkan pada *root node*

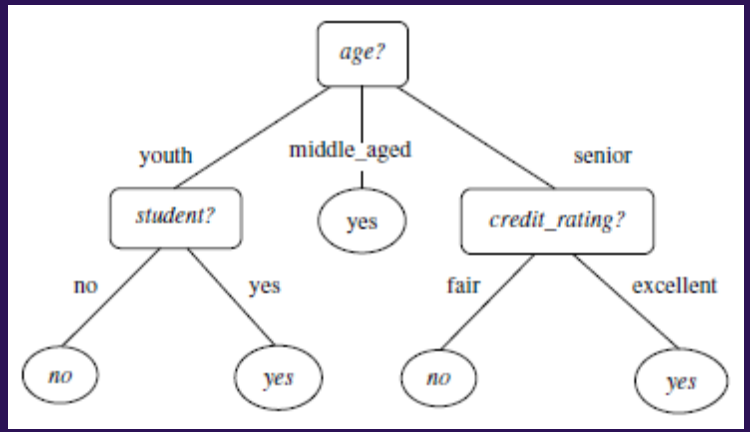
Decision Tree – Contoh Penerapan ID3



income	*student*	*credit_rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

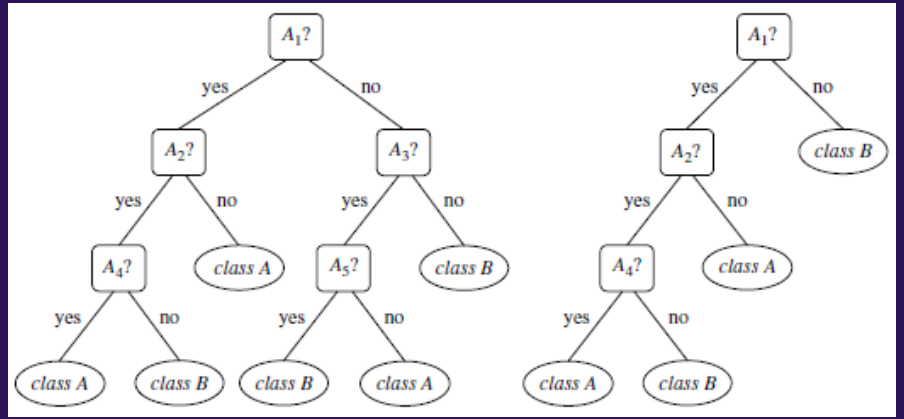
income	*student*	*credit_rating*	*class*
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	*student*	*credit_rating*	*class*
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes



Tree Pruning

- "Pemangkasan pohon"
- Menggunakan ukuran statistik untuk menghapus cabang yang kurang dapat diandalkan
- Dapat meningkatkan akurasi
- Varian:
  - *Prepruning*: cabang pohon baru dipangkas sejak awal proses pembentukannya
  - *Postpruning*: menghapus cabang pohon dari pohon yang sudah selesai dibentuk



Kelebihan & Kekurangan

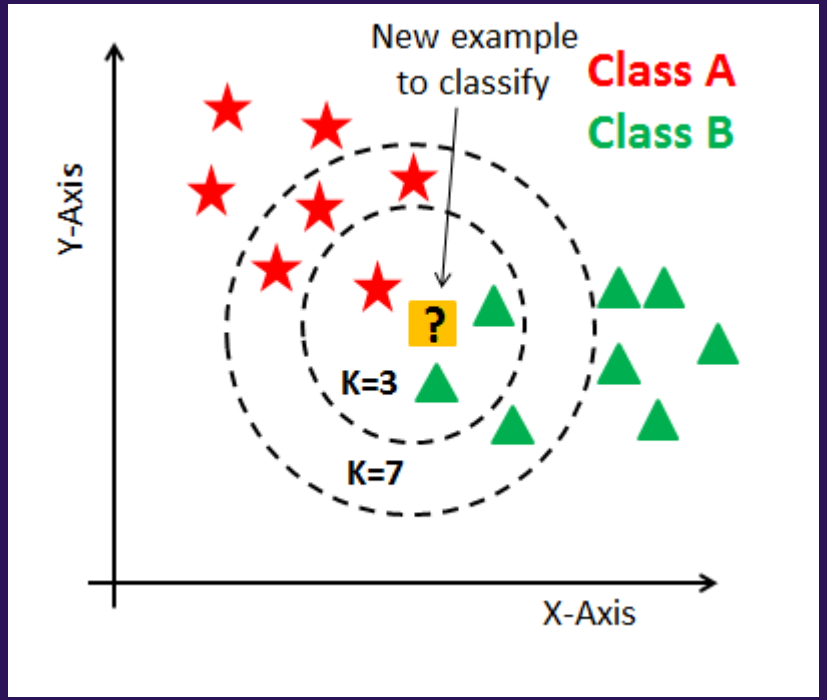
- Kelebihan:
  - Konsep yang jelas & mudah dipahami
  - Mudah diimplementasikan menggunakan algoritma rekursif
- Kekurangan:
  - Sulit diaplikasikan untuk himpunan data sangat besar dengan jumlah atribut & objek data yang banyak
  - Mudah mengalami *overfit*, karena proses pelatihan *greedy* tidak menjamin dihasilkannya *decision tree* yang optimal

K Nearest Neighbours (kNN)

- Membandingkan data baru dengan *k* buah *training tuple* yang paling dekat
- Kedekatan antara data baru dengan *training tuple* ditentukan oleh ukuran jarak, seperti jarak Euclidean

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ip} - x_{jp})^2}$$

- Data baru akan diberikan label sesuai dengan label kelas yang paling banyak yang dimiliki oleh *k* tetangga terdekatnya
- Termasuk kategori *lazy learner*, karena tidak ada proses pembuatan model klasifikasi yang digeneralisasi



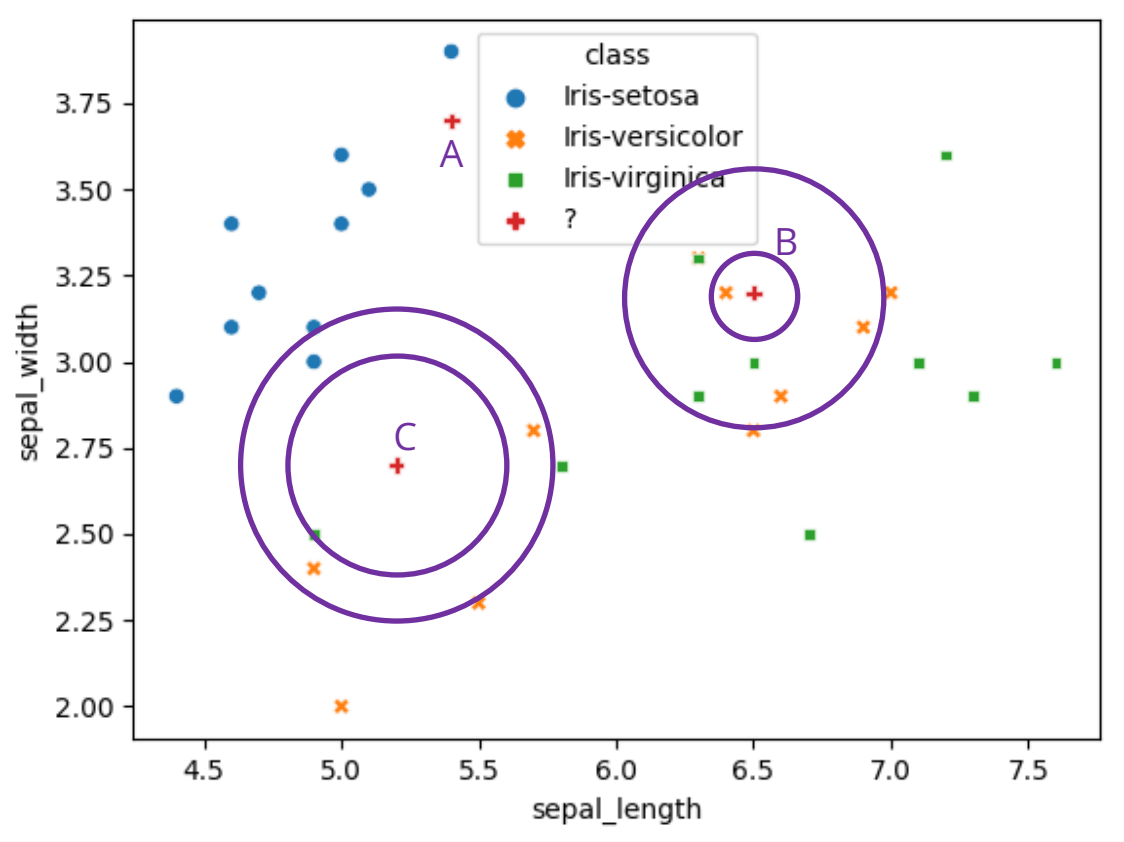
Kelas untuk data baru:

$k = 3$ → kelas = B

$k = 7$ → kelas = A

Sumber:
https://res.cloudinary.com/dyd911kmh/image/upload/f_auto,
q_auto:best/v1531424125/Knn_k1_z96jba.png

K Nearest Neighbours – Contoh



Data: klik di sini

k = 1:
Titik A → Iris-setosa
Titik B → Iris-versicolor
Titik C → Iris-virginica

k = 3:
Titik A → Iris-setosa
Titik B → Iris-versicolor
Titik C → Iris-setosa

k = 7:
Titik A → Iris-setosa
Titik B → Iris-versicolor
Titik C → Iris-versicolor

Kelebihan & Kekurangan

- Kelebihan:
  - Sangat sederhana
- Kekurangan:
  - Sangat sensitif dengan derau maupun data pencilan
  - "Mahal" secara komputasional, karena harus menyimpan semua data *training*
  - Harus berhati-hati dalam pemilihan parameter *k*, karena:
    - *k* merupakan satu-satunya parameter pada *kNN*
    - Nilai *k* tidak dapat ditentukan menggunakan rumus, tetapi ditemukan melalui percobaan

Evaluasi Klasifikasi

Diturunkan dari *confusion matrix*:

Kelas prediksi

Kelas sebenarnya	ya	tidak	Jumlah
ya	*TP*	*FN*	*P*
tidak	*FP*	*TN*	*N*
Jumlah	*P′*	*N′*	*P* + *N*

- *True Positive*: *tuple* positif yang diberikan label yang benar
- *True Negative*: *tuple* negatif yang diberikan label yang benar
- *False Positive*: *tuple* negatif yang diberikan label yang salah, sebagai positif
- *False Negative*: *tuple* positif yang diberikan label yang salah, sebagai negatif

Evaluasi Klasifikasi

Kelas prediksi

Kelas sebenarnya	ya	tidak	Jumlah
ya	*TP*	*FN*	*P*
tidak	*FP*	*TN*	*N*
Jumlah	*P′*	*N′*	*P* + *N*

- Akurasi = $\frac{TP + TN}{P + N}$
- *Error rate* = $\frac{FP + FN}{P + N}$
- *Sensitivity* / *recall* / *true positive rate* = $\frac{TP}{P}$
- *Specificity* / *true negative rate* = $\frac{TN}{N}$
- *Precision* = $\frac{TP}{TP + FP}$
- *F* / *F-score* = $\frac{2 \times precision \times recall}{precision + recall}$

Recap

- Definisi klasifikasi
- Varian metode klasifikasi
- *Decision Tree*
- *K Nearest Neighbours*
- Evaluasi klasifikasi

Next: apa perbedaan antara klasifikasi dan *clustering*?

つづく