Analitika Data I

Classification

Putra Pandu Adikara

Fakultas Ilmu Komputer
Universitas Brawijaya

Sumber:

Data Mining: Concepts and Techniques (Han, Kamber, and Pei)

Introduction to Data Mining (Tan, Steinbach, Karpatne, Kumar)

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: Data latih (pengamatan, pengukuran, dll) disertai dengan label yang menunjukkan kelas/label pengamatan
  - Data baru diklasifikasikan berdasarkan data latih

- Unsupervised learning (clustering)
  - Label kelas data latih tidak diketahui
  - Diberikan satu set pengukuran, pengamatan, dll dengan tujuan membangun keberadaan kelas atau klaster dalam data

Prediction Problems: Classification vs. Numeric Prediction

- Classification
  - memprediksi label kelas kategoris (diskrit atau nominal)
  - mengklasifikasikan data (membangun model) berdasarkan data latih dan nilai-nilai (label kelas) dalam atribut klasifikasi dan menggunakannya dalam mengklasifikasikan data baru
- Numeric Prediction
  - model fungsi bernilai berkelanjutan, yaitu, memprediksi nilai yang tidak diketahui (*unknown*) atau hilang (*missing value*)

Konsep dan Definisi Klasifikasi

- **Classification** adalah tugas untuk mempelajari fungsi target *f* dengan **tuple (*x*,*y*) pada data training** yang **memetakan** tiap himpunan **atribut *x*** ke salah satu dari **label kelas *y*** yang telah ditentukan sebelumnya
    - x: attribute, predictor, independent variable, input
    - y: class, response, dependent variable, output
- **Classification** adalah tugas menetapkan objek ke salah satu dari beberapa kategori yang sudah ditentukan
    - Task: Learn a model that maps each attribute set x into one of the predefined class labels y
- Klasifikasi adalah suatu bentuk analisis data yang mengekstraksi model, mendeskripsikan kelas data yang penting
- Model tersebut dikenal sebagai *classifier*, memprediksi label kelas kategorik (diskrit, tidak terurut).

Aplikasi dari Klasifikasi

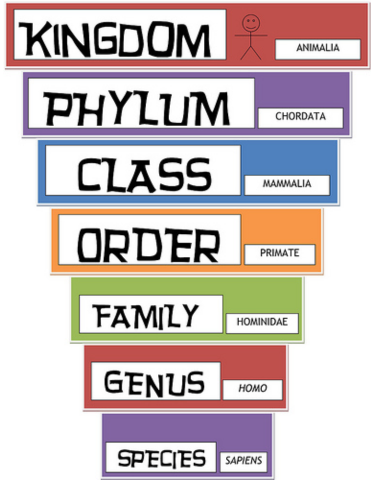
- Aplikasi khas
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is
  - Manufacturing
  - Target marketing
  - Performance prediction

Examples of Classification Task

Task	Attribute set, **x**	Class label, y
Categorizing email messages	Features extracted from email message header and content	spam or non-spam
Identifying tumor cells	Features extracted from x-rays or MRI scans	malignant or benign cells
Cataloging galaxies	Features extracted from telescope images	Elliptical, spiral, or irregular-shaped galaxies

Classification vs Prediction

- Classification
  - Membangun Model Classifier
  - Memprediksi label kelas (discrete/categorical, unordered)
  - Tidak selalu menggunakan data *time series*



- Prediction
  - Membangun Model Predictor
  - Memprediksi fungsi bernilai kontinu atau nilai berurutan
  - Biasanya melibatkan data time series

How does classification work?

- Klasifikasi data adalah proses dua tahap, terdiri dari:
  1. **Tahap pembelajaran/pelatihan (*learning step*):** tahap model klasifikasi dibangun
  2. **Tahap pengujian/klasifikasi (*classification step*):** tahap model digunakan untuk memprediksi label kelas untuk data yang diberikan

Classification—A Two-Step Process

1. **Model construction/training:** Menggambarkan kelas-kelas yang telah ditentukan
   - Setiap tuple / sampel diasumsikan milik kelas yang telah ditentukan, yang ditentukan oleh atribut label kelas.
   - Set tuples yang digunakan untuk konstruksi model adalah **data latih (*training set*).**
   - Model ini direpresentasikan sebagai aturan klasifikasi, pohon keputusan, atau rumus matematika.
2. **Model usage/test**: untuk mengklasifikasikan objek masa depan atau tidak dikenal
   - Perkiraan keakuratan model
   - **Label data uji/*test set*** yang diketahui, dibandingkan dengan hasil klasifikasi dari model.
     - Tingkat akurasi adalah persentase sampel test set yang diklasifikasikan dengan benar oleh model.
     - Data uji tidak tergantung/ada pada data latih (jika tidak, maka akan terjadi overfitting)
   - Jika keakuratannya dapat diterima, gunakan model untuk mengklasifikasikan data baru.

- Catatan: Jika data uji digunakan untuk memilih model, itu disebut **validation (test) set**

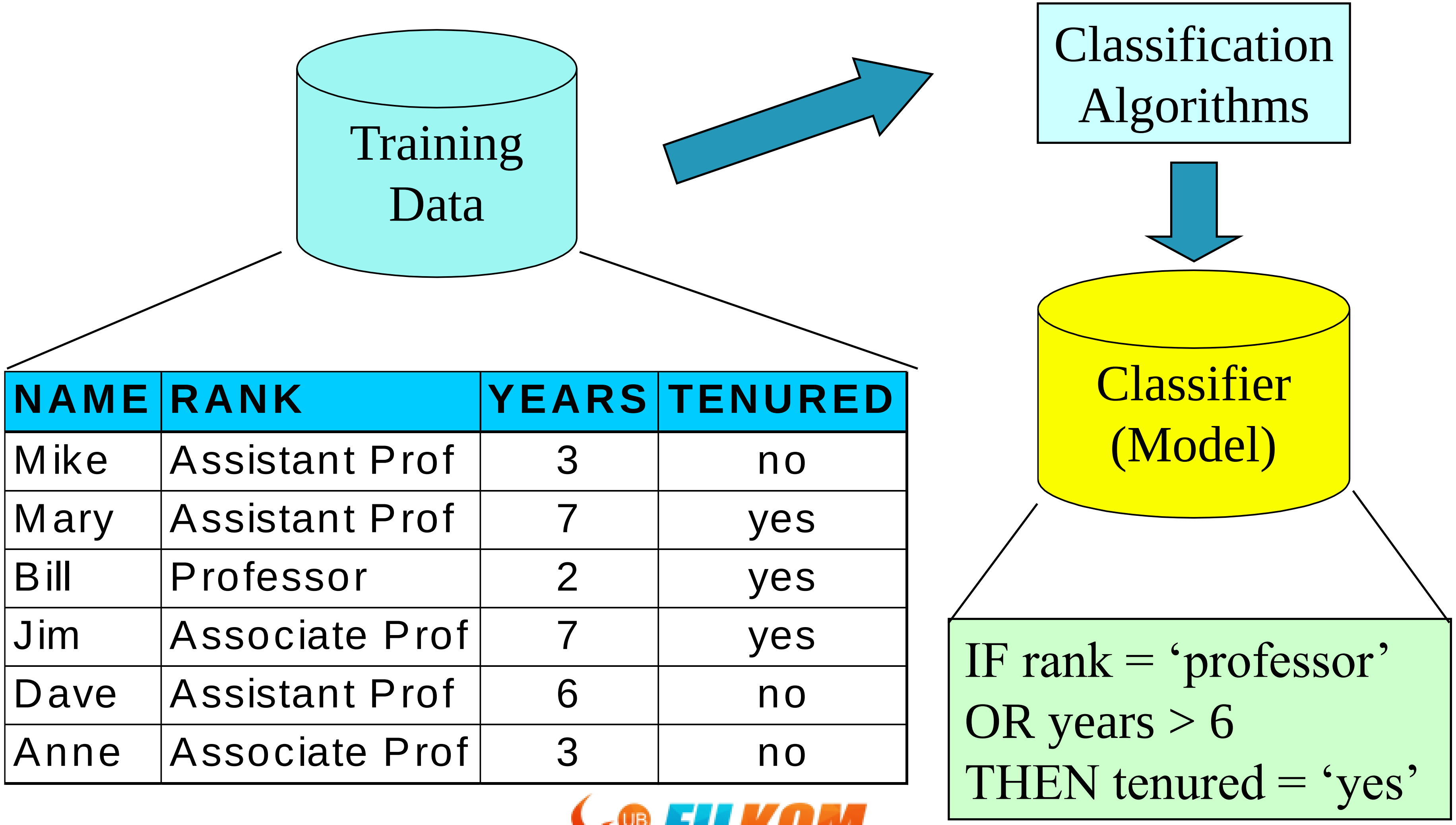
Process (1): Pelatihan (*Training*)

- Klasifikasi dibangun untuk menggambarkan sekumpulan kelas data atau konsep yang telah ditentukan.
- Langkah pembelajaran (atau fase pelatihan) → algoritme klasifikasi membangun classifier dengan menganalisis atau “belajar dari” suatu training set yang terdiri dari *tuple* basis data dan label kelas terkait.
- Setiap tuple diasumsikan milik kelas yang telah ditentukan sebagaimana ditentukan oleh atribut basis data lain yang disebut **class label attribute** (bernilai diskrit dan tidak berurutan)

Process (1): Pelatihan (*Training*)

- Karena label kelas dari masing-masing *tuple* pelatihan disediakan, langkah ini juga dikenal sebagai **pembelajaran terarah/terawasi/*supervised learning*** (mis., pembelajaran dari classifier "diawasi" karena *dibimbing* ke kelas mana masing-masing *tuple* pelatihan dimasukkan).
- Berbeda dengan **pembelajaran tanpa pengawasan/ *unsupervised learning*** (atau pengklasteran/*clustering*), yang mana label kelas dari setiap *tuple* pelatihan tidak diketahui dan jumlah atau himpunan kelas yang akan dipelajari bisa jadi tidak diketahui sebelumnya.

Process (1): Model Construction



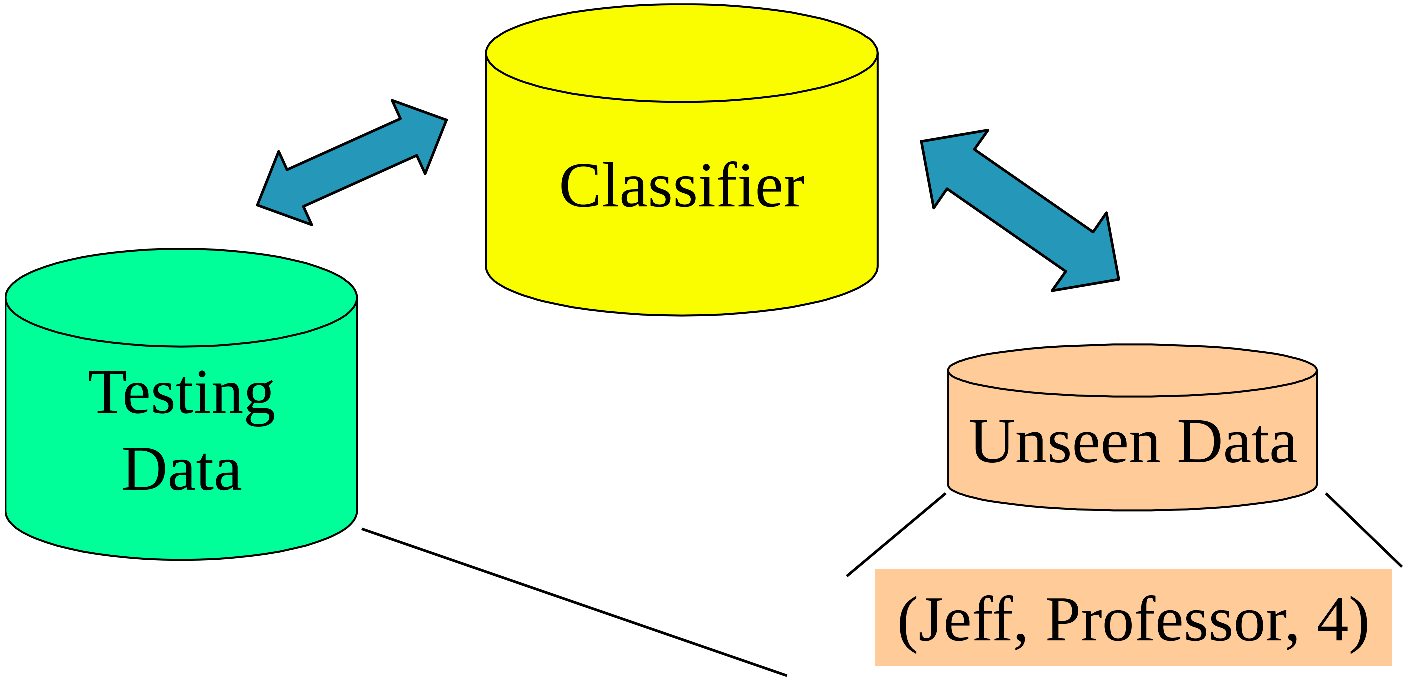
NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

FILKOM

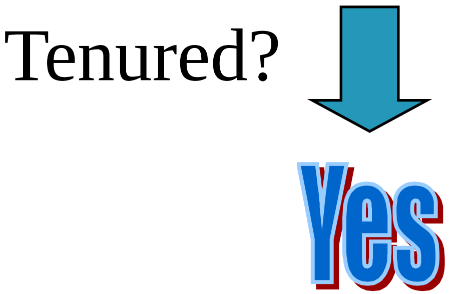
Process (2): Klasifikasi/Pengujian (*Classification/testing*)

- Pada tahap kedua, **model** (bentuk dari langkah pembelajaran) digunakan untuk klasifikasi.
- Himpunan uji/data uji digunakan, terdiri dari **test tuples/tuple uji** dan label kelas yang terkait.
- Tupel uji **tidak tergantung/terpisah** pada tuple pelatihan, artinya tuple yang tidak digunakan untuk membuat klasifikasi.
- **Akurasi** dari classifier pada data uji yang diberikan adalah persentase tuple data uji yang diklasifikasikan benar oleh *classifier*.
- Tujuan utama dari algoritme pembelajaran adalah untuk membangun model dengan kemampuan generalisasi yang baik; yaitu, model yang secara akurat memprediksi label kelas dari catatan yang sebelumnya tidak dikenal

Process (2): Using the Model in Prediction

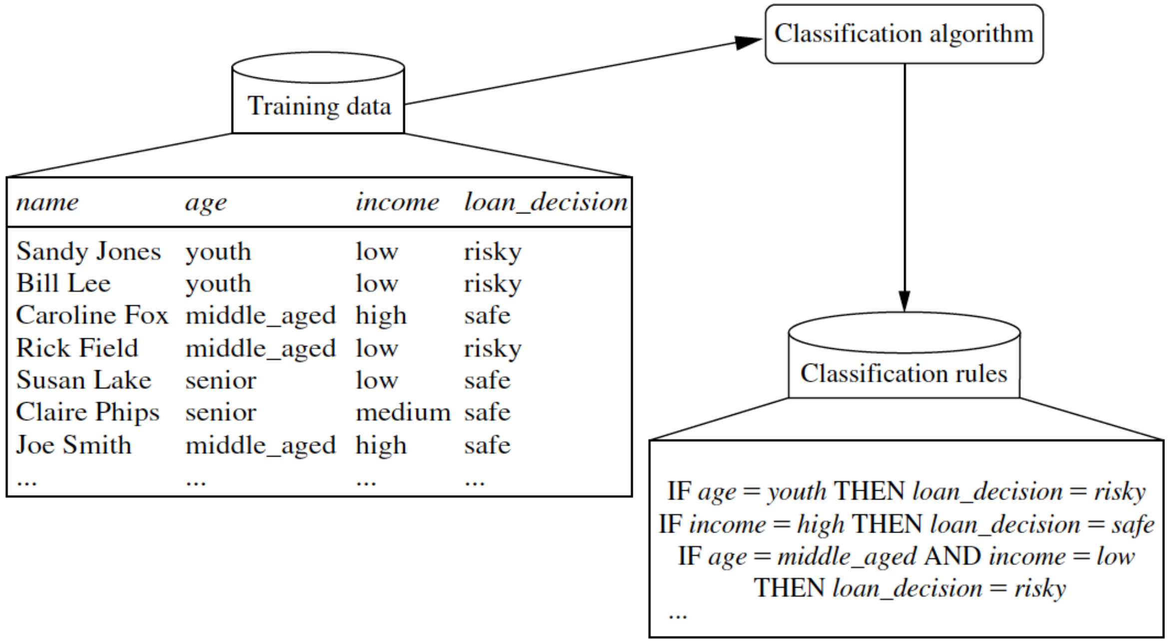


NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

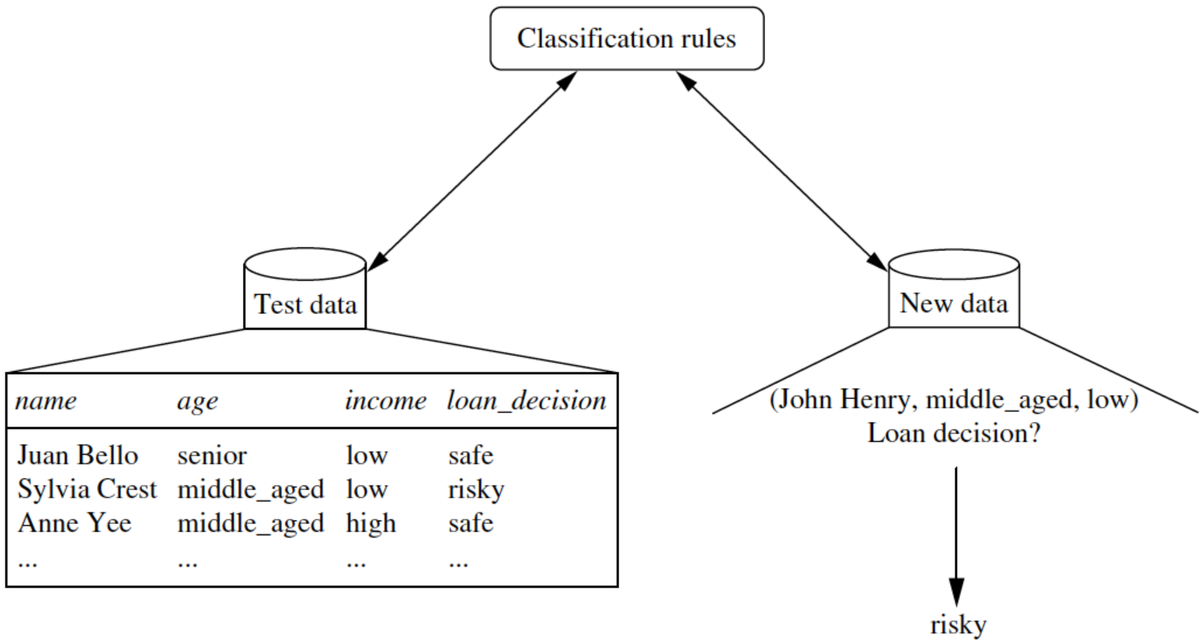


Classification

- Learning



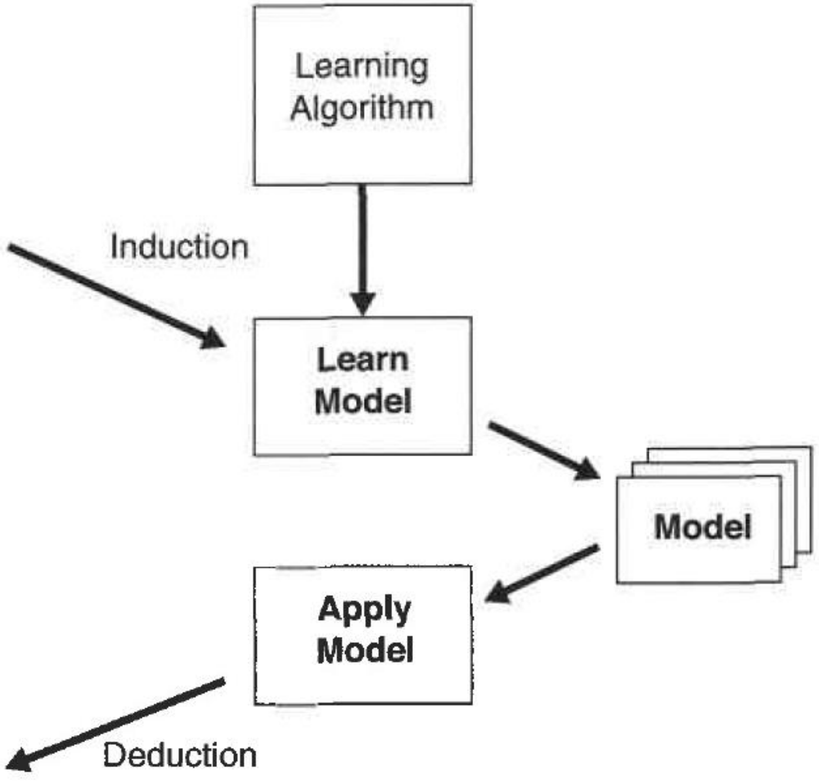
- Testing



Classification

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

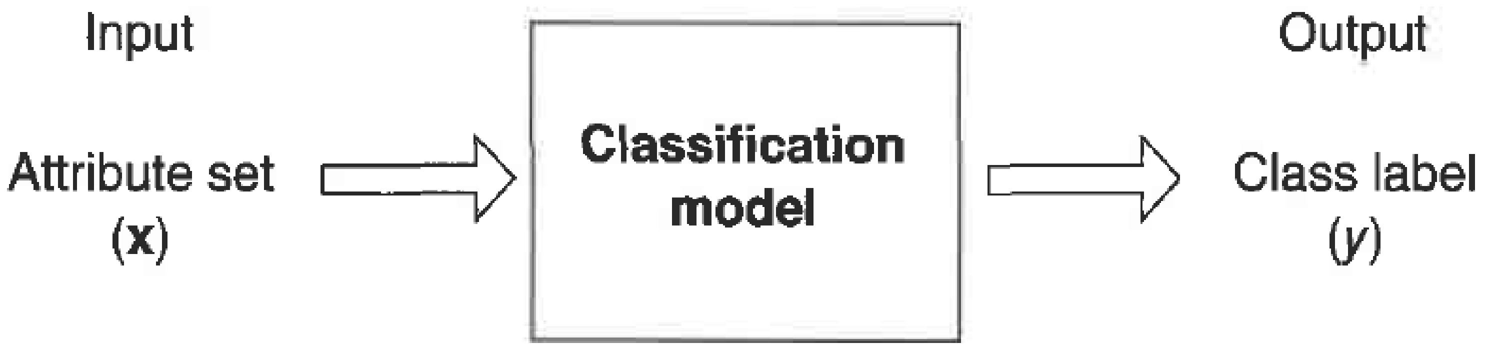


Test Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Classification

- Klasifikasi adalah tugas memetakan suatu himpunan atribut input *x* ke label kelas *y*



Penggunaan Model Klasifikasi

- **Pemodelan Deskriptif (*Descriptive Modelling*)** → Sebuah model klasifikasi dapat berfungsi sebagai alat penjelas untuk membedakan antara objek dari kelas yang berbeda.
- **Pemodelan Prediktif (*Predictive Modelling*)** → Model klasifikasi juga dapat digunakan untuk memprediksi label kelas dari data yang tidak diketahui

Teknik Klasifikasi

- Teknik klasifikasi (atau *classifier*) adalah pendekatan sistematik untuk membangun model klasifikasi dari sebuah data set input
- Base Classifiers
  - Decision Tree based Methods
  - Rule-based Methods
  - Nearest-neighbor
  - Naïve Bayes and Bayesian Belief Networks
  - Support Vector Machines
  - Neural Networks, Deep Neural Nets
- Ensemble Classifiers
  - Boosting, Bagging, Random Forests

Jenis Classification

- Binary classification
  - Proses atau tugas klasifikasi yang mengklasifikasikan data ke dalam salah satu dari dua kelas
  - Dua kelas ini bersifat “*anti*” satu dengan yang lain
  - Contoh: sehat vs tidak sehat, mati vs hidup
  - Cara mudah mengetahui: bisa kah ditambahkan kelas di antaranya? Tidak akan ada setengah sehat/tidak sehat atau setengah hidup/mati
- Multi-class classification
  - Tugas klasifikasi yang mengklasifikasikan data ke dalam salah satu dari banyak kelas
  - Contoh: sangat positif vs positif vs negatif vs sangat negatif, klasifikasi tanaman, klasifikasi jenis penyakit, optical recognition character, dll

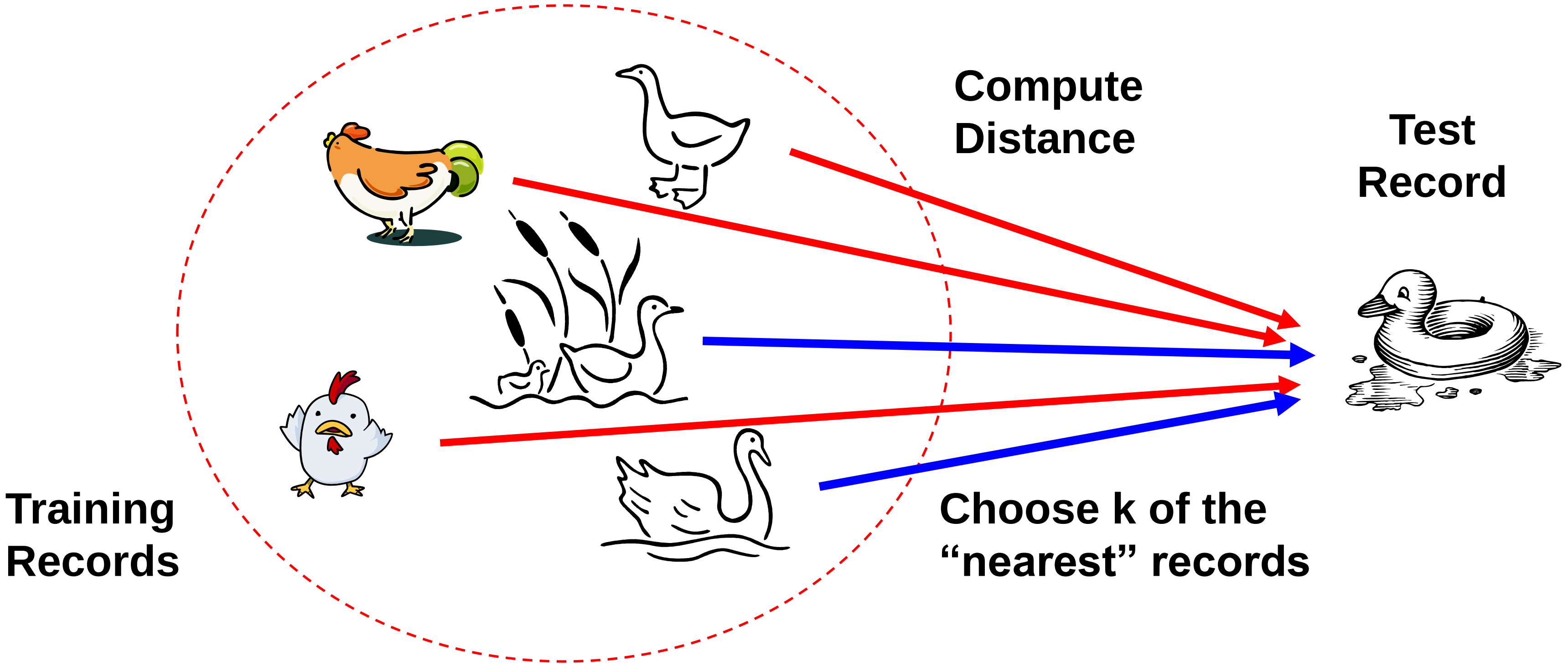
Jenis Classification

- Multilabel Classification
  - Tugas klasifikasi yang mengklasifikasikan data ke lebih dari satu kategori dari banyak kelas yang tersedia
  - Contoh: kategori berita, 1 berita bisa masuk ke beberapa kategori (ekonomi dan travel), dll.

Algoritma K-Nearest Neighbor

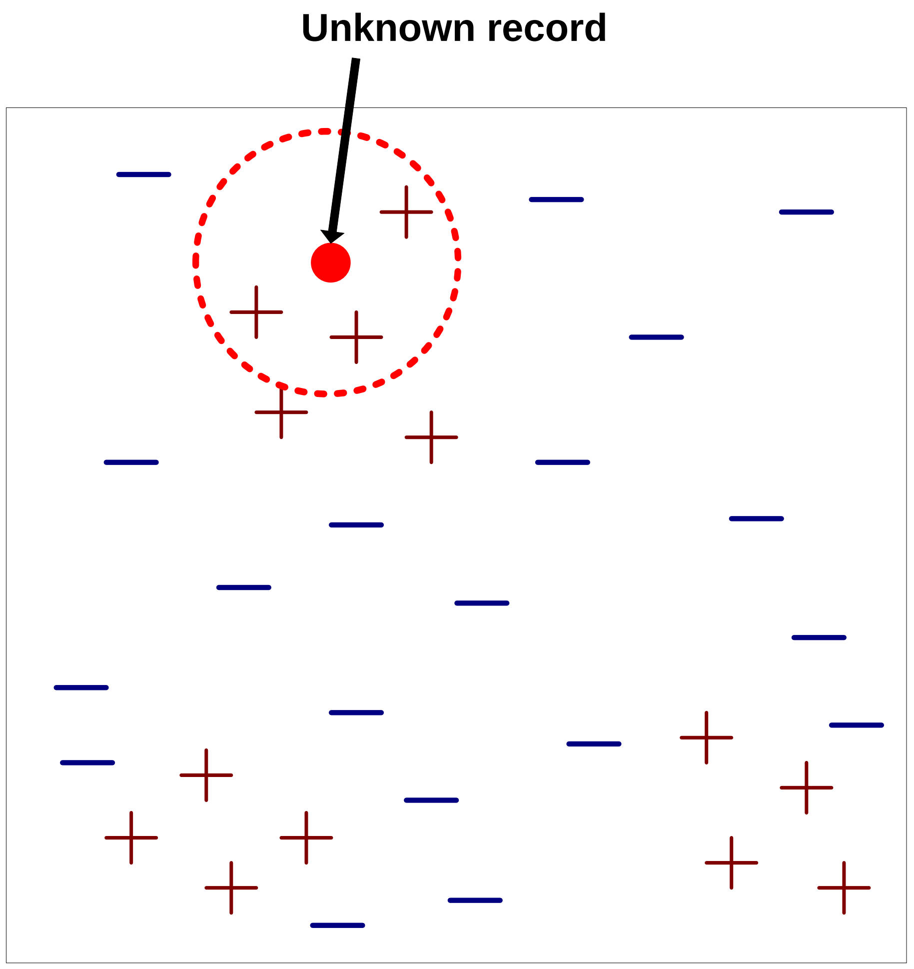
Nearest Neighbor Classifiers

- Ide dasar:
  - Jika berjalan seperti bebek, bersuara seperti bebek, maka itu mungkin bebek.



FILKOM

Nearest-Neighbor Classifiers



- Membutuhkan hal-hal berikut:
  - Satu set **data berlabel/berkelas**
  - **Metrik kedekatan** untuk menghitung jarak / kesamaan antara sepasang data
  - mis., Euclidean distance
  - **Nilai k,** jumlah tetangga terdekat untuk diambil
  - Metode untuk menggunakan label kelas K tetangga terdekat untuk menentukan label kelas data yang tidak diketahui (misalnya, dengan mengambil suara mayoritas)

How to Determine the class label of a Test Sample?

- Ambil suara mayoritas label kelas di antara tetangga terdekat k
- Bobot voting sesuai dengan jarak
  - weight factor, $w = 1/d^2$

Choice of proximity measure matters

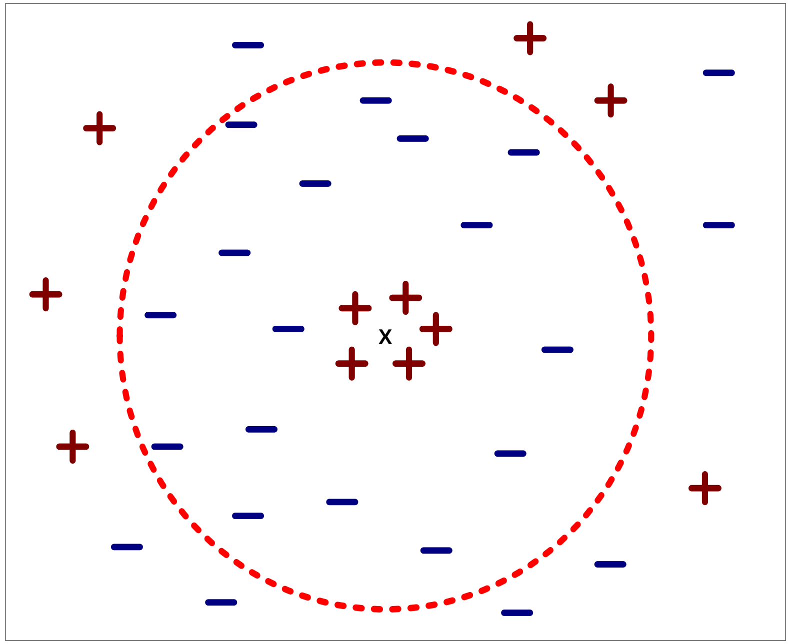
- Untuk dokumen, cosine lebih baik daripada korelasi atau Euclidean.

1 1 1 1 1 1 1 1 1 1 1 0	vs	0 0 0 0 0 0 0 0 0 0 0 1
0 1 1 1 1 1 1 1 1 1 1 1		1 0 0 0 0 0 0 0 0 0 0 0

Jarak Euclidean = 1,4142 untuk kedua pasangan, tetapi ukuran kesamaan cosine memiliki nilai yang berbeda untuk pasangan ini.

Nearest Neighbor Classification…

- Memilih nilai k:
  - Jika k terlalu kecil, sensitif terhadap data noise
  - Jika k terlalu besar, ketetanggaan dapat mencakup data dari kelas lain.

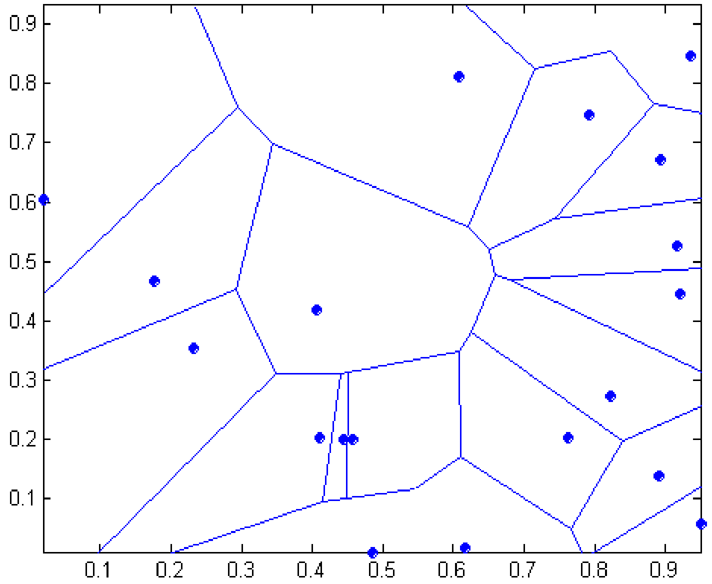


Nearest-neighbor classifiers

- Nearest neighbor classifier adalah local classifiers

- Dapat menghasilkan batas keputusan (decision boundaries) dengan bentuk sembarang (arbitrary shapes).

1-nn decision boundary is a Voronoi Diagram

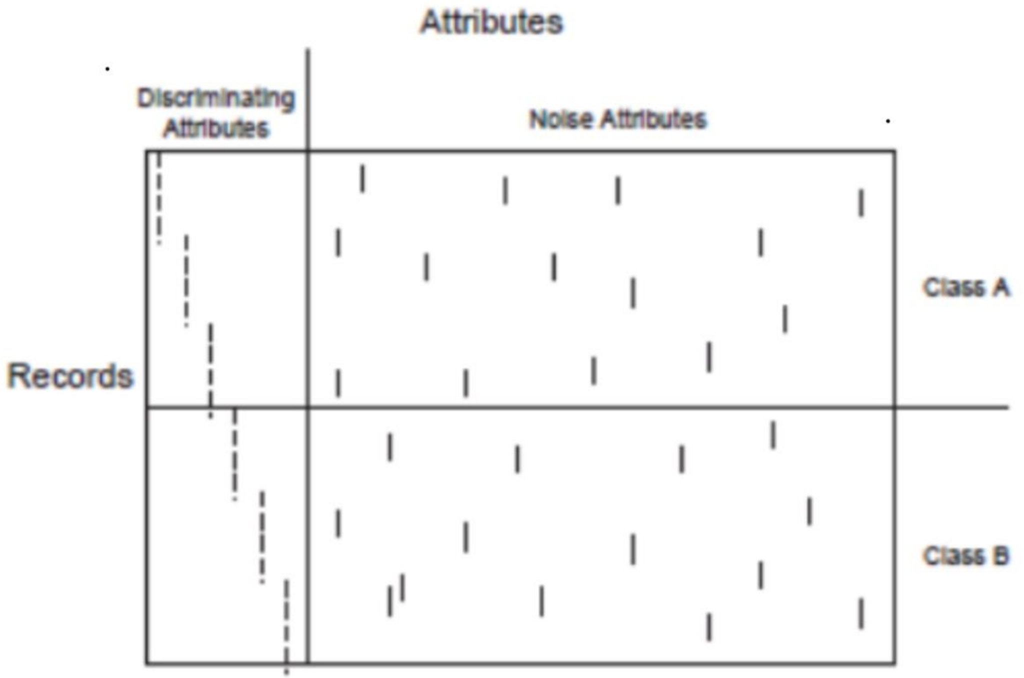


Nearest Neighbor Classification…

- **Bagaimana menangani nilai-nilai yang hilang dalam training dan test set?**
  - Perhitungan kedekatan biasanya membutuhkan kehadiran semua atribut.
  - Beberapa pendekatan menggunakan subset atribut yang ada dalam dua instance.
    - mungkin tidak menghasilkan hasil yang baik karena secara efektif menggunakan langkah-langkah kedekatan yang berbeda untuk setiap pasangan kasus.
    - Sehingga, kedekatan/proximities tidak bisa dibandingkan

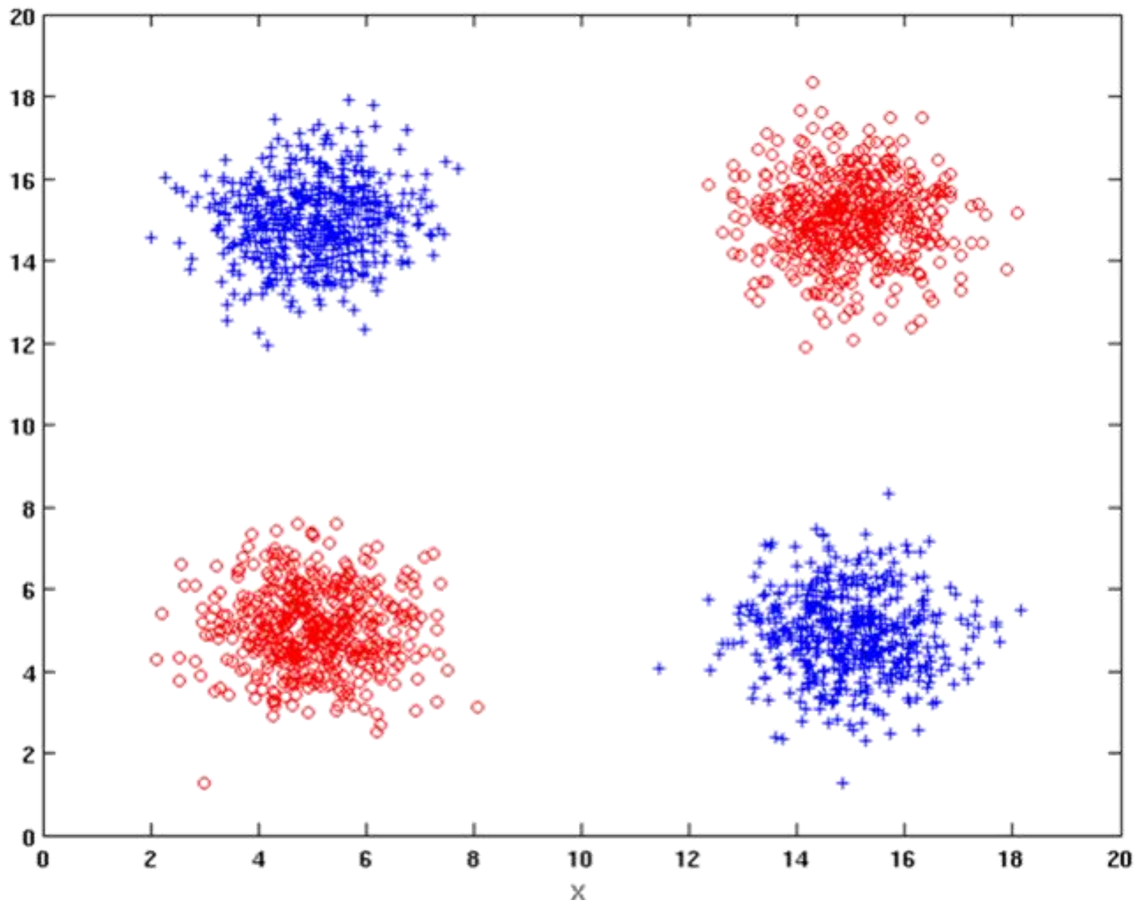
K-NN Classificiers…
Handling Irrelevant and Redundant Attributes

- Atribut yang tidak relevan menambahkan noise ke proximity measure
- Atribut redundan memberikan bias proximity measure terhadap atribut tertentu

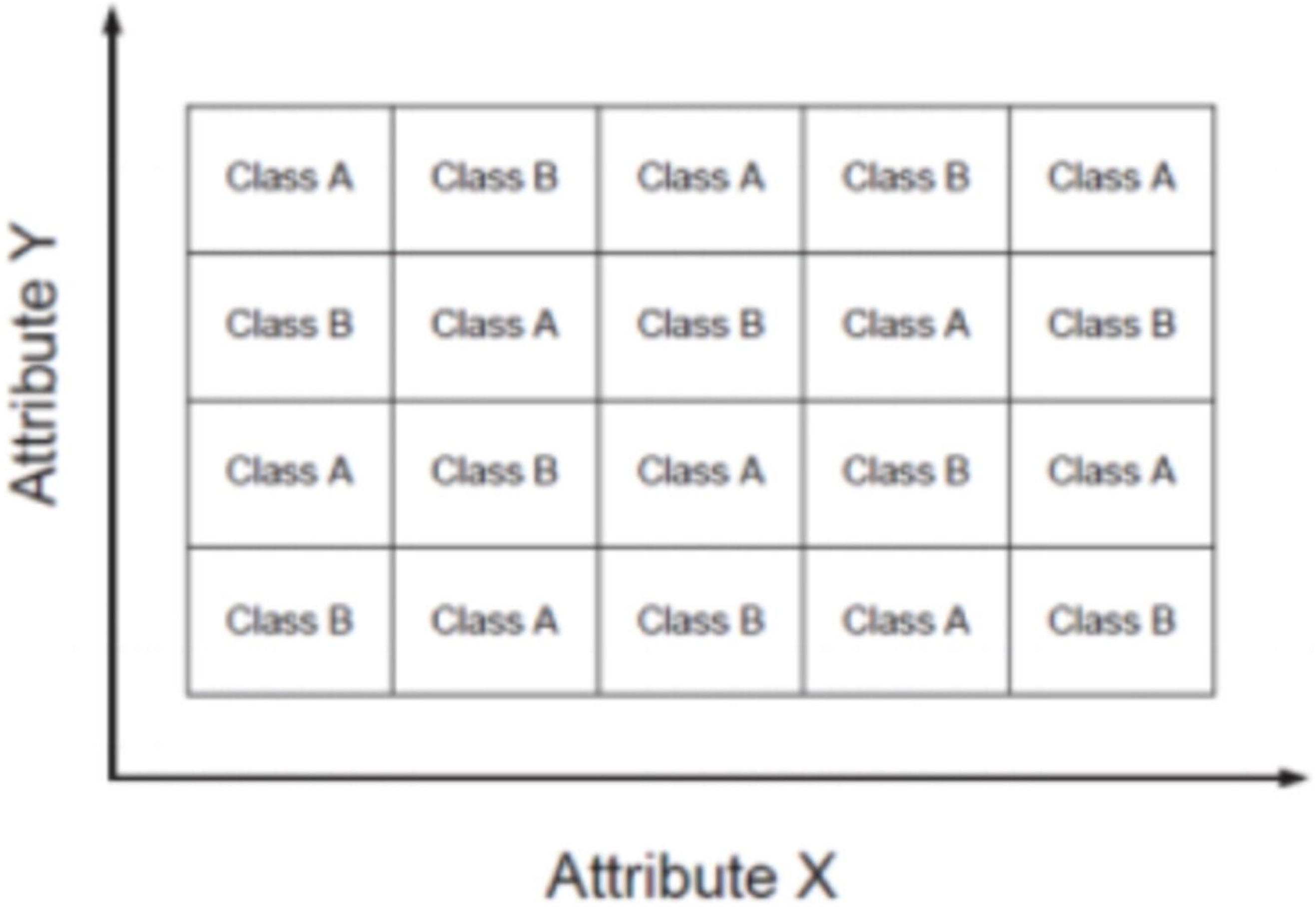


(a) Synthetic data set 1.

K-NN Classifiers: Handling attributes that are interacting



Handling attributes that are interacting



Improving KNN Efficiency

- Hindari keharusan menghitung jarak ke semua objek dalam set pelatihan
  - Multi-dimensional access methods (k-d trees)
  - Fast approximate similarity search
  - Locality Sensitive Hashing (LSH)
- Condensing
  - Menentukan satu set objek yang lebih kecil yang memberikan kinerja yang sama
- Editing
  - Menghapus objek untuk meningkatkan efisiensi

Lazy vs. Eager Learning

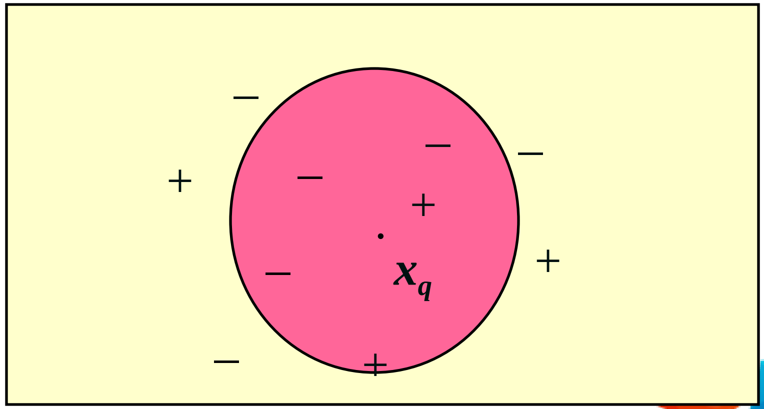
- Lazy vs. eager learning
  - **Lazy learning** (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - **Eager learning** (the above discussed methods): Given a set of training tuples, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form an implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

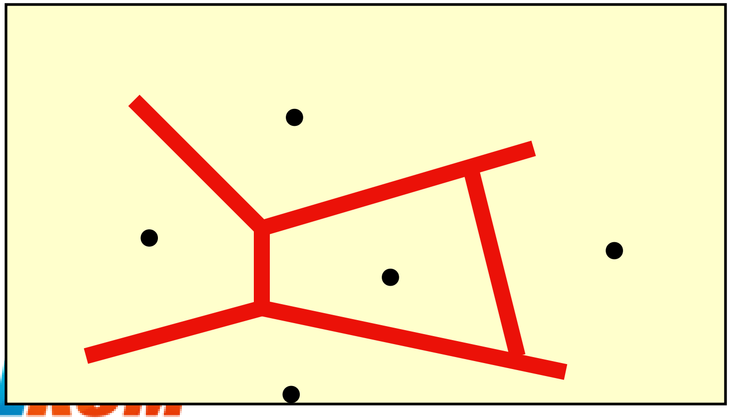
Lazy Learner: Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing (“lazy evaluation”) until a new instance must be classified
- Typical approaches
  - *k*-nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression
    - Constructs local approximation
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference

The k-Nearest Neighbor Algorithm

- All instances correspond to points in the n-D space
- The nearest neighbor are defined in terms of Euclidean distance, dist($\mathbf{X_1}$, $\mathbf{X_2}$)
- Target function could be discrete- or real- valued
- For discrete-valued, *k*-NN returns the most common value among the *k* training examples nearest to x_q
- Vonoroi diagram: the decision surface induced by 1-NN for a typical set of training examples





Discussion on the k-NN Algorithm

- *k*-NN for real-valued prediction for a given unknown tuple
  - Returns the mean values of the *k* nearest neighbors
- Distance-weighted nearest neighbor algorithm
  - Weight the contribution of each of the *k* neighbors according to their distance to the query x_q
    - Give greater weight to closer neighbors

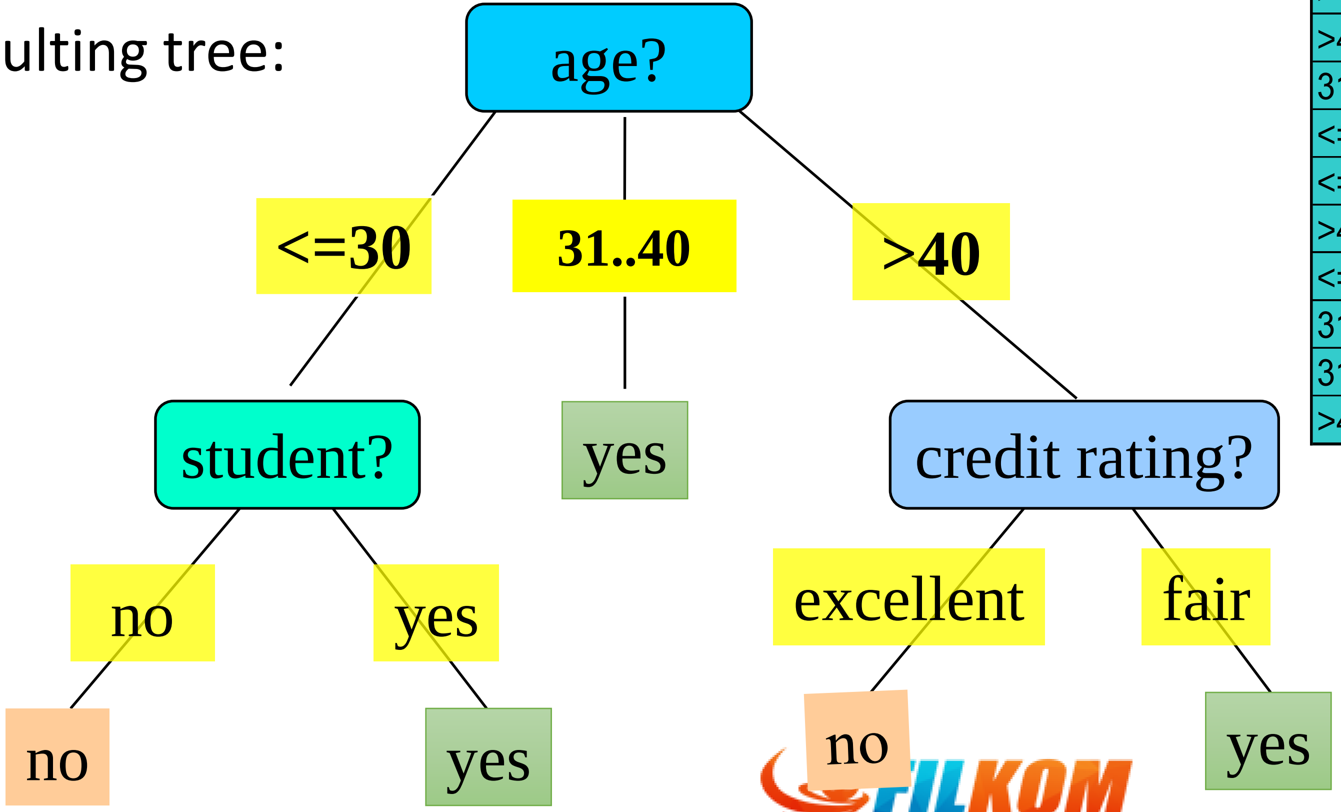
$$w \equiv \frac{1}{d(x_q, x_i)^2}$$

- Robust to noisy data by averaging *k*-nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes
  - To overcome it, axes stretch or elimination of the least relevant attributes

Algoritma Decision Tree

Decision Tree Induction: An Example

- Training data set: Buys_computer
- The data set follows an example of Quinlan's ID3 (Playing Tennis)
- Resulting tree:



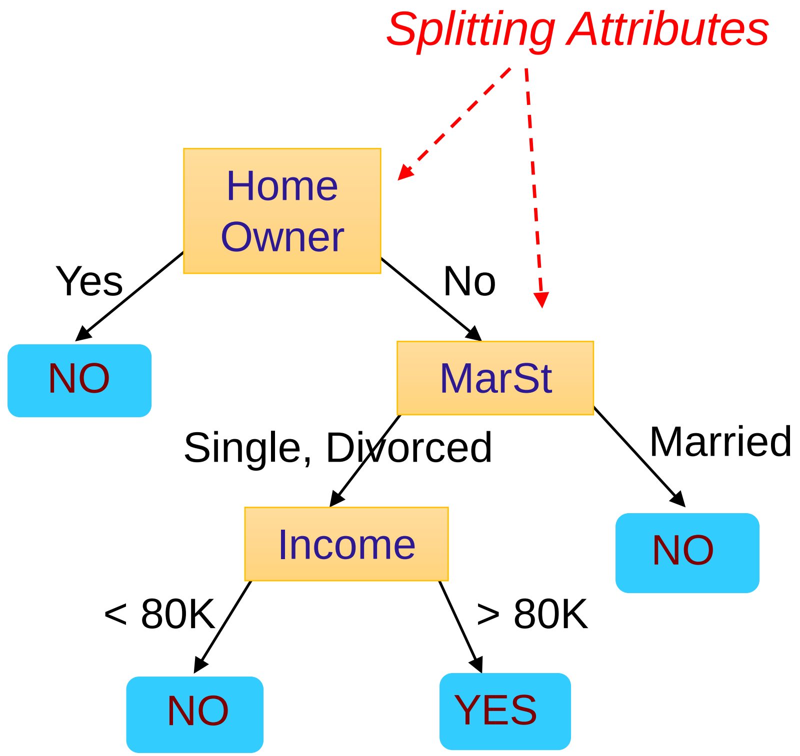
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent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

Example of a Decision Tree

categorical categorical continuous class

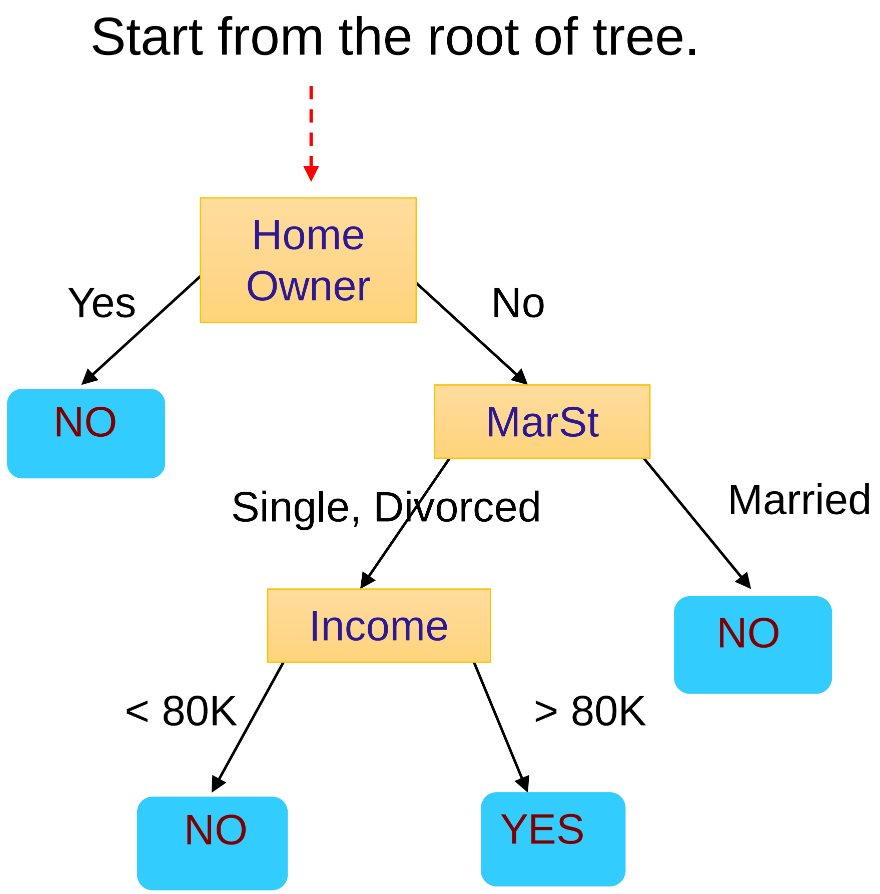
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



Model: Decision Tree

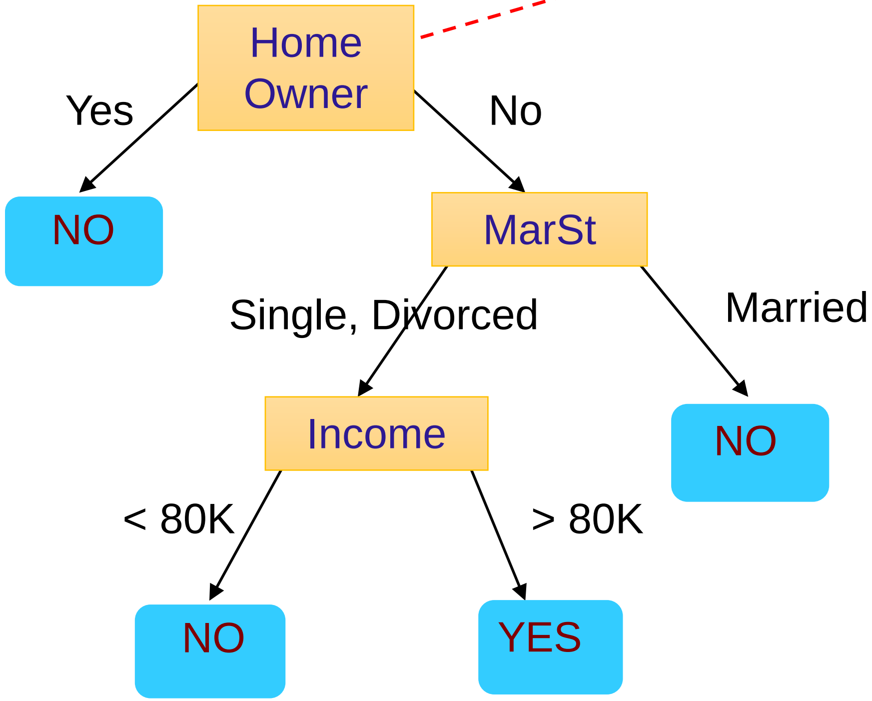
Apply Model to Test Data



Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Test Data

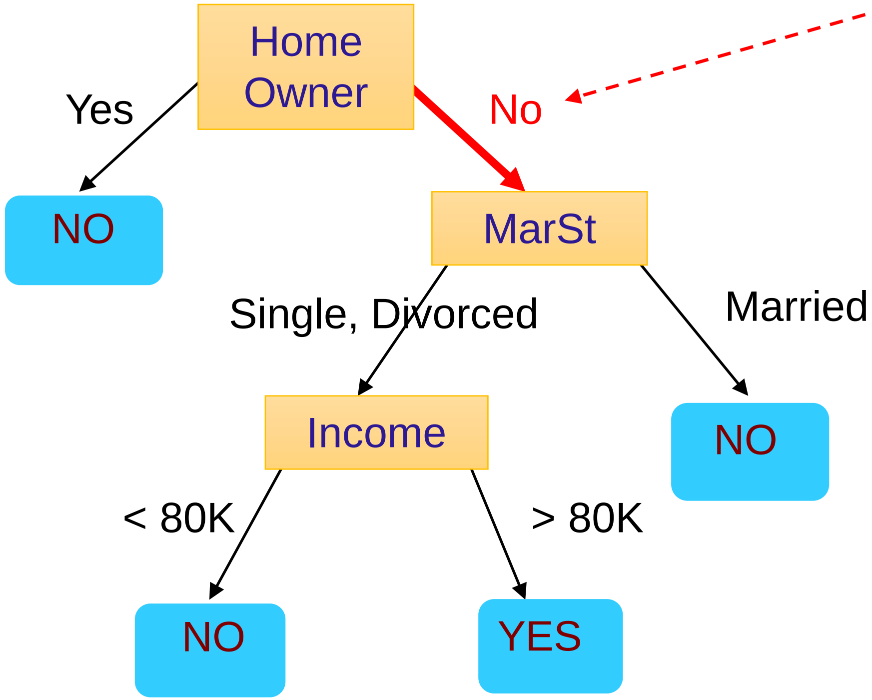
Apply Model to Test Data



Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Test Data

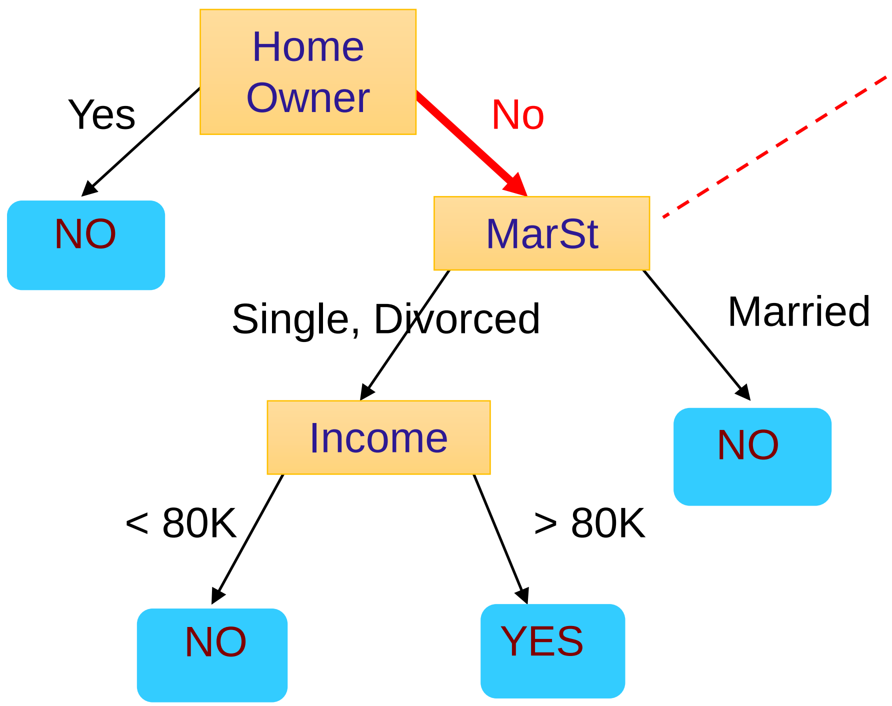
Apply Model to Test Data



Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Test Data

Apply Model to Test Data



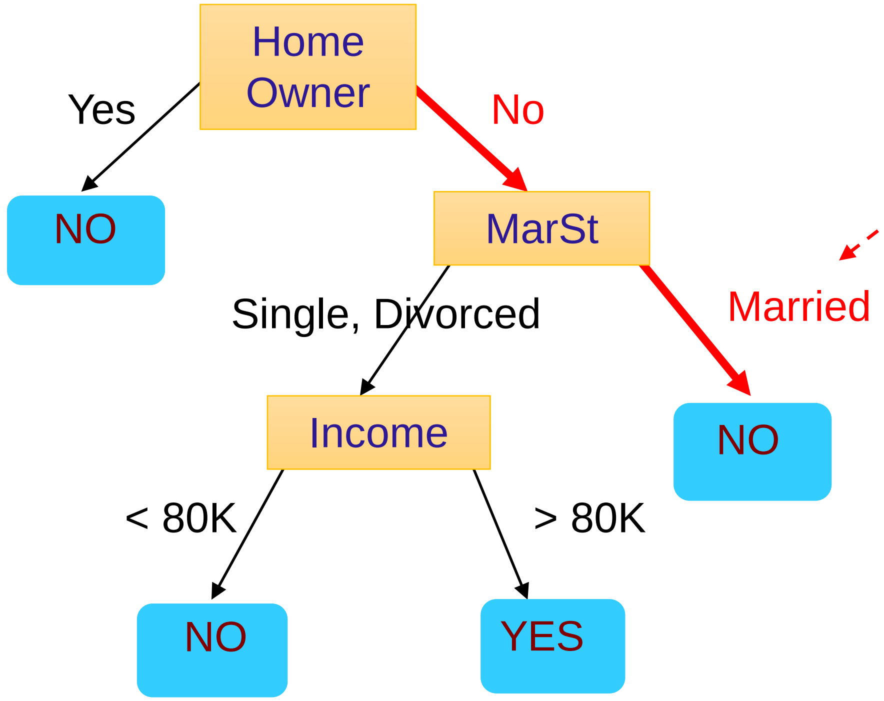
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Test Data

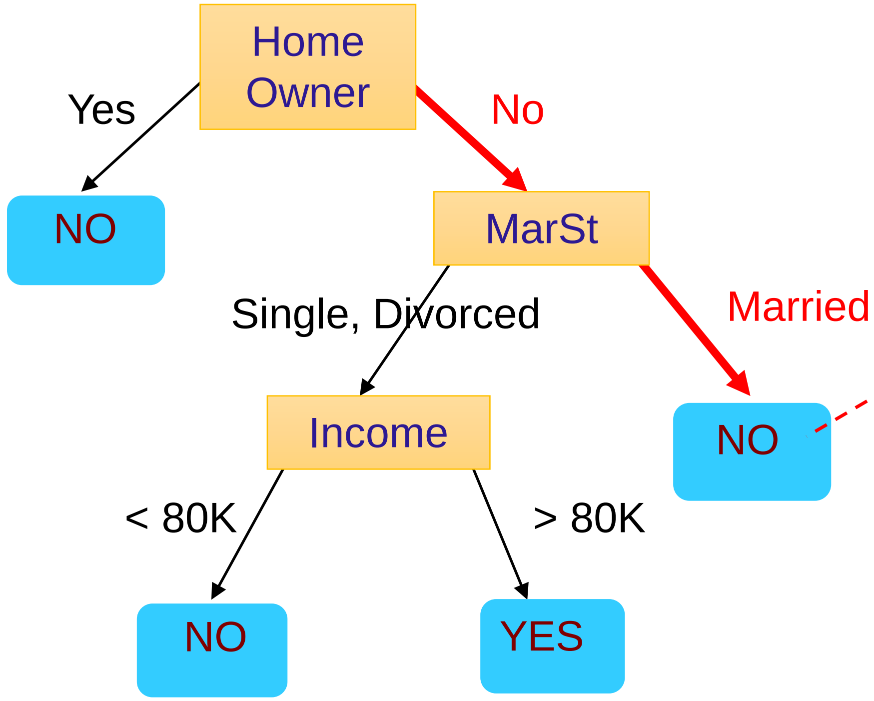
Apply Model to Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Test Data



Apply Model to Test Data



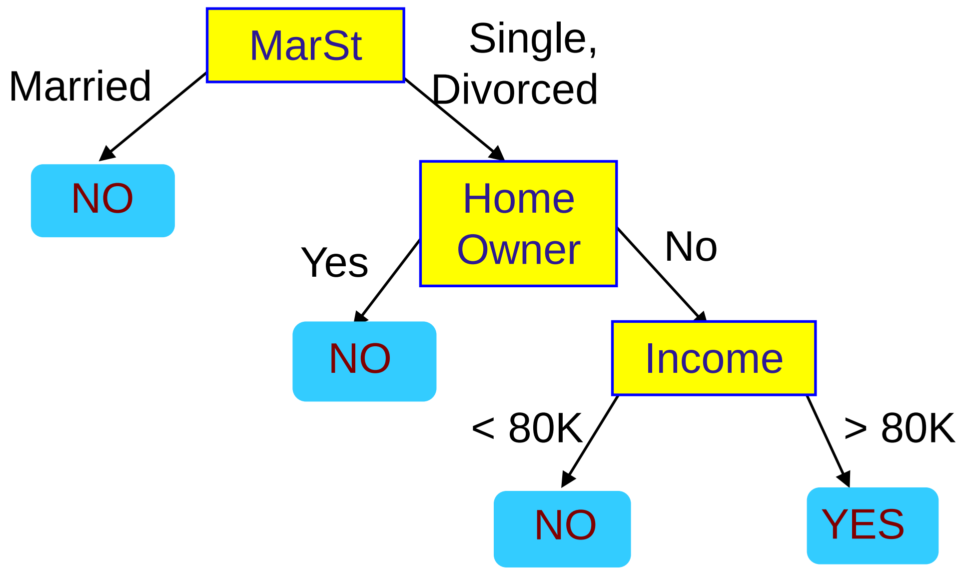
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Test Data

Assign Defaulted to “No”

Another Example of Decision Tree

ID	Home Owner (categorical)	Marital Status (categorical)	Annual Income (continuous)	Defaulted Borrower (class)
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

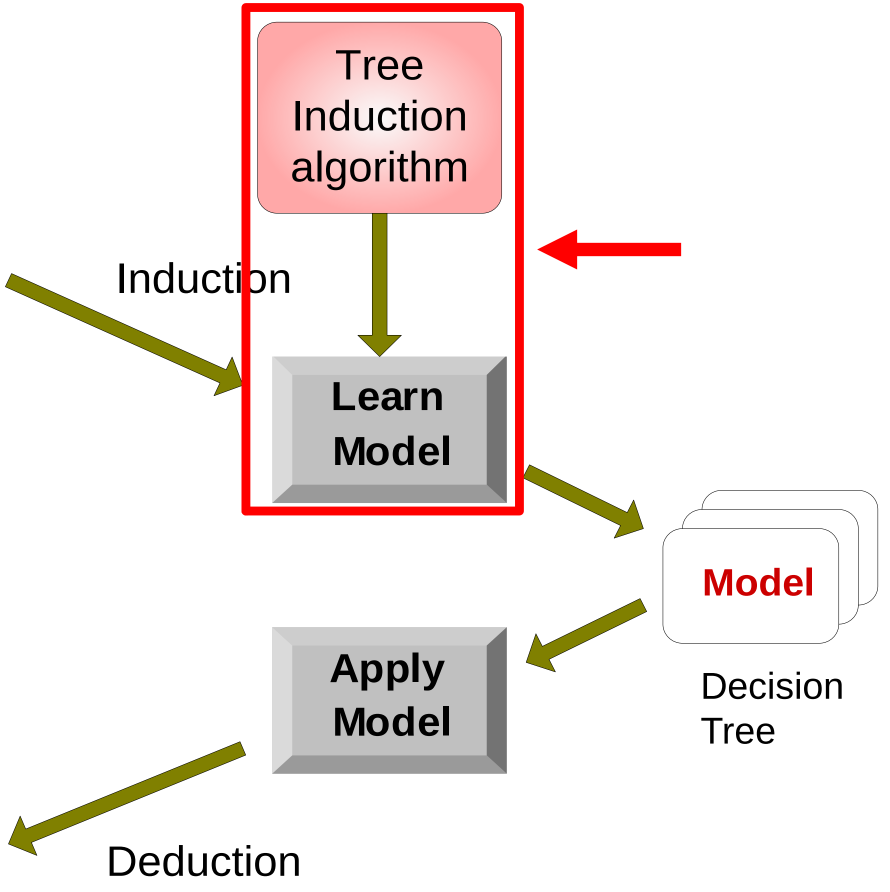


There could be more than one tree that fits the same data!

Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Decision Tree Induction

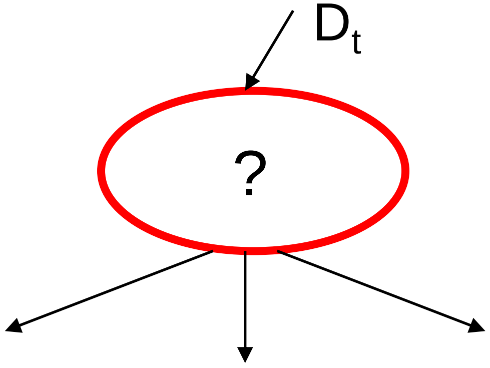
- Many Algorithms:
  - Hunt's Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ,SPRINT

General Structure of Hunt's Algorithm

- Let Dt be the set of training records that reach a node t

- General Procedure:
  - If Dt contains records that belong the same class yt, then t is a leaf node labeled as yt
  - If Dt contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Hunt’s Algorithm

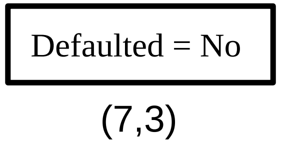
Defaulted = No

(7,3)

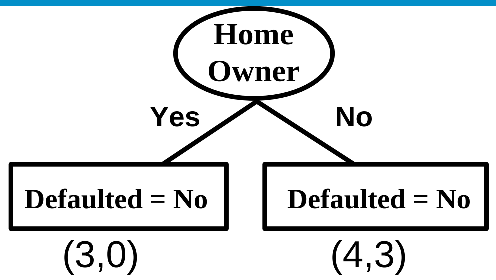
(a)

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Hunt’s Algorithm



(a)

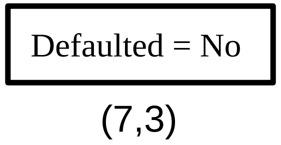


(b)

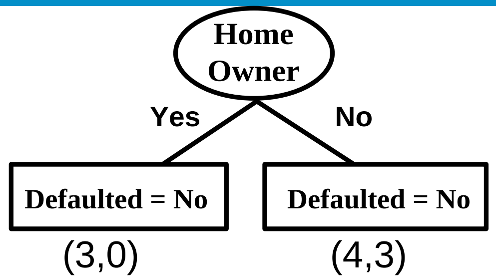
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

(d)

Hunt's Algorithm

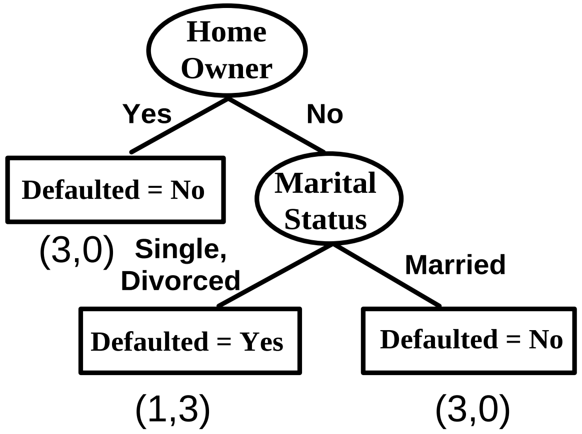


(a)



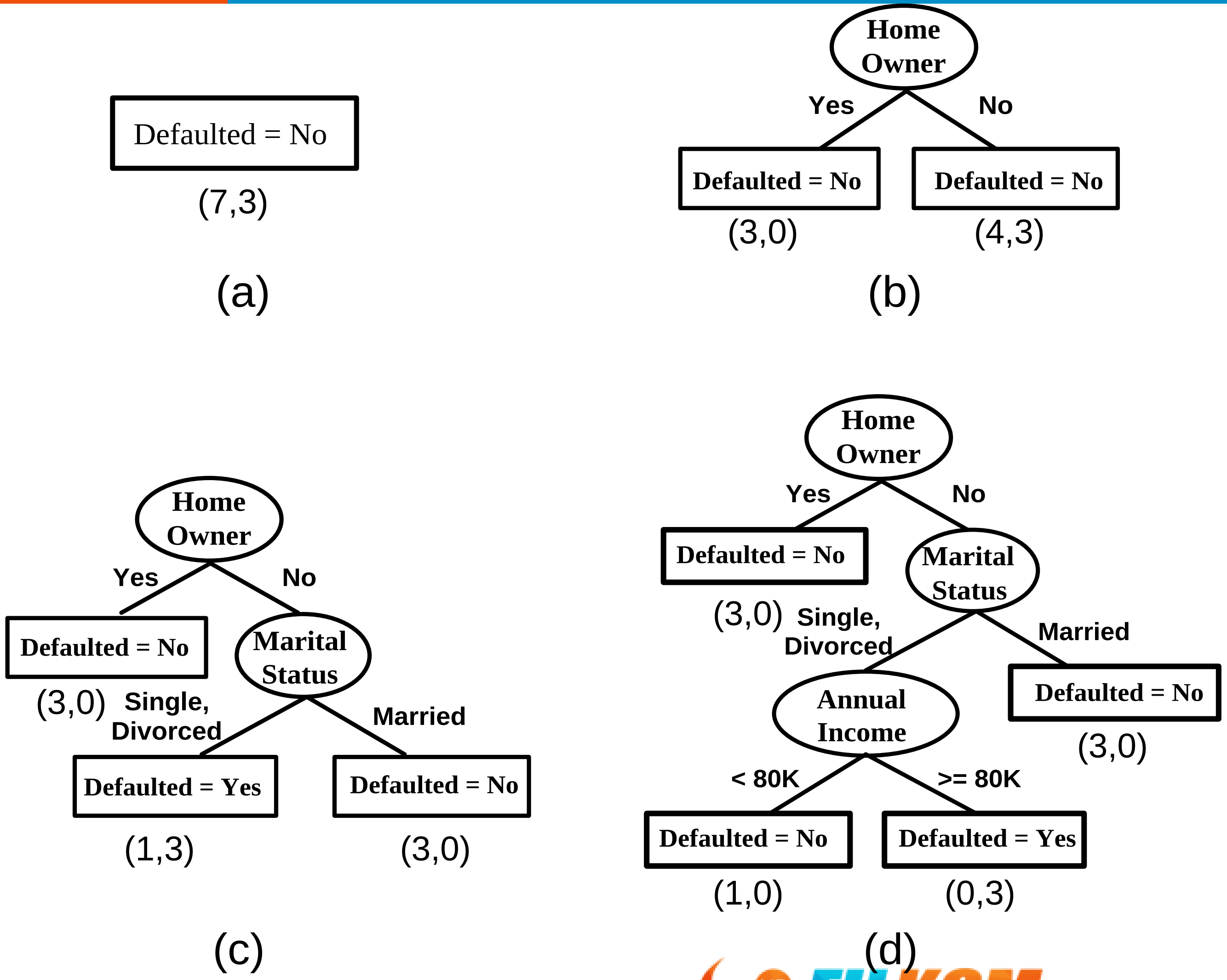
(b)

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



(c)

Hunt's Algorithm



ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Design Issues of Decision Tree Induction

- How should training records be split?
  - Method for expressing test condition
    - depending on attribute types
  - Measure for evaluating the goodness of a test condition

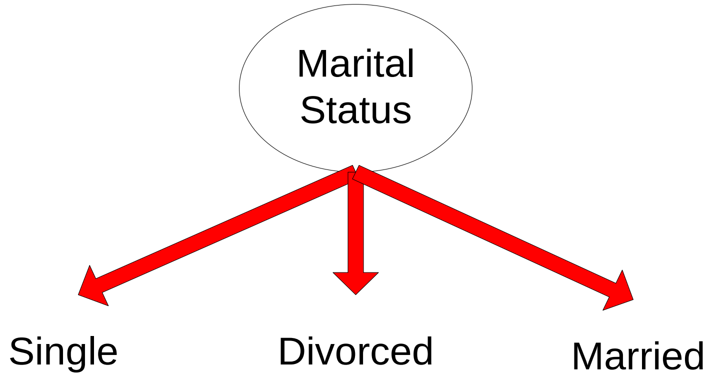
- How should the splitting procedure stop?
  - Stop splitting if all the records belong to the same class or have identical attribute values
  - Early termination

Methods for Expressing Test Conditions

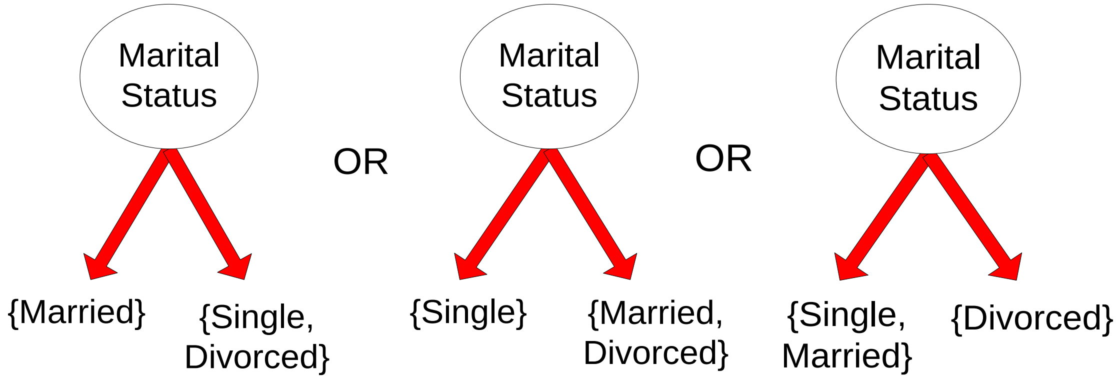
- Depends on attribute types
  - Binary
  - Nominal
  - Ordinal
  - Continuous

Test Condition for Nominal Attributes

- Multi-way split:
  - Use as many partitions as distinct values.

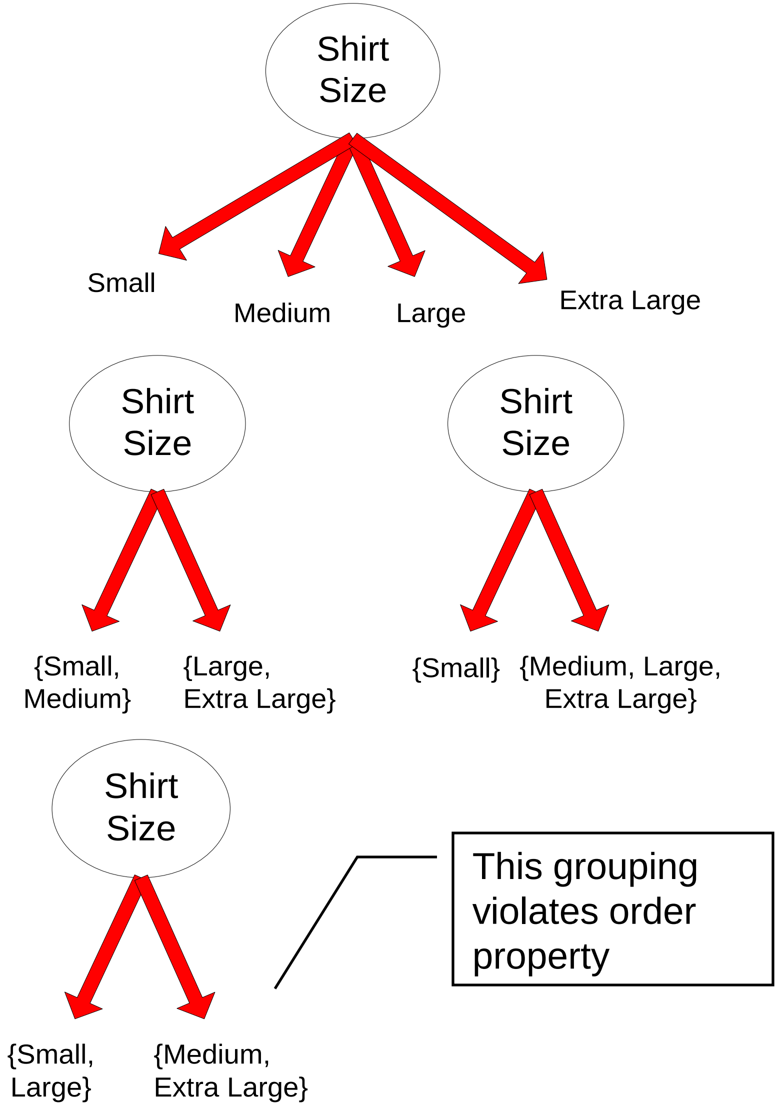


- Binary split:
  - Divides values into two subsets

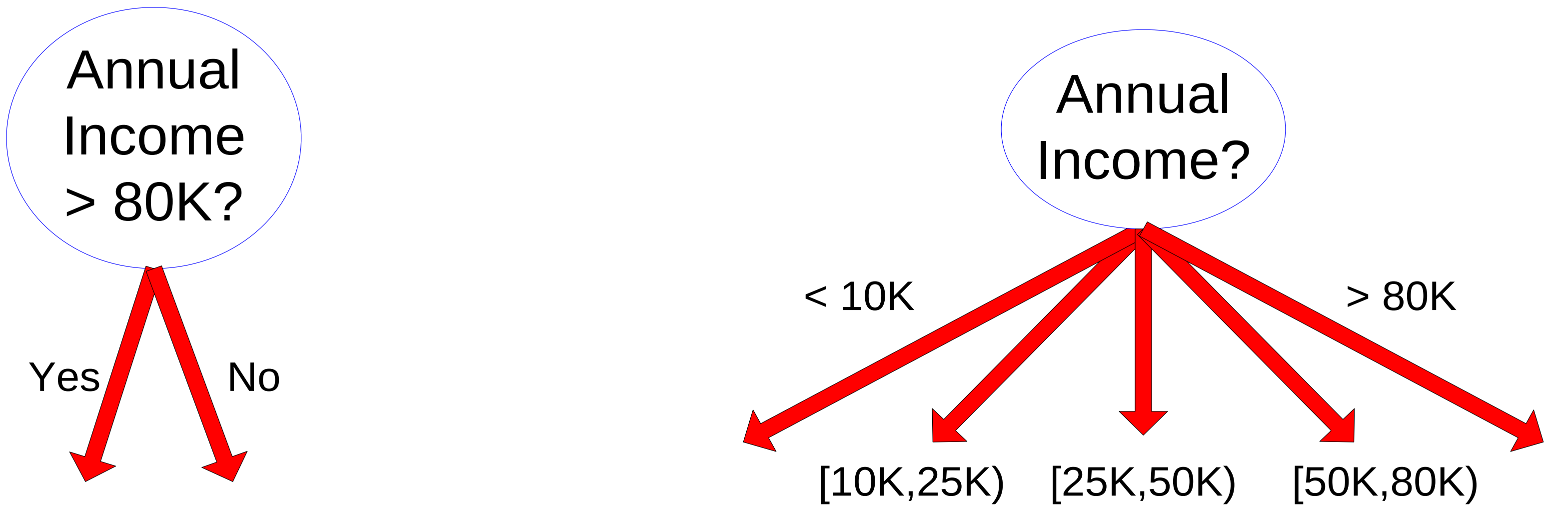


Test Condition for Ordinal Attributes

- **Multi-way split:**
  - Use as many partitions as distinct values
- **Binary split:**
  - Divides values into two subsets
  - Preserve order property among attribute values



Test Condition for Continuous Attributes



(i) Binary split

(ii) Multi-way split

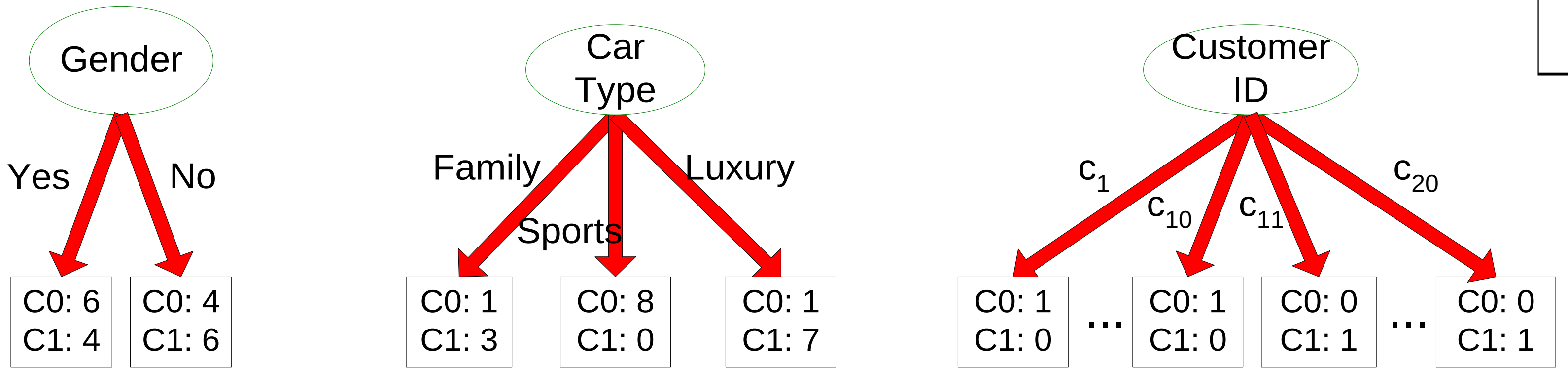
Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute
    - Ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
    - Static – discretize once at the beginning
    - Dynamic – repeat at each node
  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be more compute intensive

How to determine the Best Split

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

Before Splitting: 10 records of class 0,
10 records of class 1



Which test condition is the best?

FILKOM

How to determine the Best Split

- Greedy approach:
  - Nodes with purer class distribution are preferred

- Need a measure of node impurity:

C0: 5	C0: 9
C1: 5	C1: 1
High degree of impurity	Low degree of impurity

Measures of Node Impurity

- Gini Index

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where $\boldsymbol{p_i(t)}$ is the frequency of class $\boldsymbol{i}$ at node t, and $\boldsymbol{c}$ is the total number of classes

- Entropy

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

- Misclassification error

$$Classification\ error = 1 - \max[p_i(t)]$$

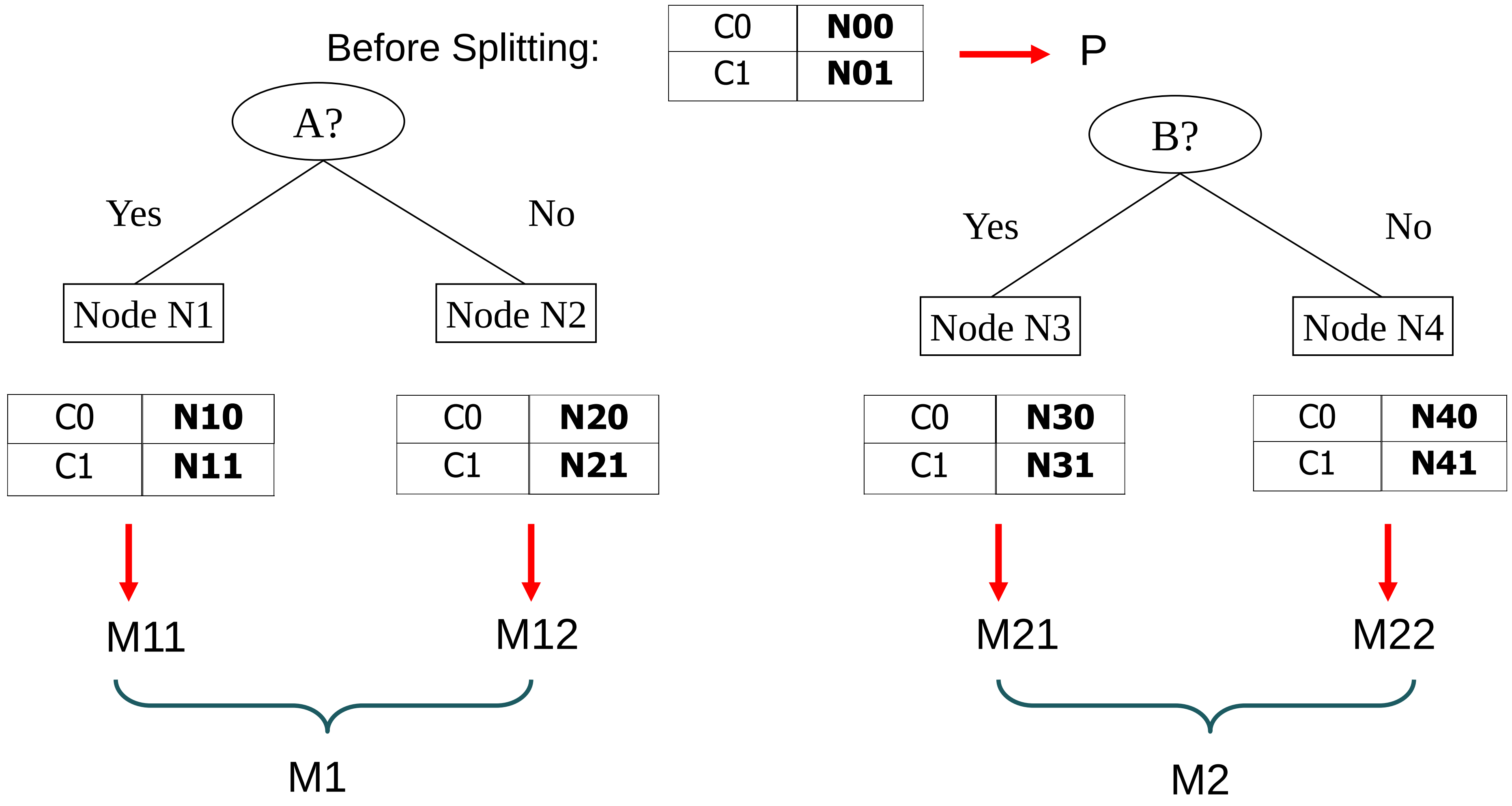
Finding the Best Split

1. Compute impurity measure (P) before splitting
2. Compute impurity measure (M) after splitting
   - Compute impurity measure of each child node
   - M is the weighted impurity of child nodes
3. Choose the attribute test condition that produces the highest gain

Gain = P - M

or equivalently, lowest impurity measure after splitting (M)

Finding the Best Split



Gain = P – M1 vs P – M2

Measure of Impurity: GINI

- Gini Index for a given node $\boldsymbol{t}$:

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where $\boldsymbol{p_i(t)}$ is the frequency of class $\boldsymbol{i}$ at node $\boldsymbol{t}$, and $\boldsymbol{c}$ is the total number of classes

- Maximum of $1 - 1/c$ when records are equally distributed among all classes, implying the least beneficial situation for classification
- Minimum of 0 when all records belong to one class, implying the most beneficial situation for classification
- Gini index is used in decision tree algorithms such as CART, SLIQ, SPRINT

Measure of Impurity: GINI

- Gini Index for a given node $\boldsymbol{t}$:

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

  - For 2-class problem (p, 1 – p):
    - GINI = $1 - p^2 - (1 - p)^2 = 2p\,(1-p)$

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Computing Gini Index of a Single Node

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – $P(C1)^2$ – $P(C2)^2$ = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – $(1/6)^2$ – $(5/6)^2$ = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – $(2/6)^2$ – $(4/6)^2$ = 0.444

Computing Gini Index for a Collection of Nodes

- When a node p is split into k partitions (children)

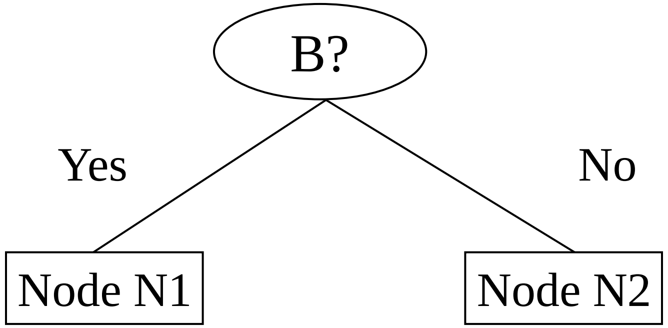
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i,

n = number of records at parent node p.

Binary Attributes: Computing GINI Index

- Splits into two partitions (child nodes)
- Effect of Weighing partitions:
  - Larger and purer partitions are sought



	Parent
C1	**7**
C2	**5**
Gini = 0.486	

Gini(N1)
$= 1 - (5/6)^2 - (1/6)^2$
$= 0.278$

Gini(N2)
$= 1 - (2/6)^2 - (4/6)^2$
$= 0.444$

	N1	N2
C1	**5**	**2**
C2	**1**	**4**
Gini=0.361		

Weighted Gini of N1 N2
$= 6/12 * 0.278 +$
$6/12 * 0.444$
$= 0.361$

Gain = 0.486 – 0.361 = 0.125

Categorical Attributes: Computing Gini Index

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
Gini	0.163		

Two-way split
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	0.468	

	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

Which of these is the best?

Continuous Attributes: Computing Gini Index

- Use Binary Decisions based on one value
- Several Choices for the splitting value
  - Number of possible splitting values = Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, A ≤ v and A > v
- Simple method to choose best v
  - For each v, scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.

ID	Home Owner	Marital Status	Annual Income	Defaulted
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Annual Income ?

	≤ 80	> 80
Defaulted Yes	0	3
Defaulted No	3	4

Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
	Annual Income									
Sorted Values →	60	70	75	85	90	95	100	120	125	220

Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

	Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
		Annual Income									
Sorted Values →		60	70	75	85	90	95	100	120	125	220

Split Positions →	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>

Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
	Annual Income									
Sorted Values	60	70	75	85	90	95	100	120	125	220

Split Positions	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes							0	3														
No							3	4														
Gini							0.343															

Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
	Annual Income									
Sorted Values	60	70	75	85	90	95	100	120	125	220

Split Positions	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes							0	3	1	2												
No							3	4	3	4												
Gini							0.343		0.417													

Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

Cheat		No		No		No		Yes		Yes		Yes		No		No		No		No		
	Annual Income																					
Sorted Values →		60		70		75		85		90		95		100		120		125		220		
Split Positions →	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		***0.300***		0.343		0.375		0.400		0.420	

Measure of Impurity: Entropy

- Entropy at a given node $\boldsymbol{t}$:

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

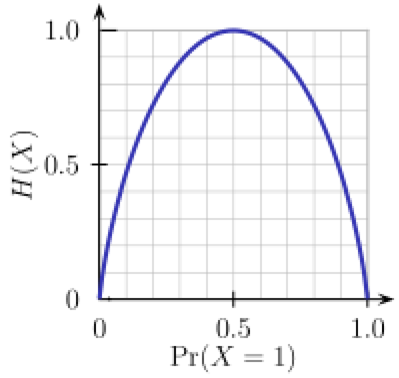
Where $\boldsymbol{p_i(t)}$ is the frequency of class $\boldsymbol{i}$ at node $\boldsymbol{t}$, and $\boldsymbol{c}$ is the total number of classes

  - Maximum of $\log_2 c$ when records are equally distributed among all classes, implying the least beneficial situation for classification
  - Minimum of 0 when all records belong to one class, implying most beneficial situation for classification

- Entropy based computations are quite similar to the GINI index computations

Brief Review of Entropy

- Entropy (Information Theory
  - A measure of uncertainty associated with a random variable
  - Calculation: For a discrete random variable Y taking m distinct values $\{y_1,\ldots,y_m\}$
    - $H(Y) = -\sum_{i=1}^{m} p_i \log(p_i)$ where $p_i = P(Y = y_i)$
  - Interpretation:
    - Higher entropy → higher uncertainty
    - Lower entropy → lower uncertainty
- Condition Entropy
  - $H(Y|X) = \sum_x p(x) H(Y|X = x)$



m = 2

Computing Entropy of a Single Node

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6   P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Computing Information Gain After Splitting

- Information Gain:

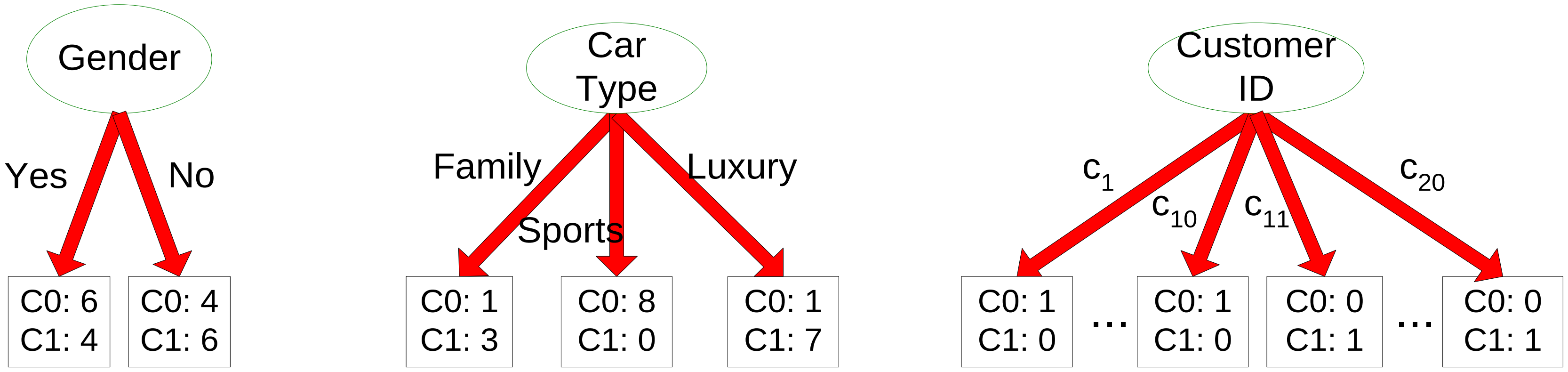
$$Gain_{split} = Entropy(p) - \sum_{i=1}^{k} \frac{n_i}{n} Entropy(i)$$

Parent Node, p is split into k partitions (children)
n_i is number of records in child node i

- Choose the split that achieves most reduction (maximizes GAIN)
- Used in the ID3 and C4.5 decision tree algorithms
- Information gain is the mutual information between the class variable and the splitting variable

Problem with large number of partitions

- Node impurity measures tend to prefer splits that result in large number of partitions, each being small but pure



- Customer ID has highest information gain because entropy for all the children is zero

Gain Ratio

- Gain Ratio:

$$Gain\ Ratio = \frac{Gain_{split}}{Split\ Info} \qquad Split\ Info = -\sum_{i=1}^{k} \frac{n_i}{n} log_2 \frac{n_i}{n}$$

- Parent Node, p is split into k partitions (children)
- n_i is number of records in child node i

- Adjusts Information Gain by the entropy of the partitioning ($Split\ Info$).
  - Higher entropy partitioning (large number of small partitions) is penalized!
- Used in C4.5 algorithm
- Designed to overcome the disadvantage of Information Gain

Gain Ratio

- Gain Ratio:

$$Gain\ Ratio = \frac{Gain_{split}}{Split\ Info} \qquad Split\ Info = -\sum_{i=1}^{k} \frac{n_i}{n} \log_2 \frac{n_i}{n}$$

- Parent Node, p is split into k partitions (children)
- n_i is number of records in child node i

	CarType		
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
Gini	0.163		

SplitINFO = 1.52

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	0.468	

SplitINFO = 0.72

	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

SplitINFO = 0.97

Measure of Impurity: Classification Error

- Classification error at a node t

$$Error(t) = 1 - \max_i [p_i(t)]$$

  - Maximum of $1 - 1/c$ when records are equally distributed among all classes, implying the least interesting situation
  - Minimum of 0 when all records belong to one class, implying the most interesting situation

Computing Error of a Single Node

$$Error(t) = 1 - \max_i [p_i(t)]$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

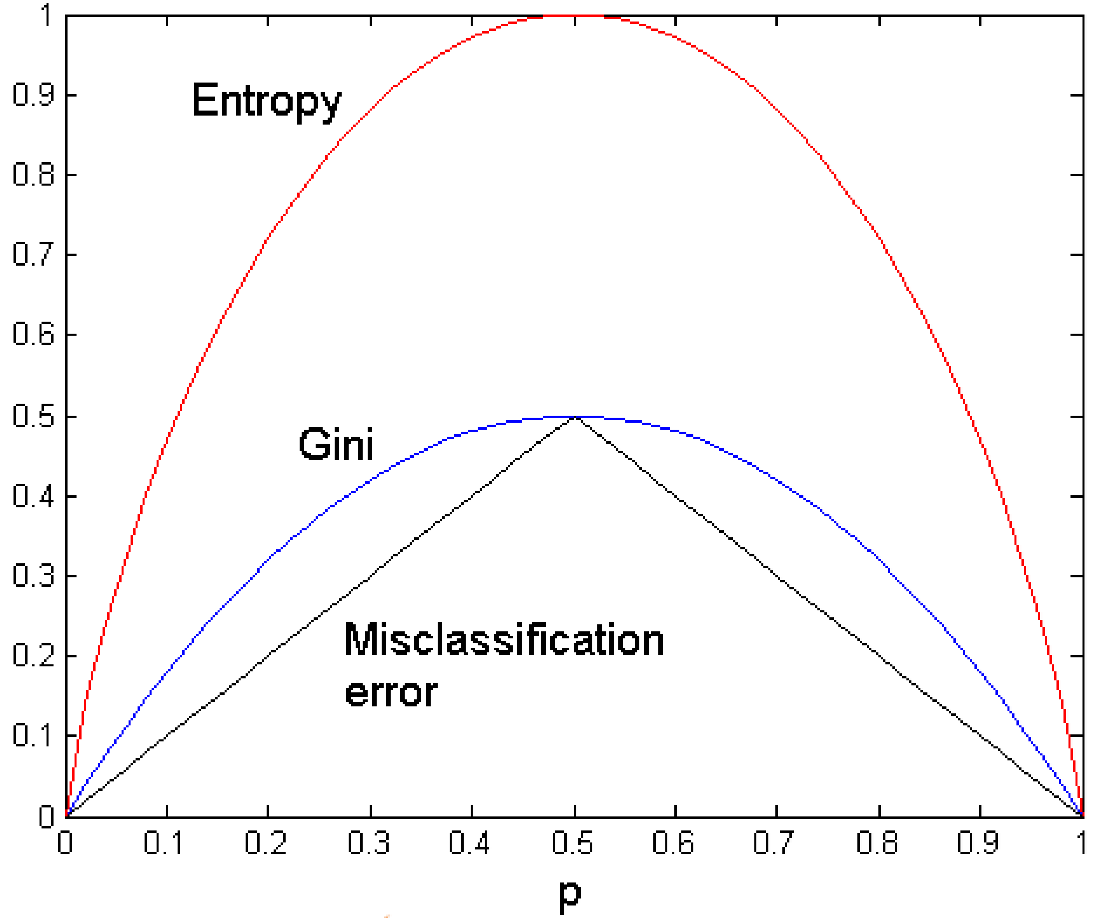
C1	**2**
C2	**4**

P(C1) = 2/6   P(C2) = 4/6

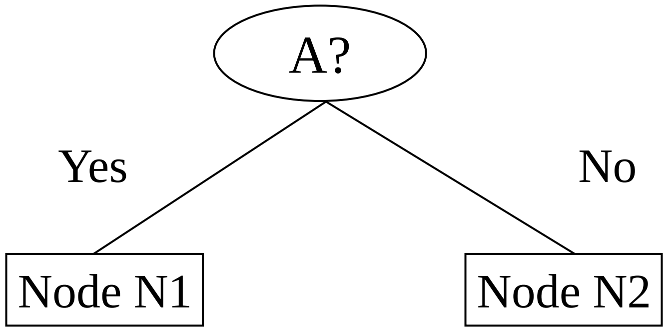
Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among Impurity Measures

- For a 2-class problem:



Misclassification Error vs Gini Index



	Parent
C1	**7**
C2	**3**
Gini = 0.42	

Gini(N1)
= 1 – $(3/3)^2$ – $(0/3)^2$
= 0

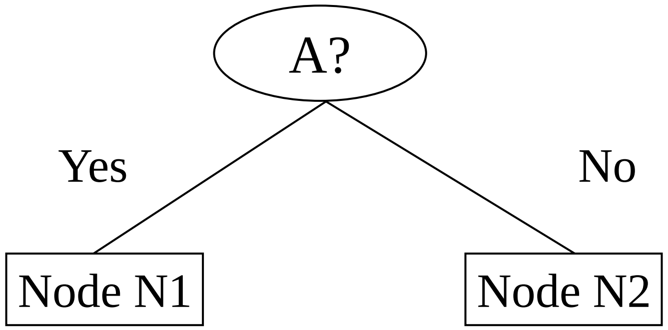
Gini(N2)
= 1 – $(4/7)^2$ – $(3/7)^2$
= 0.489

	N1	N2
C1	**3**	**4**
C2	**0**	**3**
Gini=0.342		

Gini(Children)
= 3/10 * 0
+ 7/10 * 0.489
= 0.342

Gini improves but error remains the same!!

Misclassification Error vs Gini Index



	Parent
C1	**7**
C2	**3**
Gini = 0.42	

	N1	N2
C1	**3**	**4**
C2	**0**	**3**
Gini=0.342		

	N1	N2
C1	**3**	**4**
C2	**1**	**2**
Gini=0.416		

Misclassification error for all three cases = 0.3 !

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let pi be the probability that an arbitrary tuple in D belongs to class Ci, estimated by |Ci, D|/|D|
- Expected information (entropy) needed to classify a tuple in D:

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

- Information needed (after using A to split D into v partitions) to classify D:

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

- Information gained by branching on attribute A

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

Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

age	income	student	credit_rating	buys_computer
<=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

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

- $\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

- Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

Computing Information-Gain for Continuous-Valued Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible split point
    - (ai+ai+1)/2 is the midpoint between the values of ai and ai+1
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

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

  - GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex.

$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$$

  - gain_ratio(income) = 0.029/1.557 = 0.019
- The attribute with the maximum gain ratio is selected as the splitting attribute

Gini Index (CART, IBM IntelligentMiner)

- If a data set *D* contains examples from *n* classes, gini index, *gini*(D) is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class *j* in *D*

- If a data set *D* is split on A into two subsets D_1 *and* D_2, the *gini* index *gini*(D) is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (need to enumerate all the possible splitting points for each attribute)

Computation of Gini Index

- Ex. D has 9 tuples in buys_computer = “yes” and 5 in “no” $gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$
- Suppose the attribute income partitions *D* into 10 in D_1: {low, medium} and 4 in D_2

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

$$= 0.443$$

$$= Gini_{income \in \{high\}}(D).$$

  - Gini{low,high} is 0.458; Gini{medium,high} is 0.450. Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index
- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - **Information gain:**
    - biased towards multivalued attributes
  - **Gain ratio:**
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - **Gini index:**
    - biased to multivalued attributes
    - has difficulty when # of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

Other Attribute Selection Measures

- **CHAID**: a popular decision tree algorithm, measure based on χ2 test for independence
- **C-SEP**: performs better than info. gain and gini index in certain cases
- **G-statistic**: has a close approximation to χ2 distribution
- **MDL (Minimal Description Length)** principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - **CART**: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

Overfitting and Tree Pruning

- **Overfitting**: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - **Prepruning**: *Halt tree construction early*-do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - **Postpruning**: *Remove branches* from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

Enhancements to Basic Decision Tree Induction

- Allow for **continuous-valued attributes**
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle **missing attribute values**
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- **Attribute construction**
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

Classification in Large Databases

- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why is decision tree induction popular?
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - can use SQL queries for accessing databases
  - comparable classification accuracy with other methods
- RainForest (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - Builds an AVC-list (attribute, value, class label)

Scalability Framework for RainForest

- Separates the scalability aspects from the criteria that determine the quality of the tree
- Builds an AVC-list: **AVC (Attribute, Value, Class_label)**
- **AVC-set** (of an attribute X)
  - Projection of training dataset onto the attribute X and class label where counts of individual class label are aggregated
- **AVC-group** (of a node n)
  - Set of AVC-sets of all predictor attributes at the node n

Rainforest: Training Set and Its AVC Sets

Training Examples

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

AVC-set on *Age*

student	Buy_Computer	
	yes	no
yes	6	1
no	3	4

AVC-set on *income*

Age	Buy_Computer	
	yes	no
<=30	2	3
31..40	4	0
>40	3	2

AVC-set on *Student*

Credit rating	Buy_Computer	
	yes	no
fair	6	2
excellent	3	3

AVC-set on *credit_rating*

income	Buy_Computer	
	yes	no
high	2	2
medium	4	2
low	3	1

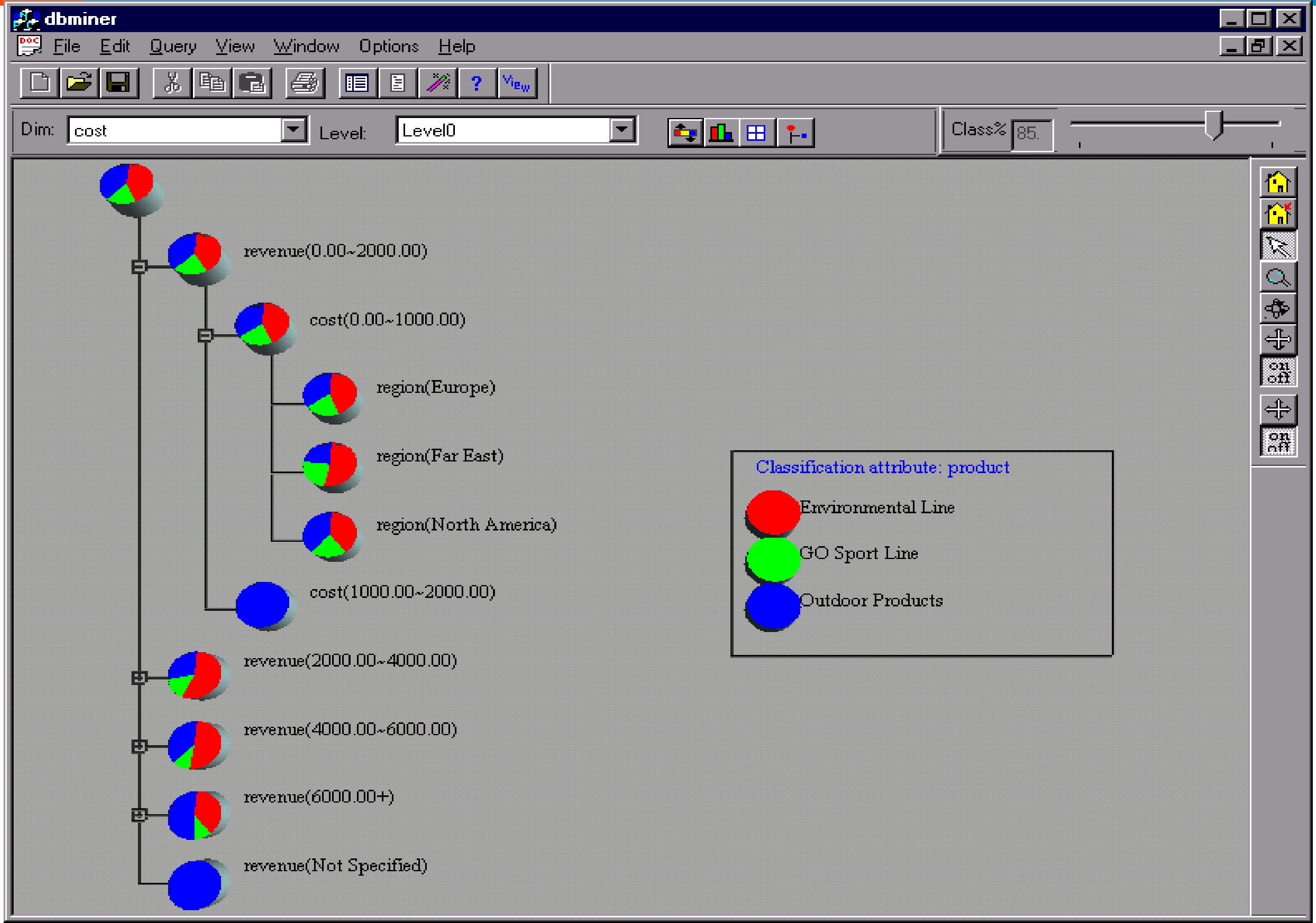
BOAT (Bootstrapped Optimistic Algorithm for Tree Construction)

- Use a statistical technique called *bootstrapping* to create several smaller samples (subsets), each fits in memory
- Each subset is used to create a tree, resulting in several trees
- These trees are examined and used to construct a new tree *T'*
  - It turns out that *T'* is very close to the tree that would be generated using the whole data set together
- Adv: requires only two scans of DB, an incremental alg.

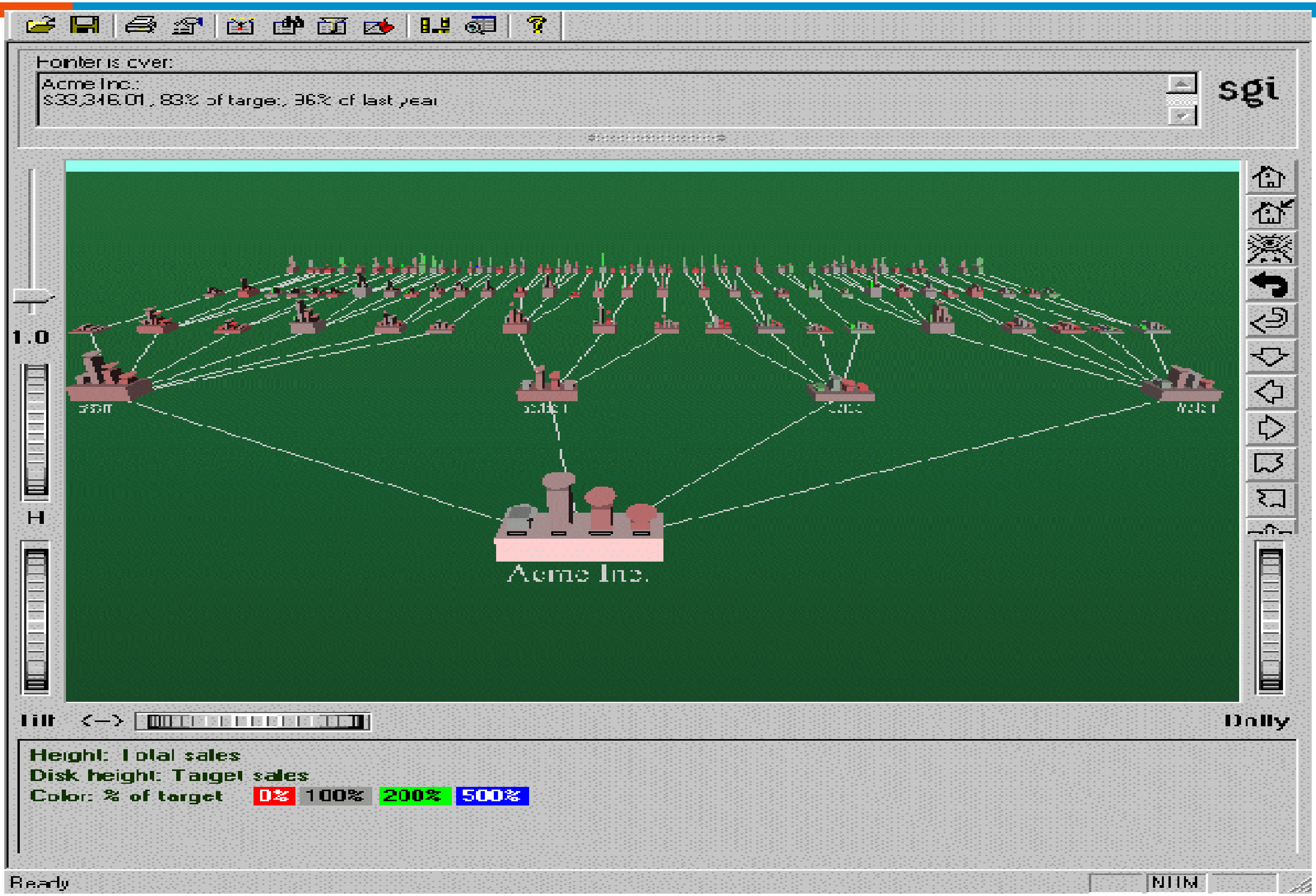
Decision Tree Based Classification

- Advantages:
  - Relatively inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Robust to noise (especially when methods to avoid overfitting are employed)
  - Can easily handle redundant attributes
  - Can easily handle irrelevant attributes (unless the attributes are interacting)
- Disadvantages: .
  - Due to the greedy nature of splitting criterion, interacting attributes (that can distinguish between classes together but not individually) may be passed over in favor of other attributed that are less discriminating.
  - Each decision boundary involves only a single attribute

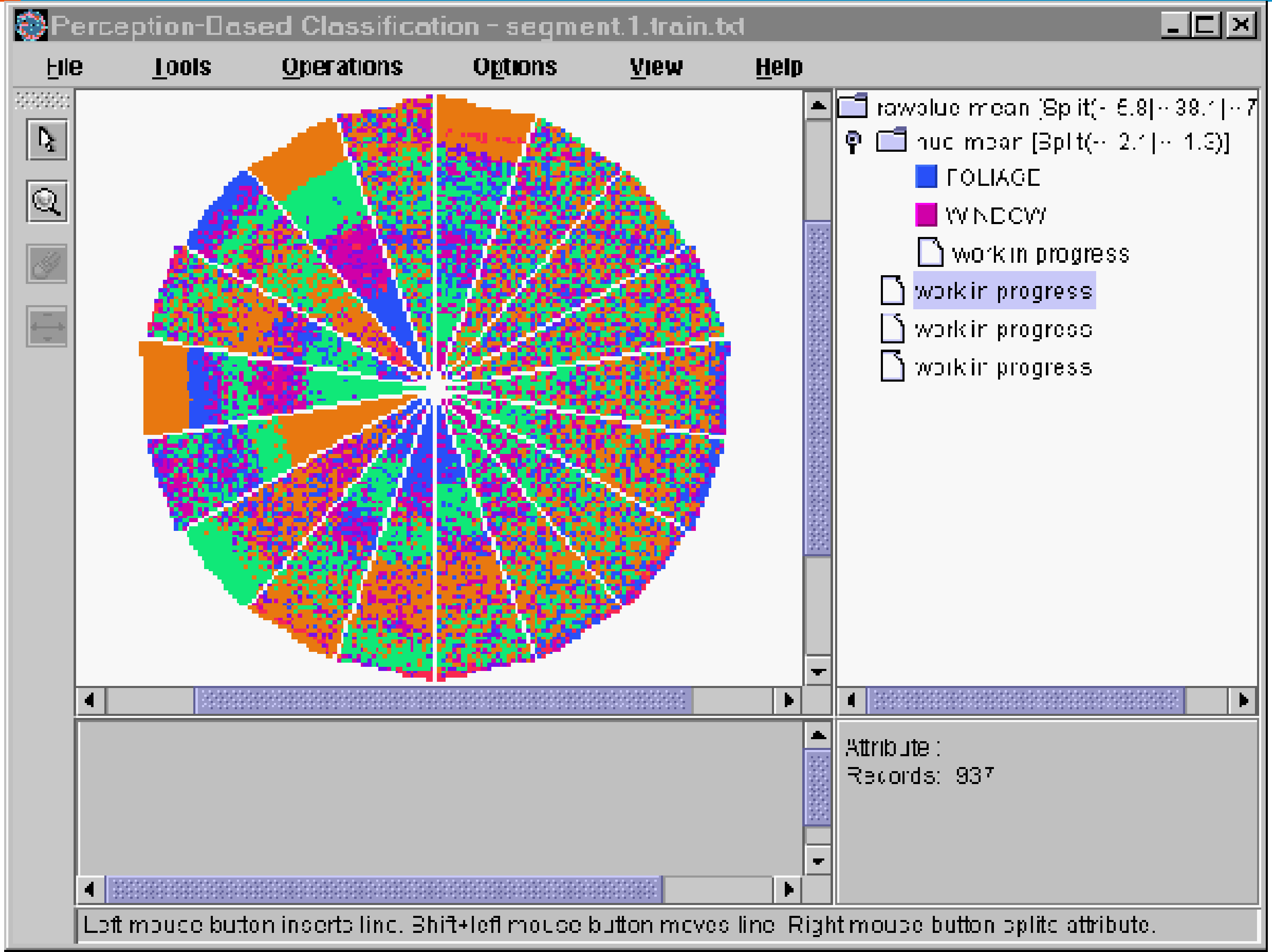
Presentation of Classification Results



Visualization of a Decision Tree in SGI/MineSet 3.0



Interactive Visual Mining by Perception-Based Classification (PBC)



Evaluasi

Tipe Galat/Error dalam Klasifikasi

- **Training Error** → banyaknya kesalahan klasifikasi yang dilakukan pada data latih.
- **Generalization Error** → kesalahan yang diharapkan dari model pada data yang sebelumnya tidak diketahui.
- **Good classifier/classification model** → harus sesuai (*fit*) dengan data latih dengan baik dan secara akurat mengklasifikasikan data yang belum pernah diketahui
- Model yang terlalu sesuai/cocok dengan data latih dapat memiliki generalization error yang lebih buruk daripada model dengan training error lebih tinggi → **model overfitting**.

Overfitting - Penyebab

- Adanya derau atau *noise*
- Sampel kurang representatif
- Prosedur pembandingan berganda (*Multiple Comparison Procedure*)
- Estimasi generalization eror:
  - gunakan resubstitution estimate (solusi alternative)
  - mengikutikan model complexity (sesederhana mungkin)
  - gunakan validation set (bagi data latih menjadi dua himpunan bagian yang lebih kecil)

Mengevaluasi Kinerja dari Classifier

- Confusion Matrix
- Holdout Method
- Random Subsampling
- Cross-Validation

Confusion Matrix

- Evaluasi kinerja dari model klasifikasi berdasarkan dari **banyaknya** data uji yang benar dan salah diprediksi oleh model
- *Banyak* ini kemudikan ditabulasikan dalam table yang dikenal dengan **confusion matrix**.
- Confusion matrix untuk masalah 2-class (binary classification):

		Predicted Class	
		$Class = 1$	$Class = 0$
Actual Class	$Class = 1$	f_{11}	f_{10}
	$Class = 0$	f_{01}	f_{00}

FILKOM

Confusion Matrix (cont)

- Tiap entri f_{ij} dalam tabel menunjukkan banyaknya data dari kelas *i* diprediksi sebagai kelas *j*
- Misalnya, f_{01} adalah banyaknya data dari kelas 0 yang salah diprediksi sebagai kelas 1
- Berdasarkan entri pada confusion matrix, total banyaknya prediksi yang benar dibuat oleh model adalah $(f_{11} + f_{oo})$ dan total salah prediksi adalah $(f_{1o} + f_{o1})$.

Confusion Matrix: Binary Class Problem (cont)

		Predicted Class	
		$Class = 1$	$Class = 0$
Actual Class	$Class = 1$	f_{11}	f_{10}
	$Class = 0$	f_{01}	f_{00}

Total number of predictions $f_{11} + f_{10} + f_{01} + f_{00}$

$$Error\ rate = \frac{Number\ of\ wrong\ predictions}{Total\ number\ of\ predictions} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

Confusion Matrix: Binary Class Problem (cont)

Varian I

		Predicted Class	
		Class = 1	Class = 0
Actual Class	Class = 1	True Positive (TP)	False Negative (FN)
	Class = 0	False Positive (FP)	True Negative (TN)

Varian II

		Actual Class	
		Class = 1	Class = 0
Predicted Class	Class = 1	True Positive (TP)	False Positive (FP)
	Class = 0	False Negative (FN)	True Negative (TN)

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

$$Recal = Sensitivity = \frac{TP}{TP + FN} \qquad Error\ \ rate = \frac{FP + FN}{TP + TN + FP + FN}$$

Confusion Matrix (Multi-Class Problem): Varian I

		Predicted				
		A	B	C	D	E
Actual	A	TP_A	E_{AB}	E_{AC}	E_{AD}	E_{AE}
	B	E_{BA}	TP_B	E_{BC}	E_{BD}	E_{BE}
	C	E_{CA}	E_{CB}	TP_C	E_{CD}	E_{CE}
	D	E_{DA}	E_{DB}	E_{DC}	TP_D	E_{DE}
	E	E_{EA}	E_{EB}	E_{EC}	E_{ED}	TP_E

FP untuk kelas A $$FP_i = \sum_{l}^{L} E_{li} - TP_i$$

FN untuk kelas A $$FN_i = \sum_{l}^{L} E_{il} - TP_i$$

TN untuk kelas A $$TN_i = \sum_{l}^{L} \sum_{k}^{L} E_{lk} - TP_i - FP_i - FN_i$$

Confusion Matrix (Multi-Class Problem): Varian I

- **Total banyak data uji** pada suatu kelas → jumlah dari baris yang bersesuaian (TP+FN untuk kelas tsb.)
- Total banyak **FN** dari suatu kelas → jumlah nilai pada baris bersesuaian (*tidak termasuk* TP)
- Total banyak **FP** dari suatu kelas → jumlah nilai pada kolom bersesuaian (*tidak termasuk* TP)
- Total banyak **TN** dari suatu kelas → penjumlahan dari seluruh kolom dan baris *tidak termasuk* kolom dan baris kelas tsb.

Confusion Matrix (Multi-Class Problem): Varian I

- **Accuracy** → jumlah hasil klasifikasi yang benar (semua TPs) dibagi total banyak data
- **Precision** = TP/(TP+FP) for each class
  - Ex: Precision A = $TP_A/(TP_A+E_{BA}+E_{CA}+E_{DA}+E_{EA})$
- **Recall** → true-positive rate of the considered class
- **Recall** = Sensitivity = TP/(TP+FN) for each class
  - Ex: Recall A = $TP_A/(TP_A+E_{AB}+E_{AC}+E_{AD}+E_{AE})$
- **Specificity** → true-negative rate of the considered class
- **Specificity** = TN/(TN+FP) for each class
  - Ex: Specificity A = $TN_A/(TN_A+E_{BA}+E_{CA}+E_{DA}+E_{EA})$

Holdout Method

- Pada **holdout method**, data orisinal beserta labelnya **dibagi/dipartisi menjadi dua himpunan terpisah**, yaitu training dan test set.
- Model klasifikasi kemudian diinduksi dari training set dan kinerjanya dievaluasi melalui test set
- **Proporsi** dari data yang digunakan untuk **training** dan untuk **testing** biasanya atas kebijakan analis (misal **50-50 atau 2/3 untuk training** dan **1/3 untuk testing**).
- Akurasi dari classifier dapat diestimasi berdasarkan keakuratan model yang diinduksi pada test set

Holdout Method (cont)

- Keterbatasan:
  - Lebih sedikit data berlabel tersedia untuk pelatihan karena beberapa data diambil untuk pengujian. Akibatnya, model yang diinduksi mungkin tidak sebagus ketika semua contoh berlabel digunakan untuk pelatihan.
  - Model ini mungkin sangat tergantung pada komposisi dari training dan test set. Semakin kecil ukuran training set, semakin besar *ragam* pada model. Di sisi lain, jika training set terlalu besar, maka perkiraan akurasi yang dihitung dari test set yang lebih kecil kurang dapat diandalkan/dipercaya.
  - Training dan test set tidak lagi terpisah satu sama lain. Dikarenakan training dan test set adalah himpunan bagian dari data asli, kelas yang terlalu terwakili dalam satu himpunan bagian akan kurang terwakili di yang lain, dan sebaliknya

Random Sampling

- **Metode holdout dapat diulangi beberapa kali** untuk **meningkatkan perkiraan** dari kinerja classifier
- Pendekatan ini dikenal sebagai **random subsampling**
- Dimisalkan acc_i adalah akurasi model selama iterasi i^{th}, maka akurasi keseluruhan dihitung dengan rumus:

$$acc_{sub} = \sum_{i=1}^{k} \frac{acc_i}{k}$$

Random Sampling (cont)

- Keterbatasan
  - Tidak memanfaatkan data sebanyak mungkin untuk pelatihan.
  - Tidak memiliki kendali atas berapa kali setiap data digunakan untuk pengujian dan pelatihan. Akibatnya, beberapa data mungkin digunakan untuk pelatihan lebih sering daripada yang lain.

Cross-Validation

- Dalam pendekatan ini, tiap data digunakan sebanyak berapa kali pelatihan dan tepat satu kali untuk pengujian.
- Untuk mengilustrasikan metode ini, anggaplah kita mempartisi data menjadi dua himpunan bagian berukuran sama.
- Pertama, pilih salah satu himpunan bagian untuk pelatihan dan yang lainnya untuk pengujian.
- Lalu, tukar peranan dari himpunan bagian sehingga yang awalnya sebagai training set menjadi test set dan sebaliknya.
- Pendekatan ini disebut **two-fold cross-validation**.
- Total galat didapat dengan menyimpulkan kesalahan pada kedua proses
- Pada contoh, tiap data digunakan tepat sekali untuk pelatihan dan sekali untuk pengujian

Cross-Validation (cont)

- Metode ***k*-fold cross-validation** menggeneralisasi-kan pendekatan ini dengan membagi data menjadi partisi sama besar sebanyak k.
- Dalam tiap proses, satu partisi dipilih untuk pengujian sementara lainnya untuk pelatihan
- Prosedur ini diulangi sebanyak *k* kali, sehingga tiap partisi pernah digunakan dalam pengujian sekali
- Total kesalahan didapat dengan menyimpulkan error dari seluruh proses *k*.

Cross-Validation (cont)

- Kasus khusus dari metode k-fold cross-validation mengatur k : N, dengan ukuran dari data set.
- Disebut sebagai **leave-one-out approach**, tiap test set berisi hanya satu data.
- Pendekatan ini memiliki kelebihan memanfaatkan sebanyak mungkin data digunakan untuk pelatihan
- Sebagai tambahan, test set bersifat *mutually exclusive* dan secara efektif mencakup seluruh data set.
- **Kelemahan** dari pendekatan ini adalah komputasi berbiaya tinggi untuk mengulangi prosedur sebanyak N kali
- Terlebih, karena tiap test set hanya berisi satu record, ragam dari metrik kinerja yang diperkirakan cenderung tinggi

Contoh cross validation k=5

- Contoh fold k=5, splitting acak dilakukan hanya sekali di awal, di tiap running tidak diacak
- Running fold-1

Data Uji

Data Latih

- Running fold-2

- Running fold-3

- Running fold-4

- Running fold-5

- *Di tiap fold hitung masing-masing performance metrics, lalu di akhir fold hitung rata-rata (averaging) untuk mengetahui kinerja secara global dari classifier*

Methods for Comparing Classifiers

- Akan sangat berguna untuk membandingkan kinerja dari classifier yang berbeda untuk mengetahui mana classifier yang bekerja lebih baik terhadap data set yang diberikan
- Namun, tergantung dari ukuran data, perbedaan yang terlihat pada akurasi terhadap dua classifier mungkin saja tidak signifikan secara statistik
- Sebagai ilustrasi, misalnya ada pasangan model klasifikasi M_A dan M_B. M_A mencapai akurasi 85% ketika dievaluasi pada teset set berisi 30 data, sementara M_B mencapai akurasi 75% pada tes set berbeda berisi 5000 data.
- Dari informasi ini, apakah M_A model yang lebih baik dari M_B?

Methods for Comparing Classifiers (cont)

- Contoh sebelumnya menimbulkan dua pertanyaan utama mengenai statistik signifikansi pada metrik kinerja

1. Meski M_A memiliki akurasi lebih tinggi dibandingkan M_B, tapi itu diuji dengan test set yang lebih kecil. Berapa besar keyakinan/*confidence* yang dapat diberikan pada akurasi M_A? → **estimating confidence interval for accuracy**
2. Apakah mungkin untuk menjelaskan perbedaan akurasi sebagai hasil dari variasi dalam komposisi test set? → **comparing the performance of two model**

Estimating Confidence Interval for Accuracy

- Confidence interval formula:

$$\frac{2 \times N \times acc + Z_{\alpha/2}^2 \pm Z_{\alpha/2}\sqrt{Z_{\alpha/2}^2 + 4Nacc - 4Nacc^2}}{2(N + Z_{\alpha/2}^2)}$$

- The following table shows the values of $Z_{\alpha/2}$ at different confidence levels:

1-α	0.99	0.98	0.95	0.9	0.8	0.7	0.5
$Z_{\alpha/2}$	2.58	2.33	1.96	1.65	1.28	1.04	0.67

Example

- Pertimbangkan model yang memiliki akurasi 80% ketika dievaluasi pada 100 data uji. Apa *confidence interval* untuk akurasi sebenarnya pada tingkat *confidence* 95% (0,95)?
- Tingkat confidence 95% berkorespondensi pada $Z_{\alpha/2}$=1,96 sesuai table di atas. Dengan memasukkan perhitungan pada rumus akan menghasilkan confidence interval antara 71,1% dan 86,7%.
- Tabel berikut menunjukkan *confidence interval* saat banyak data *N* meningkat:

N	20	50	100	500	1000	5000
Confidence Interval	0.584 - 0.919	0.670 – 0.888	0.711 – 0.867	0.763 – 0.833	0.774 – 0.824	0.789 – 0.811

Perhatikan bahwa confidence interval menjadi lebih rapat ketika N meningkat

Latihan

- Berdasarkan dataset berikut (slide berikutnya), jika Customer ID 3, 5, 9, 12, 15, and 19 sebagai tuple uji, klasifikasikan data menggunakan algoritme k-NN dan evaluasi hasilnya menggunakan:
  - Confusion Matrix (accuracy, error rate, precission, recall, specificity)
  - Hold-out 70:30 (70% training vs 30% test set)
  - k-Fold Cross Validation (k=5) (k=5 → 80% training vs 20% test set)

Customer ID	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

Terima Kasih