Lezione 5

Alberi di decisione: Estensioni, valutazione

Mercoledì, 31 Gennaio 2007

Giuseppe Manco

References:

Chapter 3, Mitchell

Chapter 7 Han, Kamber

Chapter 5 Eibe, Frank

ICAR CNR
Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Algorithm *Build-DT* (*D*, *Attributi*)

IF tutti gli esempi hanno la stessa etichetta
THEN RETURN (nodo foglia con etichetta)
ELSE
IF *Attributi* = Ø
THEN RETURN (foglia con etichetta di maggioranza)
ELSE
scegli il migliore attributo *A* come radice
FOR EACH valore *v* di *A*
Crea una diramazione dalla radice con la condizione *A* = *v*
IF {*x* ∈ *D*: *x.A* = *v*} = Ø
THEN RETURN (foglia con etichetta di maggioranza)
ELSE *Build-DT* ({*x* ∈ *D*: *x.A* = *v*}, *Attributi* ~ {A})

Criteri per trovare il migliore split

- **Information gain (ID3 – C4.5)**
  - **Entropia, un concetto basato sulla teoria dell'informazione**
    - **Misura l'impurità di uno split**
    - **Seleziona l'attributo che massimizza la riduzione di entropia**
- **Gini index (CART)**
  - **Seleziona l'attributo che minimizza l'impurità**
- **Statistica del $\chi 2$ su tabelle di contingenza (CHAID)**
  - **Misura la correlazione tra un attributo e l'etichetta di classe**
  - **Seleziona l'attributo con la massima correlazione**

Altri criteri per la costruzione di alberi di decisione

- Schemi di branching:
  - Binari o a k vie
  - Attributi categorici/continui
- Stop rule: come decidere se un nodo è una foglia:
  - Tutti gli esempi appartengono alla stessa classe
  - *La misura di qualità al di sopra di una certa soglia*
  - Non ci sono più attributi da splittare
  - Non ci sono più istanze nella partizione
- Labeling rule: un nodo foglia è etichettato con la classe a cui la maggior parte degli esempi nel nodo appartengono

Attributi con valori continui

- **Due metodi principali**
  - **Discretization**
    - **Non supervisionata**
    - **Supervisionata (e.g., ChiMerge)**
      - Esercizio: valutare come cambia l'accuratezza se si utilizza il ChiMerge
  - **Utilizzo di soglie per creare split binari**
    - **Esempio: $A \leq a$ produce I sottoinsiemi $A \leq a$ and $A > a$**
    - ***Information gain/Gini/Chi-Quadro può essere calcolata su tali soglie***
- **Come si ottengono le soglie?**
  - **FOR EACH attributo continuo *A***

    Si suddividano gli esempi $\{x \in D\}$ in base ai valori di $x.A$

    FOR EACH coppia ordinata (l, u) di valori di *A* con etichette differenti

    Si valuti il guadagno del treshold risultante $D_{A \leq (l+u)/2}$, $D_{A > (l+u)/2}$
  - **Computazionalmente molto dispendioso**

Weather data – valori nominali

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	Normal	False	Yes
…	…	…	…	…

```
If outlook = sunny and humidity = high then play = no
If outlook = rainy and windy = true then play = no
If outlook = overcast then play = yes
If humidity = normal then play = yes
If none of the above then play = yes
```

Weather data – valori numerici

Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	86	False	Yes
Rainy	75	80	False	Yes
…	…	…	…	…

```
If outlook = sunny and humidity > 83 then play = no
If outlook = rainy and windy = true then play = no
If outlook = overcast then play = yes
If humidity < 85 then play = yes
If none of the above then play = yes
```

esempio

- **Split sull'attributo temperature:**

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

  - **Esempio:** **temperature $<$ 71: yes/4, no/2**
    temperature $\geq$ 71: yes/5, no/3
  - **Gain([4,2],[5,3])**
    = 6/14 Gain([4,2]) + 8/14 Gain([5,3])
    = 0.939 bits

Complessità computazionale

- **Si ordinano le istanze in base ai valori dell'attributo numerico**
  - ***O* (*n* log *n*)**
- *D. Abbiamo bisogno di ripeterlo per ogni nodo dell'albero?*
- **A: No! Basta derivare l'ordinamento da un ordinamento predefinito**
  - ***O* (*n*)**
  - **Svantaggio: bisogna creare un array per ogni attributo**

Aumentare le prestazioni

- **Possiamo valutare l'entropia nei vari punti differenti (Fayyad & Irani, 1992)**

Tagli potenziali

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

I tagli tra i valori della stessa classe non sono ottimali

Split binari e multipli

- **Lo split su un attributo nominale "consuma" l'attributo**
  - **L'attributo nominale è testato al più una volta in ogni cammino**
- **La cosa non vale per gli attributi numerici!**
  - **Possono essere testati più di una volta**
- **Conseguenza: albero complicato da leggere**
  - **pre-discretizzazione**
  - **Split multipli**

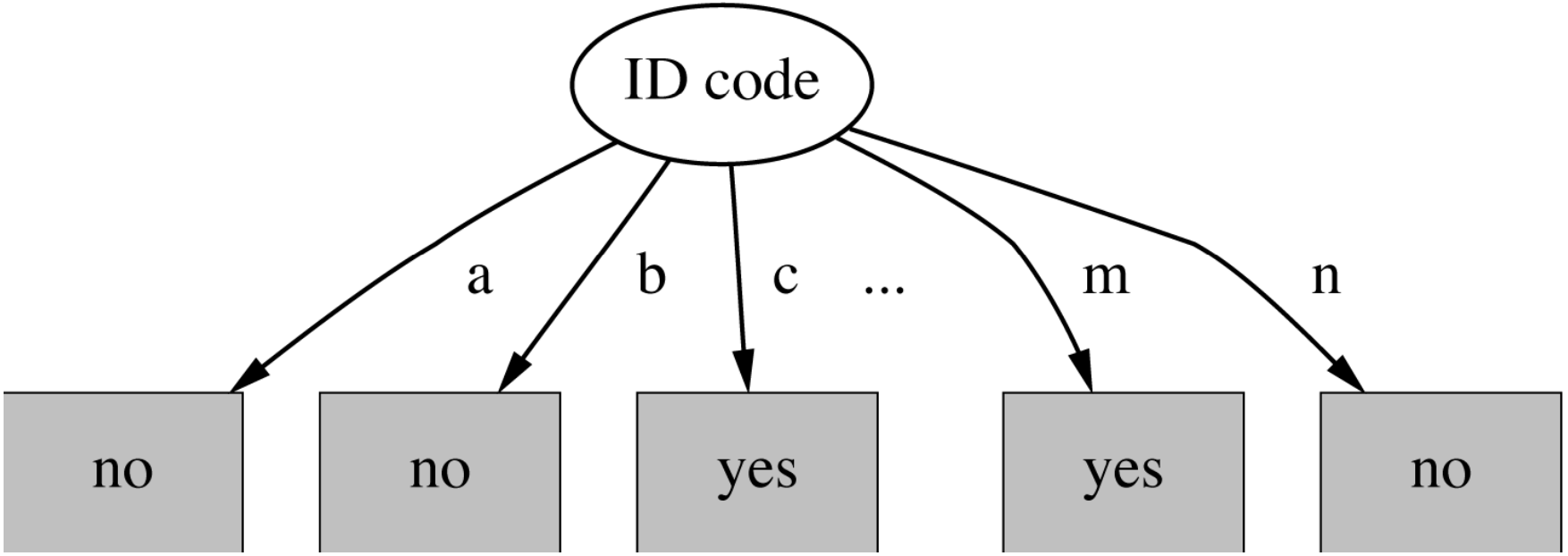
Attributi nominali

- Problema: Attributi con un ampio numero di valori (casi estremi: i codici ID)
- I sottoinsiemi è probabile che siano meno impuri se c'è un grande numero di valori
  - ⇒Information gain è portata a scegliere attributi con un grande numero di valori
  - ⇒La conseguenza è l'overfitting (selezione di un attributo non ottimale per la predizione)

PlayTennis con ID code

ID	Outlook	Temperature	Humidity	Windy	Play?
A	sunny	hot	high	false	No
B	sunny	hot	high	true	No
C	overcast	hot	high	false	Yes
D	rain	mild	high	false	Yes
E	rain	cool	normal	false	Yes
F	rain	cool	normal	true	No
G	overcast	cool	normal	true	Yes
H	sunny	mild	high	false	No
I	sunny	cool	normal	false	Yes
J	rain	mild	normal	false	Yes
K	sunny	mild	normal	true	Yes
L	overcast	mild	high	true	Yes
M	overcast	hot	normal	false	Yes
N	rain	mild	high	true	No

Split per l'attributo ID Code



L'entropia è 0 (poiché ogni nodo radice è "puro", avendo un solo caso.

Information gain è massimale per ID code

Gain ratio [1]

- *Una modifica dell'information Gain* che riduce l'influenza degli attributi nominali
- desiderata
  - Grande quando i dati sono uniformemente distribuiti
  - Piccolo quando tutti i dati appartengono ad un solo ramo
- Il Gain ratio considera sia il numero che la dimensione delle partizioni quando valuta un attributo
  - Corregge l' information gain con l'informazione intrinseca di uno split

Gain Ratio [2]

- *Gain ratio* (Quinlan'86) Normalizza l'information Gain:

$$Gain(D,A) \equiv -H(D) - \sum_{v \in values(A)} \left[\frac{|D_v|}{|D|} \bullet H(D_v) \right]$$

$$GainRatio(D,A) \equiv \frac{Gain(D,A)}{SplitInformation(D,A)}$$

$$SplitInformation(D,A) \equiv -\sum_{v \in values(A)} \left[\frac{|D_v|}{|D|} \lg \frac{|D_v|}{|D|} \right]$$

- SplitInfo: entropia della distribuzione delle istanze nella partizione

Gain Ratio [3]

- **Esempio: SplitInfo per ID code**

$$\text{info}([1,1,\ldots,1]) = 14 \times (-1/14 \times \log 1/14) = 3.807 \text{ bits}$$

- L'importanza dell'attributo diminuisce quando SplitInfo aumenta
- **Gain ratio:**

$$\text{gain\_ratio}(\text{"ID\_code"}) = \frac{0.940 \text{ bits}}{3.807 \text{ bits}} = 0.246$$

Gain ratio per weather data

Outlook		Temperature	
Info:	**0.693**	**Info:**	**0.911**
Gain: 0.940-0.693	**0.247**	**Gain: 0.940-0.911**	**0.029**
Split info: info([5,4,5])	**1.577**	**Split info: info([4,6,4])**	**1.362**
Gain ratio: 0.247/1.577	**0.156**	**Gain ratio: 0.029/1.362**	**0.021**

Humidity		Windy	
Info:	**0.788**	**Info:**	**0.892**
Gain: 0.940-0.788	**0.152**	**Gain: 0.940-0.892**	**0.048**
Split info: info([7,7])	**1.000**	**Split info: info([8,6])**	**0.985**
Gain ratio: 0.152/1	**0.152**	**Gain ratio: 0.048/0.985**	**0.049**

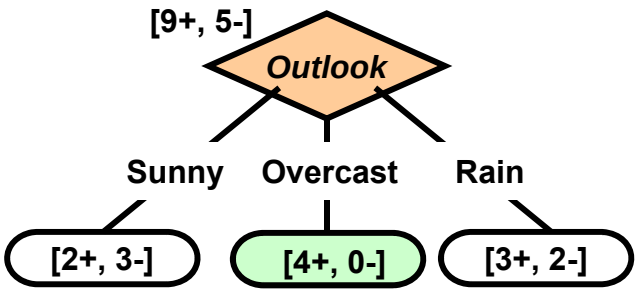
Ancora gain ratio

- "ID code" ha ancora un Gain Ratio maggiore
  - Soluzione: test *ad hoc*
- Problemi ipercompensazione
  - Un attributo può essere scelto esclusivamente in base allo SplitInfo
  - Soluzione:
    - Consideriamo solo gli attributi con un Information Gain più grande della media
    - Li confrontiamo sul gain ratio

Valori mancanti

- **Che succede se alcune istanze non hanno tutti i valori?**
  - **Due situazioni differenti**
    - **Apprendimento: si valuta *Gain* (*D*, *A*), ma per qualche $x \in D$, un valore di *A* non è dato**
    - **Classificazione: si classifica *x* senza conoscere il valore di *A***
- **Soluzioni: effettuiamo una scelta nel calcolare *Gain*(*D*, *A*)**

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis*?
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	???	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No



Approcci ai valori mancanti

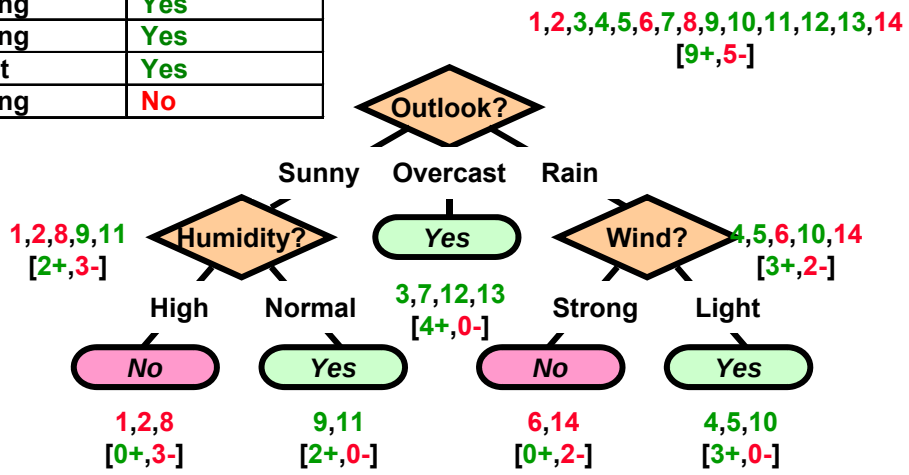
- Il valore mancante è un valore a sé
  - CHAID, C4.5
- Imputazione: scegliamo il valore più probabile
  - Sostiuiamo il valore con la media/mediana/moda
  - La sostituzione può essere globale al dataset o locale al nodo
- Proporzioniamo la scelta
  - Si assegna una probabilità p_i ad ogni valore v_i di $x.A$ [Quinlan, 1993]
    - Assegniamo una frazione p_i di x ad ogni discendente nell'albero
    - Utilizziamo i pesi per calcolare *Gain* (D, A)
- In entrambi gli approcci, classifichiamo i nuovi esempi allo stesso modo

Esempio

- imputiamo *x.A*
  - 1° variante: *Humidity* = *Normal*
  - 2° variante: *Humidity* = *High* (tutti I casi *No* sono *High*)

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis*?
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	???	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

- Scelta pesata
  - *0.5 High, 0.5 Normal*
  - *Gain* < 0.97
- Test: <*?*, *Hot*, *Normal*, *Strong*>
  - 1/3 *Yes* + 1/3 *Yes* + 1/3 *No* = *Yes*



Valori mancanti in CART: Surrogati

- CART si basa sui valori di altri attributi
- Esempio: l'attributo INCOME
- Gli attributi Education o Occupation possono essere surrogati
  - education alta = income alto
- Conseguenza: Inferiamo i valori dai surrogati

Scalabilità

- Che succede se il dataset è troppo grande per la memoria centrale?
- Approcci iniziali:
  - Costruzione incrementale dell'albero (Quinlan 86)
  - Combinazione degli alberi costruiti su partizioni separate (Chan & Stolfo 93)
  - Riduzione del dataset tramite campionamento (Cattlet 91)
- Obiettivo: gestire dati dell'ordine di 1G con 1K attributi
- Vari approcci rivelatisi di successo
  - SLIQ (Mehta et al. 96)
  - SPRINT (Shafer et al. 96)
  - PUBLIC (Rastogi & Shim 98)
  - RainForest (Gehrke et al. 98)

Il rasoio di Occam e gli alberi di decisione

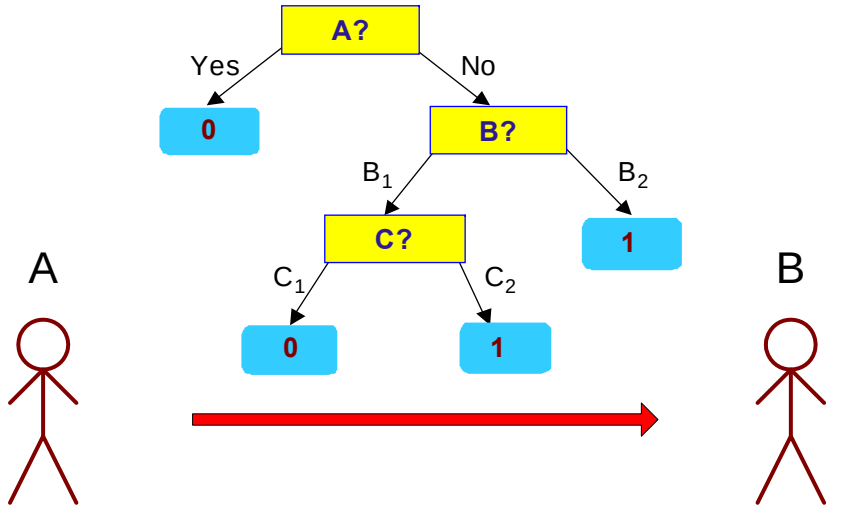
- **Preference Biases / Language Biases**
  - **Preference bias**
    - **Espresso ("codificato") nell'algoritmo di learning**
    - ***Euristica di ricerca***
  - **Language bias**
    - **Espresso nel linguaggio di rappresentazione**
    - **Restrizione dello spazio di ricerca**
- **Rasoio di Occam: motivazioni**
  - **Le ipotesi compatte sono meno frequenti delle ipotesi complesse**
    - **Conseguenza: meno probabile che siano coincidenze**

Rasoio di Occam: argomenti contro

- **Che vuol dire "ipotesi compatte"?**
  - **Esistono varie definizioni di compattezza**
    - **Alberi con un numero primo di nodi che usano Z come primo attributo**
    - **Perché non preferire alberi che assegnano priorità agli attributi?**

Minimum Description Length (MDL)

X	y
X_1	1
X_2	0
X_3	0
X_4	1
…	…
X_n	1



X	y
X_1	?
X_2	?
X_3	?
X_4	?
…	…
X_n	?

- **Cost(Model,Data) = Cost(Data|Model) + Cost(Model)**
  - **Cost rappresenta il numero di bit necessari per codificare.**
  - **Il modello più piccolo è il migliore.**
- **Cost(Data|Model) codifica gli errori di misclassificazione.**
- **Cost(Model) codifica la struttura dell'albero.**

Errore

- Si possono ottenere ipotesi consistenti?
  - È auspicabile?
  - $error_D(h)=|\{x|h(x) \neq c(x), <x,c(x)> \in D\}|$
- È sempre possibile ottenere un albero con l'errore minimale
  - Perché?
  - È auspicabile?

Valutazione dell'errore

- **Matrice di confusione**

Classe attuale

Classe predetta	A	B
A	TP	FP
B	FN	TN

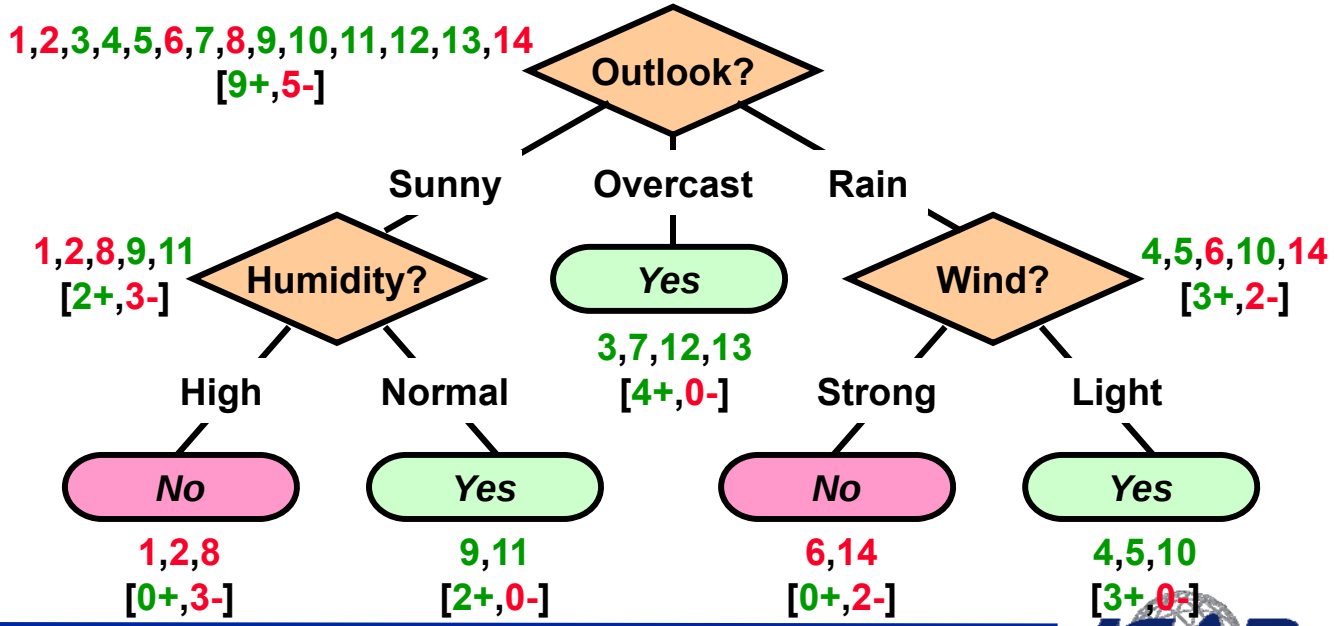
- **Accuratezza = (TP+TN)/(TP+TN+FP+FN)**
  - **Errore = 1-accuratezza**

Weather

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis*?
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

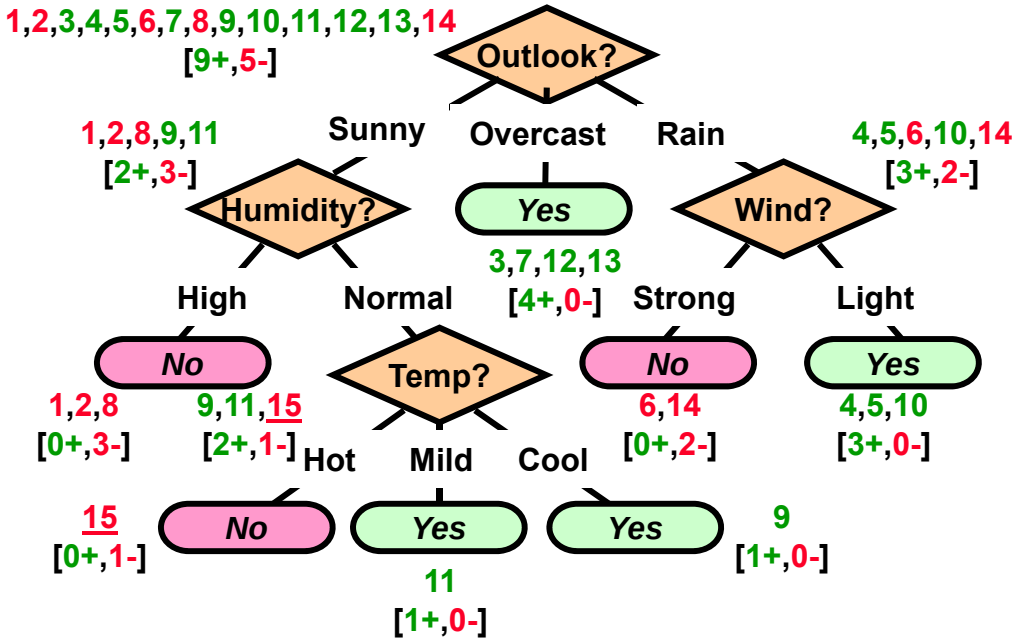
Albero di decisione C4.5

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis*?
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No



Overfitting

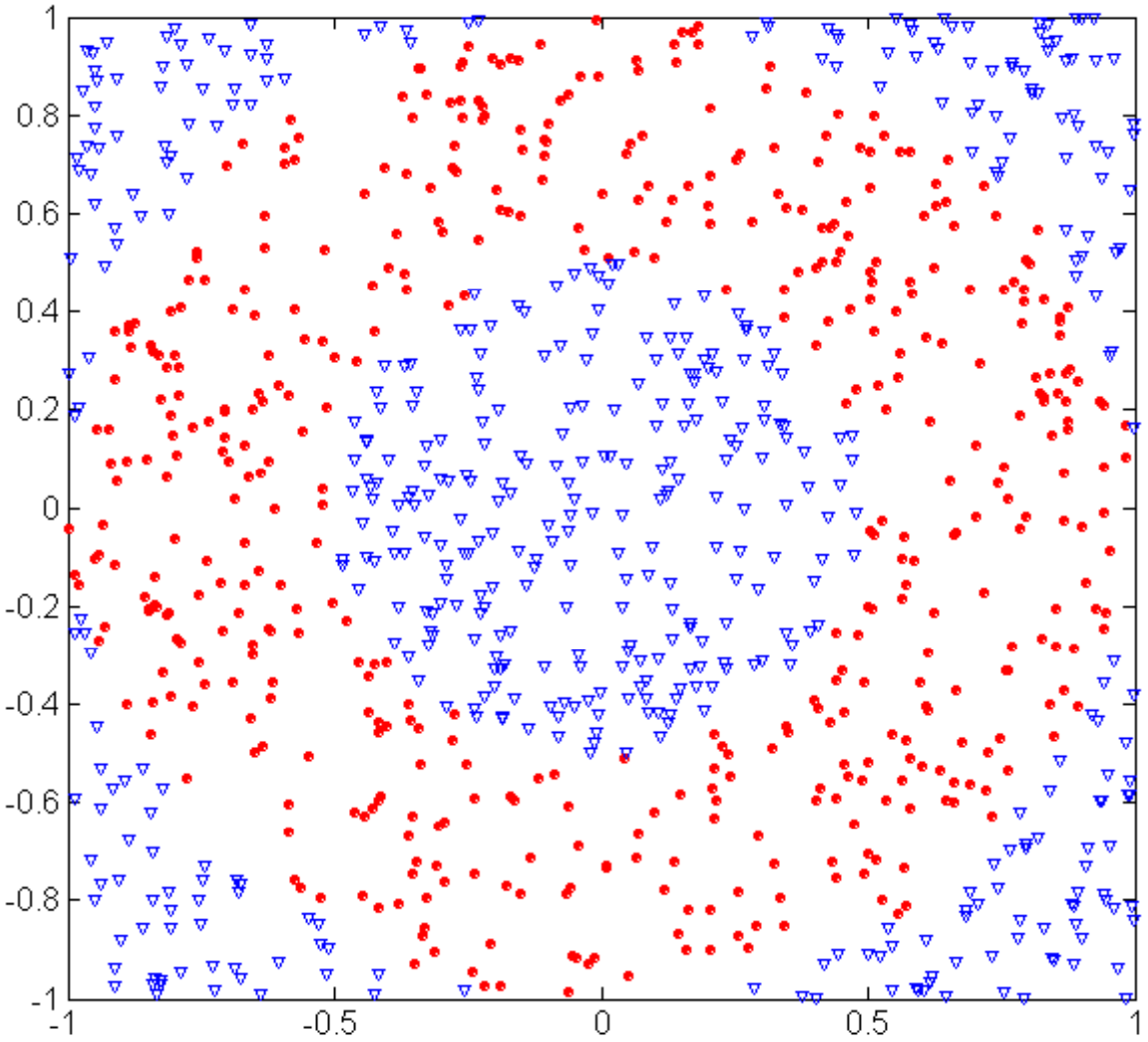
- **L'albero indotto**



Si adatta al rumore!

- **Rumore nel training set**
  - **Istanza 15: <*Sunny, Hot, Normal, Strong, ->**
    - **L'etichetta correttà è +**
    - **L'albero misclassifica l'istanza**
  - **Se aggiustiamo l'albero con questa istanza**
  - **Il nuovo albero si comporterà peggio del precedente**

Overfitting



1000 punti distinti.

Punti circolari:

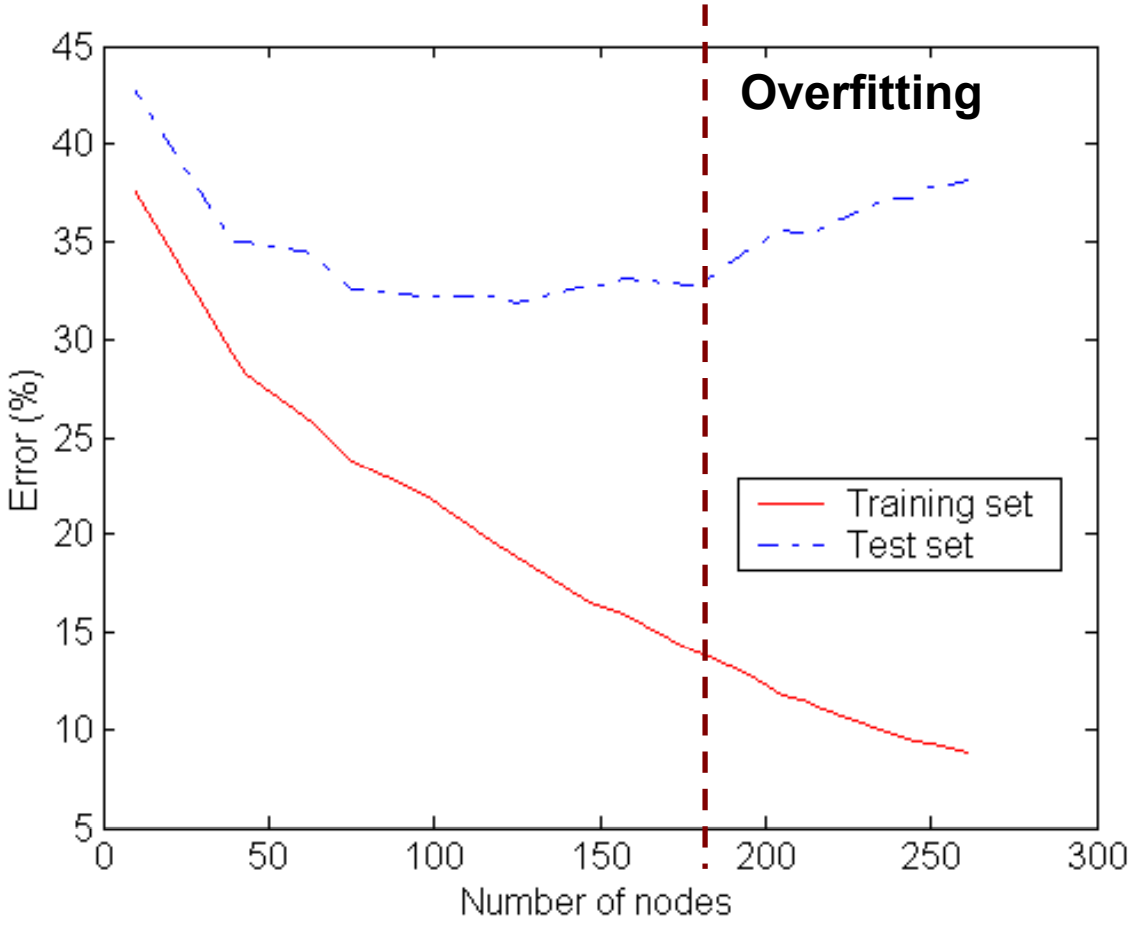
$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

Punti triangolari:

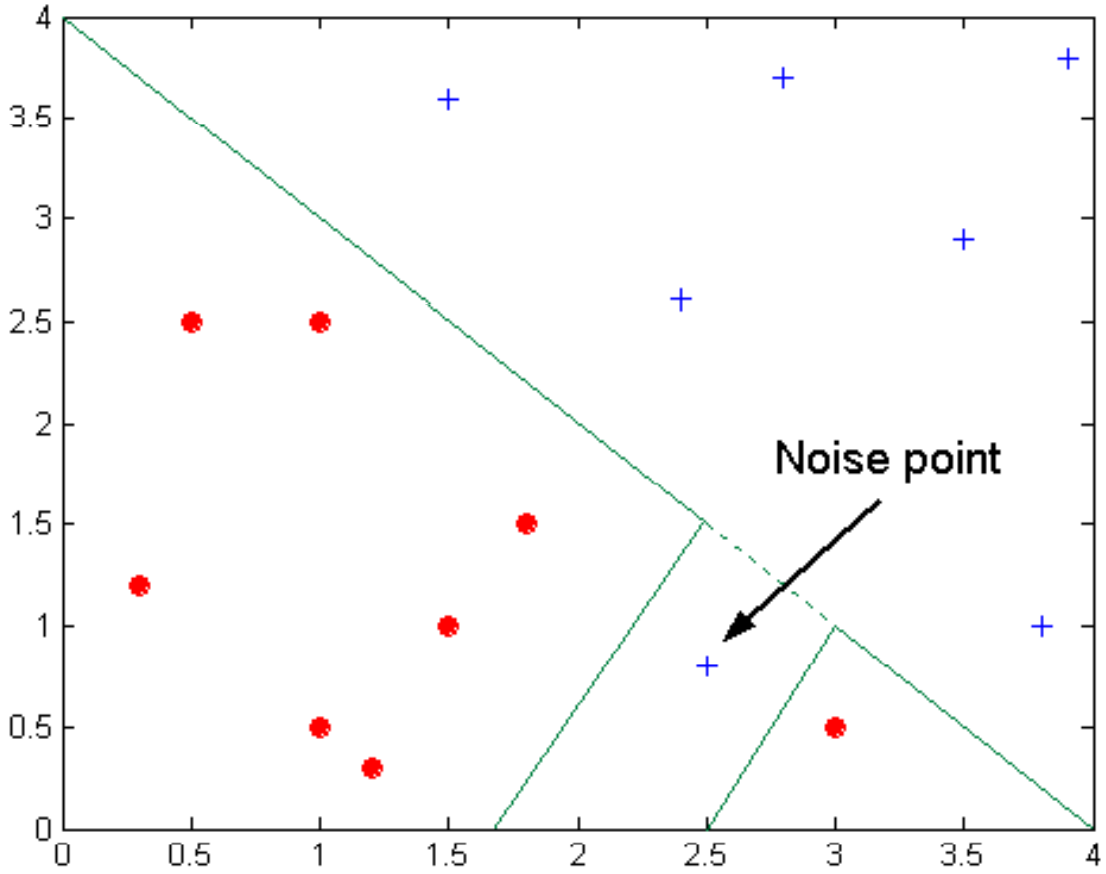
$\text{sqrt}(x_1^2+x_2^2) > 0.5$ o

$\text{sqrt}(x_1^2+x_2^2) < 1$

Overfitting

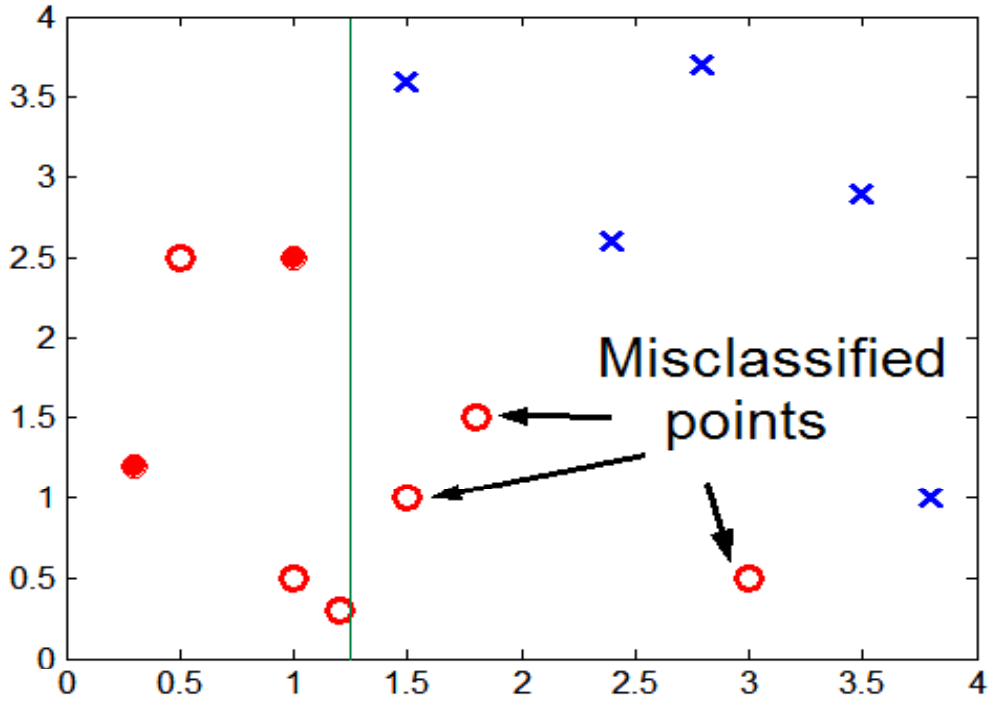


Overfitting dovuto al rumore



Decision boundary distorto

Overfitting dovuto a pochi esempi



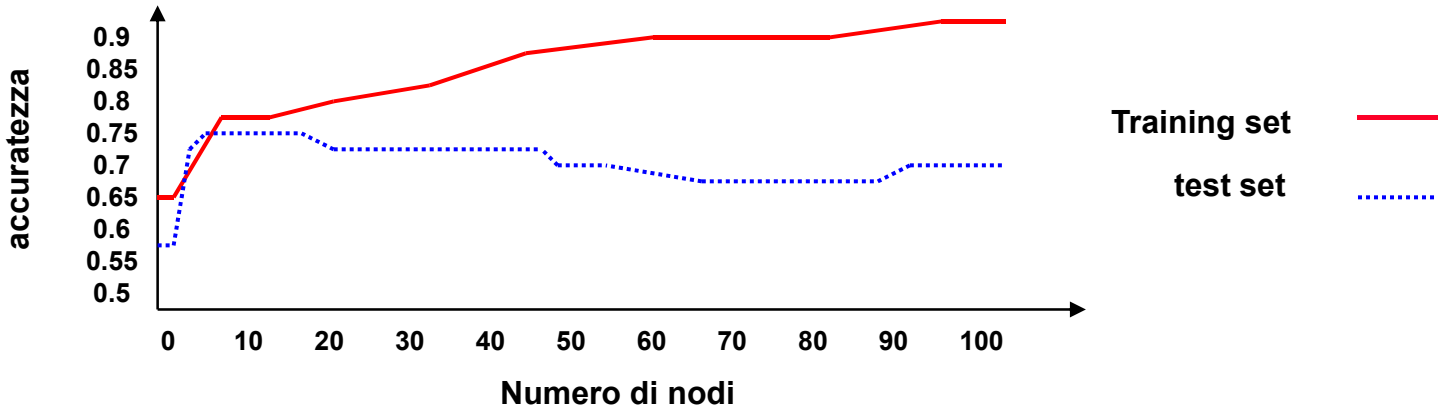
La mancanza di punti nella in basso rende difficoltosa la predizione

Overfitting

- **Definizione**
  - *h* presenta overfitting su *D* se ∃ un'ipotesi alternativa *h'* per la quale
    - $error_D(h) < error_D(h')$ but $error_{test}(h) > error_{test}(h')$
  - Cause tipiche: traiing set troppo piccolo (le decisioni sono basate su pochi dati); rumore
- **Come si allevia l'overfitting?**
  - Prevenzione
    - Selezionare solo gli attributi *rilevanti* (utili nel modello)
    - Richiede una misura della rilevanza
  - aggiramento
    - Schivare il problema quando c'è sentore che sta per avvenire
    - Valutare h su un insieme di test e fermare la costruzione del modello quando le performances scadono
  - Riabilitazione
    - "terapia di recupero"
    - Costruzione del modello, eliminazione degli elementi che contribuiscono all'overfitting

Combattere l'overfitting

- Prevenzione
- aggiramento

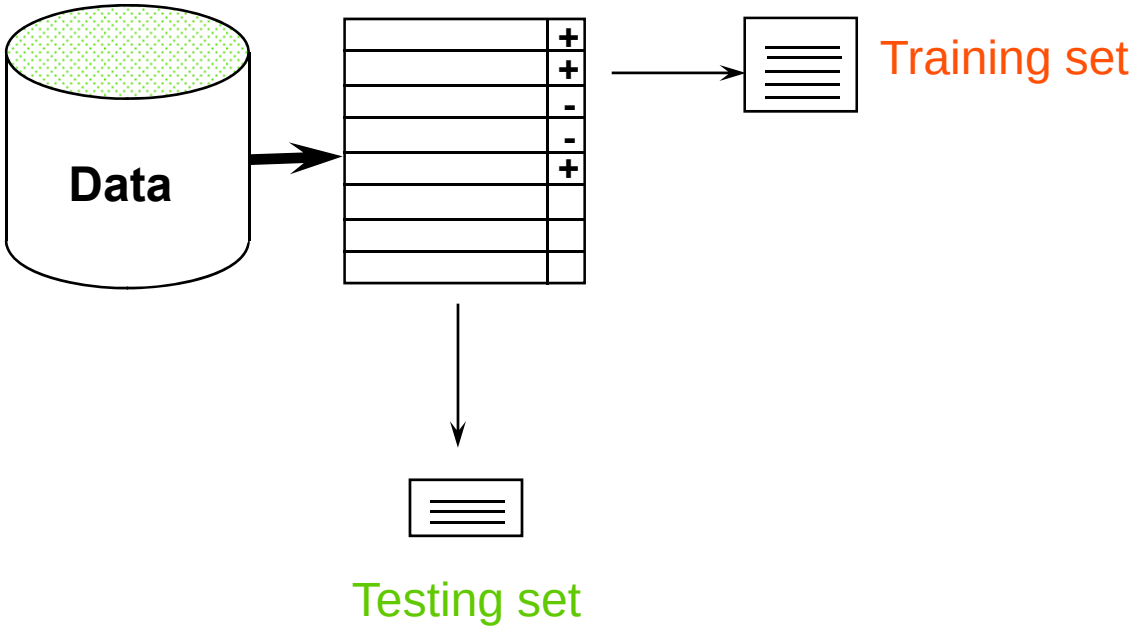


- **Come selezionare il modello migliore**
  - Si misurano le performances su training set e su un validation set separato
  - Minimum Description Length (MDL):
    si minimizza *size*(*h* ≡ T) + *size* (*misclassificazioni* (*h* ≡ T))

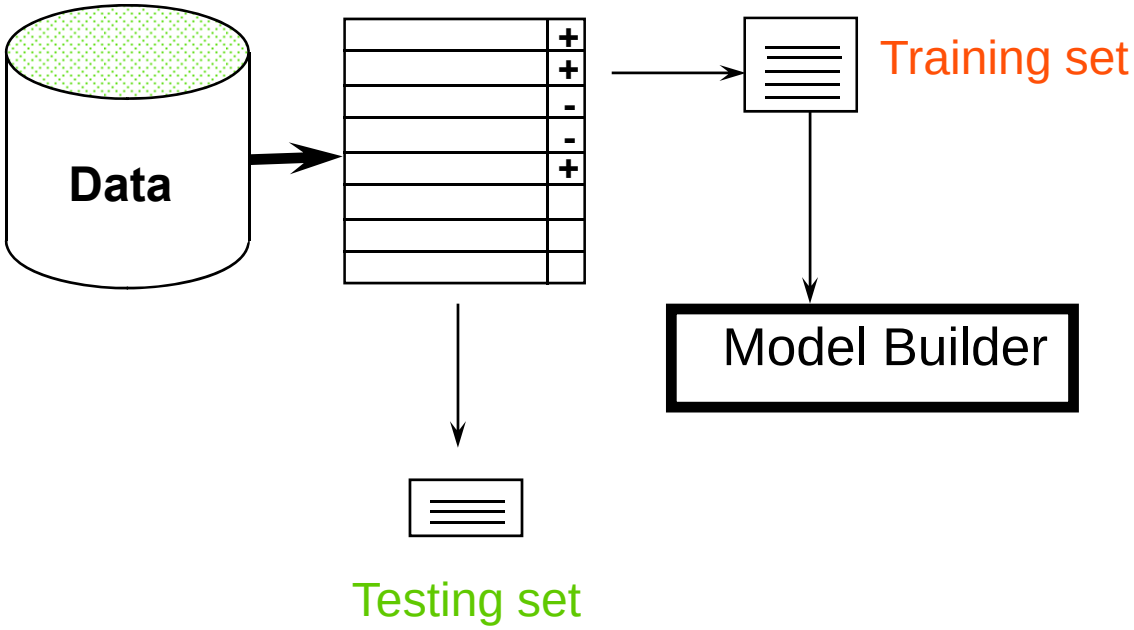
Valutazione

- **Training/test**
  - **Partizioniamo i dati in training e (ad esempio 2/3 per il training, 1/3 per il test)**
- **Si costruisce il classificatore con il training set, lo si valuta sul *test* set.**

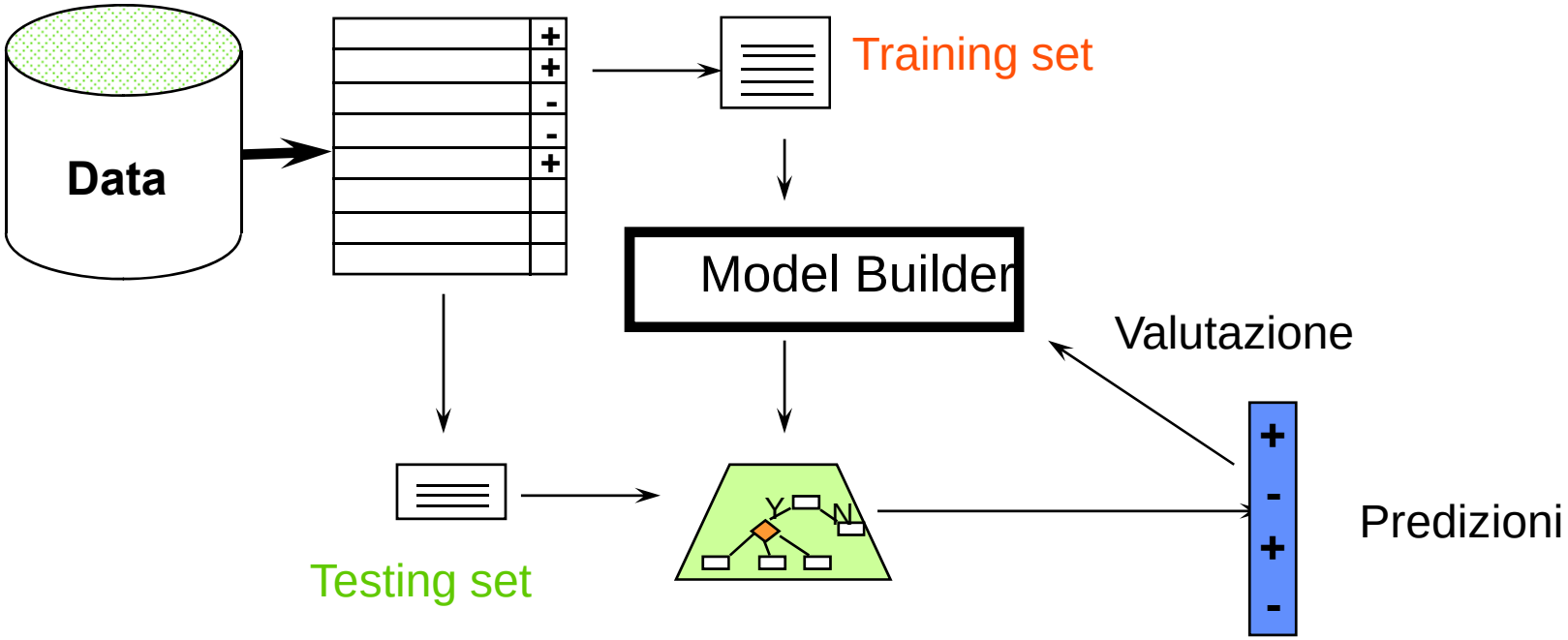
Model generation 1



Model generation 2



Model generation 3



Rimuovere l'overfitting

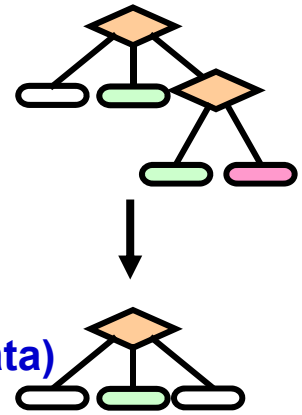
- **Due approcci**
  - **Pre-pruning (aggiramento): si ferma lo sviluppo dell'albero se durante la costruzione si determina che le scelte non sono più affidabili**
  - **Post-pruning (rimozione): si sviluppa l'intero albero e si rimuovono I nodi che non hanno una ragione sufficiente d'essere**
- **Post-pruning preferibile nella pratica: il pre-pruning si potrebbe fermare "troppo presto"**

Prepruning

- **Basato su test statistici**
  - **Quando non c'è dipendenza statisticamente rilevante tra gli attributi e la classe, il nodo non viene sviluppato**
- **CHAID: *test del chi-quadro***
  - **Utilizzato anche in ID3, insieme all'information gain**
    - **Solo gli attributi statisticamente significativi possono essere selezionati per information gain**

Pruning per la riduzione dell'errore

- Approccio Post-Pruning
- Dividiamo i dati in Training e Validation Set
- Function *Prune*(*T, n:node*)
  - Rimuovi il sottoalbero che ha la radice in n
  - Trasforma n in una foglia (con l'etichetta di maggioranza associata)



- Algorithm *Reduced-Error-Pruning* (*D*)
  - Partiziona *D* in D_{train}, $D_{validation}$
  - Costruisci l'albero *T* on D_{train}
  - WHILE l'accuratezza su $D_{validation}$ diminuisce DO

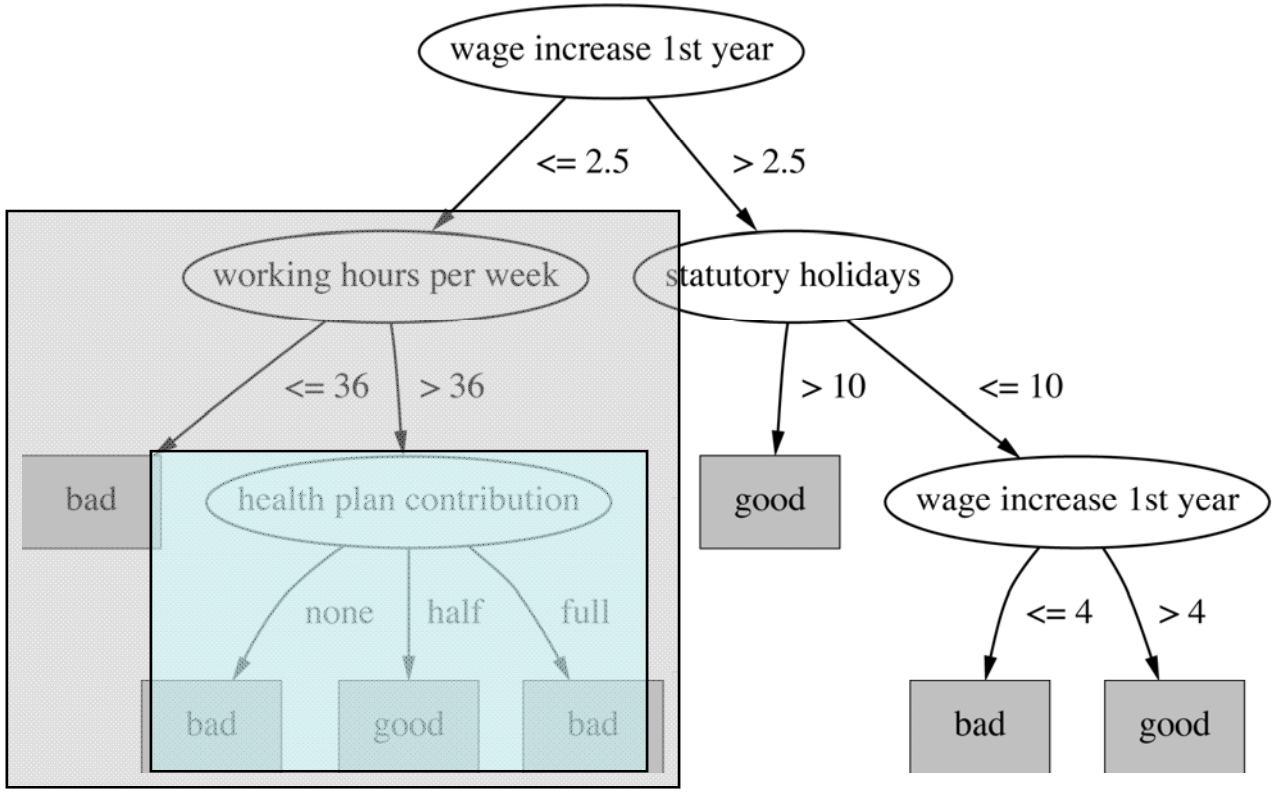
    FOR EACH nodo non-foglia *T*

    Temp[*candidate*] ← *Prune* (*T*, *candidate*)

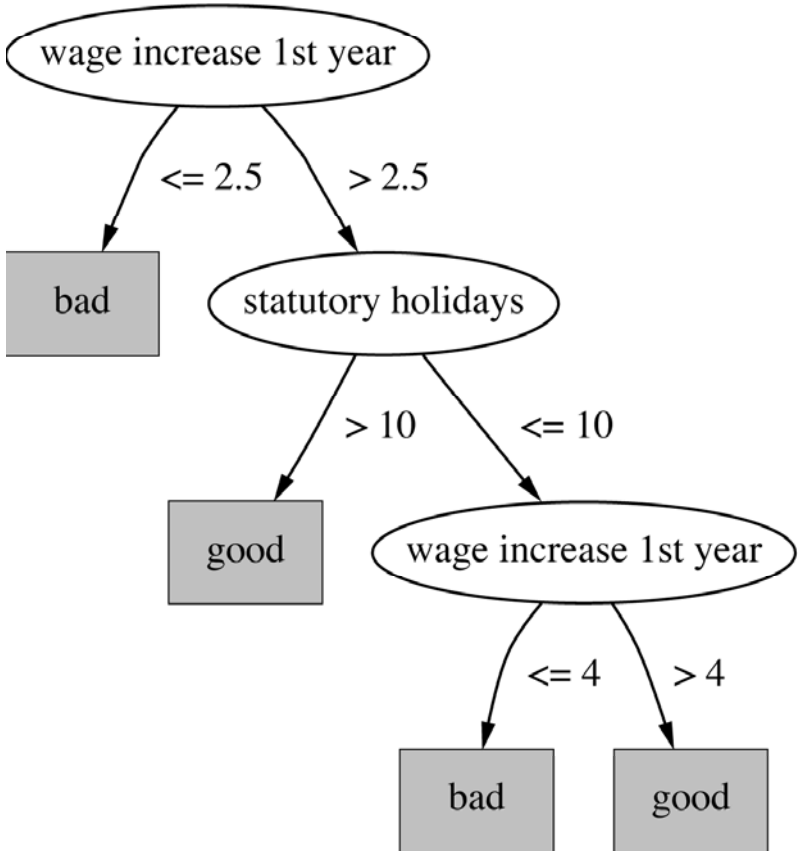
    Accuracy[*candidate*] ← *Test* (*Temp*[*candidate*], $D_{validation}$)

    T ← *T'* ∈ *Temp* con il miglior valore di accuratezza
  - RETURN *T*

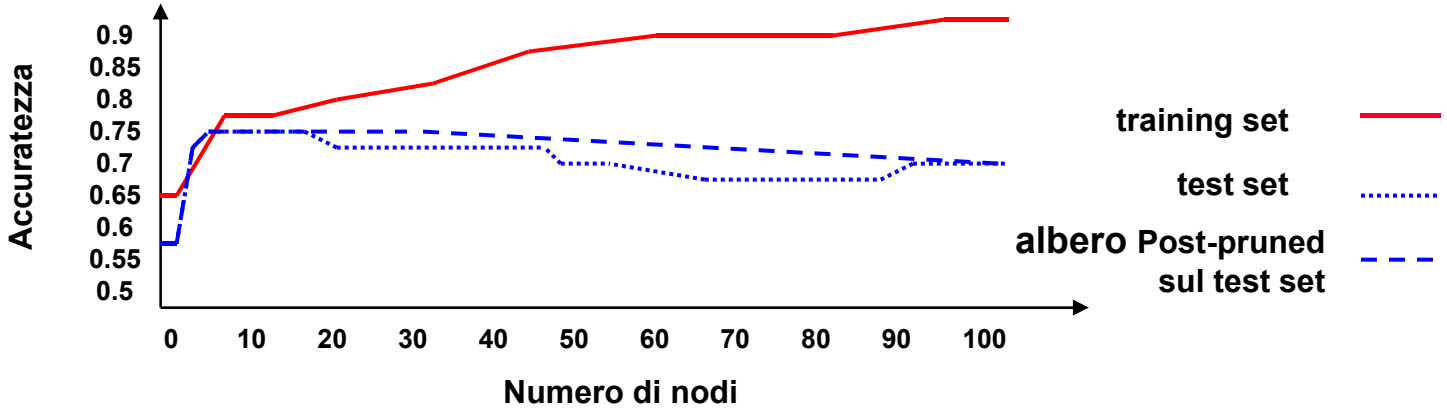
Rimpiazzamento del sottoalbero



Rimpiazzamento del sottoalbero

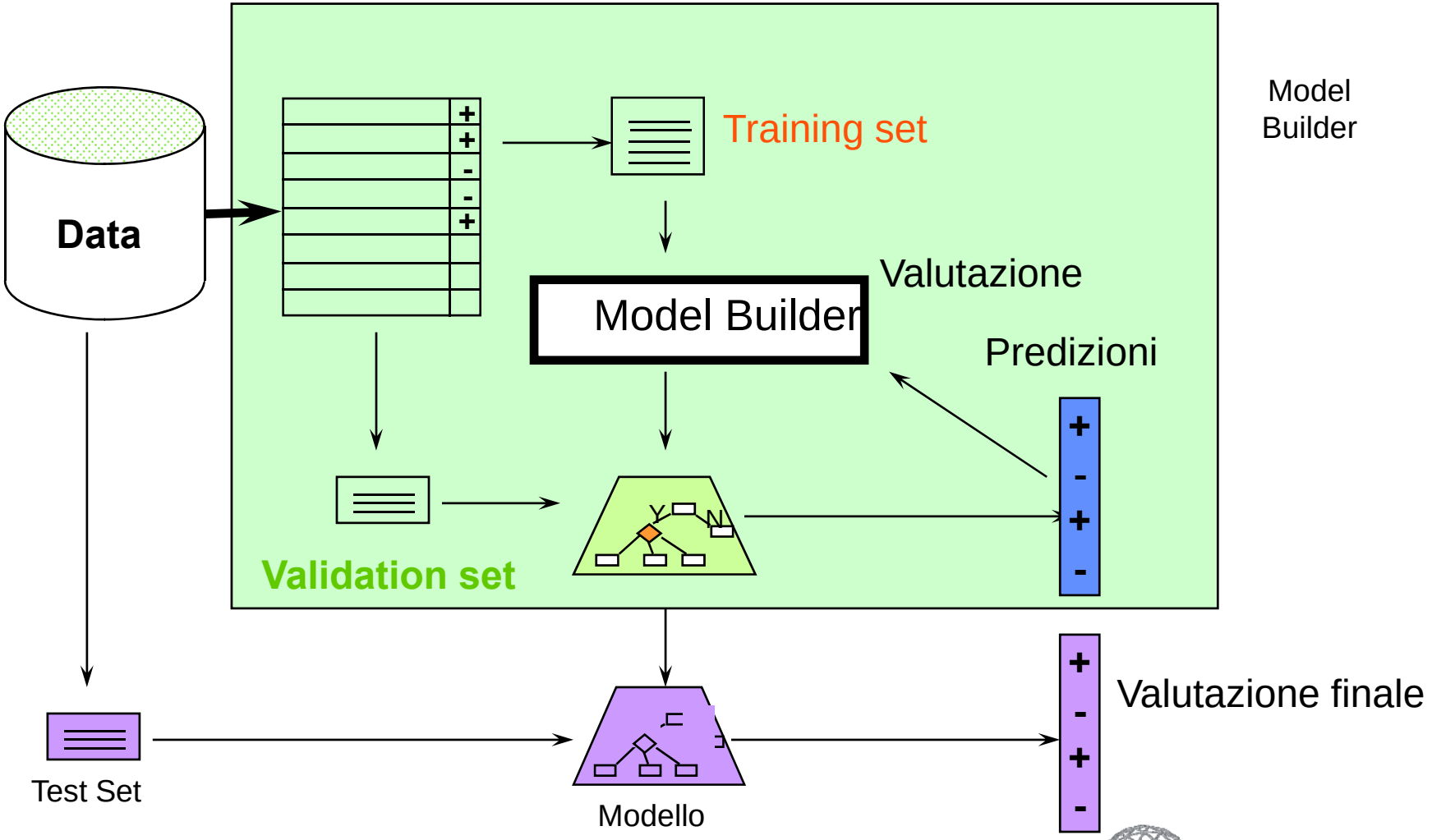


Risultati

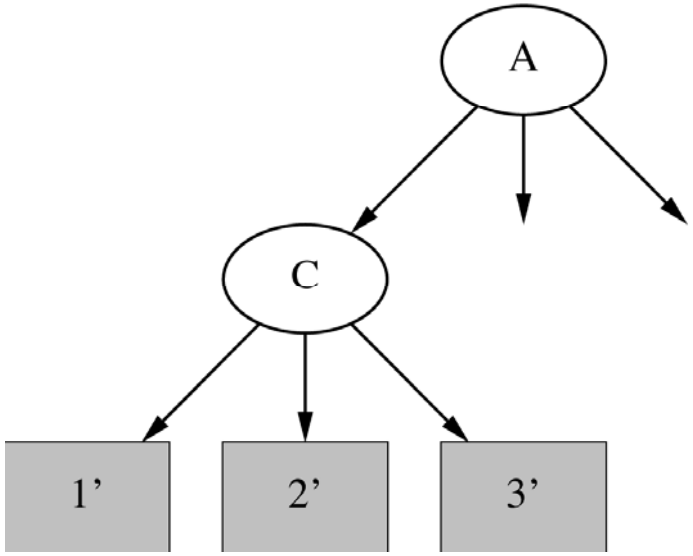


- Eliminando nodi, l'errore diminuisce
- *NB*:$D_{validation}$ è differente sia da D_{train} che da D_{test}

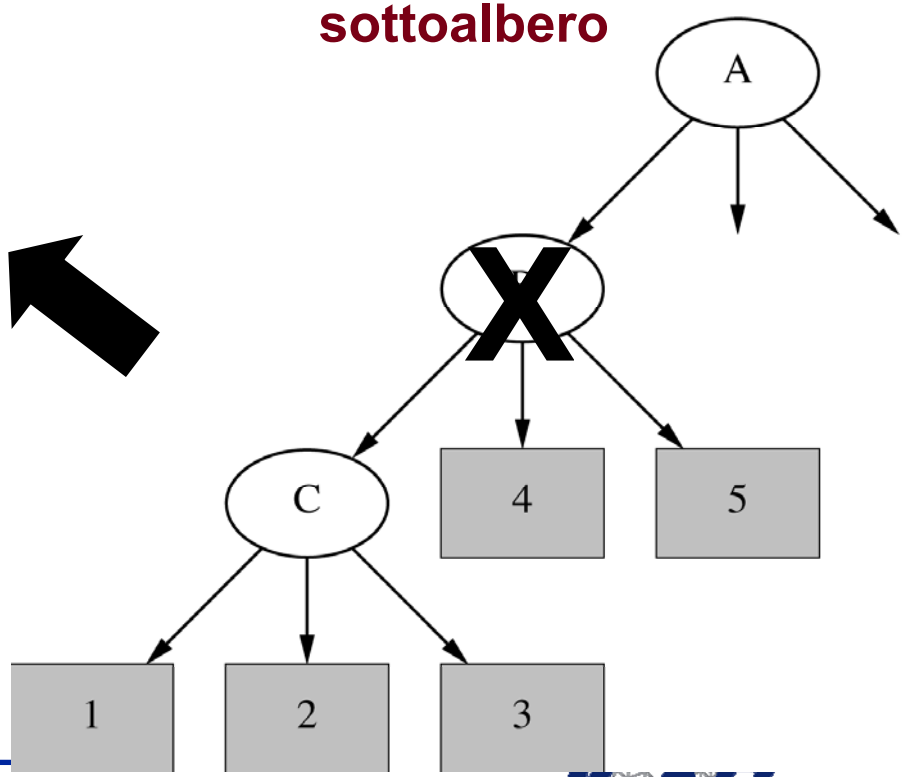
Model generation rivisitato



Riorganizzazione del sottoalbero



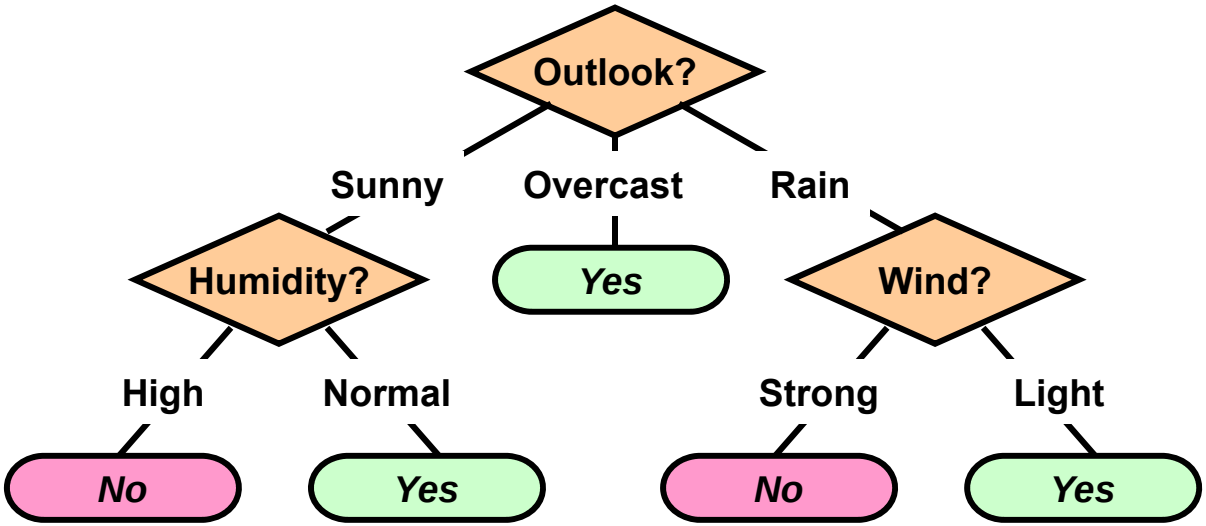
- Eliminazione di un nodo
- Ridistribuzione delle istanze
- Più lento del rimpiazzamento del sottoalbero



Post-Pruning di regole

- **Utilizzato ffrequentement**
  - **Variante utilizzata in *C4.5***
- **Algorithm *Rule-Post-Pruning* (*D*)**
  - **ottieni *T* from *D* – sviluppa l'albero su *D* finché non c'è fitting totale**
  - **Converti *T* in un insieme di regole equivalenti**
  - **generalizza ogni regola indipendentemente cancellando tutte le precondizioni la cui cancellazione causa un aumento dell'accuratezza stimata**
  - **Ordina le regole ottenute per ordine di accuratezza**

Conversione di un albero



- **Esempio**
  - IF (*Outlook* = *Sunny*) $\wedge$ (*Humidity* = *High*) THEN *PlayTennis* = *No*
  - IF (*Outlook* = *Sunny*) $\wedge$ (*Humidity* = *Normal*) THEN *PlayTennis* = *Yes*
  - …

Di più sull'errore

- Il pruning va applicato solo se riduce l'errore stimato
- C4.5
  - Otteniamo gli intervalli di confidenza sul training set
  - Utilizziamo una stima derivata da tale insieme per il pruning

Due definizioni di errore

- **Errore "vero"**
  - **Visione probabilistica**

$$error_D(h) = \underset{x \in D}{P}(c(x) \neq h(x))$$

- **Errore sul campione**
  - **Visione frequentistica**

$$error_s(h) = \frac{1}{n} \sum_{x \in S} \delta(c(x) \neq h(x))$$

- **Quanto $error_S(h)$ approssima $error_D(h)$?**

Esempio

- **h misclassifica 12 esempi su 40 S**

$$error_S(h) = \frac{12}{40} = .30$$

- **Qual'è $error_D(h)$?**

Stime,Previsioni

- Dato *S* di dimensione *n*
- Si valuti $error_S(h)$
  - $error_S(h)$ è una variabile casuale
- Cosa possiamo concludere?

Intervalli di confidenza [1]

- **Se**
  - **S contiene n istanza**
  - **n>30**
- **allora**
  - **Con probabilità 95%, $error_D(h)$ si trova nell'intervallo**

$$error_S(h) \pm 1.96\sqrt{\frac{error_S(h)(1-error_S(h))}{n}}$$

Intervalli di confidenza [2]

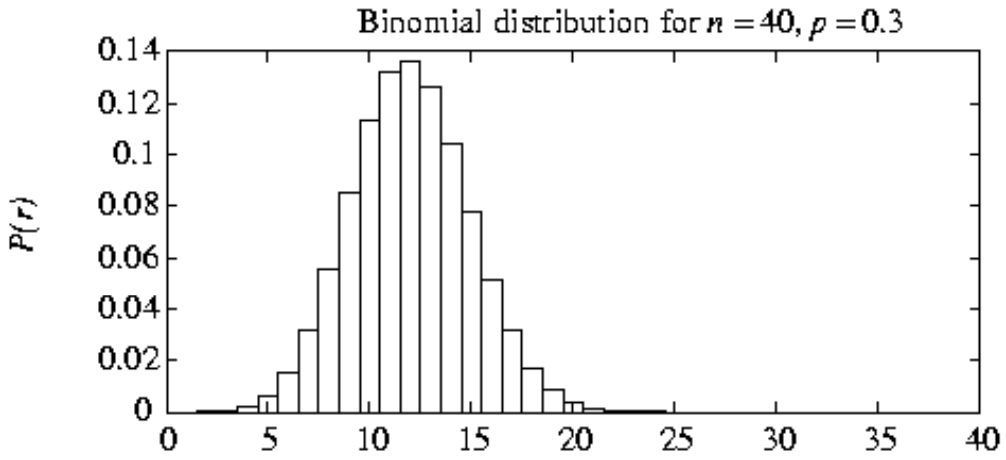
- **Se**
  - **S contiene n istanza**
  - **n>30**
- **allora**
  - **Con probabilità N%, *error$_D$(h)* si trova nell'intervallo**

$$error_S(h) \pm z_N \sqrt{\frac{error_S(h)(1-error_S(h))}{n}}$$

N%	50%	68%	80%	90%	95%	98%	99%
z_N	0.67	1.00	1.28	1.64	1.96	2.33	2.58

$error_S(h)$ è una variabile casuale

- **La probabilità di osservare r misclassificazioni:**

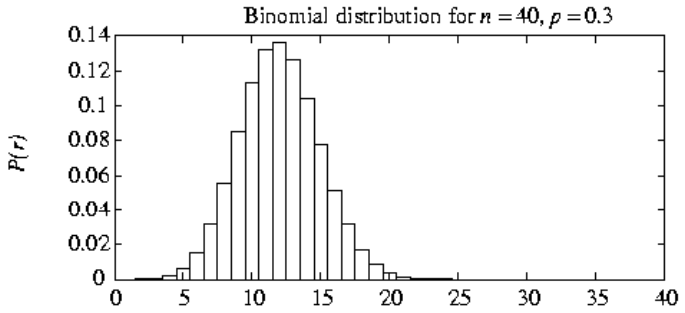


$$P(r) = \frac{n!}{r!(n-r)!} error_D(h)^r (1 - error_D(h))^{n-r}$$

Probabilità Binomiale

- **P(r) = probabilità di avere r teste nel lancio della monetina**
  - **P(head) = p**
- **Media**

$$E[X] = \sum_i P(i) = np$$



- **Varianza**

$$Var[X] = E\left[\left(X - E[X]\right)^2\right] = np(1-p)$$

- **Devianza**

$$\sigma_X = \sqrt{Var[X]} = \sqrt{np(1-p)}$$

$error_S(h)$

- **$error_S(h)$ segue una distribuzione binomiale**
  - **Per definizione,**

$$error_S(h) = \overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

$$X_i = \begin{cases} 0 & \text{se } c(x_i) = h(x_i) \\ 1 & \text{altrimenti} \end{cases}$$

  - **Assumendo**

$$E[X_i] = \mu$$

$$Var[X_i] = \sigma^2$$

  - **Otteniamo**

$$E[\overline{X}] = \mu$$

$$Var[\overline{X}] = \frac{\sigma^2}{n}$$

Approssimiamo $error_S(h)$

- **Media**

$$\mu_{error_S(h)} = error_D(h)$$

- **devianza**

$$\sigma_{error_S(h)} = \sqrt{\frac{error_D(h)(1-error_D(h))}{n}}$$

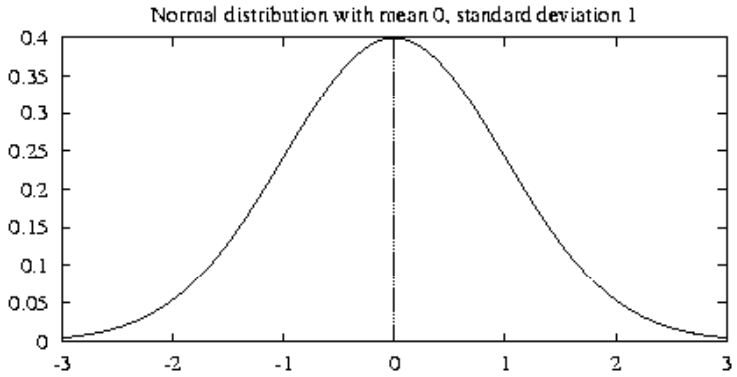
- **Utilizzando la distribuzione normale**
  - **media**

$$\mu_{error_S(h)} = error_D(h)$$

  - **varianza**

$$\sigma_{error_S(h)} \approx \sqrt{\frac{error_S(h)(1-error_S(h))}{n}}$$

Distribuzione Normale



- **densità**

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

- **distribuzione**

$$P(a \leq X < b) = \int_a^b p(x)\mathrm{d}x$$

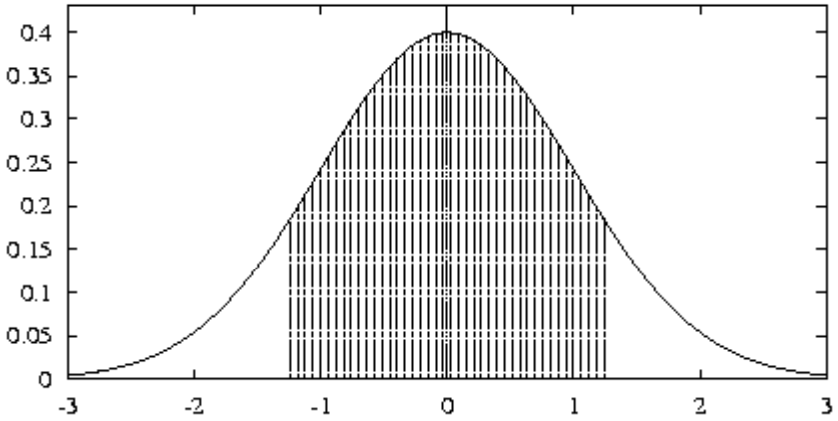
- **media**

$$E[X] = \mu$$

- **varianza**

$$Var[X] = \sigma^2$$

Distribuzione Normale



- 80% dell'area (probabilità) si trova in $\mu+1.28\sigma$
- N% dell'area (probabilità) si trova in $\mu+z_N\sigma$

N%	50%	68%	80%	90%	95%	98%	99%
z_N	0.67	1.00	1.28	1.64	1.96	2.33	2.58

Intervalli di confidenza

- Se S contiene n istanze, n>30
- allora
  - Con probabilità N%, $error_S(h)$ si trova nell'intervallo

$$error_D(h) \pm z_N \sqrt{\frac{error_D(h)(1-error_D(h))}{n}}$$

  - equivalentemente, $error_D(h)$ si trova nell'intervallo

$$error_S(h) \pm z_N \sqrt{\frac{error_D(h)(1-error_D(h))}{n}}$$

  - In base al teorema del Limite Centrale,

$$error_S(h) \pm z_N \sqrt{\frac{error_S(h)(1-error_S(h))}{n}}$$

Calcolo degli intervalli di confidenza

- **Si sceglie il parametro da stimare**
  - $error_D(h)$
- **Si sceglie un'approssimazione**
  - $error_S(h)$
- **Si determina la probabilità che governa l'approssimazione**
  - $error_S(h)$ **è binomiale, approssimata dalla distribuzione normale per n>30**
- **Si trovano gli intervalli (L,U) per cui N% della probabilità ricade in [L,U]**
  - **Si usa la tabella dei valori z_N**

L'approccio C4.5

- **Valore trasformato dell'errore (f):**

$$\frac{f-e}{\sqrt{e(1-e)/N}}$$

  - **(ovvero, sottraiamo la media e dividiamo per la devianza)**
  - **La distribuzione ottenuta è normale**

- **Equazione risultante:**

$$\Pr\left[-z \le \frac{f-e}{\sqrt{e(1-e)/N}} \le z\right] = c$$

- **Risolvendo per *p (assumendo il limite maggiore)*:**

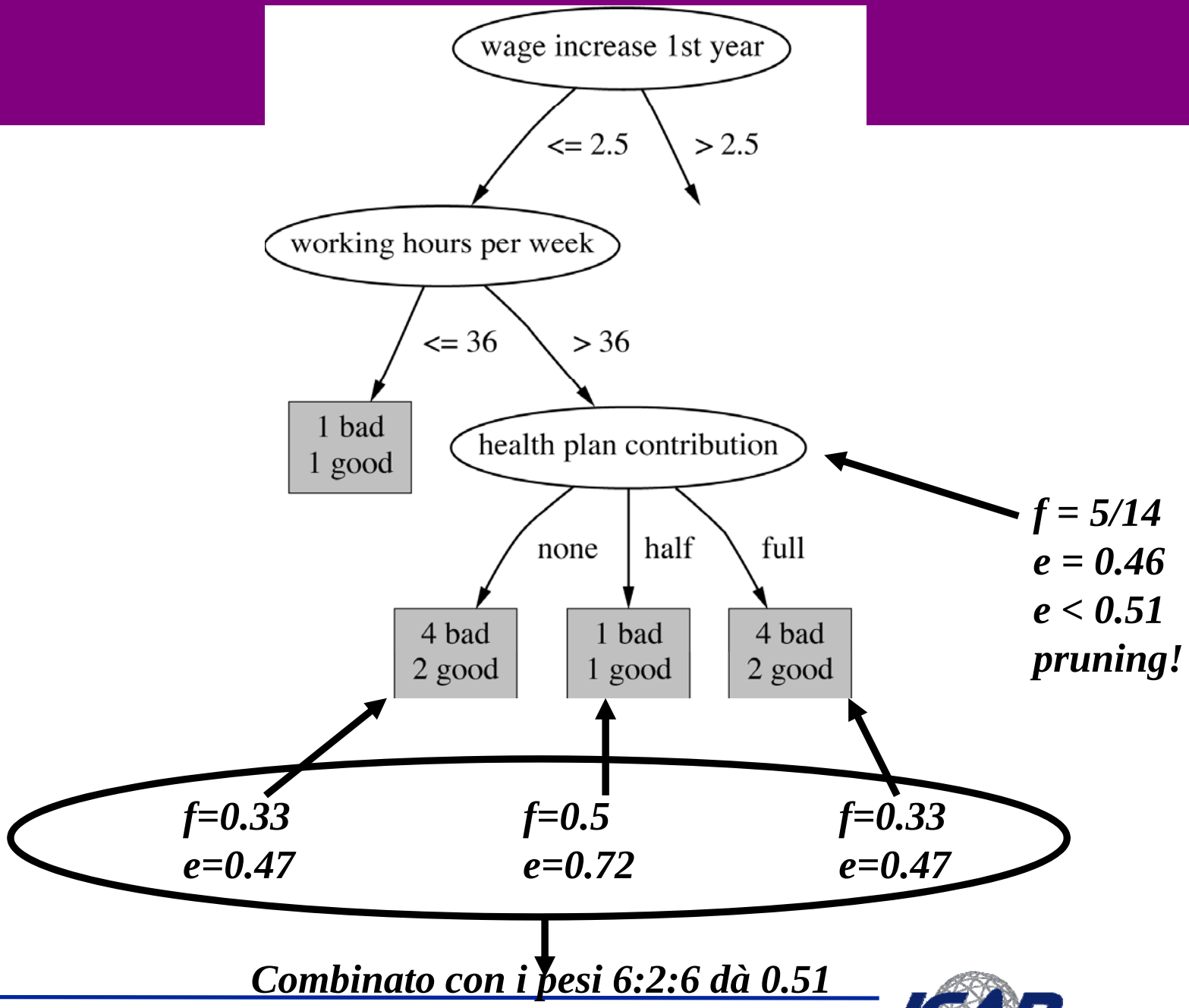
$$e \le \left(f + \frac{z^2}{2N} \pm z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}} \right) \Big/ \left(1 + \frac{z^2}{N}\right)$$

Il C4.5

- La stima dell'errore del sottoalbero è la somma pesata della stima degli errori delle sue foglie
- Stima dell'errore ad un nodo (upper bound):

$$e = \left(f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}} \right) \Bigg/ \left(1 + \frac{z^2}{N} \right)$$

- se c = 25% allora z = 0.69 (dalla distribuzione normale)
- f è l'errore del training set
- N è il numero di istanze nella foglia


wage increase 1st year
<= 2.5
> 2.5
working hours per week
<= 36
> 36
1 bad
1 good
health plan contribution
none
half
full
4 bad
2 good
1 bad
1 good
4 bad
2 good
f = 5/14
e = 0.46
e < 0.51
pruning!
f=0.33
e=0.47
f=0.5
e=0.72
f=0.33
e=0.47
Combinato con i pesi 6:2:6 dà 0.51

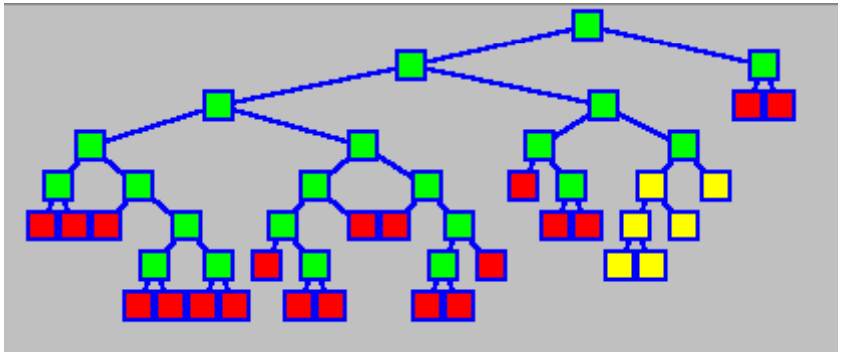
Pruning in CART

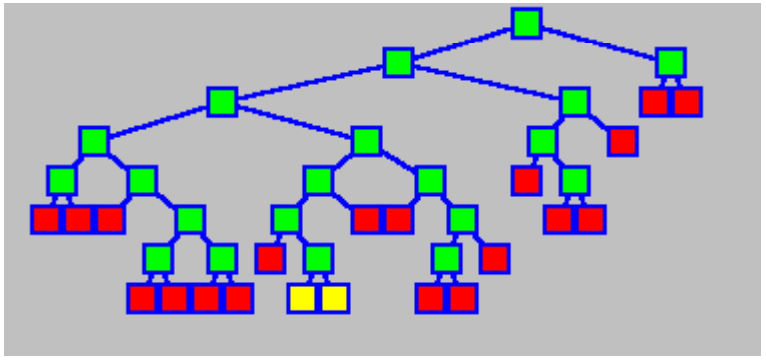
- Costruzione di un albero “massimale”
- CART determina una sequenza di pruning
  - L’ordine in cui i nodi dovrebbero essere rimossi

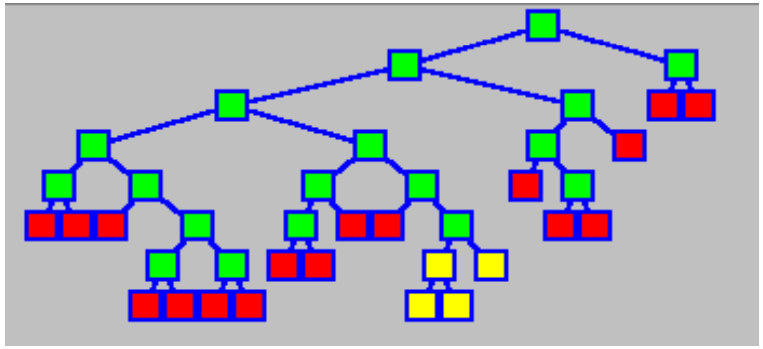
L’ordine di pruning

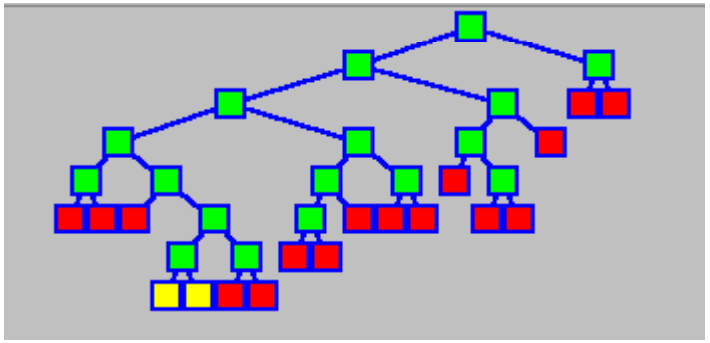
- Elimina il nodo “più debole” — Il nodo che aggiunge la minima accuratezza
  - I nodi più piccoli tendono ad essere rimossi prima
- Se più nodi hanno lo stesso contributo, vengono rimossi tutti

Esempio









Test della sequenza

- **Con il test set, scegliamo l'albero ottimale tra quelli ottenuti dalla sequenza di pruning**
  - **Le performance di tutti gli alberi sono misurate**
  - **L'ottimale non è necessariamente quello con l'errore minimo**
    - **Il più piccolo più vicino a quello d'errore minimo**

Sommario …

tool→	C4.5	CART	CHAID
Arietà dello split	Binario/multiplo	Binario	Multiplo
Criterio di split	information gain	gini index	χ^2
stop vs. pruning	prune	prune	Stop
Tipo di attributi	Categorico/continuo	Categorico/continuo	categorico

Sommario

- Attributi continui
- Valori mancanti
- Rasoio di Occam
  - Preference biases, language biases
- Overfitting
  - Prevenzione, aggiramento, aggiustamento