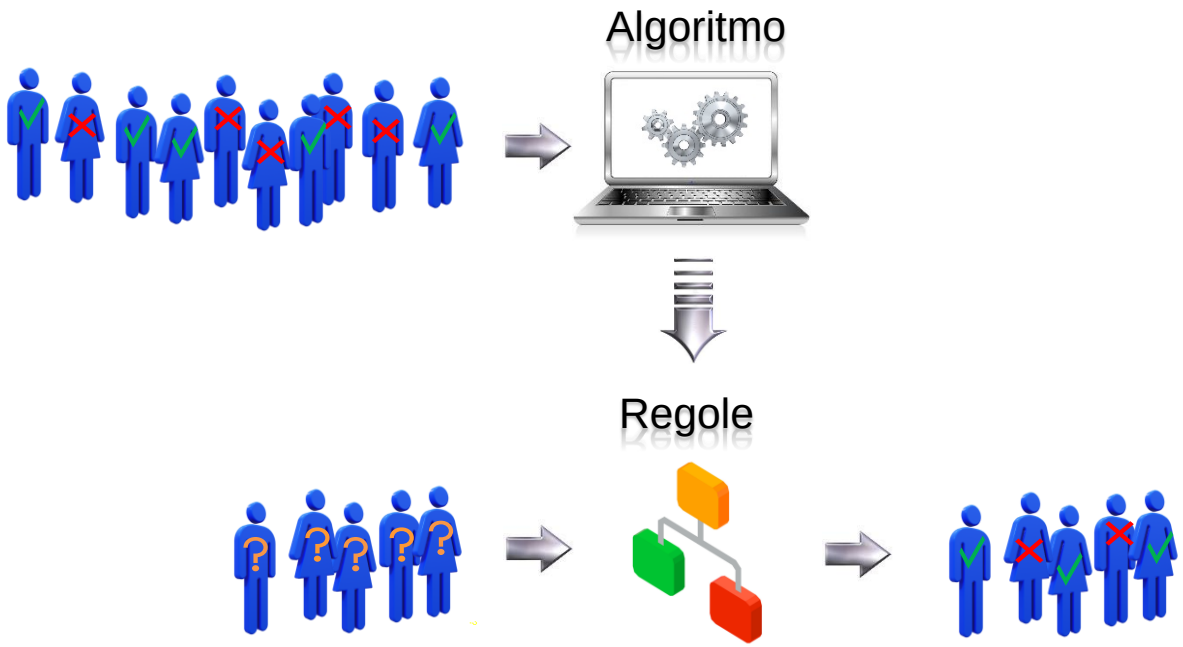
Business Analytics
Metodi statistici per il Decision Making
Modelli Predittivi

Questo libro è stato aggiornato per il software KNIME® Analytics Platform (Versione 4.5.2 e superiori), R (Versione 4.2.0 e superiori), Python (Versione 3.8.8 e superiori).

- **Modelli predittivi**
  - Cosa sono, a cosa servono
  - Il processo
  - La metodologia CRISP-DM
    - Modeling
    - Evaluation
    - Deployment

Cos'è un modello predittivo

La modellazione predittiva (***Predictive Modeling***) è un processo che utilizza i **dati storici** per poter prevedere, su nuove istanze, la probabilità del verificarsi di un certo evento (rischio oppure opportunità) allo scopo di migliorare l'efficacia dei processi decisionali.



Aree di utilizzo

Il *Predictive Modeling* viene impiegato in molti ambiti applicativi, ad esempio:

- **CRM** per indentificare i clienti a rischio di abbandono e attuare strategie per il mantenimento dei clienti più profittevoli
- **Marketing** per individuare i clienti ai quali indirizzare offerte mirate per attività di cross/up-selling
- **Credit Scoring** per valutare il rischio di credito dei clienti
- **Fraud Management** per intercettare e bloccare i comportamenti fraudolenti
- **Produzione** per rilevare le anomalie e migliorare la qualità
- **Risorse Umane** per prevenire il turnover dei dipendenti
- **Assistenza sanitaria** per la prevenzione dei rischi sanitari

- **Modelli predittivi**
  - Cosa sono, a cosa servono
  - Il processo
  - La metodologia CRISP-DM
    - Modeling
    - Evaluation
    - Deployment

Il Processo

Passi e competenze per la realizzazione di un modello predittivo

1. Definire gli **obiettivi di business**, che devono essere realistici, misurabili e tempestivi (p.e. diminuire del 20% in 3 mesi il tasso di abbandono da parte dei clienti di un operatore di telefonia mobile)
   - *Esperto di business, data scientist*
2. Definire gli **scopi del modello** (p.e. creare un modello che possa predire i clienti che disdiranno a breve il contratto telefonico con un'accuratezza maggiore del 80%)
   - *Esperto di business, data scientist*
3. **Selezionare, pulire e preparare i dati** attraverso la loro analisi e trasformazione (p.e. i dati del cliente, il numero e la durata delle chiamate in entrata/uscita, il tempo di attività/inattività)
   - *Data engineer, data scientist*
4. Scelta della tecnica e **costruzione del modello**
   - *Data scientist*
5. **Valutazione del modello** in termini di performance e redditività ed eventuale **Storytelling**
   - *Esperto di business, data scientist*
6. **Messa in produzione**
   - *Data engineer*

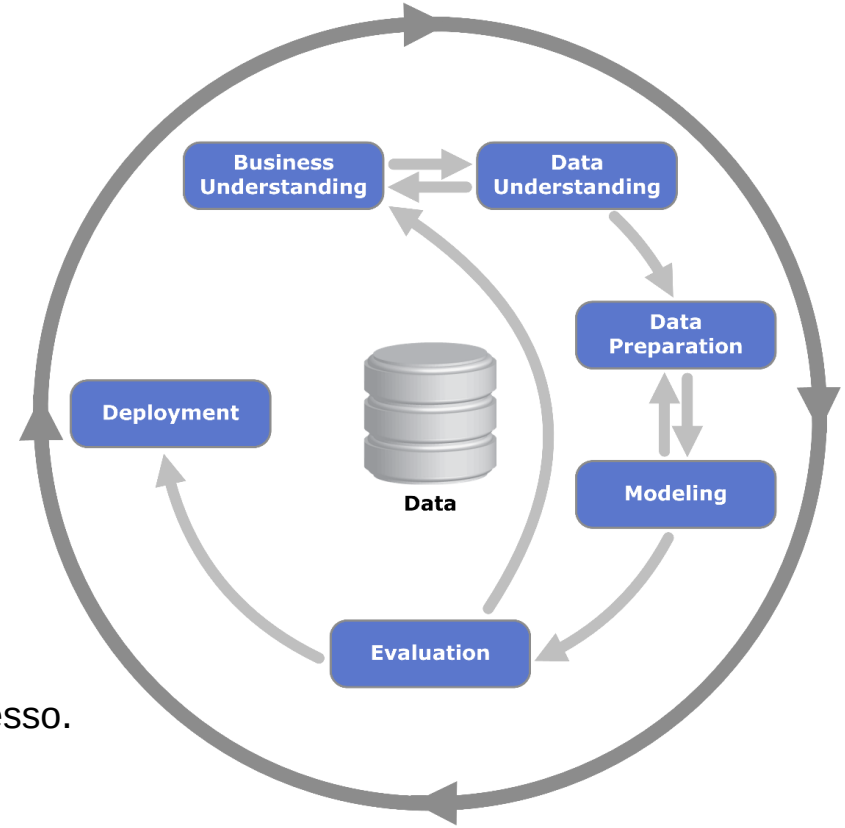
La Metodologia

La metodologia CRISP-DM

Il processo descritto precedentemente è stato standardizzato in una ben nota e ampliamente accettata metodologia, la ***Cross-Industry Standard Process for Data Mining***

Questa metodologia, dove confluiscono obiettivi di business e di analisi, prevede almeno 6 fasi non vincolate tra loro da rigidità temporali che si susseguono.

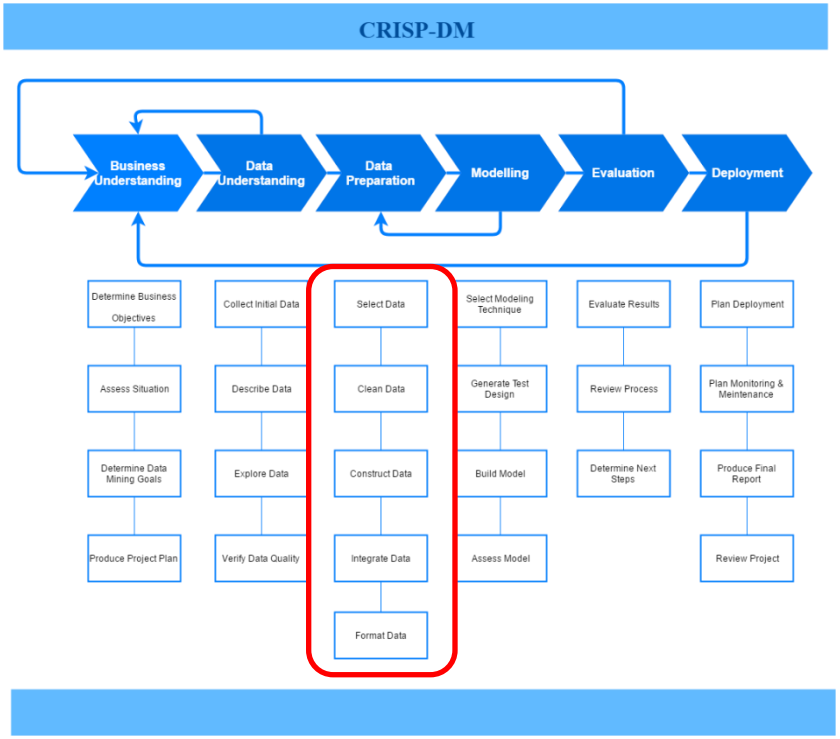
È sempre possibile tornare alla fase precedente o **rivedere le idee alla base dell'analisi**, alla luce dei risultati emersi in un punto qualunque del processo.



La metodologia CRISP-DM – Data Preparation

La fase di ***Data Preparation*** è fondamentale per la creazione di un modello performante.
Le sue attività sono le più delicate che assorbono la maggior parte del tempo dell'analista.

- Selezione delle righe e delle colonne
- Consolidamento e pulizia
- Trasformazione



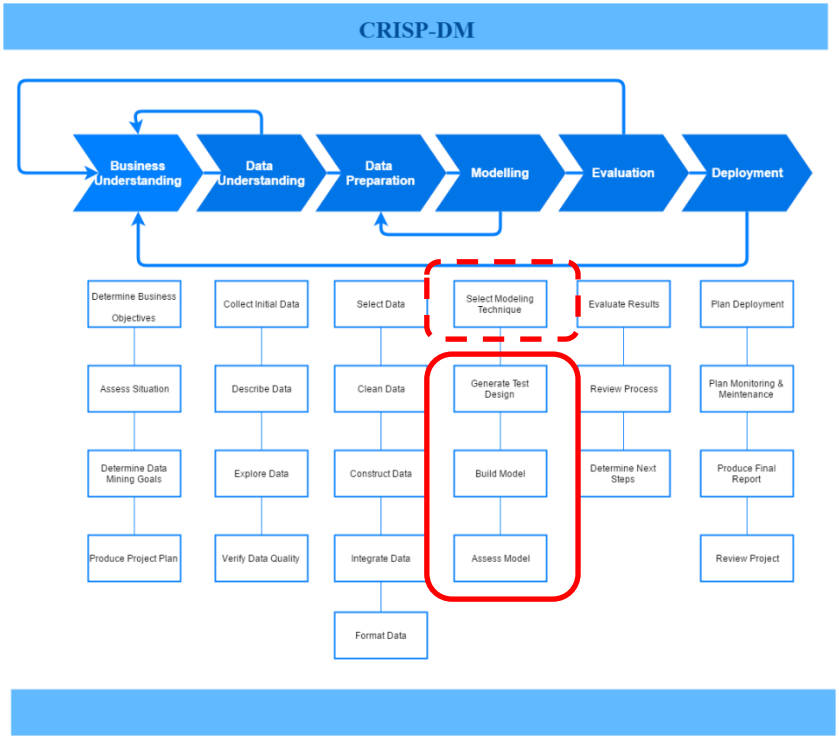
[1] Questa fase viene trattata in modo dettagliato nel corso **IA – Machine Learning – Data Processing**.

La metodologia CRISP-DM – Il Modeling

Queste note trattano ampiamente la fase del ***Modeling***, dove i modelli vengono costruiti, testati e migliorati per ottimizzare le performance.

Una volta che i dati sono stati preparati, vengono sottoposti nel ***Modeling*** alle seguenti attività:

- Selezione della tecnica da utilizzare
- Preparazione dell'ambiente per l'addestramento e la validazione
- Costruzione del modello
- Validazione



Selezione della tecnica da utilizzare

Le varie tecniche (o *algoritmi*) utilizzate dai modelli predittivi verranno discusse in modo dettagliato più avanti.

La scelta della tecnica da utilizzare dipenderà dall'esperienza dell'analista, dai dati a disposizione, dal loro confronto e, non ultima, la capacità di interpretare i risultati per capire quali variabili, e come, contribuiscono maggiormente all'accuratezza delle predizioni.

I modelli predittivi, a seconda del problema da analizzare, si dividono in due aree:

- **Regressione** dove quello che si vuole predire è una quantità (*p.e. la temperatura massima nei giorni a seguire per le previsioni meteo*)

- **Classificazione**[1] dove quello che si vuole predire è una classe o categoria (*p.e. soleggiato, piovoso o nevoso*)

[1] In queste note si discuterà **solo di modelli di classificazione binari**.

Il Software

Per gli esempi illustrati in queste note si sono utilizzati i seguenti software open-source[1]:

KNIME Analytics Platform

R Project

Python (cenni)

[1] Vedi corsi dell'autore **Introduzione al software KNIME® Analytics Platform** e **Introduzione al software R**

Preparazione dell'ambiente per il training e il testing

Per il *modeling* i **dati storici** devono essere strutturati in modo da avere una colonna chiamata *variabile dipendente* (*o **"target"***), che rappresenta la classe ovvero l'**evento da predire**, e un insieme di colonne chiamate *variabili indipendenti (o **"input")*** che servono per la costruzione del modello.

Nel caso della concessione di prestiti, ad esempio, i dati storici potrebbero avere come variabile target l'esito della cessione (solvenza oppure default) e, come variabili di input, i dati socio-demografici ed economico-finanziari dei clienti.

	Codice cliente	**$Input_1$**	**$Input_2$**	**...**	**$Input_n$**	**Target**
	ID	**Reddito**	**AA Ult. Imp.**	**...**	**Proprietà Casa**	**Default (1=Sì)**
Dati storici	0123451	86,4	11		1	**0**
	0002345	28,3	5		0	**1**
	...	...	...	...	...	**...**
	0239427	36,0	16		0	**0**
	0093486	53,4	2		1	**0**
	0002372	49,4	3		0	**1**

Preparazione dell'ambiente per il training e il testing

Una volta **addestrato**, utilizzando i dati storici, il modello può essere applicato a un insieme di nuovi dati ottenendo come risultato la **probabilità** che si verifichi l'evento d'interesse e, quindi, l'informazione ricercata.

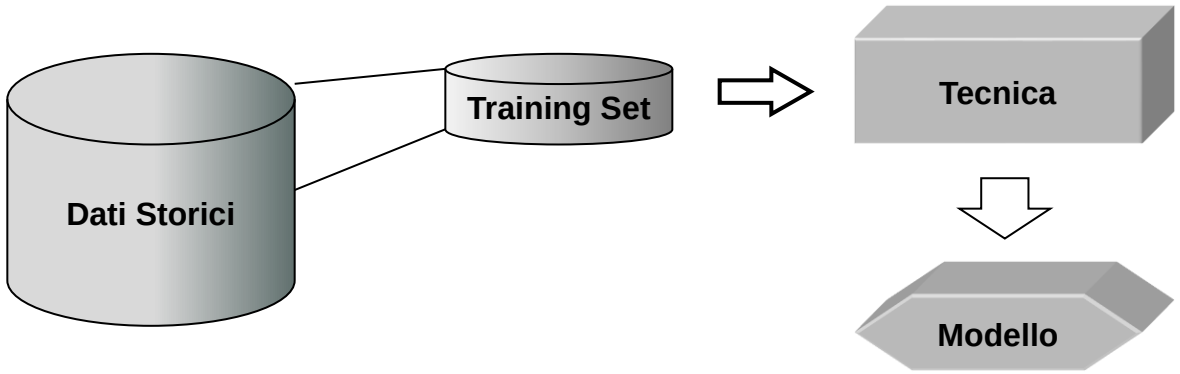
Continuando con l'esempio precedente, un modello che ottenga da **nuove istanze** dei dati i seguenti risultati, applicando una soglia di probabilità del 50%, porterebbe a queste decisioni riguardo la cessione del credito:

	Codice cliente	**$Input_1$**	**$Input_2$**	**...**	**$Input_n$**	**Probabilità**	**Decisione**
	ID	**Reddito**	**AA Ult. Imp.**	**...**	**Proprietà Casa**	**Default=1**	**Cessione**
Nuovi dati	0368348	44,2	8		0	**0,537**	**NO**
	0374010	92,5	12		1	**0,325**	**SI**
	...	...	...	...	...	...	...
	0384613	36,0	6		0	**0,894**	**NO**
	0345762	86,4	23		0	**0,423**	**SI**
	0364971	49,4	15		1	**0,493**	**SI**

Preparazione dell'ambiente per il training e il testing

Addestramento (Training)

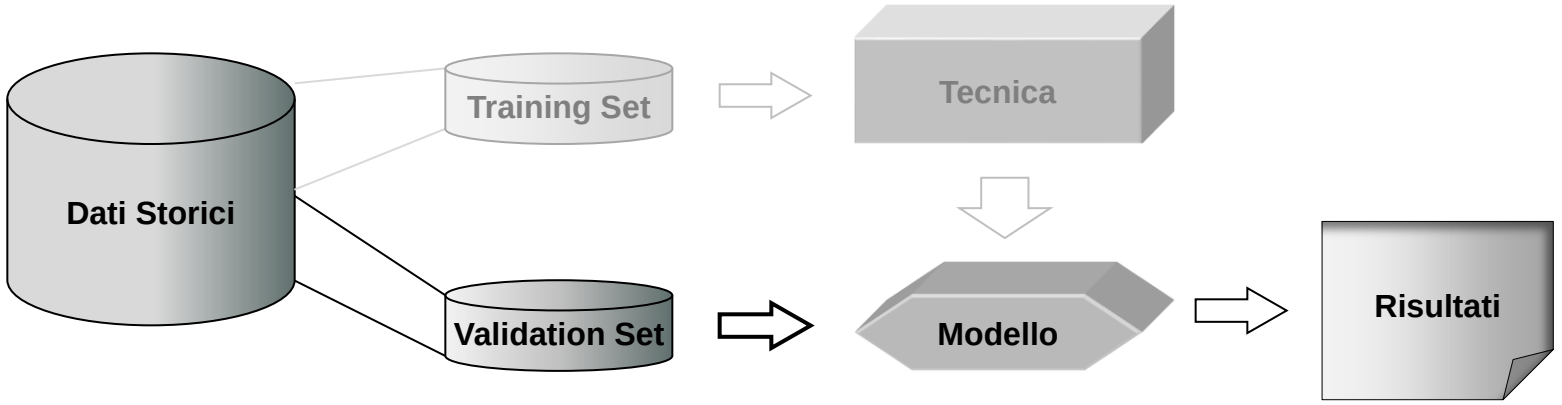
È la fase di **definizione dei parametri** del modello che viene creato in base ad esempi già noti forniti dai dati storici. Di solito viene utilizzato un **sotto-insieme rappresentativo** dei dati storici (***training set***) che ha il vantaggio di rendere più efficiente il processo di costruzione del modello.



Preparazione dell'ambiente per il training e il testing

Validazione (Testing)

E' la fase di controllo in cui si valuta la bontà del modello, ovvero la **capacità di generalizzare**[1]. Per far questo il modello viene testato su un altro insieme di dati, diversi da quelli usati per l'addestramento (***validation set***)[2]. Il modello ottimale è quello che minimizza l'errore tra i valori reali della variabile target e quelli predetti.



[1] E' la capacità di creare predizioni accurate in presenza di dati non appartenenti all'insieme originale utilizzato per l'addestramento.
[2] Per verificare quale può essere la sua affidabilità in circostanze operative, il modello ottenuto dovrebbe essere testato anche su un terzo insieme di dati non usati per la costruzione e per la validazione, detto "***Test Set***" o "***Holdout Set***".

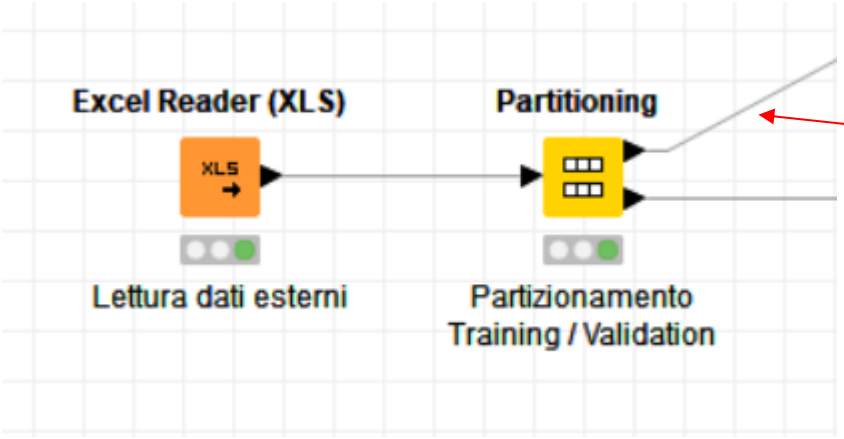
Preparazione dell'ambiente per il training e il testing

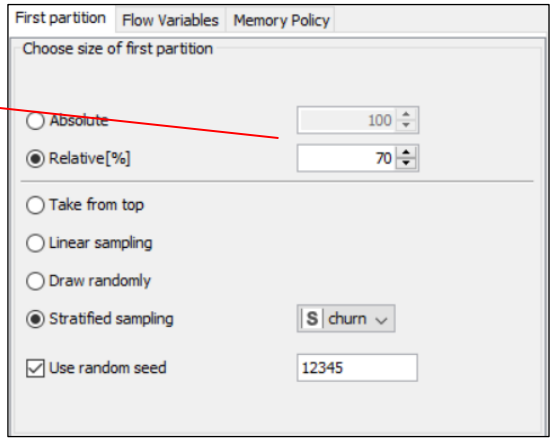
In KNIME per diversificare il training set dal validation set si utilizza il nodo **Partitioning**.

Il training set si ottiene indicando il numero delle righe o la loro percentuale, il validation set con le righe rimanenti.

La selezione avviene di solito per campionamento, casuale o stratificato. Con il **campionamento stratificato** la distribuzione dei valori della validati on colonna target è (approssimativamente) mantenuta nelle due tabelle di uscita.

Se si vogliono ottenere sempre le stesse tabelle in uscita, si può usare un **seme casuale**.





Validazione – La soglia di decisione

I risultati forniti dal modello, in **base ai dati del validation set**, forniscono in uscita delle probabilità (dette anche "***score***") che hanno valori **compresi tra 0 e 1**.

Se lo score supera un prefissato valore di **soglia** (o **"*threshold*"**), che rappresenta il valore di separazione tra "positività" e "negatività" del test, si assegna la riga alla classe degli **eventi positivi**, altrimenti alla classe degli **eventi negativi**.

Età	Scolarità	Stato Civile	Figli	Anzianità mesi	N. visite mese	Reclami	Recenza	Frequenza	Monetarietà	Campagne	Risposta	SCORE Pr(Risposta=1)	Decisione score> 0,25	score>0,5
54	Media	Sposato/Convivente	0	68	4	0	26	21	776	Adesioni_0	0	0,026	0	0
32	Media	Sposato/Convivente	0	78	8	0	38	5	46	Adesioni_0	0	0,210	0	0
34	Alta	Single	1	77	6	0	86	8	133	Adesioni_0	0	0,077	0	0
30	Alta	Divorziato/Vedovo	0	59	5	0	34	8	30	Adesioni_0	0	0,080	0	0
67	Alta	Divorziato/Vedovo	1	78	6	0	8	17	302	Adesioni_0	1	0,637	1	1
68	Alta	Single	1	68	7	0	55	6	65	Adesioni_0	0	0,059	0	0
34	Alta	Sposato/Convivente	1	73	8	0	88	7	53	Adesioni_0	0	0,065	0	0
60	Alta	Single	0	65	2	0	80	9	186	Adesioni_0	0	0,040	0	0
65	Alta	Sposato/Convivente	1	77	8	0	39	28	1319	Adesioni_0	0	0,132	0	0
55	Alta	Sposato/Convivente	0	80	3	0	2	29	1693	Adesioni_0	0	0,663	1	1
53	Media	Divorziato/Vedovo	0	72	3	0	30	21	1093	Adesioni_0	1	0,283	1	0
37	Alta	Single	0	69	1	0	12	21	1438	Adesioni>1	1	0,968	1	1

Valori predetti

Valori reali

Ogni valore di soglia porta a una decisione diversa, la scelta dipende dal tipo di errore che si vuole accettare.

Validazione – Matrice di confusione

La **matrice di confusione** è uno strumento per misurare la qualità di un sistema di classificazione. Nella matrice di confusione le **righe rappresentano le classi vere**, le **colonne le classi assegnate** dal classificatore (decisioni).

Sulla diagonale ci sono le osservazioni classificate correttamente, che vengono identificate come **veri negativi (VN)** e **veri positivi (VP)**. Le altre osservazioni sono errori, identificati come:

Falsi negativi (FN) il valore della colonna target è “positivo”, ma viene predetto come negativo.
Falsi positivi (FP) il valore della colonna target è “negativo”, ma viene predetto come positivo.

		Valori Predetti 0	1	
Valori Reali	0	Veri Negativi	Falsi Positivi	Totale Negativi
	1	Falsi Negativi	Veri Positivi	Totale Positivi
		Totale Predetti Negativi	Totale Predetti Positivi	Totale

Validazione – Matrice di confusione

Il classificatore dà buoni risultati quando è alto sia il **tasso dei veri positivi (SENSIBILITA')**, sia il **tasso dei veri negativi (SPECIFICITA').**

La valutazione delle decisioni corrette e degli errori viene fatta utilizzando i seguenti indicatori:

Accuratezza/Accuracy	= (VN + VP) / Totale	rapporto tra veri positivi più veri negativi e totale generale
Prevalenza/Prevalence	= (FN + VP) / Totale	rapporto tra totale positivi e totale generale
Specificità/Specificity	= VN / (VN + FP)	rapporto tra veri negativi e totale negativi
Sensibilità/Sensitivity[1]	= VP / (FN + VP)	rapporto tra veri positivi e totale positivi
Tasso FN/FN Rate	= FN / (FN + VP)	rapporto tra falsi negativi e totale positivi
Tasso FP/TP Rate	= FP / (VN + FP)	rapporto tra falsi positivi e totale negativi
Precisione/Precision[2]	= VP / (VP+FP)	rapporto tra veri positivi e totale predetti positivi
F-score/F1 Score	= 2(SE*PR)/(SE+PR)	media armonica di sensibilità e precisione
Kappa di Cohen		misura il grado di accuratezza e affidabilità
Lift	= Precisione / Prevalenza	miglioramento rispetto a una scelta casuale

Valori Reali \ *Valori Predetti*	0	1	
0	Specificità	% Falsi Positivi	Totale Negativi
1	% Falsi Negativi	Sensibilità	Totale Positivi
	Totale Predetti Negativi	Totale Predetti Positivi	Totale

[1] Oppure **Recall**: "*per tutti i casi che sono effettivamente positivi, quale percentuale è classificata correttamente?".*
[2] Oppure **Valore Predittivo Positivo (VPP)**/Positive Predicted Value (PPV): "*per tutti i casi classificati come positivi, quale percentuale è corretta?*".

Validazione – Matrice di confusione

In questo esempio c'è il calcolo degli indici di una matrice di confusione ottenuta dai risultati di un classificatore. Il *threshold* prescelto ha selezionato 40 veri negativi, 86 veri positivi, 4 falsi positivi e 20 falsi negativi:

Accuratezza	= (VN + VP) / Totale	(40 + 86)/150 = 84%
Prevalenza	= (FN + VP) / Totale	(20 + 86)/150 = 71%
Specificità	= VN / (VN + FP)	40/ 44 = 91%
Sensibilità	= VP / (FN + VP)	86/106 = 81%
Tasso FN	= FN / (FN + VP)	20/106 = 19%
Tasso FP	= FP / (VN + FP)	4/ 44 = 9%

		Valori Predetti		
		0	**1**	
Valori Reali	**0**	**40 (91%)**	**4 (9%)**	44
	1	**20 (19%)**	**86 (81%)**	106
		60	90	150

Precisione	= VP / (VP + FP)	86 / 90 = 95,5%
F-score	= 2(SE*PR)/(SE+PR)	2(0,81*0,955)/(0,81+0,955) = 87,8%
Cohen's kappa[1]		= 0,65
Lift	= Precisione/Prevalenza	91 / 71 = 1,35%

[1] https://en.wikipedia.org/wiki/Cohen%27s_kappa

Validazione – Scelta del threshold

Ogni valore di ***threshold*** determina quindi **una diversa matrice di confusione**.

valori alti: alta specificità; bassa sensibilità
valori bassi: bassa specificità; alta sensibilità

Dall'esempio di pag. 10, con i valori di *threshold* di 0,25 e 0,50 avremo due matrici di confusione:

Soglia >= 0.25		Valori Predetti 0	1	
Valori Reali	0	276	72	348
	1	10	52	62
		286	124	410

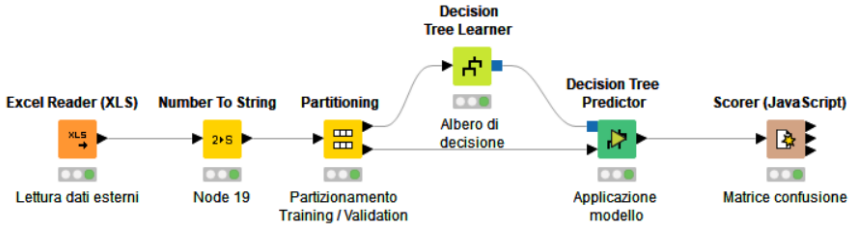
Accuratezza = (Veri Pos. + Veri Neg.) / Totale	80,0%
Sensibilità = Veri Pos. / Totale Reali Pos.	83,9%
Specificità = Veri Neg. / Totale Reali Neg.	79,3%

Soglia >= 0,50		Valori Predetti 0	1	
Valori Reali	0	335	13	348
	1	28	34	62
		363	47	410

Accuratezza = (Veri Pos. + Veri Neg.) / Totale	90,0%
Sensibilità = Veri Pos. / Totale Reali Pos.	54,8%
Specificità = Veri Neg. / Totale Reali Neg.	96,3%

Validazione – Matrice di confusione

La matrice di confusione in KNIME si ottiene con il nodo **Scorer (Java Script)**



che fornisce le seguenti statistiche (con un *threshold* di default di 0,5):

Confusion Matrix

Rows Number : 410	0 (Predicted)	1 (Predicted)	
0 (Actual)	335	13	96.26%
1 (Actual)	28	34	54.84%
	92.29%	72.34%	

Class Statistics

Class	True Positives	False Positives	True Negatives	False Negatives	Precision	Sensitivity	Specificity	F-measure
0	335	28	34	13	92.29%	96.26%	54.84%	94.23%
1	34	13	335	28	72.34%	54.84%	96.26%	62.39%

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
90.00%	10.00%	0.567	369	41

Validazione – Curva ROC

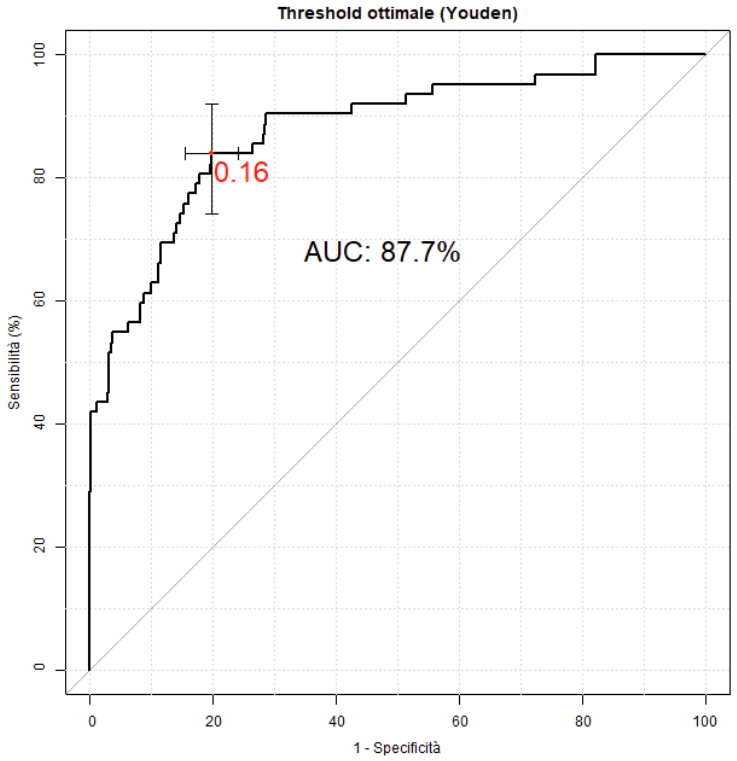
La scelta del *threshold* dipende dagli obiettivi dell'analisi e, quindi, quale errore conviene ridurre. Per valutare graficamente la bontà di un classificatore si utilizza la **curva ROC** (***Receiver Operating Characteristic***), dove l'asse verticale rappresenta la Sensibilità mentre l'asse orizzontale 1-Specificità.

La curva ROC rappresenta quindi il tasso dei Veri Positivi (Sensibilità) in funzione del tasso dei Falsi Positivi al variare del *threshold*.

La linea retta rappresenta il caso del classificatore casuale dove l'area sottesa (*Area Under Curve*, AUC) è pari a 0,5.

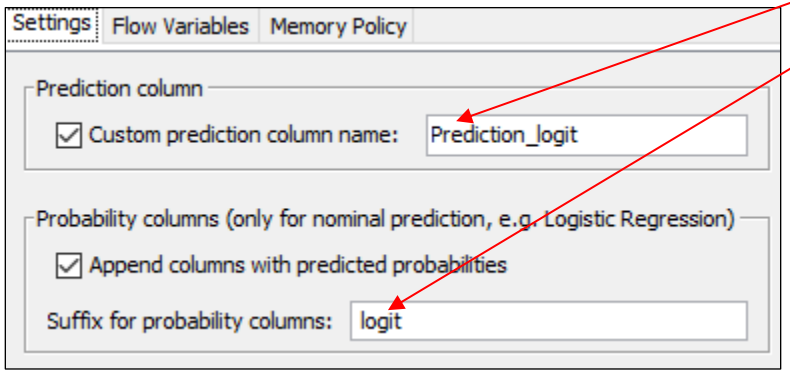
Più l'AUC della curva ROC si avvicina a 1, più è alta la bontà del classificatore.

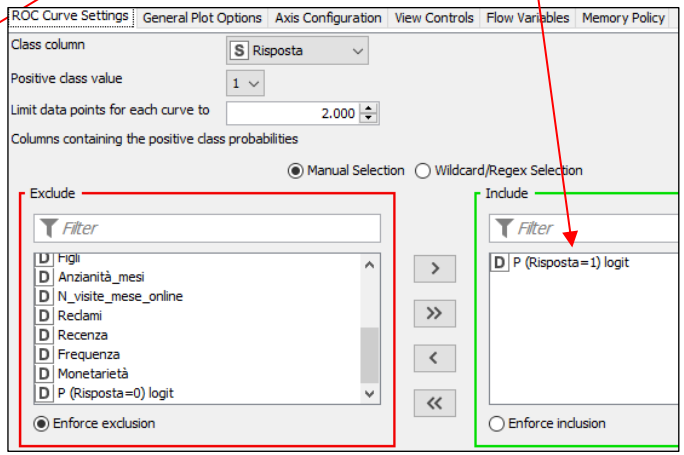
Il punto di flesso della curva è quel valore di *threshold* per il quale si minimizza la differenza assoluta tra Sensibilità e Specificità.



Validazione – Curva ROC

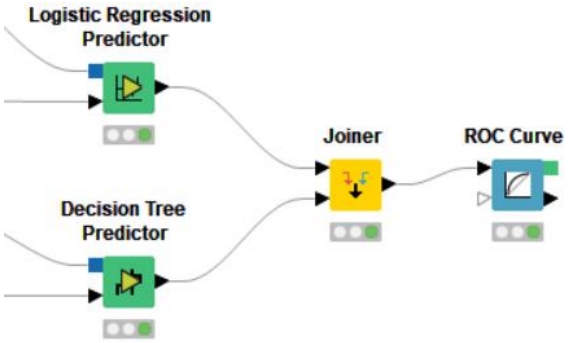
La curva ROC si ottiene in KNIME con il nodo **ROC Curve** avendo cura di aggiungere al nodo Predictor la colonna delle probabilità, indicandone il suffisso:

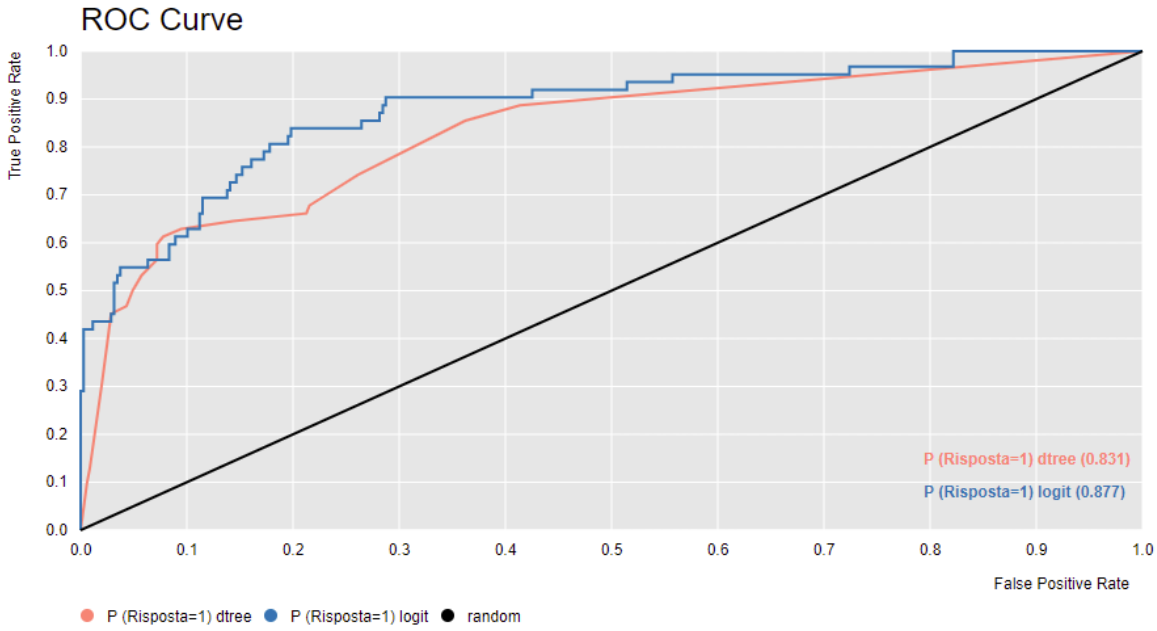




Validazione – Curva ROC

Indicando per ogni nodo Predictor un apposito suffisso è possibile confrontare insieme diversi modelli:





Validazione – Scelta del threshold

Scegliendo dall'esempio precedente il *threshold* che **minimizza la differenza tra Sensibilità e Specificità**, si ottiene la relativa matrice di confusione. Osservando i risultati si nota che, in effetti, si ottengono valori bilanciati di Specificità e Sensibilità.

Soglia >= 0,16		Valori Predetti		
		0	1	
Valori Reali	0	280	68	348
	1	12	50	62
		292	118	410

Accuratezza = (Veri Pos. + Veri Neg.) / Totale	80,5%
Sensibilità = Veri Pos. / Totale Reali Pos.	80,6%
Specificità = Veri Neg. / Totale Reali Neg.	80,5%

Soglia >= 0,50		Valori Predetti		
		0	1	
Valori Reali	0	335	13	348
	1	28	34	62
		363	47	410

Accuratezza = (Veri Pos. + Veri Neg.) / Totale	90,0%
Sensibilità = Veri Pos. / Totale Reali Pos.	54,8%
Specificità = Veri Neg. / Totale Reali Neg.	96,3%

La matrice ottenuta con un threshold di 0,5 evidenzia invece un aumento della Specificità a scapito della Sensibilità.

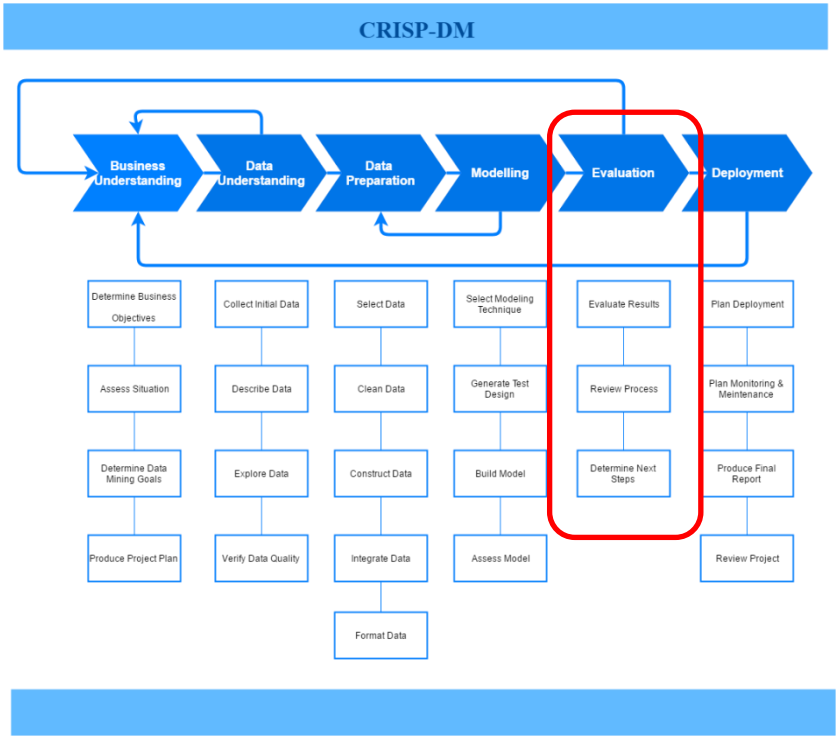
Come si è detto, la scelta del *threshold* dipende dagli obiettivi dell'analisi, dall'errore che si vuole ridurre[1].

[1] Nel caso del credit scoring, ad esempio, la migliore strategia consiste nel tener basso il numero dei falsi negativi, dove l'evento positivo è l'insolvibilità.

La Metodologia

L'***Evaluation*** *è la fase di valutazione e revisione del processo con queste attività*:

- Valutazione dei risultati
- Revisione del processo



Matrice di profitto – 1/4

Come spiegato prima, se si cambia il *threshold* di classificazione (che è impostato per default a 0,5), si cambiano le assegnazioni alla classe positiva e negativa e i conseguenti errori di classificazione.

Per valutare questi errori in termini di **profitti attesi**, si utilizza una **matrice di profitto** per assegnare dei profitti (o dei costi) ai risultati ottenuti.

		Previsti negativi	**Previsti** positivi
Osservati	negativi	**Profitto VN**	**Profitto FP**
	positivi	**Profitto FN**	**Profitto VP**

e calcolare il profitto medio per soggetto $$\boldsymbol{Profitto\ medio} = \frac{P_{VN} * VN + P_{FN} * FN + P_{FP} * FP + P_{VP} * VP}{Totale}$$

Nel caso della concessione prestiti, ad esempio, si vuole predire chi è **insolvente** (**evento positivo**) da chi non lo è (evento negativo) per cui ai **veri negativi** (solventi predetti come tali) verrà assegnato un **profitto**, mentre ai **falsi negativi** (insolventi predetti come solventi) verrà assegnato un **profitto negativo**, quindi un **costo**. Ai falsi positivi (solventi predetti come insolventi) e ai veri positivi (insolventi predetti come tali) verrà assegnato un profitto pari a zero per cui la **matrice di profitto** in questo caso sarà:

		Previsti negativi	**Previsti** positivi
Osservati	negativi	**Profitto**	**0**
	positivi	**-Profitto (Costo)**	**0**

Matrice di profitto – 3/4

Per calcolare il valore di *threshold* (teorico) che **massimizzi i profitti** si può usare la seguente formula:

$$Threshold >= \frac{1}{1 + \frac{(Profitto_{VP} - Profitto_{FN})}{(Profitto_{VN} - Profitto_{FP})}}$$

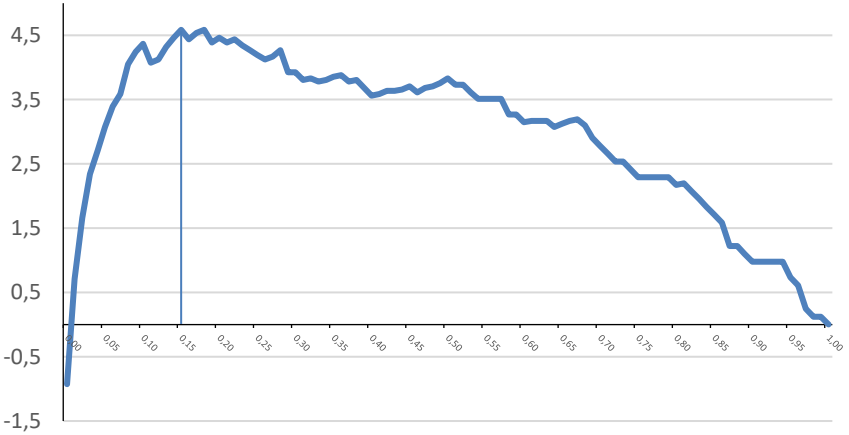
Nel caso dell'esempio di pag. 17, che riguarda una campagna di marketing, avendo un ricavo medio di 60€ contro un costo di promozione di 10€, si avrà una matrice di profitto come quella riportata a fianco.

		Previsti: negativi	Previsti: positivi
Osservati	negativi	Veri Negativi: 0	Falsi Positivi: -10
	positivi	Falsi Negativi: 0	Veri Positivi: 50

Applicando la formula si avrà un

$$Threshold \geq \frac{1}{1+\frac{(50-0)}{(0-(-10))}} \geq 0{,}167$$

come si può anche evincere dal grafico ottenuto calcolando il profitto medio per soggetto in funzione del threshold:



[1] Per approfondimento si veda il blog post: https://www.knime.com/blog/from-modeling-to-scoring

Matrice di profitto – 4/4

Continuando sempre con l'esempio di pag. 17 si può calcolare il profitto medio per alcuni valori di *threshold* (sotto la tabella, a margine, il profitto totale che si otterrebbe con i 410 soggetti del validation set):

	Nessuna previsione	Previsione per threshold			
		Max Accuratezza	Standard	Min SE-SP	Max Profit
		0,67	0,5	0,16	0,17
Falsi Positivi	348	4	13	68	64
Veri Positivi	62	27	34	50	50
Accuratezza		90,5%	90,0%	80,5%	81,5%
Profitto per richiedente	**-0,9268**	3,195	3,829	4,439	**4,537**
Profitto totale	**-380**	1.310	1.570	1.820	**1.860**

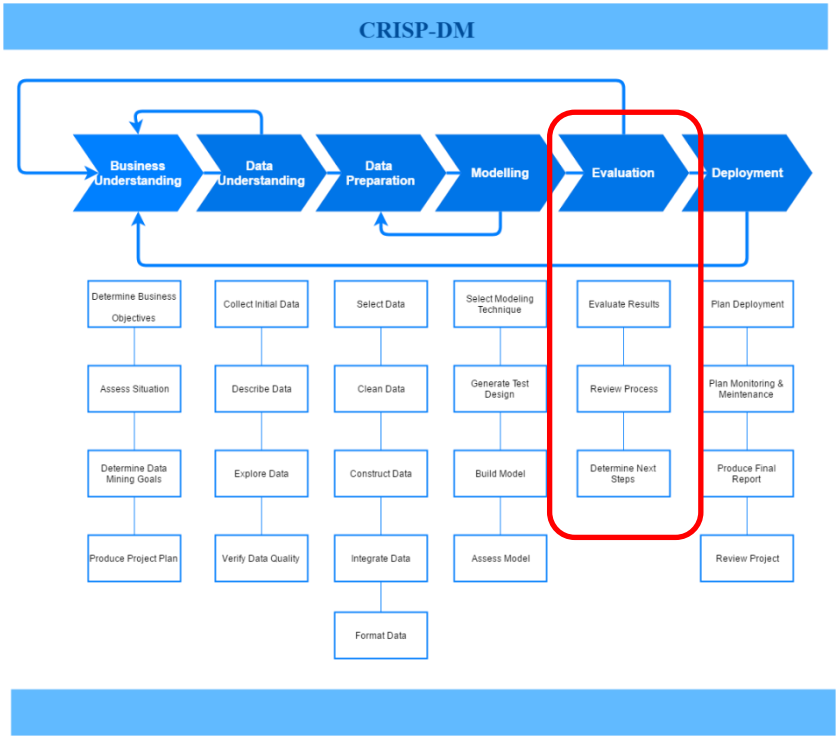
Con **nessun modello**, si avrebbe un profitto totale (negativo!) di -380€; con il *threshold* che fornisce la **massima accuratezza** di 1.310€, con quello **standard** (0,5) di 1.570€; con quello che **equilibra la sensibilità e la specificità** di 1.820€ e, con quello che fornisce il **massimo profitto**, di **1.860€**.

La Metodologia

L'***Evaluation*** *è la fase di valutazione e revisione del processo con queste attività*:

- Valutazione dei risultati

- Revisione del processo



Valutazione

Revisione del processo

Quest'attività serve a individuare i problemi che si potrebbero aver trascurato e che si potrebbero ancora correggere prima di portare rilasciare il modello in produzione.

- Qualità dei dati

- Sovra-adattamento

- Dati sbilanciati

Revisione del processo

Il problema del sovra-adattamento (overfitting) – 1/3

Se un modello ottiene buoni risultati sul training set ma non sul validation set, ed è quindi incapace di generalizzare, può significare che sia sovra-adattato (*overfitted*), cioè troppo dipendente dai dati utilizzati per l'addestramento (p.e. un modello che avesse un'accuratezza del 99% sul training set, ma solo del 55% sul validation set).

Questo problema si verifica quando il modello è **sovra-parametrizzato**, oppure quando l'**addestramento** è stato eseguito **troppo a lungo**, nel qual caso le prestazioni sui dati di addestramento aumentano sempre di più mentre peggiorano su quelli di test.

Generalmente l'*overfitting* si risolve **riducendo le variabili di input** o **semplificando il modello**.

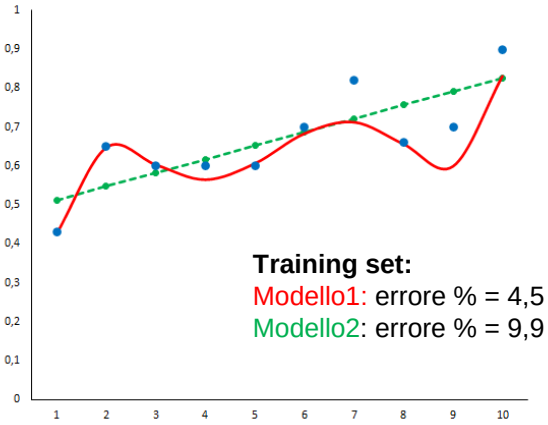
Revisione del processo

Il problema del sovra-adattamento (overfitting) – 2/3

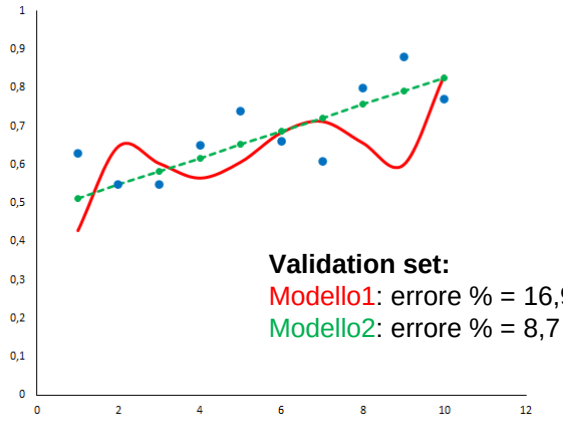
Esempio: addestramento con un modello polinomiale di grado 5 (*modello1*) e con un modello lineare (*modello2*):

$$modello1 = -0{,}6033+1{,}7085*x-0{,}8425*x^2+0{,}1846*x^3-0{,}0183*x^4+0{,}00067*x^5$$

$$modello2 = 0{,}478+0{,}0349*x$$



Training set:
Modello1: errore % = 4,5
Modello2: errore % = 9,9



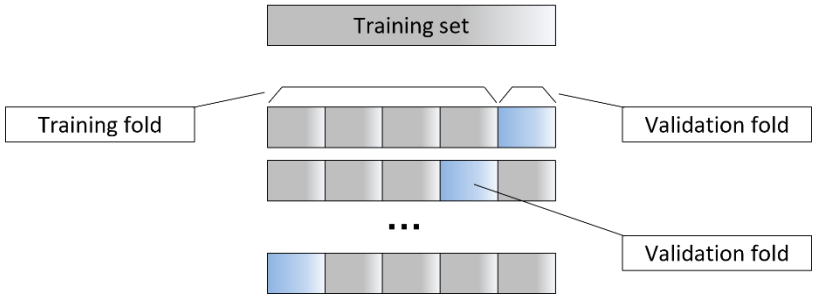
Validation set:
Modello1: errore % = 16,9
Modello2: errore % = 8,7

Si osserva che il modello sovra-parametrizzato (*modello1*) ha un basso errore nel training set, ma peggiora notevolmente nel validation set; mentre il secondo modello (*modello2*) ha un errore più alto nel training set, ma minore nel validation set.

Preparazione dell'ambiente per il training e il testing

La convalida incrociata (*k-fold cross-validation*) – 1/2

Per verificare se il modello è davvero in grado di generalizzare, si può utilizzare la tecnica *k-fold cross-validation dove il* training set viene diviso in k sottoinsiemi di uguale dimensione chiamati segmenti (*fold*). Ogni fold diventa un validation set e gli altri k-1 segmenti il training set.



Alla fine della procedura, **gli errori** calcolati **vengono mediat**i per fornire **una misura della stabilità** del modello[1] e, quindi, delle sue capacità predittive: se **l'accuratezza misurata in tutti i segmenti non varia** di molto significa che il **modello (e l'insieme dei dati) è consistente**; al contrario bisogna **rivedere il modello e i dati.**

[1] Lo scopo del cross-validation è la **verifica del modello, non la sua costruzione!** La tecnica usata invece per l'ottimizzazione del modello, che si basa anch'essa sul partizionamento dei dati di training, si chiama *bootstrap aggregation* (o "*bagging*") che verrà trattata in seguito.

Preparazione dell'ambiente per il training e il testing

La convalida incrociata (*k-fold cross-validation*) – 1/2

In questo esempio, l'accuratezza del modello, che è la media dell'accuratezza di ciascun fold, è 88,7%.

L'accuratezza è abbastanza simile in tutti i segmenti, ciò significa che il modello è consistente e si può essere confidenti che l'addestramento finale fornirà performance pressoché uguali.

D Error in %	I Size of Test Set	I Error Count	D Mean(Error in %)
13.171	410	54	11.322
11.707	410	48	11.322
11.463	410	47	11.322
9.756	410	40	11.322
10.513	409	43	11.322

- **Preparazione, ottimizzazione**

Dati sbilanciati

È noto che i modelli predittivi funzionano bene quando ogni classe è rappresentata in modo equo nei dati di addestramento.

Pertanto, se i dati sono **sbilanciati**, le **prestazioni** della maggior parte degli algoritmi di apprendimento standard saranno **compromesse,** poiché il loro scopo è massimizzare l'accuratezza complessiva.

Per una tabella di dati con eventi negativi del 99% e eventi positivi dell'1%, un modello potrebbe essere accurato al 99%, prevedendo che tutte le istanze come negative, essendo così del tutto inutile.

Per ovviare a questo problema si devono **ri-campionare** le righe nel training set in modo di alterare le proporzioni delle classi (la distribuzione a priori) del dati di training per ottenere un classificatore in grado di predire efficacemente la classe di minoranza.

Preparazione, ottimizzazione

Tecniche di ri-campionamento

Undersampling

Viene tolto un campione casuale dalla classe di maggioranza. Lo svantaggio di questo approccio è la potenziale perdita di informazioni per il processo di apprendimento.

Oversampling

Vengono replicate copie esatte degli eventi che rappresentano la classe di minoranza. Più istanze di determinate osservazioni possono rendere il classificatore troppo specifico causando problemi di *overfitting*.

SMOTE (Synthetic Minority Oversampling Technique)

La classe di minoranza viene sovra-campionata con osservazioni "sintetiche" generate attraverso la distanza tra i dati di minoranza e quelli a loro più vicini.

Preparazione, ottimizzazione

Tecniche di ri-campionamento, correzione dello score

Le ***probabilità ex-post*** ottenute utilizzando le tecniche di ri-campionamento dipendono dalla proporzione della colonna target presente nel training set. Per ottenere le **probabilità corrette** per ciascun evento, è necessario applicare una trasformazione ai risultati ottenuti.

Se P_1 è la *probabilità a priori* dell'evento nella popolazione e R_1 è la proporzione degli eventi nel training set, lo *score* ottenuto con il ri-campionamento **dovrà essere corretto** come segue[1]:

$$\Pr_{(1)}\, Corretta = \frac{\Pr_{(1)}\ (P_1/R_1)}{\Pr_{(1)}(P_1/R_1) + \ (1-\Pr_{(1)})\,(1-P_1)/(1-R_1)}$$

Ad esempio, se la probabilità a priori della popolazione è del 1% e quella nel training set è del 30% e lo score ottenuto è 0,95, con la correzione diventa 0,31:

$$\Pr_{(1)}\, Corretta = \frac{0{,}95 * 0{,}01/0{,}3}{0{,}95 * 0{,}01/0{,}3 + \ 0{,}05 * 0{,}99/0{,}7} = 0{,}3093$$

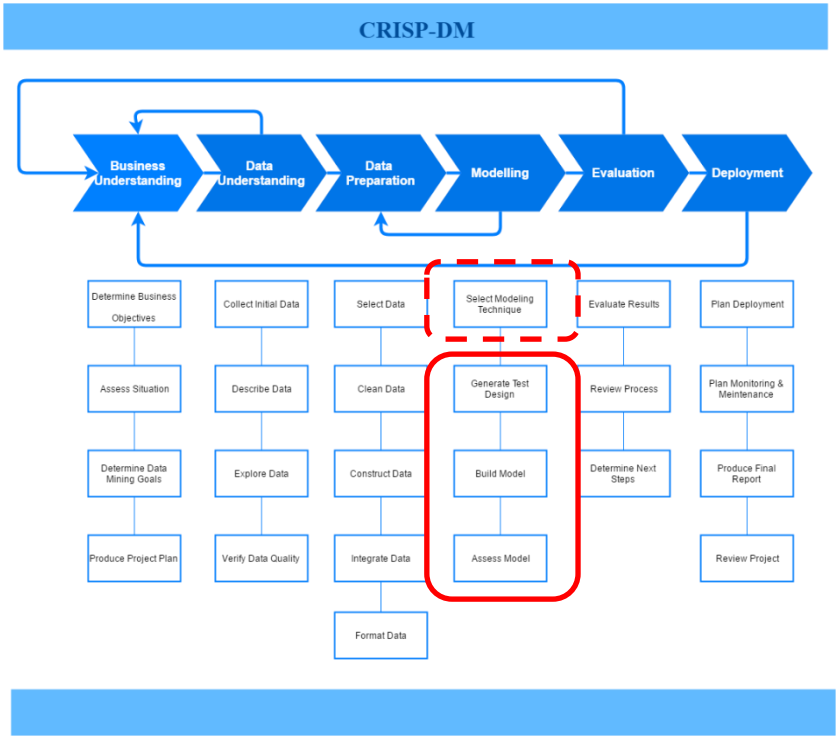
[1] Per approfondimento si veda il blog post: https://www.knime.com/blog/correcting-predicted-class-probabilities-in-imbalanced-datasets

La metodologia CRISP-DM – Il Modeling

Queste note trattano ampiamente la fase del ***Modeling***, dove i modelli vengono costruiti, testati e migliorati per ottimizzare le performance.

Il ***Modeling*** presuppone che i dati siano già stati preparati per sottoporli alle seguenti attività:

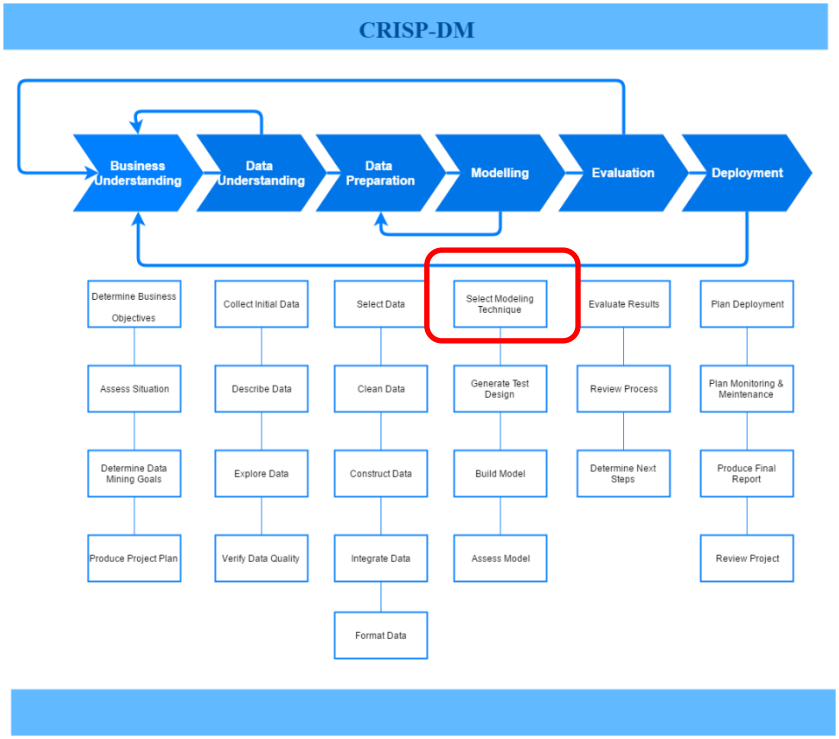
- Selezione della tecnica da utilizzare
- Preparazione dell'ambiente per l'addestramento e la validazione
- Costruzione del modello
- Validazione



Queste note trattano ampliamente la fase del ***Modeling*** dove i modelli vengono costruiti poi testati e calibrati per ottimizzare le performance.

Il ***Modeling*** presuppone che i dati siano già stati preparati per sottoporli alle seguenti attività:

- Selezione della tecnica da utilizzare
- Preparazione dell'ambiente per il training e il testing
- Messa a punto, ottimizzazione
- Validazione



Selezione della tecnica da utilizzare

Esiste una gran quantità di tecniche di modeling, da utilizzare secondo le esigenze.

- **Regressione Logistica**
- **Alberi di Decisione** e **Random Forest**
- **Reti Neurali** e **Deep Learning**
- Support Vector Machines
- Naïve Bayes
- …

La scelta sarà fatta in base al problema da analizzare, gli strumenti offerti dal software, l'esperienza di chi li applica e dal tipo di risultato che si vuole ottenere (ad esempio, molti preferiscono metodi che possano offrire risultati facili da interpretare: in questo caso si utilizzerebbero gli alberi di decisione o la regressione logistica, mentre le reti neurali potrebbero non essere accettate).

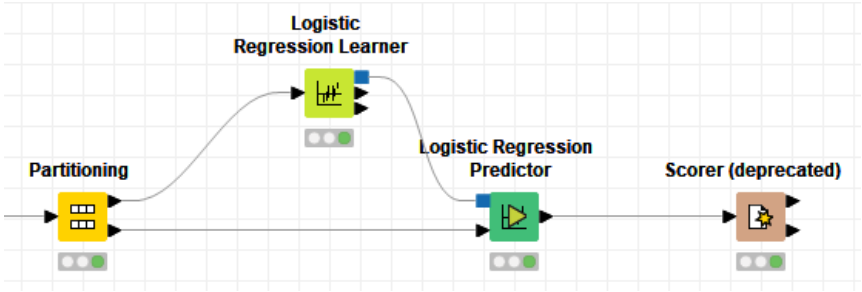
Selezione della tecnica da utilizzare

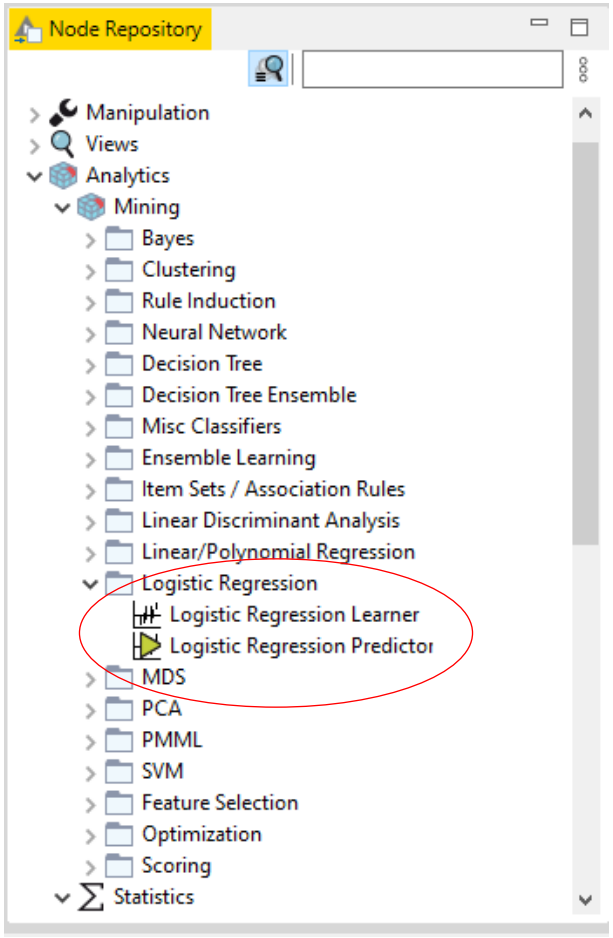
In KNIME è disponibile una vasta scelta di modelli predittivi.

Tutti i modelli predittivi si compongono di due parti:

- la parte di **apprendimento** (*learning*) che prende in input i dati del Training Set
- la parte di **predizione** (*prediction*) che prende in input i dati del Validation Set

Queste parti si distinguono in nodi di tipo **Learner** e in nodi di tipo **Predictor**.

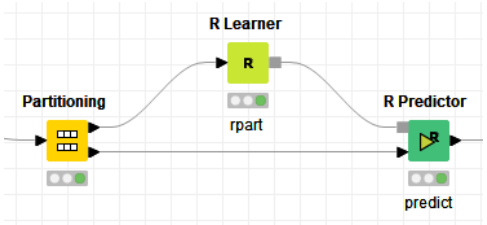




Selezione della tecnica da utilizzare

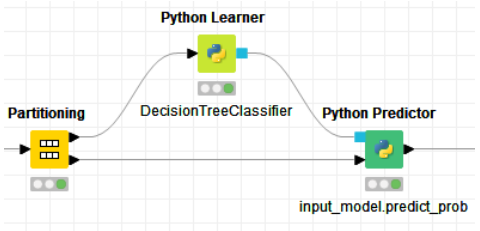
KNIME può anche eseguire modelli presi da **altri software**, per ogni software supportato ci sono nodi specifici.

Nodi R Learner e R Predictor




```
tree  <- rpart (data=knime.in, formula=Scelta ~ . ,
                parms=list(split='gini'),
                method="class",
                minsplit=2,
                maxdepth=5,
#               cp=0.000001,
#               xval=10
               )
```

Nodi Python Learner e Python Predictor




```
output_model = DecisionTreeClassifier(criterion = "gini",
                    random_state = 123,
#                   min_impurity_split=1e-07,
                    max_depth=5, min_samples_split=2)

output_model.fit(x, y)
```

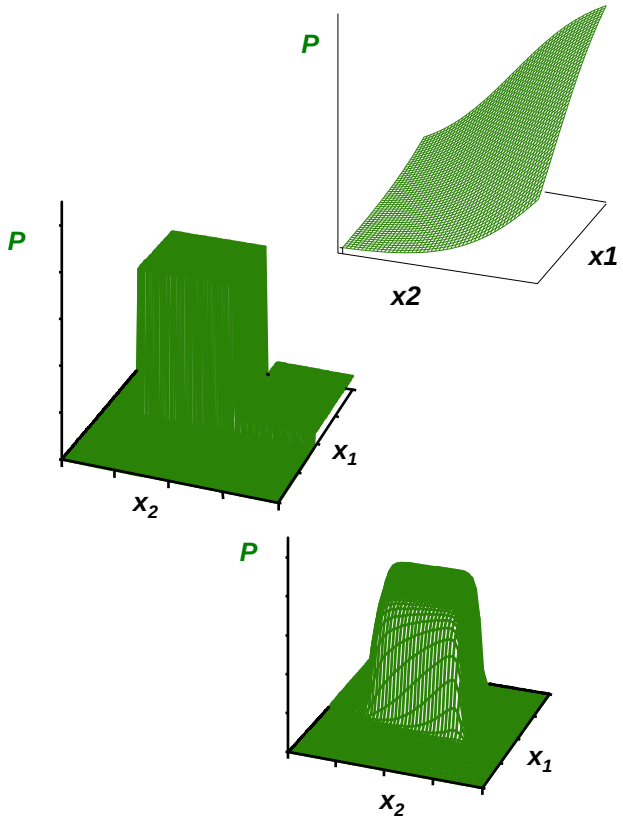
Modelli predittivi – Tipologie

La costruzione di un modello predittivo richiede di applicare ai dati di input specifiche metodologie statistiche. Tra quelle maggiormente utilizzate e qui trattate si trovano:

Regressione Logistica

Alberi di Decisione/ Foreste Casuali

Reti Neurali/ Deep Learning

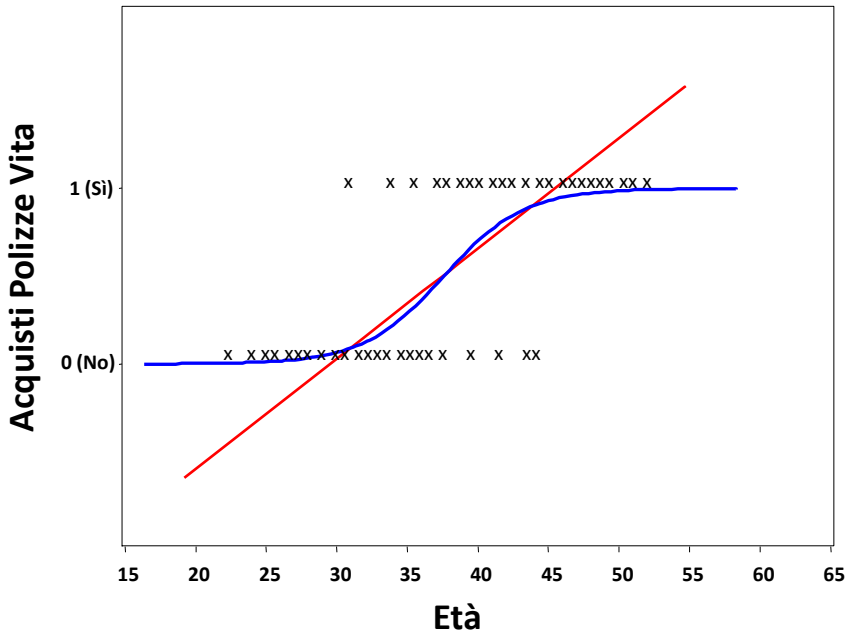


Regressione Logistica

I modelli predittivi con colonna **target di tipo binario non sono riconducibili ai modelli lineari** di tipo $y = a + bx$**.**

Una retta di regressione passerebbe, infatti, attraverso i valori di Y nei punti di maggior concentrazione e si potrebbero ottenere **predizioni superiori a 1 o negative**.

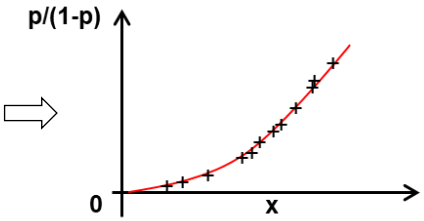
Con la regressione logistica invece, i valori predetti sono di tipo continuo e cadono **nell'intervallo tra 0 e 1** lungo una curva sigmoide.



Regressione Logistica

Per estendere allora l'intervallo di y, possibilmente tra -∞ e +∞, indicando la probabilità dell'evento **p = Pr(y=evento)**, si può scrivere l'equazione come:

$$\left(\frac{\mathbf{p}}{\mathbf{1-p}}\right) = a + bx$$



Il rapporto $\left(\frac{\mathbf{p}}{\mathbf{1-p}}\right)$, conosciuto come ***ODDS*** (o "*pronostico*"), è il rapporto tra la probabilità di un evento e quella del suo evento contrario. Viene usato per confrontare le due probabilità.

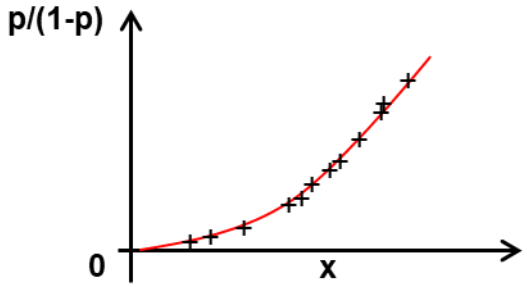
Per interpretare l'ODDS prendiamo questo studio di malattie cardiovascolari:

	Evento coronarico		
	Sì	No	Totale
Fumatore	166	1.176	1.342
Non Fumatore	50	513	563
Totale	216	1.689	1.905

La percentuale (probabilità) di eventi coronarici per i fumatori è del 12,4% (166/1.342); l'odds degli eventi coronarici per i fumatori è 0,1412 (0,124/(1-0,124) o, più semplicemente, 166/1.176), che si può semplificare come "*1 a 7*". Questo significa che, per i fumatori, c'è 1 possibilità di andare incontro a un evento coronarico contro 7 di non incorrervi. Per i non fumatori, c'è invece 1 possibilità contro 10.

Regressione Logistica

L'*ODDS*, però, **non è ancora applicabile** per un modello lineare, in quanto ha ancora un intervallo limitato (0, +∞).

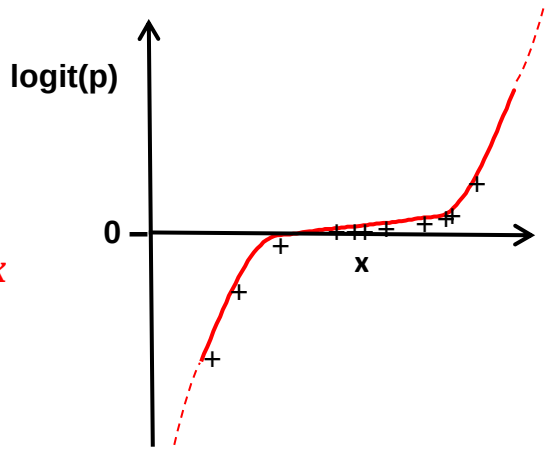


Per ovviare a questo si utilizza la funzione **logit** che è il **logaritmo naturale** dell'*ODDS*:

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$$

L'intervallo della funzione logit è infatti **compreso tra** -∞ e +∞

p	ODDS	logit
0	0	-∞
1/3	0,5	-0,69
0,5	1	0
2/3	2	0,69
1	+∞	+∞



per cui **si può applicare il modello lineare** ***logit(p) = a + bx***

Regressione Logistica – stima dei parametri

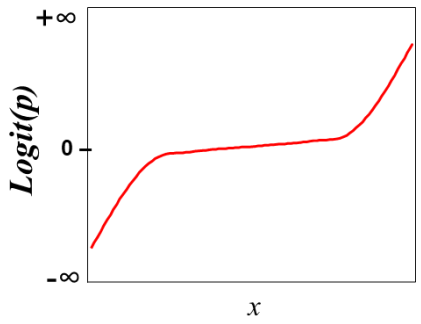
Un modello di regressione logistica multipla si può esprimere formalmente :

$$\text{logit}(\boldsymbol{p}) = \boldsymbol{b_0} + \boldsymbol{b_1 x_1} + \boldsymbol{b_2 x_2} + \ldots + \boldsymbol{b_n x_n}$$

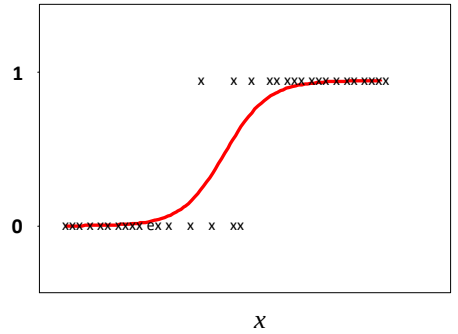
I parametri $\boldsymbol{b}$ dell'equazione, a differenza della regressione lineare, vengono stimati con il metodo della massima verosimiglianza che necessita **algoritmi di calcolo numerici** come il Fisher Scoring o il Newton-Raphson.

La probabilità si ottiene dai parametri dell'equazione applicando la trasformazione inversa

$$\ln\left(\frac{p}{1-p}\right) = a + bx \quad \rightarrow \quad \frac{p}{1-p} = e^{(a+bx)} \quad \rightarrow \quad \boldsymbol{p} = \frac{e^{(a+bx)}}{(1+e^{(a+bx)})} = \frac{\mathbf{1}}{(\mathbf{1}+\boldsymbol{e}^{-(\boldsymbol{a}+\boldsymbol{bx})})}$$



$$\boldsymbol{p} = \frac{1}{(1+e^{-\boldsymbol{Logit}(\boldsymbol{p})})}$$



per cui i valori predetti, come anticipato, sono di tipo continuo e cadono nell'**intervallo tra 0 e 1 lungo una curva sigmoide**.

Regressione Logistica – interpretazione dei risultati *(1/4)*

Dai parametri β dell'equazione lineare si può ricavare l'**ODDS RATIO (OR).**

L'OR è un rapporto di probabilità tra l'ODDS di "interesse" e l'ODDS di "riferimento".

Riprendendo l'esempio precedente a pag. 46, l'odds di un evento coronarico per i fumatori (interesse) è 0,1412 (166/1.176), mentre per i non fumatori è 0,0975 (50/513).

	Evento coronarico		
	Sì	No	Totale
Fumatore	166	1.176	1.342
Non Fumatore	50	513	563
Totale	216	1.689	1.905

L'**OR** sarà allora dato da $\frac{\frac{166}{1.176}}{\frac{50}{513}} = {}^{0,1412}/_{0,0975} = 1{,}4482$

L'odds di un evento coronarico è stimato essere circa **1,45 volte più grande per i fumatori** rispetto ai non fumatori[1].

In breve, calcolando la variazione percentuale (0,1412-0,0975)/0,0975 * 100 = 44,8%, si stima che il **fumare aumenti l'odds** di un evento coronarico **di circa il 45%**.

[1] Con una confidenza del 95%, il vero odds è compreso nell'intervallo l'intervallo [1,038-2,020].

Regressione Logistica – interpretazione dei risultati *(2/4)*

Nel caso della regressione logistica, è interessante valutare il ***logit*** quando x_i **sia uguale a 0**, nel qual caso si ha $ln\ (p/(1-p)) = b_0$, e quando **sia uguale a 1** (incremento unitario), nel qual caso si ha $ln\ (p'/(1-p')) = b_0 + b_i$, mantenendo costanti i valori di tutte le altre colonne.

Il rapporto tra l'ODDS della colonna di input x_i e l'ODDS del suo incremento unitario $x_i + 1$ è dato da:

$$ln\ (p'/(1-p')) - ln\ (p/(1-p)) = ln\ [(p'/(1-p'))/(p/(1-p))] = \mathbf{ln\ (OR)} = b_0 + b_i - b_0 = \mathbf{b_i}$$

Considerando l'OR come **rapporto tra l'ODDS della colonna di input x_i e l'ODDS del suo incremento unitario $x_i + 1$,** si ricava che

$$OR_i = e^{b_i}$$

L'OR può essere quindi interpretato come una **misura dell'associazione tra la colonna target e una colonna di input** (che è negativa se OR<1, è positiva se OR>1, nessuna se OR=1).

Ad esempio, con un coefficiente b_i=0,1856, l'OR è 1,204. Questo significa che, ad un incremento della colonna di input x_i, si attende una variazione di circa 1,2 volte (il 20,4%) dell'odds dell'evento di interesse della colonna target, mantenendo costanti tutte le altre colonne di input. Se invece b_i=0,1856 l'OR è 0,458: la variazione attesa è -0,542 (1-0,458), che corrisponde a un decremento del 54,2%.

Regressione Logistica – interpretazione dei risultati *(3/4)*

Gli OR sono però difficili da interpretare. Si può però calcolare l'**incremento di probabilità** (**Delta-p**) della colonna target **per ogni incremento unitario di una singola colonna di input**, mantenendo costanti i valori di tutte le altre colonne.

L'**incremento di probabilità,** per una singola colonna di input, è dato da $\mathbf{P_{post}} - \mathbf{P_{prior}}$ dove $\mathbf{P_{prior}}$ **è la probabilità a priori** e $\mathbf{P_{post}}$ **è la probabilità a posteriori** stimata con la trasformazione inversa del logit per la sola colonna di input x_i a un suo incremento unitario.

Per ogni colonna di input x_i , il Delta-p si calcola con questi semplici passaggi:

$$P_{prior} = Pr(y);$$

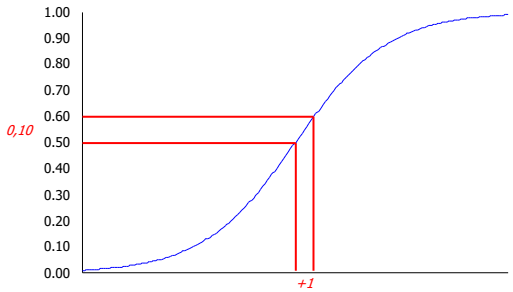
$$logit_{prior} = \ln\left(\frac{P_{prior}}{1 - P_{prior}}\right);$$

$$logit_{post} = logit_{prior} + b_i;$$

Si ha quando la x_i passa da 0 a 1

$$P_{post} = \left(\frac{e^{logit_{post}}}{1 + e^{logit_{post}}}\right);$$

$$\boldsymbol{Delta{-}p} = \mathbf{p}_{\boldsymbol{post}} - \mathbf{p}_{\boldsymbol{prior}}$$



Regressione Logistica – interpretazione dei risultati *(3/4)*

Gli OR sono però difficili da interpretare. Si può però calcolare l'**incremento di probabilità** (**Delta-p**) della colonna target **per ogni incremento unitario di una singola colonna di input**, mantenendo costanti i valori di tutte le altre colonne.

L'**incremento di probabilità,** per una singola colonna di input, è dato da $\mathbf{P_{post}} - \mathbf{P_{prior}}$ dove $\mathbf{P_{prior}}$ **è la probabilità a priori** e $\mathbf{P_{post}}$ **è la probabilità a posteriori** stimata con la trasformazione inversa del logit per la sola colonna di input x_i a un suo incremento unitario.

Per ogni colonna di input x_i , il Delta-p si calcola con questi semplici passaggi:

$$P_{prior} = Pr(y);$$

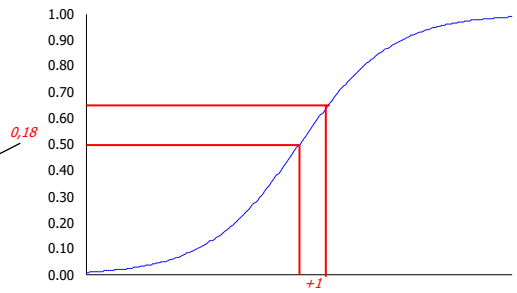
$$logit_{prior} = \ln\left(\frac{P_{prior}}{1 - P_{prior}}\right);$$

$$logit_{post} = logit_{prior} + b_i;$$

Si ha quando la x_i passa da 0 a 1

$$P_{post} = \left(\frac{e^{logit_{post}}}{1 + e^{logit_{post}}}\right);$$

$$\boldsymbol{Delta{-}p} = \mathbf{p}_{\boldsymbol{post}} - \mathbf{p}_{\boldsymbol{prior}}$$



A un incremento unitario della x_i si ha una variazione della probabilità del +18%.

Regressione Logistica – interpretazione dei parametri *(4/4)*

Da un'indagine di una Banca sulla Customer Satisfaction dei propri clienti si vuole calcolare l'**incremento percentuale dei clienti soddisfatti** che si avrebbe migliorando il solo giudizio "*capacità di risolvere i problemi*".

La **Soddisfazione Complessiva** è misurata in una scala da 1 a 5; la colonna **target** del modello è di tipo binario, che assume il valore "No" se la Soddisfazione Complessiva è 1,2 o 3, "Sì" se è 4 o 5.

La **percentuale dei clienti soddisfatti** a priori, $P_{prior} = Pr\,(y = Sì)$ è del **48,1%**; la **media** del giudizio $\boldsymbol{x_i}$, "*capacità di risolvere i problemi*", in una scala da 1 a 5, è **3,1**.

Il modello di regressione del *logit* stima, per questa colonna, il parametro $\boldsymbol{b_i}$=**0,8764**.
Applicando le formule esposte precedentemente si ha:

$$logit_{prior} = \ln\left(\frac{0{,}481}{1-0{,}481}\right) = -0{,}076\,; \quad logit_{post} = -0{,}076 + 0{,}8764 = 0{,}8004 \quad \mathrm{p}_1 = \left(\frac{e^{0{,}8004}}{1+e^{0{,}8004}}\right) = \mathbf{0{,}69};$$

$$\boldsymbol{Delta\,p = p_1 - p_0} = 0{,}69 - 0{,}481 = \mathbf{0{,}209}$$

Un **incremento unitario** del giudizio sulla "*capacità di risolvere i problemi*" (portandolo, in media, da 3,1 a 4,1), porterebbe la **percentuale di clienti soddisfatti dal 48% al 69%,** con un incremento di circa il 21%.

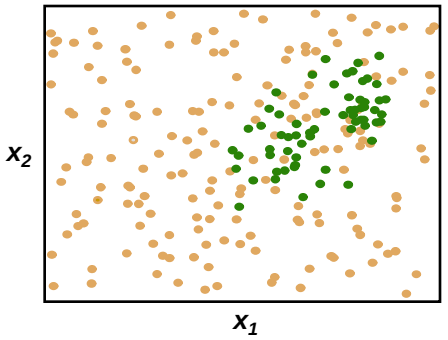
Regressione logistica - esempio teorico

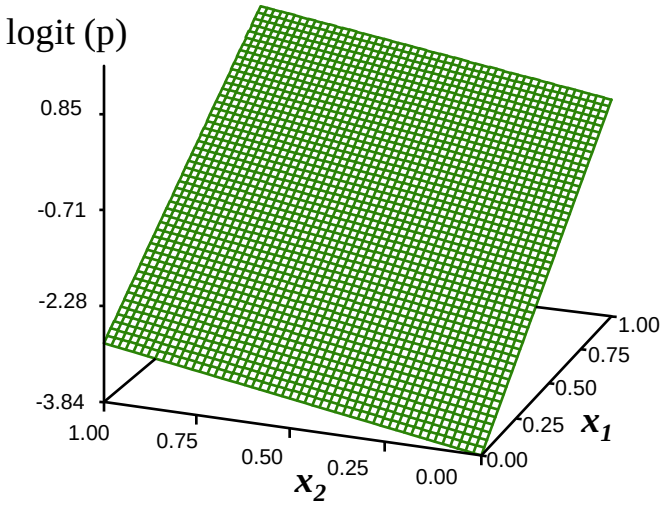
Nell'esempio seguente si è addestrato un modello logistico con un training set di 240 casi dei quali il 26% sono di classe 1 (verdi).

$\mathbf{logit}(p) = -3{,}84 + 3{,}73\, x_1 + 0{,}966\, x_2$

Il grafico del logit con due colonne di input è un piano.

Il piano interseca l'origine (0,0) a -3,84. Nella direzione x1, il piano incrementa di 3,73 per un incremento unitario di x1; nella direzione x2, il piano incrementa di 0,966 per un incremento unitario di x2.





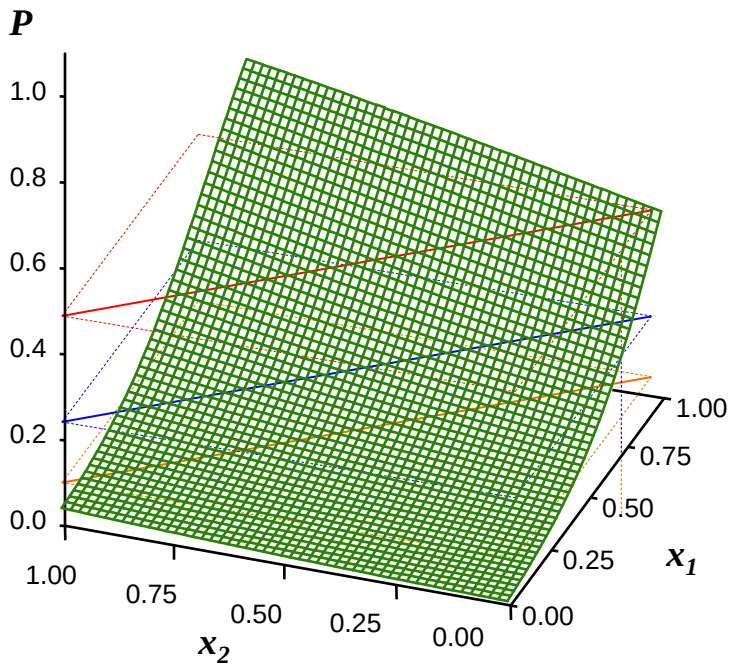
Regressione logistica - esempio teorico

Il logit viene ricondotto alla scala di probabilità con la trasformazione inversa: $p = \frac{e^{Logit}}{\left(1 + e^{Logit}\right)}$

Il grafico delle probabilità a posteriori con due colonne di input è una superficie sigmoide che ha la stessa tendenza e orientamento del piano nel grafico del logit eccetto nei punti dove la probabilità è vicina a 0 o a 1.

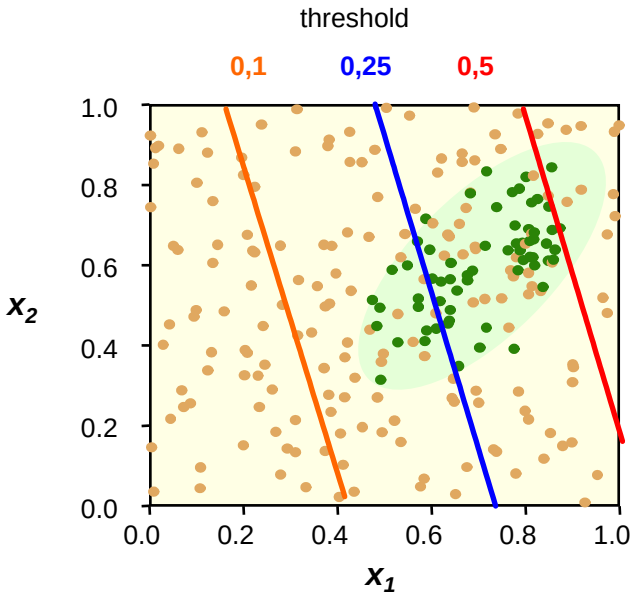
La non linearità si trova infatti agli estremi, dove la superficie deve piegarsi per rispettare i vincoli della scala (0,1). Non si vede la parte sigmoidale della metà alta della superficie perché difficilmente P tende a 1. Da notare che la superficie ottenuta riflette soprattutto i cambiamenti in x_1.

I tre piani tagliano il livello di probabilità su tre valori di *threshold*.



Regressione logistica - esempio teorico

Con il valore di *threshold* di 0,1 non verrebbero classificati molti veri negativi (il tasso dei falsi positivi sarebbe molto alto), mentre con un valore di 0,5 non verrebbero classificati correttamente moltissimi veri positivi (il tasso dei falsi negativi sarebbe altissimo).



Con il valore di *threshold* di 0,25 si avrebbe la maggior accuratezza (66,7%): in questo caso verrebbero classificati correttamente 160 casi con 68 falsi positivi e 12 falsi negativi.

		Valori Predetti 0	*Valori Predetti* 1	
Valori Reali	0	109(62%)	68(38%)	177
	1	12(19%)	51(81%)	63
		121	119	240

Utilizzo del software Knime - Modulo2_Esempio2

Questi dati riguardano uno studio degli effetti analgesici dei trattamenti su pazienti anziani sofferenti di nevralgia.

Colonna target

Dolore: No **(evento di interesse)**,
Sì **(evento di riferimento)**

Colonne di input

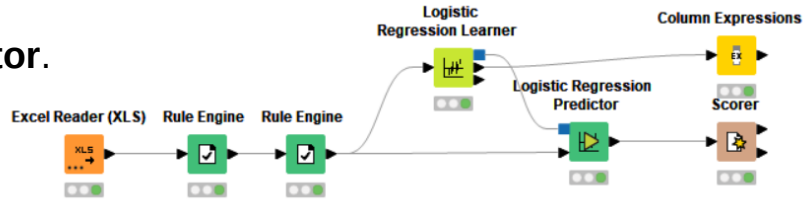
Terapia: A=farmaco A,
B=farmaco B,
P=placebo **(riferimento)**,
Sesso: F,
M **(riferimento)**,
Età: anni

Dolore	Terapia	Sesso	Eta
No	Placebo	F	68
No	Farmaco B	M	74
No	Placebo	F	67
Sì	Placebo	M	66
No	Farmaco B	F	67
No	Farmaco B	F	77
No	Farmaco A	F	71
No	Farmaco B	F	72
Sì	Farmaco B	F	76
Sì	Farmaco A	M	71
No	Farmaco A	F	63
Sì	Farmaco A	F	69
No	Farmaco B	F	66
No	Farmaco A	M	62
Sì	Placebo	F	64
No	Farmaco A	F	64

(continua)

Utilizzo del software Knime - Modulo2_Esempio2

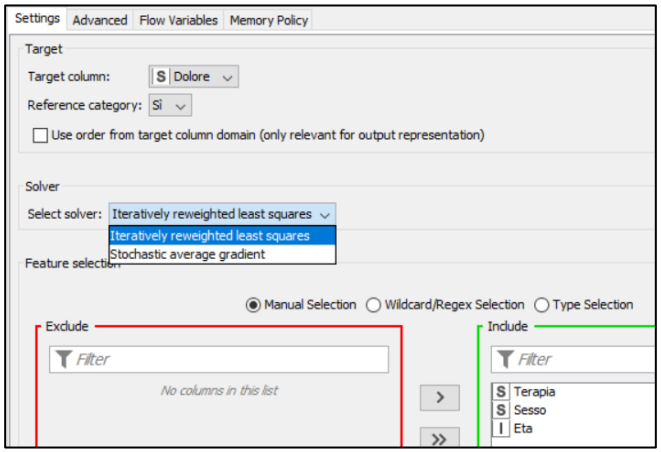
Per questo studio sono stati usati i nodi **Logistic Regression Learner** e **Logistic Regression Predictor**.



La colonna target è *Dolore*, e la categoria di riferimento (la classe per la quale **non** si desidera modellare la probabilità) è "*Sì*"

Sono disponibili 2 tipi di metodi numerici iterativi (*solver*):

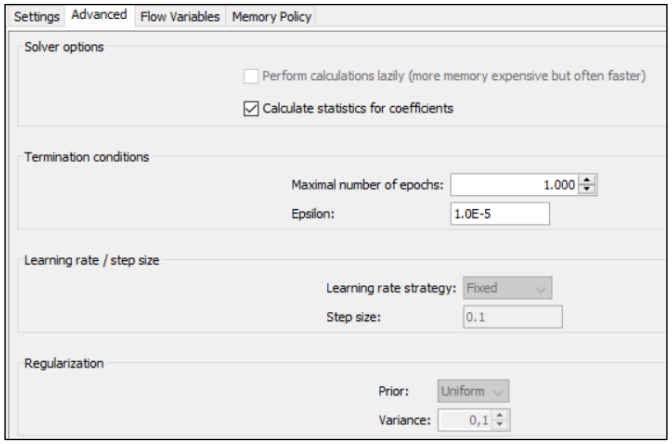
- *Iteratively reweighted least squares*, che equivale al Fisher Scoring[1]
- *Stochastic average gradient*



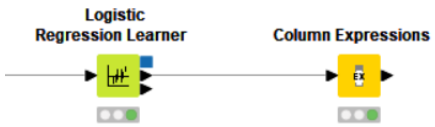
[1] Consigliato.

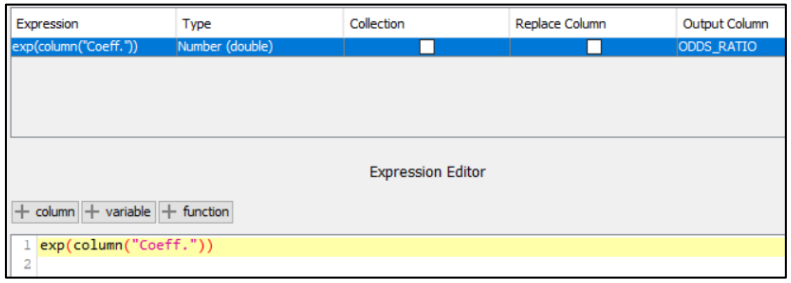
Utilizzo del software Knime - Modulo2_Esempio2

Nella scheda Advanced si possono impostare opzioni più avanzate (alcune sono specifiche per il solver utilizzato):



Il nodo Logistic Regression Learner non fornisce nella porta di output l'ODDS RATIO: per questo è stato aggiunto il nodo **Column Expressions** per il suo calcolo (e^{b_i}).





Utilizzo del software Knime - Modulo2_Esempio2

I risultati del modello che calcola la probabilità che la colonna target *Dolore* assuma il valore "*No*" portano a queste conclusioni:

Variable	Coeff.	Std. Err.	P>|z|	ODDS_RATIO	Delta-P
Terapia=3. Farmaco B	3.726	1.134	0.001	41.528	39,98%
Terapia=2. Farmaco A	3.179	1.014	0.002	24.022	38,78%
Sesso=2. F	1.824	0.792	0.021	6.194	31,33%
Eta	-0.265	0.096	0.006	0.767	-6,55%

- Tutti i parametri sono significativi al 95% (P>|z| < 0,05)

- A parità di sesso ed età, **entrambe le terapie**, in particolare la B, **sono efficaci rispetto al placebo** nella riduzione del dolore come si evince dagli ODDS RATIO confrontando il farmaco A e il farmaco B con il placebo (rispettivamente 24 e 41 volte di più, con un incremento di probabilità del 38,8% e 40%).

- A parità di terapia ed età, le **donne riportano 6 volte più sollievo degli uomini** (ODDS RATIO F vs M = 6,194, con un incremento di probabilità del 31,3%) .

- A parità di terapia e sesso, l'ODDS RATIO per l'età (0,77) mostra che **i pazienti più anziani riportano meno sollievo di quelli più giovani**. Vale a dire che i pazienti più giovani hanno un miglioramento maggiore rispetto ai pazienti più anziani (lo si può anche dedurre dal suo parametro che ha un valore negativo: -0,265 e dal decremento di probabilità del 6,6%).

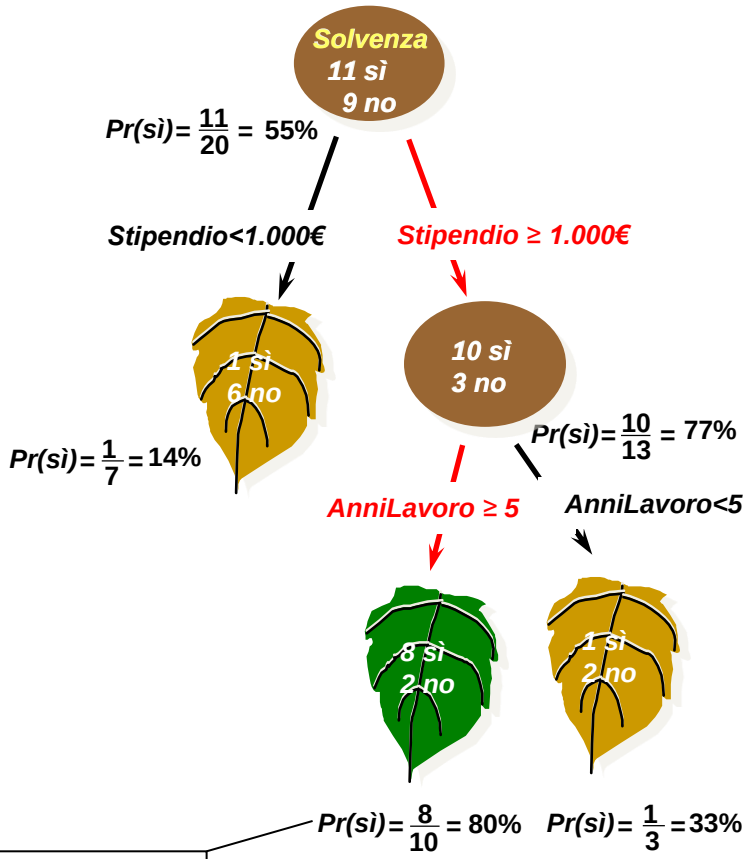
Alberi di Decisione

È una delle tecniche di classificazione maggiormente utilizzate che permette di rappresentare con un albero un insieme di **regole di classificazione**.

La struttura è di tipo gerarchico e consiste di insieme di nodi (dove ogni nodo rappresenta la colonna di input che meglio suddivide i valori della colonna target) e di rami (dove ogni ramo rappresenta una **regola di separazione**).

L'insieme delle **regole di separazione** che si incontrano lungo il cammino, dal nodo **radice** a un nodo **terminale** (foglia), è una **regola di classificazione**.

La capacità di spiegare il modello sotto la forma di regole esplicite, permette una facile interpretazione dei risultati.



SE *(Stipendio ≥ 1000)*
E *(AnniLavoro ≥ 5)*
ALLORA *ProbabilitaSolvenza=0,8*

Alberi di Decisione – Criteri di suddivisione

La suddivisione avviene attraverso una **procedura di tipo top-down**: l'algoritmo sceglie **la colonna di input più importante** ai fini della classificazione e applica la **condizione che fornisce la miglior separazione** dei valori della colonna target creando così dei **nodi figli** sui quali si ripete lo **stesso calcolo** finché non risulta soddisfatta una condizione per l'arresto della procedura.

Tra i criteri più conosciuti su cui si basano gli algoritmi più diffusi per trovare la miglior suddivisione ci sono:

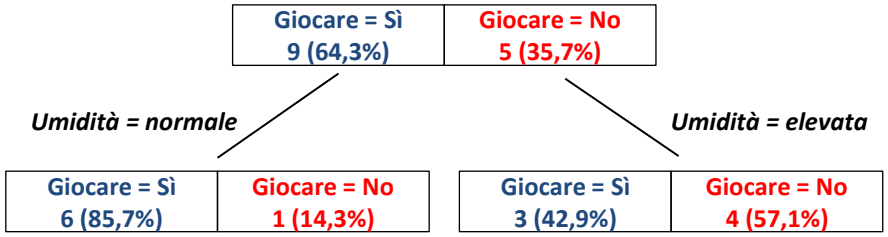
Criterio	Algoritmo
Test del chi-quadrato	CHAID
Indice di eterogeneità di Gini	CART
Misura dell'Entropia	C4.5, C5.0

Alberi di Decisione – Criteri di suddivisione, dati di esempio

Per illustrare i criteri di suddivisione, viene usata la famosa tabella del gioco del tennis:

Giocare	Meteo	Temperatura	Umidità	Vento
No	soleggiato	caldo	elevata	debole
No	soleggiato	caldo	elevata	forte
Sì	nuvoloso	caldo	elevata	debole
Sì	piovoso	mite	elevata	debole
Sì	piovoso	freddo	normale	debole
No	piovoso	freddo	normale	forte
Sì	nuvoloso	freddo	normale	forte
No	soleggiato	mite	elevata	debole
Sì	soleggiato	freddo	normale	debole
Sì	piovoso	mite	normale	debole
Sì	soleggiato	mite	normale	forte
Sì	nuvoloso	mite	elevata	forte
Sì	nuvoloso	caldo	normale	debole
No	piovoso	mite	elevata	forte

Prendendo come esempio come colonna target *Umidità*, la suddivisione sarà:



Alberi di Decisione – Criterio *chi-quadrato (2/4)*

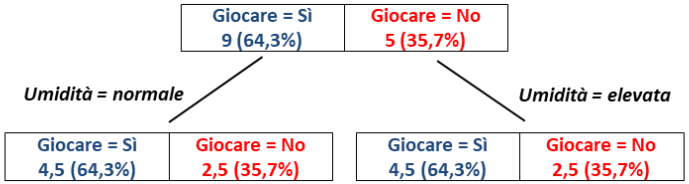
Il test chi quadrato (**chi-squared test**) di Pearson viene utilizzato per valutare la correlazione tra la colonna padre e la colonna figlio organizzate in una tabella di contingenza.

$$\chi^2 = \sum_{i=1}^{r*c} \frac{(o_i - e_i)^2}{e_i}$$

Dove o_i sono le frequenze osservate, mentre e_i sono le frequenze attese in caso di indipendenza e r, c sono rispettivamente le righe e colonne della tabella. Dal valore χ^2 si ottiene il valore p (***p-value***[1]): più basso è il valore, maggiore è la correlazione tra nodo genitore e nodo figlio.

Verrà scelta per la suddivisione **la colonna di input che fornirà il *p-value* più basso.**

Dalla suddivisione precedente, si ottiene le tabella delle frequenze osservate e di quelle attese:



Per la colonna Umidità si avrà un χ^2 del 2,8 e un p-value del **0,094**

[1] E' l'errore che si commette rifiutando l'ipotesi che le due colonne siano indipendenti.

Alberi di Decisione – Impurity e Information Gain

Gli altri criteri si basano sul concetto di Impurità (o "*Impurity*").

L'impurità è una **misura della proporzione delle osservazioni classificate in ciascuna classe**; è massima quando tutte le classi del nodo padre sono presenti nella stessa proporzione nel nodo figlio, mentre è minima quando il nodo figlio contiene osservazioni appartenenti a un'unica classe.

I metodi comunemente usati per calcolare l'impurità sono il Tasso di Errata Classificazione, l'Entropia e l'indice di Gini.

Alberi di Decisione – Criterio *Misclassification Rate (1/4)*

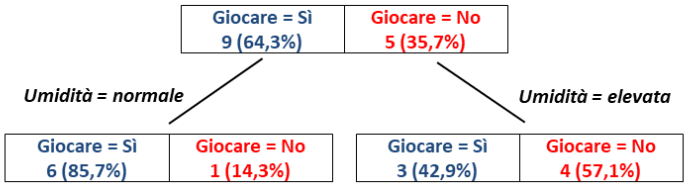
Il tasso di errata classificazione[1] (***Misclassification Rate***) è la percentuale p dei dati non classificati correttamente in ogni nodo figlio i.

$$Error(i) = 1 - \max_{j \in c}(p_{(j|i)})$$ dove c è il numero delle classi.

Verrà scelta per la suddivisione la **colonna che fornirà il valore di impurità (Impurity) più basso**. L'*Impurity* della suddivisione s è data dalla somma pesata degli errori (dove m sono le osservazioni del nodo padre e m_i quelle del nodo figlio i).

$$Impurity\,(s) = \sum_{i=1}^{n} \frac{m_i}{m} Error\,(i)$$

Riprendendo la suddivisione:



Si avrà

$$Impurity = 7/14 * (1- 0{,}857) + 7/14 * (1 – 0{,}571) = \mathbf{0{,}2857}$$

[1] **Questo criterio ha delle carenze in quanto non favorisce nodi puri**, cioè nodi che appartengono ad una sola classe. Prendendo ad esempio due suddivisioni, una (10,40) e (40,10), l'altra (50, 20) e (0, 30: l'errore di classificazione è sempre del 20%, ma la seconda ha un nodo puro che è da preferirsi rispetto alla prima.

Alberi di Decisione – Impurity e Information Gain

Per determinare la bontà della divisione può essere usato il criterio del **Guadagno Informativo** (o "*Information gain*") che è la differenza tra l'impurità presente nel nodo padre e quella, pesata, nel nodo figlio: maggiore è la loro differenza, migliore è la condizione scelta.

$$Information\ gain = \begin{matrix} Impurità \\ prima\ della\ divisione \end{matrix} - \begin{matrix} Impurità\ pesata \\ dopo\ la\ divisione \end{matrix}$$

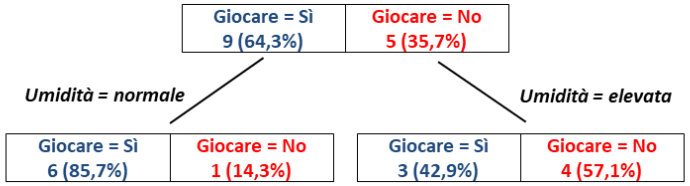
Alberi di Decisione – Criterio *Entropia (3/4)*

Questo criterio calcola l'entropia (**Entropy**) del nodo i **prima** e **dopo** la suddivisione s.

$$Entropy(i) = -\sum_{j=1}^{c} p_{(j|i)}\, log_2 p_{(j|i)}$$

La differenza tra i due indici darà il "guadagno" di informazione (***Information Gain)*** dopo la suddivisione s. Verrà quindi scelta **la colonna che avrà il valore di Information Gain più alto**.

Utilizzando la suddivisione precedente



si calcolerà anche qui l'*Impurity* pesata
(m oss. nodo padre e m_i oss. nodo figlio i)

$$Impurity\,(s) = \sum_{i=1}^{n} \frac{m_i}{m}\, Entropy(i)$$

Si avrà **prima** della suddivisione

Impurity = -9/14*log(9/14) -5/14*log(5/14) = 0,94029

e **dopo**

Impurity = 7/14*-(1/7*log(1/7)+6/7*log(6/7)) + 7/14*-(4/7*log(4/7)+3/7*log(3/7)) = 0,78845

$$Information\ Gain = 0.94029 - 0,78845 = \mathbf{0,15184}$$

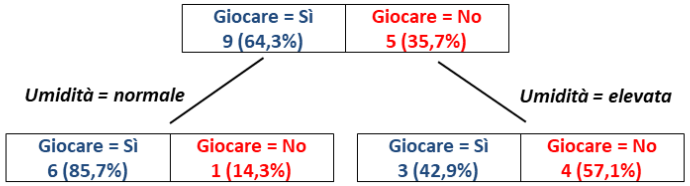
Alberi di Decisione – *Criterio Gini (4/4)*

Questo criterio calcola l'indice Gini (**Gini index**) del nodo i **prima** e **dopo** la suddivisione s.

$$Gini\,(i) = 1 - \sum_{j=1}^{c} p_{(j|i)}^2$$

Anche qui la differenza tra i due indici darà il "guadagno" di informazione (***Gain)*** dopo la suddivisione s. **Verrà quindi scelta la colonna che avrà il valore di Gain più alto.**

Utilizzando la suddivisione precedente



si calcolerà anche qui l'*Impurity* pesata
(m oss. nodo padre e m_i oss. nodo figlio i)

$$Impurity\,(s) = \sum_{i=1}^{n} \frac{m_i}{m}\, Gini\,(i)$$

Si avrà **prima** della suddivisione

Impurity = 1 - ((5/14)^2 + (9/14)^2) = 0,45918

e **dopo**

Impurity = 7/14(1-(3/7)^2-(4/7)^2) + 7/14*(1-(6/7)^2-(1/7)^2) = 0,36735*

$$Gini\ Gain = 0.45918 - 0,36735 = \mathbf{0.09184}$$

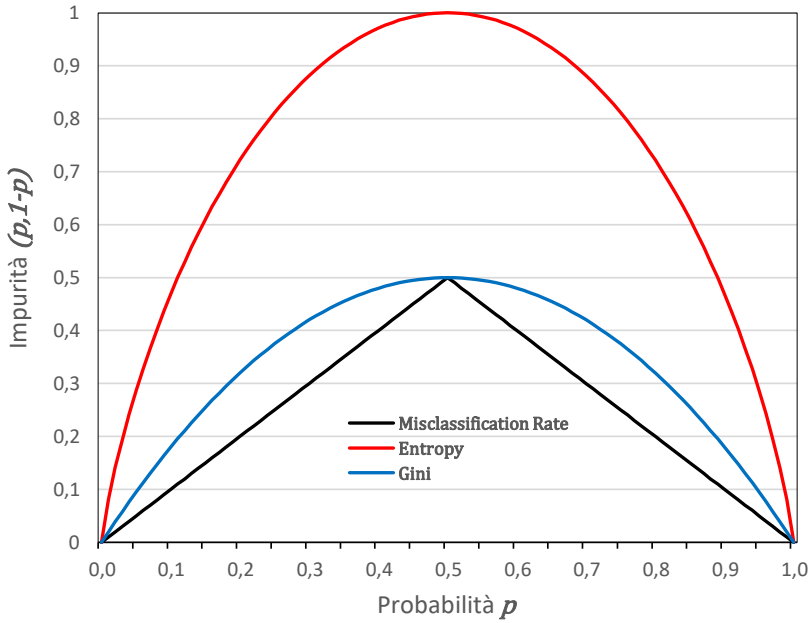
Alberi di Decisione – Comparazione dei criteri di suddivisione

Riassumendo, la caratteristica fondamentale degli alberi di decisione è quella di calcolare l'**impurità** nei dati in quanto fornisce una **misura della loro eterogeneità**. Questo può essere fatto con vari criteri, ognuno dei quali fornisce un valore a seconda della **probabilità** di appartenere a una certa **classe**.

La **colonna di suddivisione** che verrà scelta in ogni nodo sarà quella che avrà la **minor impurità**.

Nel caso di suddivisione in 2 classi, i valori di impurità in funzione delle probabilità sono illustrati nel grafico a lato.

Ogni criterio dà un valore di **impurità uguale a 0** quando la **probabilità** di appartenere a una classe è **0** oppure **1** (p.e. in un cesto di palline non c'è alcuna nera o lo sono tutte); il **valore massimo di impurità** si raggiunge quando entrambe le classi hanno **la stessa probabilità,** per cui si ha il massimo di eterogeneità (la metà sono bianche e l'altra metà sono nere).



Alberi di Decisione – Criteri di suddivisione, scelta della colonna

Eseguendo i calcoli per le altre colonne di input della tabella si ottengono questi risultati:

	Misclassification Rate (⇩)	Chi-square p-value (⇩)	Gini Gain (⇧)	Information Gain (⇧)
Umidità	**0,28571**	**0,09426**	0,09184	0,15184
Vento	0,35714	0,33400	0,03061	0,04813
Meteo	**0,28571**	0,16977	**0,11633**	**0,24675**
Temperatura	0,35714	0,75188	0,01871	0,02922

Per 3 criteri (Misclassification Rate, Gini ed Entropy), la colonna **Meteo** *(soleggiato, nuvoloso, piovoso)* risulta essere il **miglior classificatore** di primo livello e la colonna Temperatura il peggiore.

Da notare il tasso di errata classificazione, che è uguale sia per le colonne Umidità e Meteo sia per le colonne Vento e Temperatura.

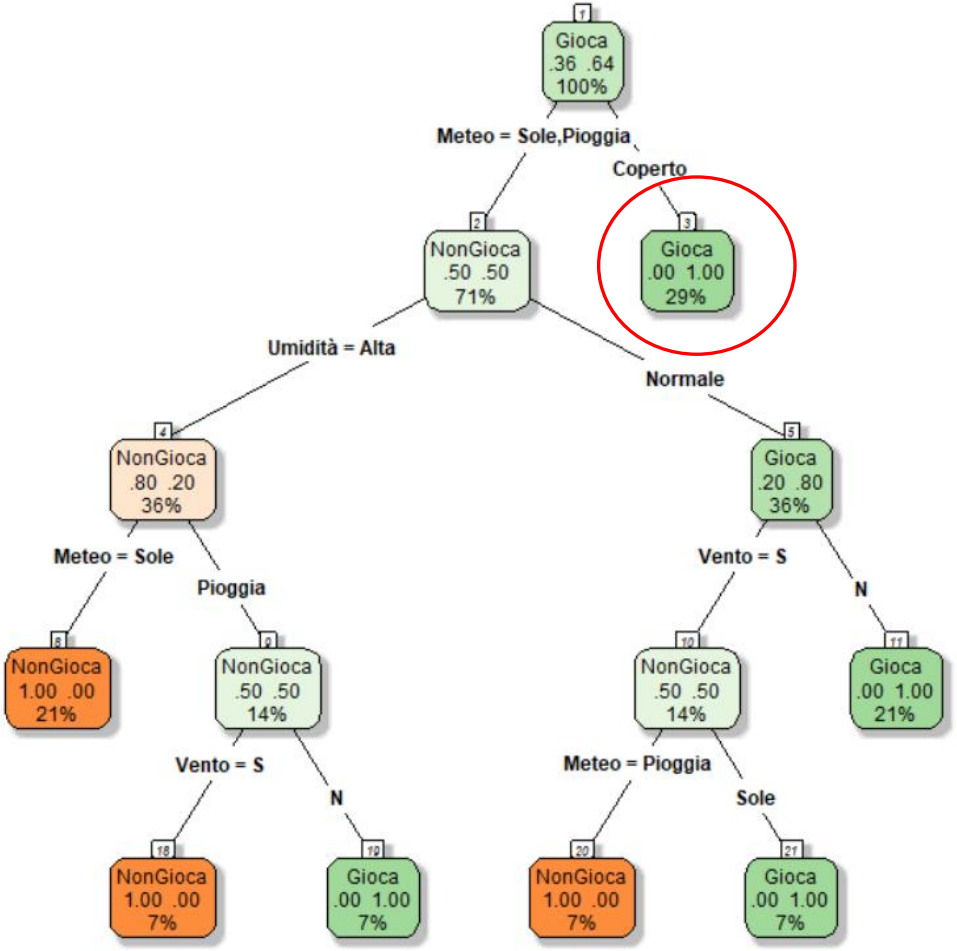
Alberi di Decisione – Rappresentazione grafica

Scegliendo come criterio di suddivisione l'**indice di Gini** si avrà un albero così composto:

Da notare la presenza al primo livello di un nodo terminale (foglia) dove non è possibile un'ulteriore suddivisione, avendo una purezza del 100%

Si applica lo stesso procedimento per ogni nodo figlio, ottenendo altre suddivisioni.

La costruzione dell'albero si ferma quando ogni nodo è una foglia, oppure risulta soddisfatta una condizione per l'arresto della procedura.



Alberi di Decisione – Regole

Gli alberi di decisione, a ogni nodo finale (foglia), associano una regola che può essere facilmente compresa dall'utente finale essendo espressa in linguaggio naturale con costrutti *IF THEN* nella forma:

IF *condizione1 AND condizione2* AND *condizione3 THEN decisione*[1].

Regola	Scelta	N. oss. Foglia	N. oss. Corrette	N. oss. Errate
($Meteo$ IN ("Coperto"))	Gioca	4	4	0
($Meteo$ IN ("Sole") AND $Umidita$ IN ("Alta") AND $Meteo$ IN ("Sole", "Pioggia"))	NonGioca	3	3	0
($Meteo$ IN ("Coperto", "Pioggia") AND $Umidita$ IN ("Alta") AND $Meteo$ IN ("Sole", "Pioggia"))	NonGioca	2	1	1
($Vento$ IN ("N") AND $Umidita$ IN ("Normale") AND $Meteo$ IN ("Sole", "Pioggia"))	Gioca	3	3	0
($Vento$ IN ("S") AND $Umidita$ IN ("Normale") AND $Meteo$ IN ("Sole", "Pioggia"))	NonGioca	2	1	1

[1] In base al valore di probabilità e dal valore di soglia impostato (threshold).

Alberi di Decisione – Il problema del sovra-adattamento (*overfitting*)

Per alberi molto complessi questo metodo può portare al fenomeno dell'*overfitting*. Ci sono diversi modi per evitare l'*overfitting*, i più semplici consistono nel "potare" l'albero (*pruning*):

Pre-Pruning

- **minsplit (R)** /
  min_samples_split (Python) /
  min number records per node (KNIME)

  è il numero minimo di record che deve esserci in un nodo per poter fare una suddivisione.

- **maxdepth (R)** /
  max_depth (Python)

  è la profondità massima del modello, il percorso più lungo dal nodo radice al nodo terminale (foglia).

- **minbucket (R)** /
  min_samples_leaf (Python)

  è il numero minimo di record che ci possono essere in un nodo foglia.

Post-Pruning

- rimuove i rami che fanno uso di caratteristiche che hanno poca importanza. Uno dei metodi usati è Minimum Description Lenght (**MDL**).

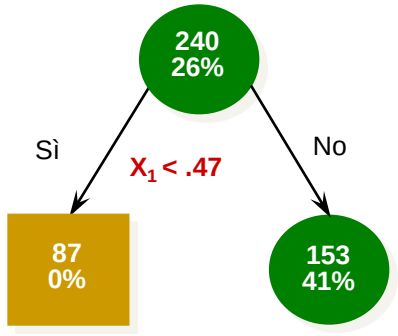
Alberi di Decisione – Esempio teorico (1/2)

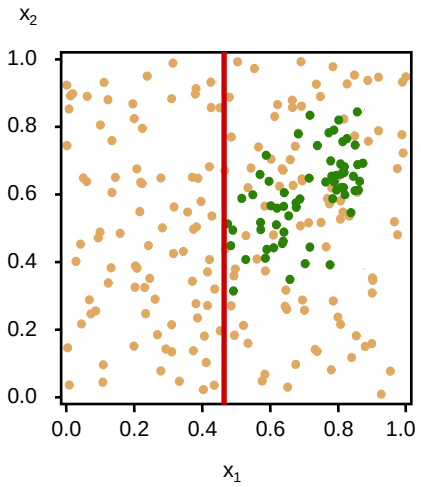
Il nodo radice contiene 240 osservazioni, di cui il 26% sono di classe 1 (verdi).

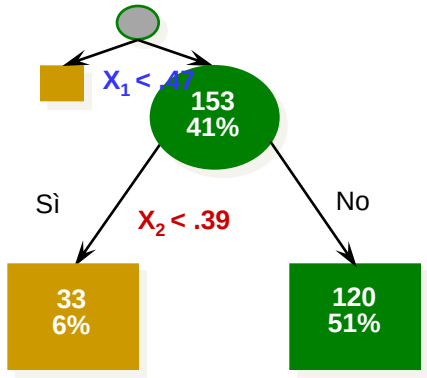
L'algoritmo di partizione effettua tutte le partizioni possibili dividendo i dati in due gruppi basandosi sui valori delle colonne di input e individua la **divisione migliore**. In questo caso è la disuguaglianza ***x1*<0,47**: 153 osservazioni da una parte (**41% verdi**) e 87 dall'altra (**0% verdi**).

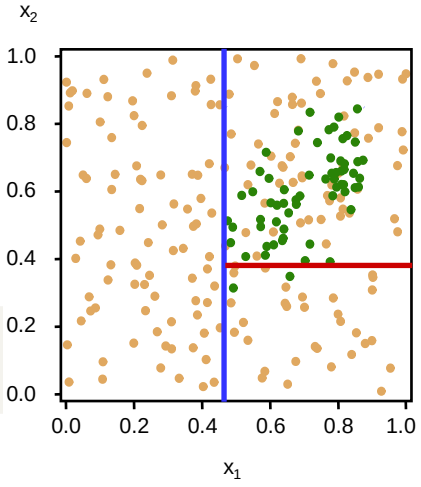
Il nodo figlio di sinistra, essendo puro (tutti gialli), diventa un nodo terminale (foglia).

Il nodo figlio di destra viene sottoposto a un'ulteriore suddivisione: vengono ricalcolate tutte le partizioni possibili delle 153 osservazioni nel nodo individuando come divisione migliore la disuguaglianza ***x2*<0,39**: 120 osservazioni da una parte (**51% verdi**) e 33 dall'altra (**6% verdi**).







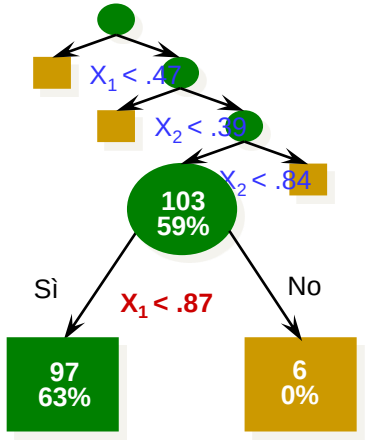


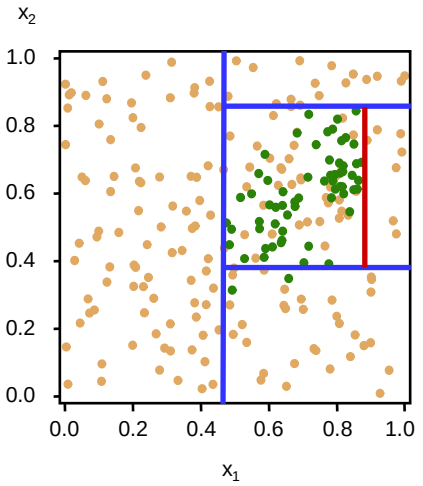
Alberi di Decisione – Esempio teorico (2/2)

Dopo la disuguaglianza **x2<0,84**, che porta a 103 osservazioni da una parte (**59% verdi**) e 17 dall'altra (**0% verdi**), la ramificazione dell'albero procede fino alla disuguaglianza **x1<0,87** che porta a 97 osservazioni da una parte (**63% verdi**) e 6 dall'altra (**0% verdi**) in quanto ulteriori suddivisioni non comportano miglioramenti significativi.

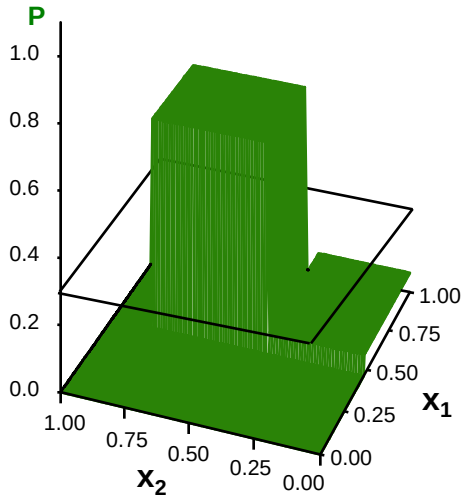
L'albero finale ha 5 foglie, di cui tre con probabilità di essere verde dello 0%, una con probabilità del 6% e una con probabilità del 63%.

Con un valore di threshold di 0,25 il confine della decisione è un rettangolo chiuso che contiene 97 osservazioni, il 63% delle quali è verde, con un'accuratezza del 84,2%, con 36 falsi positivi e 2 falsi negativi.





Valori Reali \ Valori Predetti	0	1	
0	141(80%)	36(20%)	177
1	2(3%)	61(97%)	63
	143	97	240



Alberi di Decisione – Modulo2_Esempio3

L'esempio qui proposto è in ambito finanziario e riguarda la **valutazione del rischio di credito**, il cui scopo è quello di creare delle regole per distinguere in modo efficace i clienti richiedenti un prestito tra i **clienti solventi e quelli insolventi** in base alle loro caratteristiche.

La tabella è composta da 1.000 clienti. Gli input sono le caratteristiche individuali dei soggetti censiti che possono essere socio-demografiche, finanziarie e personali, relative al prestito come l'ammontare del prestito, lo scopo della sottoscrizione, gli indicatori di ricchezza, ecc. Il target è la valutazione effettuata in passato della "bontà" del cliente da parte della banca (**1=solvente, 0=insolvente**).

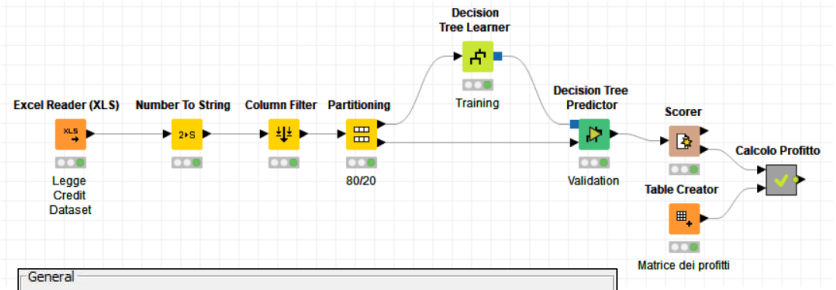
ID	Solvenza	Saldo_Con	Durata_in_	Storia	Scopo	Importo	Risparmi_1	Durata_Im	Tasso	Eta	Stato_civile
1	1	<0	6	Criticità/Crediti con altri	Televisore/HiFi	598	N/D	>7 Anni	4	67	M SINGLE
2	0	0<100	48	Crediti in essere ok	Televisore/HiFi	3.043	<50	1<=4 Anni	2	22	F SEP/DIV/SPO
3	1	N/D	12	Criticità/Crediti con altri	Istruzione	1.072	<50	4<=7 Anni	2	49	M SINGLE
4	1	<0	42	Crediti in essere ok	Mobili/Arredamento	4.030	<50	4<=7 Anni	2	45	M SINGLE
5	0	<0	24	Pagamenti in ritardo	Auto (nuova)	2.490	<50	1<=4 Anni	3	53	M SINGLE
6	1	N/D	36	Crediti in essere ok	Istruzione	4.630	N/D	1<=4 Anni	2	35	M SINGLE
7	1	N/D	24	Crediti in essere ok	Mobili/Arredamento	1.450	250-500	>7 Anni	3	53	M SINGLE
8	1	0<100	36	Crediti in essere ok	Auto (usata)	3.552	<50	1<=4 Anni	2	35	M SINGLE
9	1	N/D	12	Crediti in essere ok	Televisore/HiFi	1.564	>500	4<=7 Anni	2	61	M SEP/DIV
10	0	0<100	30	Criticità/Crediti con altri	Auto (nuova)	2.676	<50	Disoccupa	4	28	M SPO
11	0	0<100	12	Crediti in essere ok	Auto (nuova)	662	<50	<1 Anno	3	25	F SEP/DIV/SPO
12	0	<0	48	Crediti in essere ok	Attività commerciale	2.203	<50	<1 Anno	3	24	F SEP/DIV/SPO

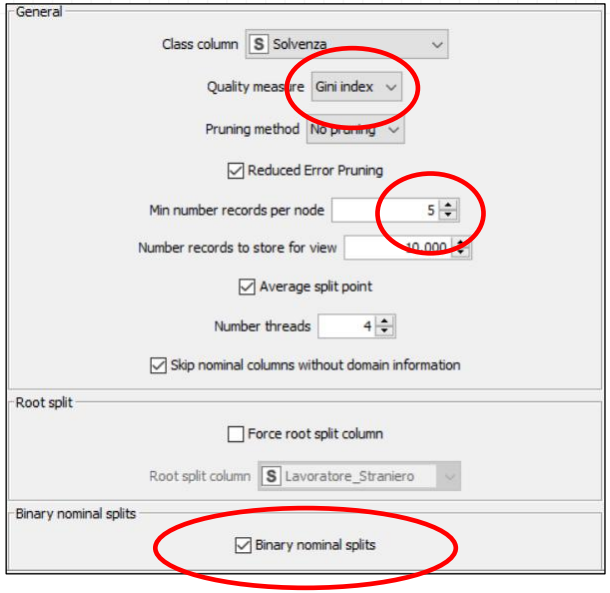
Alberi di Decisione – Modulo2_Esempio3

In questa tabella 700 clienti (il **70%**) sono stati classificati come **solventi** e 300 (il **30%**) come i**nsolventi**.

Il dati vengono suddivisi in due sottoinsiemi: l'**80%** delle osservazioni (**800**) viene utilizzato per la costruzione del modello (training set), mentre il restante **20%** delle osservazioni (**200**) viene utilizzato per il test (validation set).

Nel nodo Decisione Tree Learner è stato scelto il Gini index come criterio di suddivisione, 5 come valore di Min number records per node e si producono solo alberi binari; non viene adottato alcun metodo di pruning.





Alberi di Decisione – Modulo2_Esempio3

Dalla matrice di confusione ottenuta con il nodo Scorer (JavaScript) si vede la performance del classificatore: l'accuratezza è del 71%.

Il 77,1% (sensibilità) dei **veri solventi** sono stati predetti correttamente mentre l'80,6% (precisione) dei **predetti solventi** sono stati classificati correttamente.

Rows Number : 200	0 (Predicted)	1 (Predicted)	
0 (Actual)	34	26	56.67%
1 (Actual)	32	108	77.14%
	51.52%	80.60%	

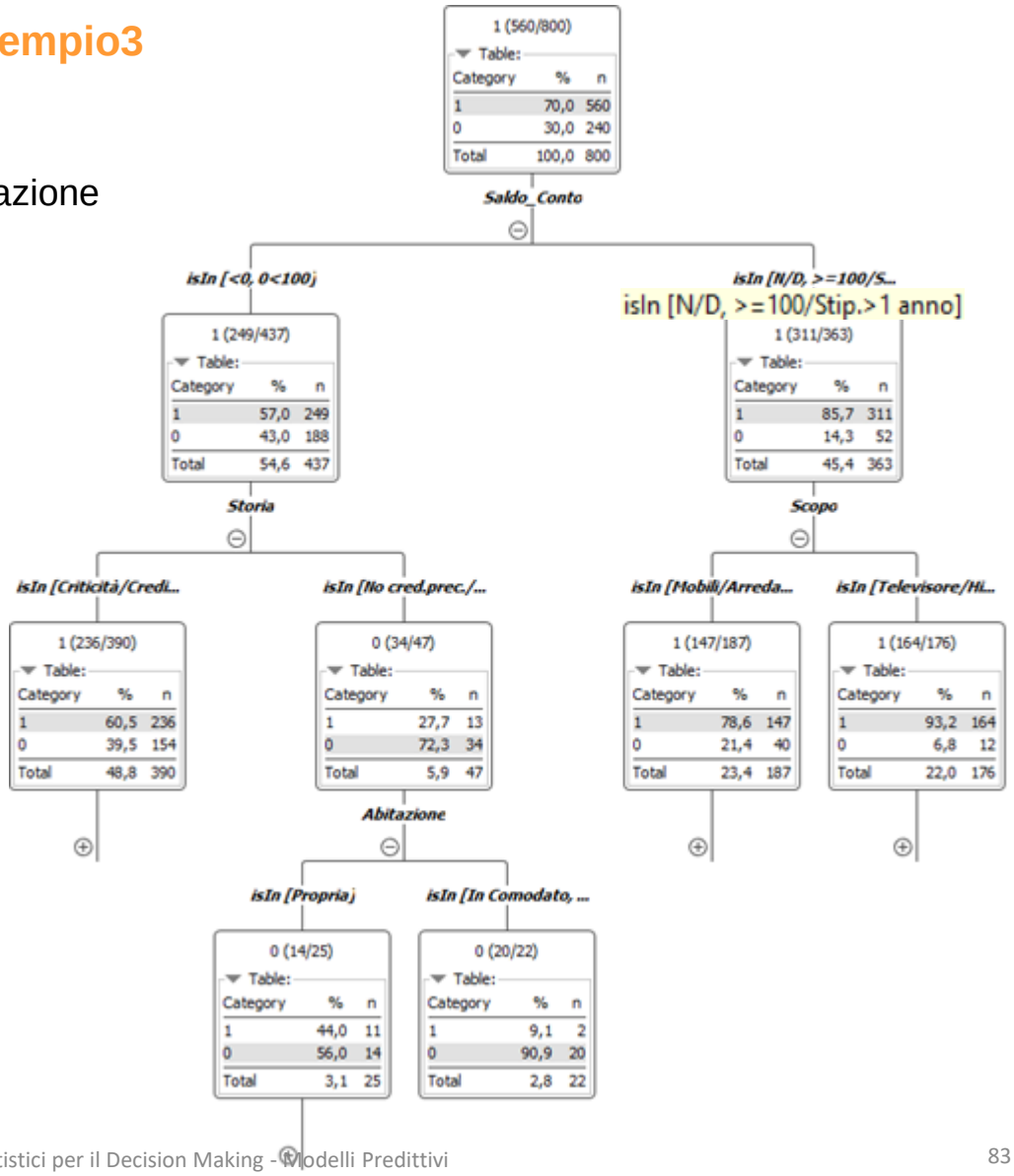
Class	True Positives	False Positives	True Negatives	False Negatives	Recall	Precision	Sensitivity	Specificity	F-measure
0	34	32	108	26	56.67%	51.52%	56.67%	77.14%	53.97%
1	108	26	34	32	77.14%	80.60%	77.14%	56.67%	78.83%

Alberi di Decisione – Modulo2_Esempio3

La rappresentazione grafica dell'albero di decisione consente una facile interpretazione delle regole:

il nodo più a destra è il risultato della **prima suddivisione** basata sulla colonna ***Saldo_Conto*** (*N/D, >=100, Stipendio da più di 1 anno*) e contiene 363 osservazioni, per la maggior parte associate a clienti buoni e viene **suddiviso ulteriormente** in base alla colonna ***Scopo*** producendo altri rami.

Il nodo più a sinistra contiene 437 osservazioni, **Saldo_conto** *(<0, <100*) e viene **suddiviso ulteriormente** in base alla colonna ***Storia***, producendo altri rami.



Alberi di Decisione – Modulo2_Esempio3

A puro titolo di esempio, per le **valutazioni economiche**, è stata creata una **matrice di costi/profitti** così strutturata:

		Previsti 0	Previsti 1
Osservati	0	0	-0,25
	1	0	0,10

Si immagina quindi un **profitto del 10% per ogni cliente** predetto solvente quando lo è, e una **perdita del 25% per ogni cliente** predetto solvente quando non lo è.

Con questi dati, **non utilizzando alcun modello**, ipotizzando un prestito medio a persona di 5.000 Euro, si avrebbe un **profitto** per cliente di -0,005 (0,70*0,10+0,30*-0,25) con un importo di -25 Euro (-0,005*5.000) e, **sul totale dei 1.000 clienti**, **di -25.000 Euro** (-25*1.000).

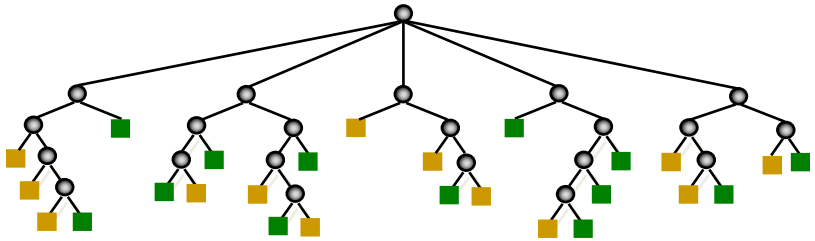
Il profitto per cliente sarebbe ora, **in base alle decisioni del modello**, di 0,0215 (108/200*0,10+26/200*-0,25) che, moltiplicato per l'importo, sarebbe di 107,5 Euro (0,0257* 5.000) e, su un ipotetico **insieme di 1.000 nuovi clienti richiedenti il prestito, di 107.500 Euro** (107,5*1.000).

Rows Number : 200	0 (Predicted)	1 (Predicted)
0 (Actual)	34	26
1 (Actual)	32	108

Il vantaggio dell'utilizzo di un modello predittivo rispetto al non utilizzo in termini economici è evidente (**107.500 Euro contro i -25.000**).

Foreste casuali

L'idea fondamentale che sta dietro una *Random Forest* o **Foresta Casuale** (FC) è quella di combinare molti alberi di decisione in un unico modello.



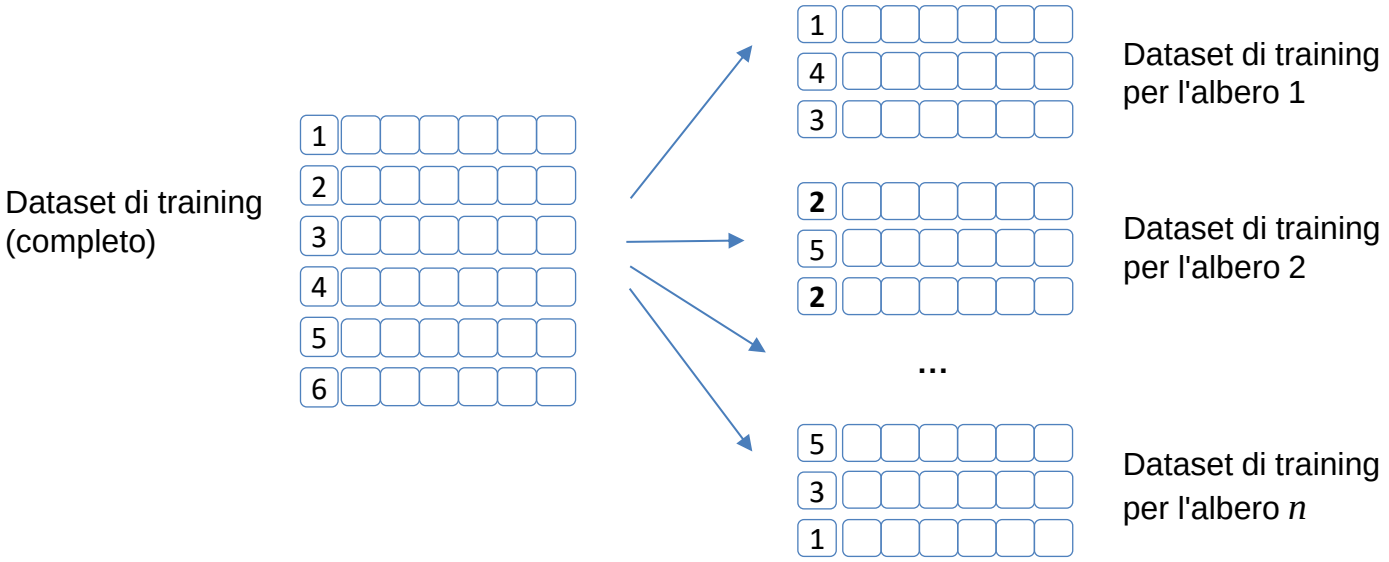
Un albero di decisione può fare predizioni non accurate, in quanto soggetto all'*overfitting*. Come si è visto precedentemente, ci sono diverse soluzioni per questo problema.

In ogni caso, però, l'albero di decisione darà comunque importanza a un **particolare insieme di colonne** di input per cui non vi è certezza di ottenere il massimo come capacità di generalizzare.

Utilizzando più alberi di decisione, addestrando ciascuno con un **insieme diverso di osservazioni** prese casualmente e con un **numero limitato di colonne** di input, anch'esse scelte in modo casuale, si possono ottenere predizioni più accurate in quanto il risultato finale altro non è che quello restituito dalla maggioranza degli alberi.

Foreste casuali – I sottoinsiemi di osservazioni

Ciascun albero di decisione verrà addestrato con un diverso **campione** casuale del **dataset di training**. Il campionamento viene effettuato **con reinserimento** (tecnica conosciuta come *bootstrap aggregation o «bagging»*), il che significa che alcuni campioni possono essere usati più volte all'interno dello stesso albero (in questo esempio il n.2).

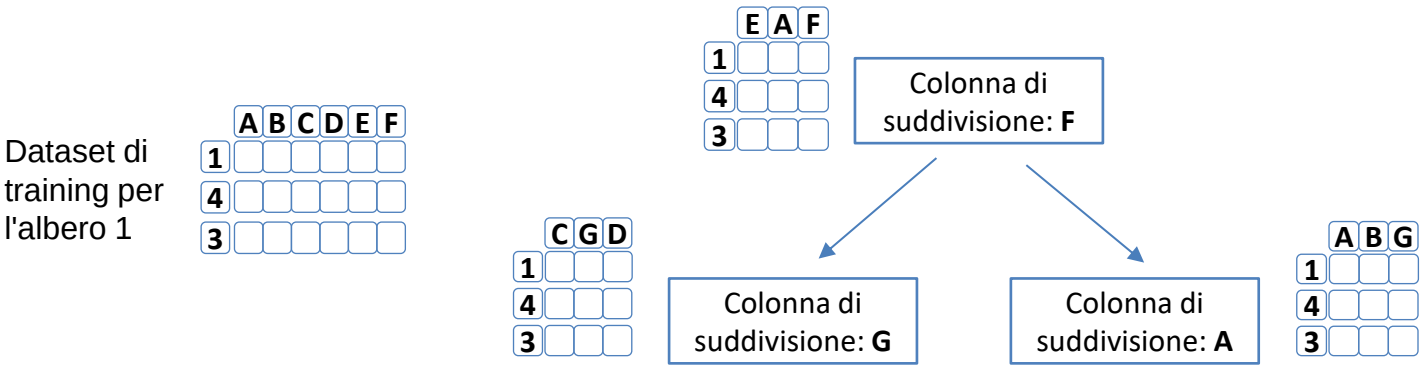


Foreste casuali – I sottoinsiemi di colonne

Nell'albero di decisione non si possono avere due colonne di input che si comportano in modo indipendente perché ogni colonna interagisce con le altre che stanno a livelli più alti nell'albero mascherandone così l'importanza.

Nelle FC invece **ogni nodo di ogni albero** di decisione utilizza un **diverso sottoinsieme delle colonne** di input evitando così di avere alberi altamente correlati per il fatto di avere colonne molto predittive.

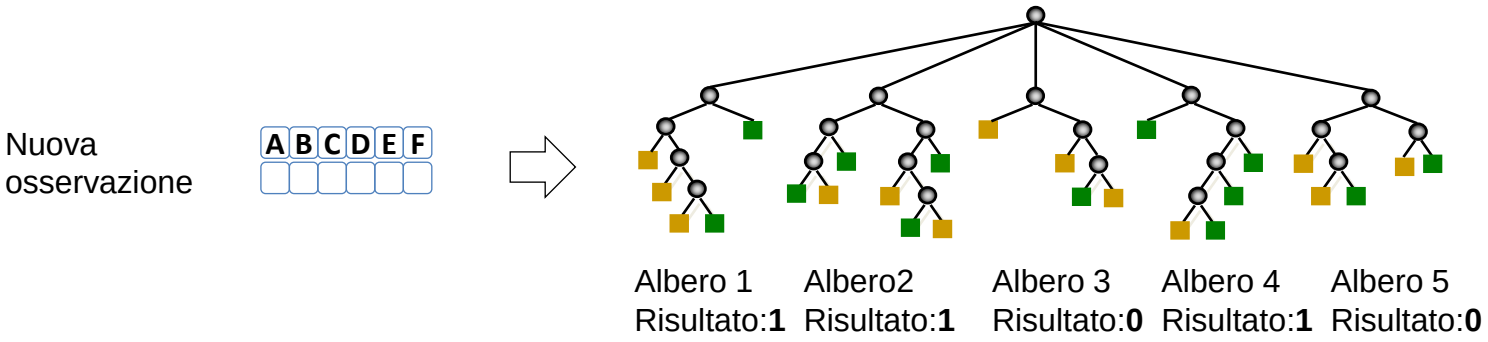
In questo modo le FC sono in grado di cogliere effetti di **interazione tra le colonne** di input e quindi anche l'**importanza** delle stesse.



Con questo approccio, nessun albero può vedere **tutti i dati del training** set evitando così il problema dell'*overfitting*.

Foreste casuali – Predizione

La predizione di una FC su ogni nuova osservazione sarà quella che otterrà la maggioranza[1] ottenuta da tutti gli alberi dell'insieme.



In questo esempio si hanno 3 predizioni con il risultato "1" e 2 con il risultato" 0": viene scelta come predizione il valore "1".

[1] Nel caso in cui la colonna target sia di tipo continuo, sarà la media.

Foreste casuali – Importanza delle variabili

Uno svantaggio dell'utilizzo delle FC è che **non sono interpretabili** come un singolo albero di decisione.

Si può però determinare l' "**importanza**" ai fini predittivi di ogni colonna di input cercando quella che ha contribuito di più alle suddivisioni ad ogni livello di suddivisione.

L'importanza può essere ulteriormente ricalcolata empiricamente dividendo – per ogni livello di suddivisione – il numero di volte che ha contribuito a una suddivisione diviso il numero di volte che è stata candidata.

Ad esempio calcolando l'importanza per i soli primi 2 livelli:

Colonna	#Contr. Liv. 0	#Contr. Liv. 1	…	#Candid. Liv. 0	#Candid. Liv. 1	…
A	*20*	*35*		*26*	*48*	
B	*20*	*24*		*20*	*31*	

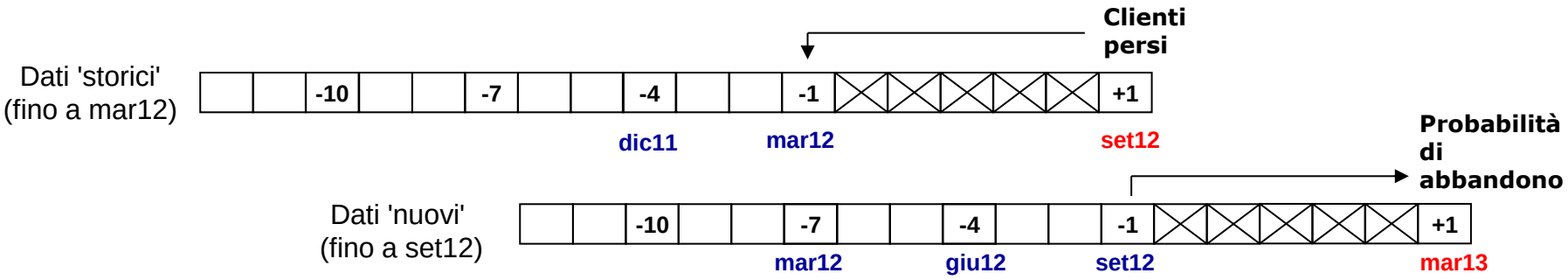
Importanza Totale *B: (20/20) + (24/31) = 1,77*
Importanza Totale *A: (20/26) + (35/48) = 1,50*

Foreste Casuali – Modulo2_Esempio4

In questo esempio si vuole prevedere i clienti di una banca che abbandoneranno nei prossimi mesi.

- ✓ Si utilizzano i dati storici per costruire un modello predittivo
- ✓ Le regole fornite dal modello verranno applicate ai dati attuali per predire la probabilità di abbandono di ogni cliente in base al periodo di latenza[1] prescelto
- ✓ I risultati verranno incrociati con valutazioni di profittabilità dei clienti per definire le priorità in base alle quali saranno attivati i contatti delle iniziative commerciali

Il **mese "attuale", in base ai dati disponibili, è settembre 2012**, si vuole prevedere chi verosimilmente **abbandonerà a marzo 2013.**



[1] Il periodo di latenza è il tempo che intercorre tra il mese relativo agli ultimi dati di riferimento e il mese previsto di abbandono; questo periodo tiene conto sia del tempo necessario per la disponibilità dei nuovi dati e il tempo necessario per contattare il cliente.

Foreste Casuali – Modulo2_Esempio4

I **dati storici coprono un periodo di 18 mesi** e riguardano i clienti del segmento "***Upper Mass***" (disponibilità 50-100k€), con anzianità di rapporto > 16 mesi, senza informazioni negative, esclusi i dipendenti.

Le colonne originarie sono del tipo:

Anagrafiche
- *Age*
- *Tenure*
- …

Comportamentali
- *Transactions and Amounts for Withdrawal and Purchases (Credit/Debit cards)*
- *Average Balances*
- *Transaction and Amounts for Regular Income*
- *Transactions and Amounts for Standing Order*
- *Transactions and Amounts for Direct Debits*
- *Transactions and Amounts for Outgoing and Incoming Bank Transfers*
- *Transactions and Amounts for Withdrawals and Deposits*
- *Transactions and Amounts for Withdrawals and Deposits at the Branch Cash*
- …

Foreste Casuali – Modulo2_Esempio4

Le colonne di calcolo

Per ciascuna colonna di tipo comportamentale sono state calcolate le **variazioni percentuali** tra l'ultimo mese (-1) e i rispettivi 3, 6, 9 mesi prima (-4, -7, -10):

$$Nome\_Colonna\_\mathbf{P3} = \frac{(Nome\_Colonna\_\mathbf{1} - Nome\_Colonna\_\mathbf{4})}{ass\ (Nome\_Colonna\_\mathbf{4})}$$

$$Nome\_Colonna\_\mathbf{P6} = \frac{(Nome\_Colonna\_\mathbf{1} - Nome\_Colonna\_\mathbf{7})}{ass\ (Nome\_Colonna\_\mathbf{7})}$$

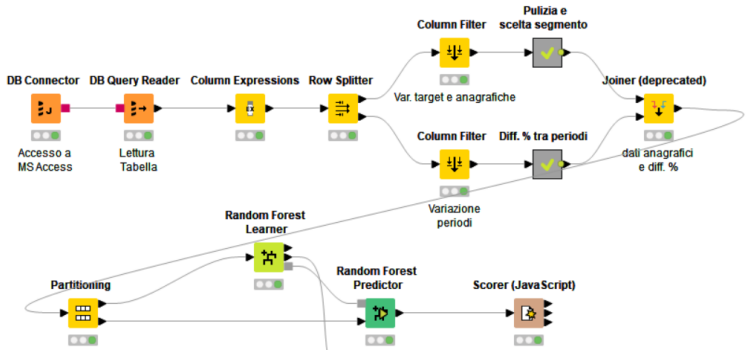
$$Nome\_Colonna\_\mathbf{P9} = \frac{(Nome\_Colonna\_\mathbf{1} - Nome\_Colonna\_\mathbf{10})}{ass\ (Nome\_Colonna\_\mathbf{10})}$$

ed è stato calcolato un indice di profittabilità, il ***Customer_Value_Index***.

Foreste Casuali – Modulo2_Esempio4

I dati di questa tabella di **Microsoft Access** di circa 430.000 osservazioni riguardano **13.831** clienti di una banca che comprendeva anche quei 1.103 (il **7,97%**) che avevano abbandonato a settembre 2012.

I dati sono stati suddivisi in due sottoinsiemi: il 90% delle osservazioni viene utilizzato per la costruzione del modello (training set), mentre il restante 10% viene utilizzato per la valutazione (validation set).

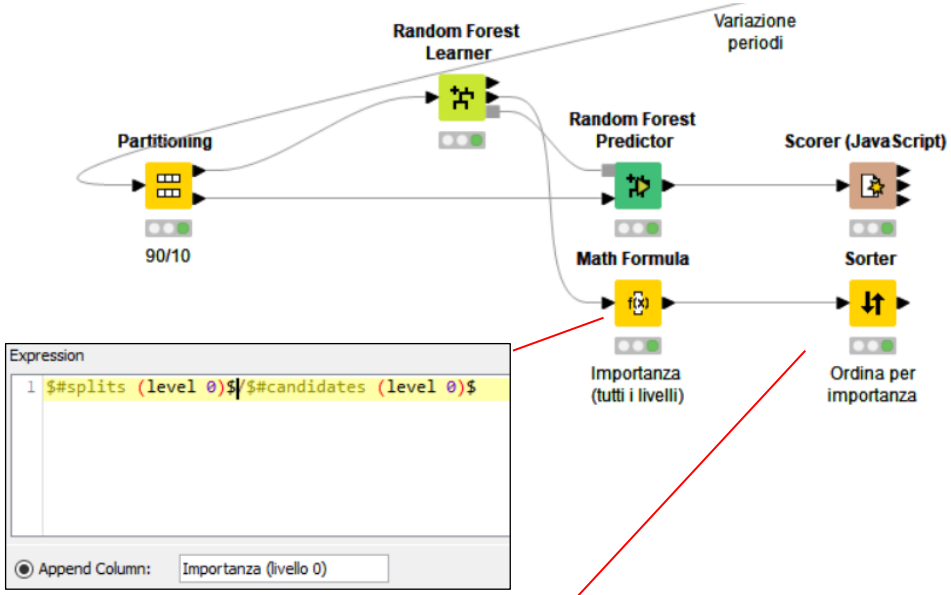


La strutturazione dei dati viene effettuata nei 2 metadati.

Per l'apprendimento è stato usato il nodo ***Random Forest Learner*** scegliendo Gini come criterio di suddivisione e 500 il numero di alberi di decisione e 2 come numero minimo di record che ci possono essere in un nodo foglia.

Foreste Casuali – Modulo2_Esempio4

Dal prospetto ottenuto si nota che *TOT_ASSETS_P9* (la var. % delle disponibilità totali rispetto a 9 mesi prima), *CNT_REG_INCOME_P3* (la var. % del numero di operazioni di accredito regolare rispetto a 3 mesi prima), *TOT_CNT_P3* (la var. % del numero di operazioni totali rispetto al trimestre precedente) e *AMT_REG_INCOME_P3* (la var. % dell'importo di accredito regolare rispetto a 3 mesi prima) hanno un valore di importanza maggiore delle altre colonne.



Row ID	I #splits (level 0)	I #candidates (level 0)	D Importanza (livello 0)
TOT_ASSETS_P9	50	51	0.98
CNT_REG_INCOME_P3	51	56	0.911
TOT_CNT_P3	34	44	0.773
AMT_REG_INCOME_P3	42	56	0.75

Foreste Casuali – Modulo2_Esempio4

Come si evince dalla matrice di confusione, l'accuratezza è del 93,8%;

la sensibilità (la percentuale di eventi trovati correttamente dal modello sul totale degli eventi) è del 32,7;

la precisione (la percentuale di eventi predetti correttamente dal modello sul totale di eventi predetti) è del 75%.

Il modello finale è stato poi applicato ai dati di marzo 2013; la probabilità di abbandono e l'indice di valore del cliente sono stati divisi in 3 classi, ottenendo così una **matrice Rischio/Valore** per definire su quali clienti intervenire in base a 3 **segmenti di priorità**.

Confusion Matrix

Rows Number : 1384	0 (Predicted)	1 (Predicted)	
0 (Actual)	1262	12	99.06%
1 (Actual)	74	36	32.73%
	94.46%	75.00%	

Class Statistics

Class	True Positives	False Positives	True Negatives	False Negatives	Precision	Sensitivity	Specificity	F-measure
0	1262	74	36	12	94.46%	99.06%	32.73%	96.70%
1	36	12	1262	74	75.00%	32.73%	99.06%	45.57%

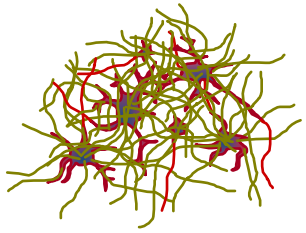
Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
93.79%	6.21%	0.428	1298	86

Probability of leaving

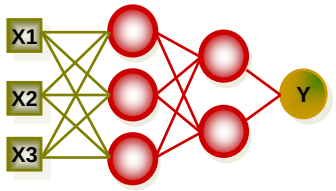
Customer value	LOW	MEDIUM	HIGH			
HIGH	3.673	1.741	**3.620**	**Segment A**	3.620	24%
MEDIUM	741	731	1.738	**Segment B**	4.210	28%
LOW	383	286	1.952	**Segment C**	7.035	47%
					14.865	100%

Reti Neurali – Introduzione



Le *Neural Networks* o **Reti Neurali** (RN) sono vasta classe di **modelli matematici** che provano a riprodurre i principi di funzionamento del cervello animale e dei suoi neuroni biologici.

Una RN è tipicamente organizzata in una struttura costituita da **strati e nodi**.



Tutti i nodi di uno strato sono collegati a quelli dello strato successivo attraverso **connessioni** alle quali è associato un valore (**peso**) che aumenta o diminuisce la potenza del "segnale" in ingresso al nodo.

Ogni nodo elabora i segnali ricevuti in ingresso e trasmette il risultato ai nodi successivi fino al nodo d'uscita finale.

Le RN vengono addestrate a trovare il **minimo di una funzione** complessa (che rappresenta l'**errore** tra i valori di uscita reali e quelli calcolati) attraverso un **processo iterativo** che **modifica**, a seconda degli esempi (i dati forniti in ingresso), i **pesi** delle connessioni.

La RN "**impara**" così a riconoscere le **relazioni esistenti nei dati attraverso esempi**, proprio come avviene l'apprendimento di un bambino.

Reti Neurali – Introduzione

Grazie la loro capacità di scoprire **relazioni non lineari** nei dati, possono risolvere problemi complessi di **regressione e classificazione**.

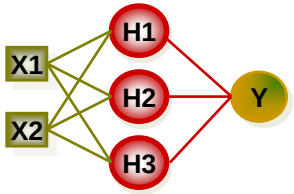
Oltre alle applicazioni conosciute in ambito business, vengono spesso utilizzate nel campo della programmazione delle **intelligenze artificiali** per:

- Il riconoscimento e la classificazione di immagini (scrittura o fotografie)
- La trascrizione del parlato
- L'analisi semantica dei testi
- Previsioni finanziarie

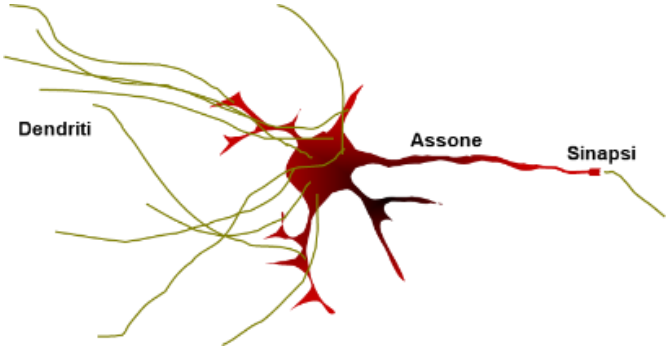
Reti Neurali – Struttura

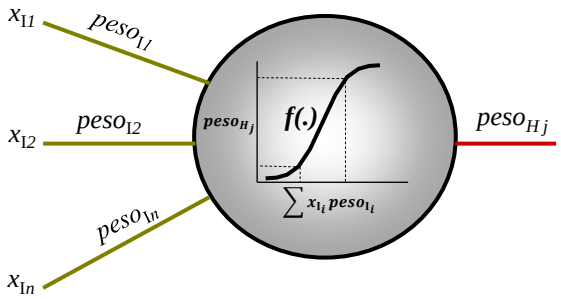
La struttura di una RN di tipo **Feedforward Multilayer Perceptron (MLP)** è composta da:

- uno strato di input contenente i nodi di input (le colonne di input, $x_1, x_2, \ldots x_n$)
- uno più strati "nascosti" contenenti i nodi nascosti ("*hidden nodes*", $h_1, h_2, \ldots$)
- uno strato contenente il nodo di output (la colonna target, y)



Ogni nodo nascosto o di output è collegato ai nodi precedenti attraverso delle connessioni (per analogia, i dendriti) dalle quali riceve un segnale caratterizzato dal **valore presente in input** moltiplicato per il **peso assegnato alla connessione**. Questi **segnali vengono sommati** e il risultato viene sottoposto a una **funzione di attivazione** (simulando così la sinapsi), che trasferisce il segnale alla connessione di uscita (l'assone). I pesi rappresentano così la "**conoscenza**" della RN.



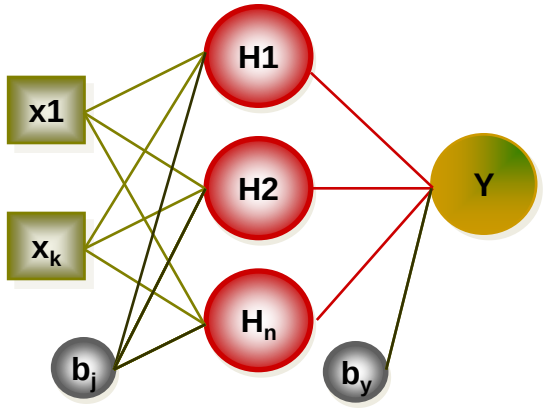


Reti Neurali – Architettura (Multilayer Perceptron)

Matematicamente, una RN di tipo MLP con uno strato nascosto, si può rappresentare come una combinazione lineare degli input:

$$f_0^{-1}(Y) = b_O + w_{O1}H_1 + w_{O2}H_2 + \ldots + w_{Oh}H_h \quad \text{dove} \quad H_i = f_h\,(b_I + w_{i1}x_1 + w_{i2}x_2 + \ldots + w_{ik}x_k)$$

k	è il numero dei nodi di input
h	è il numero dei neuroni o nodi nascosti (*hidden*)
b_j	sono gli input fittizi[1] (*bias*)
w_{ij}	sono i pesi (*weight)*
$f_h(.)$	è la funzione di attivazione dei nodi nascosti
$f_o^{-1}(.)$	è la funzione di attivazione inversa del nodo di output

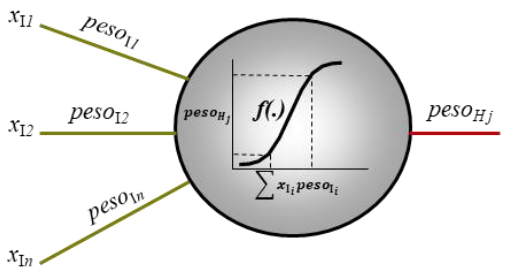


[1] Un nodo nascosto è caratterizzato dalle connessioni di input, dalla funzione di attivazione e da un'ulteriore connessione, che funge da soglia di attivazione, che si indica con il nome di ***bias***. Questa ha in input un valore costante, uguale a 1, e un peso il cui effetto è di traslare la soglia di attivazione. Il *bias* corrisponde alla soglia di potenziale elettrico del neurone biologico e all'intercetta di un modello di regressione lineare.

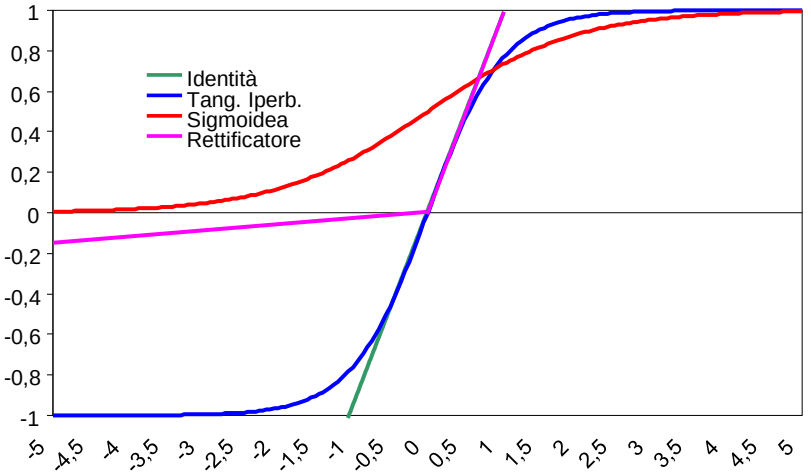
Reti Neurali – Funzione di attivazione

La funzione di attivazione determina il valore di output che emette un neurone (nodo nascosto) dato un insieme di valori di input.

Una **funzione di attivazione non lineare** dà alla RN la capacità cogliere relazioni nei dati di natura più complessa e verosimilmente poco lineare.



Funzione	Simbolo	Range	Formula
Identità (o lineare)	Identity	-∞, +∞	x
Sigmoidea	Sigmoid	0,1	$1/(1+e^{x})$
Tangente Iperbolica	TanH	-1,+1	$(e^{x}-e^{-x})/(e^{x}+e^{-x})$
Rettificatore (o rampa)	ReLu	0,x	$Max(0,x)$
Rettificatore (o rampa)	LeakyReLu	0,01x,x	$Max(0,01,x)$



Reti Neurali – La funzione di errore

Il processo di apprendimento modifica i pesi in modo da **minimizzare una funzione di errore**.

La funzione di errore è detta anche funzione di perdita[1] (*loss function*) o di costo[2] (*cost function*) rappresenta la **differenza tra il valore di uscita calcolato** dalla RN **e il valore atteso** presente nel dataset di training.

Ci sono diverse funzioni di costo, tra cui:

- Cross-Entropy per target binari (0,1)

$$E(y,\hat{y}) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$$

- Mean Squared Error per target continui

$$E(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

Uno dei metodi più utilizzati per minimizzare la funzione di errore è la **discesa del gradiente** (*gradient descent*), poiché essa segue la **direzione di massima pendenza** di una funzione a più variabili, permettendo di trovare un minimo locale (non è necessariamente il minimo globale della funzione di costo) attraverso un **processo iterativo** che modifica **i pesi** delle connessioni.

[1] Su singolo esempio.
[2] Su tutto il training set, è una media della funzione di perdita.

Reti Neurali – Discesa del gradiente e backpropagation

Il **gradiente è un vettore** dove ciascuna delle sue componenti è una **derivata parziale** rispetto a una specifica variabile. Nel caso di una funzione a due variabili sarà:

$$\nabla f(x_1, x_2) = \begin{bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \end{bmatrix}$$

Nel caso della **funzione di costo** con pesi e bias

$$\nabla E(y, \hat{y}) = \partial E/\partial w$$

L'aggiornamento dei pesi sarà dato dall'equazione $\boldsymbol{w^+ = w - \alpha \frac{\partial E}{\partial w}}$ che espansa appare così:

$$\begin{bmatrix} w_1^+ \\ w_2^+ \\ \dots \\ w_n^+ \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_n \end{bmatrix} - \boldsymbol{\alpha} \begin{bmatrix} \partial E/\partial w_1 \\ \partial E/\partial w_2 \\ \dots \\ \partial E/\partial w_n \end{bmatrix}$$

dove $\boldsymbol{w^+}$ sono i nuovi pesi, $\boldsymbol{\partial E/\partial w}$ è il gradiente e $\boldsymbol{\alpha}$ è il tasso di apprendimento (*learning rate*); il segno meno indica la direzione in discesa.

Durante ogni iterazione, calcolato il gradiente, i pesi e i bias vengono aggiornati, tipicamente in modo retroattivo, utilizzando l'algoritmo di **backpropagation**.

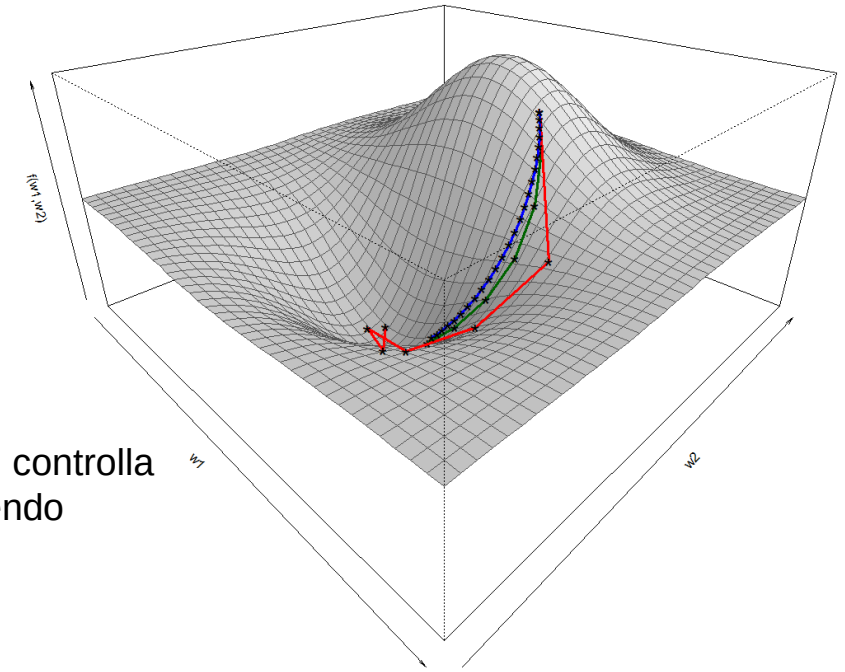
Questo processo continua finché non si raggiunge il numero massimo di iterazioni o finché l'errore non scende sotto una soglia prefissata.

Reti Neurali – Tasso di apprendimento e momento

Il tasso di apprendimento (***learning rate)*** α equivale alla lunghezza del passo affinché la funzione di costo raggiunga un minimo. In pratica l'aggiornamento dei pesi viene scalato da questo parametro.

- se troppo piccolo, convergenza lenta (linea blu)
- se troppo grande, scarsa capacità di convergenza (linea rossa)

Il suo valore viene di solito fissato a **0,01.**



Sono disponibili diverse estensioni dell'algoritmo *gradient descent* che rendono il *learning rate* adattivo nel senso che cambia il suo comportamento nel momento in cui viene eseguito tra cui:

Il momento (***momentum***) $\boldsymbol{\beta}$ è un altro parametro che controlla l'importanza delle iterazioni precedenti, $\boldsymbol{w}^{-}$, aggiungendo "inerzia" al processo di aggiornamento.

$$\boldsymbol{w}^{+} = \boldsymbol{w} - \boldsymbol{\alpha}\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{w}} + \boldsymbol{\beta}\,(\mathbf{w} - \boldsymbol{w}^{-})$$

Un valore empirico per tale parametro è solitamente fissato a **0,9.**

Reti Neurali – Apprendimento Batch

Un aspetto importante da tener presente nel processo delle RN è la scelta del **numero di osservazioni del training set** più appropriata per l'apprendimento dovuta alla dimensione dei dati e dei limiti di memoria del computer.

Se il dataset di training non può essere elaborato per intero, si deve dividere in **lotti** (batch). La dimensione del lotto (**batch-size**[1]) è il numero di righe presenti in ogni batch.

Si definisce **epoca**(*epoch*) l'aggiornamento dei pesi sull'**intero dataset di training**. Se ogni **iterazione** porta a un aggiornamento dei pesi, per completare un'epoca sono necessari un numero di iterazioni pari al numero di batch.

N. osservazioni Training set	Batch-size	N. Iterazioni (per l'intero Training set)	Epoche	N. Iterazioni (per completare l'addestramento)
2.000	**1**	2.000	**1**	2.000
2.000	**50**	40	**1**	40
2.000	**1.000**	2	**1**	2
2.000	**2.000**	1	**1**	1
2.000	**1**	2.000	**10**	20.000
2.000	**50**	40	**10**	400
2.000	**1.000**	2	**10**	20
2.000	**2.000**	1	**10**	10

[1] Un batch size troppo piccolo introduce rumore, troppo grande si incorre in problemi di memoria. Sono usati nella pratica multipli della potenza di 2 (p.e. **32, 64, 128**)

Reti Neurali – Altri algoritmi di ottimizzazione

Tra gli algoritmi di ottimizzazione della funzione di costo oltre al **Gradient Descent** si trovano:

- **RProp** (Resilient BackPropagation)
- **Stochastic Gradient Descent** (SGD)

  Sostituisce il Gradient Descent (calcolato dall'intero dataset di training) con una sua stima ottenuta da un **sottoinsieme di dati** (**batch**) selezionato casualmente.

  Soprattutto alcune sue varianti:

  - RMSProp (Root Mean Square Propagation)
  - Adam[1] (Adaptive moment estimation)

- **Conjugate Gradient** (CG)

oltre a questi ne esistono di più efficaci, ma che fanno un *uso intensivo della memoria*:

- Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS)

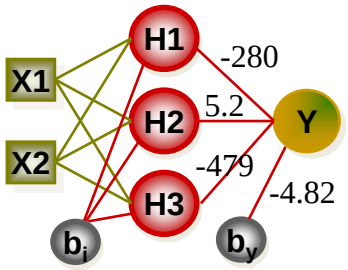
[1] Computazionalmente efficiente, con bassi requisiti di memoria.

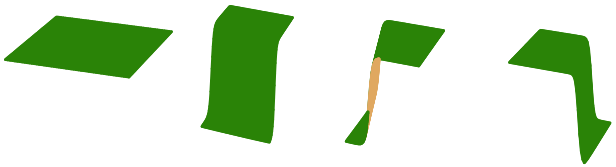
Reti Neurali – Esempio teorico

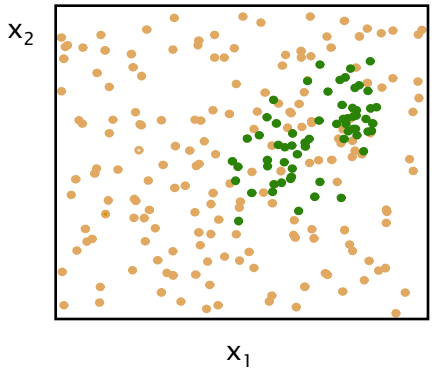
La funzione di output di **una RN con uno strato nascosto** dipende da un gran numero di parametri non lineari. Questo la rende estremamente versatile in moltissimi contesti applicativi in quanto **riesce ad approssimare qualsiasi funzione continua** (Cybenko, 1989).

Il risultato di una RN, come questa sotto riportata è, quindi, se usata la funzione di attivazione sigmoidea, una combinazione lineare di superfici sigmoidee.

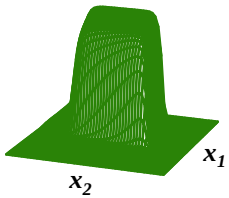
$$\text{logit} = \ln\left(\frac{p}{1-p}\right) = -4.82 - 280\,H_1 + 5.2\,H_2 - 479\,H_3$$

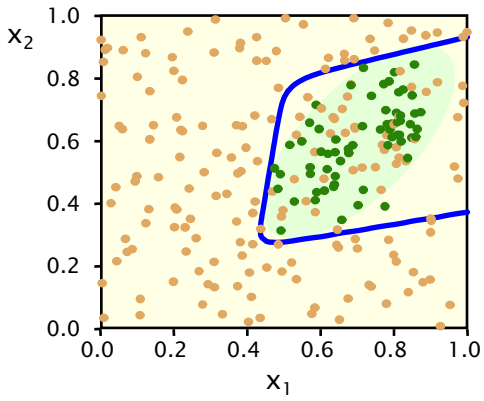






$$\boldsymbol{p} = f_0^{-1}(logit) = \frac{1}{(1 + e^{-(logit)})}$$



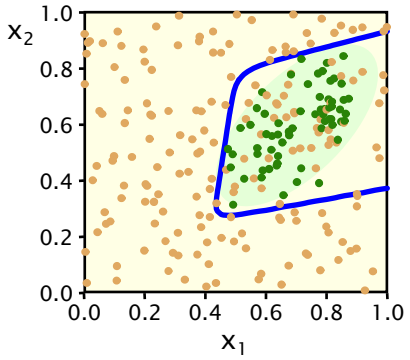


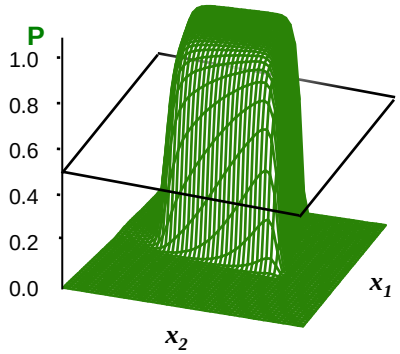
Reti Neurali – Esempio teorico

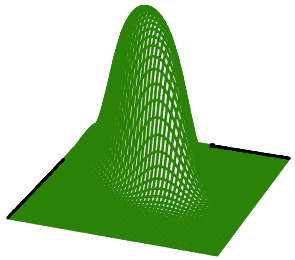
Le RN sono molto flessibili e ricreare anche forme molto complesse.

In questo caso la RN si è avvicinata molto alla forma generale della probabilità reale (anche se più squadrata).

Con un valore di threshold di 0,5 il confine della decisione è una superfice contenente 109 righe, il 100% delle quali è verde, con una accuratezza del 80,8%, classificando 46 falsi positivi e 0 falsi negativi.





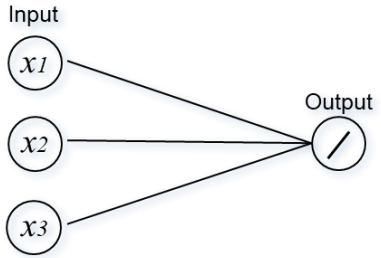


		Valori Predetti 0	Valori Predetti 1	
Valori Reali	0	131 (74%)	46 (26%)	177
	1	0 (0%)	63 (100%)	63
		131	109	240

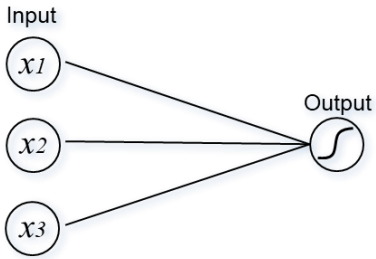
Reti Neurali – Relazioni con i modelli statistici

Funzione di attivazione: / Lineare; ∫ Sigmoidea; ⌐ Gradino

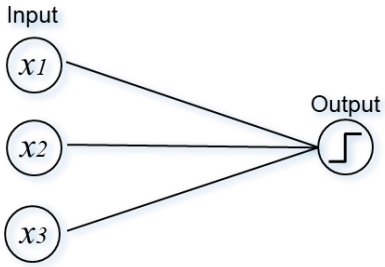
Regressione lineare semplice



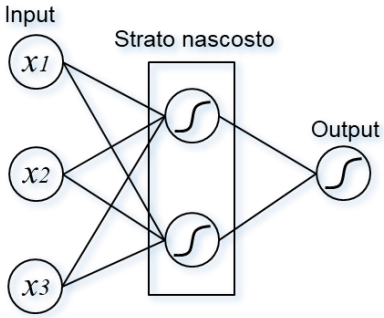
Regressione lineare multipla



Regressione logistica



Funzione discriminante lineare



Regressione non lineare multipla

Reti Neurali – Calcolo dei nodi nascosti

Una RN con un numero di unità nascoste insufficiente si ottengono errori elevati sia sul training set sia sul validation set; un numero eccessivo può portare al fenomeno dell'*overfitting* con pochi errori sul training set ma grandi sul validation set.

Un possibile calcolo iniziale, per una RN con una colonna target di tipo binario, può essere dato da:

$$h \geq \frac{min(n_1, n_0) - \alpha}{\alpha\ (I + O + 1)}$$

dove n_1 e n_0 sono rispettivamente la classe meno e più rappresentata, I è il numero di unità di input, O è il numero di unità di output e α è un fattore di scala che può andare da 2 a 10[1].

Ad esempio, avendo 2.500 righe per il training (delle quali il 10% ha il valore 1 e il 90% ha il valore 0), 3 nodi di input e 1 nodo di output dovremo avere almeno 5 unità nascoste (α=10):

$h \geq (250 - 10) / (10*(3+1+1)) \geq 240/50 \geq 4{,}8$

Ovviamente la ***miglior regola è quella selezionare sempre la rete che funziona meglio sul validation set***. ▷

[1] Harrell consiglia un minimo di 10 righe per ciascun peso (Harrell et al,1996, *Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors*, "Statistics in Medicine",15, pp. 361-387).

Reti Neurali – Modulo2_Esempio5

In questo esempio si vuole prevedere la probabilità di cancellazione dell'account email da parte dei clienti di una grande azienda.

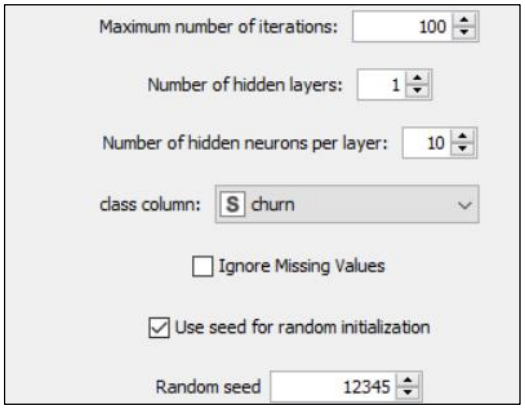
I dati riguardano l'attività di posta elettronica dei clienti tra cui *giorni_attivo, send_outs_ultimi_3mesi, send_outs, open_u, click_u, open_u_ultimi_3mesi, Open_Rate, Open_Rate_ultimi_3mesi, …*

Il tasso di abbandono è stato del **6,2%**. I dati sono stati suddivisi in due sottoinsiemi: il 90% delle righe viene utilizzato per la costruzione del modello (training set), mentre il restante 10% viene utilizzato per la valutazione (validation set).

Per l'apprendimento è stato usato il nodo **RProp MLP Learner** (una RN feedforward che usa l'algoritmo **R**esilient Back**Prop**agation).

In questo si possono impostare:

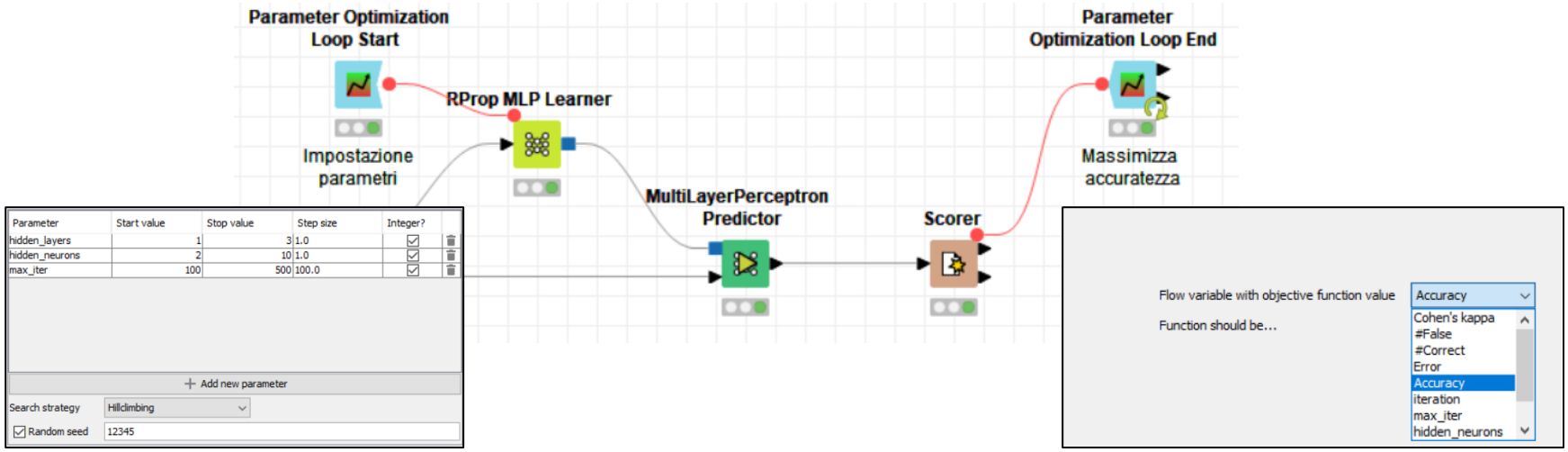
- iterazioni (un'iterazione passa attraverso tutto il dataset)
- strati
- unità nascoste all'interno di uno strato



Reti Neurali – Modulo2_Esempio5

Per il dimensionamento ottimale degli strati e delle unità nascoste si è usato un metodo empirico: vengono impostati il numero di iterazioni, il numero di strati e il numero di unità nascoste, alla fine si sceglie la combinazione che **massimizza l'accuratezza**.

Per questo si sono usati i nodi **Parameter Optimization Start** e **Parameter Optimization Loop End**.



Reti Neurali – Modulo2_Esempio5

L'uscita del nodo Parameter Optimization Loop End mostra che la combinazione ideale si ha con 2 strati nascosti con 9 neuroni ciascuno con un numero massimo di 500 iterazioni (epoche).

Il modello ottenuto ha un'accuratezza del 97%, un 72% di sensibilità e un 81% di precisione.

I hidden_layers	I hidden_neurons	I max_iter	D Objective value
2	9	433	0.75
1	8	433	0.643
1	10	433	0.712
1	9	333	0.674
3	9	433	0.659
2	8	433	0.719
2	10	433	0.702
2	9	333	0.716

Confusion Matrix

Rows Number : 1812	0 (Predicted)	1 (Predicted)	
0 (Actual)	1681	19	98.88%
1 (Actual)	31	81	72.32%
	98.19%	81.00%	

Class Statistics

Class	True Positives	False Positives	True Negatives	False Negatives	Precision	Sensitivity	Specificity	F-measure
0	1681	31	81	19	98.19%	98.88%	72.32%	98.53%
1	81	19	1681	31	81.00%	72.32%	98.88%	76.42%

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
97.24%	2.76%	0.750	1762	50

Deep Learning

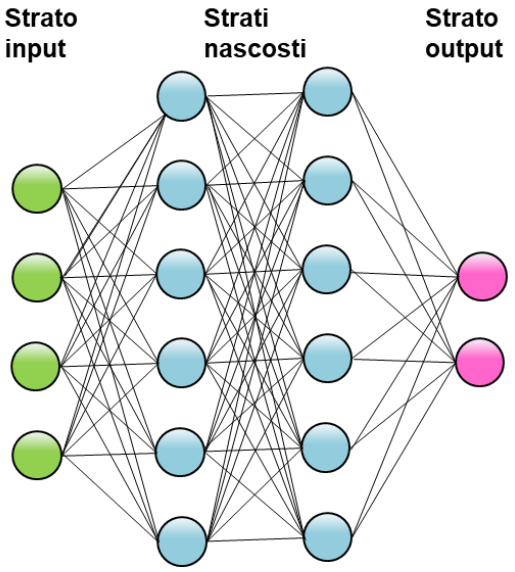
Per *Deep Learning* (*DL*) si intende un insieme di architetture di RN organizzate in diversi strati, dove ogni strato calcola i valori per quello successivo affinché l'informazione venga elaborata in maniera sempre più completa.

Viene applicato soprattutto nel **riconoscimento delle immagini** e della **lingua parlata**, nell'elaborazione del **linguaggio naturale** e nella **bioinformatica**[1].

Nel deep learning ci sono molti modi per configurare l'architettura di rete e la processo di apprendimento.

È pertanto essenziale che tutti i parametri siano scelti con cura poiché un valore di parametro o un metodo non adatto può portare a risultati di scarsa utilità.

Un buon punto di partenza per acquisire familiarità con il *DL* è la documentazione sul Deeplearning4j (Deep Learning for Java).



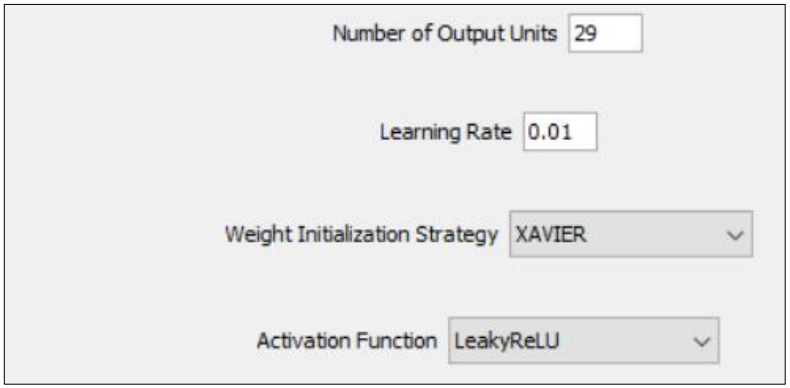
Deep Learning - Architetture

Ci possono essere diversi tipi di architetture Deep Learning come le **RNN** (Recurrent Neural Networks) usate per il riconoscimento vocale e le **CNN** (Convolutional Neural Networks) usate nell'elaborazione delle immagini[1].

Queste RN fanno un utilizzo massivo di strati nascosti e neuroni all'interno di essi. Ci possono essere diverse tipologie di strati: quelli pienamente connessi (*fully connected layer*), come nell'architettura MLP, vengono chiamati **Dense layer**.

Questi strati prevedono:

- numero di unità nascoste
- learning rate
- metodo per l'inizializzazione dei pesi
- funzione di attivazione



[1] In queste note viene utilizzata per semplicità l'architettura **Feedforward**.

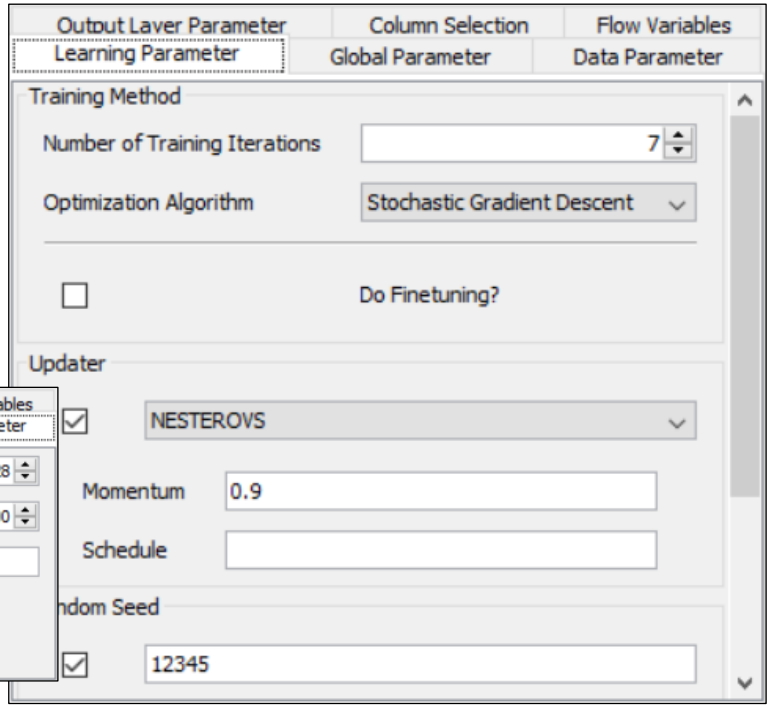
Deep Learning - Parametri

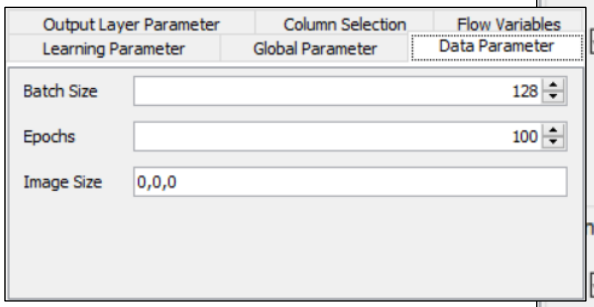
Come per gli altri tipi di RN c'è la possibilità di indicare parametri per

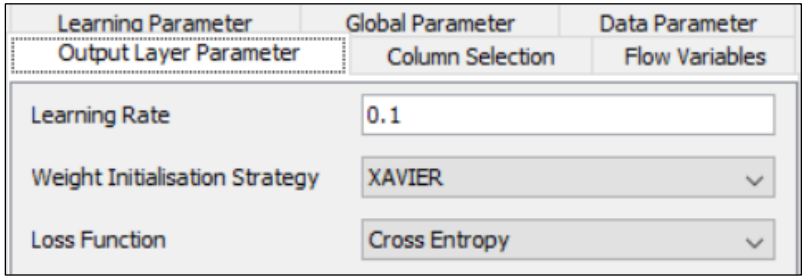
Apprendimento
- l'algoritmo di ottimizzazione
- il momento
- le iterazioni

Dati
- il batch-size
- le epoche

Strato di output
- il learning rate
- il metodo per l'inizializzazione dei pesi
- la funzione di costo



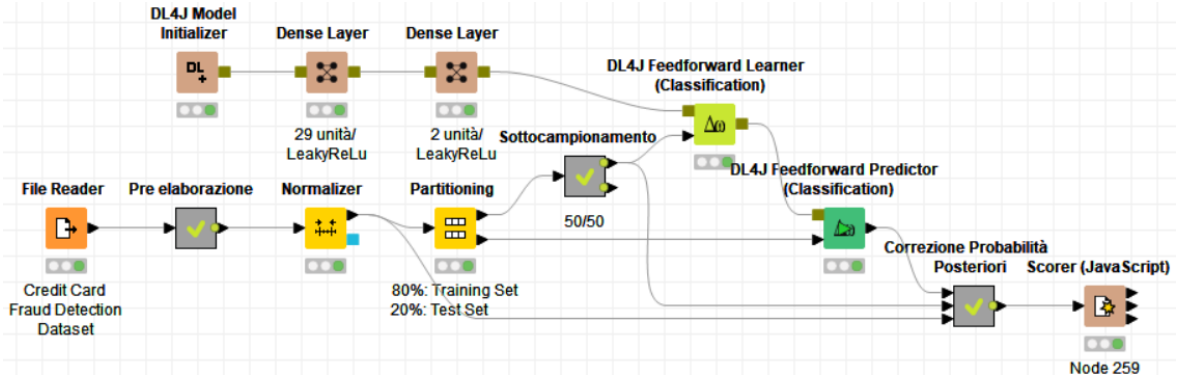




Deep Learning – Modulo2_Esempio6

In questa tabella di 284.807 di transazioni, solo 492 (**0,17%**) sono state individuate come fraudolente.

Per il forte sbilanciamento della classe di interesse si è applicato un **ri-campionamento** qui rappresentato dal Metanodo "*Sottocampionamento*" che riproporziona di dati del training set a un 50/50.



I dati sono stati poi suddivisi in due sottoinsiemi: l'80% delle righe viene utilizzato per la costruzione del modello (training set), mentre il restante 20% viene utilizzato per la valutazione (validation set).

I risultati ottenuti dal nodo DL4J Feedforward Predictor vengono poi corretti con il Metanodo *"Correzione Probabilità Posteriori"* per riportarli alla distribuzione originaria della colonna target.

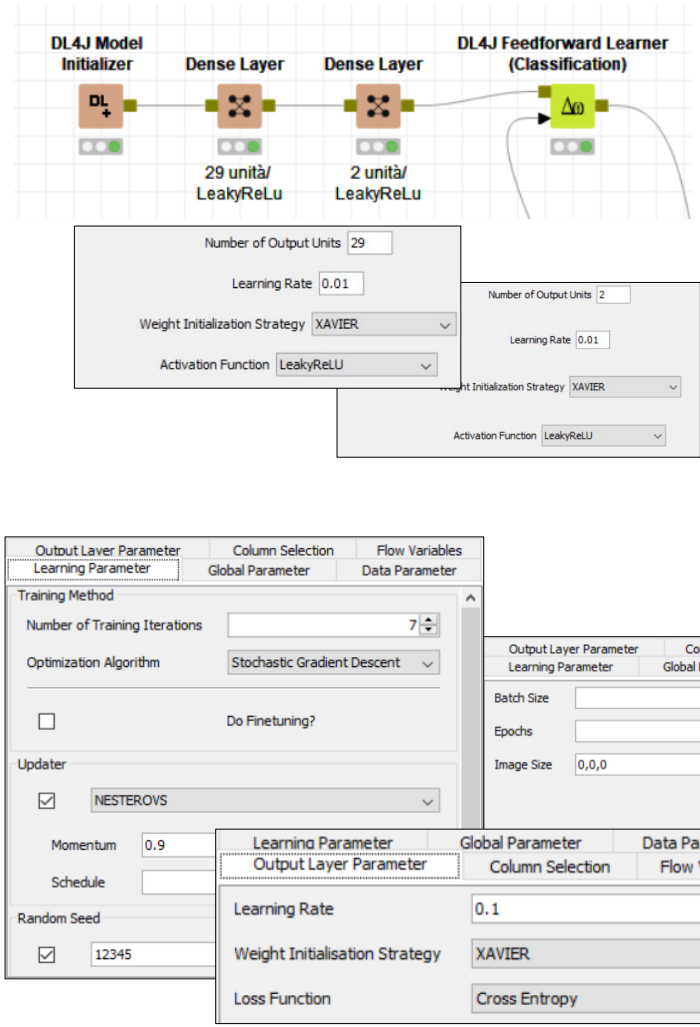
Gli input sono 28 componenti principali prese dal Credit Card Fraud Detection dataset.

Deep Learning – Modulo2_Esempio6

In questo esempio vengono usati due "dense layer": uno contenente 29 unità nascoste (tante quante gli input più il target) e l'altro 2 (target binario).

A ognuno viene imposto un learning rate di 0,01 con una funzione di attivazione di tipo LeakyReLu e Xavier come metodo di inizializzazione di pesi.

Per l'apprendimento si è applicato un batch-size di 128 righe e 7 iterazioni in modo da comprendere tutte le righe del training set (788), 100 epoche e la "cross entropy" come funzione di perdita.



Deep Learning – Modulo2_Esempio6

La matrice di confusione ottenuta dimostra la bontà del modello con un 99,9% di accuratezza (ricordando che la probabilità a priori della colonna target è dello 0,17%!) e buone capacità predittive sia per i dati presenti 78,6% sia per quelli nuovi 80,2%.

Confusion Matrix

Rows Number : 56962	0 (Predicted)	1 (Predicted)	
0 (Actual)	56845	19	99.97%
1 (Actual)	21	77	78.57%
	99.96%	80.21%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
99.93%	0.07%	0.793	56922	40

Modelli – Vantaggi e svantaggi

Nel machine learning non c'è un algoritmo che funziona meglio per ogni tipo di problema, in quanto bisogna tener conto di molti fattori, tra i quali la dimensione e la struttura del dati e, non da ultima, la capacità di interpretare il modello.

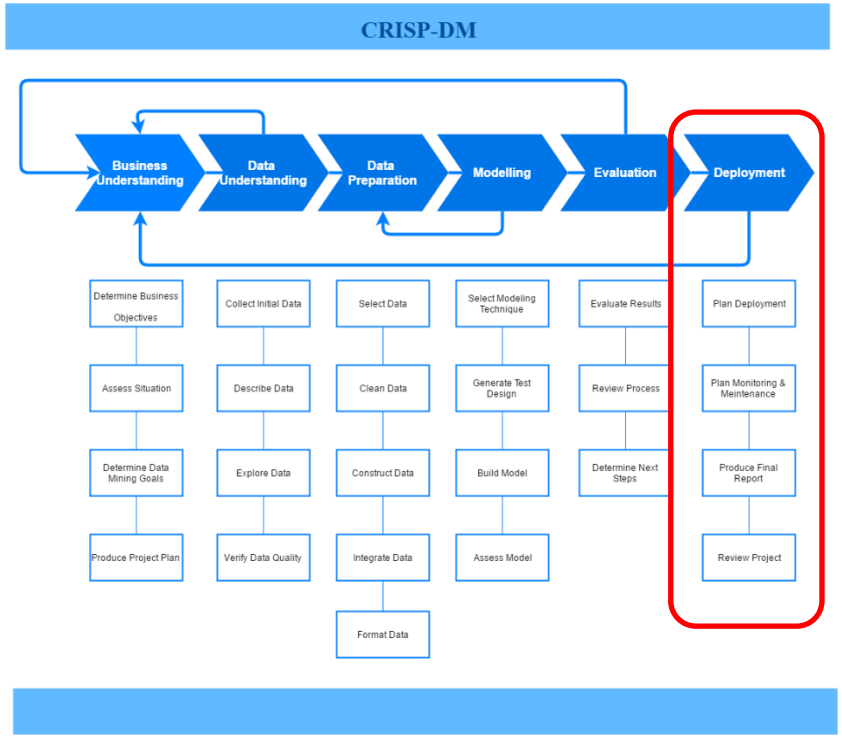
Come sempre la soluzione sta nel provare diversi algoritmi e testare I risultati su un validation set per valutare le capacità predittive e scegliere il migliore.

Metodo	Vantaggi	Svantaggi
Logistic Regression	Il più utilizzato; facile implementazione; addestramento veloce; i risultati si possono interpretare.	Applicabili a soluzioni di tipo lineare; poco robusto (outlier e valori mancanti).
Decision Trees	Risultati molto facili da interpretare e da spiegare; Non parametrico (applicabile a dati non separabili linearmente).	Considerano una colonna alla volta per cui non colgono le interazioni tra le colonne; sono portati all'*overfitting;* poco adatti per colonne di tipo continuo.
Random Forest	Facili da usare, molto performanti; *overfitting* ridotto.	Sono delle black box. Si può valutare l'importanza di ogni singola colonna.
Neural Network	Molto performanti, scoprono relazioni non lineari tra i dati.	Sono delle black box; tempi di addestramento lunghi; richiedono grandi risorse computazionali
Deep Learning	Adatte per classificare dati di tipo immagini, audio e testo.	Come Neural Network.

La Metodologia

Il ***Deployment*** *è la fase di messa in funzione del processo*:

- Rilascio in produzione
- Monitoraggio e manutenzione
- Reporting

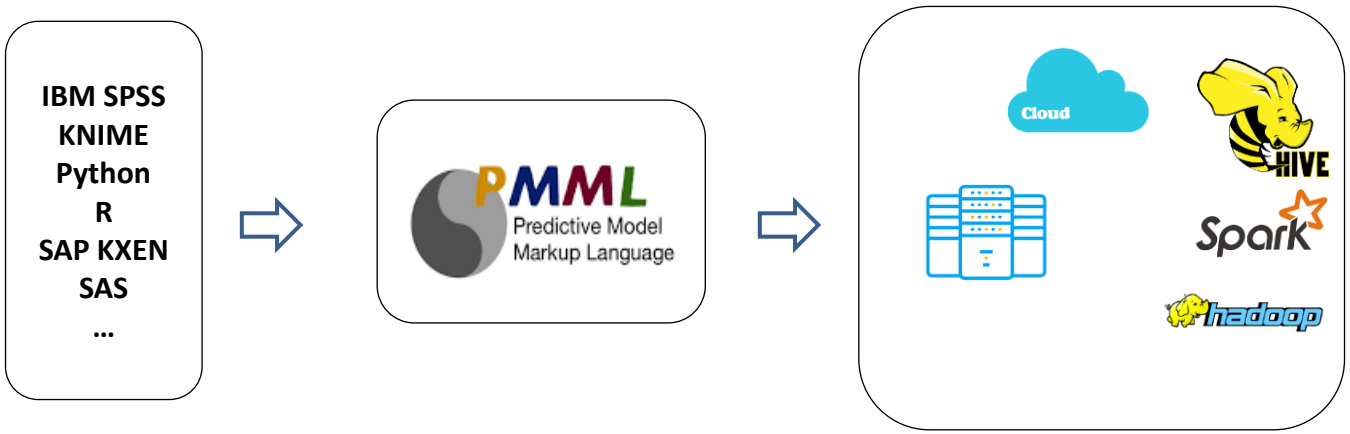


Rilascio in produzione

Il codice prodotto dai modelli predittivi, una volta costruiti e validati nei sistemi informativi aziendali, deve essere ricodificato per calcolare lo score su nuovi dati.

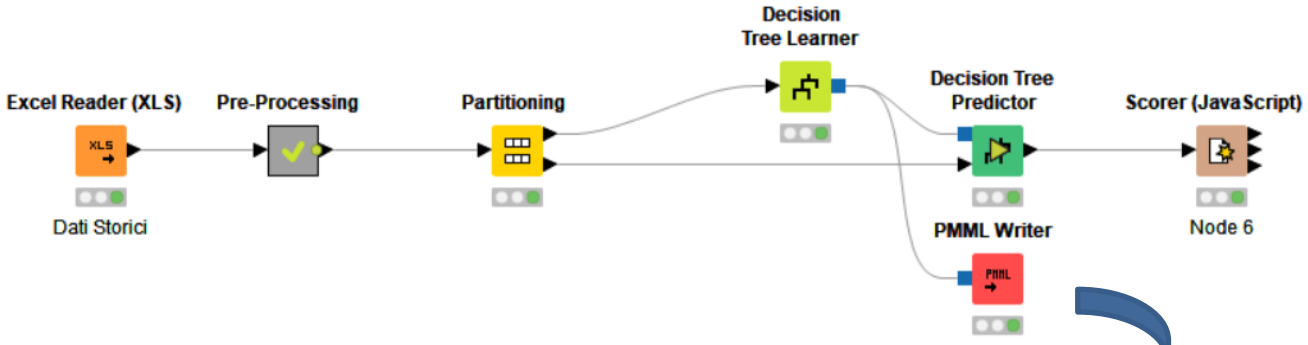
Ciò può essere fatto in vari modi, il più utilizzato è il PMML.

Il Predictive Model Markup Language è uno standard basato su XML come specificato dal Data Mining Group (http://www.dmg.org) che è supportato da tutti i migliori strumenti di data mining commerciali e open source. Un modello predittivo una volta rappresentato come file PMML può essere facilmente spostato nell'ambiente di produzione conforme a questo standard (non c'è bisogno del software che l'ha prodotto).

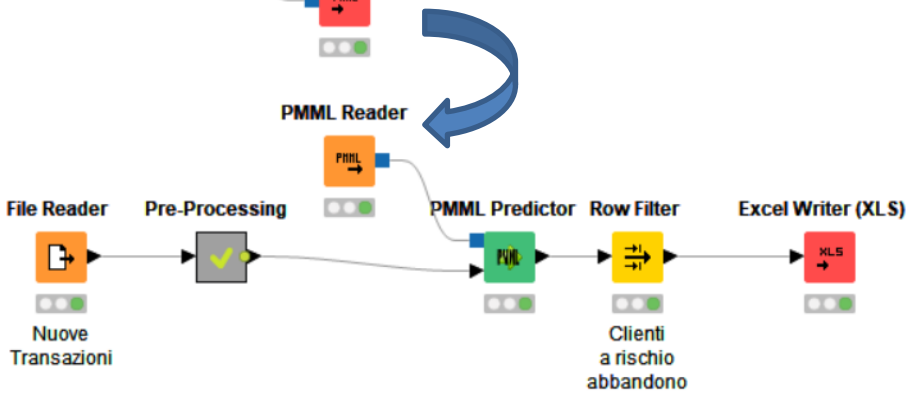


Rilascio in produzione – Modulo2_Esempio7

In questo esempio si vuole prevedere chi è in procinto di abbandonare la sottoscrizione all'utenza email.



In questo esempio viene usato un nodo di Knime per l'importazione del PMML solo come esempio; può essere qualsiasi piattaforma conforme a questo standard.



Deployment

Monitoraggio e manutenzione

Il modello dovrà essere valutato periodicamente, per garantirne l'efficacia e, possibilmente, per essere migliorato.

Reporting

È importante presentare i risultati alle persone coinvolte nel progetto, che può comprendere sia i tecnici che saranno responsabili della manutenzione del processo, sia chi dovrà prendere decisioni (marketing, gestione, …) in base ai risultati ottenuti.

Utilizzo del software Knime – Modulo2_Esercizio1 (parte 1)

- Importare con il nodo **Excel Reader** dalla cartella ***Dati*** nella chiavetta Usb la tabella ***Carta_Prepagata.xls***;

- cambiare la tipologia della colonna *Risposta* con il nodo **Number to String**;

- aggiungere il nodo **Partitioning** per portare in uscita la tabella di training (70%) e quella di validation (il restante 30%), il campionamento dev'essere stratificato sulla colonna *Risposta*;

- Collegare l'uscita del training set al nodo **Decision Tree Learner** e quella del validation set al nodo **Decision Tree Predictor**; collegare la porta di uscita del modello del Learner alla porta di ingresso del modello del Predictor;

- Visualizzare la scelta di menu Vie: Decision Tree View la struttura dell'albero e interpretare i risultati;

- Collegare il nodo **Scorer (JavaScript)** e selezionare come Actual Column la colonna *Risposta* e come Predicted Column la colonna *Prediction (Risposta)*

Utilizzo del software Knime – Modulo2_Esercizio1 (parte 2)

- collegare l'uscita del training set al nodo **Logistic Regression Learner** e quella del validation set al nodo **Logistic Regression Predictor**; collegare la porta di uscita del modello del Learner alla porta di ingresso del modello del Predictor;

- collegare la seconda porta in uscita del nodo Learner il nodo **Math Formula** per creare la colonna ODDS RATIO la cui formula è *exp($Coeff.$)*;

- interpretare i risultati;

- configurare il nodo **Decision Tree Predictor** selezionando Change prediction column name inserendo nel campo Append column with normalized class distribution il suffisso il testo *" Dtree*";

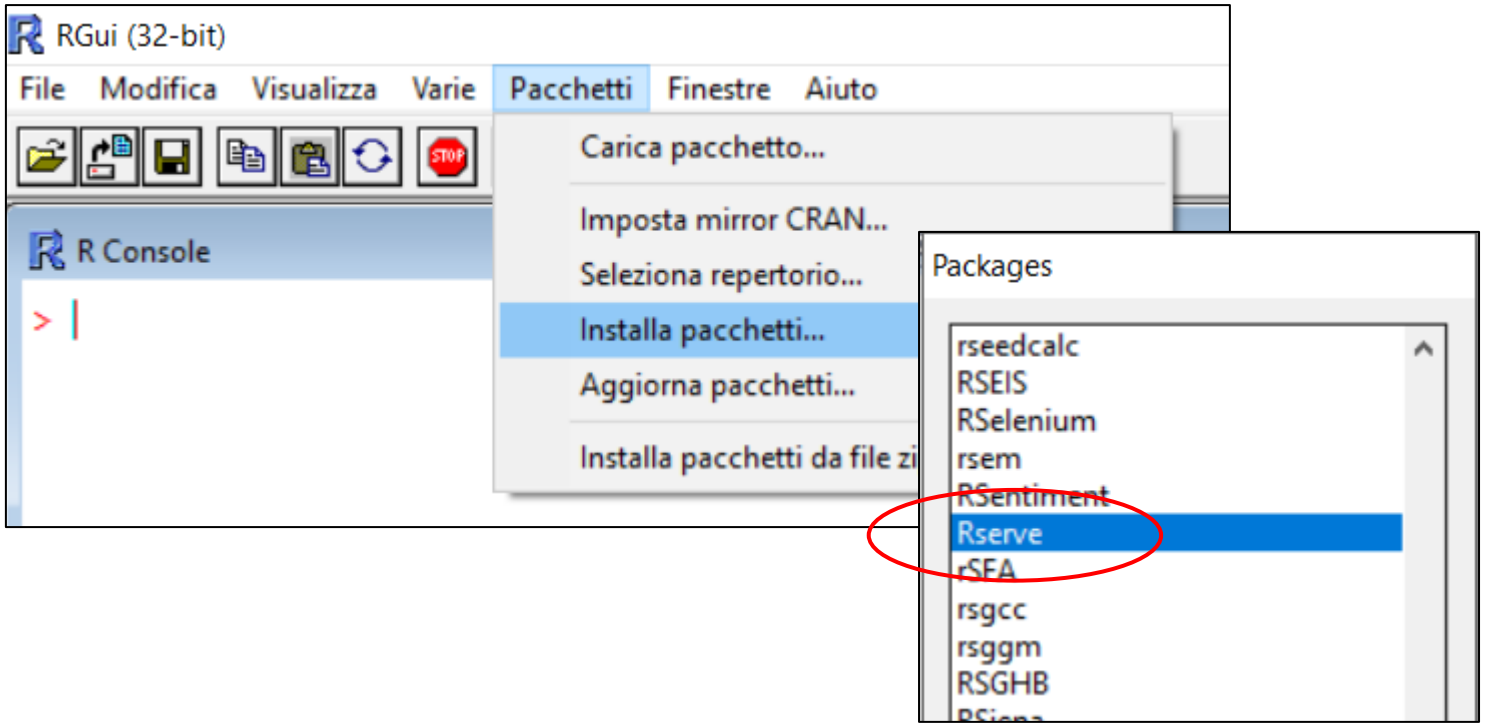
- configurare il nodo **Regression Tree Predictor** selezionando Change prediction column name inserendo nel campo *Append column with normalized class distribution* il suffisso " *LR*";

- unire le uscite di questi due nodi con il nodo **Joiner** per la colonna ID_Cliente selezionando solo i campi Risposta, P (Risposta=1) Dtree e P (Risposta=1) LR;

- collegare il nodo **ROC Curve** selezionando 1 come positive class value; valutare le 2 curve.

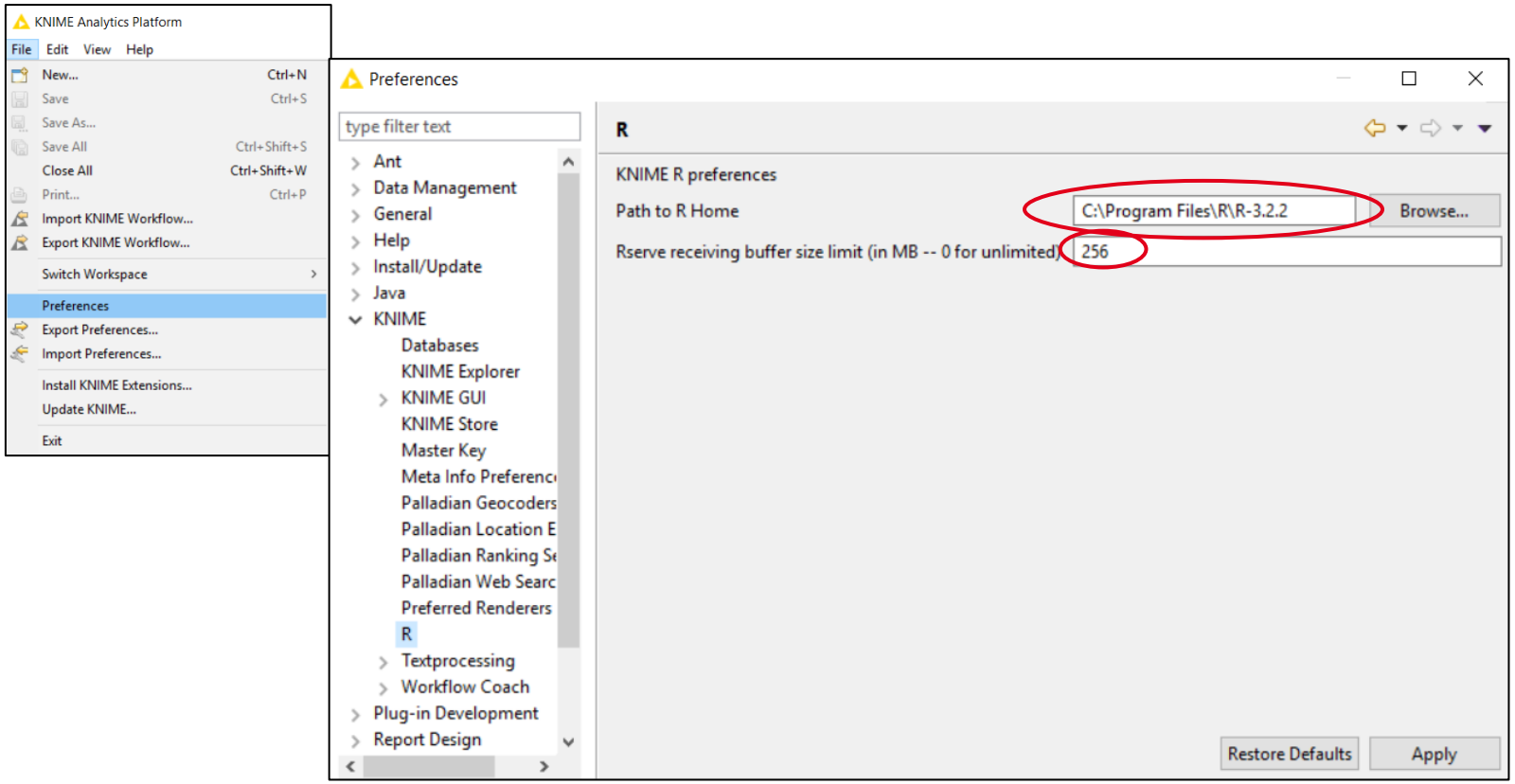
Integrazione con R – 1/2

Installazione in R del package **Rserve**



Integrazione con R – 2/2

Indicare nelle preferenze il percorso dell'installazione di R



Appendice

Integrazione con Python[1] – *(1/2)*

1. Installare il package Anaconda: https://www.anaconda.com/distribution/

2. Scaricare questo file di configurazione in una cartella (p.e in ***C:\Temp***)

3. Aprire il prompt di Anaconda ed eseguire questo comando:
   conda env create -f C:\Temp\py36_knime.yml

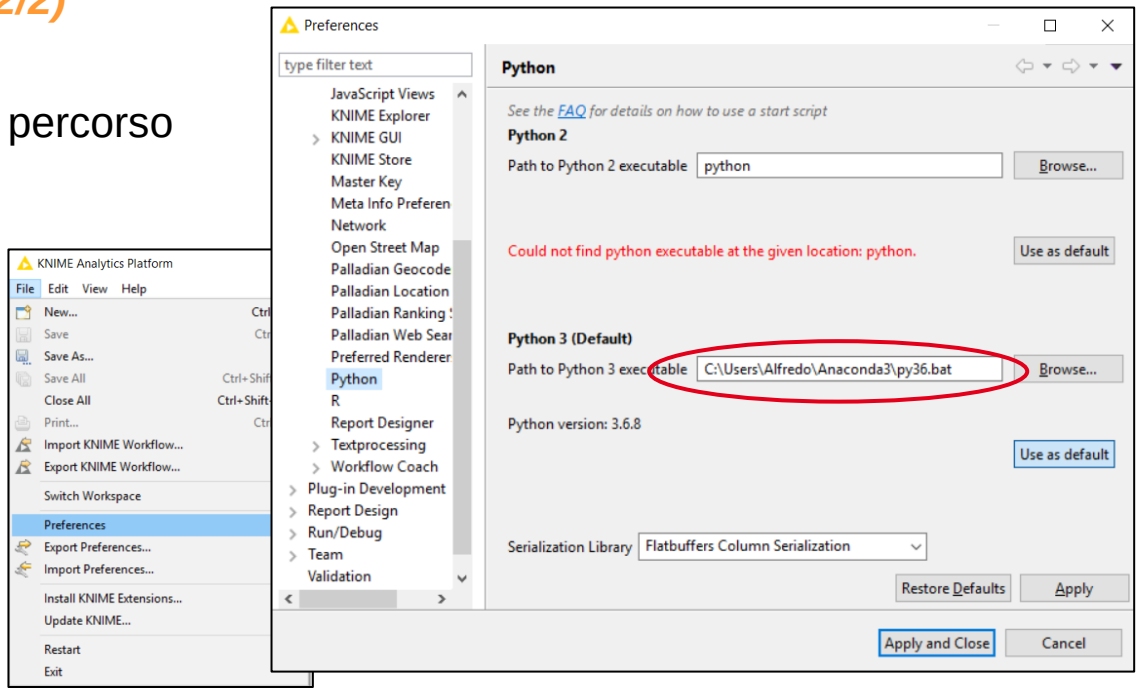
4. Creare il file **py36.bat** in una cartella (p.e. ***C:\Users\<utente>\Anaconda3\Script***) contenente queste righe:

```
@REM Adapt the folder in the PATH to your system
@SET PATH=C:\Users\Alfredo\Anaconda3;%PATH%
@CALL activate py36_knime || ECHO Activating python environment failed
@python %*
```

[1] Queste istruzioni sono si trovano in https://docs.knime.com/2018-12/python_installation_guide/python_installation_guide.pdf

Integrazione con Python – *(2/2)*

5. Indicare nelle preferenze il percorso dell'installazione di Python

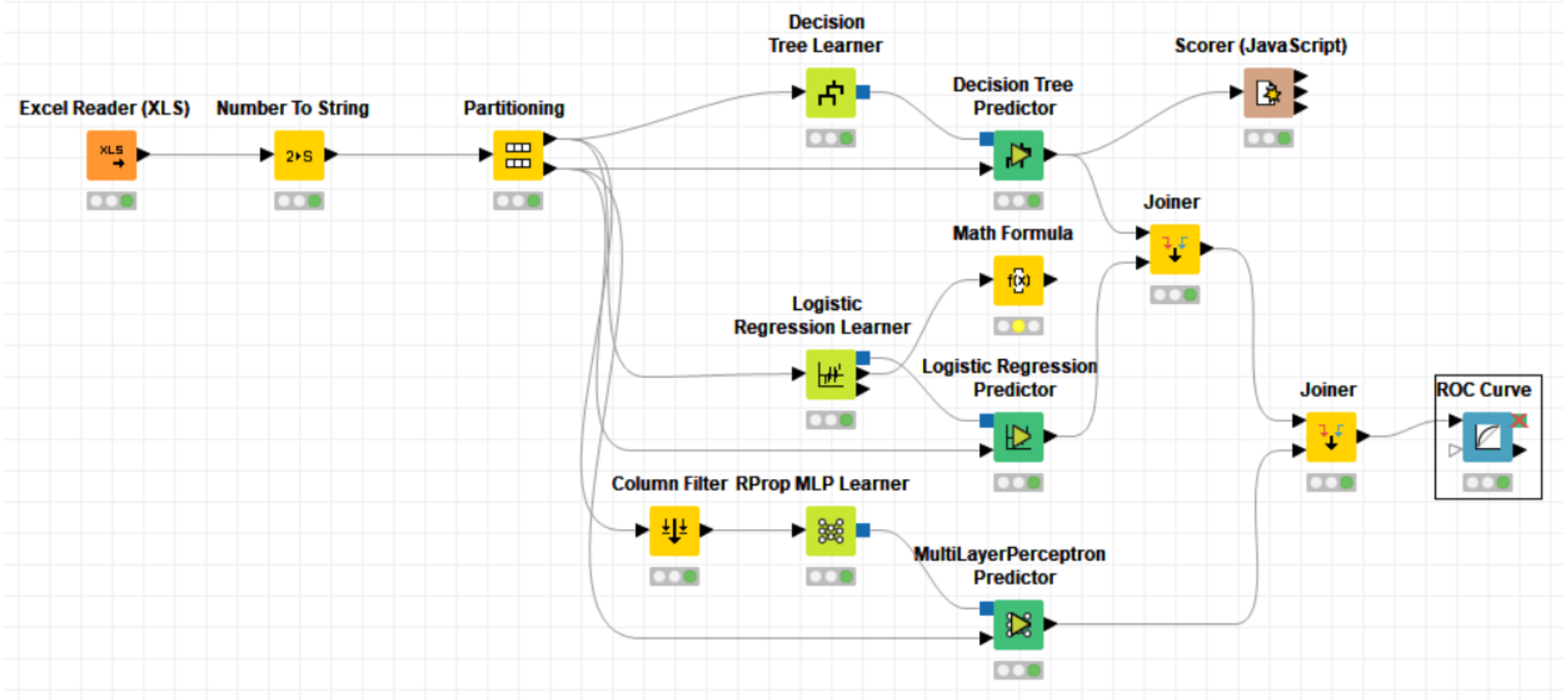


N.B. Non tutti i package di Python sono presenti nell'integrazione. Per poterne utilizzare altri bisogna installarli. Per esempio, per il package ***scikit-learn*** bisogna prima installarlo in Anaconda e poi in Knime. Aprire il prompt di Anaconda ed eseguire questi comandi:

conda install scikit-learn oppure ***pip install -U scikit-learn***
conda install -n py36_knime scikit-learn

Appendice

- Soluzione Modulo2_Esercizio1



"*The cleverest algorithms are no substitute for human intelligence and knowledge of the data in the problem*."

Leo Breiman e Adele Cutler