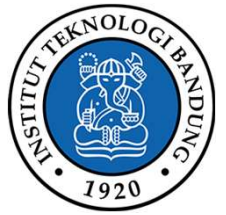


Klasifikasi #01

IF4041 – Semester I 2020/2021

Slide diadopsi sumber:
Sumber: Jiawei Han, Micheline Kamber, and Jian Pei,

Data Mining: Concepts & Techniques , 3rd Ed.

Ikhtisar

- Pengantar
- K-Nearest Neighbor
- Decision Tree
- Naïve Bayes Classifier
- Metrik Evaluasi untuk Klasifikasi
- Cross Validation
- Metode Ensemble

Pengantar

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by **labels** indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Prediction Problems: Classification vs. Numeric Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Numeric Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

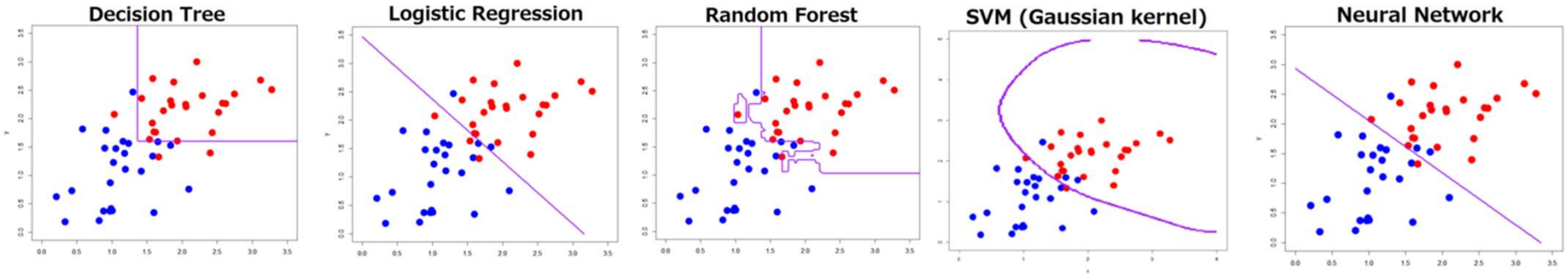
Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model to classify new data
- Note: If *the test set* is used to select models, it is called validation (test) set

Pengantar

- **Pengertian:** sebuah tugas yang ***mengidentifikasi kelas*** suatu data/pengamatan, dari domain kelas yang diberikan. Secara teknis, klasifier bertugas untuk membuat ***decision boundary***.

- **Contoh tugas klasifikasi:**
  - ✓ Membedakan email spam *vs.* bukan spam
  - ✓ Mengklasifikasikan sentimen dari suatu teks
  - ✓ OCR (*optical character recognition*), khususnya setelah memisahkan teks menjadi per karakter
  - ✓ Fitur login menggunakan wajah
  - ✓ Mendeteksi *fraud* dalam suatu transaksi

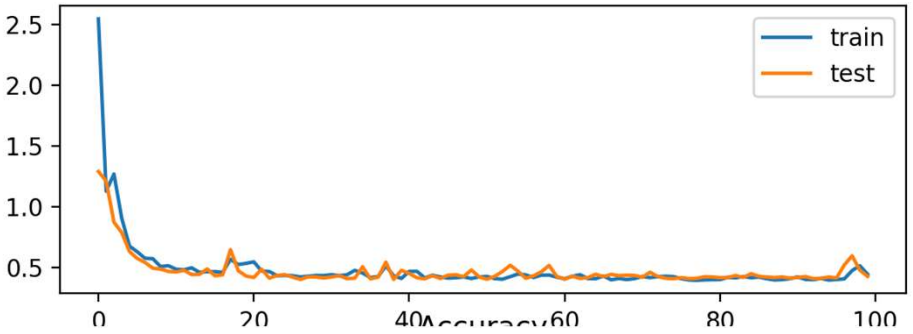


Alur Klasifikasi

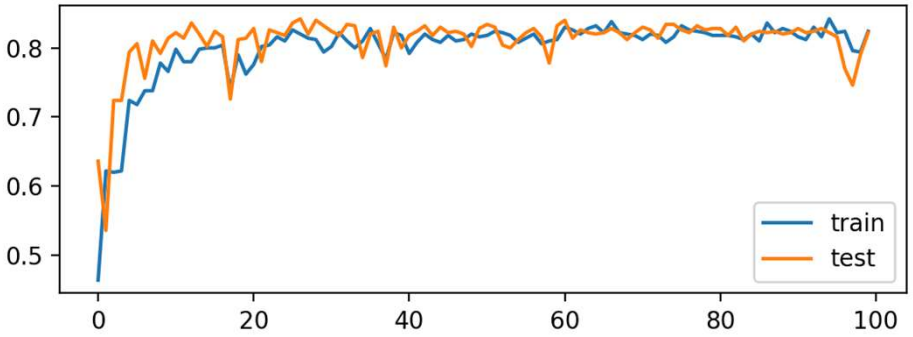
- (i) data training dan (ii) testing
- (i) data training, (ii) validation, (iii) testing

model klasifier

loss



akurasi



Contoh hasil pelatihan dari model klasifier: grafik akurasi dan loss

Contoh Klasifikasi Menggunakan Tools

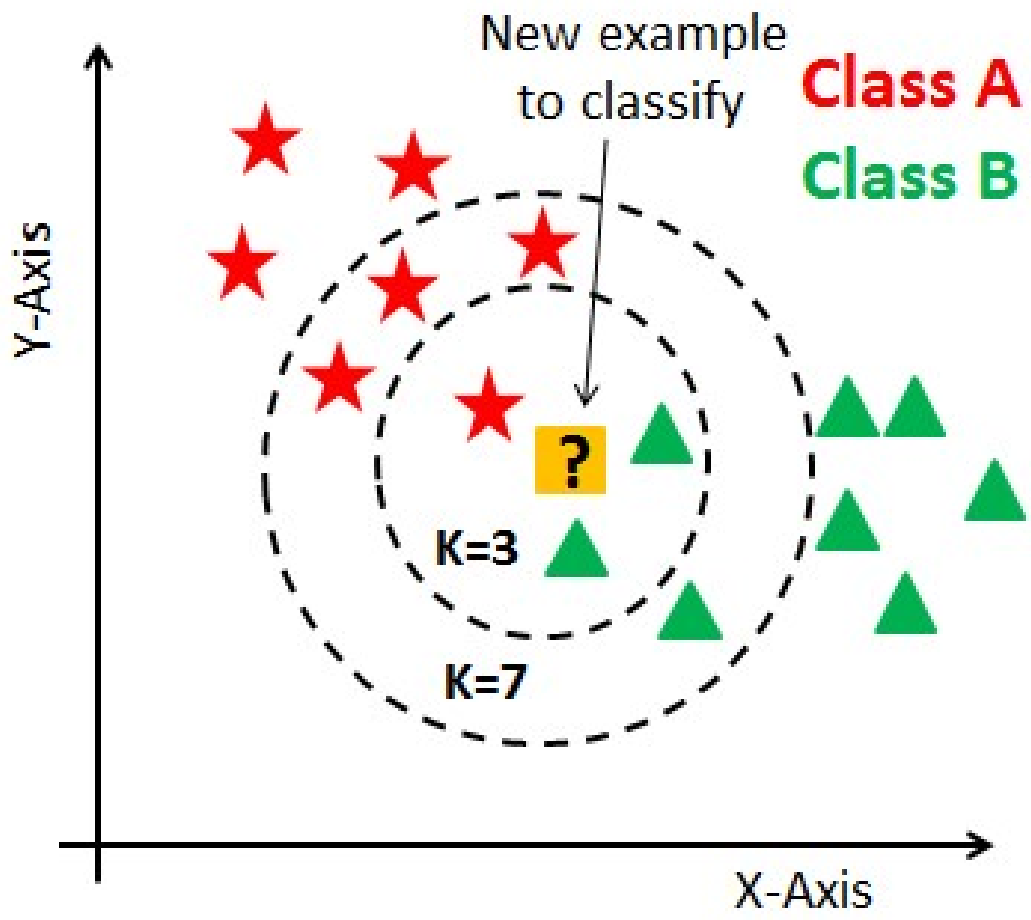
```
from sklearn import neighbors #import untuk KNN
from sklearn import tree #import untuk decision tree

MODEL = ['knn', 'svm', 'dt']; MODEL = MODEL[0]
if MODEL == 'knn':
    model = neighbors.KNeighborsClassifier(n_neighbors=5) #mendefinisikan model
    model.fit(X_train, y_train) #mentraining model
    hasil_prediksi = model.predict(X_test) #mengetes model
elif MODEL == 'dt':
    model = tree.DecisionTreeClassifier() #mendefinisikan model
    model.fit(X_train, y_train) #mentraining model
    hasil_prediksi = model.predict(X_test) #mengetes mode
```

- Sangat mudah bukan?

K-Nearest Neighbor

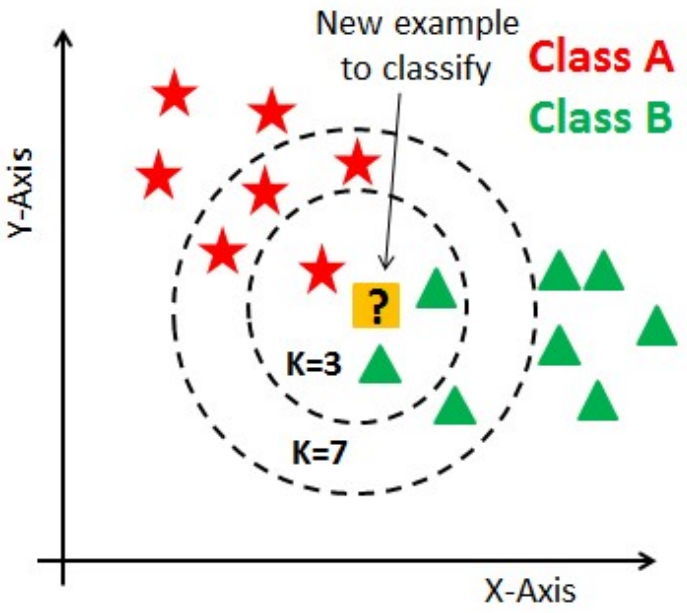
Cara Kerja KNN



- K = 3, '*new example*'
  → kelas B
- K = 7, '*new example*'
  → kelas ?

Cara Kerja KNN

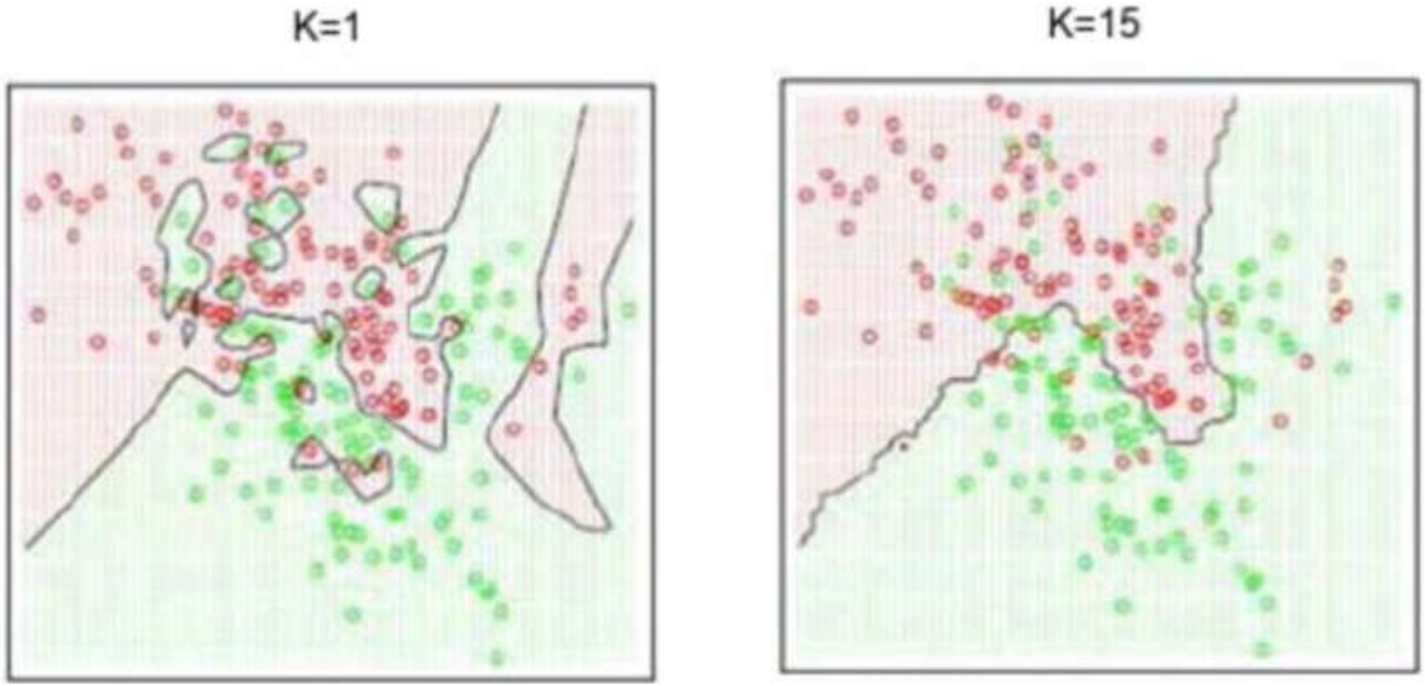
a. **Input**: set data training beserta labelnya, data testing, nilai *k*
b. Untuk data testing *i*, hitung jaraknya ke setiap data training
c. Tentukan *k*-data training yang jaraknya paling dekat dengan data testing *i*
d. Periksa label dari *k* data tersebut
e. Tentukan label yang frekuensinya paling banyak
f. Masukan data testing *i* tersebut ke kelas dengan frekuensi yang paling banyak
g. Ulangi langkah (3) sampai (6) untuk semua data testing yang lainnya.



Jarak yang lazim digunakan: Euclidean distance

$$Euc(X,Y) = \sqrt{\sum_{i=1}^{n}(X_i - Y_i)^2}$$

Efek Nilai *k* pada KNN

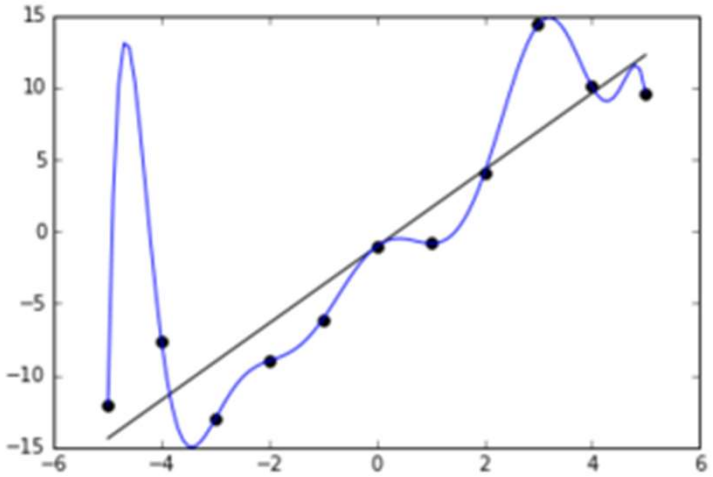


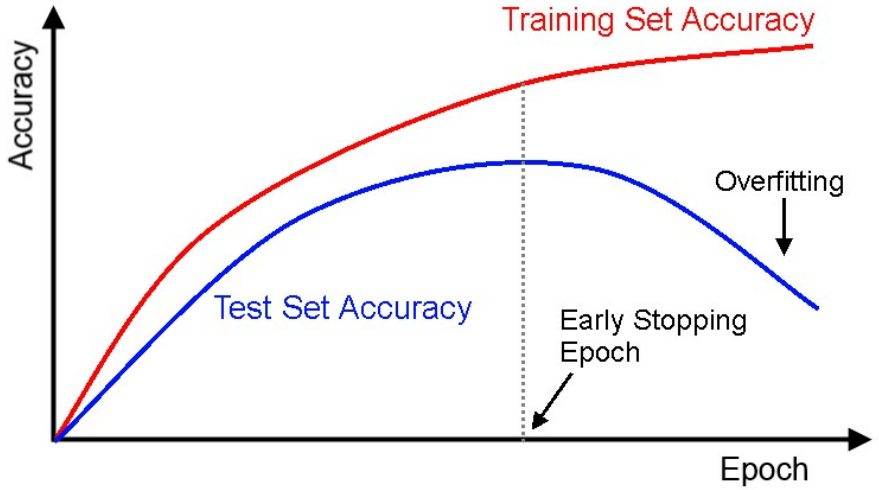
Efek nilai *k* terhadap hasil dar KNN

- Nilai *k* yang kecil cenderung membuat model *overfitting* (*less general*)
- Sebaliknya, nilai *k* yang besar cenderung membuat model yang general, dengan konsekuensi beberapa objek *'miss classified'*.

Konsep Overfitting

- **Pengertian**: memiliki kinerja bagus pada data training, tetapi kinerja akan turun drastis pada data testing
- Ilustrasi paling mudah adalah dengan model regresi (kiri) dan metode iteratif (kanan) berikut:





- *Trade-off* dari *overfitting* adalah ***generalization***.

Decision Tree

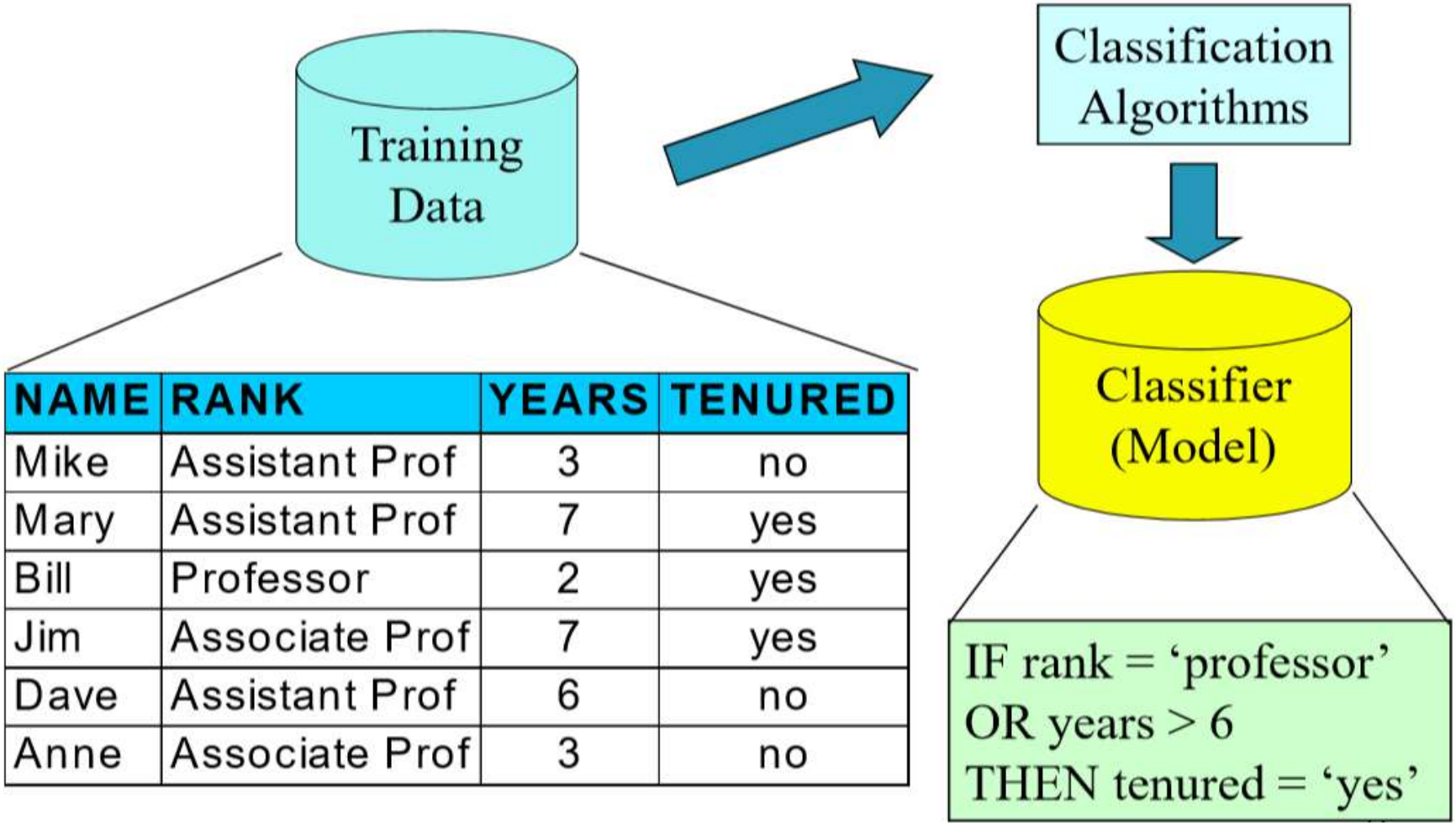
Pengantar Decision Tree

a. Salah satu metode klasifikasi yang paling populer, karena mudah untuk diinterpretasi oleh manusia, yakni berupa "**aturan-aturan dasar**", atau mirip dengan "***if-else***".

b. **Ilustrasi**: dari tabel ini, cobalah buat "aturan-aturan dasar" untuk membedakan "tenured professor" dan "bukan tenured professor"!

NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

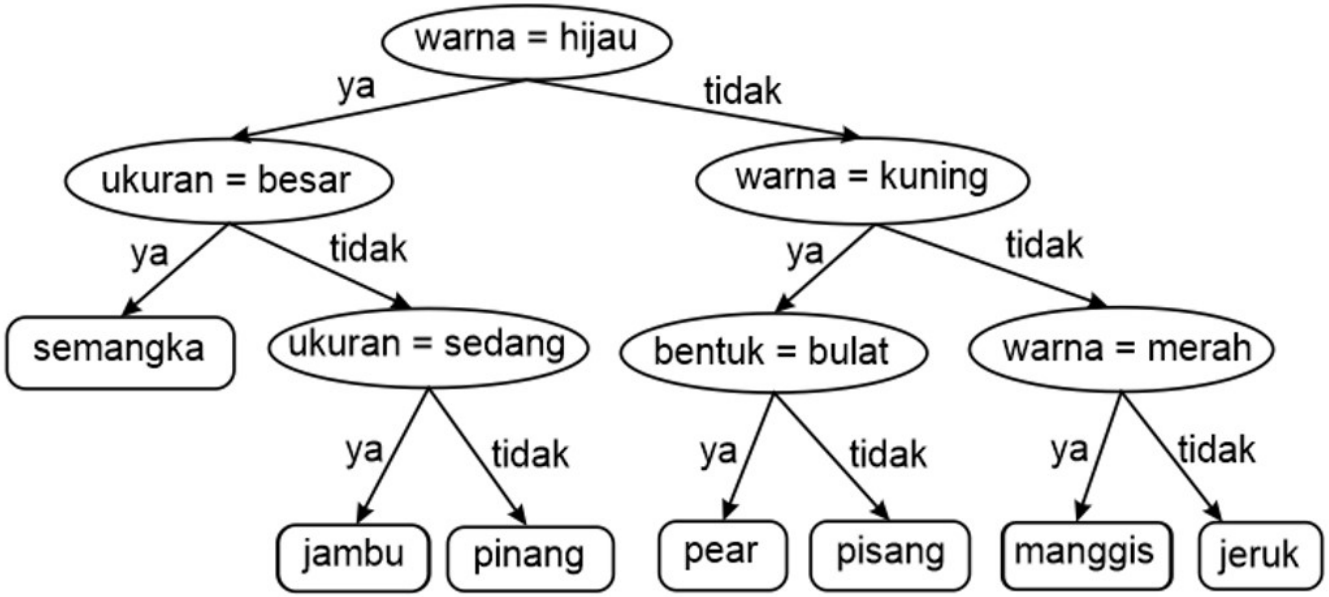
Pengantar Decision Tree



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Pengantar Decision Tree

- Seiring dengan tugas yang lebih kompleks, diperlukan aturan yang kompleks pula. Hal ini dapat distrukturkan menjadi sebuah “pohon keputusan”, seperti ilustrasi di bawah ini.



- Bagaimana cara memilih simpul (*node*) yang tepat? Decision tree adalah jawabannya.

Cara Kerja Decision Tree

- Diberikan sebuah tabel data $\boldsymbol{D}$ (‘main’ *vs* ‘tidak main’) sebagai berikut, kita akan memilih **atribut paling bagus** untuk dijadikan sebagai **simpul pemecah pertama**.

Cuaca	Temperatur	Kelembaban	Angin	Keputusan
x1	x2	x3	x4	Y
Cerah	Panas	Tinggi	kecil	tidak
Cerah	Panas	Tinggi	besar	tidak
Mendung	Panas	Tinggi	kecil	Ya
Hujan	Sedang	Tinggi	kecil	Ya
Hujan	Dingin	Normal	kecil	Ya
Hujan	Dingin	Normal	besar	Tidak
Mendung	Dingin	Normal	besar	Ya
Cerah	Sedang	Tinggi	kecil	Tidak
Cerah	Dingin	Normal	kecil	Ya
Hujan	Sedang	Normal	Kecil	Ya
Cerah	Sedang	Normal	Besar	Ya
Mendung	Sedang	Tinggi	Besar	Ya
Mendung	Panas	Normal	Kecil	Ya
Hujan	Sedang	Tinggi	Besar	Tidak

- Salah satu metodenya = ***information gain*.**

Cara Kerja Decision Tree

- Diberikan tabel data $\boldsymbol{D}$, banyaknya informasi yang dibutuhkan untuk pengklasifikasian data $\boldsymbol{D}$ tesebut adalah sebanyak nilai *entropy-nya,* yang dirumuskan dengan:

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

Dengan:

$p_i = probabilitas\ kelas_i$

$m = banyaknya\ kelas$

- Informasi yang dibutuhkan setelah memilih atribut A untuk memecah data $\boldsymbol{D}$ menjadi v bagian:

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

- Informasi gain = selisih kedua nilai tersebut di atas

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

- **Rule:** pilih atribut A yang memiliki nilai $Gain$ yang paling besar.

Cara Kerja Decision Tree

- Contoh perhitungan

Cuaca	Temperatur	Kelembaban	Angin	Keputusan
x1	x2	x3	x4	Y
Cerah	Panas	Tinggi	kecil	tidak
Cerah	Panas	Tinggi	besar	tidak
Mendung	Panas	Tinggi	kecil	Ya
Hujan	Sedang	Tinggi	kecil	Ya
Hujan	Dingin	Normal	kecil	Ya
Hujan	Dingin	Normal	besar	Tidak
Mendung	Dingin	Normal	besar	Ya
Cerah	Sedang	Tinggi	kecil	Tidak
Cerah	Dingin	Normal	kecil	Ya
Hujan	Sedang	Normal	Kecil	Ya
Cerah	Sedang	Normal	Besar	Ya
Mendung	Sedang	Tinggi	Besar	Ya
Mendung	Panas	Normal	Kecil	Ya
Hujan	Sedang	Tinggi	Besar	Tidak

- 9 ‘main’ , 5 ‘tidak’ dari total 14 baris data
- 5 cuaca cerah = 2 ‘main’ , 3 ‘tidak’
- 4 cuaca mendung = 4 ‘main’, ‘0’ tidak
- 5 cuaca hujan = 3 ‘main’, 2 ‘tidak’

- Entropy awal

$$Info(D) = Info(9,5)$$

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0{,}940$$

- Entropy setelah memilih atribut cuaca

$$Info_{cuaca}(D) = \frac{5}{14}info(2,3) + \frac{4}{14}info(4,0) + \frac{5}{14}info(3,2) = 0{,}693$$

- Informasi gain:

$$Gain(cuaca) = 0{,}940 - 0{,}693 = \mathbf{0{,}247}$$

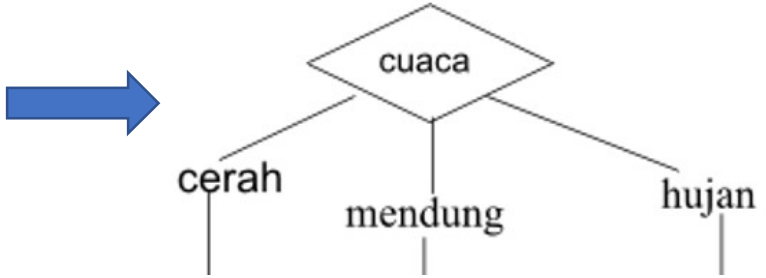
Dengan cara yang sama, akan didapatkan:

- ✓ $Gain(temp.) = 0{,}029$
- ✓ $Gain(kelembapan) = 0{,}152$
- ✓ $Gain(angin) = 0{,}048$

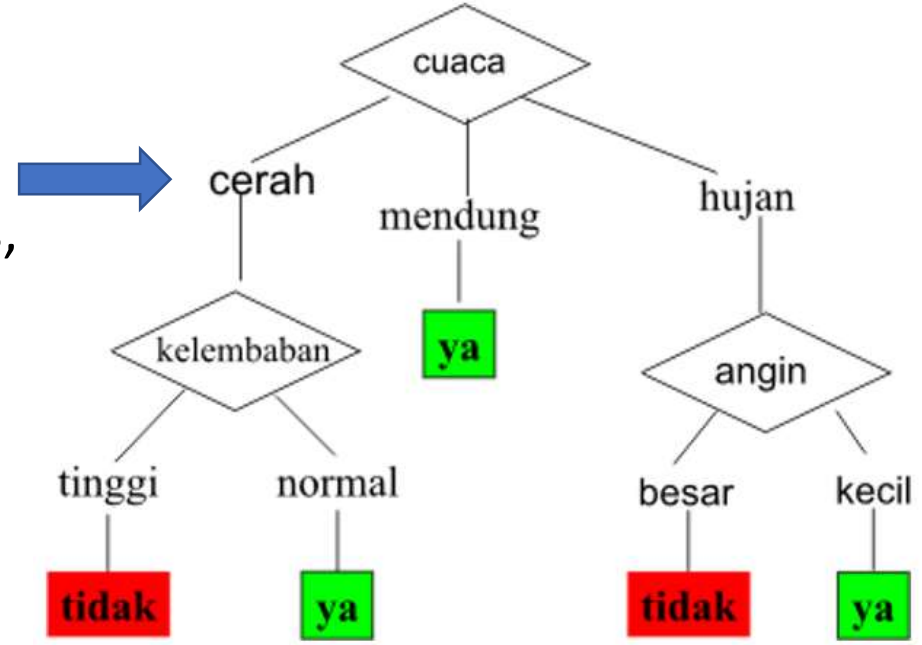
*__Note:__ jika subset atribut hanya 1 kelas, maka $nilai\ entropy = 0$

Cara Kerja Decision Tree

- Karena atribut cuaca memiliki nilai informasi gain paling besar, maka akan dijadikan sebagai simpul pemecah pertama.



- Di masing-masing anak simpul ‘cerah’, ‘mendung’ dan ‘hujan’, jika masih terdapat lebih dari satu kelas, maka ulangi langkahnya lagi untuk mencari simpul pemecah terbaik, begitu seterusnya.



Kriteria-kriteria Lain Untuk Menentukan Simpul Tree

- Selain *information gain,* kita dapat menggunakan:
  - ✓ *Gain ratio* (pilih yang paling besar). Gunakan split info untuk normalisasi

$$Gain\ ratio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

dengan:

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

  - ✓ *Indeks Gini* (pilih yang paling kecil)
    - Dalam sebuah pemecahan, hitung:
      - persentase per cabang (w_i)
      - IG_i)

$$IG_i = 1 - \sum_{k=1}^{C} p_k^2$$

Dengan p_k = probabilitas kelas $\{1,2,..C\}$

    - Hitung $\sum_{i=1}^{n}(w_i \times IG_i)$, dengan n adalah banyaknya cabang

Decision Tree – Beberapa Detail

- Menghindari overfitting
  - ✓ Salah satunya dapat dilakukan post-pruning
    - Tree yang kompleks akan rentan dengan overfitting
    - Maka dari itu, hapus simpul yang “kurang penting”
    - Ukuran “kurang penting” dapat diuji dengat set data di luar dari data training
- Atribut kontinyu
  - ✓ Secara dinamis membagi ke atribut diskret
- Beberapa varian dari metode decision tree yang diusulkan secara utuh: **ID3**, **C4.5**, **CART**
- Beberapa pengembangan dari klasifier tree-based: random forest, Xgboost (*Extreme Gradient Boosting*)

Klasifikasi Bayesian

Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers

Bayes' Theorem: Basics

- Total probability Theorem: $$P(B) = \sum_{i=1}^{M} P(B|A_i)P(A_i)$$

- Bayes' Theorem: $$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

  - Let **X** be a data sample ("*evidence*"): class label is unknown
  - Let H be a *hypothesis* that X belongs to class C
  - Classification is to determine P(H|**X**), (i.e., *posteriori probability):* the probability that the hypothesis holds given the observed data sample **X**
  - P(H) (*prior probability*): the initial probability
    - E.g., **X** will buy computer, regardless of age, income, …
  - P(**X**): probability that sample data is observed
  - P(**X**|H) (likelihood): the probability of observing the sample **X**, given that the hypothesis holds
    - E.g., Given that **X** will buy computer, the prob. that X is 31..40, medium income

Prediction Based on Bayes' Theorem

- Given training data **X**, *posteriori probability of a hypothesis* H, P(H|**X**), follows the Bayes' theorem

$$P(H \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid H)P(H)}{P(\mathbf{X})} = P(\mathbf{X} \mid H) \times P(H) / P(\mathbf{X})$$

- Informally, this can be viewed as

  posteriori = likelihood x prior/evidence

- Predicts **X** belongs to C_i iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|X)$ for all the *k* classes

- Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost

Classification Is to Derive the Maximum Posteriori

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector **X** = (x_1, x_2, …, x_n)
- Suppose there are *m* classes C_1, C_2, …, C_m.
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|\mathbf{X})$
- This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since P(X) is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times ... \times P(x_n|C_i)$$

- This greatly reduces the computation cost: Only counts the class distribution

- If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$ (# of tuples of C_i in D)

- If A_k is continous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

and $P(x_k|C_i)$ is

$$P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

Naïve Bayes Classifier: Training Dataset

Class:
C1:buys_computer = ‘yes’
C2:buys_computer = ‘no’

Data to be classified:
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair)

age	income	student	credit_rating	_comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayes Classifier: An Example

- $P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643
  P(buys_computer = "no") = 5/14= 0.357
- Compute $P(X|C_i)$ for each class
  P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222
  P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6
  P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
  P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
  P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
  P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
  P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
  P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4
- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = "yes")
= 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no")
= 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$:
P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

age	income	student	credit_rating	co
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Avoiding the Zero-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be **non-zero**. Otherwise, the predicted prob. will be zero

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)
- Use **Laplacian correction** (or Laplacian estimator)
  - *Adding 1 to each case*

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - The "corrected" prob. estimates are close to their "uncorrected" counterparts

Naïve Bayes Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.

      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier
- How to deal with these dependencies? Bayesian Belief Networks (Chapter 9)

Metrik Evaluasi untuk Klasifikasi

Confusion Matrix

- Diberikan data prediksi, misalkan:

n=165	Prediksi = -1	Prediksi = +1	
Aktual = -1	$TN = 50$	$FP = 10$	60
Aktual = +1	$FN = 5$	$TP = 100$	105
	55	110	

1. **TN (*True Negative*)**: output kelas negatif yang berhasil ditebak sebagai kelas negatif
2. **TP (*True Positive*)**: output kelas positif yang berhasil ditebak sebagai kelas positif
3. **FN (*False Negative*)**: output kelas positif yang salah ditebak sebagai kelas negatif
4. **FP (*False Positive*)**: output kelas negatif yang salah ditebak sebagai kelas positif

Confusion Matrix

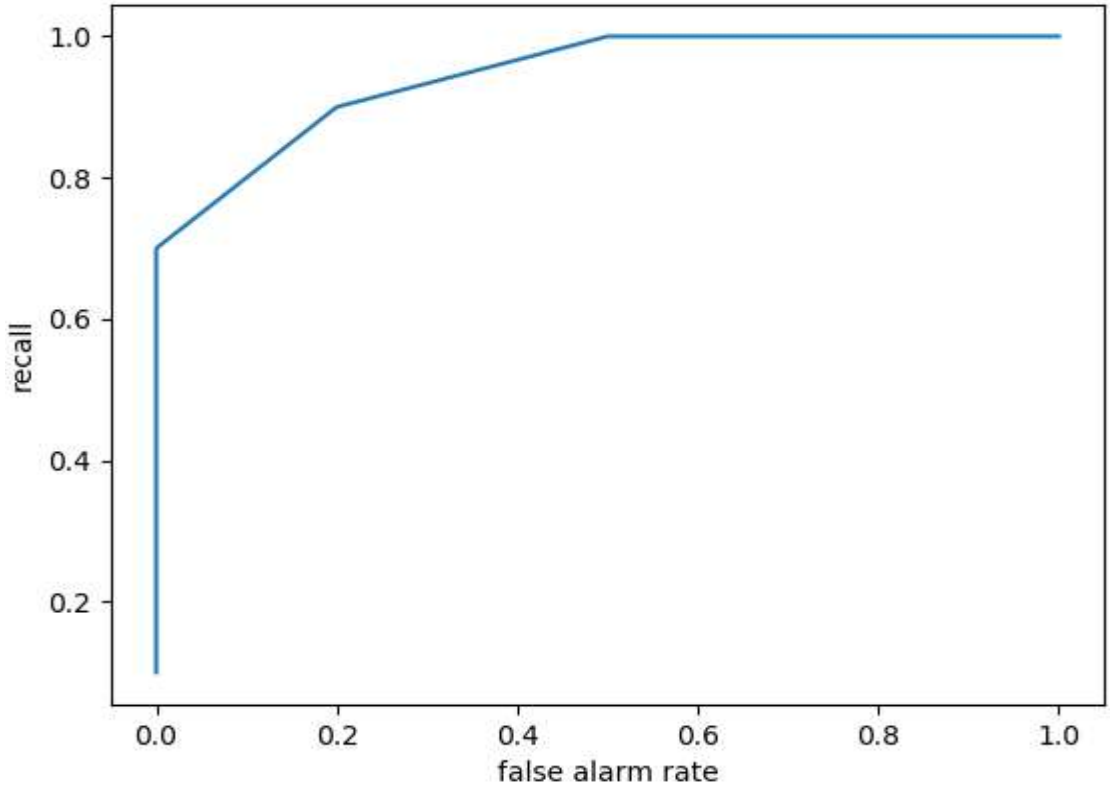
- Diberikan data prediksi, misalkan:

n=165	Prediksi = -1	Prediksi = +1	
Aktual = -1	$TN = 50$	$FP = 10$	60
Aktual = +1	$FN = 5$	$TP = 100$	105
	55	110	

1. **Akurasi**: (*TN*+*TP*)/jumlah data = (*50*+*100*)/*165*.
2. ***Miss classification rate*/error rate**: (*FP*+*FN*)/jumlah data = (*10*+*5*)/*165*.
3. ***Recall*** atau ***sensitivity*** atau ***true positive rate***: *TP*/aktual "+1"= 100/*105*.
4. **Presisi**: *TP*/prediksi "+1" = *100*/*110*.
5. ***False positive rate*** atau ***false alarm rate***: *FP*/aktual "-1" = 10/*60*.
6. ***Specificity***: *TN*/aktual "-1" = 50/*60*.

Kurva ROC (*Receiver Operating Characteristic*)

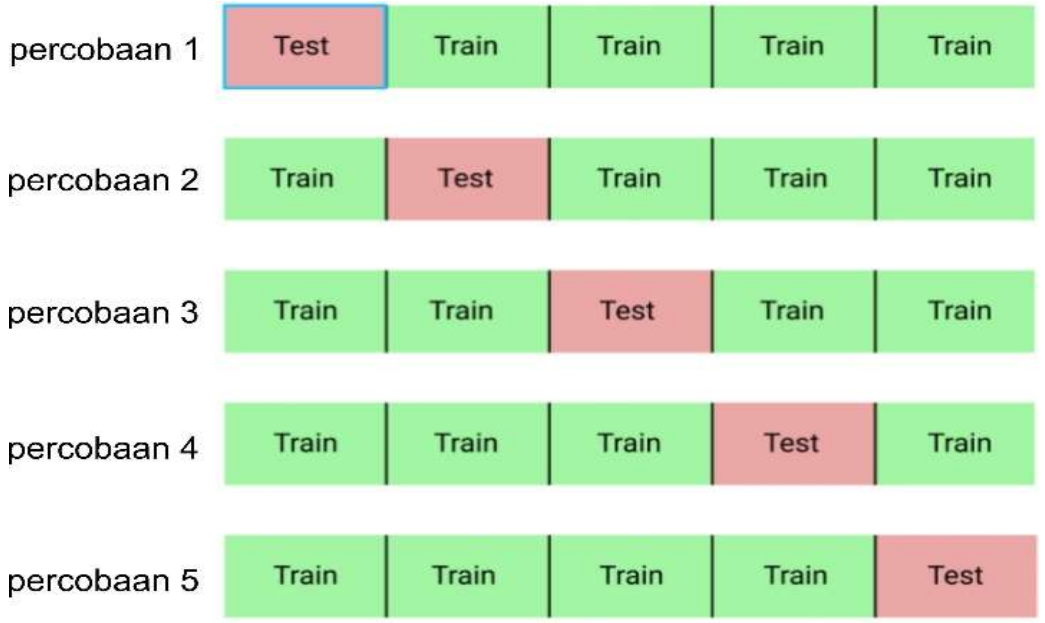
- Ketika kita menaikkan *recall*, konsekuensinya adalah menaikkan *false alarm*.
- Kurva ROC mengakum trade-off antar kedua hal tersebut, yakni dengan menghitung luas area di bawah kurva.



Cross Validation

Cross Validation

- Membagi menjadi ***k-folds***
- 1 fold menjadi data testing, selebihnya menjadi data training
- Kemudian fold data testing diganti dengan fold yang lain, begitu seterusnya seperti ilustrasi berikut, untuk 5-folds.



- Model ditraining di semua percobaan untuk selanjutnya divalidasi

Metode Ensemble

Metode Ensemble

- **Pengertian**: menggabungkan beberapa model agar didapatkan performa yang lebih bagus
- Misal diberikan 5 model:
  - ✓ Lakukan prediksi data dengan 5 model tersebut
  - ✓ Lihat hasil prediksi dari kelima model tersebut
  - ✓ Hasil prediksi akhir dapat diambil dari: majority vote, dirata-rata atau dibobot.

End..