# Algoritmi di apprendimento

- Ci sono diversi algoritmi per l'apprendimento di alberi di decisione, fra questi:
  - Algoritmo di Hunt
  - ID3, C4.5
  - CART
  - ...



# Algoritmo di Hunt

L'albero viene costruito procedendo *ricorsivamente* e suddividendo il learning set in sottoinsiemi *via via più "puri"*. Il generico passo di suddivisione di un nodo dell'albero esegue questi passi:

**Dati**:
$D_t$ = sottoinsieme del learning set associato al nodo ***t***
$y=\{y_1, y_2, \cdots, y_c\}$ = insieme delle etichette che identificano le classi

**passo 1**: se tutte le istanze in $D_t$ appartengono alla stessa classe $y_t$ allora il nodo è una foglia etichettata dalla classe $y_t$ delle sue istanze

p**asso 2**: si sceglie un attributo fra quelli che descrivono le istanze, si produce un nodo figlio per ogni possibile valore dell'attributo (il range è dato dal learning set?).

A ciascun nodo figlio si associa uno specifico valore e
si associano ad asso anche quelle istanze, già associate al padre, per le quali l'attributo assume il valore corrispondente al nodo

# Algoritmo di Hunt

**nota 1**: se una certa combinazione di valori non è rappresentata da nessuna istanza, questa sarà associata alla classe di default (se esiste)

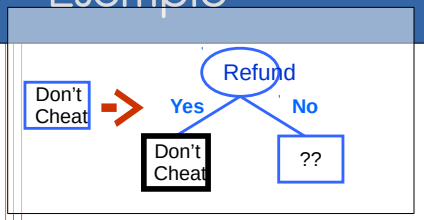
**nota 2**: se tutte le istanze associate a un nodo sono identiche come tuple ma corrispondono a classi differenti (non-determinismo), il nodo non può essere scisso.
Diventa una foglia che ha associata la classe più rappresentata

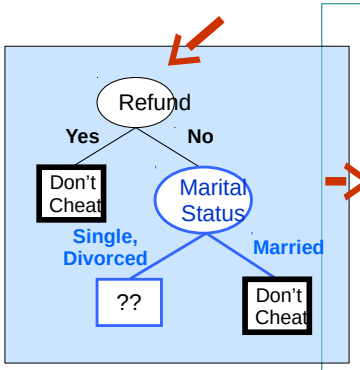
**nota 3**: quando si termina la costruzione dell'albero?

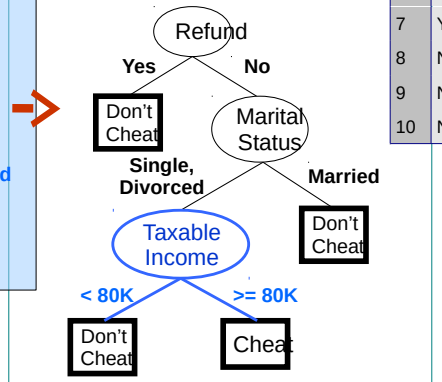
**nota 4**: come si sceglie l'attributo di split?



# Esempio



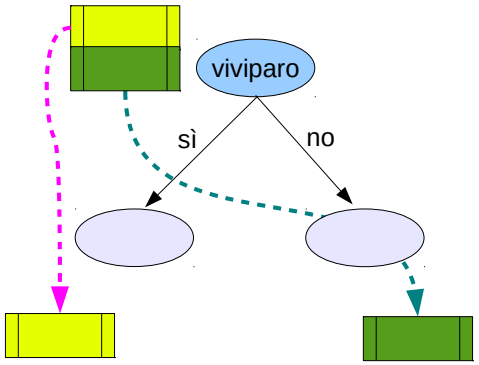

| *Tid* | Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |

# In generale ...

- **Strategia greedy**
  - Gli attributi su cui effettuare gli split sono selezionati in modo da massimizzare una qualche misura di riferimento
- **Problemi**:
  - Come **specificare la condizione di test** sugli attributi scelti?
    - *Attributi binari*
    - *Attributi nominali*
    - *Attributi ordinali*
    - *Attributi continui*
  - Come **determinare lo split migliore**?
  - **Quando fermarsi** nella costruzione dell'albero?



# Split su attributi binari




La risposta non può che essere sì oppure no

Il nodo corrente avrà due figli a seconda del valore rappresentato. Gli esempi associati al nodo radice verranno suddivisi fra i due figli a seconda del valore riportato in corrispondenza dell'attributo

**Altri esempi**: animale acquatico, volatile, a sangue caldo, ...

# Split su attributi nominali

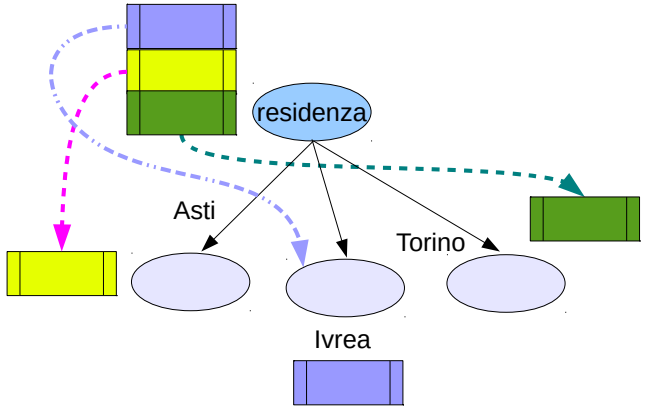
L'attributo assume valori su un insieme (finito) di etichette {L1, L2, ..., Ln}

Gli split possono essere **binari** oppure **multivalore**



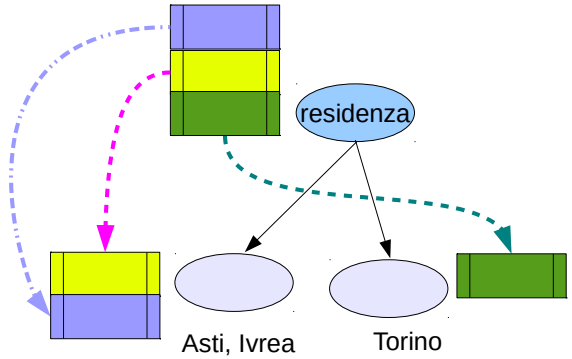
# Split su attributi nominali

**Split multivalore**: il nodo avrà tanti figli quanti sono i possibili valori dell'attributo



# Split su attributi nominali

**Split binari**: il nodo avrà due figli, uno corrisponde a un valore, l'altro all'insieme dei rimanenti valori



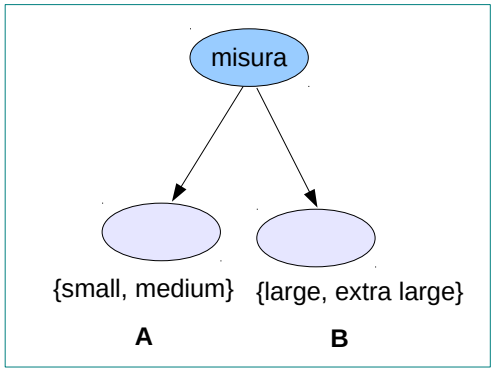

Ci sono $2^{k-1}-1$ possibili alternative se l'attributo ha k valori alternativi possibili



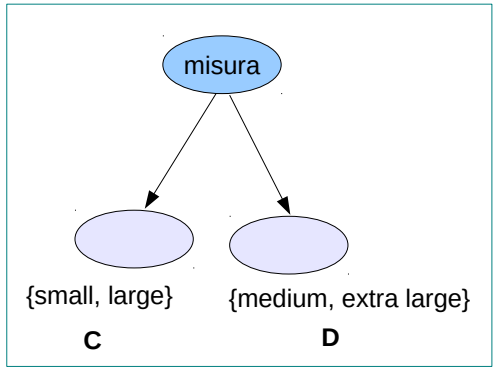
# Split su attributi ordinali

Anche in questo caso si possono avere split binari o multivalore con un vincolo: il raggruppamento dei valori deve rispettare l'ordinamento

**Esempio**: supponiamo di avere le misure *small* < *medium* < *large* < *extralarge*



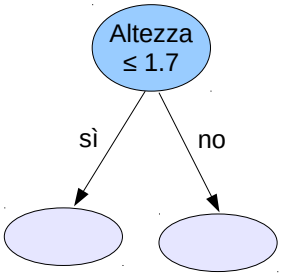

Corretto!! A < B




Errato!! C ?? D

# Split binari di attributi continui

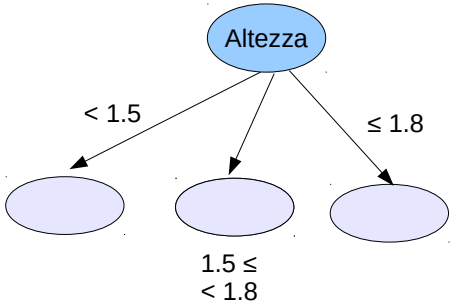
In questo caso il test prevedono l'identificazione di un valore possibile v per l'attributo A in questione






# Split multivalore di attributi continui

In questo caso il test prevedono l'identificazione di un insieme di valori v_i per l'attributo A in questione e la produzione di una serie di test $v_i \leq A < v_{i+1}$




Occorre **discretizzare** la variabile continua identificando un numero finito di intervalli significativi di valori

## Rischi?



## Rischi?

- Imparare per induzione significa:
  generalizzare gli esempi contenuti in un learning set

## Esempio

- Esempio
  distinguere “frutta” da altre cose



## Esempio

- Servono dei dati

## Esempio

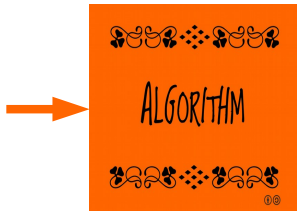
- Servono dei dati

Eccoli !



## Esempio

- Servono dei dati

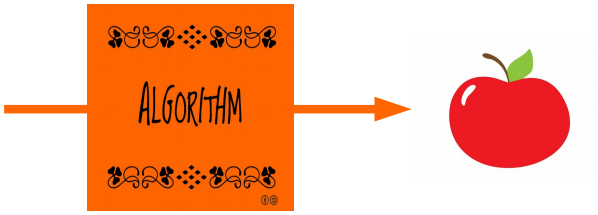



Esempi di frutti

## Esempio

- Servono dei dati

Esempi di frutti

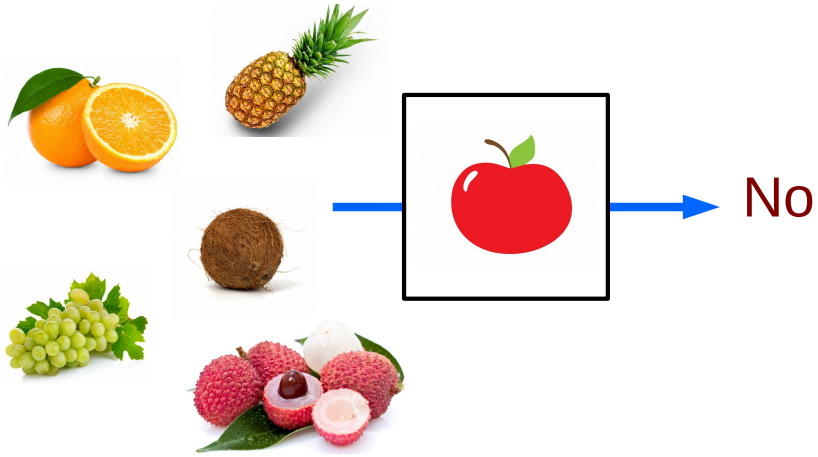



Modello di frutta (?)



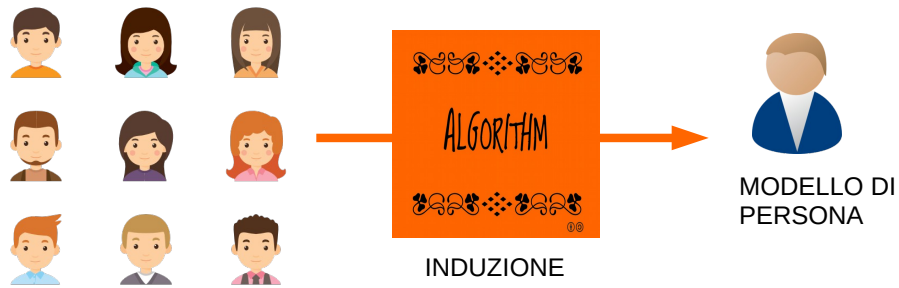
## Esempio

- Modello troppo povero
- Insufficiente, riconosce solo le mele



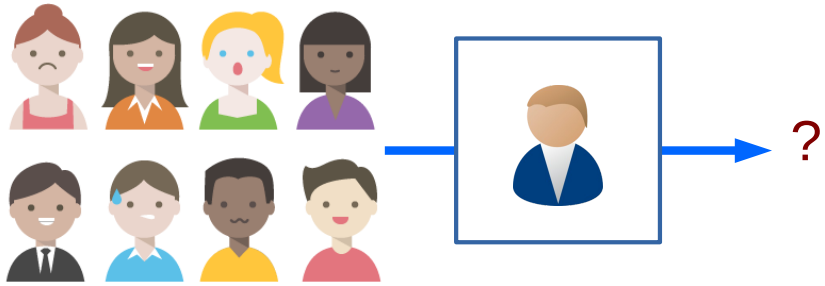
## Cambiamo domanda

- E se invece dei frutti dovessimo identificare persone?

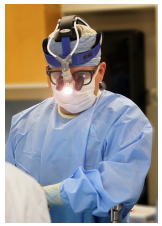




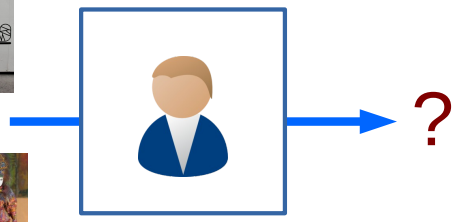

## Abbiamo un problema etico



# Varie varietà





Abiti, disabilità, forme, colori, ...



# Apprendimento umano ≠ automatico

Costruire un learning set è difficile

**Quel che è rilevante per un uomo è diverso da quel che è rilevante per uno strumento matematico**

**No**: instanze troppo simili. Non tutti i casi inclusi

# Apprendimento umano ≠ automatico

**No**: solo una ragazza verde. Statisticamente sopraffatta diventa un “errore tollerabile”

I dati sono generalizzati in modo statistico

Un piccolo errore è tollerabile

**Lo strumento costruirà un modello che tiene conto di tutti tranne la ragazza verde**



# Supporto ≠ Delega

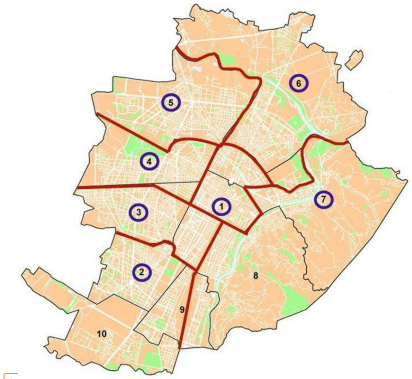
Sistemi di classificazione automatica usati spesso per supportare la decisione nei settori bancario e finanziario

# Sistemi di supporto alla decisione

Sistemi di classificazione automatica usati spesso per supportare la decisione nei settori bancario e finanziario.

**Dai giornali**:
Prestito universitario non assegnato a causa del quartiere di provenienza del richiedente

SUPPORTO o DELEGA?



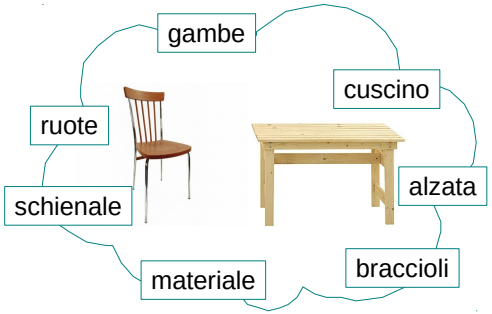

Usati meccanismi a base statistica
“ greedy”, massimizzano una funzione di utilità

Fz di utilità finalizzata a massimizzare il recupero del prestito per chi offre il servizio

Strumento privo di visione, sensibilità complessiva, sociale



# Quale split?




Supponiamo di dover costruire un DT che consenta di distinguere sedie da tavoli.
È indifferente l'ordine con il quale scegliamo di effettuare gli split?

Partire da *materiale* da *ruote* oppure da *schienale* fa qualche differenza?

# Bontà degli split

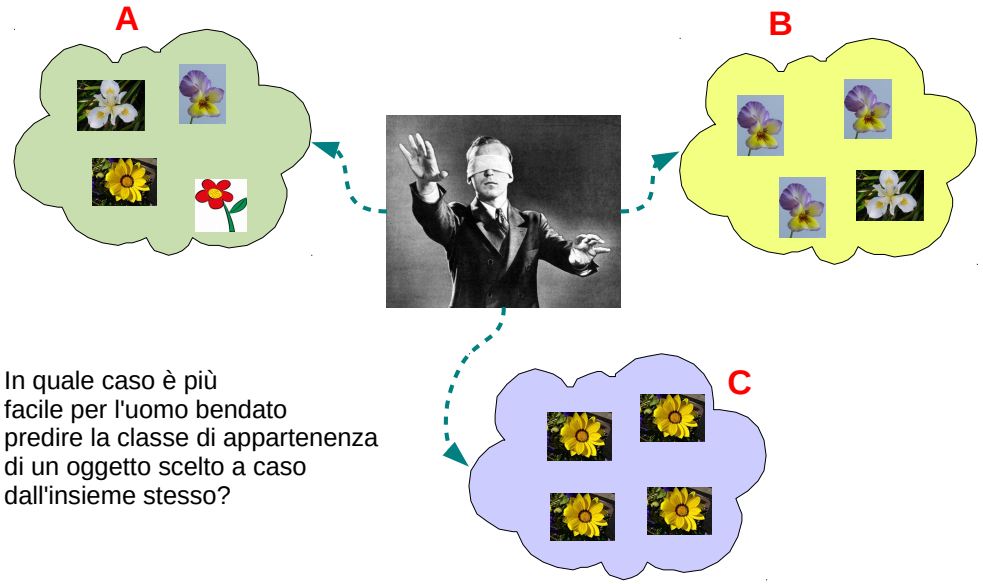
**Criterio generale**: alberi compatti sono preferiti ad alberi che consentono di raggiungere lo stesso grado di accuratezza (e di error rate) usando un maggior numero di test

**Rasoio di Occam**: a parità di assunzioni, la spiegazione più semplice è da preferire

William of Ockham

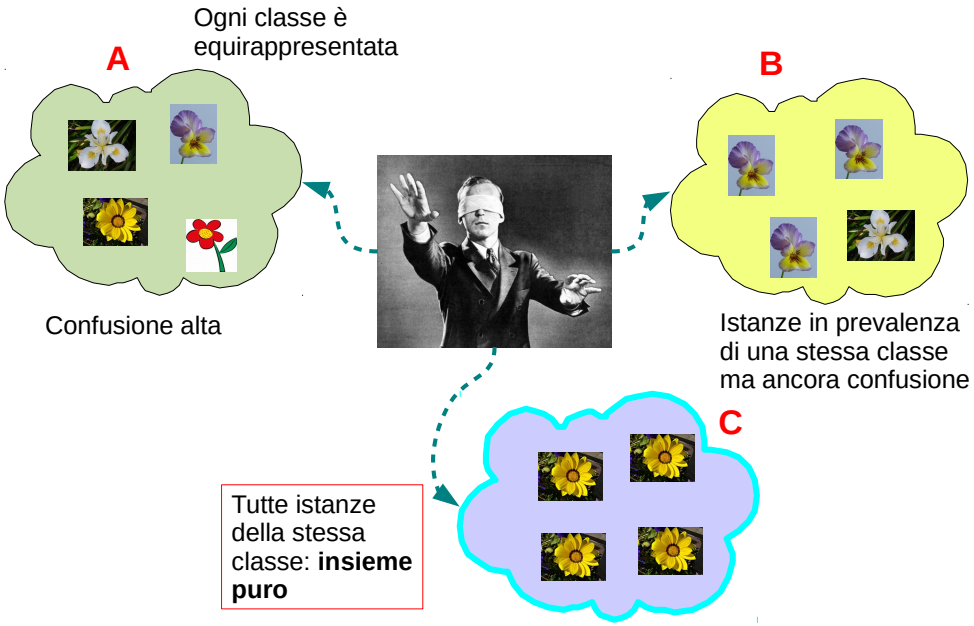


# Misure per determinare la bontà di uno split



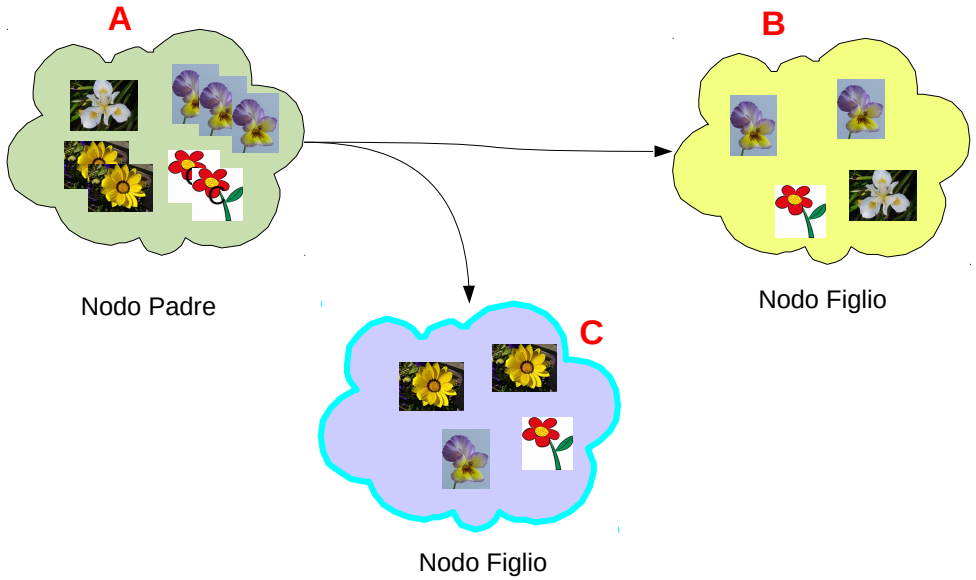

In quale caso è più facile per l'uomo bendato predire la classe di appartenenza di un oggetto scelto a caso dall'insieme stesso?

# Misure per determinare la bontà di uno split

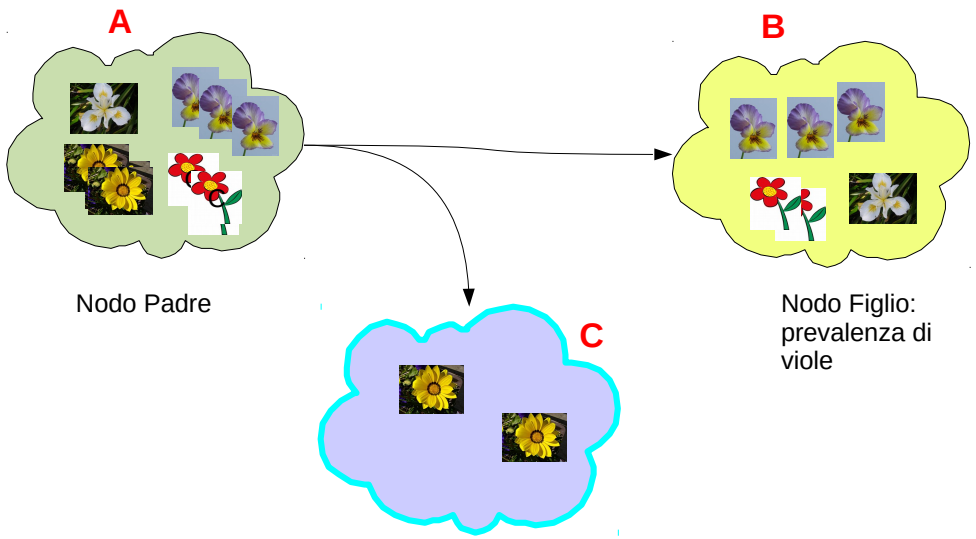





# È meglio questo split ...



## … o questo split?



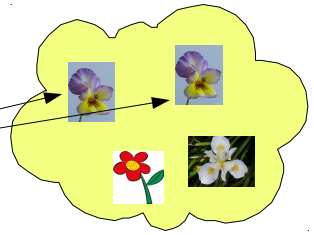



## Criterio generale

- Sono preferiti gli split che producono nodi figli la cui estensione prevede minore confusione (il cui grado di purezza è maggiore)
- Misure alternative:
  - *Entropia*
  - *Gini*
  - *Errore di classificazione*

# Base delle misure di selezione

Dato un nodo *t*, sia ***p(i | t)*** la probabilità che un elemento estratto casualmente dall'insieme sia di classe *i*



p(viola | t) = 0.5 (2 su 4)

p(iris | t) = 0.25 (1 su 4)

p(finto | t) = 0.25 (1 su 4)

(0.5, 0.25, 0.25) è la **distribuzione di probabilità** di appartenenza di un record estratto a caso dall'estensione associata al nodo *t* a una delle classi in questione



# Entropia

$$Entropia(t) = -\sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t)$$

Per ogni classe

Probabilità che l'elemento appartenga alla classe i-ma

Assunto nel calcolo dell'entropia: $0 \log_2 0 = 0$

# Entropia

$$Entropia(t) = -\sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t)$$

Assunto nel calcolo dell'entropia: $0 \log_2 0 = 0$

Supponiamo di avere due sole classi:

le distribuzioni (0, 1) e (1, 0) sono le migliori, purezza massima dell'insieme, nessuna confusione. Calcoliamo per il caso **(0, 1)**

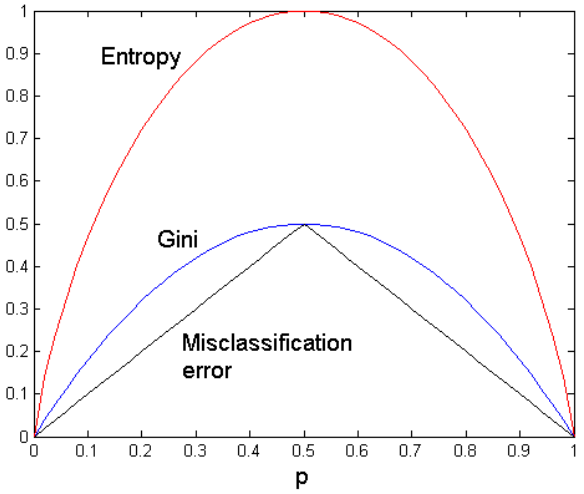
**Entropia**: $-0 \log_2 0 - 1 \log_2 1 = 0$

La distribuzione **(0.5, 0.5)** è la peggiore, massimo grado di confusione:

**Entropia**: $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = -\log_2 2^{-1} = 1$



# Misure: confronto

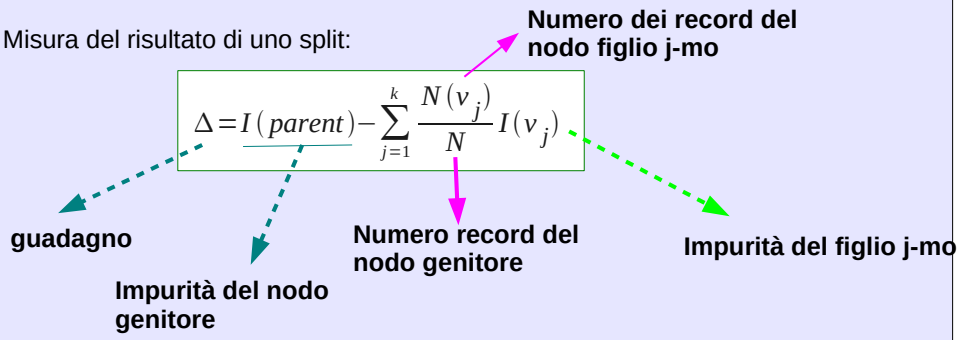



Valore dell'entropia per una sorgente binaria

# Calcolo del guadagno

**Problema della scelta dello split:** valuto i diversi split che posso fare, usando attributi diversi, tramite una delle misure viste e scelgo quello che mi restituisce il risultato col minor grado di confusione

Misura del risultato di uno split:




Dall'impurità del nodo genitore viene sottratta la media pesata delle impurità dei nodi figli. Di solito la misura dell'impurità è scelta in modo tale da minimizzare l'impurità / massimizzare il guadagno



# Information gain

Per **information gain** si intende una misura del guadagno ottenuta usando l'**entropia** come valore dell'impurità dei nodi:

$$\Delta = entropia(parent) - \sum_{j=1}^{k} \frac{N(v_j)}{N} entropia(v_j)$$

# Information gain

Per **information gain** si intende una misura del guadagno ottenuta usando l'**entropia** come valore dell'impurità dei nodi:

$$\Delta = entropia(parent) - \sum_{j=1}^{k} \frac{N(v_j)}{N} entropia(v_j)$$

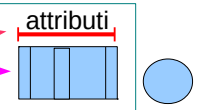
**Nota**: le misure del grado di confusione, come Gini ed entropia tendono a favorire attributi che hanno *molti valori diversi* rispetto ad attributi con *pochi valori* alternativi

**Osservazione**: un *identificatore univoco* (es. un numero di matricola) annulla l'entropia (ogni nodo figlio conterrà una sola istanza) ma non è un attributo significativo!

**Possibile soluzione**: usare solo split binari



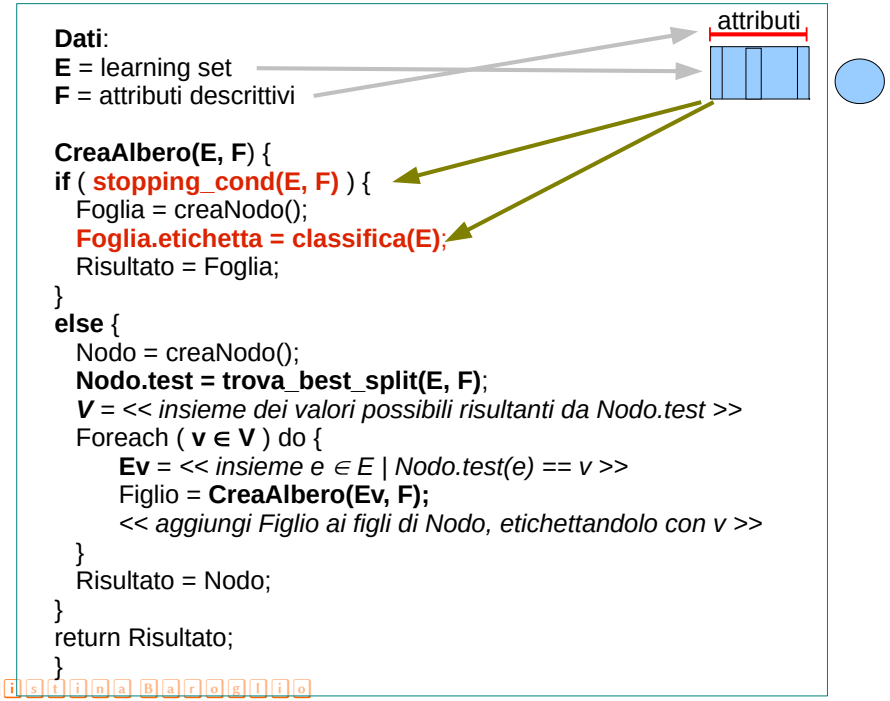
# Algoritmo




**Dati**:
**E** = learning set
**F** = attributi descrittivi

```
CreaAlbero(E, F) {
if ( stopping_cond(E, F) ) {
   Foglia = creaNodo();
   Foglia.etichetta = classifica(E);
   Risultato = Foglia;
}
else {
   Nodo = creaNodo();
   Nodo.test = trova_best_split(E, F);
   V = << insieme dei valori possibili risultanti da Nodo.test >>
   Foreach ( v ∈ V ) do {
        Ev = << insieme e ∈ E | Nodo.test(e) == v >>
        Figlio = CreaAlbero(Ev, F);
        << aggiungi Figlio ai figli di Nodo, etichettandolo con v >>
   }
   Risultato = Nodo;
}
return Risultato;
}
```

# Algoritmo

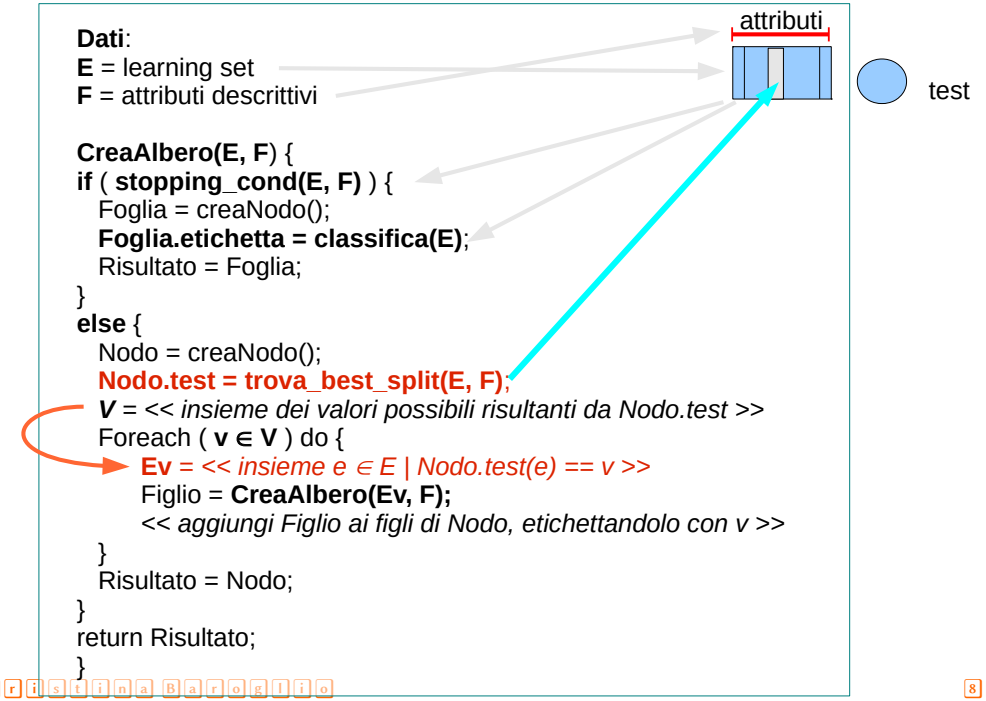



```
Dati:
E = learning set
F = attributi descrittivi

CreaAlbero(E, F) {
if ( stopping_cond(E, F) ) {
   Foglia = creaNodo();
   Foglia.etichetta = classifica(E);
   Risultato = Foglia;
}
else {
   Nodo = creaNodo();
   Nodo.test = trova_best_split(E, F);
   V = << insieme dei valori possibili risultanti da Nodo.test >>
   Foreach ( v ∈ V ) do {
        Ev = << insieme e ∈ E | Nodo.test(e) == v >>
        Figlio = CreaAlbero(Ev, F);
        << aggiungi Figlio ai figli di Nodo, etichettandolo con v >>
   }
   Risultato = Nodo;
}
return Risultato;
}
```



# Algoritmo



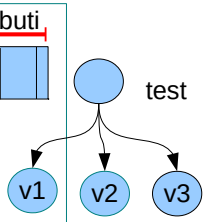

```
Dati:
E = learning set
F = attributi descrittivi

CreaAlbero(E, F) {
if ( stopping_cond(E, F) ) {
   Foglia = creaNodo();
   Foglia.etichetta = classifica(E);
   Risultato = Foglia;
}
else {
   Nodo = creaNodo();
   Nodo.test = trova_best_split(E, F);
   V = << insieme dei valori possibili risultanti da Nodo.test >>
   Foreach ( v ∈ V ) do {
        Ev = << insieme e ∈ E | Nodo.test(e) == v >>
        Figlio = CreaAlbero(Ev, F);
        << aggiungi Figlio ai figli di Nodo, etichettandolo con v >>
   }
   Risultato = Nodo;
}
return Risultato;
}
```

# Algoritmo

```
Dati:
E = learning set
F = attributi descrittivi

CreaAlbero(E, F) {
if ( stopping_cond(E, F) ) {
   Foglia = creaNodo();
   Foglia.etichetta = classifica(E);
   Risultato = Foglia;
}
else {
   Nodo = creaNodo();
   Nodo.test = trova_best_split(E, F);
   V = << insieme dei valori possibili risultanti da Nodo.test >>
   Foreach ( v ∈ V ) do {
      Ev = << insieme e ∈ E | Nodo.test(e) == v >>
      Figlio = CreaAlbero(Ev, F);
      << aggiungi Figlio ai figli di Nodo, etichettandolo con v >>
   }
   Risultato = Nodo;
}
return Risultato;
}
```






# Dettagli

- trova_best_split: può essere individuato per esempio tramite il calcolo dell'entropia;
- classifica: può per esempio restituire la classe più rappresentata
- stopping_cond: può restituire vero per esempio quando tutte le istanze associate al nodo appartengono alla stessa classe oppure quando il numero di istanze è al di sotto di una certa soglia

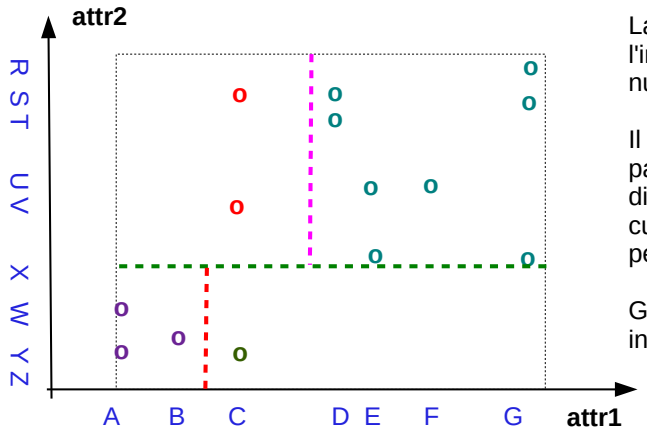
# Partizionamento dello spazio

Supponiamo di poter rappresentare le istanze del learning set come punti in uno spazio multidimensionale: ogni test corrisponde a un **taglio** (una **partizione**) di tale spazio, fatta lavorando su un **singolo attributo**



# Partizionamento dello spazio

Supponiamo per semplicità di avere due soli attributi, corrispondenti ai due assi cartesiani. Le lettere rappresentano i valori possibili dei due attributi. Sono stati ordinati con un qualche criterio (anche solo associando numeri a label)



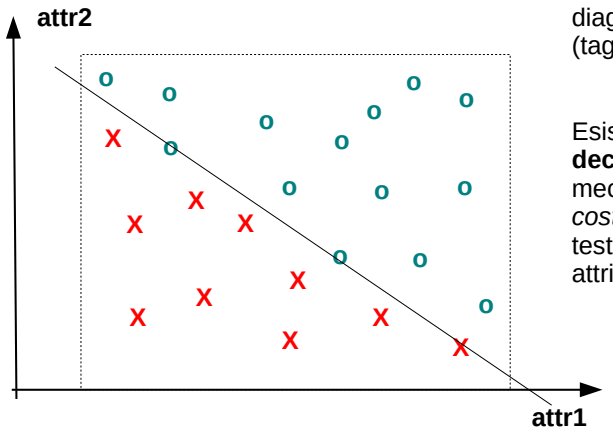

La radice ha associato l'intero learning set, contenuto nel rettangolo

Il primo test (linea verde) partiziona il learning set dividendo gli esempi per cui **[attr2 >= X]** da quelli per cui **[attr2 < X]**

Gli altri test partizionano insiemi più piccoli di esempi

# Partizionamento dello spazio

E se gli esempi fossero messi così?




Lavorando su un singolo attributo non è possibile effettuare tagli diagonali o più complessi (tagli curvi)

Esistono in letteratura **alberi di decisione obliqui**, prodotti da meccanismi noti come *induzione costruttiva*, in grado di realizzare test basati su composizioni di attributi



# Induzione di alberi di decisione: commenti 1/2

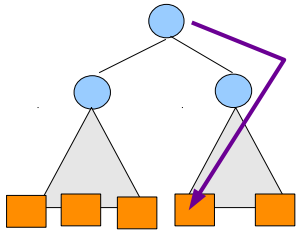
- Gli **algoritmi di induzione** di DT sono **non-parametrici**, non occorrono particolari assunzioni sulle distribuzioni di probabilità
- La costruzione di un albero ottimale è un **problema NP-completo,** solitamente si adottano delle euristiche
- La costruzione di un DT è **computazionalmente poco costosa**; dato un albero, la **classificazione** ha una **complessità nel caso peggiore O(w)**, dove w rappresenta la profondità dell'albero
- Un DT è di **semplice interpretazione**, soprattutto se l'albero è piccolo
- I DT **non sono adatti a risolvere certi problemi di tipo booleano**, ad esempio a calcolare la funzione di parità (restituisci 1 se il #1 in una sequenza di bit è pari, 0 altrimenti) – vedere esercizio pag 198
- La presenza di **attributi irrilevanti non influenza negativamente** la costruzione dell'albero

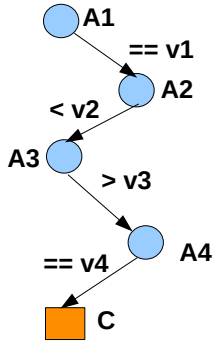
# Induzione di alberi di decisione: commenti 2/2

- È possibile incorrere nella **frammentazione dei dati**: procedendo top-down a un certo punto i nodi hanno associato un numero di istanze troppo piccolo per essere statisticamente significativo
- Si può avere **replicazione di sottoalberi**
- Partizionamento dello spazio tramite **tagli rettilinei e paralleli agli assi**
- Poiché molte **misure di impurità sono consistenti le une con le altre**, variare funzione di impurità spesso non modifica sostanzialmente la qualità degli alberi costruiti



# Interpretazione di un DT






**If (A1 == v1 && A2 < v2 && A3 > v3 && A4 == v4) then C**