

Machine Learning

Università Roma Tre
Dipartimento di Ingegneria
Anno Accademico 2021 - 2022

Classificazione:
Algoritmo C4.5

Algoritmi Induzione DT

- Algoritmo Greedy Decision Tree Learning
 - Scelta della migliore feature utilizzando come metrica il **Classification Error** sui dati di training —> problema NP-Hard —> servono euristiche
- Algoritmo C4.5

Algoritmo C4.5

- J. Ross Quinlan. 1993. **C4.5: Programs for Machine Learning.** Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- X. Wu, V. Kumar, **J. R. Quinlan**, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. 2007. **Top 10 Algorithms in Data Mining.** *Knowledge and Information Systems*, Volume 14, Issue 1, December 2007, Pages 1-37, Springer-Verlag New York, Inc. New York, NY, USA. DOI=<http://dx.doi.org/10.1007/s10115-007-0114-2>

Algoritmo C4.5

Knowl Inf Syst (2008) 14:1–37
DOI 10.1007/s10115-007-0114-2

SURVEY PAPER

Top 10 algorithms in data mining

Xindong Wu · Vipin Kumar · J. Ross Quinlan · Joydeep Ghosh · Qiang Yang ·
Hiroshi Motoda · Geoffrey J. McLachlan · Angus Ng · Bing Liu · Philip S. Yu ·
Zhi-Hua Zhou · Michael Steinbach · David J. Hand · Dan Steinberg

Received: 9 July 2007 / Revised: 28 September 2007 / Accepted: 8 October 2007
Published online: 4 December 2007
© Springer-Verlag London Limited 2007

Abstract This paper presents the top 10 data mining algorithms identified by the IEEE International Conference on Data Mining (ICDM) in December 2006: C4.5, k -Means, SVM, Apriori, EM, PageRank, AdaBoost, k NN, Naive Bayes, and CART. These top 10 algorithms are among the most influential data mining algorithms in the research community. With each algorithm, we provide a description of the algorithm, discuss the impact of the algorithm, and review current and further research on the algorithm. These 10 algorithms cover classification, clustering, statistical learning, association analysis, and link mining, which are all among the most important topics in data mining research and development.

Algoritmo C4.5

- Obiettivo: generare un Albero di Decisione da una Tabella di Dati
- Sviluppato da J. R. Quinlan nel 1993 come estensione dell'*Algoritmo ID3*
- L'Albero ottenuto può essere usato per la classificazione, per cui l'*Algoritmo C4.5* è spesso indicato come *Statistical Classifier*
- Basato sulla Teoria dell'Informazione (Claude E. Shannon, *A Mathematical Theory of Communication*, 1948)
- Strategia “*divide and conquer*” (suddivisione del problema in sottoproblemi più semplici e loro risoluzione ricorsiva):
 - Scelta di uno degli attributi come nodo radice
 - Creazione ramo per ciascun valore di quell'attributo
 - Suddivisione delle istanze lungo i rami
 - Ripetizione del processo per ciascun ramo finché tutti le istanze nel ramo hanno la stessa classe di appartenenza (si dice che tutti i sottoalberi sono “puri”)
- Assunzione di fondo: quanto più semplice è l'albero che classifica le istanze, tanto meglio è

Entropia

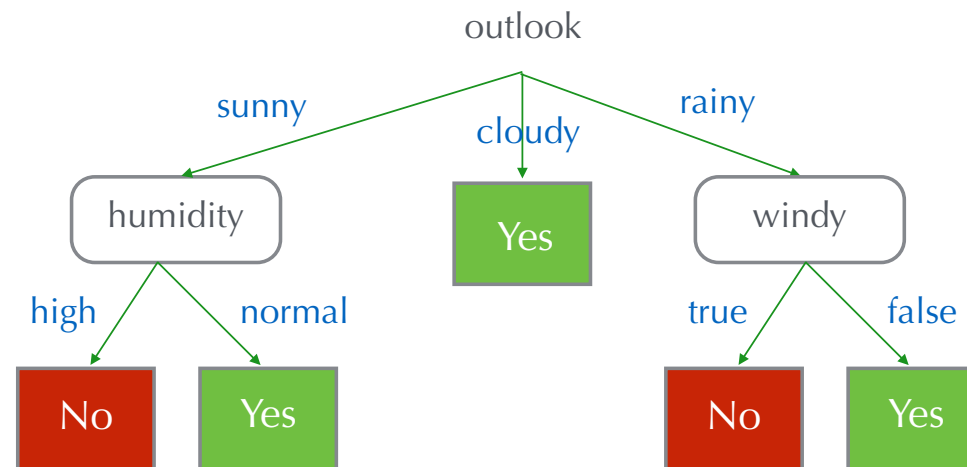
Introduciamo il concetto di *Entropia* (*Entropy*) [(dal greco antico ἐν en, "dentro", e τροπή tropé, "trasformazione")]

- Entropia in Meccanica Statistica: grandezza interpretata come misura del disordine presente in un sistema fisico qualsiasi, incluso - come caso limite - l'universo
- Entropia in Teoria dell'Informazione: quantità di incertezza o informazione presente in un segnale aleatorio
 - Primo Teorema di Shannon (Codifica di Sorgente): *“Una sorgente casuale d'informazione non può essere rappresentata con un numero di bit (da cui la base 2 del logaritmo) inferiore alla sua entropia, cioè alla sua autoinformazione media.”*
Tale teorema ha quindi un'implicazione in termini di rappresentazione dati, in quanto l'Entropia può essere interpretata anche come la minima complessità descrittiva di una variabile aleatoria, ovvero il limite inferiore della compressione dei dati

Tabella di Dati

<i>Outlook</i>	<i>Temperature</i>	<i>Humidity</i>	<i>Windy</i>	<i>Play</i>
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Cloudy	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Cloudy	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Cloudy	Mild	High	True	Yes
Cloudy	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Decision Tree



- Nodi interni = test sugli attributi (feature)
- Archi uscenti = risultati dei test
- Nodi foglia = etichette classe di appartenenza

Entropia

Introduciamo il concetto di *Entropia* (*Entropy*) di un set di istanze

- S è un set di istanze (i.e., record della tabella)
- A è una *feature* (*Play* nell'esempio)
- $\{S_1 \dots S_i \dots S_n\}$ sono le *partizioni* di S secondo gli n valori che può assumere A ("Yes" e "No" nell'esempio)
- $\{p_1 \dots p_i \dots p_n\}$ sono le *proporzioni* di $\{S_1 \dots S_i \dots S_n\}$ in S

Si definisce *Entropia* di S la seguente grandezza

$$Entropy(S) = \sum_{i=1}^n (-p_i * \log_2 p_i)$$

Entropia

Nel caso dell'esempio

Outlook	Temperature	Humidity	Windy	Play	
Sunny	Hot	High	False	No	"No" case "Yes" case
Cloudy	Hot	High	False	Yes	
...	

- S è il set di 14 istanze
- L'obiettivo è classificare le istanze secondo i valori della feature *Play*, ossia "Yes" e "No"
- La proporzione delle istanze con valore "Yes" è 9 su 14 ($9/14=0.64$)
- La proporzione delle istanze con valore "No" è 5 su 14 ($5/14=0.36$)
- L'Entropia misura l'impurezza di S e in questo caso vale $\text{Entropy}(S) = -0.64 (\log_2 0.64) - 0.36 (\log_2 0.36) =$
 $= -0.64 (-0.644) - 0.36 (-1.474) = 0.41 + 0.53 = 0.94$

Guadagno

Introduciamo ora il *Guadagno* (*Gain*) di un attributo A

- Calcolo di $Gain(S, A)$ per ciascun attributo A
 - *Riduzione* di Entropia attesa a seguito dell'ordinamento del set di istanze S basato su A
- Scelta dell'attributo con il valore di Guadagno più elevato come nodo dell'albero
- $Gain(S, A) = Entropy(S) - Expectation(A)$

$$Gain(S, A) = Entropy(S) - \sum_{i=1}^n \frac{|S_i|}{|S|} * Entropy(S_i)$$

dove $\{S_1 \dots S_i \dots S_n\}$ sono le partizioni di S secondo i valori dell'attributo A , n il numero di valori distinti di A , $|S_i|$ il numero di istanze nella partizione S_i e $|S|$ il numero totale di istanze in S

Scelta Nodo Radice

Se *Outlook* è radice dell'albero ci sono 3 partizioni sulle istanze (S_1 per *Sunny*, S_2 per *Cloudy*, S_3 per *Rainy*)

- S_1 (*Sunny*) = {istanze 1,2,8,9,11}
- $|S_1| = 5$ (di queste 5 istanze, i valori per *Play* sono 3 *No* e 2 *Yes*)

$$\begin{aligned} \text{Entropy}(S_1) &= \\ &= -2/5 (\log_2 2/5) - 3/5 (\log_2 3/5) = \\ &= -0.4 (-1.322) - 0.6 (-0.737) = \\ &= 0.53 + 0.44 = 0.97 \end{aligned}$$

Analogamente si ottiene

$$\text{Entropy}(S_2) = 0$$

$$\text{Entropy}(S_3) = 0.97$$

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Cloudy	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Cloudy	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Cloudy	Mild	High	True	Yes
Cloudy	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Scelta Nodo Radice

$$\begin{aligned} \text{Gain}(S, \text{Outlook}) &= \text{Entropy}(S) - \text{Expectation}(\text{Outlook}) = \\ &= \text{Entropy}(S) - [|S_1|/|S| * \text{Entropy}(S_1) + |S_2|/|S| * \text{Entropy}(S_2) + \\ &+ |S_3|/|S| * \text{Entropy}(S_3)] = 0.94 - [5/14 * 0.97 + 4/14 * 0 + 5/14 * 0.97] \end{aligned}$$

da cui si ottiene

$$\text{Gain}(S, \text{Outlook}) = 0.247$$

Analogamente

$$\text{Gain}(S, \text{Temperature}) = 0.029$$

$$\text{Gain}(S, \text{Humidity}) = 0.152$$

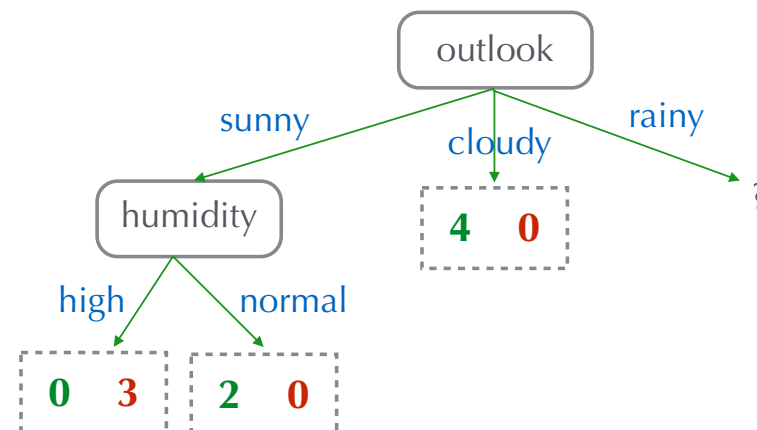
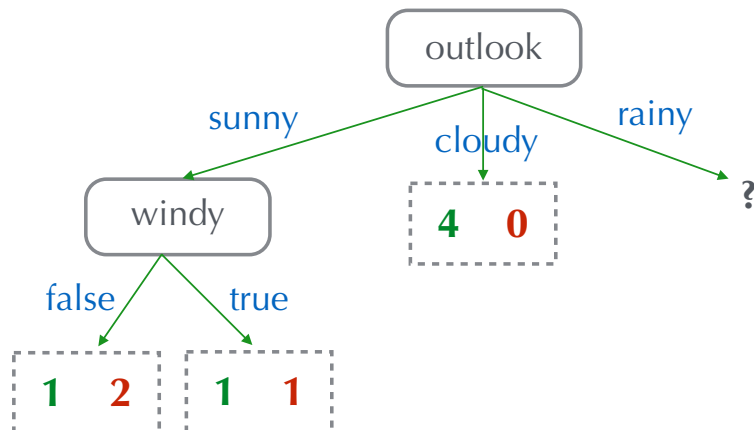
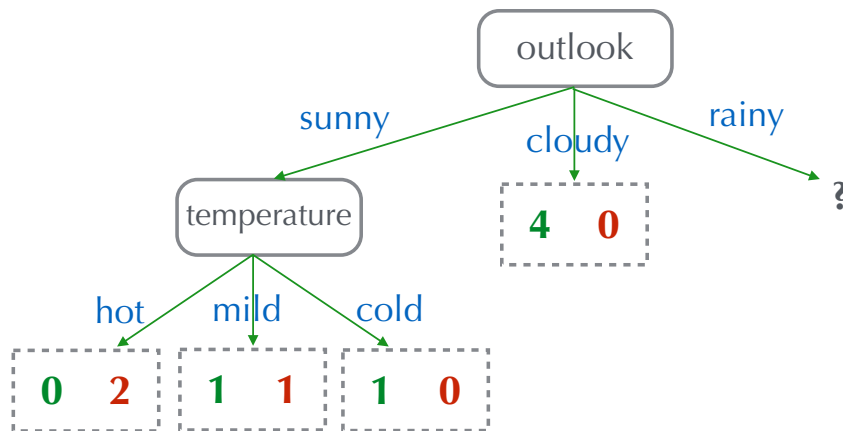
$$\text{Gain}(S, \text{Windy}) = 0.048$$

In conclusione $\text{Gain}(S, \text{Outlook})$ è il guadagno più elevato e quindi *Outlook* dovrebbe essere scelto come radice dell'Albero di Decisione

Scelta Nodi Successivi

Ripetiamo il procedimento per il ramo *Sunny* ...

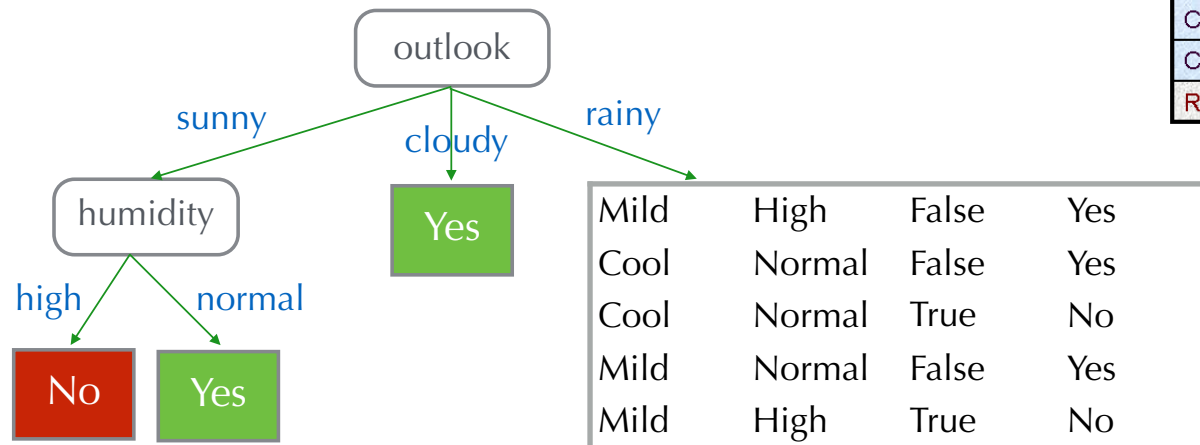
Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Cloudy	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Cloudy	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Cloudy	Mild	High	True	Yes
Cloudy	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



Scelta Nodi Successivi

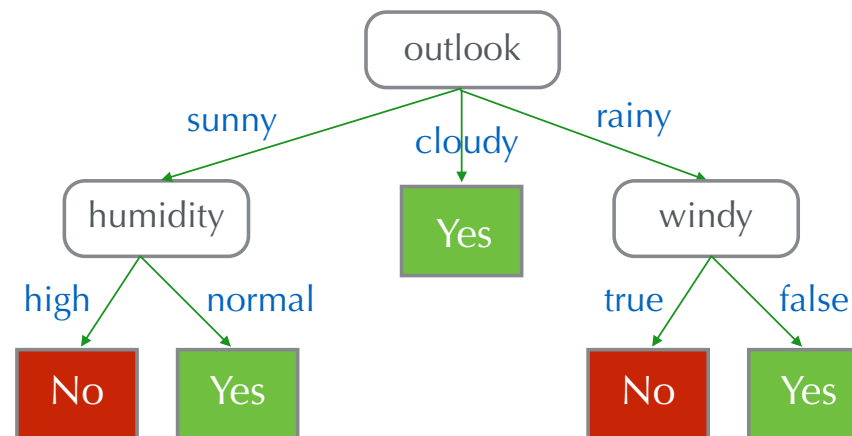
... e per il ramo Rainy

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Cloudy	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Cloudy	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Cloudy	Mild	High	True	Yes
Cloudy	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



Decision Tree

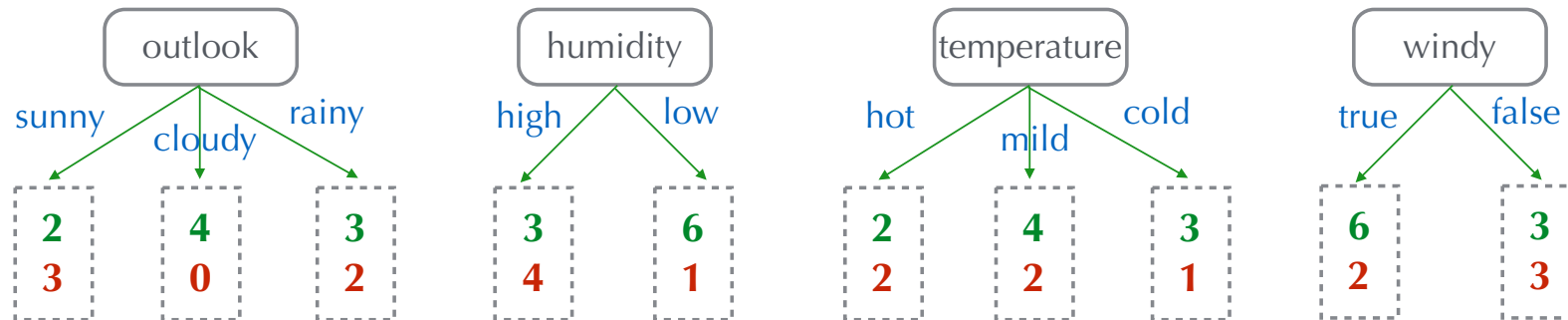
In conclusione, si ottiene il seguente Albero di Decisione



- Nodi interni = test sugli attributi (feature)
- Archi uscenti = risultati dei test
- Nodi foglia = etichette classe di appartenenza

Scelta Nodo Radice


Qual è l'attributo migliore per essere nodo radice dell'albero?



La selezione dell'attributo come nodo radice è eseguita valutando il *Guadagno di Informazione* (*Information Gain*) per ciascun attributo e scegliendo quello che dà il valore maggiore

Algoritmo C4.5

Algorithm 1.1 C4.5(D)

Input: an attribute-valued dataset D  Set di dati (tabella) attributo-valore

- 1: $Tree = \{\}$
- 2: **if** D is “pure” OR other stopping criteria met **then**
- 3: terminate
- 4: **end if**
- 5: **for all** attribute $a \in D$ **do**
- 6: Compute information-theoretic criteria if we split on a
- 7: **end for**
- 8: a_{best} = Best attribute according to above computed criteria
- 9: $Tree$ = Create a decision node that tests a_{best} in the root
- 10: D_v = Induced sub-datasets from D based on a_{best}
- 11: **for all** D_v **do**
- 12: $Tree_v = C4.5(D_v)$
- 13: Attach $Tree_v$ to the corresponding branch of $Tree$
- 14: **end for**
- 15: **return** $Tree$

N.B. *Pure*: all instances in the subset fall in the same class

Algoritmo C4.5

Salvatore Ruggieri. 2000. **Efficient C4.5**. Technical Report. University of Pisa.

Abstract: We present an analytic evaluation of the run-time behavior of the C4.5 algorithm which highlights some efficiency improvements. We have implemented a more efficient version of the algorithm, called EC4.5, that improves on C4.5 by adopting the best among three strategies at each node construction. The first strategy uses a binary search of thresholds instead of the linear search of C4.5. The second strategy adopts a counting sort method instead of the quicksort of C4.5. The third strategy uses a main-memory version of the RainForest algorithm for constructing decision trees. Our implementation computes the same decision trees as C4.5 with a performance gain of up to 5 times.

Esercizio

Creare l'Albero di Decisione (Indice) per la
Previsione di Rischio per Richieste di Prestito

	Income	Credit Rating	Debt	Collateral	<i>Risk</i>
1	\$0 to \$15k	bad	high	none	<i>high</i>
2	\$15 to \$35k	unknown	high	none	<i>high</i>
3	\$15 to \$35k	unknown	low	none	<i>moderate</i>
4	\$0 to \$15k	unknown	low	none	<i>high</i>
5	over \$35k	unknown	low	none	<i>low</i>
6	over \$35k	unknown	low	adequate	<i>low</i>
7	\$0 to \$15k	bad	low	none	<i>high</i>
8	over \$35k	bad	low	adequate	<i>moderate</i>
9	over \$35k	good	low	none	<i>low</i>
10	over \$35k	good	high	adequate	<i>low</i>
11	\$0 to \$15k	good	high	none	<i>high</i>
12	\$15 to \$35k	good	high	none	<i>moderate</i>
13	over \$35k	good	high	none	<i>low</i>
14	\$15 to \$35k	bad	high	none	<i>high</i>

Esercizio

[illegible]

Esercizio

$$Entropy(S) = \sum_{i=1}^n - p_i * \log_2 p_i$$

$$Expectation(A) = \sum_{i=1}^n \frac{|S_i|}{|S|} * Entropy(S_i)$$

$$Gain(S, A) = Entropy(S) - Expectation(A)$$

Algoritmi Induzione DT

- Algoritmo Greedy Decision Tree Learning
 - Scelta della migliore feature utilizzando come metrica il **Classification Error** sui dati di training —> problema NP-Hard —> servono euristiche
- Algoritmo C4.5
 - Scelta della migliore feature utilizzando come metrica l'**Information Gain** sui dati di training —> problema risolvibile tramite strategia *divide&conquer*

Algoritmi Induzione DT

- Algoritmo Greedy Decision Tree Learning
 - Scelta della migliore feature utilizzando come metrica il **Classification Error** sui dati di training —> problema NP-Hard —> servono euristiche
- Algoritmo C4.5
 - Scelta della migliore feature utilizzando come metrica l'**Information Gain** sui dati di training —> problema risolvibile tramite strategia *divide&conquer*
- Algoritmo CART

Algoritmo CART

- Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). **Classification and Regression Trees**. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software.

Algoritmo CART

- Obiettivo: generare un Albero di Decisione da una Tabella di Dati
- Si basa sul *Gini Index* (o *Indice di Gini*)
- In corrispondenza di un certo nodo t dell'albero in costruzione, e rispetto alla corrispondente partizione del dataset di training, si definisce l'Indice di Gini come segue:

$$Gini(t) = 1 - \sum_j [p(j/t)]^2$$

dove $p(j/t)$ è la frequenza relativa (proporzione) della classe j al nodo t

- L'Indice di Gini misura l'*impurezza* (o *disordine*) del dataset corrispondente a t
 - Massimo valore ($1-1/n_c$, con n_c =numero di classi equiprobabili) quando i record sono equamente distribuiti fra tutte le classi
 - Minimo valore (0) quando tutti i record appartengono a una sola classe

Indice di Gini

$$Gini(t) = 1 - \sum_j [p(j/t)]^2$$

- Nel caso di una sola classe:

$$Gini(t) = 1 - 1^2 = 0$$

- Nel caso di n_c classi equiprobabili

$$Gini(t) = 1 - \sum_j ((n/n_c)/n)^2 = 1 - \sum_j (1/n_c)^2 = 1 - n_c(1/n_c)^2 = 1 - 1/n_c$$

dove n è il numero di record del dataset al nodo t

Indice di Gini

$$Gini(t) = 1 - \sum_j [p(j/t)]^2$$

- C1=0, C2=6 \rightarrow $P(C1)=0/6=0$, $P(C2)=6/6=1$

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

- C1=1, C2=5 \rightarrow $P(C1)=1/6$, $P(C2)=5/6$

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

- C1=2, C2=4 \rightarrow $P(C1)=2/6$, $P(C2)=4/6$

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

- C1=3, C2=3 \rightarrow $P(C1)=3/6=0.5$, $P(C2)=3/6=0.5$

$$Gini(t) = 1 - 0.5^2 - 0.5^2 = 0.500$$

Algoritmo CART

- Criterio di Splitting: *Minimizzare l'Indice di Gini della suddivisione*
- Quando un nodo t è suddiviso in k partizioni (figli), la qualità della suddivisione è calcolata come:

$$Gini_{split} = \sum_{i=1}^k n_i/n * Gini(i)$$

dove

n_i = numero di record della partizione (figlio) i

n = numero di record del dataset al nodo t

n_i/n = peso dei vari $Gini(i)$

- Dato il dataset associato al nodo t , si sceglie l'attributo che fornisce il più piccolo $Gini_{split}(t)$ per partizionare il dataset
 - E' necessario enumerare tutti i possibili punti di splitting per ciascun attributo, ovverosia tutte le possibili partizioni

Algoritmi Induzione DT

- Algoritmo Greedy Decision Tree Learning
 - Scelta della migliore feature utilizzando come metrica il **Classification Error** sui dati di training —> problema NP-Hard —> servono euristiche
- Algoritmo C4.5
 - Scelta della migliore feature utilizzando come metrica l'**Information Gain** sui dati di training —> problema risolvibile tramite strategia *divide&conquer*
- Algoritmo CART
 - Scelta della migliore feature utilizzando come metrica il **Gini Index** sui dati di training —> problema risolvibile tramite strategia *divide&conquer*