

Data Mining

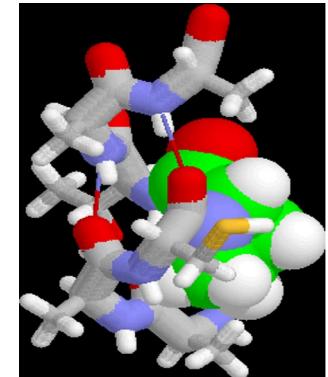
Classificazione: Concetti e tecniche di base

Classificazione: Definizione

- Data una collezione di record (**training set**)
Ogni record è composto da un insieme di **attributi**, di cui uno esprime la **classe** di appartenenza del record.
- Trova un **modello** per l'attributo di classe che esprima il valore dell'attributo in funzione dei valori degli altri attributi.
- Obiettivo: record **non noti** devono essere assegnati a una classe nel modo più accurato possibile
Viene utilizzato un **test set** per determinare l'accuratezza del modello.
Normalmente, il data set fornito è suddiviso in **training set** e **test set**.
Il primo è utilizzato per costruire il modello, il secondo per validarlo.
- I classificatori possono essere utilizzati sia a scopo descrittivo sia a scopo predittivo
- Sono più adatti ad attributi nominali (binari o discreti) poiché faticano a sfruttare le relazioni implicite presenti negli attributi ordinali, numerici o in presenza di gerarchie di concetti (es. scimmie e uomini sono primati).

Applicazioni

- Predire se una cellula tumorale è benigna o maligna in base alle sue caratteristiche
- Classificare se una transazione con carta di credito sia o meno fraudolenta
- Classificare le strutture proteiche secondarie in alpha-helix, beta-sheet, or random coil
- Classificare le news in base all'argomento: finanza, meteo, sport, intrattenimento, ecc.
- Catalogazione di email (spam o non-spam)
- Catalogazione di galassie (forma a spirale o ellettica o irregolare)



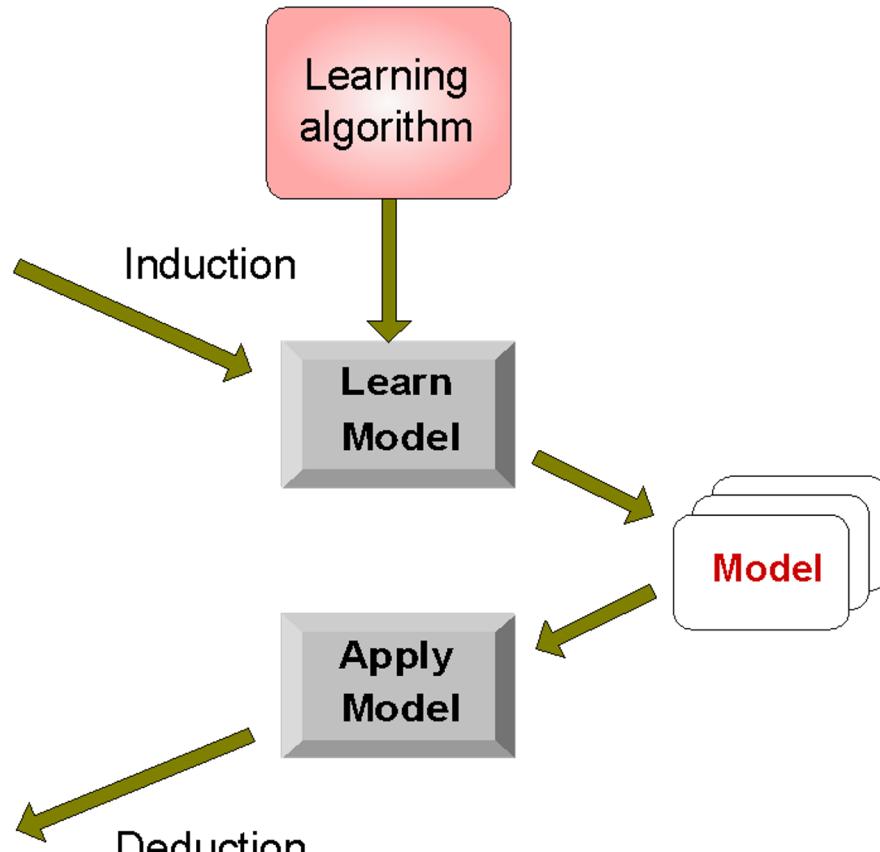
Approccio generale

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



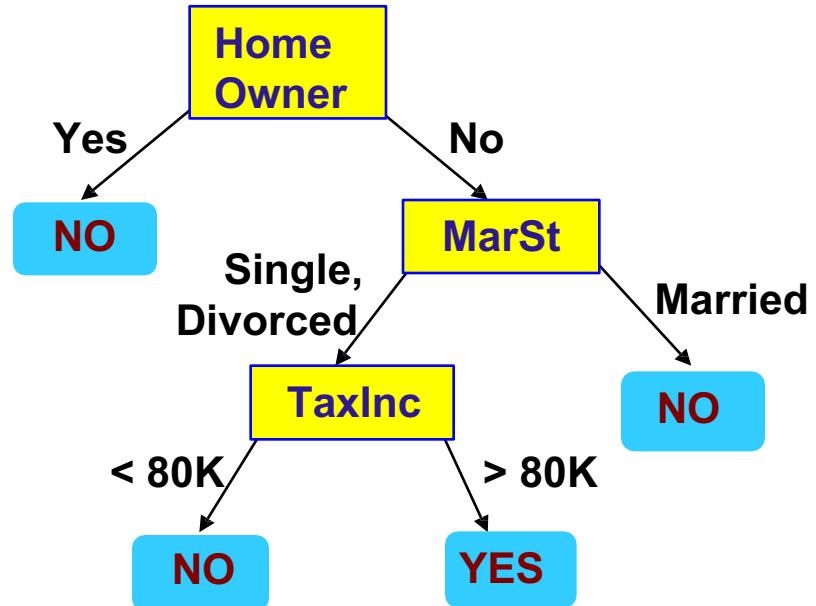
Tecniche di Classificazione

- Classificatori di base
 - Metodi basati su Alberi Decisionali
 - Metodi basati su regole
 - Nearest-neighbor
 - Reti neurali
 - Deep Learning
 - Classificazione Baiesiana/Reti Bayesiane
 - Support Vector Machines
 - Logistic regression
- Classificatori Ensemble
 - Boosting, Bagging, Random Forests

Alberi decisionali (Decision Tree)

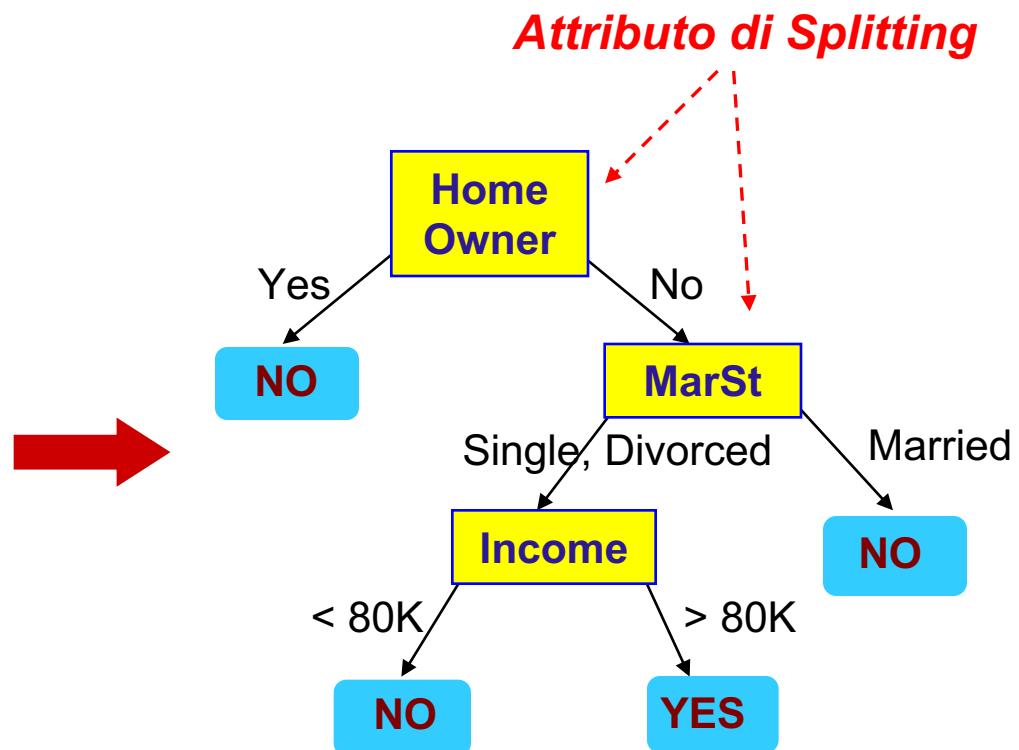
- È una delle tecniche di classificazione maggiormente utilizzate che permette di rappresentare con un albero un insieme di regole di classificazione.
- Struttura ad albero dove i nodi interni denotano attributi, I nodi foglia valori dell'attributo di classe e le etichette degli archi denotano possibili valori degli attributi dal quale dipartono.

Ciascun percorso radice-foglia rappresenta una regola di Classificazione



Esempio di Albero di Decisione

		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	

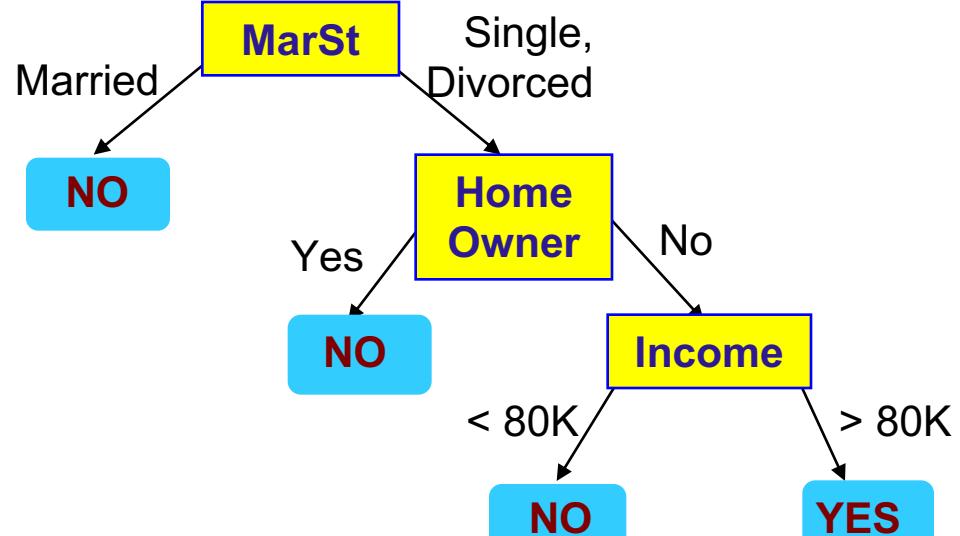


Training Data

Modello: Decision Tree

Esempio 2 di Albero di Decisione

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower	
				Categorical	categorical
1	Yes	Single	125K	No	continuous
2	No	Married	100K	No	class
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	



Ci possono essere più alberi di decisione per lo stesso data set

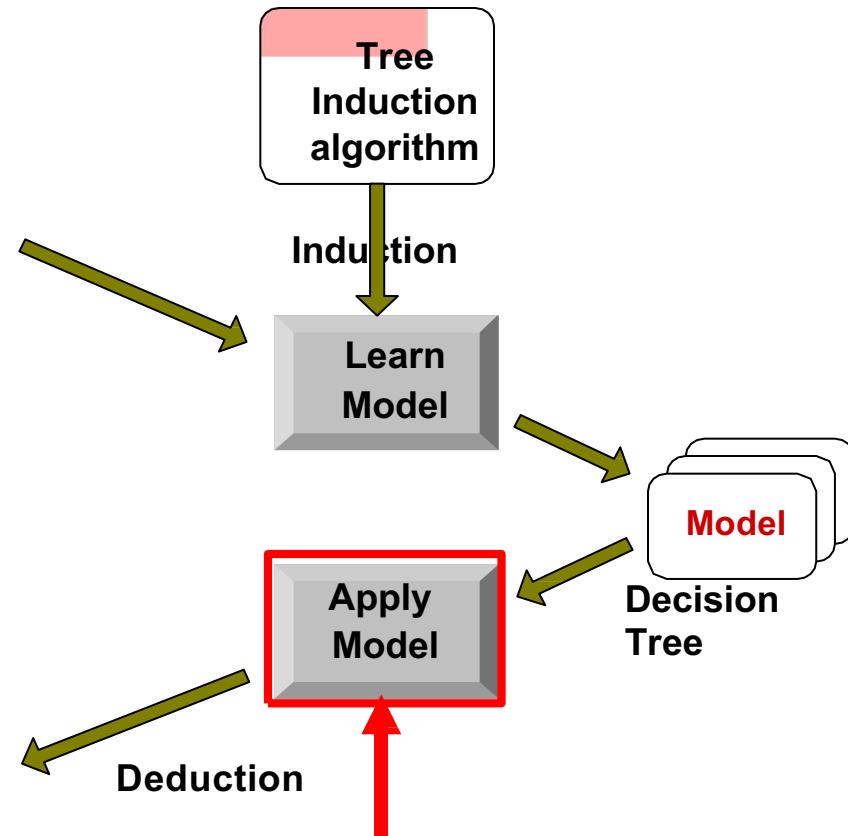
Applicare il modello al data set

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

Training Set

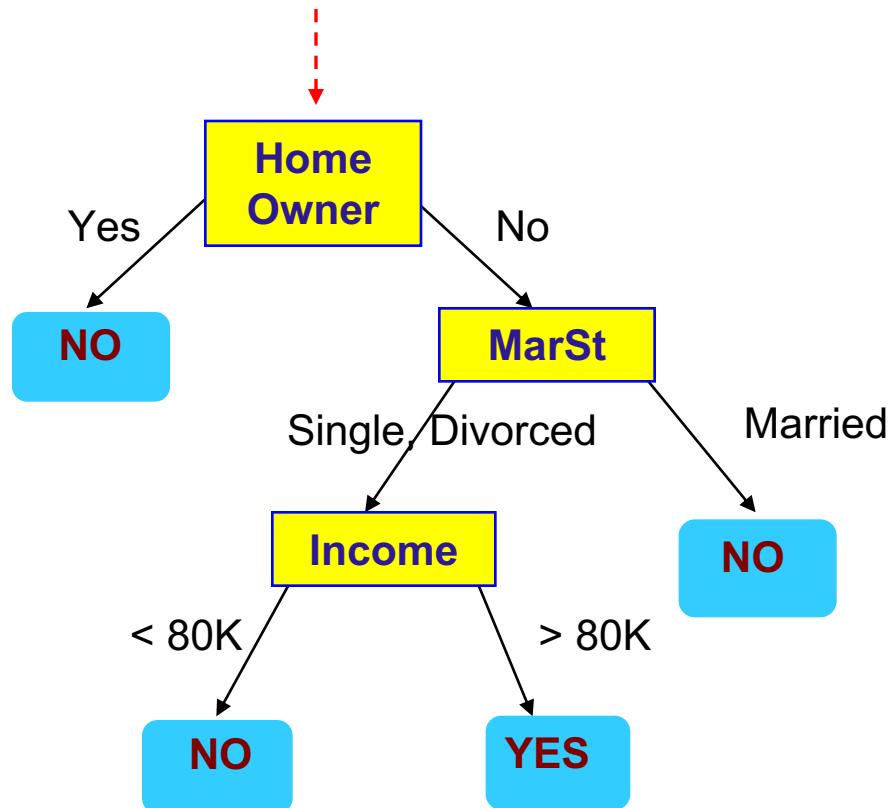
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Applica il Modello al Test Data Set

Si parte dalla radice



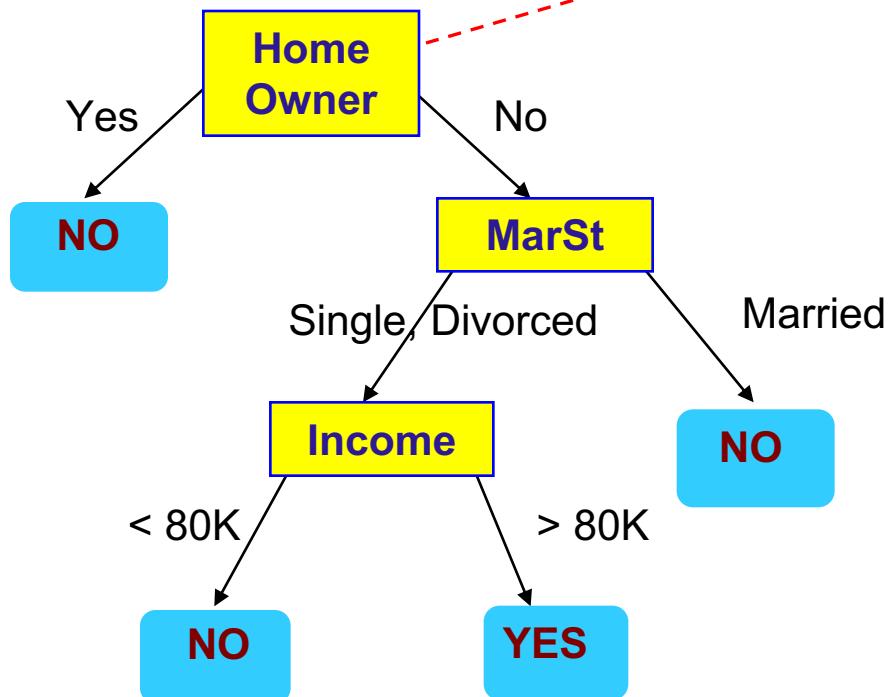
Test Data

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

Applica il Modello al Test Data Set

Test Data

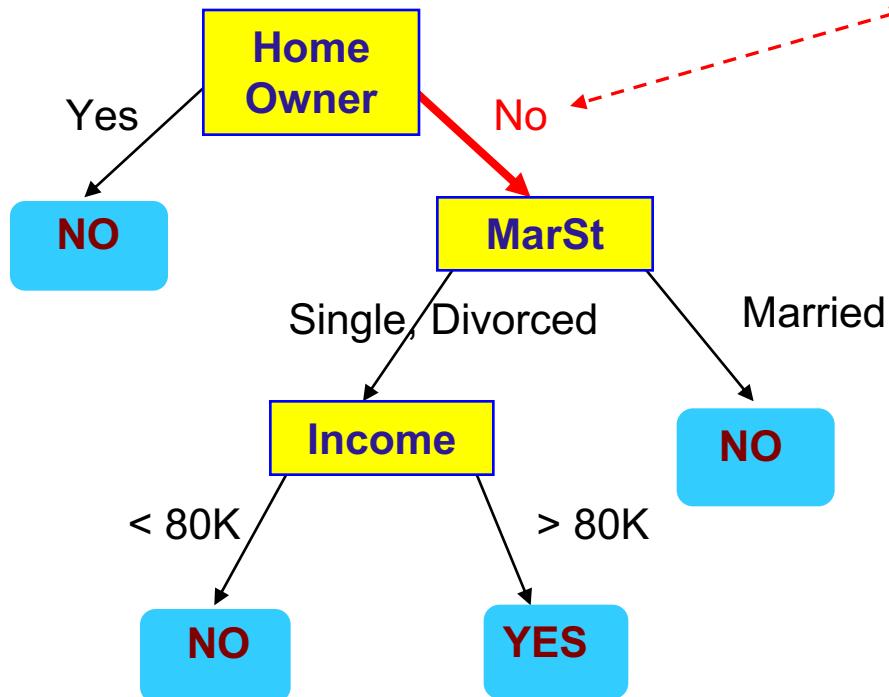
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Applica il Modello al Test Data Set

Test Data

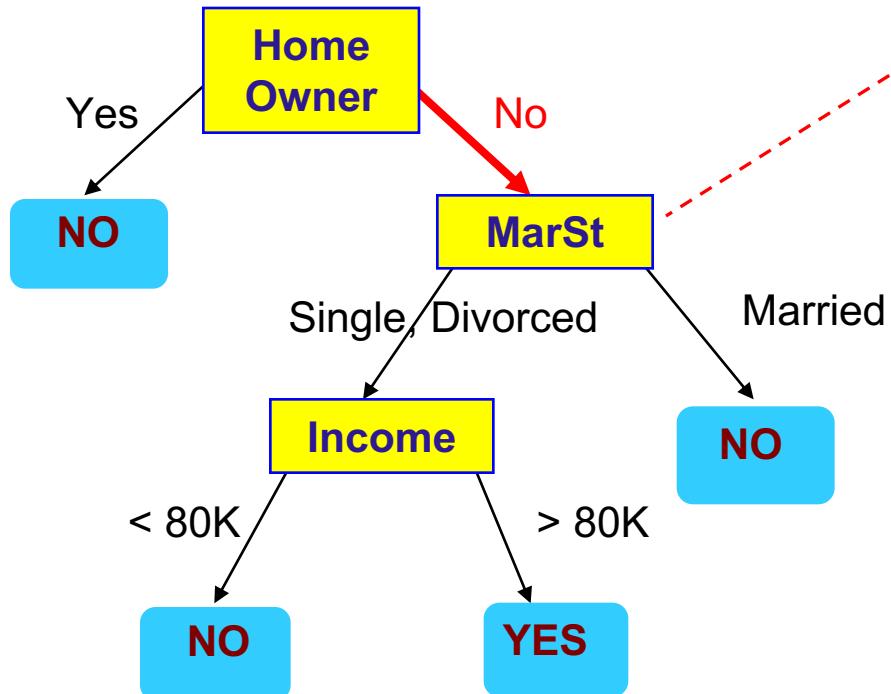
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Applica il Modello al Test Data Set

Test Data

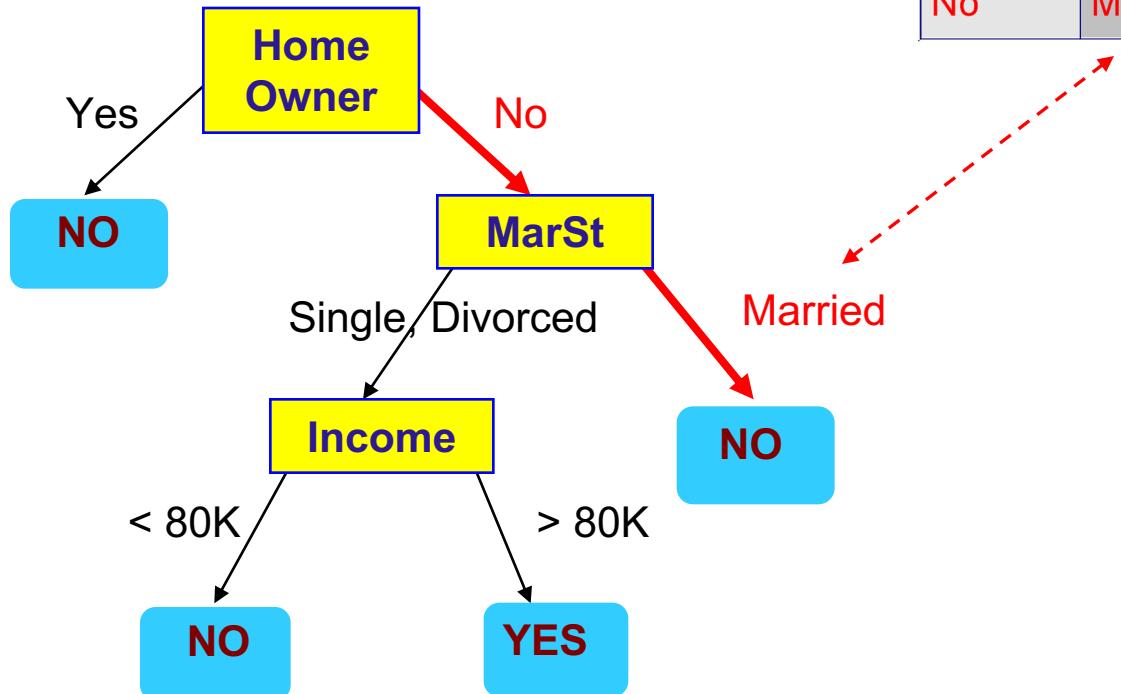
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Applica il Modello al Test Data Set

Test Data

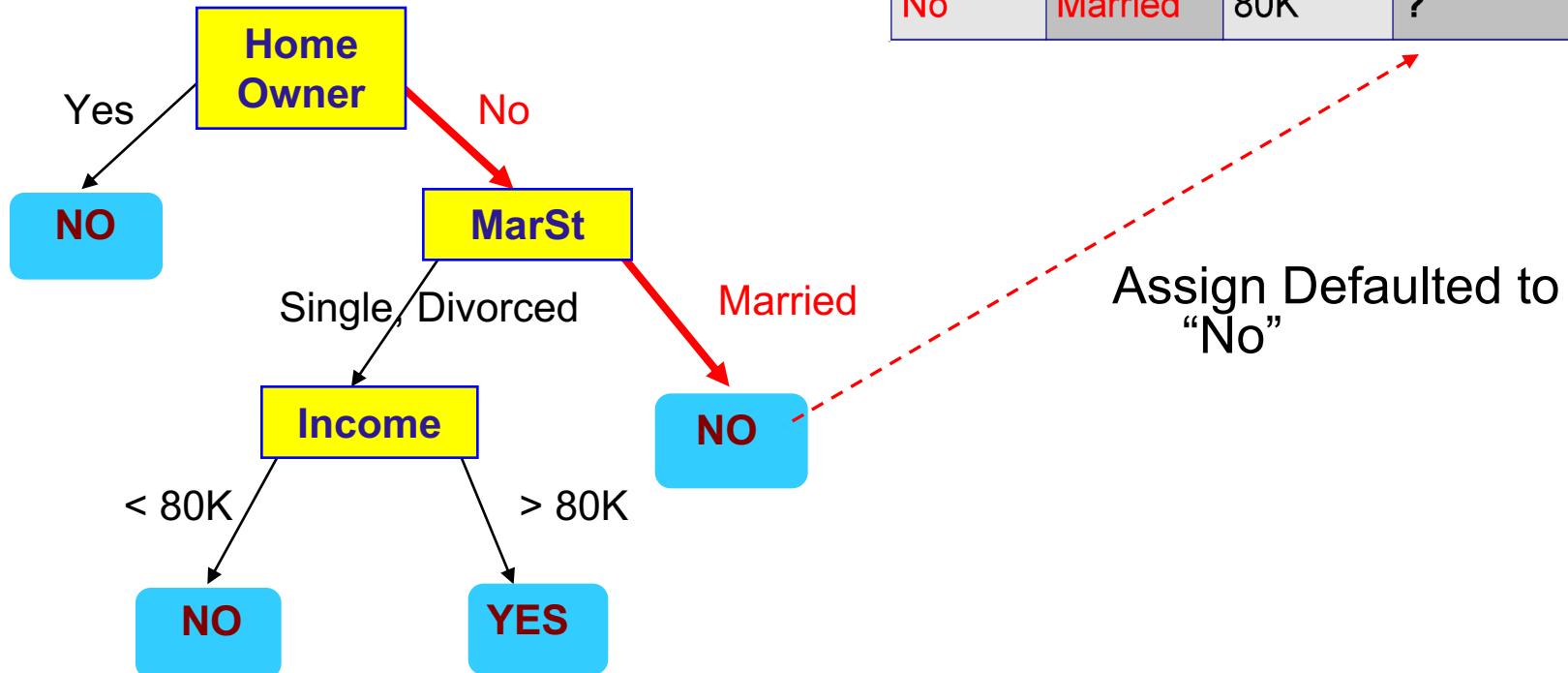
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Applica il Modello al Test Data Set

Test Data

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



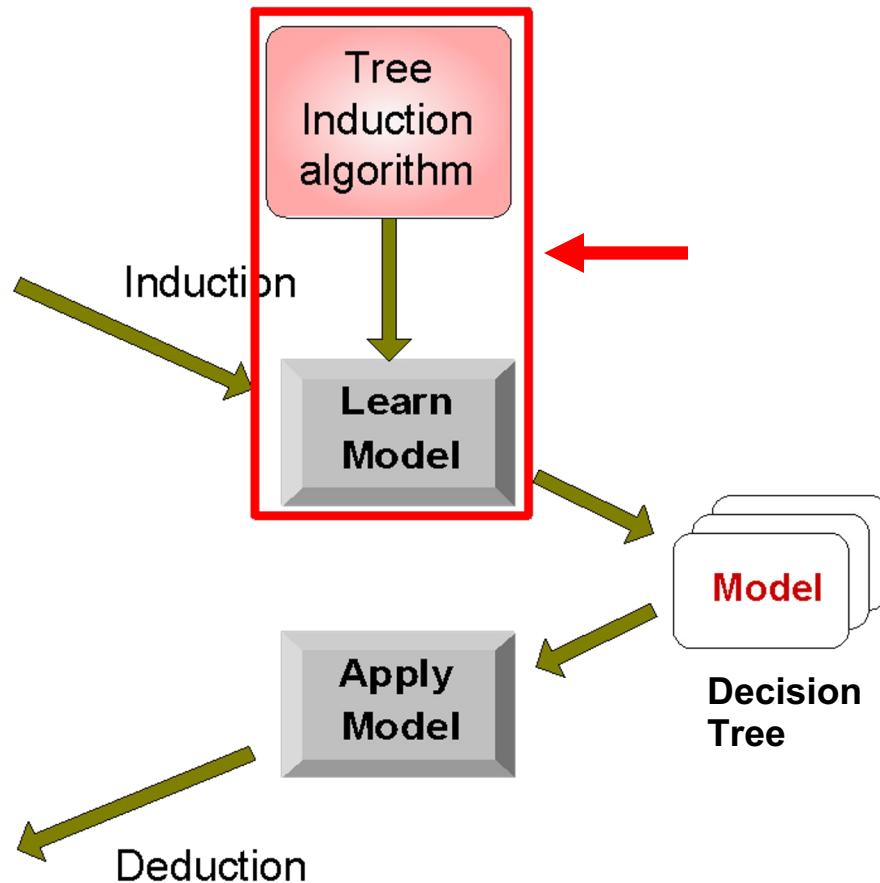
Decision Tree Classification Task

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



Induzione con Decision Tree

- Il numero di decision tree cresce esponenzialmente con il numero di attributi
- Gli algoritmi utilizzano generalmente tecniche greedy che fanno localmente la scelta “migliore”
- Sono a disposizione molti algoritmi:
 - Hunt's Algorithm
 - CART
 - ID3, C4.5
 - SLIQ, SPRINT
- Devono essere affrontati diversi problemi
 - Scelta del criterio di split
 - Scelta del criterio di stop
 - Underfitting
 - Overfitting
 - Frammentazione dei dati
 - Criterio di ricerca
 - Espressività
 - Replicazione degli alberi

Approccio generale (algoritmo di Hunt)

- Approccio ricorsivo che suddivide progressivamente un insieme di record D_t in insiemi di record via via più puri
- Sia D_t l'insieme dei record del training set corrispondenti al nodo t e $y_t = \{y_1, \dots, y_k\}$ le possibili etichette di classe

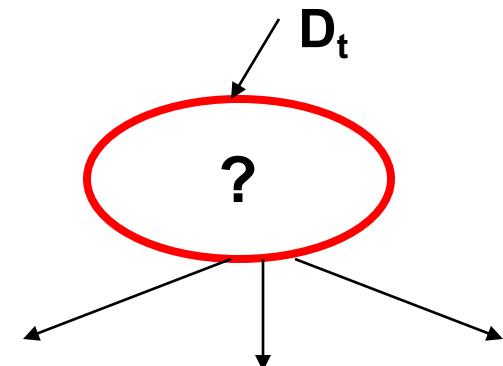
Algoritmo

- Se D_t contiene solo record appartenenti alla sola classe y_j , allora t è un nodo foglia con label y_j

Se D_t contiene record appartenenti a più classi, si scelga un **attributo e un criterio di split** per partizionare i record in più sottoinsiemi (non vuoti).

Si riapplichi ricorsivamente la procedura generale ai sottoinsiemi

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



Algoritmo di Hunt

Defaulted = No

(7,3)

(a)

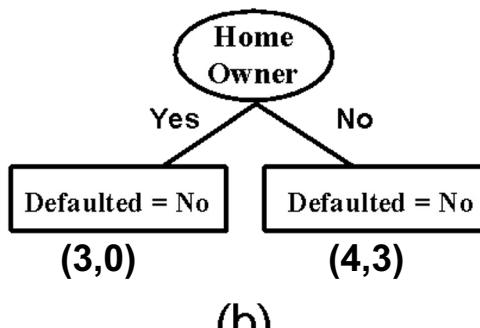
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Algoritmo di Hunt

Defaulted = No

(7,3)

(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

Algoritmo di Hunt

Defaulted = No

(7,3)

(a)

Home
Owner

Yes

No

Defaulted = No

(3,0)

Defaulted = No

(4,3)

(b)

Defaulted = No

Marital
Status

(3,0) Single,
Divorced

Defaulted = Yes

Defaulted = No

(1,3)

(3,0)

(c)

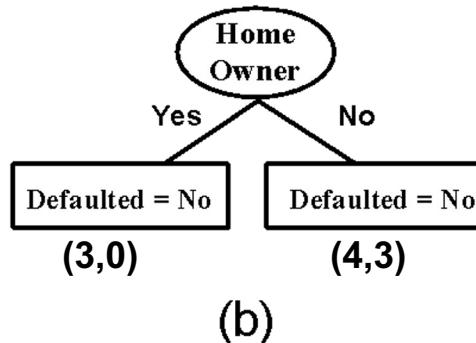
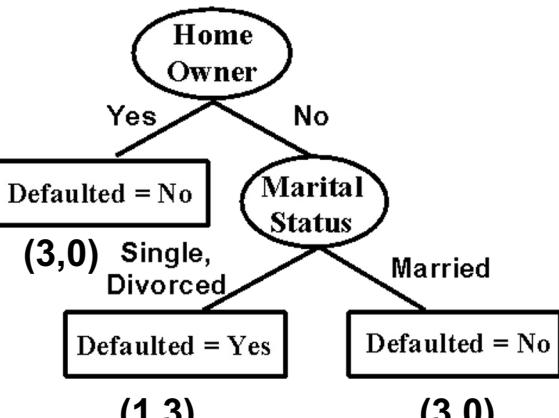
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Algoritmo di Hunt

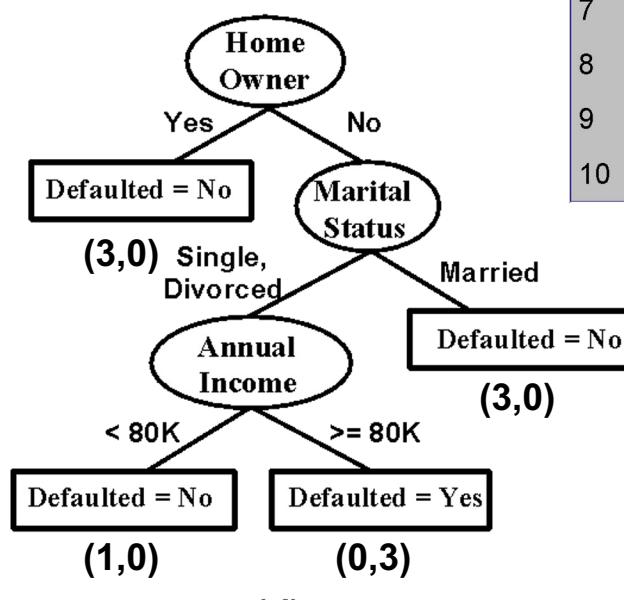
Defaulted = No

(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



Pseudo-codice

```
// Let E be the training set and F the attributes

Result = PostPrune(TreeGrowth(E,F)) ;

TreeGrowth(E,F)
    if StoppingCond(E,F) = TRUE then
        leaf = CreateNode();
        leaf.label = Classify(E);
        return leaf;
    else
        root = CreateNode();
        root.test_cond = FindBestSplit(E,F);
        let V = {v | v is a possible outcome of root.test_cond}
        for each v ∈ V do
            Ev = {e | root.test_cond(e)= v and e ∈ E}
            child = TreeGrowth(Ev,F);
            add child as descendants of root and
            label edge (root→child) as v
        return root;
    end;
```

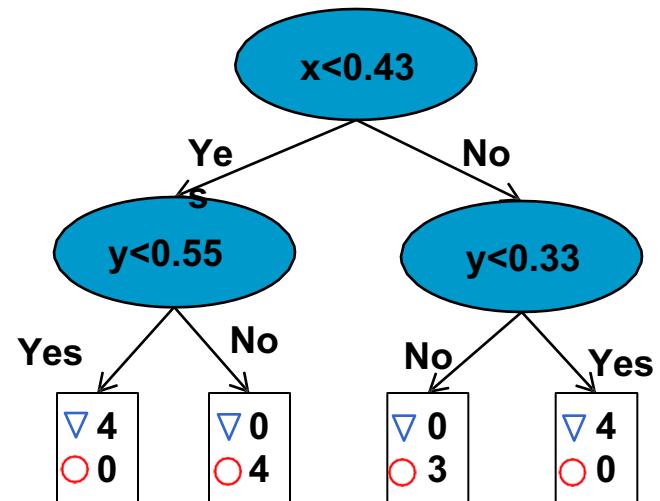
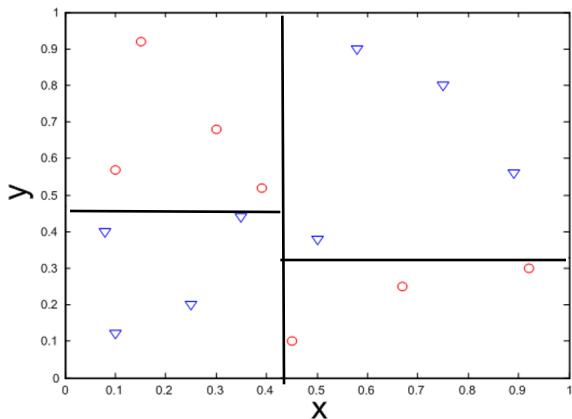
Alcune considerazioni...

- La ricerca di un albero di decisione ottimo è un problema NP-Completo, ma gli algoritmi euristici utilizzati sono molto efficienti
 - La maggior parte degli approcci eseguono una partizione ricorsiva top down basata su criteri greedy
- La classificazione utilizzando un albero decisionale è estremamente veloce e offre una facile interpretazione dei criteri
 - Il caso peggiore è $O(w)$ dove w è la profondità dell'albero
- Gli alberi di decisione sono sufficientemente robusti rispetto alla presenza di attributi fortemente correlati
 - Uno dei due attributi non sarà considerato
 - E' anche possibile cercare di scartare uno degli attributi in fase di preprocessing mediante opportune tecniche di feature selection

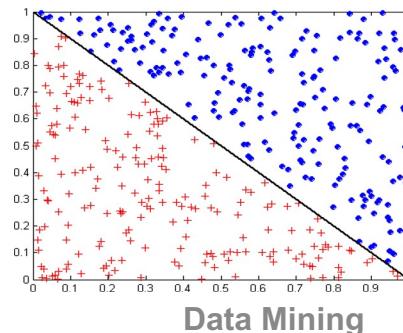
Alcune considerazioni...

- L'espressività degli alberi decisionali è limitata alla possibilità di effettuare partizionamenti dello spazio di ricerca con condizioni che coinvolgono un solo attributo per volta

Decision boundary paralleli agli assi



- Questa suddivisione non è ottenibile con alberi decisionali tradizionali



$$X - Y = 1$$

Elementi caratterizzanti

- A parte la logica di base per definire completamente un algoritmo per la costruzione di alberi decisionali è necessario definire:

La condizione di split

Il criterio che definisce lo split migliore

Il criterio per interrompere lo splitting

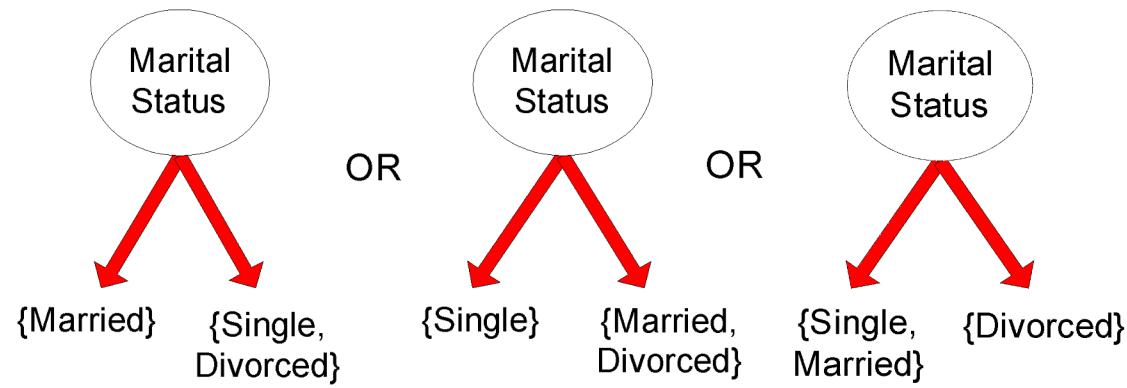
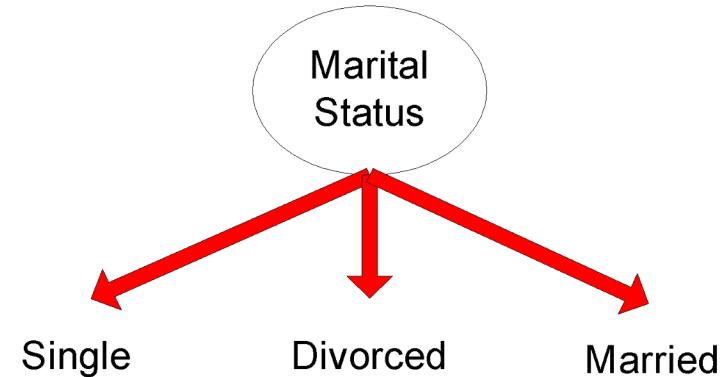
Le modalità per valutare la bontà di un albero decisionale

Condizione di split

- Dipende dal tipo di attributo
 - Binario
 - Nominale
 - Ordinale
 - Continuo
- Dipende dal numero di split applicabili ai valori dell'attributo
 - A 2-vie
 - A multi-vie

Split con attributi nominali

- **Split a più vie:**
- crea tante partizioni quanti sono i valori dell'attributo
- **Split binario:**
- crea due partizioni e richiede di suddividere i possibili valori dell'attributo in modo ottimale



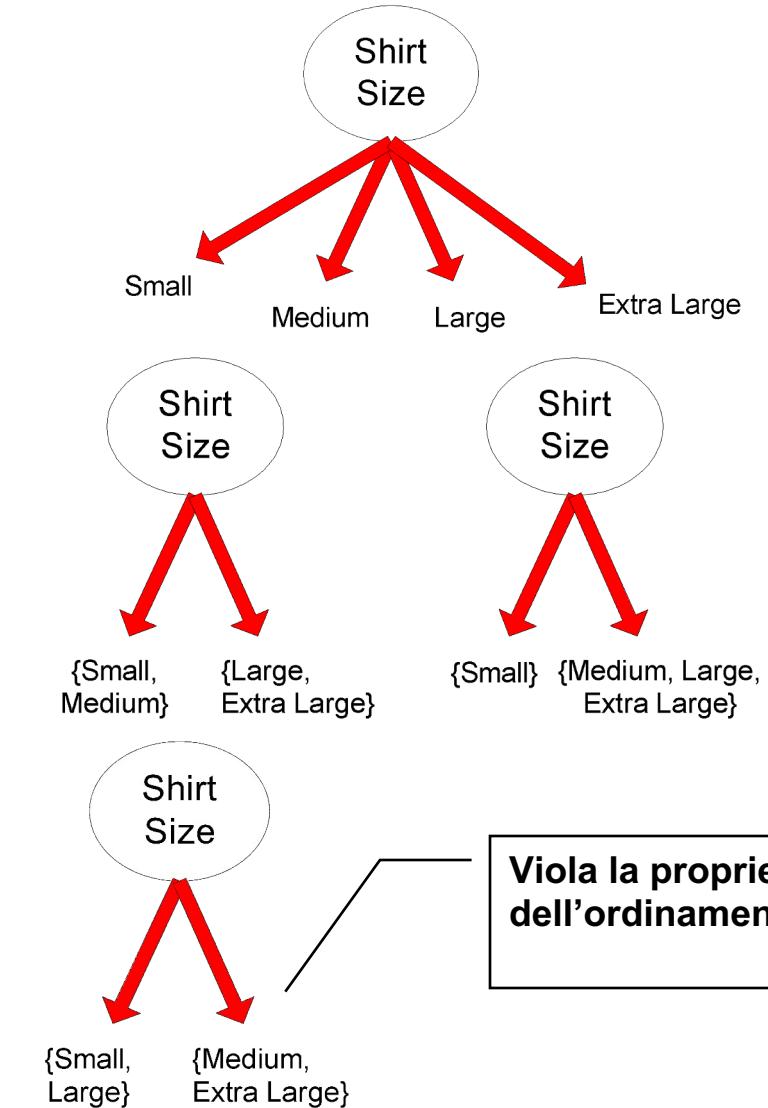
Split con attributi ordinali

- **Split a più vie:**

- crea tante partizioni quanti sono i valori dell'attributo

- **Split binario:**

- Divide i valori in due sotto-insiemi
- I sotto-insiemi preservano l'ordinamento dei valori



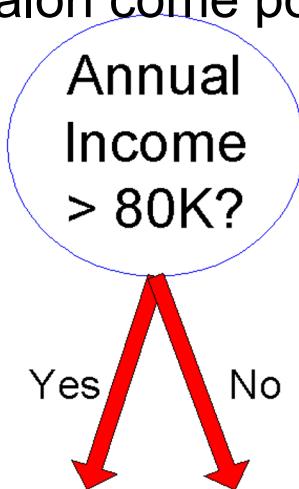
Split con attributi continui

- Diverse tecniche:
 - **Discretizzazione** per creare unattributo categorico ordinale

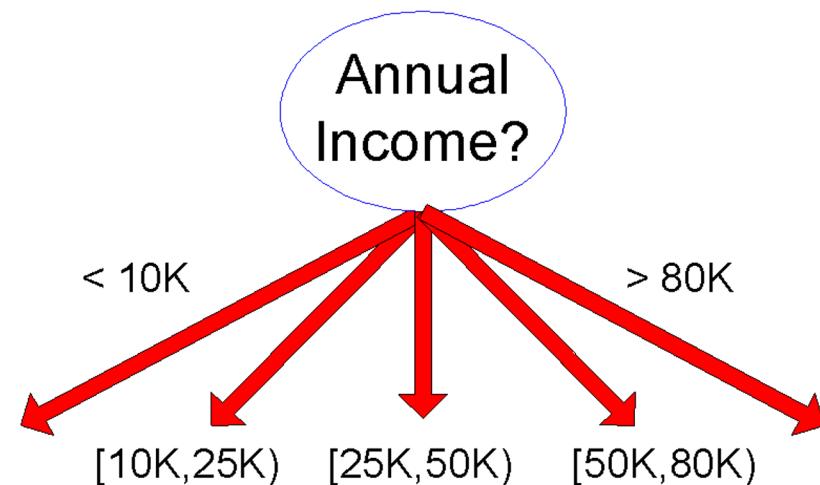
Gli intervalli possono essere definiti considerando intervalli di uguale dimensione, con frequenza uguale (percentili), clustering.
 - ◆ Dinamica – applicata a ciascun nodo
- **Partizione binaria**: $(A < v)$ or $(A \geq v)$
 - ◆ considera tutti i possibili splits e trova il “taglio” migliore
 - ◆ Può richiedere maggiore impegno computazionale.

Split con attributi continui

- **Split a più vie:** la condizione di split può essere espressa come un test di comparazione che ha per risultato più intervalli di valori. L'algoritmo deve considerare tutti i possibili intervalli di valori come possibili punti di split
- **Split a 2 vie:** la condizione di split può essere espressa come un test di comparazione con risultato binario. L'algoritmo deve considerare tutti i valori come possibili punti di split



(i) Binary split



(ii) Multi-way split

Elementi caratterizzanti

- A parte la logica di base per definire completamente un algoritmo per la costruzione di alberi decisionali è necessario definire:
 - La condizione di split
 - Il criterio che definisce lo split migliore**
 - Il criterio per interrompere lo splitting
 - Le modalità per valutare la bontà di un albero decisionale

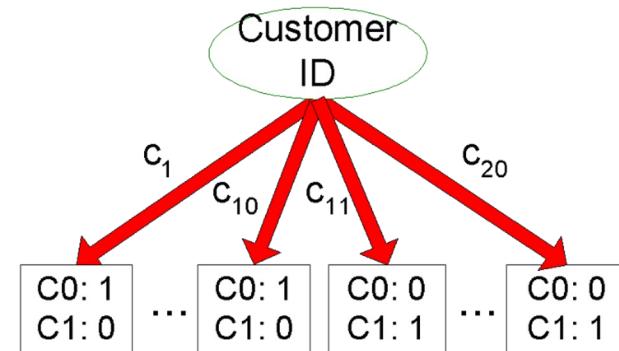
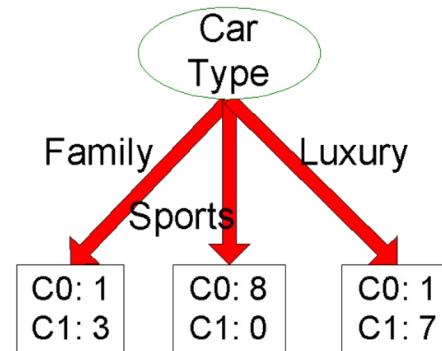
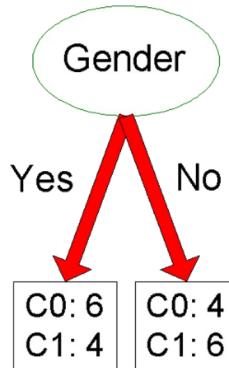
Come determinare lo split migliore

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

Come determinare lo split migliore

Prima dello split: 10 records di classe 0,
10 records di classe 1

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



Quale condizione di split è la migliore?

Come determinare lo split migliore

- Approccio Greedy :
 - Nodi con una distribuzione più “pura” sono preferiti
- Necessaria una misura della impurità dei nodi.

C0: 5
C1: 5

Elevato grado di impurità

C0: 9
C1: 1

Basso grado di impurità

Misure di impurità dei nodi

- Gini Index

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

- Entropy

$$Entropy(t) = -\sum_j p(j | t) \log p(j | t)$$

- Misclassification error

$$Error(t) = 1 - \max_i P(i | t)$$

- Impurità complessiva
(**meas()** è una delle misure introdotte)

$$Impurity_{split} = \sum_{i=1}^n \frac{m_i}{m} meas(i)$$

Trova il partizionamento migliore

1. Calcola l'impurità (P) prima dello splitting
2. Calcola l'impurità (M) dopo lo splitting
 - Calcola l'impurità per ciascun nodo figlio
 - M è l'impurità pesata dei figli
3. Scegli la condizione relativa all'attributo che produce il **guadagno maggiore**:

$$\text{Gain} = P - M$$

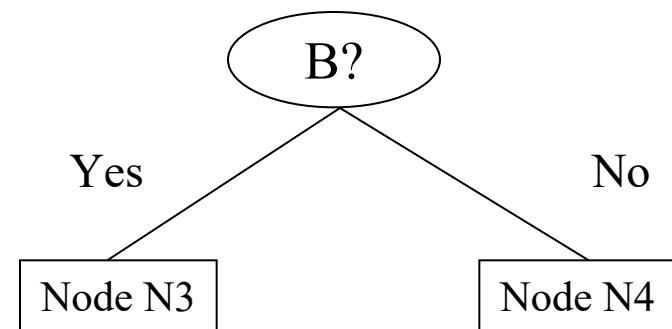
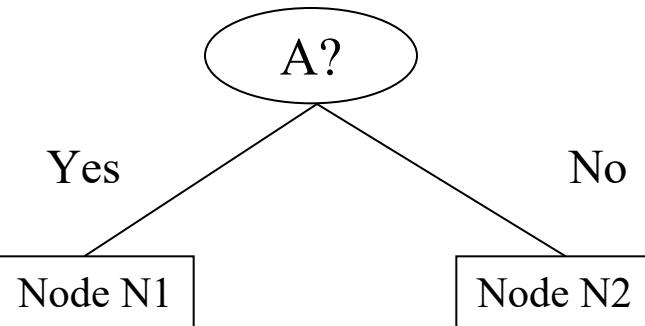
o equivalentemente, l'impurità più bassa dopo lo splitting.

Trova il partizionamento migliore

Prima del partizionamento:

C0	N00
C1	N01

→ P



C0	N10
C1	N11

C0	N20
C1	N21

C0	N30
C1	N31

C0	N40
C1	N41

M1

M2

M3

M4

M12

M34

$$\text{Gain} = P - M_{12} \quad \text{vs} \quad P - M_{34}$$

Misure di impurità (memento)

- Dato un nodo p con record appartenenti a k classi e un suo partizionamento in n nodi figli

M = numero di record nel padre p

m_i = numero di record nel figlio i

ATTENZIONE a non confondere il numero delle classi (k) e quello dei nodi figli (n)

- Gini index: usato in CART, SLIQ, SPRINT.**
- Entropia usato in ID3 e C4.5**
- Errore di classificazione**
- Impurità complessiva dello split è data dalla seguente formula dove $\text{meas}()$ è una delle misure introdotte**

$$GINI(i) = 1 - \sum_{j=1}^k [p(j|i)]^2$$

$$Entropy(i) = - \sum_{j=1}^k p(j|i) \cdot \log_2 p(j|i)$$

$$Error(i) = 1 - \max_{j \in K} p(j | i)$$

$$Impurity_{split} = \sum_{i=1}^n \frac{m_i}{m} \text{meas}(i)$$

Misure di impurità: GINI Index

- Gini Index per un dato nodo t :

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

dove:

- $p(j | t)$ è the frequenza relativa della classe j al nodo t;
- n_c = numero di classi.
- **Massimo (1 - 1/n_c)** quando i records sono equamente distribuiti tra tutte le classi. Ciò implica una informazione di interesse minima.
- **Minimo (0.0)** quando tutti I record appartengono ad una sola

Measura di Impurità: GINI Index

- Gini Index di un fissato nodo t :

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

dove:

$p(j | t)$ è la frequanza relativa della classe j al nodo t.

- Per uno split a due vie con frequenze **p** e **1 – p**:
 - ◆ $GINI = 1 - p^2 - (1 - p)^2 = 2p(1-p)$

Esempio

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	

Calcolo dell'indice di GINI di un nodo

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Gini} = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$\text{Gini} = 1 - (2/6)^2 - (4/6)^2 = 0.444$$

C1	3
C2	3

$$P(C1) = 3/6 \quad P(C2) = 3/6$$

$$\text{Gini} = 1 - (3/6)^2 - (3/6)^2 = 0.5$$

Calcolo dell'indice di GINI per più nodi

- Quando un nodo p è suddiviso in k partizioni (children)

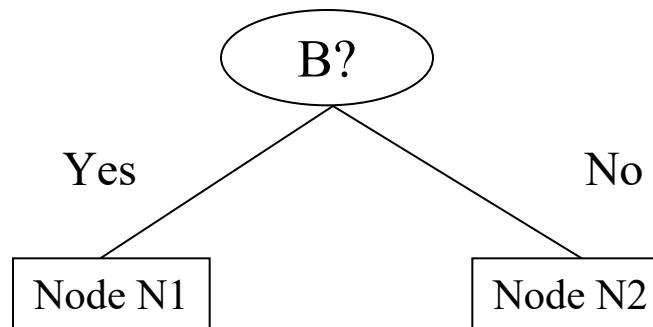
$$GINI_{split} = \sum_{i=1}^n \frac{m_i}{m} GINI(i)$$

dove, m_i = numero di record del figlio i,
 m = numero di record nel nodo padre p.

- **Scegli l'attributo che minimizza l'indice di GINI medio (pesato) dei nodi figli.**
- Gini index è usato in diversi algoritmi di alberi decisionali come CART, SLIQ, SPRINT

Attributi Binari: Calcolo GINI Index...

- Suddividi il nodo in due partizioni
- Effetto del partizionamento pesato:
 - Sono preferibili le partizioni più grandi e più pure.



Gini(N1)

$$\begin{aligned} &= 1 - (5/6)^2 - (1/6)^2 \\ &= 0.278 \end{aligned}$$

Gini(N2)

$$\begin{aligned} &= 1 - (2/6)^2 - (4/6)^2 \\ &= 0.444 \end{aligned}$$

	N1	N2
C1	5	2
C2	1	4
Gini=0.361		

	Parent
C1	7
C2	5
Gini = 0.486	

Weighted Gini of N1 N2

$$\begin{aligned} &= 6/12 * 0.278 + \\ &\quad 6/12 * 0.444 \\ &= 0.361 \end{aligned}$$

$$\text{Gain} = 0.486 - 0.361 = 0.125$$

Attributi non binari

- Possibili partizionamenti binari o a molte vie.

Multi-way split

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

Two-way split
(find best partition of values)

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

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

Quale partizionamento è il migliore?

Attributi non binari

- Possibili partizionamenti binari o a molte vie.

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	

Le misure di impurità dei nodi tendono a preferire partizionamenti in un elevato numero di partizioni, ciascuna delle quali è di piccole dimensioni, ma pura.

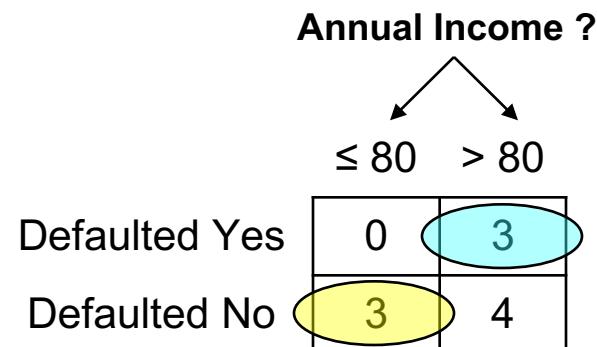
Attributi Continui: Calcolo Gini Index...

- Per ogni valore distinto, conta le occorrenze nel dataset
- Usa la matrice dei conteggi per prendere decisioni.

Attributi Continui: Calcolo Gini Index...

- Usa alberi decisionali binari basati su un valore v
- Diversi partizionamenti possibili:
 - Numero di possibili valori di split = = Numero di valori distinti
- Ciascun partizionamento (su v) ha una matrice di conteggio associata:
 - Vengono contati i record in ciascuna classe con A, a seconda che $A < v$ oppure $A \geq v$
- Metodo per scegliere il migliore v
 - Per ogni v , scansiona il dataset e calcola il relative Gini index
 - Computazionalmente Inefficiente! Ripete le operazioni.

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



Attributi Continui: Calcolo Gini Index...

- Calcolo efficiente: per ciascun attributo,
 - Ordina i valori
 - Scansiona linearmente questi valori, ad ogni passo aggiorna la matrice di conteggio e il GINI index
 - Scegli il valore che ha il minimo GINI index

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

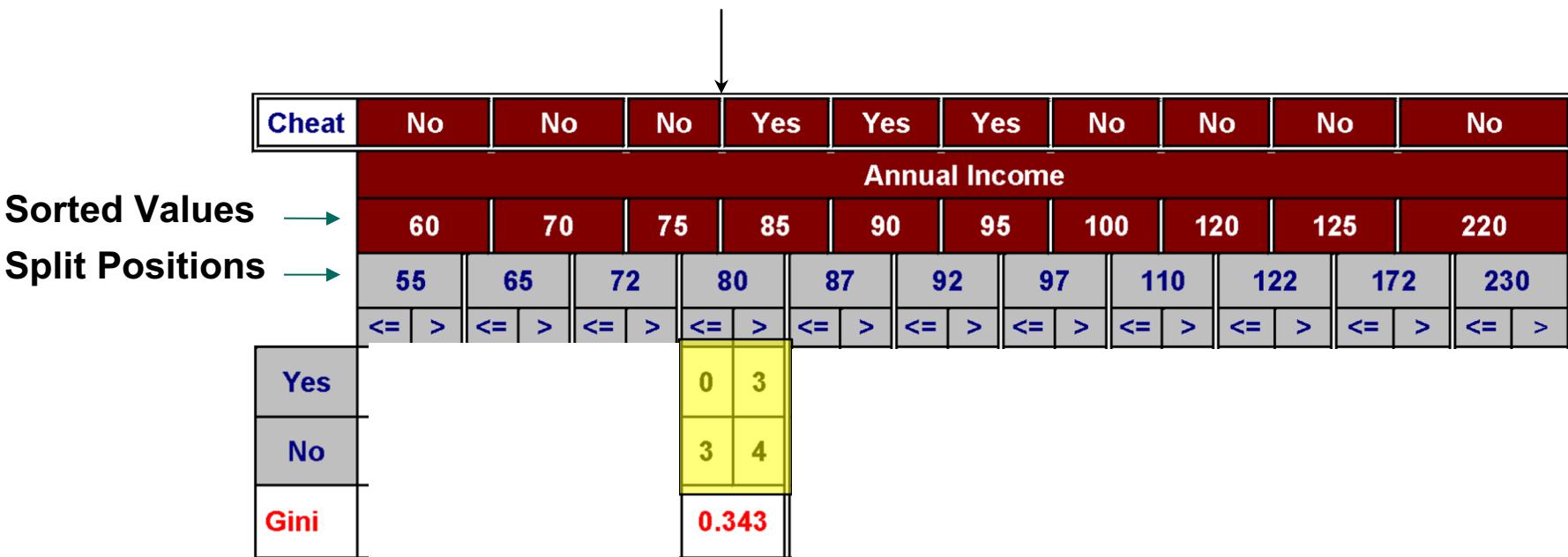
Attributi Continui: Calcolo Gini Index...

- Calcolo efficiente: per ciascun attributo,
 - Ordina i valori
 - Scansiona linearmente questi valori, ad ogni passo aggiorna la matrice di conteggio e il GINI index
 - Scegli il valore che ha il minimo 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
	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	

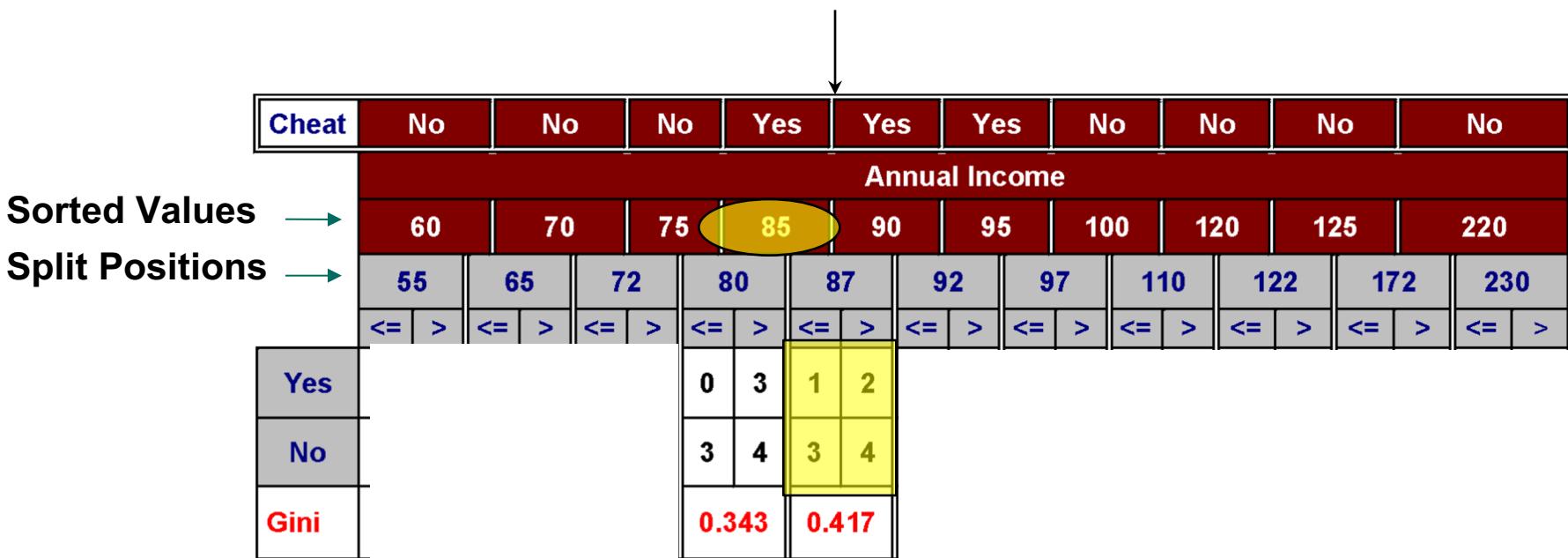
Attributi Continui: Calcolo Gini Index...

- Calcolo efficiente: per ciascun attributo,
 - Ordina i valori
 - Scansiona linearmente questi valori, ad ogni passo aggiorna la matrice di conteggio e il GINI index
 - Scegli il valore che ha il minimo GINI index



Attributi Continui: Calcolo Gini Index...

- Calcolo efficiente: per ciascun attributo,
 - Ordina i valori
 - Scansiona linearmente questi valori, ad ogni passo aggiorna la matrice di conteggio e il GINI index
 - Scegli il valore che ha il minimo GINI index



Attributi Continui: Calcolo Gini Index...

- Calcolo efficiente: per ciascun attributo,
 - Ordina i valori
 - Scansiona linearmente questi valori, ad ogni passo aggiorna la matrice di conteggio e il GINI index
 - Scegli il valore che ha il minimo 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	
No	0 7	1 6	2 5	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

Misura di Impurità: Entropia

- Entropia in un dato nodo t:

$$Entropy(t) = -\sum_j p(j | t) \log p(j | t)$$

dove:

- $p(j | t)$ è the frequenza relativa della classe j al nodo t;
 - n_c = numero di classi.
-
- ◆ **Massimo ($\log n_c$)** quando i record sono equamente distribuiti tra tutte le classi. Implica una informazione minima.
 - ◆ **Minimo (0.0)** quando tutti i record appartengono alla stessa classe. Implica una massima informazione
- Distribuzione uniforme. $massimo = -n_c \times \frac{1}{n_c} \log \frac{1}{n_c} = -\log \frac{1}{n_c} = \log n_c$
- Il calcolo dell'entropia è molto simile al calcolo del GINI index.

Calcolo dell'entropia in un singolo nodo

$$Entropy(t) = -\sum_j p(j | t) \log_2 p(j | t)$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Entropy} = -0 \log_2 0 - 1 \log_2 1 = -0 - 0 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$\text{Entropy} = -(1/6) \log_2 (1/6) - (5/6) \log_2 (5/6) = 0.65$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$\text{Entropy} = -(2/6) \log_2 (2/6) - (4/6) \log_2 (4/6) = 0.92$$

C1	3
C2	3

$$P(C1) = 3/6 \quad P(C2) = 3/6$$

$$\text{Entropy} = -(3/6) \log_2 (3/6) - (3/6) \log_2 (3/6) = 1$$

Split basato sul guadagno (GAIN)

- Utilizzando misure di impurità delle classi (come Gini), l'Entropia richiede di scegliere il valore di split che massimizza il “guadagno” in termini di riduzione dell'impurità delle classi dopo lo split.
- Il guadagno del partizionamento di un nodo p in n nodi figli è

$$GAIN_{split} = Entropy(p) - \sum_{i=1}^n \frac{m_i}{m} Entropy(i)$$

- **Selezionare il valore di split che massimizza GAIN** tende a determinare criteri di split che generano un numero molto elevato di classi molto pure e con pochi record.

Partizionare gli studenti in base alla loro matricola garantisce che tutte le classi (formate da un solo studente) siano totalmente pure!!

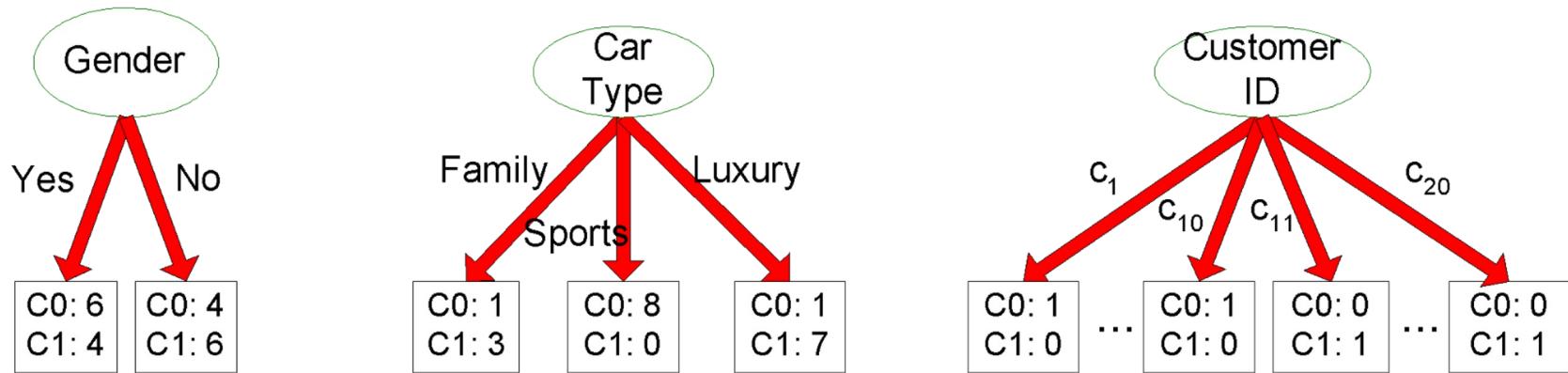
Usato negli algoritmi per alberi decisionali ID3 e C4.5.

Come determinare lo split migliore

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

Problemi con elevato numero di partizioni

- Le misure di impurità dei nodi tendono a preferire partizionamenti in un elevato numero di partizioni, ciascuna delle quali è piccola, ma pura.



- Customer ID ha il Massimo Guadagno poiché l'entropia di ciascun nodo è zero

GainRATIO

- Per evitare il problema della polverizzazione delle classi è preferibile massimizzare il **Gain Ratio**:

n = numero di nodi figli

m = numero di record nel padre p

m_i = numero di record nel figlio i

$$GainRATIO_{split} = \frac{Gain_{split}}{SplitINFO}$$

$$SplitINFO = - \sum_{i=1}^n \frac{m_i}{m} \log_2 \frac{m_i}{m}$$

Maggiore il numero dei figli, maggiore il valore di SplitINFO con una conseguente riduzione del GainRATIO

Per esempio, assumendo che ogni nodo figlio contenga lo stesso numero di record, $SplitInfo = \log n$.

C4.5 utilizza il criterio basato su SplitINFO

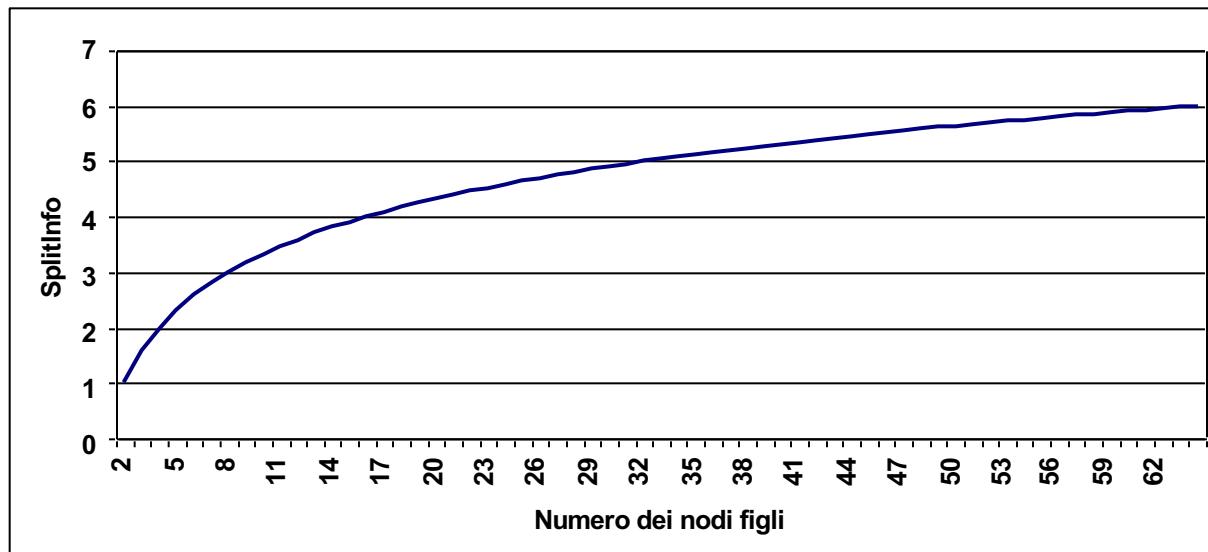
Split basato sulle INFO

- Per evitare il problema della polverizzazione delle classi è preferibile massimizzare il Gain Ratio:

$n = \text{da } 2 \text{ a } 64$

$m = 100$

$m_i = m/n$



Gain Ratio

- Gain Ratio:

$$GainRATIO_{split} = \frac{Gain_{split}}{SplitINFO}$$

$$SplitINFO = - \sum_{i=1}^n \frac{m_i}{m} \log_2 \frac{m_i}{m}$$

Nodo padre p (con n records) partizionato in n partizioni (**Gini(p) = 0,5**)
m_i è il numero di records nella partizione i-esima

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

$$\begin{aligned}Gain &= 0,338 \\ SplitINFO &= 1.522 \\ GainRATIO &= 0,222\end{aligned}$$

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

$$\begin{aligned}Gain &= 0,031 \\ SplitINFO &= 0.722 \\ GainRATIO &= 0,043\end{aligned}$$

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

$$\begin{aligned}Gain &= 0,333 \\ SplitINFO &= 0.971 \\ GainRATIO &= 0,343\end{aligned}$$

Gain Ratio

- Gain Ratio:

$$GainRATIO_{split} = \frac{Gain_{split}}{SplitINFO}$$

$$SplitINFO = - \sum_{i=1}^n \frac{m_i}{m} \log_2 \frac{m_i}{m}$$

Nodo padre p (con m record) partizionato in k partizioni (**Entropy(p) = 1**)
n_i è il numero di records nella partizione i-esima

CarType			
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
Entropy	0,380		

$$\begin{aligned}Gain &= 0,620 \\ SplitINFO &= 1.522 \\ GainRATIO &= 0,408\end{aligned}$$

CarType		
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Entropy	0,953	

$$\begin{aligned}Gain &= 0,047 \\ SplitINFO &= 0.722 \\ GainRATIO &= 0,065\end{aligned}$$

CarType		
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Entropy	0,390	

$$\begin{aligned}Gain &= 0,610 \\ SplitINFO &= 0.971 \\ GainRATIO &= 0,628\end{aligned}$$

Misure di Impurità: Classification Error

- Classification error in un nodo t :

$$Error(t) = 1 - \max_i P(i | t)$$

- **Massimo ($1 - 1/n_c$)** quando tutti i record sono equamente distribuiti tra le classi. Implica un contenuto informativo minimo.
- **Minimo (0.0)** quando tutti i record appartengono ad una sola classe. Implica un contenuto informativo massimo.

Calcolo dell'errore in un nodo

$$Error(t) = 1 - \max_i P(i | t)$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Error} = 1 - \max(0, 1) = 1 - 1 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$\text{Error} = 1 - \max(1/6, 5/6) = 1 - 5/6 = 1/6$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$\text{Error} = 1 - \max(2/6, 4/6) = 1 - 4/6 = 1/3$$

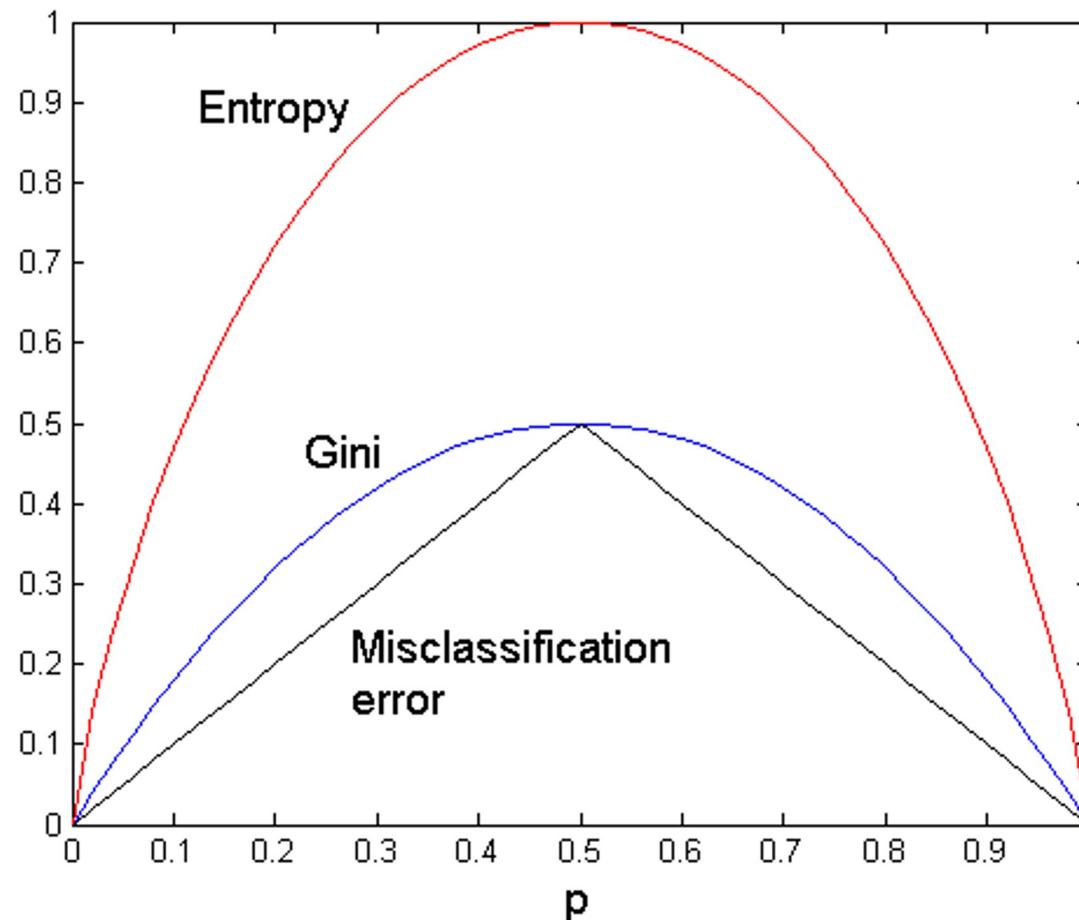
C1	3
C2	3

$$P(C1) = 3/6 \quad P(C2) = 3/6$$

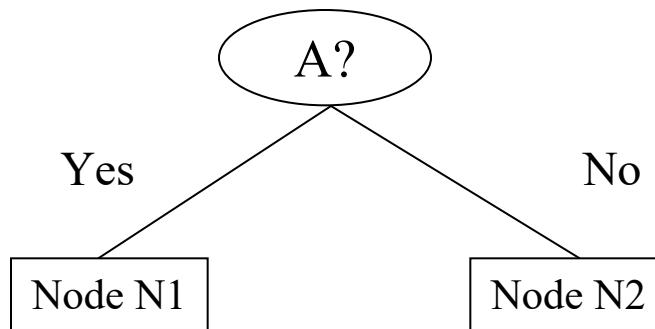
$$\text{Error} = 1 - \max(3/6, 4/6) = 1 - 3/6 = 1/2$$

Cofronto tra misure d'impurità

Per un problema con 2 classi:



Errore di classificazione vs Gini Index



	Parent
C1	7
C2	3
Gini = 0.42	

	N1	N2
C1	3	4
C2	0	3
Gini=0.342		

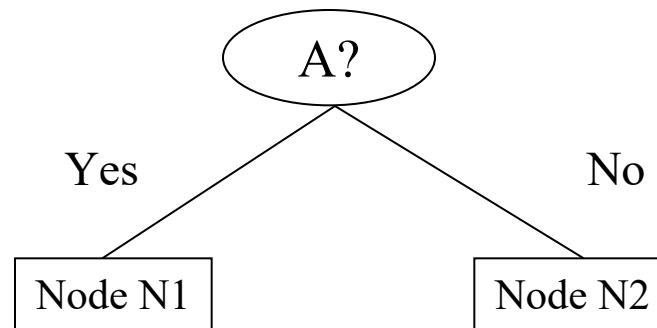
$$\begin{aligned}\text{Gini}(N1) &= 1 - (3/3)^2 - (0/3)^2 \\ &= 0\end{aligned}$$

$$\begin{aligned}\text{Gini}(N2) &= 1 - (4/7)^2 - (3/7)^2 \\ &= 0.489\end{aligned}$$

$$\begin{aligned}\text{Gini(Children)} &= 3/10 * 0 \\ &+ 7/10 * 0.489 \\ &= 0.342\end{aligned}$$

**Gini migliora
ma l'errore
resta invariato!!**

Errore di classificazione vs Gini Index



	Parent
C1	7
C2	3
Gini = 0.42	

Error = 0,3

	N1	N2
C1	3	4
C2	0	3
Gini=0.342		

	N1	N2
C1	3	4
C2	1	2
Gini=0.416		

L'errore di classificazione In tutti I tre casi è 0.3 !

Elementi caratterizzanti

- A parte la logica di base per definire completamente un algoritmo per la costruzione di alberi decisionali è necessario definire:

La condizione di split

Il criterio che definisce lo split migliore

Il criterio per interrompere lo splitting

Le modalità per valutare la bontà di un albero decisionale

Criteri di stop per l'induzione di alberi

- Interrompere lo split di un nodo quando tutti i suoi record appartengono alla stessa classe
- Interrompere lo split di un nodo quando tutti i suoi record hanno valori similiari su tutti gli attributi
 - La classificazione sarebbe poco significativa e dipendente da piccole fluttuazioni dei valori
- Interrompere lo split quando il numero dei record nel nodo è inferiore a una certa soglia (*data fragmentation*)
 - Il criterio selezionato non sarebbe statisticamente rilevante

Classificazione con alberi decisionali

- **Vantaggi:**
 - Basso costo
 - Estremamente veloce nel classificare record
 - Facile da interpretare per alberi di dimensione contenuta
 - Robusto rispetto al rumore (in modo particolare quando si utilizzano metodi per evitare l'overfitting)
 - Gestisce facilmente attributi ridondanti o irrilevanti (a meno che tali attributi interagiscano tra loro o con altri)
- **Svantaggi:**
 - Lo spazio dell'albero decisionale può crescere esponenzialmente. Gli approcci Greedy spesso non sono capaci di trovare l'albero migliore.
 - Non considerano le interazioni tra attributi
 - Ciascuna decisione coinvolge un solo attributo.

Alberi decisionali: Elementi caratterizzanti

- A parte la logica di base per definire completamente un algoritmo per la costruzione di alberi decisionali è necessario definire:
 - La condizione di split
 - Il criterio che definisce lo split migliore
 - Il criterio per interrompere lo splitting
- Le modalità per valutare la bontà di un albero decisionale**

Metriche per la valutazione del modello Model Overfitting

Errori di classificazione

- **Training error:** sono gli errori che si commettono sul training set
- **Test error:** sono gli errori che si commettono sul test set
- **Generalization error:** sono gli errori che si commettono sul data set reale (che non è noto).
- **Underfitting:** il modello è troppo semplice e non consente una buona classificazione né del training set, né del test set
- **Overfitting:** il modello è troppo complesso, consente un'ottima classificazione del training set, ma una pessima classificazione del test set

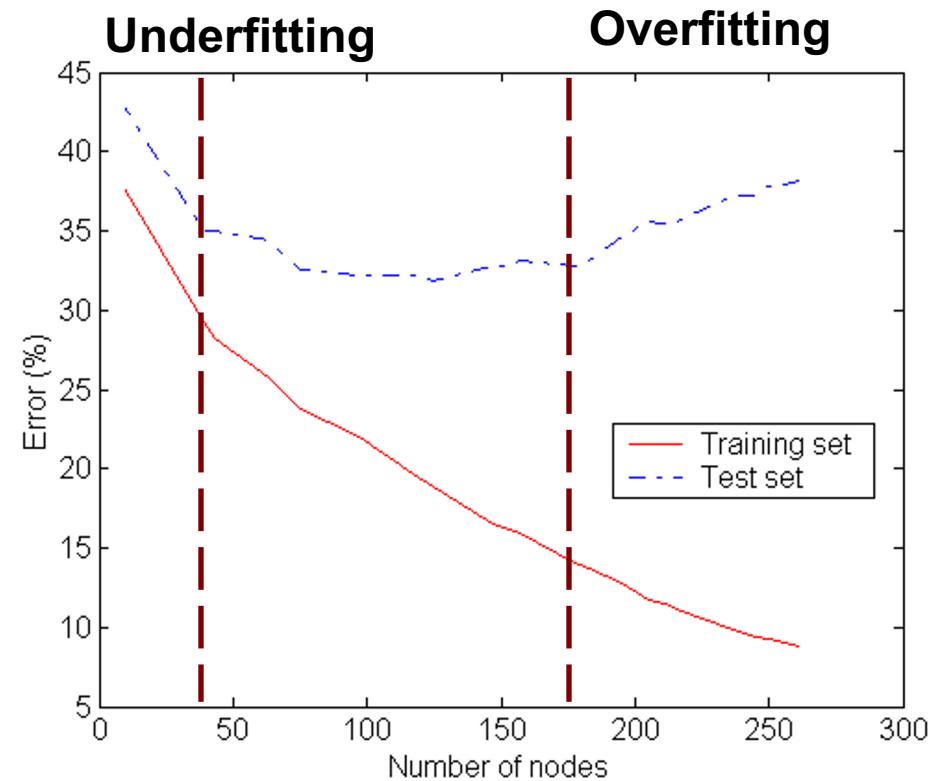
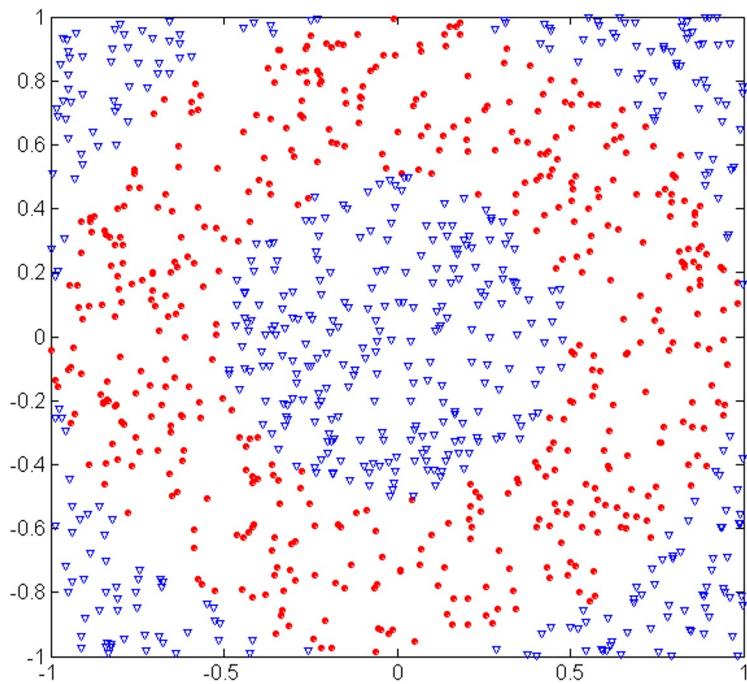
Il modello non riesce a generalizzare poiché è basato su peculiarità specifiche del training set che non si ritrovano nel test set (es. rumore presente nel training set)

Underfitting e Overfitting

500 cerchi e 500 triangoli

Punti circolari: $0.5 \leq \sqrt{x^2+y^2} \leq 1$

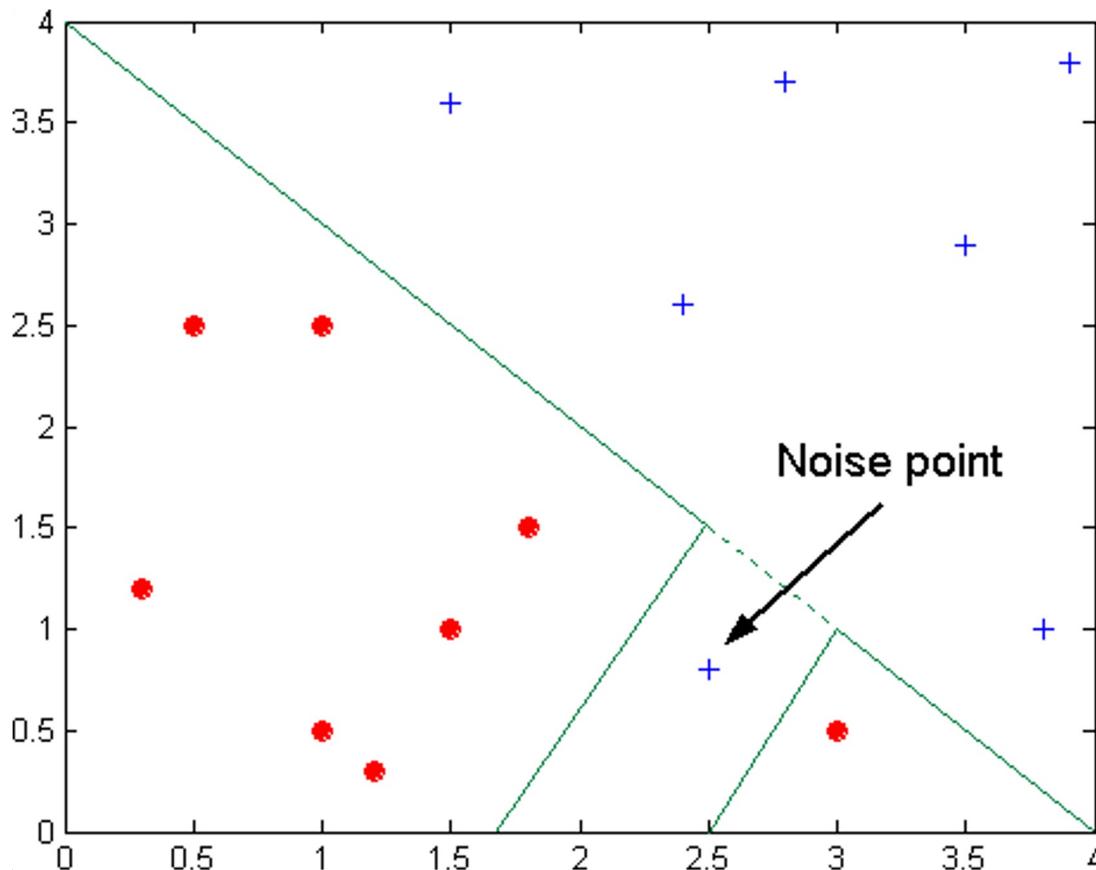
Punti triangolari: $\sqrt{x^2+y^2} < 0.5 \text{ o } \sqrt{x^2+y^2} > 1$



Cause dell'overfitting

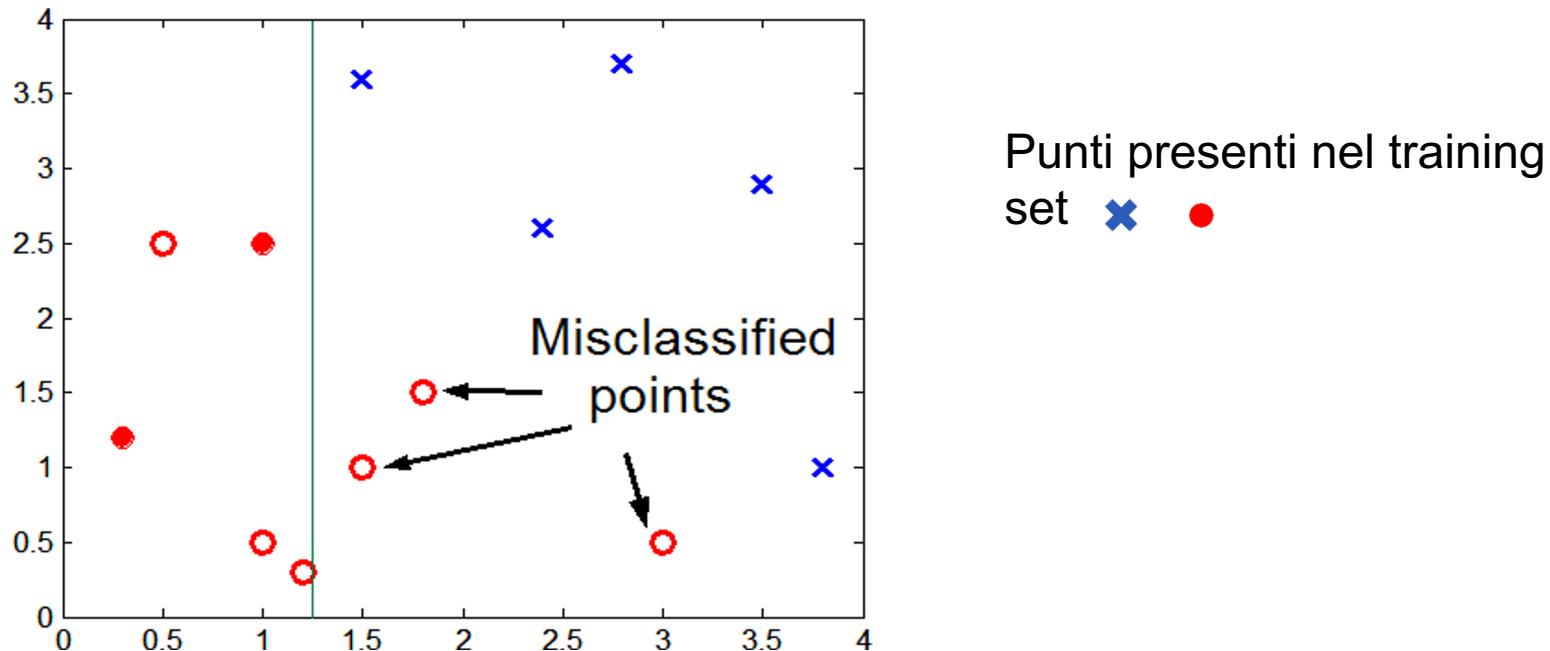
- Rumore
- Training Size Limitato
- Modello molto complesso
 - Multiple Comparison Procedure

Overfitting dovuto al rumore



- I confini delle aree sono distorte a causa del rumore

Overfitting per ridotto training set

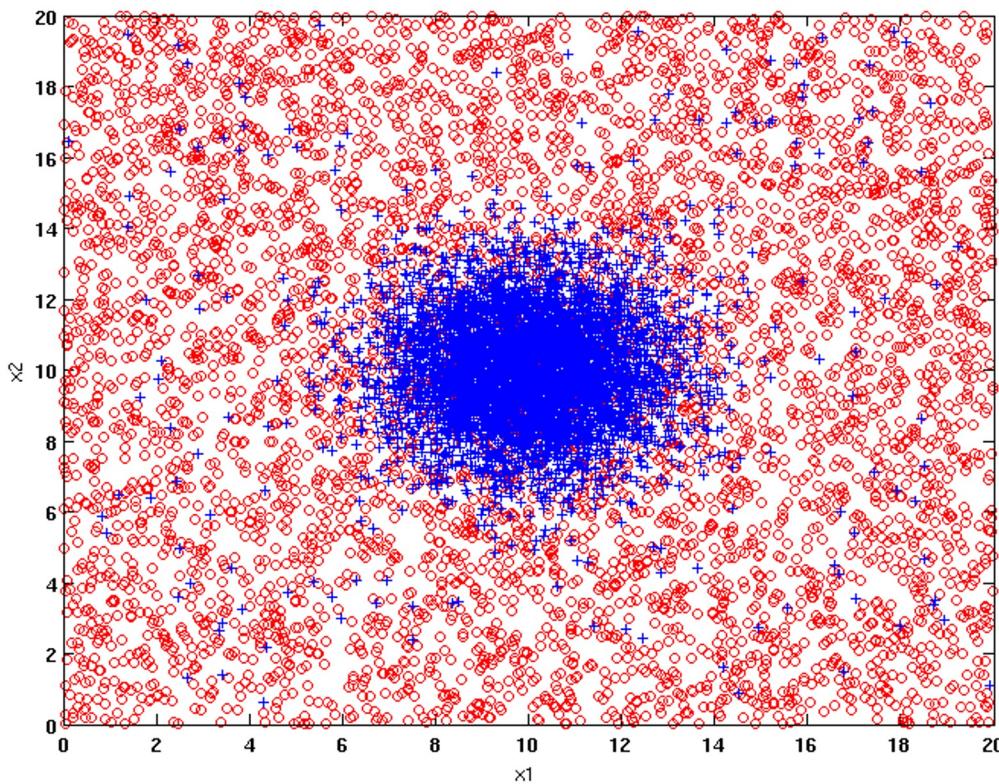


- La mancanza dei punti nella parte bassa del diagramma rende difficile individuare una corretta classificazione per quella porzione di regione

Note sull'overfitting

- L'overfitting determina alberi decisionali più complessi del necessario
- L'errore di classificazione compiuto sul training set non fornisce stime accurate circa il comportamento dell'albero su record sconosciuti
- Richiede nuove tecniche per stimare gli errori di generalizzazione

Esempio di Data Set



Two class problem:

+ : 5200 instances

- 5000 instances generated from a Gaussian centered at (10,10)

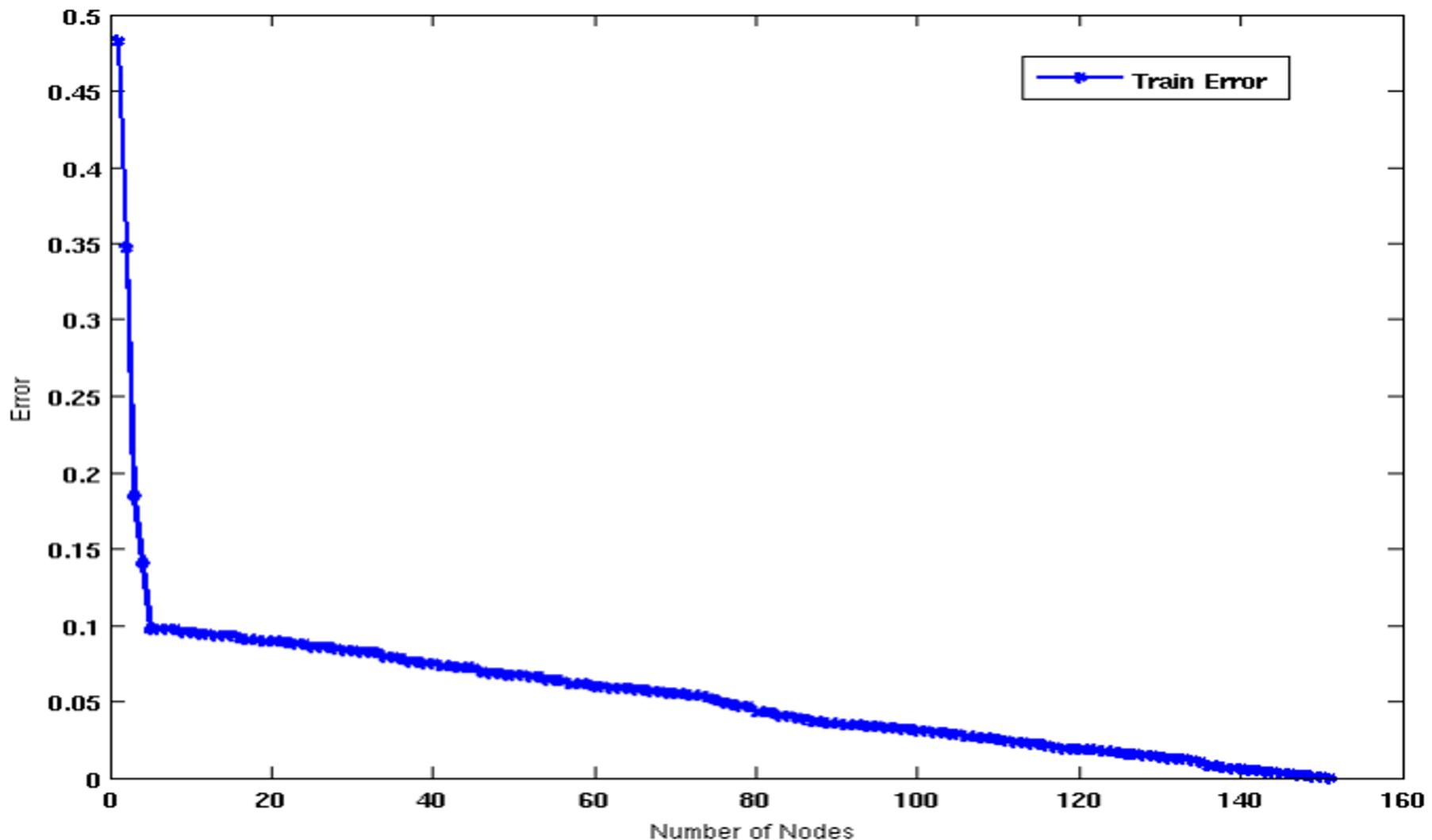
- 200 noisy instances added

o : 5200 instances

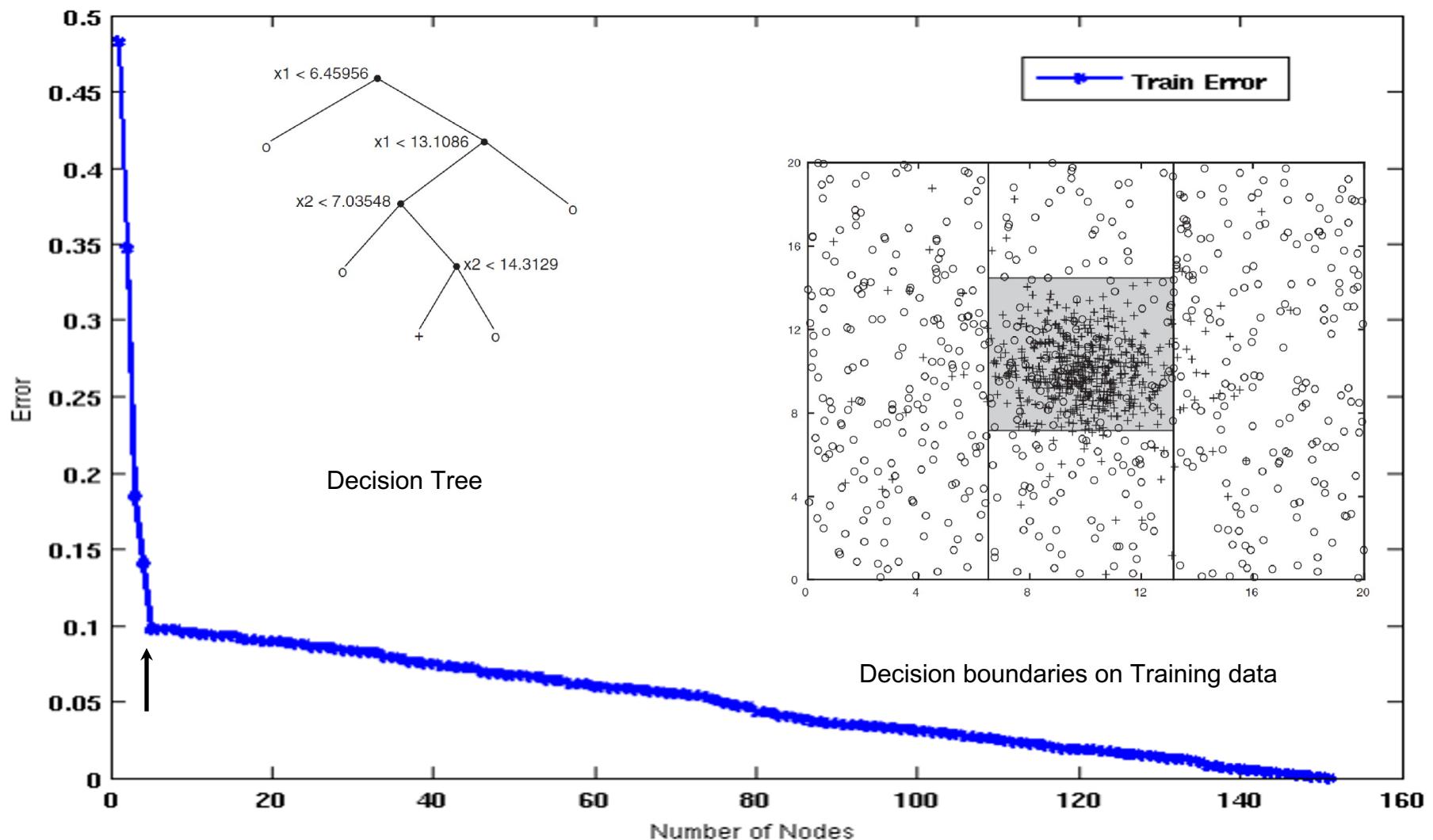
- Generated from a uniform distribution

10 % of the data used for training and 90% of the data used for testing

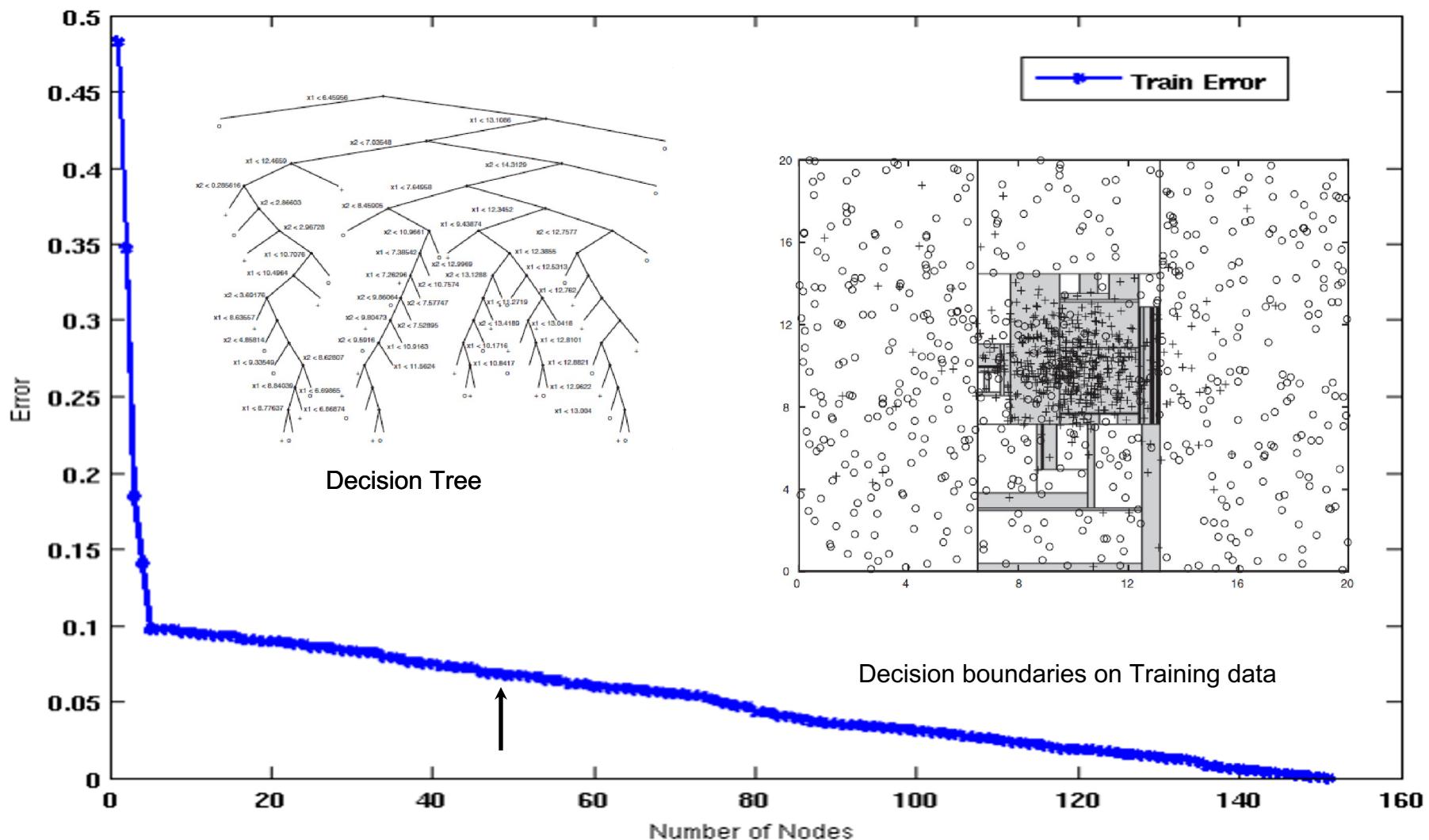
Increasing number of nodes in Decision Trees



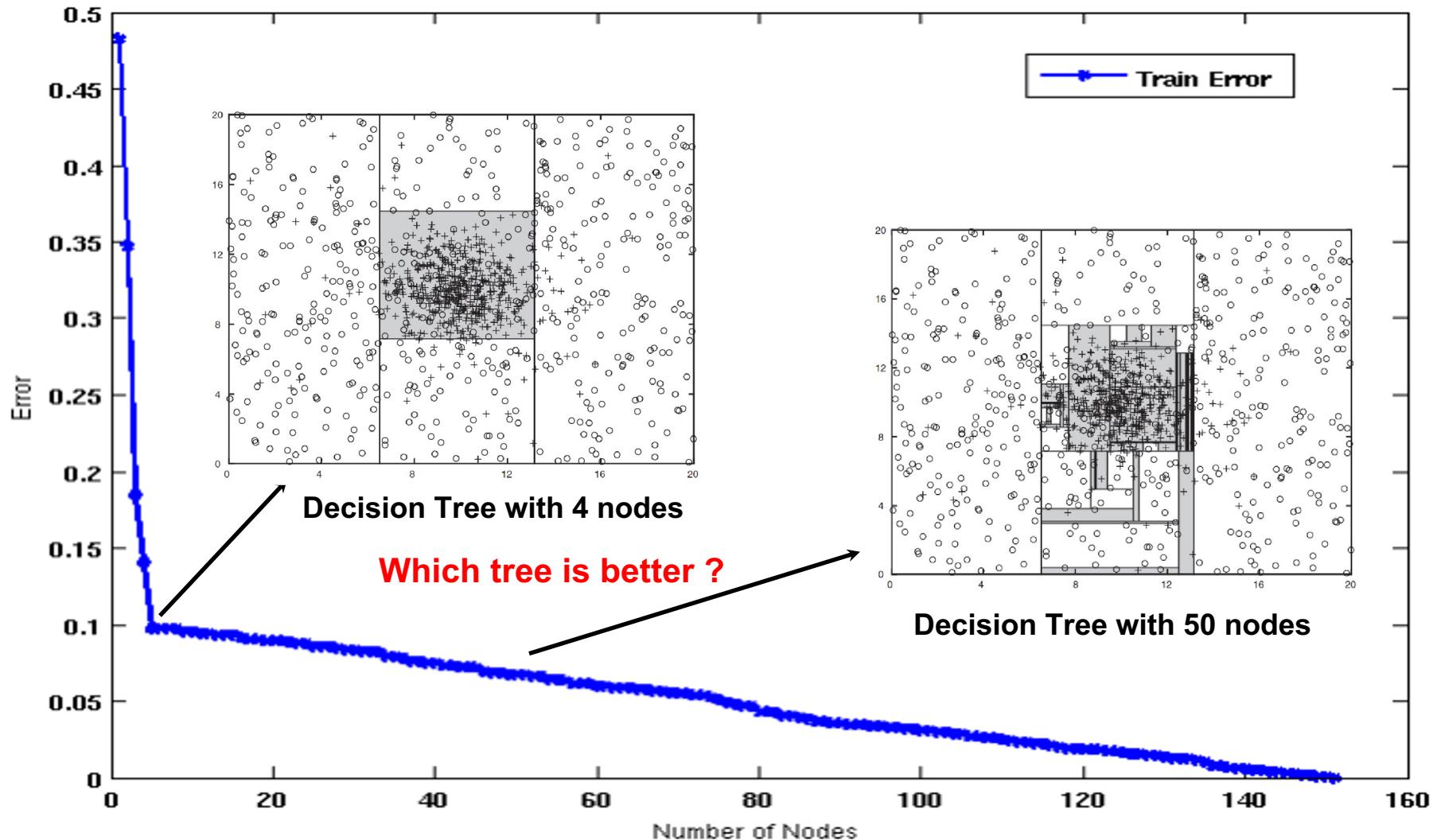
Decision Tree with 4 nodes



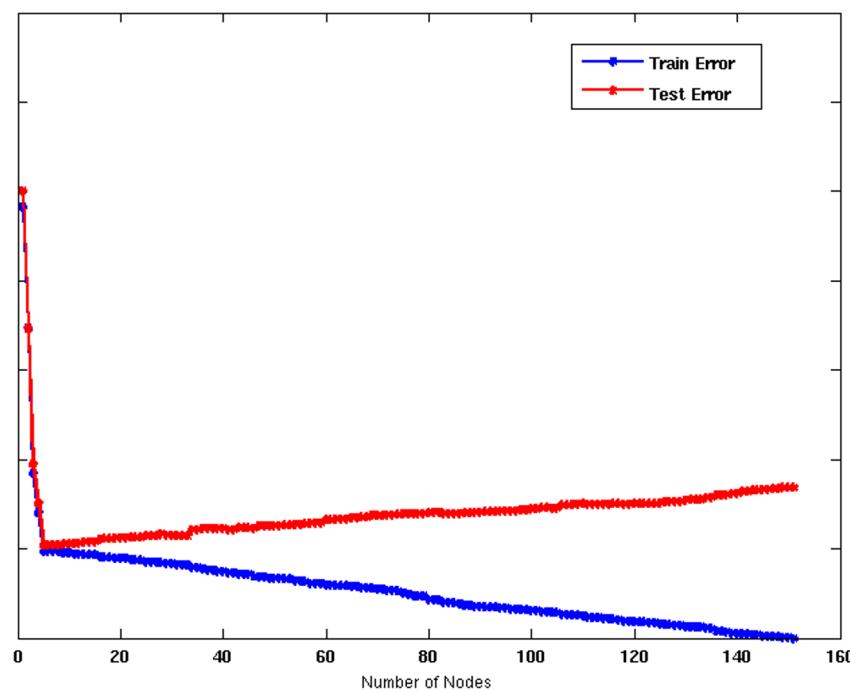
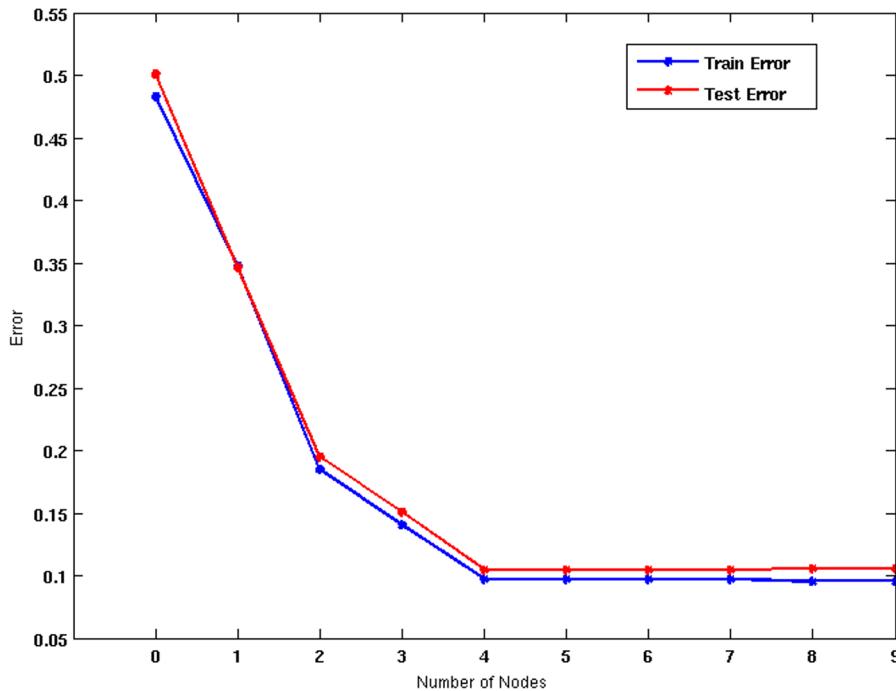
Decision Tree with 50 nodes



Which tree is better?



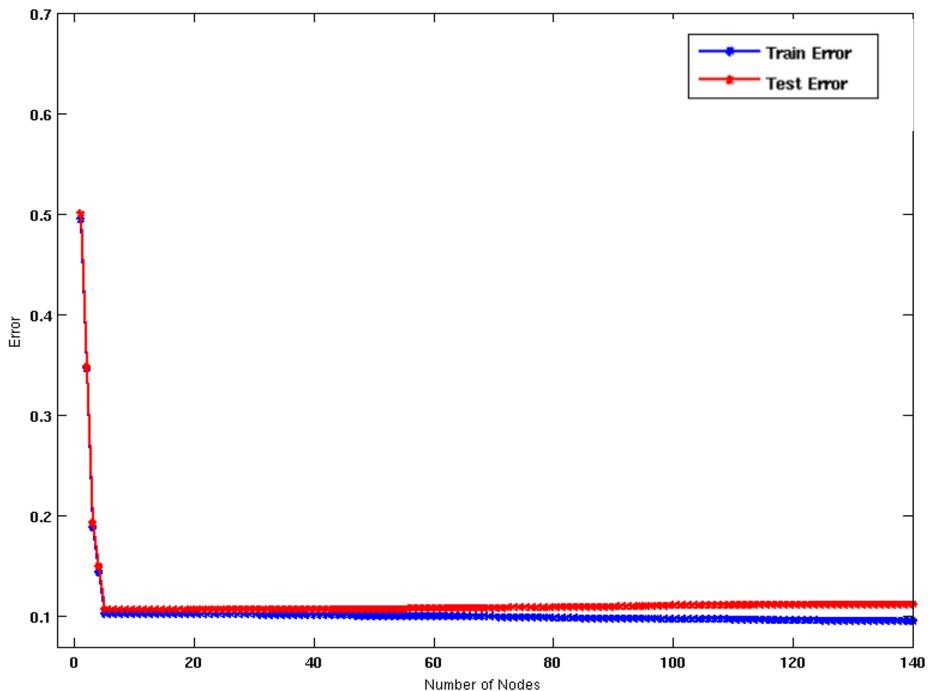
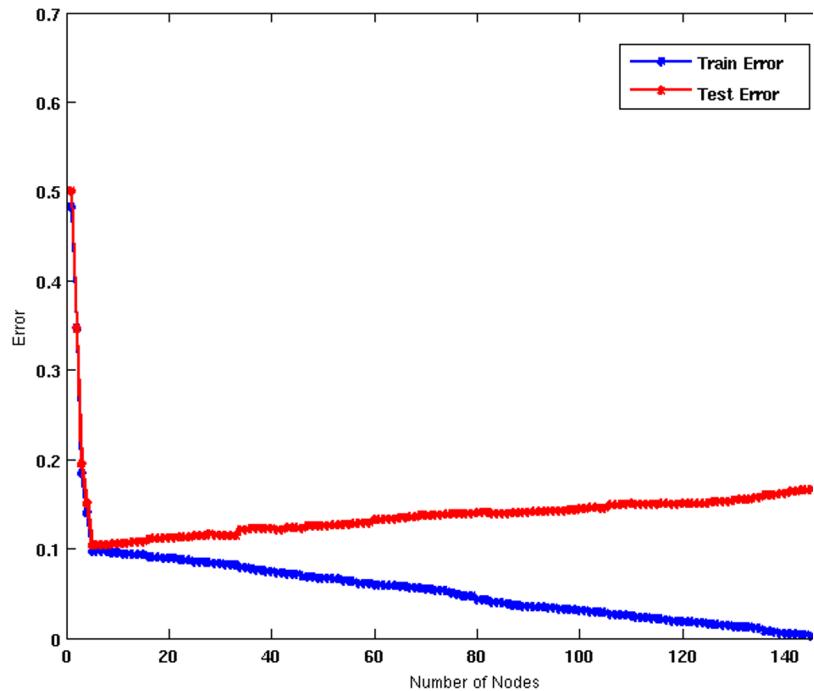
Model Overfitting



Underfitting: quando il modello è troppo semplice sia il training error che il test error sono elevati

Overfitting: quando il modello è troppo complesso il training error è basso mentre il test error è elevato

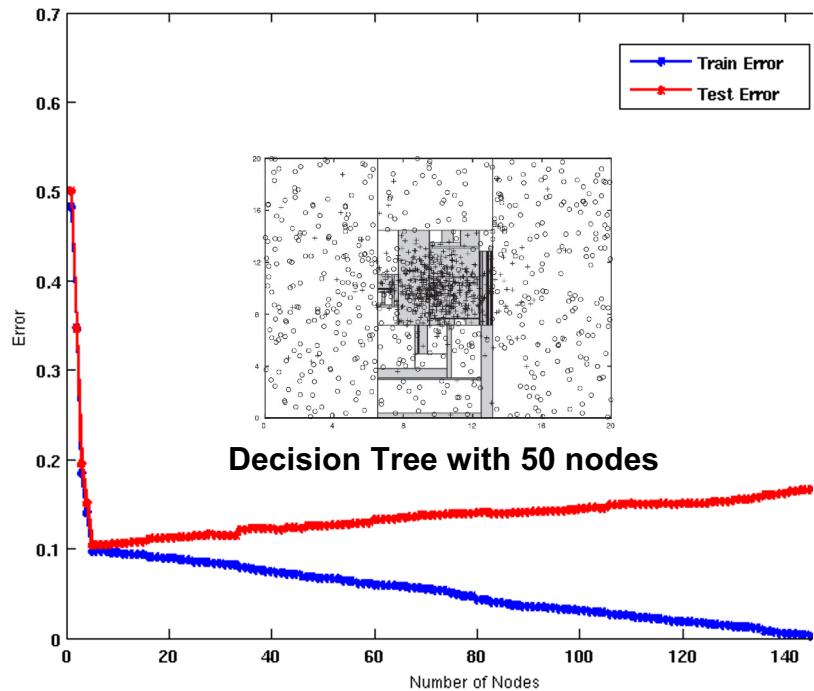
Model Overfitting



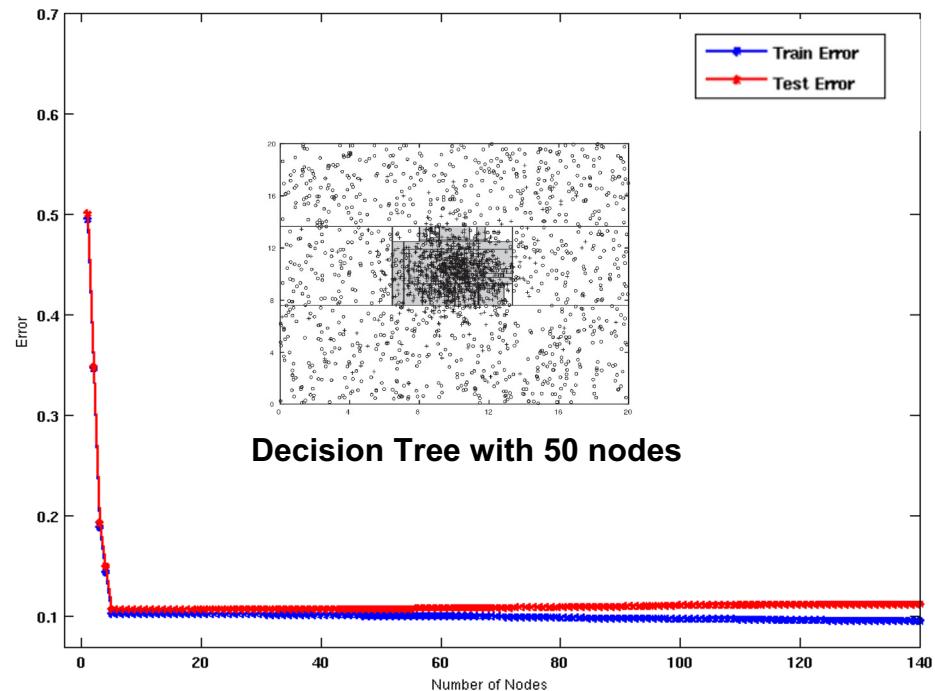
Con un training set di dimensione doppia

- Se il training set non è sufficientemente rappresentativo, il test error cresce e il training error decresce con l'aumento del numero di nodi
- L'aumento della dimensione del training set produce una riduzione della differenza tra training e test error (per un fissato numero di nodi)

Model Overfitting



Decision Tree with 50 nodes



Decision Tree with 50 nodes

Using twice the number of data instances

- Se il training set non è sufficientemente rappresentativo, il test error cresce e il training error decresce con l'aumento del numero di nodi
- L'aumento della dimensione del training set produce una riduzione della differenza tra training e test error (per un fissato numero di nodi)

Note sull'overfitting

- L'overfitting determina alberi decisionali più complessi del necessario
- L'errore di classificazione compiuto sul training set non fornisce stime accurate circa il comportamento dell'albero su record sconosciuti
- Richiede nuove metriche per misurare anche la complessità del modello
- Richiede nuove tecniche per stimare gli errori di generalizzazione

Complessità del Modello

- Selezione del modello eseguita durante la sua generazione (pre-pruning)
- L'obiettivo è garantire che il modello non sia troppo complesso (evitare l'overfitting)
- Necessario stimare il generalization error
 - Usando il Validation Set
 - Incorporando la Complessità del Modello
 - Stima Statistica di Bounds

Insieme di validazione

- Dividi i dati del training in due parti:
 - Training set:
 - ◆ usato per costruire il modello
 - Validation set:
 - ◆ Usato per stimare il generalization error
 - ◆ Nota: il validation set è diverso dal test set
- Problema:
 - Il training set è di dimensione più piccola

Stimare gli errori di generalizzazione

- Un albero di classificazione dovrebbe minimizzare l'errore sul data set reale, purtroppo in fase di costruzione si ha a disposizione solo il training set. Quindi l'errore sul data set reale deve essere stimato.

Training error: numero degli errori commessi sul training set ($e(t)$)

Generalization error: numero degli errori commessi sul data set reale ($e'(t)$)

- I metodi per stimare l'errore di generalizzazione sono:

Approccio ottimistico: $e'(t) = e(t)$

Approccio pessimistico

Minimum Description Length (MDL)

Utilizzo del test set: l'errore di generalizzazione è pari all'errore commesso sul test set.

- Normalmente il test set è ottenuto estraendo dall'iniziale training set 1/3 dei record
- Offre buoni risultati ma il rischio è quello di operare con un training set troppo piccolo

Occam's Razor

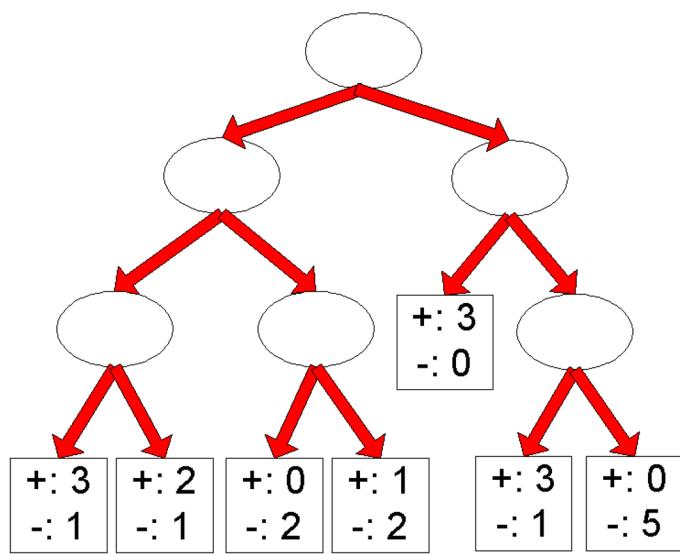
- Dati due modelli con errori di generalizzazione simili è preferibile quello più semplice

Per modelli complessi c'è maggiore probabilità che il livello di errore sia determinato da condizioni accidentali sui dati
- E' quindi utile considerare la complessità del modello quando si valuta la bontà di un albero decisionale
- Nota: principio metodologico espresso nel XIV secolo dal filoso e frate francescano inglese William of Ockham

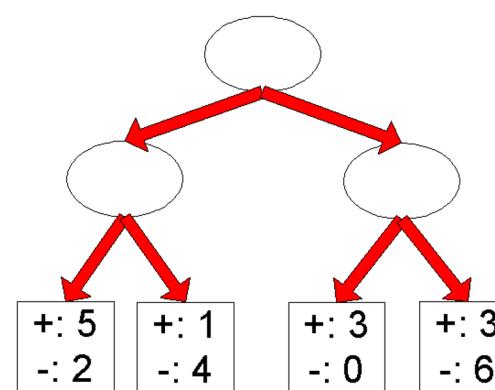
Complessità modello: approccio ottimistico

- Stima ottimistica:

- Usa il training error come stima ottimistica del generalization error



Decision Tree, T_L



Decision Tree, T_R

$$e(T_L) = 4/24$$

$$e(T_R) = 6/24$$

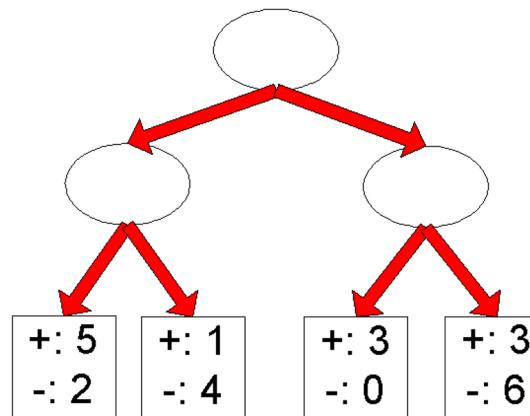
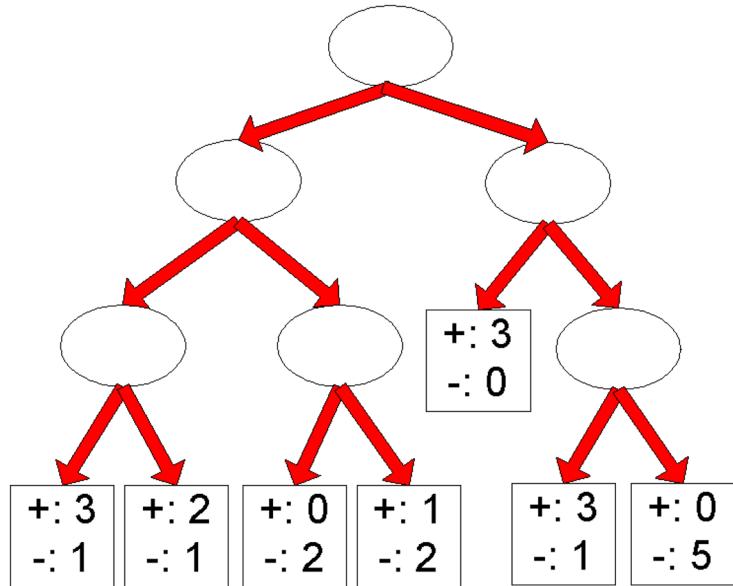
Complessità modello: approccio pessimistico

- **Stima pessimistica dell'errore** di un albero decisionale con k nodi foglia:

$$err_{gen}(T) = err(T) + \Omega \times \frac{k}{N_{train}}.$$

- $err(T)$: errore complessivo del training set
 - Ω : parametron che stima la penalità dell'aggiunta di un nodo foglia
 - k: numero di nodi foglia
 - N_{train} : numero totale di record
-
- Per alberi binari una penalità pari a 0.5 implica che un nodo debba sempre essere espanso nei due nodi figli se migliora la classificazione di almeno un record

Stima della Complessità: Esempio



$$e(T_L) = 4/24$$

$$e(T_R) = 6/24$$

$$\Omega = 1$$

$$e_{\text{gen}}(T_L) = 4/24 + 1 * 7/24 = 11/24 = 0.458$$

$$e_{\text{gen}}(T_R) = 6/24 + 1 * 4/24 = 10/24 = 0.417$$

Minimum Description Length (MDM)

- Dati due modelli si sceglie quello che minimizza il costo per descrivere una classificazione



x	y
X1	1
X2	0
X3	1
...	...
Xn	0

x	y
X1	?
X2	?
X3	?
...	...
Xn	?

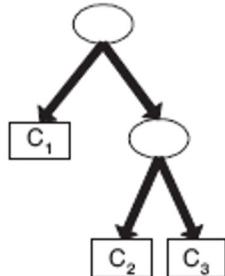


- Cost(Model,Data) = Cost(Data|Model) + $\alpha \times \text{Cost(Model)}$**
 - Cost è il numero di bit usati per la codifica;
 - Viene preferito il modello con costo minimo.
- Cost(Data|Model)** misura gli errori di classificazione sui dati (identificatore del record nel training set).
- Cost(Model)** misura la dimensione dell'albero, cioè la codifica binaria dei nodi interni (valore degli attributi) e dei nodi foglia (classe associata).

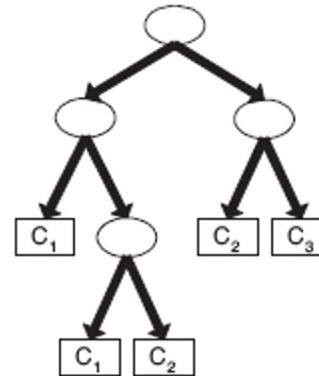
MDM: un esempio

- Dataset con n record descritti da 16 attributi binari e 3 valori di classe

T1
7 errori



T2
4 errori



Ogni nodo interno è modellato con l'ID dell'attributo usato: $\log_2(16)=4$ bit

Ogni foglia è modellata con l'ID della classe: $\log_2(3)=2$ bit

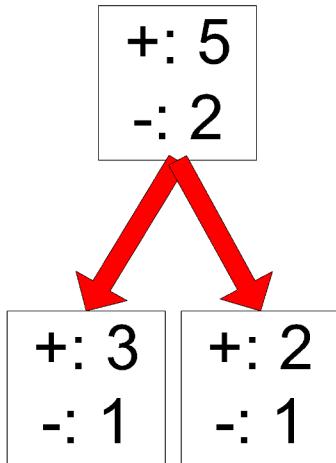
Ogni errore è modellato con la sua posizione nel training set
considerando n record: $\log_2(n)$

$$\text{Cost}(T1) = (4 \times 2 + 2 \times 3) + 7 \times \log_2(n) = 14 + 7 \times \log_2(n)$$

$$\text{Cost}(T2) = (4 \times 4 + 2 \times 5) + 4 \times \log_2(n) = 26 + 4 \times \log_2(n)$$

$$\text{Cost}(T1) < \text{Cost}(T2) \quad \text{se } n < 16$$

Stima statistica (NO)



$$e'(N, e, \alpha) = \frac{e + \frac{z_{\alpha/2}^2}{2N} + z_{\alpha/2} \sqrt{\frac{e(1-e)}{N} + \frac{z_{\alpha/2}^2}{4N^2}}}{1 + \frac{z_{\alpha/2}^2}{N}}$$

Prima del partizionamento

$$g: \quad e = 2/7, \quad e'(7, 2/7, 0.25) = 0.503$$

$$e'(T) = 7 \times 0.503 = 3.521$$

Dopo il partizionamento:

$$e(T_L) = 1/4, \quad e'(4, 1/4, 0.25) = 0.537$$

$$e(T_R) = 1/3, \quad e'(3, 1/3, 0.25) = 0.650$$

$$e'(T) = 4 \times 0.537 + 3 \times 0.650 = 4.098$$

Quindi, non partizionare!

Data Mining

Pruning: pre- e post-pruning

Costruzione del modello

- **Pre-Pruning (Early Stopping Rule)**

- Interrompe l'algoritmo prima che diventi un albero completamente sviluppato
- Tipiche condizioni di arresto per un nodo
 - Interrompi se tutte le istanze appartengono alla stessa classe
 - Interrompi se tutti i valori degli attributi sono uguali
- Condizioni più restrittive.
 - Interrompi se:
 - il numero di istanze è inferiore ad una soglia specificata
 - la distribuzione delle classi delle istanze è indipendente dalle caratteristiche dei dati disponibili (e.g., utilizzando il test χ^2)
 - l'espansione del nodo corrente non migliora le misure di impurità (ad es. Gini o guadagno informativo).

Come gestire l'Overfitting: post-pruning

- Esegui tutti gli split possibili
- Esamina in modo bottom-up i nodi del decision tree ottenuto
- Collassa un sottoalbero in un nodo foglia se questo permette di ridurre l'errore di generalizzazione (ossia sul validation set)

Scegli di collassare il sottoalbero che determina la massima riduzione di errore (N.B. scelta greedy)

- Le istanze nella nuova foglia possono essere etichettate
 - In base all'etichetta che compare più frequentemente nel sottoalbero
 - In base all'etichetta che compare più frequentemente nelle istanze del training set che appartengono al sottoalbero
- Il post-pruning è più efficace ma implica un maggior costo computazionale
 - Si basa sull'evidenza del risultato di un albero completo

Post-Pruning - Esempio

Class = Yes	20
Class = No	10
Error = 10/30	

PRIMA DEL PARTIZIONAMENTO

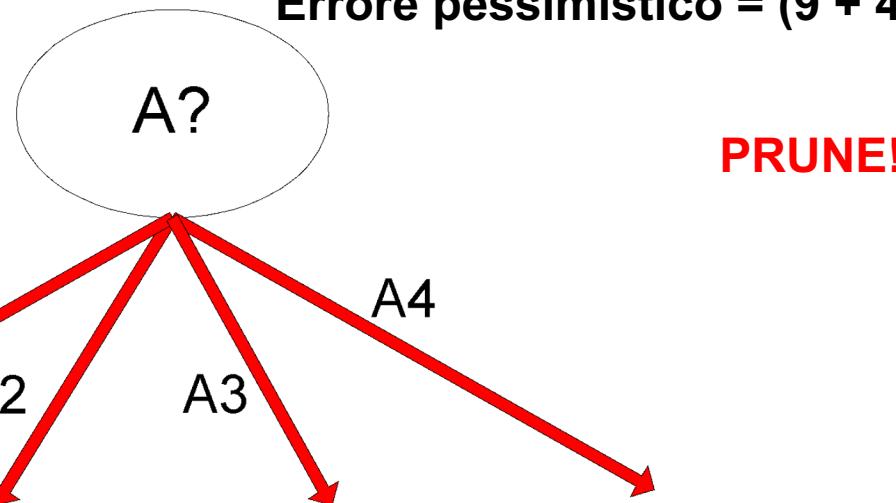
Training Error = 10/30

Errore pessimistico = $(10 + 0.5)/30 = 10.5/30$

DOPO IL PARTIZIONAMENTO

Training Error = 9/30

Errore pessimistico = $(9 + 4 \times 0.5)/30 = 11/30$



Class = Yes	8
Class = No	4

Class = Yes	3
Class = No	4

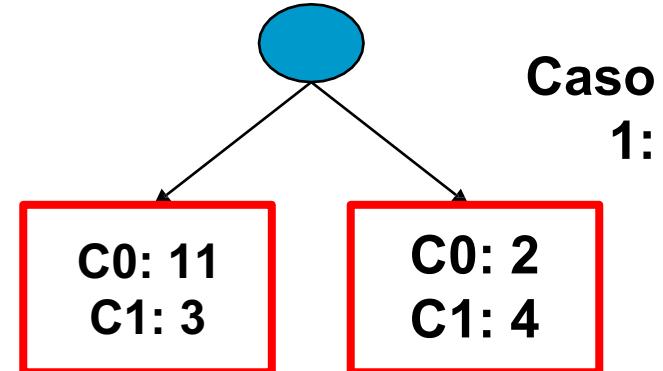
Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

Post-Pruning - Esempio

Errore ottimistico?

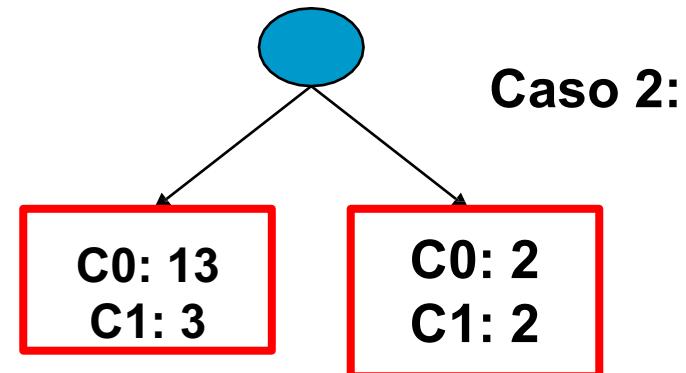
Non tagliare in nessuno dei casi



Errore pessimistico (penalità 0.5)?

Non tagliare nel caso 1, taglia nel caso 2

Errore pessimistico (penalità 1)?



Data Mining

Costruzione dei data set

Costruzione dei data set

■ Holdout

Utilizzare 2/3 dei record per il training e 1/3 per il test

Svantaggi:

- Opera con un training set ridotto
- Il risultato dipende dalla composizione del training set e del test set

■ Random subsampling

Consiste in una esecuzione ripetuta del metodo holdout in cui il dataset di training è scelto casualmente

■ Cross validation

Partiziona i record in k sotto-insiemi distinti

Esegui il training su k-1 partizioni ed il test sulla rimanente

Ripeti il test k volte e calcola l'accuracy media

ATTENZIONE: la cross validation crea k classificatori diversi e quindi la validazione indica quanto il tipo di classificatore e i suoi parametri sono adatti per lo specifico problema

- I k alberi decisionali costruiti potrebbero avere attributi e condizioni di split diverse a seconda delle caratteristiche del k-esimo training set

Bootstrap

- Prevede il reimbussolamento dei record già selezionati
- Se il dataset iniziale è composto da n record è possibile creare un training set di N record in cui ogni record ha circa il 63.2% di probabilità di comparire (con n sufficientemente grande)

$$\log_{n \rightarrow \infty} 1 - \left(1 - \frac{1}{n}\right)^n = 1 - \frac{1}{e} = 1 - 0,368 = 0,632$$

I record non utilizzati nemmeno una volta nel training set corrente compongono il validation set

$$\log_{n \rightarrow \infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e} = 0,368$$

- Il bootstrap non crea un (nuovo) dataset con più informazioni, ma permette di stabilizzare i risultati ottenibili del dataset a disposizione.
- E' quindi utilie soprattutto nel caso di dataset di piccole dimensioni.

C4.5

- **Algoritmo per la costruzione di alberi decisionale**

Estende l'algoritmo ID3 e Hunt

Una sua versione denominata **J48** è implementata in WEKA

- **Caratteristiche:**

Utilizza il GainRatio come criterio per determinare l'attributo di split

Gestisce gli attributi continui determinando uno split point che divide in due l'intervallo dei valori

Gestisce dati con valori mancanti. Gli attributi con valori mancanti non sono considerati per calcolare il GainRatio.

Può gestire attributi a cui sono associati pesi diversi

Esegue post-pruning dell'albero creato

- **La costruzione dell'albero si interrompe quando:**

Il nodo contiene record appartenenti a una sola classe

Nessun attributo permette di determinare un GainRatio positivo

Esercizio

- Utilizzando l'errore di classificazione come misura, identificare quale attributo deve essere scelto per primo e quale per secondo

Calcolare il gain e il gainRatio

A	B	C	# istanze	
			+	-
T	T	T	5	0
F	T	T	0	20
T	F	T	20	0
F	F	T	0	5
T	T	F	0	0
F	T	F	25	0
T	F	F	0	0
F	F	F	0	25

Come cambiano i risultati se si utilizza come attributo di split quello peggiore? Commentare il risultato



Imbalanced Class Problem e altre metriche di valutazione

Class Imbalance Problem

- In molti problemi di classificazione le classi sono distorte (molti più record in una classe rispetto a un'altra)
 - Frodi con carta di credito
 - Rilevamento delle intrusioni
 - Prodotti difettosi nella catena di montaggio di produzione
- Le misure di valutazione come l'accuratezza non sono adatte per le classi sbilanciate
- Rilevare la classe rara diventa difficilissimo

Metriche per la valutazione del modello

- La **Confusion Matrix** valuta la capacità di un classificatore sulla base dei seguenti indicatori
 - a) TP (true positive): record correttamente classificati come classe Yes
 - b) FN (false negative): record incorrettamente classificati come classe No
 - c) FP (false positive): record incorrettamente classificati come classe Yes
 - d) TN (true negative) record correttamente classificati come classe No

		Classe prevista	
		Class=Yes	Class>No
Classe effettiva	Class=Yes	TP (a)	FN (b)
	Class>No	FP (c)	TN (d)

- Se la classificazione utilizza n classi, la matrice di confusione sarà di dimensione $n \times n$

Accuratezza

		Classe prevista	
		Class=Yes	Class>No
Classe effettiva	Class=Yes	TP (a)	FN (b)
	Class>No	FP (c)	TN (d)

- L'accuratezza è la metrica maggiormente utilizzata per sintetizzare l'informazione di una confusion matrix

$$\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + FN + FP + TN}$$

- Equivalentemente potrebbe essere utilizzata la frequenza dell'errore

$$\text{Error rate} = \frac{b + c}{a + b + c + d} = \frac{FN + FP}{TP + FN + FP + TN}$$

Limiti dell'accuratezza

- L'accuratezza non è una metrica adeguata nel caso in cui le classi contengano un numero fortemente diverso di record (**class imbalance problem**)

Consideriamo un problema di classificazione binario in cui

- # record della classe 0 = 9990
- # record della classe 1 = 10

Un modello che predice sempre l'appartenenza alla classe 0 avrà un'accuratezza di $9990/10000 = 99.9\%$

- Nel caso di problemi di classificazione binaria la classe “rara” è anche chiamata *classe positiva*, mentre la classe che include la maggioranza dei record è chiamata *classe negativa*

Misure alternative

- Precision e Recall sono due metriche usate per la corretta classificazione dei record della classe positiva

Precision misura la frazione di record risultati effettivamente positivi tra tutti quelli che sono stati classificati come tali

Valori elevati indicano che pochi record della classe negativa sono stati erroneamente classificati come positivi.

Recall misura la frazione di record positivi correttamente classificati

Valori elevati indicano che pochi record della classe positiva sono stati erroneamente classificati come negativi.

		Classe prevista	
		Cl.=Yes	Cl.=No
Classe effettiva	Cl.=Yes	TP (a)	FN (b)
	Cl.=No	FP (c)	TN (d)

$$\text{Precision } (p) = \frac{a}{a + c} = \frac{TP}{RP + FP}$$

$$\text{Recall } (r) = \frac{a}{a + b} = \frac{TP}{TP + FN}$$

F-measure

- Una metrica che combina *precision* e *recall* è denominata **F-measure**:

$$F\text{-measure} = \frac{2 p r}{p+r} = \frac{2 T P^2}{2 T P^2 + F P + F N}$$

- F-measure rappresenta la media armonica tra precision e recall

- ✓ La media armonica tra due numeri x e y tende a essere vicina al più piccolo dei due numeri. Quindi se la media armonica è elevata significa che sia precision, sia recall lo sono.
 - ✓ ... e quindi non si è verificato un elevato numero di falsi sia positivi che negativi

$$F\text{-measure} = \frac{2}{\frac{1}{p} + \frac{1}{r}} = \frac{2 p r}{p+r}$$

Measure Alternative - Esempio

		Classe prevista	
Classe effettiva		Class=Yes	Class>No
	Class=Yes	10	0
	Class>No	10	980

$$\text{Precision (p)} = \frac{10}{10+10} = 0.5$$

$$\text{Recall (r)} = \frac{10}{10+0} = 1$$

$$\text{F - measure (F)} = \frac{2 * 1 * 0.5}{1 + 0.5} = 0.62$$

$$\text{Accuracy} = \frac{990}{1000} = 0.99$$

		Classe prevista	
Classe effettiva		Class=Yes	Class>No
	Class=Yes	1	9
	Class>No	0	990

$$\text{Precision (p)} = \frac{1}{1+0} = 1$$

$$\text{Recall (r)} = \frac{1}{1+9} = 0.1$$

$$\text{F - measure (F)} = \frac{2 * 0.1 * 1}{1 + 0.1} = 0.18$$

$$\text{Accuracy} = \frac{991}{1000} = 0.991$$

Measure Alternative - Esempio

	Classe prevista	
Class=Yes	40	10
Class>No	10	40

Precision (p) = 0.8

Recall (r) = 0.8

F - measure (F) = 0.8

Accuracy = 0.8

Measure Alternative - Esempio

	PREDICTED CLASS	
Class=Yes	40	10
Class>No	10	40

Precision (p) = 0.8

Recall (r) = 0.8

F - measure (F) = 0.8

Accuracy = 0.8

	Classe prevista	
Class=Yes	40	10
Class>No	1000	4000

Precision (p) = ~ 0.04

Recall (r) = 0.8

F - measure (F) = ~ 0.08

Accuracy = ~ 0.8

Ulteriori Misure

		Classe prevista	
		Yes	No
Classe effettiva	Yes	TP	FN
	No	FP	TN

TP rate (recall/sensitivity)

FN rate (β) = 1 - sensitivity

TN rate (specificity)

FP rate (α) = 1 – specificity

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

$$ErrorRate = 1 - accuracy$$

$$Precision = Positive\ Predictive\ Value = \frac{TP}{TP + FP}$$

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

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

$$FP\ Rate = \alpha = \frac{FP}{TN + FP} = 1 - specificity$$

$$FN\ Rate = \beta = \frac{FN}{FN + TP} = 1 - sensitivity$$

$$Power = sensitivity = 1 - \beta$$

Altre misure alternative

	Classe prevista	
Class=Yes	40	10
Class>No	10	40

Precision (p) = 0.8
TPR = Recall (r) = 0.8
FPR = 0.2
F - measure (F) = 0.8
Accuracy = 0.8

	Classe prevista	
Class=Yes	40	10
Class>No	1000	4000

Precision (p) = ~ 0.04
TPR = Recall (r) = 0.8
FPR = 0.2
F - measure (F) = ~ 0.08
Accuracy = ~ 0.8

Altre misure alternative

		Classe prevista	
Classe effettiva		Class=Yes	Class>No
	Class=Yes	10	40
	Class>No	10	40

Precision (p) = 0.5

TPR = Recall (r) = 0.2

FPR = 0.2

		Classe prevista	
Classe effettiva		Class=Yes	Class>No
	Class=Yes	25	25
	Class>No	25	25

Precision (p) = 0.5

TPR = Recall (r) = 0.5

FPR = 0.5

		Classe prevista	
Classe effettiva		Class=Yes	Class>No
	Class=Yes	40	10
	Class>No	40	10

Precision (p) = 0.5

TPR = Recall (r) = 0.8

FPR = 0.8

Imbalanced Class Problem

- Ordinamento basato su classi (e.g. RIPPER)
 - Le regole per le classi rare hanno una priorità più alta
- Classificazione sensibile ai costi
 - Classificare in modo errato la classe rara come classe di maggioranza è più costoso che classificare in modo errato la classe di maggioranza come classe rara
- Approcci basati sul campionamento

Matrice dei costi

- Codifica la **penalità** in cui si incorre nel classificare un record in una classe diversa

Una penalità negativa indica il “premio” che si ottiene per una corretta classificazione

$$C(M) = TP \times C(Yes|Yes) + FP \times C(Yes|No) + FN \times C(No|Yes) + TN \times C(No|No)$$

Cost Matrix		Classe prevista j	
Classe effettiva i	C(i j)	Class=Yes	Class=No
	Class=Yes	C(Yes Yes)	C(Yes No)
	Class=No	C(No Yes)	C(No No)

- Un modello costruito struttando, come funzione di purezza, una matrice di costo tenderà a fornire un modello a costo minimo rispetto ai pesi specificati

Matrice dei

Confusion Matrix	Classe prevista		
Classe effettiva		Class=Yes	Class>No
	Class=Yes	$f(\text{Yes}, \text{Yes})$	$f(\text{Yes}, \text{No})$
	Class>No	$f(\text{No}, \text{Yes})$	$f(\text{No}, \text{No})$

$C(i,j)$: Costo di classificare di un elemento della classe i nella classe j

Cost Matrix	Classe prevista		
Classe effettiva	$C(i, j)$	Class=Yes	Class>No
	Class=Yes	$C(\text{Yes}, \text{Yes})$	$C(\text{Yes}, \text{No})$
	Class>No	$C(\text{No}, \text{Yes})$	$C(\text{No}, \text{No})$

$$\text{Cost} = \sum C(i, j) \times f(i, j)$$

Calcolo del costo

Cost Matrix		Classe prevista	
Classe effettiva	C(i j)	+	-
	+	-1	100
	-	1	0

Model M ₁	Classe prevista		
Classe effettiva		+	-
	+	150	40
	-	60	250

Accuracy = 80%

Cost = 3910

Model M ₂	Classe prevista		
Classe effettiva		+	-
	+	250	45
	-	5	200

Accuracy = 90%

Cost = 4255

Classificazione Cost Sensitive

- Regola generale:

classifica il nodo **t** con classe **i** se il valore **i**

$$\text{minimizza } C(i|t) = \sum_j p(j|t) \times C(j, i)$$

Esempio

- Per un problema con 2 classi (+ e -) e nodo **x**

$$\text{Cost}(+|x) = p(+|x) C(+,+) + p(-|x) C(-,+)$$

$$\text{Cost}(-|x) = p(+|x) C(+,-) + p(-|x) C(-,-)$$

- classifica x come + se $\text{Cost}(+|x) < \text{Cost}(-|x)$

- if $C(+,+) = C(-,-) = 0$:

$$p(+ | x) > \frac{C(-,+)}{C(-,+) + C(+,-)}$$

Approccio basato sul campionamento

- Modifica la distribuzione dei dati di allenamento in modo che la classe rara sia ben rappresentata nel set di allenamento
- Sotto-campionamento della classe di maggioranza
Sopra-campionamento della classe rara
- Comporta vantaggi e svantaggi