

---

ANNO ACCADEMICO 2025/2026

---

# Apprendimento Automatico

---

## Teoria

Altair's Notes



**UNIVERSITÀ  
DI TORINO**



---

DIPARTIMENTO DI INFORMATICA

---



<b>CAPITOLO 1</b>	<b>INTRODUZIONE</b>	<b>PAGINA 5</b>
1.1	Che Cos'è l'Apprendimento Automatico? Terminologia — 5 • Tasks — 7 • Spazio di Ipotesi — 7 • No Free Lunch Theorem — 9	5
1.2	Selezione del Modello e Valutazione Errore Empirico e Overfitting — 9 • Metodi di Valutazione — 10	9



# Premessa

## Licenza

Questi appunti sono rilasciati sotto licenza Creative Commons Attribuzione 4.0 Internazionale (per maggiori informazioni consultare il link: <https://creativecommons.org/licenses/by/4.0/>).



## Formato utilizzato

Box di "Concetto sbagliato":

### Concetto sbagliato 0.1: Testo del concetto sbagliato

Testo contenente il concetto giusto.

Box di "Corollario":

### Corollario 0.0.1 Nome del corollario

Testo del corollario. Per corollario si intende una definizione minore, legata a un'altra definizione.

Box di "Definizione":

### Definizione 0.0.1: Nome delle definizioni

Testo della definizione.

Box di "Domanda":

### Domanda 0.1

Testo della domanda. Le domande sono spesso utilizzate per far riflettere sulle definizioni o sui concetti.

Box di "Esempio":

### Esempio 0.0.1 (Nome dell'esempio)

Testo dell'esempio. Gli esempi sono tratti dalle slides del corso.

**Box di "Note":**

**Note:-**

Testo della nota. Le note sono spesso utilizzate per chiarire concetti o per dare informazioni aggiuntive.

**Box di "Osservazioni":**

**Osservazioni 0.0.1**

Testo delle osservazioni. Le osservazioni sono spesso utilizzate per chiarire concetti o per dare informazioni aggiuntive. A differenza delle note le osservazioni sono più specifiche.



# 1

## Introduzione

### 1.1 Che Cos'è l'Apprendimento Automatico?

#### Definizione 1.1.1: Machine Learning

Un programma informatico apprende dall'esperienza  $E$  rispetto a una classe di task  $T$  e una performance  $P$ , se la sua performance nel task  $T$ , misurata da  $P$ , aumenta con l'esperienza  $E$ .

Sostanzialmente:

- L'esperienza viene data sotto forma di esempi "risolti" al computer.
- Un task (compito) da risolvere.
- Con un modo per valutare la risoluzione (performance).

#### 1.1.1 Terminologia

Attributes/Features				Label/Class   $y$
ID	Color	Root	Sound	Ripe
Training set	1	green	curly	muffled
	2	dark	curly	muffled
	3	green	straight	crisp
	4	dark	slightly curly	dull
Test set		1	green	curly
Sample/Instance   $x_i$				?
Attribute value/Feature value   $x_{i1}$				

Figure 1.1: Terminologia.

- Attributi (Features): le colonne.
- Etichetta (Classe): elemento che indica come risolvere un task.
- Istanza (Sample): una riga.
- Valori: le celle.
- Set di Training: insieme su cui si va a dedurre una regola per classificare.
- Test Set: insieme per vedere quanto si sarà accurati su insiemi futuri.

**Note:-**

Si usa un set diverso per il training e il test perché se si usasse lo stesso il modello farebbe risultati elevati essendo addestrato su quello.

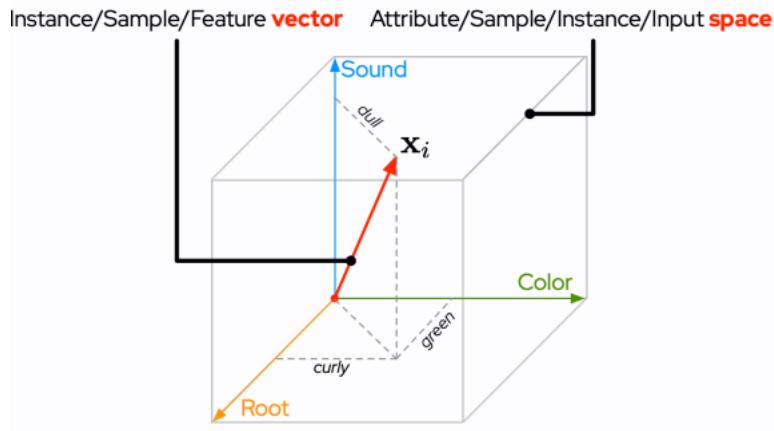


Figure 1.2: Terminologia 2.

Immaginando gli oggetti in un qualche campo euclideo:

- *Features vector*: ogni esempio corrisponde a un vettore.
- *Attribute space*: l'insieme di tutti gli esempi.

Symbol	Meaning
$\mathcal{X}$	instance space, in many cases $\mathcal{X} = \mathbb{R}^d$
$\mathcal{Y}$	label space, e.g., $\mathcal{Y} = \{0, 1\}$ for binary classification
$\mathbf{x}_i \in \mathcal{X}$	$i$ -th instance/sample, $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{id}]^\top$
$d$	dimensionality of the instance space
$y_i \in \mathcal{Y}$	label of the $i$ -th instance, e.g., $y_i = 1$ if the watermelon is ripe, $y_i = 0$ otherwise
$D$	Dataset, $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$
$n$	number of instances in the dataset
$h$	model/hypothesis, a function $h : \mathcal{X} \mapsto \mathcal{Y}$ that maps instances to labels
$\mathcal{L}$	Learning algorithm $\mathcal{L} : \mathcal{P}(\mathcal{X} \times \mathcal{Y}) \rightarrow \mathcal{H}$ , where $\mathcal{P}$ is the power set and $\mathcal{H}$ is the hypothesis space
$\mathbb{I}$	Indicator function, $\mathbb{I}(P)$ (sometime denoted as $\mathbb{I}[P]$ ) is 1 if $P$ is true, 0 otherwise

Figure 1.3: Tabella di riferimento.

**Definizione 1.1.2: Learning (Training)**

Il learning è un processo in cui si usano algoritmi di apprendimento automatico per costruire dei modelli.

- I dati utilizzati in questo processo sono detti training data.
- Ogni istanza è un training example.
- L'insieme di tutti i training example è il training set.

**Note:-**

Un modello addestrato corrisponde a una serie di regole sui dati, quindi si chiama anche *ipotesi* e le regole sono i *fatti* (grounded-truth).

**1.1.2 Tasks****Definizione 1.1.3: Tasks predittivi**

Un task predittivo è focalizzato sul prevedere una variabile sulla base degli esempi. Si parte da problemi vecchi per trovare la soluzione a nuovi problemi.

I tasks predittivi possono essere:

- *Binari e Multi-classe*: di categorizzazione.
- *Regressivi*: con un target numerico.
- *Clustering*: un target sconosciuto.

**Definizione 1.1.4: Tasks descrittivi**

Un task descrittivo si concentra sul fornire regolarità nel dataset.

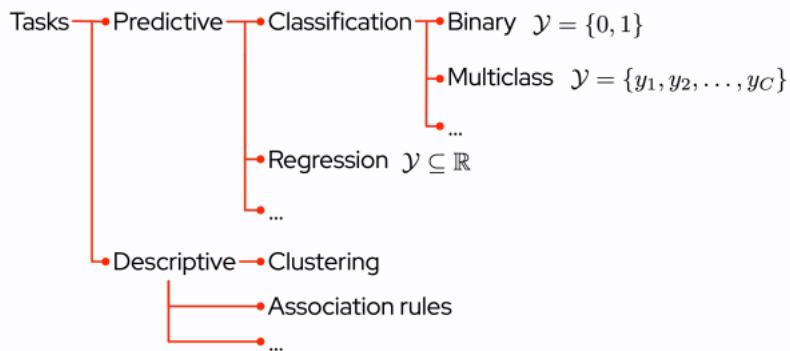


Figure 1.4: Vari tasks.

**Assunzione:**

- Si assume che i dati siano *indipendenti*.
- Si assume che i dati siano *identicamente distribuiti*.

**1.1.3 Spazio di Ipotesi**

Nei sistemi *assiomatici* il processo di derivare un teorema da assiomi è detto *deduzione*. Questo è un processo corretto se si assume che gli assiomi siano veri. L'*induzione* è il processo opposto ed è il principio su cui si basa tutto l'apprendimento automatico.

**Note:-**

ATTENZIONE: l'induzione come intesa in questo corso non è l'induzione matematica.

**Osservazioni 1.1.1** Deduzione vs. Induzione

- La deduzione è valida: assumere le premesse vere garantisce che le conclusioni siano vere.
- L'induzione non è valida come forma di ragionamento: non garantisce che la conclusione sia vera anche se tutte le osservazioni sono corrette.

**Definizione 1.1.5: Boolean Concept Learning**

L'obiettivo è quello di apprendere una funzione booleana  $h : \mathcal{X} \mapsto \{0, 1\}$

**Note:-**

Siamo nel campo del *symbolic concept learning*, studiato nel campo della *inductive logic programming*.

**Definizione 1.1.6: Spazio di Ipotesi**

Lo spazio di Ipotesi è l'insieme di tutte le possibili ipotesi che possono essere imparate da un algoritmo di apprendimento.

**Note:-**

Il Machine Learning è la ricerca attraverso lo spazio delle ipotesi per trovare l'insieme di tutte le ipotesi che sono consistenti con i training data e selezionare i migliori secondo un qualche criterio.

**Corollario 1.1.1 Spazio delle Versioni**

Sottoinsiem dello spazio delle ipotesi in cui tutte le ipotesi sono consistenti con il training data.

**Definizione 1.1.7: Bias Induttivo**

Il bias induttivo è un insieme di assunzioni che permette agli algoritmi di apprendimento di scegliere un'ipotesi dallo spazio delle versioni.

**Osservazioni 1.1.2**

- Ogni algoritmo ha un bias induttivo.
- L'unica altra possibilità sarebbe quella di scegliere a caso, ma è dumb as fuck come cosa.

**Domanda 1.1**

Possiamo adottare un'algoritmo con il miglior bias induttivo possibile?

**Rasoio di Occam:** tra ipotesi in competizione si sceglie la più semplice.

- Sarebbe troppo bello se fosse così.
- Ma cosa vuol dire "più semplice"?
  - Minor numero di parole/termini?
  - Più facile da interpretare?
  - Che faccia meno assunzioni sui dati?

### 1.1.4 No Free Lunch Theorem

#### Teorema 1.1.1 No Free Lunch Theorem

Nessun algoritmo è universalmente migliore di tutti gli altri quando la loro performance è la media su tutti i possibili problemi.

Denotiamo con:

- $P(h|X, \mathcal{L}_a)$ : la probabilità che l'algoritmo  $\mathcal{L}_a$  restituisca l'ipotesi  $h$  dato il training set  $X$ .
- $f$  la funzione che si vuole apprendere.
- L'errore out-of-sample, media dell'errore sugli esempi non nel training set:

$$E_{ote}(\mathcal{L}_a | X, f) = \sum_h \sum_{x \in \mathcal{X}-X} P(x) \mathbb{I}(h(x) \neq f(x)) P(h | X, \mathcal{L}_a)$$

Sommiamo tutte le possibili funzioni  $f_i$  otteniamo:

$$\begin{aligned} \sum_f E_{ote}(\mathcal{L}_a | X, f) &= \sum_f \sum_h \sum_{x \in \mathcal{X}-X} P(x) \mathbb{I}(h(x) \neq f(x)) P(h | X, \mathcal{L}_a) \\ &= \sum_{x \in \mathcal{X}-X} P(x) \sum_h P(h | X, \mathcal{L}_a) \sum_f \mathbb{I}(h(x) \neq f(x)) \\ &= \sum_{x \in \mathcal{X}-X} P(x) \sum_h P(h | X, \mathcal{L}_a) \frac{1}{2} 2^{|X|} \\ &= \frac{1}{2} 2^{|X|} \sum_{x \in \mathcal{X}-X} P(x) \sum_h P(h | X, \mathcal{L}_a) \\ &= \frac{1}{2} 2^{|X|} \sum_{x \in \mathcal{X}-X} P(x) \cdot 1 \end{aligned}$$

**Note:-**

Il No Free Lunch theorem si basa sull'assunzione che tutte le funzioni  $f$  sono ugualmente probabili. Ciò implica che la scelta dell'algoritmo di learning dovrebbe essere guidata dalle caratteristiche del problema.

## 1.2 Selezione del Modello e Valutazione

### 1.2.1 Errore Empirico e Overfitting

#### Domanda 1.2

Come misuriamo la bontà di un modello?

- **Error Rate:** si contano le predizioni sbagliate fatte,  $E = \frac{a}{m}$ .
- **Accuratezza:** accuracy =  $1 - E$ .
- L'errore valutato sul training set è chiamato *empirical error* o *training error*.
- L'errore valutato su nuovi esempi al di fuori del training set è chiamato *generalization error* o *test error*.

**Note:-**

Lo scopo dell'apprendimento è quello di minimizzare il generalization error. Però non si può fare direttamente, per cui si tende a minimizzare l'errore di training.

**Definizione 1.2.1: Overfitting**

Quando un modello si adatti ai dati troppo bene si rischia che esso sia troppo legato ai dati del training.

**Note:-**

Il fenomeno opposto è l'underfitting.

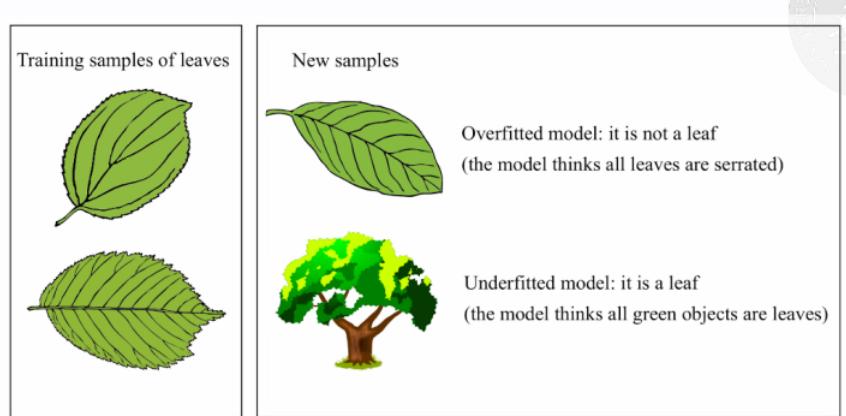


Figure 1.5: Overfitting e Underfitting.

### Osservazioni 1.2.1

- L'underfitting si risolve facilmente usando modelli più complessi.
- L'overfitting richiede un bilancio tra complessità del modello e capacità di generalizzazione.
- Molti algoritmi di apprendimento automatico hanno meccanismi per prevenire l'overfitting come *tecniche di regolarizzazione* o *early stopping*.

### 1.2.2 Metodi di Valutazione

Quello che si vuole fare è misurare l'errore di generalizzazione su nuovi esempi.

**Definizione 1.2.2: Hold-out Validation**

Dato l'insieme dei dati se ne tiene nascosta una parte all'algoritmo (test set).

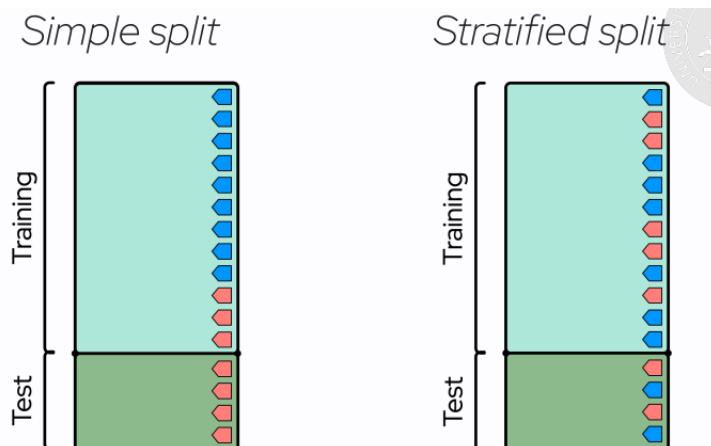


Figure 1.6: Validazione Hold-out.

**Gli split:**

- Split semplice: si prendono i primi  $n$  dati come test set. I problemi sorgono quando i dati non sono ben distribuiti.
- Split stratificato: si mescolano i dati prima di estrarli.

**Osservazioni 1.2.2**

- Ripetere la validazione Hold-out più volte produrrà risultati diversi poiché lo split è randomico.
- Per ottenere una stima migliore dell'errore di generalizzazione possiamo ripetere più volte e fare una media.
- La media di  $n$  variabili indipendenti è una nuova variabile avente media:

$$\frac{1}{n} \sum_{i=1}^n E_i = \frac{1}{n} \sum_{i=1}^n \mu = \mu$$

- E varianza:

$$\frac{1}{n^2} \sum_{i=1}^n \sigma^2 = \frac{\sigma^2}{n}$$

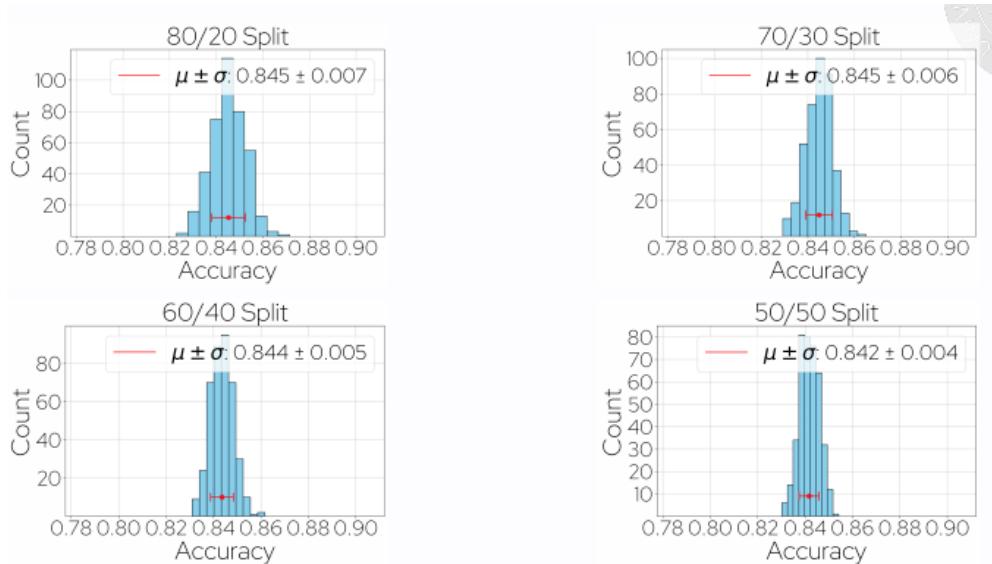


Figure 1.7: Validazione Hold-out con differenti split ratio.

**Note:-**

Se il dataset è abbastanza grande la validazione Hold-out è semplice ed efficace. Se il dataset è piccolo si ricorre alla *cross-validation*.

**Definizione 1.2.3: Cross-Validation**

Ripetizione dell'Hold-out validation  $k$  volte, la prima volta si usa come test set  $[0, \frac{1}{k}]$ , la seconda  $[\frac{1}{k}, \frac{2}{k}]$  fino all'ultima in cui si usa  $[\frac{k-1}{k}, 1]$ .

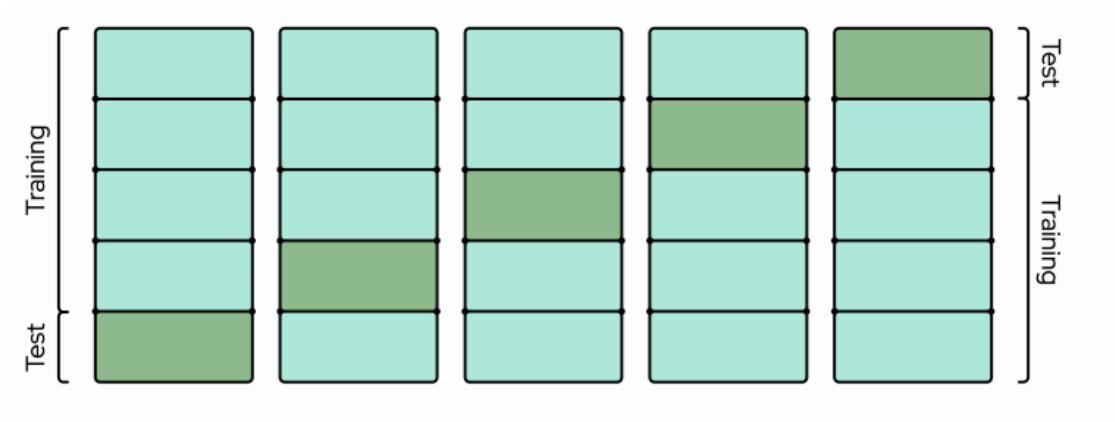


Figure 1.8: Cross-Validation.

**Domanda 1.3**

Perché usare la cross-validation invece di prendere la media degli hold-out ripetuti?

- Nella cross-validation ogni esempio è usato come test esattamente una volta, mentre ripetendo hold-out alcuni esempi possono essere usati più volte e alcuni mai usati.
- È più accurata, ma se il dataset è sufficientemente grande la differenza è trascurabile.



