



The
Artificial Intelligence

vector background

BUSINESS INTELLIGENCE

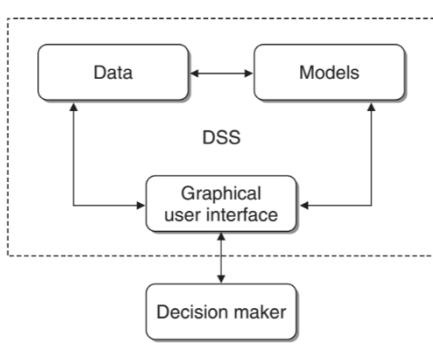
Il **Business Pressure-Response Model** è un quadro concettuale che descrive come le aziende rispondono alle pressioni esterne per rimanere competitive e sostenibili. Questo modello è usato per analizzare le dinamiche tra ambiente esterno, strategia aziendale e tecnologie dell'informazione.

Ci sono dei fattori ambientali, detti business pressure, che influenzano l'attività di business e che quindi la costringono a rispondere in certo modo.

La risposta può essere:

- **reattiva** l'azienda risponde solo quando la pressione diventa critica e richiede un'azione immediata.
- **anticipativa** l'azienda prevede le pressioni future e si prepara in anticipo per minimizzare l'impatto.
- **adattiva** l'azienda si adatta continuamente alle mutevoli condizioni di mercato e alle pressioni esterne.
- **proattiva**, l'azienda non solo si adatta, ma cerca attivamente di influenzare l'ambiente esterno, guidando il cambiamento.

Tutte queste scelte, venivano prese con il **Managerial Decision Making** attraverso il quale i manager prendono decisioni strategiche, operative e tattiche per permettere all'azienda di raggiungere i propri obiettivi tenendo conto di diversi fattori, come gli obiettivi aziendali, le risorse disponibili, e le condizioni di mercato.



Un **sistema di supporto alle decisioni** (DSS) è un'applicazione interattiva computer-based che combina dati e modelli matematici per aiutare i decisori a risolvere problemi complessi affrontati nella gestione di imprese. Quindi un DSS ha bisogno di un database, un repository di modelli matematici e un modulo per gestire il dialogo tra sistema e gli utenti.

A seconda del tipo di decisione da prendere, ci sono diversi modelli di DSS che possono supportare le aziende in modi differenti.

1. DSS basato sui dati (Data-Driven DSS)

Questo tipo di sistema è un enorme archivio che raccoglie e analizza dati per aiutare a prendere decisioni. Ad esempio, un'azienda che vende abbigliamento potrebbe usare un **DSS basato sui dati** per capire quali prodotti vendono meglio in certe stagioni e quindi rifornire i negozi di conseguenza. In pratica, questi DSS

permettono di visualizzare grafici, statistiche e trend che aiutano a prendere decisioni basate su fatti concreti.

2. DSS basato su modelli (Model-Driven DSS)

Immagina di dover prendere una decisione complessa, come scegliere il prezzo giusto per un nuovo prodotto. Un **DSS basato su modelli** ti aiuta a simulare diversi scenari per vedere cosa succederebbe se fissassi un prezzo più alto o più basso. Utilizzando formule matematiche e modelli statistici, questo sistema ti mostra le possibili conseguenze delle tue scelte, aiutandoti a trovare l'opzione migliore.

Definire una tassonomia delle decisioni può rivelarsi utile durante la progettazione di un DSS, poiché è probabile che processi decisionali con caratteristiche simili possano essere supportati dallo stesso set di metodologie. Le decisioni possono essere classificate in termini di due dimensioni principali, in base alla loro **natura e portata**. In base alla loro natura, le decisioni possono essere classificate come strutturate, non strutturate o semi-strutturate.

Decisioni strutturate Una decisione è strutturata se si basa su una procedura decisionale ben definita e ricorrente. Nella maggior parte dei casi, le decisioni strutturate possono essere ricondotte a un algoritmo, che può essere più o meno esplicito per i decisori e quindi più adatto all'automazione. Più specificamente, abbiamo una decisione strutturata se i flussi di input, i flussi di output e le trasformazioni eseguite dal sistema possono essere chiaramente descritti nelle tre fasi di intelligenza, progettazione e scelta. In questo caso, diremo anche che ogni fase componente è strutturata a sua volta. In realtà, anche le decisioni che appaiono pienamente strutturate richiedono nella maggior parte dei casi l'intervento diretto dei decisori per far fronte a eventi inattesi, causati ad esempio da valori insoliti di alcuni flussi di input.

Decisioni non strutturate Una decisione si dice non strutturata se anche le tre fasi di intelligence, progettazione e scelta sono non strutturate. Ciò significa che per ogni fase c'è almeno un elemento nel sistema (flussi di input, flussi di output e processi di trasformazione) che non può essere descritto in dettaglio e ridotto a una sequenza predefinita di passaggi. Un evento del genere può verificarsi quando un processo decisionale viene affrontato per la prima volta o se accade molto raramente. In questo tipo di decisioni il ruolo dei knowledge worker è fondamentale e i sistemi di business intelligence possono fornire supporto ai decisori attraverso un accesso tempestivo e versatile alle informazioni.

Decisioni semi-strutturate Una decisione è semi-strutturata quando alcune fasi sono strutturate e altre no. La maggior parte delle decisioni affrontate dai

knowledge worker nella gestione di imprese o organizzazioni pubbliche o private sono semi-strutturate. Quindi, possono trarre vantaggio dai DSS e da un ambiente di business intelligence principalmente in due modi. Per le fasi non strutturate del processo decisionale, gli strumenti di business intelligence possono offrire un tipo di supporto passivo che si traduce in un accesso tempestivo e versatile alle informazioni. Per le fasi strutturate è possibile fornire una forma di supporto attivo attraverso modelli matematici e algoritmi che consentono di automatizzare parti significative del processo decisionale

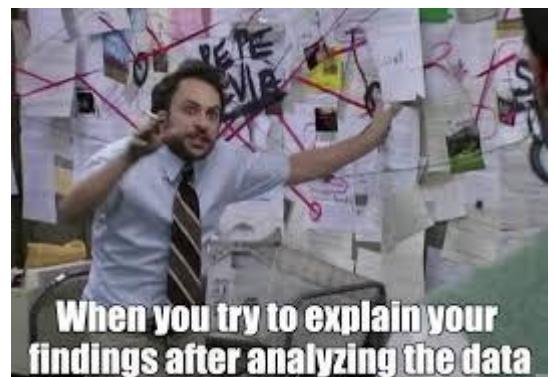
Le decisioni possono essere classificate come

- **strategiche** quando influenzano l'intera organizzazione o almeno una parte sostanziale di essa per un lungo periodo di tempo. Le decisioni strategiche influenzano fortemente gli obiettivi e le politiche generali di un'impresa.
- **Tattiche** quando influenzano solo parti di un'impresa e sono solitamente limitate a un singolo reparto. Le decisioni tattiche si inseriscono nel contesto determinato dalle decisioni strategiche;
- **Operative** che si riferiscono ad attività specifiche svolte all'interno di un'organizzazione e hanno un impatto modesto sul futuro. Le decisioni operative sono inquadrati negli elementi e nelle condizioni determinati dalle decisioni strategiche e tattiche.

Tutte queste decisioni possono essere prese con supporto tramite i sistemi Informativi fornendo strumenti per l'analisi dei dati e business intelligence; l'automazione dei processi ecc.

La **data analytics** è il processo di esaminare, trasformare e modellare i dati per estrarre informazioni utili, supportare il processo decisionale e identificare pattern o tendenze. Questo campo utilizza tecniche di statistica, matematica, machine learning e strumenti software per analizzare dati strutturati e non strutturati.

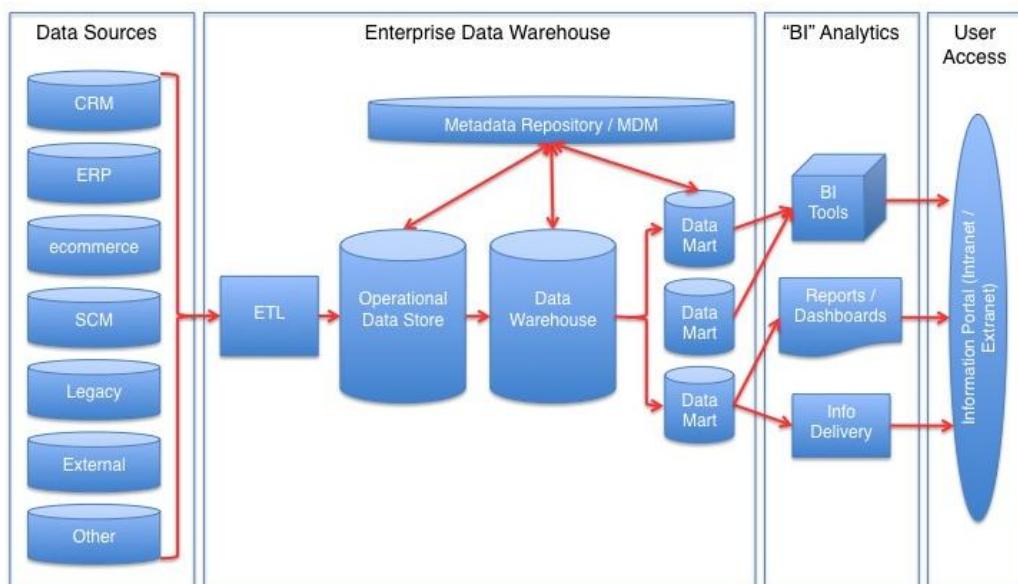
La **Business Intelligence** è una particolare applicazione della **Data Analytics** focalizzata sull'analisi e l'interpretazione dei dati aziendali per supportare le decisioni strategiche. Il **Machine Learning** si basa sulla creazione di modelli in grado di **apprendere dai dati** per fare previsioni o prendere decisioni in modo automatico. Il **Data Mining** analizza grandi quantità di dati per trovare pattern, correlazioni o informazioni utili che magari non erano subito evidenti.



La BI aiuta a trasformare i dati, in informazioni (e conoscenza), in decisioni e infine in azioni. I **dati** sono rappresentazioni elementari di fatti, eventi o entità che, da soli, non hanno necessariamente un significato. L'**informazione** è il risultato di attività di estrazione ed elaborazione svolte sui dati e appare significativa per chi la riceve in un dominio specifico. L'informazione si trasforma in conoscenza quando viene utilizzata per prendere decisioni e sviluppare le corrispondenti è detta **conoscenza**. Le conoscenze estratte porteranno ad **azioni** volte a risolvere il problema rilevato

Gli elementi che caratterizzano un sistema di BI sono:

- le **sorgenti dati** che sono le fonti da cui vengono prese le informazioni ai quali un processo ETL è applicato. Questo processo prevede l'estrazione, la trasformazione dei dati grezzi cioè vengono puliti organizzati e convertiti nel formato utile, il load cioè i dati vengono caricati nel **data warehouse**. I sistemi di archiviazione usati sono in generale dei database dove vengono memorizzati i dati strutturati utili al sistema informativo aziendale. Possono essere memorizzati nell'archivio centrale o in un **datamart** che rappresenta un sottoinsieme dei dati del data warehouse in generale usato per rappresentare esigenze specifiche
- una componente software detta **OLAP OnLine Analytical Processing** che serve per analizzare grandi quantità di dati che permette di raggruppare i dati e svolge l'operazione di interrogazioni su di essi
- strumenti di analisi messi a disposizione agli utenti per interrogare i dati fare report e usare algoritmi di data mining per fare previsioni.



ETL Extract, Transform, Load è un processo usato per integrare dati provenienti da diverse fonti in un unico repository.

1. **Estrazione:** Questa fase comporta la lettura e l'ottenimento di dati da una o più fonti di dati. Queste fonti possono essere molto diverse, come database relazionali, sistemi legacy, applicazioni ERP e CRM, file flat, fogli di calcolo Excel o persino code di messaggi. L'obiettivo è raccogliere i dati rilevanti per l'analisi.
2. **Trasformazione:** In questa fase, i dati estratti vengono convertiti dal loro formato originale in un formato che è più adatto per l'analisi e per essere caricato nel sistema di destinazione.
3. **Caricamento:** Questa è la fase finale in cui i dati trasformati vengono scritti nel repository di destinazione, che di solito è un data warehouse o un altro database centralizzato.

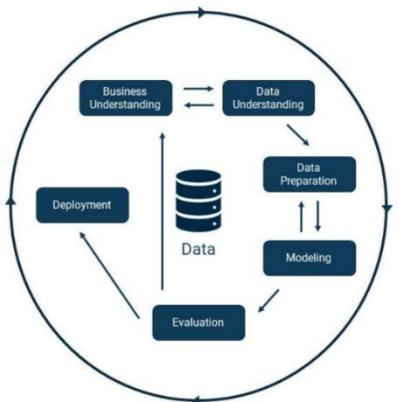
Gli stessi risultati si potrebbero ottenere utilizzando metodi statistici basati su variabili casuali e relazioni di probabilità usando indici di distribuzioni di probabilità e altre tecniche per capire l'andamento di un fenomeno. Si tratta di un modello rigoroso che non sempre può essere usato per fare analisi dei dati.

In questo caso conviene usare tecniche di data Analytics che combina tecniche statistiche matematiche e di machine learning per effettuare analisi dei dati. Utilizzando i dati disponibili e applicando particolari algoritmo di machine learning è possibile ottenere dai dati delle informazioni. Con il ML è possibile fare:

- **Analitiche descrittive** con la quale raccontiamo la storia passata tramite i dati e l'area assumiamo tramite indici statistici
- **Analisi predittiva** cioè prevediamo il futuro della base dei dati presenti in modo da prevedere l'andamento di un certo utente o di prendere decisioni. Il **trend** è un pattern o una direzione ricorrente di come i dati evolvono nel tempo.

CRISP-DM

Il modello **CRISP-DM (CROSS Industry Standard Process for Data Mining)** descrive il ciclo di vita di un progetto di data mining attraverso sei fasi. Tuttavia, queste fasi non devono necessariamente essere seguite in modo rigido o lineare, poiché è possibile tornare ai passaggi precedenti per rivedere e migliorare le decisioni prese, qualora non si rivelassero efficaci.



Si tratta di un processo iterativo e continuo: una volta completato un progetto, si avvia un nuovo ciclo di data mining con l'obiettivo di perfezionare i risultati ottenuti e migliorare le strategie adottate.

Business understanding comprensione degli obiettivi e dei requisiti del progetto da una prospettiva aziendale, quindi sulla conversione di questa conoscenza in una definizione del problema di data mining e in un piano preliminare progettato per raggiungere gli obiettivi

Data Understanding raccolta di dati e prosegue con attività volte a familiarizzare con i dati, identificare problemi di qualità dei dati, scoprire le prime intuizioni sui dati o rilevare sottoinsiemi interessanti per formulare ipotesi su informazioni nascoste.

Data preparation attività per costruire il set di dati dai dati grezzi iniziali. È probabile che le attività di preparazione dei dati vengano eseguite più volte e non in un ordine prescritto. Le attività includono la selezione di tabelle, record e attributi, nonché la trasformazione e la pulizia dei dati per gli strumenti di modellazione.

Modelling applicate varie tecniche di modellazione e i loro parametri vengono calibrati su valori ottimali. Pertanto, è spesso necessario tornare alla fase di preparazione dei dati.

Evaluation prima di procedere all'implementazione finale del modello, è importante valutare più il modello e rivedere i passaggi eseguiti per costruirlo per essere certi che raggiunga correttamente gli obiettivi aziendali. Un obiettivo chiave è determinare se ci sono importanti problemi aziendali che non sono stati sufficientemente considerati. Alla fine di questa fase, si dovrebbe prendere una decisione sull'uso dei risultati del data mining.

Deployment La creazione del modello non è la fine del progetto. Anche se lo scopo del modello è quello di aumentare la conoscenza dei dati, la conoscenza acquisita dovrà essere organizzata e presentata in modo che il cliente possa utilizzarla.

ML CANVAS

Il **Machine Learning Canvas** è uno strumento visivo pensato per guidare in modo strutturato la progettazione e la gestione di progetti di machine learning. Si tratta di una sorta di mappa che permette di tenere traccia degli aspetti fondamentali di un progetto, dalla definizione dell'obiettivo all'implementazione finale, passando per le risorse necessarie e le metriche di successo.

1. Proposta di Valore (Value Proposition)

Definisce il problema aziendale che si intende risolvere e il valore che il modello ML apporterà agli utenti finali. È fondamentale comprendere chi beneficerà delle previsioni e in che modo.

2. Compito di Predizione (Prediction Task)

Describe ciò che il modello deve prevedere. Include il tipo di output atteso (classificazione, regressione, ecc.) e le metriche di valutazione delle prestazioni.

3. Dati (Data)

Identifica le fonti di dati disponibili, la loro qualità, quantità e pertinenza rispetto al compito di predizione. Considera anche eventuali limitazioni legali o etiche nell'uso dei dati.

4. Preparazione dei Dati (Data Preparation)

Dettaglia le operazioni necessarie per trasformare i dati grezzi in un formato utilizzabile per l'addestramento del modello, come la pulizia, la normalizzazione e l'ingegneria delle caratteristiche.

5. Addestramento del Modello (Model Training)

Specifica gli algoritmi da utilizzare, le strategie di validazione e i criteri per la selezione del modello migliore.

6. Valutazione del Modello (Model Evaluation)

Describe come verrà valutato il modello, includendo le metriche chiave e i benchmark di riferimento.

7. Implementazione (Deployment)

Pianifica come il modello sarà integrato nel sistema esistente, considerando aspetti come l'infrastruttura, l'automazione e la scalabilità.

8. Monitoraggio (Monitoring)

Stabilisce le modalità di monitoraggio delle prestazioni del modello in produzione, per rilevare eventuali degradi o anomalie nel tempo.

9. Decisioni (Decisions)

Analizza come le previsioni del modello influenzano le decisioni aziendali e quali azioni saranno intraprese in base ai risultati.

10. Valutazione Offline (Offline Evaluation)

Consente di testare il modello su dati storici per valutarne l'efficacia prima della messa in produzione.

Prediction Task	?	Decisions	?	Value Proposition	?	Data Collection	?	Data Sources
Type of task Classification Input object Output definition, parameters, and possible values.		Process for turning predictions into proposed value for the end-user? Mention decision-making parameters.		Who is the end-user? Which of their objectives are we serving?		Strategy for initial train set Output acquisition cost? Continuous update rate? Holdout ratio on prod inputs?		Which raw data sources can we use (internal, external)? Mention databases and tables, or APIs and methods of interest.
Cost/gain values? Deployment criteria (min performance value, fairness)?	✓	When do we make real-time / batch predictions? Time available for this + featurization + post-processing? Compute target?	↔	How will they benefit from the ML system? Mention workflow and interfaces.		How many prod models are needed? When would we update? Time available for this (including featurization and analysis)?	⚙️	Input representations available at prediction time, extracted from raw data sources.
Live Monitoring Metrics to quantify value creation and measure the ML system's impact in production (on end-users and business)?								



STEP1 - Scelta del dataset

quindi i dati, controllando la **clearness** e la **quality**.

Uno degli aspetti più critici riguarda la **selezione dei dati**: un campione non rappresentativo può portare a conclusioni fuorvianti. Questo fenomeno, noto come **bias di selezione**, può essere ridotto utilizzando tecniche di **campionamento casuale**, che garantiscono una raccolta dati più equa e bilanciata.

Oltre alla rappresentatività, anche la **dimensione del dataset** influisce sulla qualità delle analisi. Dataset troppo piccoli o omogenei possono limitare la capacità di generalizzazione dei modelli, mentre set di dati ampi e diversificati permettono di individuare pattern più solidi e fare previsioni più accurate. Se non abbiamo abbastanza dati allora effettuiamo un'operazione di data mining.

Valutiamo l'**inter-rater reliability** misura con la quale si valutano quanto le osservazioni o valutazioni di diversi valutatori siano consistenti tra loro.

La formula di **Cohen's Kappa** tiene conto del caso che i valutatori possano essere d'accordo per pura casualità.

$$\kappa = \frac{P_o - P_e}{1 - P_e}$$

Dove:

- P_o è la **probabilità osservata di accordo** (il numero di volte in cui i valutatori sono d'accordo diviso per il numero totale di valutazioni).
- P_e è la **probabilità attesa di accordo casuale** (basata sulla distribuzione delle classificazioni).

Il valore di κ può variare da -1 (accordo perfettamente negativo) a +1 (accordo perfetto), con valori più alti che indicano un accordo migliore.

L'**Indice di Krippendorff's Alpha** è una misura inter-rater reliability che può essere utilizzata per valutare la coerenza tra diversi valutatori. Questo indice è molto utile quando ci sono più di due valutatori e può essere applicato anche a situazioni in cui i dati non sono simmetrici o non distribuiti normalmente.

$$\alpha = 1 - \frac{D_o}{D_e}$$

Dove:

- D_o è la **discrepanza osservata** tra i valutatori, ossia la somma delle differenze tra le valutazioni date da diversi valutatori, ponderata per il tipo di variabile (nominale, ordinal, intervallare, ecc.).
- D_e è la **discrepanza attesa** sotto l'ipotesi di accordo casuale, ossia la somma delle differenze che ci si aspetterebbe se le valutazioni fossero dovute al caso.



STEP2 – Analisi dei dati

Lo scopo principale dell'analisi esplorativa dei dati è evidenziare le caratteristiche rilevanti di ciascuna feature contenuta in un set di dati, utilizzando metodi grafici e calcolando statistiche riassuntive, e identificare l'intensità delle relazioni sottostanti tra gli attributi.

L'analisi esplorativa dei dati comprende tre fasi principali:

- **analisi univariata**, in cui vengono esaminate le proprietà di ciascun singolo attributo di un set di dati;
- **analisi bivariata**, in cui vengono considerate coppie di attributi, per misurare l'intensità della relazione esistente tra di essi;
- **analisi multivariata**, in cui vengono esaminate le relazioni che sussistono all'interno di un sottoinsieme di attributi.

Analisi univariata ha lo scopo di valutare la tendenza dei valori di un dato attributo a disporsi attorno a uno specifico valore centrale (posizione), misurare la propensione della variabile ad assumere un intervallo più o meno ampio di valori (dispersione) ed estrarre informazioni sulla distribuzione di probabilità.

Media

Rappresenta il valore medio della variabile

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

Moda

La mediana è il valore che si trova nel mezzo di un insieme ordinato di dati.

$$\text{Mediana} = x_{\left(\frac{n+1}{2}\right)}$$

- Se il numero di osservazioni è dispari, è il valore centrale.
- Se è pari, è la media dei due valori centrali.

$$\text{Mediana} = \frac{1}{2} \left(x_{\left(\frac{n}{2}\right)} + x_{\left(\frac{n}{2}+1\right)} \right)$$

Varianza

Misura la dispersione dei dati intorno alla media. Indica quanto variano i valori all'interno del dataset.

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

Deviazione Standard

Fornisce una misura della dispersione nella stessa unità dei dati originali.

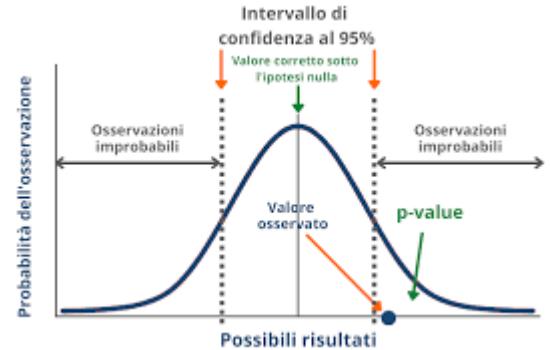
$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2}$$

Distribuzione normale. Se la distribuzione dell'attributo a è normale, o almeno a forma di campana e approssimativamente normale, possiamo dire che:

- l'intervallo $(\mu \pm \sigma)$ contiene approssimativamente il 68% dei valori osservati;
- l'intervallo $(\mu \pm 2 \sigma)$ contiene approssimativamente il 95% dei valori osservati;
- l'intervallo $(\mu \pm 3 \sigma)$ contiene approssimativamente il 100% dei valori osservati.

Un **intervallo di confidenza** rappresenta un **range di valori** entro cui, con una certa probabilità, ci si aspetta che cada il vero parametro della popolazione, basandosi su un campione.

- CIT: “non so il valore esatto, ma so che con alta probabilità è tra A e B”.

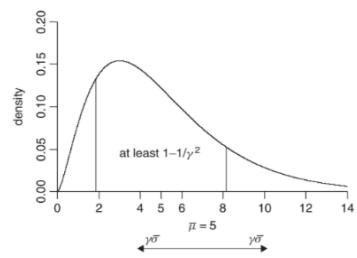


I valori del campione che ricadono al di fuori dell'intervallo $(\mu \pm 3 \sigma)$ possono quindi essere considerati valori anomali sospetti e candidati per la rimozione dal campione.

Distribuzione arbitraria Se la distribuzione dell'attributo a differisce dalla normale, è ancora possibile ottenere intervalli entro i quali ci si può ragionevolmente aspettare che i valori del campione rientrino. Tali intervalli si applicano a qualsiasi distribuzione e sono più conservativi degli intervalli corrispondenti per distribuzioni normali.

Teorema di Chebyshev

Dato un numero $\gamma \geq 1$ e un gruppo di m valori $a = (x_1, x_2, \dots, x_m)$, una proporzione almeno uguale a $(1 - 1/\gamma^2)$ dei valori rientrerà nell'intervallo $(\mu \pm \gamma \sigma)$, vale a dire a non più di γ deviazioni standard dalla media del campione



Per gli **attributi categoriali**, non è possibile usare le classiche misure come media, deviazione standard o percentili. In questi casi, è più utile valutare quanto sono distribuiti i dati tra le diverse categorie, l'**eterogeneità**.

Le **frequenze relative**, indicate con f_h , ci dicono quanto è rappresentata ciascuna categoria rispetto al totale. Se tutte le categorie hanno frequenze simili, vuol dire che i dati sono distribuiti in modo equilibrato: c'è alta eterogeneità, cioè varietà. Quando una sola categoria domina, cioè ha una frequenza molto alta rispetto alle altre, indicata con f_g , allora c'è poca varietà: si parla di bassa eterogeneità, o alta omogeneità. Due misure di eterogeneità per gli attributi categoriali sono **l'indice di Gini** e **l'indice di entropia**.

Indice di Gini L'indice di Gini è definito come

$$G = 1 - \sum_{i=1}^n f_h^2$$

Il suo valore è uguale a 0 nel caso di eterogeneità minima, ovvero quando una classe è assunta a una frequenza uguale a 1 e tutte le altre classi non sono mai assunte. Al contrario, quando tutte le classi hanno la stessa frequenza empirica relativa e viene registrata l'eterogeneità più elevata, l'indice di Gini raggiunge il suo valore massimo $(H - 1)/H$.

È possibile normalizzare l'indice di Gini in modo che assuma valori nell'intervallo $[0,1]$, utilizzando la seguente trasformazione che porta all'indice di Gini relativo:

$$G_{\text{rel}} = \frac{G}{(H - 1)/H}$$

Indice di entropia L'indice di entropia è definito come

$$H = - \sum_{i=1}^n f_h \log(f_h)$$

Il suo valore è uguale a 0 nel caso di eterogeneità minima, cioè quando una classe è assunta a una frequenza uguale a 1 e tutte le altre classi non sono mai assunte. Al contrario, quando tutte le classi hanno la stessa frequenza empirica relativa e viene registrata la massima eterogeneità, l'indice di entropia raggiunge il suo valore massimo $\log_2 H$.

È anche possibile normalizzare l'indice di entropia in modo che assuma valori nell'intervallo [0,1], utilizzando la seguente trasformazione che porta all'indice di entropia relativa:

$$H_{\text{rel}} = \frac{H}{\log_2 H}$$

Analisi bivariata considera **coppie di attributi** per misurare l'intensità della relazione esistente tra loro. È utile distinguere tre casi che possono verificarsi:

- Entrambi gli attributi sono numerici.
- Un attributo è numerico e l'altro è categorico.
- Entrambi gli attributi sono categorici.

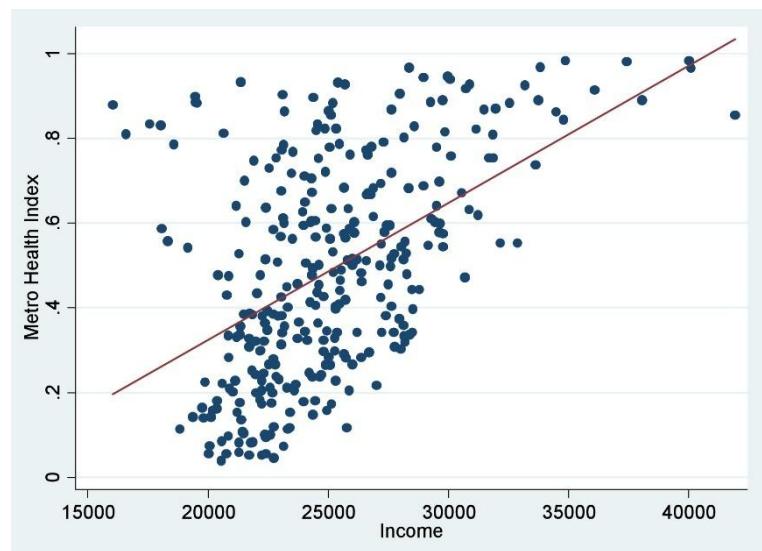
Scatter Plot (Grafico a dispersione)

Un **scatter plot** è un grafico che mostra la relazione tra due variabili numeriche. Ogni punto nel grafico rappresenta una coppia di valori (x, y) di due variabili. Viene spesso utilizzato per visualizzare la **correlazione** tra due variabili.

Ogni punto nel grafico corrisponde a una coppia di valori, uno sull'asse delle **X** (variabile indipendente) e uno sull'asse delle **Y** (variabile dipendente).

Se i punti sono disposti in una linea retta, può indicare una **forte correlazione lineare** tra le due variabili.

La **densità** dei punti in un'area specifica può indicare una **relazione più forte** o un **pattern**.



Utilità

Mostra la **correlazione** tra due variabili (positiva, negativa o nulla).

Aiuta a individuare eventuali **outlier** visibili come punti lontani dalla nube di punti.

Identifica **pattern** non lineari o **tendenze** nei dati.

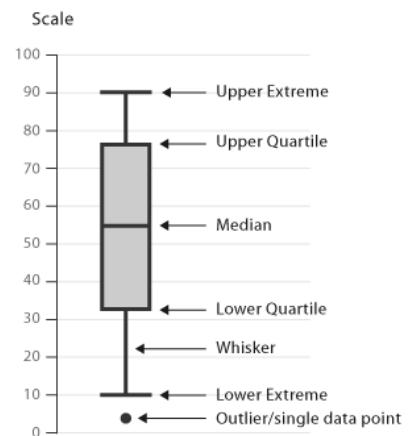
Boxplot

Il **Boxplot** è una rappresentazione grafica che mostra la distribuzione di un insieme di dati attraverso cinque valori chiave: il **minimo**, il **primo quartile** (Q1), la **mediana** (Q2), il **terzo quartile** (Q3), e il **massimo**.

La **scatola** rappresenta l'intervallo interquartile (IQR), che va da Q1 a Q3. Il 50% centrale dei dati si trova all'interno della scatola.

La linea orizzontale dentro la scatola rappresenta la **mediana** (Q2), che è il punto centrale dei dati.

Le "whiskers" (frecce) si estendono dai quartili fino ai valori minimi e massimi, ma non oltre 1,5 volte l'IQR. I dati che si trovano al di fuori di queste linee (oltre i cosiddetti "limiti di Tukey") sono considerati **outlier** e sono rappresentati come punti separati.



Utilità:

Permette di vedere rapidamente la **dispersione** dei dati.

Mostra la **simmetria** o **asimmetria** della distribuzione.

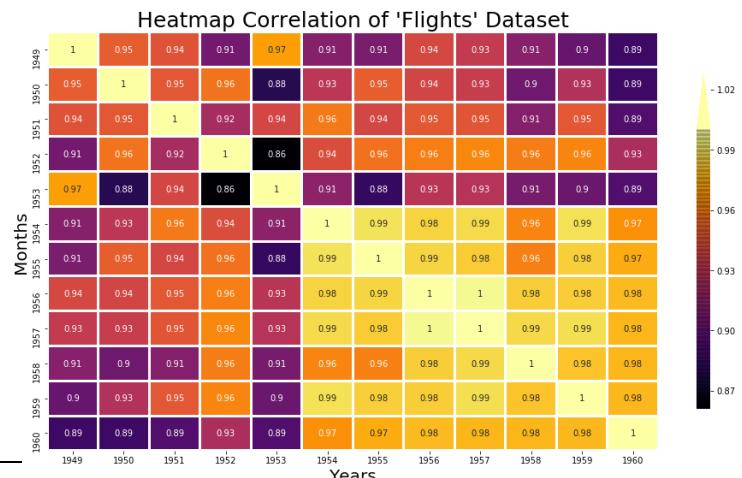
Identifica facilmente gli **outlier** (valori estremi).

Heatmap (Mappa di Calore)

Una **heatmap** è una rappresentazione grafica in cui i valori di una matrice (o di un dataset bidimensionale) sono rappresentati con colori. I colori più scuri o più chiari indicano rispettivamente valori più alti o più bassi.

Ogni cella della matrice rappresenta un valore, e il colore di ciascuna cella indica l'intensità di quel valore.

Può essere usata per rappresentare dati come una **tabella di correlazione** o la **densità** dei dati.



Utilità

Mostra visivamente la **forza delle relazioni** tra variabili, come nel caso delle **matrici di correlazione**.

Utilizza **colori** per evidenziare pattern o anomalie (ad esempio, valori molto alti o bassi).

Eccellente per visualizzare dati complessi in modo facilmente interpretabile.

La **correlazione** di due variabili mostra un legame, una tendenza comune, che può essere positiva (vanno nella stessa direzione) o negativa (vanno in direzioni opposte). Il fatto che due variabili siano correlate non significa automaticamente che una causi l'altra.

La **causalità** cerca di determinare se una variabile provoca direttamente un cambiamento in un'altra.

! CORRELAZIONE NON IMPLICA CAUSALITÀ

Il **test di causalità di Granger** si basa sull'idea che una variabile X causa Y se le passate osservazioni di X (i suoi valori passati) possono essere utilizzate per migliorare la previsione di Y rispetto a una previsione basata solo sui valori passati di Y.

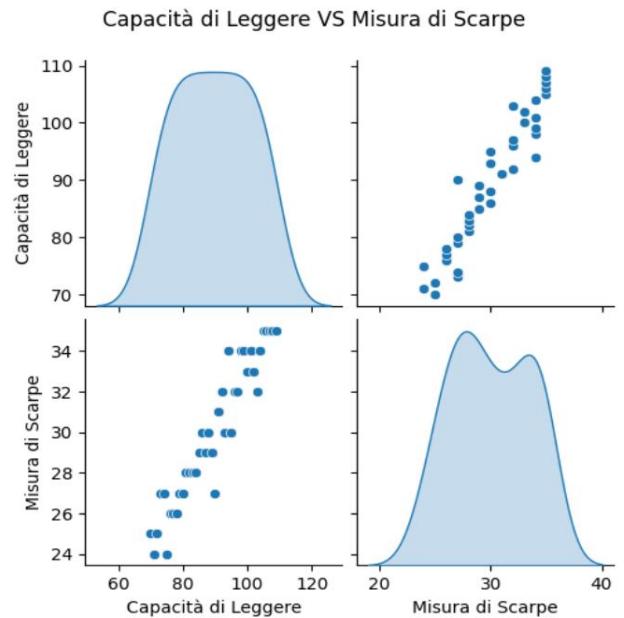
La causalità di Granger non implica necessariamente una relazione di causa ed effetto in senso stretto, ma solo che i valori passati di una variabile aiutano a prevedere quelli di un'altra.

Il test esamina se l'inclusione dei **lag** (ritardi) dei valori di una variabile X (ovvero i valori passati di X) aiuta a migliorare la previsione di una variabile Y. Il procedimento consiste nel confrontare due modelli:

1. Modello senza lag di X: Prevediamo Y solo usando i valori passati di Y.
2. Modello con lag di X: Prevediamo Y usando sia i valori passati di Y che i valori passati di X.

Se il modello con il lag di X riduce significativamente l'errore di previsione di Y, possiamo dire che X causa Granger Y.

Per visualizzare graficamente la distribuzione relativa di diverse features usiamo il **pairplot** che ci permette di visualizzare le relazioni tra le variabili in un unico grafico. Il pairplot riporta su entrambi gli assi le variabili del nostro dataset e ogni cella corrisponde ad una combinazione di queste. Sulla diagonale abbiamo quindi la variabile rispetto a se stessa e la cella mostra la sua distribuzione. In ogni altra cella abbiamo la distribuzione relativa delle due variabili in X e Y della cella stessa, rappresentata tramite scatterplot.



L'indice di correlazione di Pearson tra due variabili statistiche esprime una relazione di linearità tra esse.

Dato un insieme di due variabili statistiche X e Y, l'indice di correlazione di Pearson è definito come:

$$\rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}$$

$$-1 \leq \rho_{XY} \leq 1$$

Dove:

σ_{XY} è la covarianza tra X e Y, σ_X, σ_Y sono le deviazioni standard di X e Y

$\rho_{XY} > 0 \Rightarrow$ *correlazione positiva*

$\rho_{XY} = 0 \Rightarrow$ *nessuna correlazione*

$\rho_{XY} < 0 \Rightarrow$ *correlazione negativa*

DATA VISUALIZATION

La data visualization è utile anche per mostrare i risultati del modello a chi non capisce cosa si è fatto con le azioni di ML. Quando rappresentiamo i dati è necessario che vengono rappresentati in modo da evitare **misleading**, cioè che chi legge le rappresentazioni trae dai dati informazioni sbagliate.

Bisogna quindi usare delle rappresentazioni che siano utili a chi le osserva e noi a noi che le rappresentiamo.

La **data visualization**, o più precisamente **information visualization**, consiste nel rappresentare visivamente le informazioni che derivano dall'elaborazione dei dati grezzi. Non si tratta quindi solo di "dati" in senso stretto, ma della loro interpretazione e sintesi in forma visuale.

Questa attività è fondamentale nelle prime fasi di un progetto di **Business Intelligence**, perché consente di:

- Comprendere meglio i processi aziendali,
- Valutare la qualità dei dati a disposizione,
- Esplorare le proprietà delle variabili,
- Individuare relazioni significative tra i dati.

Oltre a essere uno strumento di analisi, la data visualization è anche un mezzo di **comunicazione**, soprattutto quando si devono presentare i risultati di modelli complessi, come quelli di **machine learning**, a un pubblico non tecnico.

È importante che i dati vengano rappresentati in modo chiaro e corretto, evitando ogni forma di **misleading**, cioè rappresentazioni che possono indurre in errore l'osservatore, portandolo a trarre conclusioni sbagliate. La scelta del tipo di grafico, dei colori, delle proporzioni e dell'ordine degli elementi deve essere guidata non solo dalla chiarezza, ma anche dall'efficacia comunicativa.

Un concetto chiave in questo ambito è quello del **Preattentive Processing**: si tratta della capacità del cervello umano di riconoscere immediatamente e inconsciamente, alcune caratteristiche visive come colore, forma, dimensione o posizione, ancora prima di focalizzare l'attenzione consapevole. Sfruttare questo meccanismo nella progettazione delle visualizzazioni permette di **guidare l'attenzione** dell'osservatore verso le informazioni più importanti, rendendo la visualizzazione più efficace e intuitiva. Esistono diverse caratteristiche visive che il cervello elabora in modalità preattentiva, tra cui:

- **Colore:** Un oggetto con un colore diverso dagli altri attira subito l'attenzione (es. un punto rosso in mezzo a punti grigi).
- **Forma:** Elementi con forme diverse si distinguono immediatamente all'interno di un insieme omogeneo.
- **Dimensione:** Un oggetto più grande o più piccolo rispetto agli altri viene subito percepito come diverso.
- **Orientamento:** Linee o oggetti inclinati si notano immediatamente tra elementi orientati in modo uniforme.
- **Movimento:** Oggetti in movimento risaltano rispetto a quelli statici.
- **Gruppi visivi:** Il cervello raggruppa automaticamente elementi vicini o simili, facilitando la lettura di dati complessi.

Una visualizzazione efficace dei dati deve facilitare la comprensione e la comunicazione delle informazioni. I seguenti principi aiutano a creare rappresentazioni chiare e significative:

1. **Conoscere il pubblico:** Adatta il livello di dettaglio e la tipologia di visualizzazione alle esigenze degli utenti.
2. **Semplicità e chiarezza:** Evita elementi superflui che possano distrarre. Un design pulito migliora l'interpretazione.
3. **Scelta del grafico appropriato:**
4. **Uso efficace dei colori:** Evidenzia le informazioni importanti senza eccessi. Assicurati che la combinazione cromatica sia accessibile a tutti.
5. **Fornire contesto:** Titoli, etichette e annotazioni migliorano la comprensione e l'interpretazione dei dati.
6. **Evita la distorsione dei dati:** Rappresenta le informazioni in modo accurato per non generare interpretazioni errate.
7. **Minimizza il "rumore" visivo:** Riduci gli elementi decorativi inutili che potrebbero confondere l'utente.

La rappresentazione dei dati deve considerare diversi fattori:

- **Adeguatezza al tipo di dato:** La scelta della visualizzazione deve rispettare la natura degli attributi (categorici, numerici, binari, ecc.).
- **Struttura dei dati:** Dati tabellari, transazionali o spaziotemporali richiedono modelli di rappresentazione specifici.
- **Obiettivi dell'analisi:** La rappresentazione varia a seconda dello scopo (classificazione, analisi di associazioni, data warehousing, ecc.).

È utile creare grafici interattivi che permettono agli utenti di interagire con le visualizzazioni per esplorare i dati in modo più approfondito.

DATA QUALITY

La **data quality** può essere definita come il grado in cui i dati sono idonei per il loro uso previsto. Dati di alta qualità sono essenziali per ottenere risultati accurati ed efficaci dai sistemi di Business Intelligence e dai modelli decisionali. Il principio **garbage in garbage out (GIGO)** sottolinea che dati inaccurati o di scarsa qualità portano inevitabilmente a informazioni e decisioni errate.

- **Accuratezza:** Il grado di conformità di una misurazione a uno standard o a un valore vero. I dati devono essere corretti e rappresentare ciò che era inteso dalla fonte originale.

- **Completezza:** La misura in cui tutti i dati richiesti sono conosciuti, rispetto a profondità, ampiezza e portata. I dati non dovrebbero includere un numero elevato di valori mancanti per evitare di compromettere l'accuratezza delle analisi.
- **Coerenza:** Il grado in cui un insieme di dati è equivalente in database ridondanti o distribuiti. La forma e il contenuto dei dati devono essere coerenti tra diverse fonti dopo le procedure di integrazione.
- **Tempestività:** I dati devono essere disponibili in tempo utile e rappresentare lo stato attuale del business. I dati "datati" possono portare a modelli e pattern obsoleti.
- **Rilevanza:** La misura in cui i dati sono utili nel contesto previsto e soddisfano le esigenze del sistema di Business Intelligence per aggiungere valore reale alle analisi.
- **Interpretabilità:** Il significato dei dati dovrebbe essere ben compreso e correttamente interpretato dagli analisti, anche grazie alla documentazione disponibile nei metadati.
- **Accessibilità:** I dati devono essere **facilmente accessibili** dagli analisti e dalle applicazioni di supporto decisionale.
- **Affidabilità:** Una caratteristica dell'infrastruttura informativa per archiviare e recuperare informazioni in modo accessibile, sicuro, manutenibile e veloce.
- **Non-ridondanza:** La ripetizione e la ridondanza dei dati dovrebbero essere evitate per prevenire sprechi di memoria e possibili incongruenze, a meno che la denormalizzazione non migliori i tempi di risposta.
- **Validità:** La corrispondenza tra i valori effettivi e attesi di una data variabile, in base a definizioni di dati accettabili.
- **Ricchezza:** Tutti gli elementi di dati richiesti sono inclusi nel set di dati, fornendo una rappresentazione sufficientemente dimensionale del soggetto.
- **Veracità:** (Specifico per Big Data) Conformità ai fatti: accuratezza, qualità, veridicità o affidabilità dei dati.

Le fonti di scarsa qualità dei dati, spesso definite come **rumore**, possono essere molteplici:

- **Fonti di informazione inaffidabili, dispositivi di misurazione scadenti, errori di battitura, confusione dell'utente e molte altre ragioni.**
- **Rumore stocastico:** Variazioni naturali o errori umani occasionali.
- **Rumore sistematico:** Errori che trascinano tutti i valori nella stessa direzione (es. termometro mal calibrato).

- **Errori di misurazione e di raccolta dati:** Problemi derivanti dal processo di misurazione stesso (es. valore registrato diverso dal valore vero) o errori nell'omissione/inclusione inappropriata di dati.
- **Valori mancanti:** Dati non registrati, non disponibili o rimossi.
- **Valori errati o outlier:** Dati insoliti dovuti a malfunzionamenti di dispositivi, errori di registrazione/trasmissione o vere anomalie.
- **Incoerenze:** Discrepanze dovute a cambiamenti nei sistemi di codifica o alla presenza di dati espressi in unità di misura eterogenee.
- **Dati duplicati:** Record multipli che si riferiscono alla stessa entità.

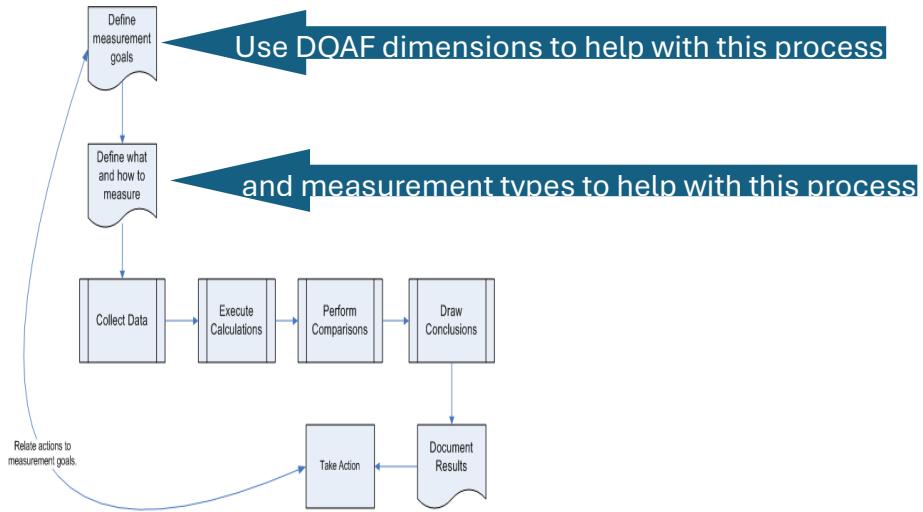
Il miglioramento della qualità dei dati è un processo continuo chiamato spesso **data cleaning** o **data scrubbing**. Il **Data Quality Assessment Framework (DQAF)** è un modello sviluppato dal Fondo Monetario Internazionale (FMI) per valutare la qualità dei dati statistici valutando la qualità dei dati in diversi contesti, come le statistiche economiche, finanziarie e sociali.

Il **Data Enrichment** è il processo di migliorare un set di dati esistente integrandolo con informazioni esterne, al fine di aggiungere valore e approfondire la comprensione. Questo processo può includere l'aggiunta di nuovi dati, la creazione di nuove variabili o l'aggiornamento di dati esistenti, con l'obiettivo di ottenere un quadro più completo o preciso per analisi, decisioni aziendali o modelli predittivi.

Il framework si basa su **cinque dimensioni fondamentali** per la valutazione della qualità dei dati:

1. **Integrità** – I dati devono essere prodotti senza influenze politiche o economiche e con procedure trasparenti.
2. **Metodologia solida** – Devono essere raccolti seguendo standard internazionali e tecniche statistiche riconosciute.
3. **Accuratezza e affidabilità** – I dati devono essere precisi, aggiornati e basati su fonti attendibili.
4. **Accessibilità e trasparenza** – Le informazioni devono essere facilmente reperibili e spiegate chiaramente agli utenti.
5. **Coerenza e comparabilità** – I dati devono essere confrontabili nel tempo e tra diversi paesi o settori.

Una **dimensione della qualità dei dati** è una categoria generale e misurabile per una caratteristica distintiva (qualità) posseduta dai dati. Permettono di comprendere la qualità in relazione a una scala e in relazione ad altri dati misurati sulla stessa scala.



AZIONI DI DATA CLEANING

- **Validazione dei dati:** Identificare e rimuovere anomalie e incongruenze. Ciò può includere la verifica che i valori siano all'interno di intervalli ammissibili e che i codici siano corretti.
- **Gestione dei valori mancanti:** Tecniche come l'eliminazione di record con valori mancanti o l'imputazione (sostituzione con valori probabili).
- **Identificazione e gestione degli outlier:** Rilevare e trattare valori anomali, che possono essere errori o reali eventi rari.
- **Correzione delle incongruenze:** Risolvere discrepanze dovute a diversi sistemi di codifica o unità di misura.
- **Eliminazione dei dati duplicati:** Identificare e unire o rimuovere record che rappresentano la stessa entità.
- **Standardizzazione e trasformazione dei dati:** Convertire i dati in formati omogenei e applicare trasformazioni (es. normalizzazione, discretizzazione, aggregazione) per migliorare l'accuratezza e l'efficienza degli algoritmi di apprendimento.
- **Integrazione dei dati:** Combinare dati provenienti da diverse fonti, risolvendo conflitti a livello di schema e di dati (data fusion).
- **Data profiling:** Ottenere una panoramica dei dati e dei possibili problemi attraverso analisi statistiche o basate su pattern.

CLASS IMBALANCE PROBLEM

Il **problema dello squilibrio di class** si verifica quando le classi in un dataset di classificazione non sono rappresentate in modo uguale, una o più classi (la classe minoritaria) hanno significativamente meno istanze rispetto ad altre classi (la classe maggioritaria).

! Un classificatore addestrato su un set di dati sbilanciato può mostrare una propensione a migliorare le prestazioni sulla classe maggioritaria

Gli algoritmi spesso mirano a migliorare l'accuratezza complessiva, che può essere raggiunta classificando la maggior parte delle istanze come appartenenti alla classe maggioritaria. Di conseguenza, le istanze della classe minoritaria possono essere frequentemente classificate in modo errato.

Esistono diverse strategie per affrontare il problema dello squilibrio di classe:

Sovracampionamento (Oversampling): Questa tecnica mira ad **aumentare il numero di istanze nella classe minoritaria**. Un approccio comune è la **replicazione casuale** di istanze della classe minoritaria. Tecniche più sofisticate includono la generazione di nuove istanze sintetiche per la classe minoritaria, come la tecnica **SMOTE (Synthetic Minority Over-sampling Technique)**. SMOTE crea nuove istanze positive in punti intermedi lungo i segmenti di linea che uniscono un'istanza positiva a uno dei suoi k-vicini più prossimi scelti casualmente.

Sottocampionamento (Undersampling): Questa tecnica mira a **ridurre il numero di istanze nella classe maggioritaria**. Ciò può essere fatto **rimuovendo casualmente** istanze dalla classe maggioritaria. Tuttavia, la rimozione di istanze può comportare la perdita di informazioni preziose.

SMOTE Synthetic Minority Over-sampling Technique una tecnica di sovracampionamento che genera esempi sintetici della classe minoritaria invece di duplicare semplicemente le istanze esistenti.

1. SMOTE sceglie casualmente un punto dalla classe minoritaria.
2. Trova i vicini più vicini dove vengono individuate le istanze della classe minoritaria più vicine a quella selezionata.
3. Si sceglie uno di questi vicini e si genera un punto sintetico sulla linea che collega entrambi i punti, a una distanza casuale.

Questo processo viene ripetuto finché non viene raggiunto l'equilibrio desiderato tra le classi.



STEP3 – Feature engineering



La **feature engineering** è il processo di selezionare, modificare, creare e trasformare le variabili (**feature**) di un dataset con l'obiettivo di migliorarne la qualità e la rilevanza per il modello di machine learning.

Consiste nel convertire il contesto in un input comprensibile al modello, capendo le relazioni che ci sono fra le feature. Analizzando le feature è possibile già farsi una idea del modello da usare.

Possiamo distinguere due categorie:

- **Feature Creation/Extraction** che consiste nella generazione di nuove variabili a partire da quelle originali. La **feature extraction** consiste nella creazione di nuove variabili che riassumono le informazioni contenute in un sottoinsieme degli attributi
- **Feature Selection/Reduction** si concentra sull'identificazione e la scelta del sottoinsieme più utile di variabili esistenti per un modello. L'obiettivo è ridurre la dimensionalità dei dati, semplificare i modelli, migliorarne la generalizzazione e ridurre il rumore.

IMPUTAZIONE

Questa tecnica si usa quando il dataset è una matrice sparsa, questo vuole dire quando mancano molti dati.

Per renderlo più completo e corretto, possiamo inserire il valore medio dei dati, dove questi mancano (**regression imputation**) se abbiamo una certa varianza fra i dati. Se non abbiamo abbastanza informazioni possiamo eliminare quella feature oppure usare SMOTE, andando ad **aumenta la varietà dei dati minoritari** perché creiamo datapoint artificiali. Con SMOTE si riduce il rischio di overfitting che può derivare dalla duplicazione dei dati esistenti e **migliora le prestazioni del modello** perché il modello diventa più sensibile alla classe minoritaria, migliorando l'accuratezza, la precisione e il recall per quella classe.

HANDLING OUTLIERS

Gli **outliers** sono osservazioni che si discostano significativamente dal resto dei dati. Questi sono problematici perché **influenzano i modelli di regressione**: distorcendo i risultati, in quanto questi modelli cercano di adattarsi a tutte le osservazioni, inclusi gli outliers.

CATEGORICAL ENCODING

Gli attributi contenuti in un set di dati possono essere categorizzati come categoriali o numerici, a seconda del tipo di valori che assumono.

Categoriali Gli attributi categoriali assumono un numero finito di valori distinti, nella maggior parte dei casi limitati a meno di cento, che rappresentano una proprietà qualitativa di un'entità a cui si riferiscono.

Numerici. Gli attributi numerici assumono un numero finito o infinito di valori e si prestano a operazioni di sottrazione o divisione.

Le features categoriche non possono essere utilizzate direttamente in forma testuale perché i modelli di machine learning si basano su operazioni matematiche e statistiche, e trattare le features categoriche come stringhe di testo potrebbe portare a risultati inaccurati o inefficaci. È necessario trasformare le features categoriche in un formato numerico o binario che possa essere interpretato correttamente.

Il processo di **Label Encoding** sostituisce ogni categoria di una variabile con un numero intero unico. Ogni categoria viene associata a un intero. Questa tecnica conviene usarla quando le categorie hanno un ordine naturale, altrimenti conviene usare lo **One-Hot Encoding** trasforma ogni categoria di una variabile in una **colonna separata** (caratteristica), assegnando un valore binario (0 o 1) a ciascuna colonna. Ogni categoria della variabile originale viene rappresentata come un **vettore** con un singolo valore 1 (indicante che quella categoria è presente) e 0 nelle altre posizioni.



NORMALIZZAZIONE/STANDARDIZZAZIONE

La **normalizzazione/standardizzazione** dei dati sono tecniche di trasformazione dei dati cruciali nella fase di preparazione dei dati per il business intelligence e il data mining. L'obiettivo di queste tecniche è quello di modificare la scala o la distribuzione delle variabili numeriche per renderle più adatte agli algoritmi di apprendimento e migliorare le loro prestazioni. La standardizzazione si riferisce a diverse tecniche per ridimensionare i valori degli attributi numerici in un intervallo specifico o a farli avere determinate proprietà statistiche.

Assicura che attributi con scale diverse non influenzino in modo sproporzionato il risultato di algoritmi basati sulla distanza, come il clustering o i k-nearest neighbors.

La scelta della tecnica di standardizzazione o normalizzazione dipende dalle caratteristiche dei dati e dai requisiti specifici dell'algoritmo che verrà utilizzato. È importante applicare la stessa trasformazione ai dati di test o ai nuovi dati che è stata applicata ai dati di training per garantire la coerenza del modello.

PRINCIPAL COMPONENT ANALYSIS

L'**analisi delle componenti principali (PCA)** è una tecnica di riduzione degli attributi mediante proiezione. Lo scopo di questo metodo è ottenere una trasformazione proiettiva che sostituisca un sottoinsieme degli attributi numerici originali con un numero inferiore di nuovi attributi ottenuti come loro combinazione lineare, senza che questa modifica causi una perdita di informazioni. Una trasformazione degli attributi può portare in molti casi a una migliore accuratezza nei modelli di apprendimento sviluppati successivamente.

L'obiettivo della PCA è quello di **ridurre la dimensionalità dei dati** mantenendo la maggior parte della **varianza** presente nel dataset originale. Questo viene fatto identificando le direzioni (o componenti principali) in cui i dati variano maggiormente.

1. Standardizzare i dati, in modo da ottenere per tutti gli attributi lo stesso intervallo di valori, solitamente rappresentato dall'intervallo $[-1, 1]$.
La PCA cerca di trovare un nuovo insieme di variabili, chiamate **componenti principali**, che sono combinazioni lineari delle variabili originali.
2. La prima componente principale è scelta in modo da spiegare la **massima quantità di varianza** nei dati.
3. Le successive componenti principali sono scelte in modo da spiegare la massima varianza rimanente, con il vincolo di essere **non correlate (ortogonali)** alle componenti precedenti.
 1. Le componenti principali sono ordinate per importanza in base alla quantità di varianza che spiegano.
 2. Per ridurre la dimensionalità, si selezionano solo le **prime k componenti principali** (dove k è inferiore al numero di variabili originali) che catturano una frazione significativa della varianza totale. I dati originali possono quindi essere approssimati utilizzando queste k componenti.

L'applicazione della PCA a un dataset produce un nuovo insieme di variabili non correlate.

FEATURE SELECTION

La **feature selection** è quello di eliminare dal dataset un sottoinsieme di variabili che non sono considerate rilevanti ai fini delle attività di data mining. Grazie alla presenza di meno colonne, gli algoritmi di apprendimento possono essere eseguiti più rapidamente sul dataset ridotto rispetto a quello originale.

Metodi filtro I metodi filtro selezionano gli attributi rilevanti prima di passare alla successiva fase di apprendimento e sono quindi indipendenti dall'algoritmo specifico utilizzato. Sono state proposte diverse metriche statistiche alternative per valutare la capacità predittiva e la rilevanza di un gruppo di attributi.

Un esempio può essere quello della **mutua informazione** che misura la dipendenza statistica tra due variabili quantificando la quantità di informazioni che una variabile contiene sull'altra.

$$MI(X, Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log\left(\frac{p(x, y)}{p(x)p(y)}\right)$$

Metodi wrapper I metodi wrapper sono in grado di valutare **l'accuracy**, poiché valutano un gruppo di variabili utilizzando lo stesso algoritmo di classificazione o regressione utilizzato per prevedere il valore della variabile target. Ogni volta, l'algoritmo utilizza un diverso sottoinsieme di attributi per l'apprendimento, identificato da un motore di ricerca che lavora sull'intero set di tutte le possibili combinazioni di variabili e seleziona il set di attributi che garantisce il miglior risultato in termini di accuratezza. Questi algoritmi hanno un costo computazionale molto elevato, infatti hanno un approccio greedy.

Metodi embedded Per i metodi embedded, il processo di selezione degli attributi si trova all'interno dell'algoritmo di apprendimento, in modo che la selezione del set ottimale di attributi venga effettuata direttamente durante la fase di generazione del modello.

TRASFORMATION

La **transformation** consiste nell'applicare una funzione o un'operazione ai valori degli attributi esistenti per crearne di nuovi o per modificare la loro rappresentazione, con l'obiettivo di cambiare la rappresentazione dei dati, in uno spazio più comodo da modellare.

Le trasformazioni delle variabili vengono applicate per diverse **facilitare l'interpretazione dei dati** e dei modelli nuovi che possono catturare trend o relazioni sottostanti nei dati in modo più esplicito.

Ci sono varie tecniche con cui si può fare trasformazione dei dati che dipendono dal tipo di dato e dal problema che stiamo analizzando.

Single Threshold tecnica utilizzata per **binarizzare** i dati numerici, ovvero per trasformare una variabile continua in una variabile categoriale con due possibili valori (0 o 1), utilizzando una singola soglia. L'idea alla base di questa tecnica è piuttosto semplice: si stabilisce un valore di soglia e, a partire da questo, si decide a quale classe appartiene un dato. Se il valore della variabile è maggiore o uguale alla soglia, viene assegnato un valore di 1, altrimenti viene assegnato 0.

Set of Thresholds si applica a un contesto in cui i dati vengono suddivisi in **più classi** utilizzando più di una soglia. In altre parole, invece di avere una sola soglia che separa due categorie, si impiegano più soglie per creare diverse classi. Questo approccio permette di **dividere i dati in più gruppi**, andando a definire un intervallo di valori per ciascuna classe.

→ **problemi di classificazione multi-classe** in cui la variabile dipendente ha più di due possibili valori.



STEP4 – Addestramento del modello

Un modello di Machine Learning è una rappresentazione matematica di un problema che la macchina utilizza per fare previsioni o prendere decisioni basate sui dati.

I modelli di Machine Learning possono essere classificati in base al tipo di apprendimento.



Supervised learning è una modalità di apprendimento con la quale la macchina impara da un dataset e di etichettato cioè i dati in input sono associati a un altro. Questo dataset è usato per creare il modello e le regole per fare previsioni. È possibile risolvere problemi di classificazione o di regressione nel caso di previsione numerica.

Unsupervised learning è una modalità di apprendimento in cui invece si usano dati non etichettati, quindi l'algoritmo con i suoi dati di input deve capire i pattern che esistono. Produce un modello descrittivo piuttosto che un modello predittivo e si basano sulla definizione di una metrica per calcolare la distanza fra i dati di input e l'algoritmo userà questa distanza per raggruppare in base alla distanza che ne definisce la somiglianza, permettono quindi di fare clustering e di ridurre la dimensionalità dei dati.

Reinforcement learning è una modalità di apprendimento basato sull'interazione con l'ambiente e di un agente che ogni volta che compilazione riceva una ricompensa o una penalità che migliora la qualità della previsione. Ciò che si cerca di fare è di massimizzare la ricompensa totale nel lungo termine sono usati nell'ottimizzazione delle decisioni e per fare simulazioni

La differenza fra le modalità di apprendimento dipende dai dati che abbiamo a disposizione e dal problema che dobbiamo risolvere.

Nel machine learning, esistono due tipi di parametri:

- **Parametri calcolati dal modello:** Questi parametri sono calcolati dallo stimatore durante il processo di apprendimento dai dati forniti.
- **Iperparametri:** Questi parametri vengono specificati in anticipo quando si crea il modello. Gli iperparametri sono parametri che non sono appresi direttamente dai dati, ma sono impostati dall'utente prima che il modello venga addestrato.

Si dovrà sperimentare con diversi valori degli iperparametri per produrre il migliore modello possibile per lo studio in analisi. Questo processo è chiamato **regolazione dell'iperparametro**.

Prima di addestrare il modello conviene dividere i dati del dataset in **training set** e **testing set**.

In generale il 75% dei dati è per il training in modo che possano essere coperti tutti i casi possibili, mentre il 25% è usato per il test e verificare che l'algoritmo funzioni correttamente. Questo dipende dalla quantità di dati che abbiamo.

Se abbiamo pochi dati conviene invece usare il **K-Fold Cross Validation** usato anche per la selezione degli iperparametri.

Nella k-fold cross-validation, l'insieme di dati etichettati D (di dimensione N) viene diviso casualmente in k partizioni (o fold) di dimensioni. La procedura viene eseguita k volte (le **run**). In ogni run i:

1. Una delle k partizioni $D_{\text{test}}^{(i)}$ viene utilizzata come set di test.
2. Le restanti $k-1$ partizioni $D_{\text{train}}^{(i)}$ vengono utilizzate come set di training per addestrare un modello $m^{(i)}$.
3. Il modello $m^{(i)}$ viene applicato al set di test $D_{\text{test}}^{(i)}$ per ottenere una stima dell'errore (o accuratezza) della generalizzazione $\text{err}^{(i)}$.
L'errore di test complessivo err_{test}
4. viene quindi calcolato come la media degli errori ottenuti in ciascuna delle k run:
$$\text{err}_{\text{test}} = \frac{\sum_{i=1}^k \text{err}^{(i)}}{k}$$

Ogni istanza del dataset viene utilizzata esattamente una volta per il test e $k-1$ volte per l'addestramento. Questo rende la k-fold cross-validation utile quando la quantità di dati è limitata. La media delle prestazioni su diverse partizioni fornisce una stima più affidabile di come il modello si comporterà su dati non visti rispetto a una singola divisione train-test (holdout method).

Eseguendo k run, si ottengono k stime delle prestazioni, consentendo di valutare la variabilità della performance del modello in diverse configurazioni di dati di training e test.

La scelta del valore di k può influenzare i risultati. Valori comuni per k sono 5 e 10. Un piccolo valore di k può portare a un set di training più piccolo in ogni run, risultando in una stima più pessimistica dell'errore di generalizzazione. Un valore elevato di k (fino a N, noto come **leave-one-out**) utilizza quasi tutti i dati per l'addestramento in ogni run, riducendo la distorsione nella stima dell'errore, ma può essere computazionalmente costoso e può produrre risultati fuorvianti in alcuni casi.

BIAS

Il **bias** in un modello di machine learning è una tendenza sistematica dell'algoritmo a commettere errori in una certa direzione, spesso dovuta ad assunzioni semplificative fatte durante l'apprendimento o alla presenza di dati distorti. È importante distinguere i vari tipi di bias a seconda del contesto per comprendere il ruolo e l'impatto sul model.

Bias Induttivo Il bias induttivo è l'insieme di ipotesi aggiuntive che un algoritmo di apprendimento fa per giustificare le sue inferenze induttive, come se fossero inferenze deduttive basate sui dati di addestramento e sulla nuova istanza.

non c'è apprendimento senza un bias, l'apprendimento induttivo richiede sempre un bias per generalizzare.

L'apprendimento "bias-free" (senza bias) è inutile. L'ingegnere deve definire lo spazio delle ipotesi con un bias significativo affinché il processo di machine learning abbia successo. Esistono diversi tipi di bias induttivo:

- **Restriction Bias:** Una restrizione categorica sull'insieme delle ipotesi considerate, che limita lo spazio delle ipotesi. Un esempio è il bias dell'algoritmo CANDIDATE-ELIMINATION, la cui ipotesi è che il concetto target sia contenuto nello spazio di ipotesi dato H. Questo tipo di bias comporta il rischio di escludere del tutto la funzione target sconosciuta.
- **Preference Bias:** Una preferenza per certe ipotesi rispetto ad altre (ad esempio, per ipotesi più semplici o brevi) all'interno di uno spazio di ipotesi potenzialmente completo. L'algoritmo ID3 esibisce un bias puramente di preferenza. Generalmente, un bias di preferenza è più desiderabile di un bias di restrizione.

Bias Term Questo è un parametro specifico in alcuni modelli, in particolare nei modelli lineari come i classificatori lineari. Nel caso degli iperpiani (usati dai classificatori lineari), i coefficienti w_1, \dots, w_n definiscono l'angolo dell'iperpiano, mentre l'ultimo coefficiente, w_0 , chiamato **bias**, determina lo scostamento (offset) dell'iperpiano dall'origine del sistema di coordinate. Un valore più alto del bias sposta il classificatore più lontano dall'origine, mentre un bias di 0 lo fa intersecare con l'origine.

Bias da Dati Sbilanciati: Si verifica quando un modello viene addestrato su un set di dati in cui una classe è significativamente più rappresentata di altre. Avere dati sbilanciati ha un impatto negativo sulle prestazioni del modello. C'è il rischio di addestrare un modello distorto verso la previsione della classe di maggioranza. Ciò può portare a prestazioni inferiori su metriche come il recall

per la classe di minoranza, anche se l'accuratezza generale potrebbe sembrare accettabile.

SUPERVISED ML: modello di classificazione

I dati per un'attività di classificazione sono costituiti da una raccolta di istanze. Ogni istanza di questo tipo è caratterizzata dalla tupla (x, y) , dove x è l'insieme di valori di attributo che descrivono l'istanza e y è l'etichetta di classe dell'istanza. L'insieme di attributi x può contenere attributi di qualsiasi tipo, mentre l'etichetta di classe y deve essere categoriale. Un modello di classificazione è una rappresentazione astratta della relazione tra l'insieme di attributi e l'etichetta di classe.

Non tutti gli attributi sono rilevanti per la classificazione, quindi possono essere scartati. Gli attributi rimanenti potrebbero non essere in grado di distinguere le classi da soli e devono essere utilizzati insieme ad altri attributi.

Viene utilizzato come modello predittivo per classificare istanze precedentemente non etichettate. Funge anche da modello descrittivo per identificare le caratteristiche che distinguono le istanze da classi diverse.

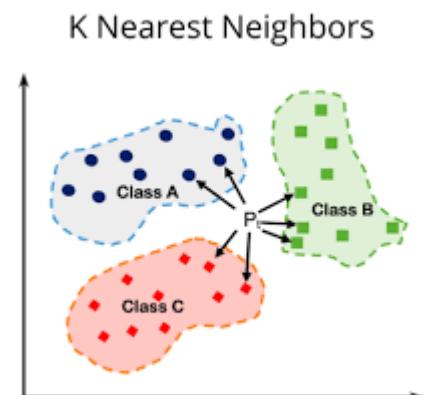
Il modello viene creato utilizzando un set di istanze, noto come set di addestramento, che contiene valori di attributo e etichette di classe per ogni istanza. Il processo di utilizzo di un algoritmo di apprendimento per costruire un modello di classificazione dai dati di addestramento è noto come **induzione**. E porta alla costruzione di un modello. L'applicazione di un modello di classificazione su istanze di test invisibili per prevedere le loro etichette di classe è nota come **deduzione**.

ALGORITMO ML: k-Nearest Neighbors

Un **k-Nearest Neighbors** rappresenta ogni campione come un punto dati in uno spazio d-dimensionale, dove d è il numero di attributi.

Data un'istanza di test, calcoliamo la sua prossimità alle istanze di training. I vicini più prossimi di k di una data istanza di test z si riferiscono ai k esempi di training più vicini a z .

La discussione precedente sottolinea l'importanza di scegliere il valore corretto per k . Se k è troppo piccolo, allora il classificatore del vicino più prossimo potrebbe essere suscettibile di overfitting a causa del rumore, ovvero esempi etichettati in modo



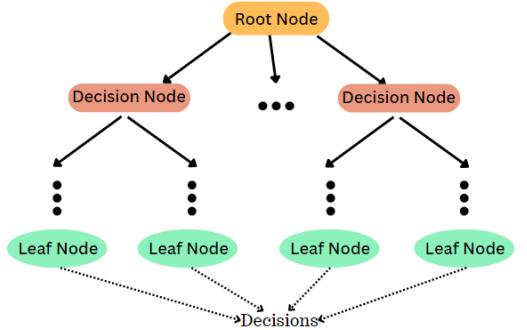
errato nei dati di training. D'altro canto, se k è troppo grande, il classificatore del vicino più prossimo potrebbe classificare in modo errato l'istanza di test perché il suo elenco di vicini più prossimi include esempi di training che si trovano molto lontano dal suo vicinato.

ALGORITMO ML: Decision tree learning

I **decision tree** modellano i processi decisionali, specialmente quelli induttivi, in una struttura ad albero in cui:

- ogni **nodo interno** dell'albero rappresenta un test sul valore di un attributo dell'istanza.
- ogni **ramo** che discende da un nodo corrisponde a uno dei possibili valori per quell'attributo.
- ogni **nodo foglia** rappresenta il risultato.

Un attributo è classificato partendo dal nodo radice e muovendosi fra i vari rami, ma mano che passa le condizioni, fino ad arrivare ai nodi foglia che rappresentano la decisione. Gli alberi decisionali sono facili da visualizzare e interpretare, il che li rende uno strumento eccellente per spiegare la logica alla base delle previsioni.



Per costruire un albero decisionale dai dati di addestramento si applica un processo chiamato **induzione**. Gli alberi decisionali rappresentano una disgiunzione di congiunzioni dei valori dei vari attributi. Molti algoritmi per l'apprendimento degli alberi decisionali sono variazioni di un algoritmo di base che impiega una **ricerca top-down e greedy** attraverso lo spazio dei possibili alberi decisionali.

COSTRUZIONE DELL'ALBERO

Le osservazioni del training set inizialmente contenute nel nodo radice dell'albero vengono suddivise in sottoinsiemi disgiunti che vengono provvisoriamente collocati in due o più nodi discendenti (ramificazione). A ciascuno dei nodi così determinati, viene applicato un controllo per verificare se sono soddisfatte le **condizioni per l'arresto** dello sviluppo del nodo. Se almeno una di queste condizioni è soddisfatta, non viene eseguita alcuna ulteriore suddivisione e il nodo diventa una foglia dell'albero. In caso contrario, viene eseguita la suddivisione effettiva delle osservazioni contenute nel nodo. Al termine della procedura, quando nessun nodo dell'albero può essere ulteriormente suddiviso, ciascun nodo foglia viene etichettato con il valore della classe a cui appartiene la maggior parte delle osservazioni nel nodo, secondo un criterio chiamato **voto di maggioranza**. La suddivisione degli esempi in

ciascun nodo viene effettuata mediante una **regola di suddivisione**, da selezionare in base a una specifica funzione di valutazione. Una caratteristica che descrive gli alberi decisionali è lo **spazio delle ipotesi** che rappresenta l'insieme di tutti i modelli che possono essere generati a partire dai dati e dalla struttura dell'albero stesso. Lo spazio è **enormemente grande**, anche con un numero moderato di attributi e valori, perché ogni albero è una combinazione di condizioni (test su attributi) e risultati (classi o valori).

ID3-FUNZIONAMENTO

Il processo di induzione inizia con un singolo nodo radice associato a tutti gli esempi di addestramento. L'algoritmo sceglie l'attributo migliore su cui eseguire il test al nodo radice. Questa scelta si basa tipicamente su un **criterio preimpostato** o su una **misura di selezione dell'attributo** che quantifica quanto bene l'attributo da solo classifica gli esempi di addestramento. Un meccanismo comune è l'**information gain**, calcolato tramite l'**entropia dell'informazione** o il **Gini index**.

Le misure di selezione degli attributi cercano di dare preferenza alle condizioni di test che partizionano le istanze di addestramento in sottoinsiemi "più puri" nei nodi figlio (sottoinsiemi che contengono prevalentemente istanze della stessa classe).

Una volta selezionato l'attributo migliore, viene creato un nodo discendente per ogni possibile valore di questo attributo, e gli esempi di addestramento vengono distribuiti al nodo discendente appropriato.

L'intero processo viene quindi ripetuto ricorsivamente per ogni nodo discendente che non soddisfa i **criteri di arresto** che possono includere il raggiungimento di un nodo in cui tutte le istanze appartengono alla stessa classe o quando un nodo è dominato da un'unica etichetta di classe.

Da un insieme di addestramento, possono essere creati molti alberi decisionali alternativi. In genere, gli alberi più piccoli sono da preferire perché **rimuovono gli attributi irrilevanti e ridondanti** e hanno meno rischio di overfitting. Un albero grande vuol dire che si è legato troppo ai dati e quindi non è in grado di dedurre, di generalizzare

Un albero decisionale rappresenta una **disgiunzione di congiunzioni** di vincoli sui valori degli attributi. Ogni percorso dalla radice a una foglia corrisponde a una congiunzione di test sugli attributi (una regola "if-then"), e l'albero stesso corrisponde a una disgiunzione di queste congiunzioni.

ID3(Examples, Target_attribute, Attributes)

Examples are the training examples. *Target_attribute* is the attribute whose value is to be predicted by the tree. *Attributes* is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = -
- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
- Otherwise Begin
 - $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*
 - The decision attribute for *Root* $\leftarrow A$
 - For each possible value, v_i , of *A*,
 - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
 - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for *A*
 - If $Examples_{v_i}$ is empty
 - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
 - Else below this new branch add the subtree $ID3(Examples_{v_i}, Target_attribute, Attributes - \{A\})$
- End
- Return *Root*

* The best attribute is the one with highest *information gain*, as defined in Equation (3.4).

Per lo spazio delle risorse, ID3, esegue una ricerca **top-down** che va **dal semplice al complessa**, partendo dall'albero vuoto e considerando dopo ipotesi progressivamente più elaborate, nel tentativo di trovare un albero decisionale che classifichi correttamente i dati di addestramento. La funzione di valutazione che permette di fare questo è l'**information gain** che è una misura di quanto si riduce l'**entropia** del sistema se dividiamo i dati in base a un certo attributo.

L'**entropia** è una metrica per misurare l'impurità in un dato attributo. Specifica la casualità nei dati. In un albero decisionale, l'obiettivo è ridurre l'entropia del set di dati creando sottoinsiemi di dati più puri. Poiché l'entropia è una misura dell'impurità, diminuendo l'entropia, stiamo aumentando la purezza dei dati.

$$E = - \sum_{i=1}^N p_i \log_2 p_i$$

Pi è la probabilità di selezionare casualmente un esempio nella classe i. Il logaritmo dà un valore negativo, e quindi un segno '-' viene utilizzato nella formula dell'entropia per negare questi valori negativi.

L'**information gain** è la differenza tra l'entropia prima della divisione e l'entropia media dopo la divisione.

$$\text{Information Gain}(S, A) = \text{Entropia}(S) - \sum_{v \in \text{Valori}(A)} \frac{|S_v|}{|S|} \cdot \text{Entropia}(S_v)$$

dove:

- S è il dataset,
- A è la caratteristica usata per dividere,
- v sono i valori possibili della caratteristica A,
- S_v è il sottoinsieme di S dove $A=v$,
- $|S_v|$ è la dimensione del sottoinsieme,
- Si fa la media pesata delle entropie dei sottoinsiemi.

L'attributo che fornisce la maggiore riduzione dell'entropia (cioè più informativo) viene scelto come **nodo decisionale**.

ASSUNZIONI

- Lo **spazio delle decisioni è completo** (uno spazio in grado di esprimere qualsiasi funzione finita a valori discreti), evitando così di non contenere la funzione target. Nella costruzione dell'albero mantiene una solo ipotesi man mano che esplora lo spazio delle decisioni, ma in questo modo perde la possibilità di rappresentare ipotesi coerenti.
- Non esegue backtracking, quindi una volta considerato un certo attributo, non riconsidera questa scelta. **Converge a soluzioni localmente ottimali**
- Usa tutti i datapoint in ogni fase di ricerca per creare le ipotesi

Per poter generalizzare le osservazioni dl training set fa delle assunzioni sui dati mai visti, questo è l'**inductive bias**. Poiché i dati di training da soli non bastano per garantire una soluzione univoca, l'algoritmo deve “**scegliere**” tra più ipotesi. Il **bias** guida questa scelta.

Bias induttiva approssimativa di ID3: gli alberi più corti sono preferiti rispetto agli alberi più grandi.

Un'approssimazione più vicina al bias induttivo di ID3: gli alberi più corti sono preferiti a quelli più lunghi. Gli alberi che posizionano gli attributi ad alto guadagno informativo vicino alla radice sono preferiti a quelli che non lo fanno.

Spazio delle Ipotesi Completo ma Ricerca Incompleta: L'algoritmo ID3, che costruisce alberi decisionali in modo top-down, cerca in uno spazio di ipotesi che è **completamente espressivo** per funzioni a valori discreti finite. Ma l'algoritmo ID3 esegue una ricerca **incompleta** attraverso questo spazio. Si tratta di una ricerca avida (greedy) e top-down, che parte da alberi semplici (vicino alla radice) e considera ipotesi progressivamente più elaborate. Non effettua backtracking.

Il bias induttivo di ID3 non deriva da una restrizione dello spazio delle ipotesi (poiché è completo), ma dalla sua **strategia di ricerca**. Si tratta di un **bias di preferenza** (o search bias). L'algoritmo ID3 sceglie la prima ipotesi accettabile che incontra nella sua ricerca dal semplice al complesso. Che sono favoriti a quelli restricted perché permettono di lavorare anche senza avere lo spazio delle ipotesi completo. Un bias di restrizione che limita rigorosamente l'insieme delle potenziali ipotesi è generalmente meno desiderabile, perché introduce la possibilità di escludere del tutto la funzione target sconosciuta

Approssimativamente, questo bias favorisce **alberi più corti** rispetto a quelli più lunghi. L'algoritmo cresce l'albero solo quanto necessario per classificare gli esempi di addestramento disponibili.

Questo bias di preferenza per alberi più semplici è spesso riferito al principio del

Rasoio di Occam: Preferire l'ipotesi più semplice che si adatta ai dati. L'argomento a favore di questo principio è che un'ipotesi più semplice che si adatta ai dati di addestramento è meno probabile che sia una coincidenza statistica rispetto a un'ipotesi complessa che si adatta ai dati.

Corollary: the longer the hypothesis the higher the chance of **coincidental**ity;



Per come funziona l'algoritmo, l'albero cresce in profondità man mano che esplora lo spazio delle ipotesi e le feature del dataset. Questo è un problema nell'apprendimento degli alberi decisionali perché potrebbe **overfittare**, e quindi apprenderebbe troppo i dati compreso il rumore e quindi avrebbe minore capacità di generalizzazione. Esistono diversi approcci per evitare il sovradattamento nell'apprendimento degli alberi decisionali. Questi possono essere raggruppati in due classi:

1. approcci che interrompono la crescita dell'albero prima che raggiunga il punto in cui classifica perfettamente i dati di addestramento,

- approcci che consentono all'albero di sovra adattare i dati e quindi di potarlo successivamente.

La potatura è un modo per gestire la complessità del modello e quindi il bias induttivo, cercando di trovare un equilibrio che minimizzi l'errore di generalizzazione limitando la crescita dell'albero o di rimuovere rami per rendere l'albero più semplice. La potatura tende a dare risultati migliori rispetto alla pre-potatura perché le decisioni vengono prese dopo aver costruito l'albero completo.

Per evitare l'overfitting, si utilizzano diverse strategie, raggruppate in due classi:

- Interrompere la crescita dell'albero prima** che classifichi perfettamente i dati di addestramento (nota come **pre-pruning** o regola di arresto anticipato). Questo approccio utilizza una condizione di arresto più restrittiva durante l'induzione.
- Post-pruning** dopo che è stato completamente indotto dai dati. Il post-pruning prevede la sostituzione di un sottoalbero con un nodo foglia. La scelta di quale sottoalbero potare è guidata da un criterio, spesso basato sull'accuratezza su un **validation set**. Il pruning basato sull'errore ridotto pota iterativamente i nodi la cui rimozione aumenta maggiormente l'accuratezza sull'insieme di validazione.

Il pruning mira a semplificare l'albero, riducendo il numero di nodi, per migliorare le performance di generalizzazione.

Oppure un'altra tecnica è il **Minimum Description Length (MDL)** che afferma che:

La **migliore ipotesi** (o modello) è quella che **comprime al massimo i dati**, cioè quella che ha **la descrizione totale più breve**.

Il principio di MDL cerca l'albero TT che **minimizza** la somma:

$$\text{MDL}(T, D) = L(T) + L(D | T)$$

Dove:

- $L(T)$ è la lunghezza in bit per descrivere la **struttura dell'albero**
- $L(D|T)$ è la lunghezza in bit per descrivere gli **errori di classificazione** dell'albero sui dati D
- Se l'albero è troppo **complesso** $\rightarrow L(T)$ è alto (serve più informazione per descriverlo)
- Se l'albero è troppo **semplice** $\rightarrow L(D|T)$ è alto (commette più errori sui dati)

P Possiamo usare il **validation set** per prevenire l'**overfitting** con una tecnica chiamata **reduced-error pruning** (potatura a errore ridotto). Questa tecnica funziona così:

- Si considera ogni nodo dell'albero decisionale come possibile candidato da potare.
- Potare un nodo significa rimuovere tutto il sottoalbero che parte da quel nodo e trasformarlo in una foglia, assegnandogli la classe più frequente tra gli esempi di training associati.
- Un nodo viene potato solo se l'albero risultante non peggiora in termini di prestazioni sul validation set.
- Questo processo elimina nodi che sono stati creati per “cogliere” coincidenze casuali nel training, perché tali coincidenze difficilmente si ripetono nel validation set.
- I nodi vengono potati uno alla volta, scegliendo sempre quello la cui rimozione migliora maggiormente l'accuratezza sul validation set.
- La potatura continua finché non peggiora l'accuratezza sul validation set.

All'inizio, l'albero è grande e può avere bassa accuratezza su dati nuovi; potandolo, diventa più semplice e migliora la capacità di generalizzazione. Infine, i dati sono divisi in tre parti: training, validation (per la potatura) e test (per stimare l'accuratezza finale). Lo svantaggio principale è che, se i dati sono pochi, usare un validation set sottrae esempi preziosi al training.

ALGORITMO ML: Support Vector Machine

Le **Support Vector Machines** sono una famiglia di metodi di separazione per la classificazione e la regressione. Sono un modello di classificazione **discriminativo** che apprende confini decisionali **lineari o non lineari** nello spazio degli attributi per separare le classi. Si basano sulla teoria dell'apprendimento statistico e si basano sul **principio della minimizzazione del rischio strutturale**.

L'idea di base dietro le SVM per due classi è quella di utilizzare una funzione lineare nelle variabili di input, w , e definire la regola di classificazione in base al segno di $f(x)$. L'equazione di un **iperpiano separatore** generico può essere scritta come

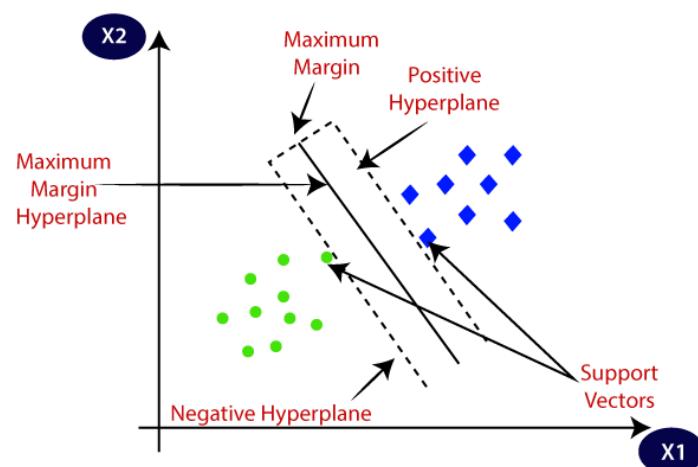
$$w^T x + b = 0.$$

$f(x) = \text{sign}(w^T x + b) \rightarrow$ decide dove va il prossimo datapoint nello spazio

La retta migliore sarà quella che effettuerà meno errori di classificazione e che quindi separa meglio i due semipiani. In un problema di classificazione binario,

trovandoci in uno spazio euclideo sarà sempre possibile trovare la retta che separa le classi, a patto che i datapoint siano linearmente separabili.

1. Iniziamo a trovare la retta migliore per i due semipiani per ridurre lo spazio su dove si potrebbe trovare l'iperpiano
2. Nello spazio formato dall'intersezione di queste due rette, scegliamo una retta a cazzum e avremo la certezza che qualunque retta prendiamo non peggiori il funzionamento del modello
3. Scelta la retta andiamo a definire gli **iperpiani di supporto** o **margini** che sono i vettori più vicini alle classi del dataset.



Le SVM cercano un **iperpiano separatore con il margine più ampio**.

L'iperpiano separatore ottimale è quello dotato della massima capacità di generalizzazione e del minimo errore empirico.

Anche se diversi iperpiani possono avere errore di addestramento pari a zero, quelli con margine più ampio tendono a fornire risultati migliori su istanze mai viste prima,

essendo più robusti a piccole perturbazioni. Massimizzare M è equivalente a minimizzare $\|w\|^2$.

Per un problema di classificazione binaria con istanze di addestramento (x_i, y_i) dove $y_i \in \{-1, 1\}$, le condizioni per una corretta separazione con margine M sono

$$y_i(w^T x_i + b) \geq M / \|w\|$$

Dobbiamo evitare però che queste rette inglobino alcuni datapoint, perché vorrebbe dire che si sovrapporrebbero alle classi.

Gestione di Dati Non Linearmente Separabili (Soft-Margin SVM)

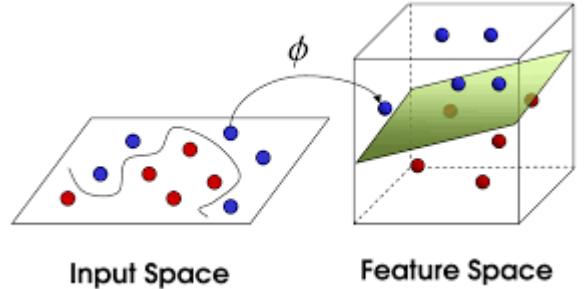
Le formulazioni descritte inizialmente costruiscono un confine decisionale lineare per separare gli esempi. Tuttavia, insiemi di esempi possono essere caratterizzati da un pattern intrinsecamente non lineare.

Per gestire dati non linearmente separabili, si introduce il concetto di **soft-margin**. Questo permette di **rilassare i vincoli** per accomodare alcune violazioni su un piccolo numero di istanze di addestramento introducendo **variabili di slack** $\xi_i \geq 0$ per ogni istanza. La variabile ξ_i è non nulla solo se l'iperpiano di margine non è in grado di posizionare x_i correttamente.

Il problema di ottimizzazione soft-margin cerca di bilanciare la massimizzazione del margine (minimizzando $\|w\|^2$) con la minimizzazione dell'errore di addestramento, penalizzando le violazioni. Questo bilanciamento è controllato da un iper-parametro C . Un valore elevato di C pone maggiore enfasi sulla minimizzazione dell'errore di addestramento. Questa formulazione fornisce un modo naturale per bilanciare la complessità del modello e l'errore di addestramento per massimizzare la performance di generalizzazione.

Kernel Trick

Un metodo per applicare le SVM a dati con confini decisionali non lineari è quello di **trasformare i dati** dallo spazio degli attributi originale (x) a un nuovo spazio delle feature ($\phi(x)$) in modo che un iperpiano lineare possa separare le istanze nello spazio trasformato. L'iperpiano appreso può poi essere proiettato nuovamente nello spazio degli attributi originali, risultando in un confine decisionale non lineare.



Questo approccio potrebbe sembrare soggetto alla "maledizione della dimensionalità" se la trasformazione aumenta la dimensionalità. Tuttavia, le SVM non lineari evitano questo problema utilizzando le **funzioni kernel**.

Una funzione kernel, $\kappa(x,y) = \langle \phi(x), \phi(y) \rangle$, permette di **calcolare i prodotti interni** tra vettori nello spazio trasformato $\langle \phi(x_i), \phi(x_j) \rangle$ **senza dover eseguire esplicitamente la trasformazione** $\phi(.)$.

Tipo di Kernel	Formula	Parametri
Lineare	$K(x, x') = x^T x'$	Nessuno
Polinomiale	$K(x, x') = (x^T x + c)^d$	$c = \text{costante}, d = \text{grado}$
RBF / Gaussiano	$K(x, x') = \exp(-\gamma \ x - x'\ ^2)$	$\gamma = \text{parametro di scala}$
Sigmoide	$K(x, x') = \tanh(ax^T x' + c)$	$a, c = \text{parametri da regolare}$

DATA MINING: modello associativo

Le **regole associative**, sono utilizzate per **identificare associazioni interessanti e ricorrenti tra gruppi di record** in un set di dati di grandi dimensioni. Sono una tecnica per **scoprire relazioni tra variabili in grandi database**. L'obiettivo è identificare modelli e ricorrenze regolari all'interno di un ampio set di transazioni.



Una regola è un'implicazione del tipo $Y \Rightarrow Z$, con il significato "se Y è vero, allora Z è anche vero".

Le regole associative sono spesso rappresentate nella forma $X \rightarrow Y$.

X è chiamato **antecedente (body)** della regola. Contiene una congiunzione di condizioni di test sugli attributi.

Y è chiamato **conseguente (head)** della regola. Contiene l'etichetta di classe prevista o, nel contesto delle associazioni di itemset, l'itemset associato.

X e Y sono **itemset** e devono essere **disgiunti ($X \cap Y = \emptyset$)**. Un itemset è un sottoinsieme di un insieme di oggetti $O = \{o_1, o_2, \dots, o_n\}$.

Una **transazione** rappresenta un itemset generico registrato in un database in congiunzione con un'attività o un ciclo di attività.

Le regole associative vengono valutate in base a due misure principali: **supporto e confidenza**.

Il **Supporto (s)** determina **quanto spesso una regola è applicabile** a un dato set di dati. Indica la frequenza con cui l'itemset formato dall'unione di antecedente e conseguente appare nelle transazioni.

Formalmente, per una regola $L \Rightarrow H$ ($o X \rightarrow Y$), il supporto è

$$s = \frac{f(L \cup H)}{m} = \frac{\sigma(X \cup Y)}{N} = \frac{\text{Numero di transazioni che contengono } X \cup Y}{\text{Numero totale di transazioni}}$$

dove

- f è la frequenza (support count σ)
- m è il numero totale di transazioni (N).

Il supporto si riferisce all'itemset piuttosto che alla regola.

La **Confidenza (c)** determina **con quale frequenza gli item nel conseguente (Y) appaiono nelle transazioni che contengono l'antecedente (X)**. Misura l'affidabilità dell'inferenza fatta da una regola. Indica la proporzione di transazioni che contengono il set H tra quelle che includono il set L .

La confidenza approssima la probabilità condizionata $\Pr\{H \subseteq T | L \subseteq T\}$.

Formalmente, la confidenza è

$$p = \frac{f(L \cup H)}{f(L)} = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

Le regole con un supporto $s \geq s_{\min}$ e una confidenza $p \geq p_{\min}$ (dove s_{\min} e p_{\min} sono valori soglia minimi prefissati) sono considerate **regole forti**.

Queste non sono sempre significative o interessanti per i decisori. L'analisi basata solo sul supporto e sulla confidenza può essere fuorviante. Ad esempio, una regola come $\{\text{camera}\} \Rightarrow \{\text{printer}\}$ potrebbe avere un elevato supporto e confidenza ($s = 0.4$, $p = 0.66$), suggerendo che l'acquisto di una fotocamera induce l'acquisto di una stampante. Tuttavia, se la probabilità di acquistare una stampante *in generale* è ancora più alta (0.75), l'acquisto di una fotocamera in realtà *riduce* la probabilità di acquistare una stampante, indicando una *correlazione negativa* nonostante la regola sia "forte" per supporto e confidenza. Questo esempio evidenzia un difetto nella valutazione basata solo su supporto e confidenza.

Il **lift (l)** è una misura della significatività di una regola associativa. È definito come il rapporto tra la confidenza della regola e la frequenza relativa (o supporto) del conseguente (testa).

Formalmente, per una regola $L \Rightarrow H$:

$$l = \text{lift}(L \Rightarrow H) = \frac{\text{conf}(L \Rightarrow H)}{f(H)} = \frac{f(L \cup H)}{f(L) \cdot f(H)} = \frac{\text{Supp}(X \cup Y)}{\text{Supp}(X) \cdot \text{Supp}(Y)}$$

Qui, $f()$ o $\text{Supp}()$ rappresentano la frequenza o il supporto dell'itemset tra il numero totale di transazioni. L è l'antecedente (body o LHS) e H è il conseguente (head o RHS).

I valori di lift **maggiori di 1** indicano che la regola considerata è **più efficace** della frequenza relativa del conseguente nel prevedere la probabilità che il conseguente sia contenuto in una transazione. In questo caso, antecedente e conseguente sono **positivamente associati**.

Se il lift è **minore di 1**, la regola è **meno efficace** della stima ottenuta tramite la frequenza relativa del conseguente. In questo caso, antecedente e conseguente sono **negativamente associati**. Se il lift è minore di 1, la regola che nega il conseguente ($\{L \Rightarrow (O - H)\}$) è più efficace della regola originale e ha un lift maggiore di 1.

Se il lift è **uguale a 1**, significa che la probabilità di trovare il conseguente (Y o H) in una transazione **non cambia** se sappiamo che l'antecedente (X o L) è presente. In termini statistici, questo indica che l'antecedente e il conseguente della regola sono **indipendenti** l'uno

Processo di Mining Il problema dell'estrazione delle regole associative forti è generalmente diviso in due fasi principali:

- **Generazione di itemset frequenti** Trovare tutti gli itemset che soddisfano la soglia di supporto minimo (minsup). Questa fase è computazionalmente più onerosa.
- **Generazione di regole:** Estrarre tutte le regole ad alta confidenza dagli itemset frequenti trovati nella fase precedente. Queste regole sono chiamate regole forti. Il calcolo della confidenza per queste regole non richiede ulteriori scansioni del set di dati se i conteggi di supporto sono già noti.

Algoritmi

Gli algoritmo per l'identificazione delle regole sono tutti euristici, non hanno fondamenti matematici e in linea teorica tutti gli algoritmi dovrebbero restituire lo stesso risultato se applicato allo stesso dataset transazionale.

AIS ALGORITHM L'algoritmo AIS è uno dei primi algoritmi sviluppati per l'estrazione di **regole associative** (*Association Rule Mining*), introdotto nel 1993 da **Agrawal, Imielinski e Swami**, da cui prende il nome (AIS).

Ha lo scopo di trovare **itemset frequenti** (combinazioni di elementi che ricorrono spesso nelle transazioni); generare **regole associative** utili a scoprire relazioni nei dati.

1. Inizializza:

- Scorri tutte le transazioni nel dataset.
- Parti da itemset singoli (1-itemset).

2. Generazione incrementale degli itemset candidati:

- Per ogni transazione, genera nuovi candidati **aggiungendo uno item alla volta** a quelli già presenti nella transazione e nella lista dei frequent itemset.
- Solo le combinazioni valide (che si trovano nella transazione) vengono considerate.

3. Conteggio:

- Mantiene i contatori di supporto per ciascun candidato generato.

4. Filtraggio:

- Alla fine, vengono selezionati solo i **frequent itemset**, cioè quelli il cui **supporto** è sopra una soglia min_support.

AIS presenta alcuni svantaggi:

- **Genera troppi candidati:** molti di questi non sono frequenti.
- **Inefficiente in termini di spazio e tempo.**
- **Generazione ridondante di itemset.**

SETM ALGORITHM L'algoritmo **SETM**, sviluppato anch'esso da Agrawal e Srikant, sta per **Set-Oriented Mining**, cioè *estrazione orientata agli insiemi*. È stato progettato per essere più **ordinato e compatibile con i database relazionali**, migliorando alcuni aspetti critici dell'AIS.

SETM cerca di trovare **combinazioni frequenti di elementi** (*frequent itemsets*) nelle transazioni e generare regole $X \Rightarrow Y$

1. Generazione dei candidati

Per ogni transazione, l'algoritmo esamina quali combinazioni di *itemsets* sono presenti. Le combinazioni candidate (cioè le possibili regole da valutare) vengono generate combinando itemset frequenti trovati nel passaggio precedente. Ogni combinazione candidata è **memorizzata** insieme all'ID della transazione da cui proviene. Questo rende facile contare quante volte ogni combinazione appare in modo preciso e organizzato.

2. Conteggio del supporto

Dopo aver generato tutti i candidati, SETM **scorre la lista dei (transazione, itemset)** salvati e **calcola la frequenza** di ogni itemset.

Poi, come per altri algoritmi, **filtra** quelli che superano una soglia minima di support.

Cerca di superare il fatto di partire da associazioni più piccole, restituendo la rule più grande quando il supporto è massimo. Raggiunto questo punto inizia a costruire una nuova regola.

I limiti di SETM sono che può **generare troppi candidati**, rallentando l'elaborazione; e l'uso di molta **memoria**, per memorizzare tutti i candidati con i rispettivi ID di transazione.

APRIORI ALGORITHM Il suo obiettivo principale è identificare i sottoinsiemi di elementi (itemset) che sono frequenti nel database e, successivamente, generare regole da questi itemset frequenti
L'algoritmo Apriori opera in due fasi principali:

FASE1-Generazione di itemset frequenti: In questa fase, l'algoritmo trova tutti gli itemset che soddisfano una soglia minima di supporto definita dall'utente. Utilizza un approccio **bottom-up** e **level-wise** per generare gli itemset frequenti. Inizia trovando tutti gli item singoli (1-itemset) che soddisfano il supporto minimo, per poi estendere iterativamente questi itemset per trovare itemset di dimensioni maggiori (k-itemset) basandosi sugli itemset frequenti di dimensione

$k-1$ trovati nel passo precedente. Questo processo continua finché non vengono trovati ulteriori itemset frequenti

L'efficacia dell'algoritmo Apriori si basa sul

Principio di Apriori Se un itemset è frequente, allora tutti i suoi sottoinsiemi devono essere anch'essi frequenti

Sulla base di questo se un sottoinsieme di dimensione $k-1$ di un k -itemset candidato non è presente nell'insieme degli $(k-1)$ -itemset frequenti (F_{k-1}), allora il k -itemset candidato viene scartato.

Viene eseguita una nuova scansione del dataset per contare il supporto di ciascun k -itemset candidato rimasto in C_k . Per velocizzare questo processo, possono essere utilizzate strutture dati come gli hash tree per raggruppare i candidati e confrontarli efficientemente con le transazioni.

I k -itemset candidati il cui supporto è maggiore o uguale alla soglia minima di supporto vengono aggiunti all'insieme dei k -itemset frequenti (F_k).

Il processo si ripete (tornando al passo 2) incrementando k finché non vengono più generati k -itemset frequenti (F_k è vuoto).

FASE2-Generazione di regole forti: Una volta identificati gli itemset frequenti, l'algoritmo genera regole associative da questi itemset. Queste regole sono valutate in base a misure come la confidenza

Se la confidenza della regola supera la **soglia minima di confidenza** predefinita, la regola viene considerata una **regola forte** e viene inclusa nell'insieme delle regole associative scoperte.

Potatura basata sulla confidenza: se una regola ha una bassa confidenza, tutte le regole generate da essa che hanno lo stesso antecedente ma un conseguente più ampio avranno una confidenza non superiore.

L'algoritmo Apriori può diventare computazionalmente intensivo per dataset molto grandi o con molte transazioni, specialmente nella fase di conteggio del supporto e nella generazione di un gran numero di candidati

APRIORI-TID ALGORITHM

L'**algoritmo AprioriTID** è una variante dell'algoritmo Apriori per l'estrazione di regole associative. AprioriTID è differente nel modo in cui gestisce il calcolo del supporto degli itemset, cercando di migliorare l'efficienza, nelle fasi successive dell'elaborazione.

AprioriTID parte dalla **seconda iterazione**, non utilizzadirettamente il database originale per contare il supporto, ma lavora con una **struttura dati temporanea**, derivata dal passaggio precedente. Questa struttura contiene, per ogni transazione, un elenco di itemset candidati che possono ancora verificarsi in quella transazione. A ogni iterazione, si aggiorna questa struttura e si utilizza per

determinare la frequenza dei nuovi candidati, evitando di rileggere l'intero database.

Questo approccio diventa particolarmente vantaggioso man mano che il numero di item in ogni transazione aumenta e i candidati diventano più complessi: la struttura temporanea tende infatti a essere più piccola del database originale, rendendo i calcoli più veloci e meno costosi in termini di memoria e tempo di accesso.

Nella **prima iterazione**, AprioriTID si comporta esattamente come l'algoritmo Apriori standard, eseguendo una scansione completa del database per trovare gli item singoli frequenti. Solo successivamente entra in gioco la strategia basata sulla struttura intermedia.

Problemi dell'analisi associativa

Il **problema del singolo minsup** è dovuto al fatto di utilizzare una singola soglia di supporto minimo per tutti gli itemset. Questa può essere un problema perché ci possono essere itemset composti da elementi che appaiono raramente nel dataset, ma che potrebbero rappresentare associazioni interessanti.

Se si imposta una soglia di supporto troppo alta per ridurre il numero di itemset frequenti generati, tali itemset rari ma potenzialmente significativi potrebbero non essere identificati; ma se si abbassa la soglia di supporto per catturare questi itemset rari, si potrebbe generare un numero eccessivamente elevato di itemset frequenti che includono elementi molto comuni, molti dei quali potrebbero non portare a regole associative interessanti.

Una soluzione è quella del **supporto multiplo** che permette a diversi itemset o gruppi di itemset di avere diverse soglie di supporto minimo. Questo consentirebbe di identificare sia le associazioni tra item più frequenti con una soglia più alta, sia le associazioni tra item meno frequenti ma potenzialmente più specifici o di nicchia con una soglia più bassa.

La **Class Association Rule Mining (CAR)** è una **variante dell'association rule mining** in cui l'obiettivo non è solo scoprire associazioni generiche tra item, ma **associare insiemi di attributi a una classe specifica**.

Nelle **Class Association Rules** il conseguente è **sempre una classe**:

$$X \rightarrow c$$

dove:

- X è un insieme di condizioni (item o attributi),
- c è una **classe target** (etichetta).

CAR viene utilizzato per:

- Costruire **classificatori basati su regole**,

- Scoprire regole interpretabili che aiutano a **capire il comportamento dei dati rispetto a una classe**.

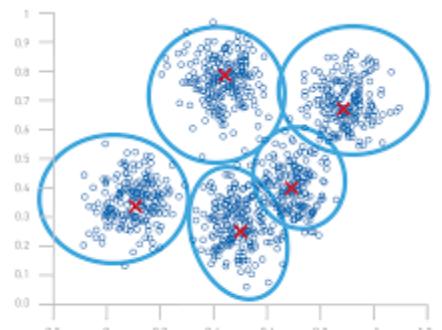
UNSUPERVISED ML: modello di clustering

L'analisi dei cluster divide i dati in gruppi (cluster) che sono significativi, utili o entrambi. Se l'obiettivo sono gruppi significativi, allora i cluster dovrebbero catturare la struttura naturale dei dati. In alcuni casi, tuttavia, l'analisi dei cluster viene utilizzata per la sintesi dei dati al fine di ridurre la dimensione dei dati.

L'analisi dei cluster raggruppa gli oggetti dati in base alle informazioni trovate solo nei dati che descrivono gli oggetti e le loro relazioni. L'obiettivo è che gli oggetti all'interno di un gruppo siano simili (o correlati) tra loro e diversi (o non correlati) dagli oggetti in altri gruppi. Maggiore è la somiglianza (o l'omogeneità) all'interno di un gruppo e maggiore è la differenza tra i gruppi, migliore o più distinta è la clusterizzazione.

ALGORITMO ML: K-means

Il K-means è un metodo di quantizzazione vettoriale che definisce un modello in termini di un centroide, che di solito è la media di un gruppo di punti, ed è tipicamente applicato a oggetti in uno spazio continuo n-dimensionale. Il centroide non corrisponde quasi mai a un punto dati effettivo.



1. scegliamo K centroidi iniziali, dove K è un iperparametro e indicata il numero di cluster che vogliamo creare;
2. ogni punto viene quindi assegnato al centroide più vicino e ogni raccolta di punti assegnata a un centroide è un cluster;
3. Il centroide di ogni cluster viene quindi aggiornato in base ai punti assegnati al cluster.
4. Ripetiamo i passaggi di assegnazione e aggiornamento finché nessun punto cambia cluster o, finché i centroidi rimangono gli stessi

Per assegnare un punto al centroide più vicino, abbiamo bisogno di una misura di prossimità che quantifichi la nozione di "più vicino" si può usare la **distanza euclidea** anche se si possono scegliere tipi di misura diversi per il tipo di dato

che usiamo come ad esempio, la **distanza di Manhattan** può essere utilizzata per i dati euclidei, o la **misura di Jaccard** per i documenti.

Per la funzione obiettivo, che misura la qualità di un clustering, utilizziamo la **somma dell'errore quadratico (SSE)**, che è anche nota come dispersione. Calcoliamo l'errore di ogni data-point, ovvero la sua distanza dal centroide più vicino, e calcoliamo la somma totale degli errori quadratici. Dati due diversi set di cluster prodotti da due diverse serie di K-means, il migliore è quello con l'errore quadratico più piccolo poiché ciò significa che i centroidi di questo clustering sono una migliore rappresentazione dei punti nel loro cluster.

Scelta del centroide iniziale

Si posizionamento del centroide iniziale potrebbe essere anche fatto in modo casuale, ma se i centroidi iniziali non sono ben distribuiti, l'algoritmo può convergere a una soluzione subottimale invece di trovare il miglior raggruppamento. Questo accade perché l'algoritmo è **sensibile alla posizione iniziale** dei centroidi. Oppure se alcuni centroidi iniziali vengono scelti in zone **molto dense**, alcuni cluster risulteranno **sovraffollati**, mentre altri saranno quasi vuoti o completamente assenti. Questo porta a una distribuzione **non uniforme** dei punti tra i cluster.

Un approccio alternativo è quello di prendere un campione di punti e di raggrupparli utilizzando una tecnica di clustering gerarchico. K cluster vengono estratti dal clustering gerarchico e i centroidi di tali cluster vengono utilizzati come centroidi iniziali. Questo approccio spesso funziona bene, ma è pratico solo se

- il campione è relativamente piccolo, ad esempio, da poche centinaia a poche migliaia (il clustering gerarchico è costoso)
- K è relativamente piccolo rispetto alla dimensione del campione.

Un approccio migliore è il **K-mean++** nel quale si sceglie la posizione del primo centroide a caso e poi gli altri rimanenti si posizionano il più lontano possibile dal primo. La posizione è scelta, in modo che il nuovo centroide, abbia distanza proporzionale al quadrato della sua distanza dal centroide più vicino.

Algorithm 5.2 K-means++ initialization algorithm.

- 1: For the first centroid, pick one of the points at random.
 - 2: **for** $i = 1$ to *number of trials* **do**
 - 3: Compute the distance, $d(x)$, of each point to its closest centroid.
 - 4: Assign each point a probability proportional to each point's $d(x)^2$.
 - 5: Pick new centroid from the remaining points using the weighted probabilities.
 - 6: **end for**
-

L'**elbow plot** è una tecnica utilizzata per determinare il numero ottimale di cluster in un algoritmo di clustering.

L'idea è di calcolare la **somma delle distanze quadratiche (interna)** di ogni punto all'interno di ciascun cluster (chiamata **inertia**) per vari valori di **K** (il numero di cluster).

Si esegue l'algoritmo K-means per diversi valori di **K** (ad esempio da 1 a 10) e si traccia un grafico della **somma delle distanze** rispetto ai valori di **K**.

In genere, quando si aumenta **K**, la somma delle distanze diminuirà, poiché più cluster permettono una migliore separazione dei dati.

L'**elbow** si riferisce al punto in cui l'ulteriore aumento del numero di cluster non porta a un miglioramento significativo della somma delle distanze (cioè la curva si appiattisce). Questo punto indica il numero ottimale di cluster, poiché oltre a questo punto, il guadagno in termini di separazione dei dati diminuisce.

Quindi serve a **determinare il numero di cluster ottimale** per un algoritmo di clustering, riducendo il rischio di overfitting (troppi cluster) o underfitting (pochi cluster), e **valutare la qualità del clustering** perché un numero eccessivo di cluster potrebbe suggerire che i dati non sono ben separabili, mentre un numero troppo basso potrebbe indicare che il modello non comprende la complessità dei dati.

REINFORCEMENT LEARNING: modello di decisione markoviano

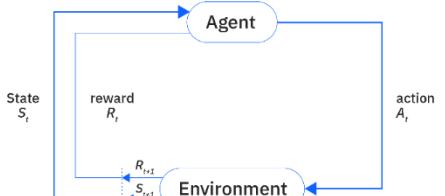
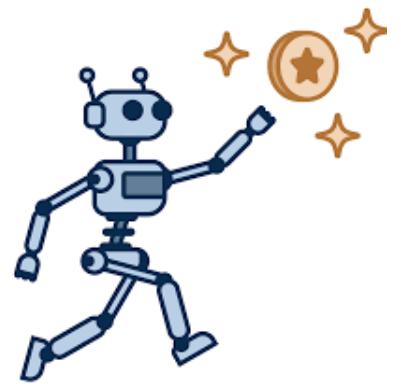
Un **Markov Decision Process (MDP)** è un modello matematico utilizzato per descrivere un ambiente in cui un agente prende decisioni in modo sequenziale, interagendo con l'ambiente in un contesto di incertezze. Un **agente** cerca di massimizzare una certa ricompensa nel tempo, facendo scelte in un ambiente stocastico.

Un **agente**, ha un set di sensori per osservare lo stato del suo ambiente e un set di azioni che può eseguire per alterare questo stato. Il suo compito è apprendere una strategia di controllo, o **policy**, per scegliere azioni che raggiungano i suoi obiettivi. Supponiamo che gli obiettivi dell'agente possano essere definiti da una **funzione di ricompensa** che assegna un valore numerico, un guadagno immediato, a ciascuna azione distinta che l'agente può intraprendere da ogni stato distinto. Il compito dell'agente è eseguire sequenze di azioni, osservarne le conseguenze e apprendere una politica di controllo. La politica di controllo che desideriamo è quella che, da qualsiasi stato iniziale, sceglie azioni che massimizzano la ricompensa accumulata nel tempo dall'agente.

In un **processo decisionale di Markov (MDP)** l'agente ha un insieme S di stati distinti del suo ambiente e ha un insieme A di azioni che può eseguire. Ad ogni passo temporale discreto t , l'agente rileva lo stato corrente s_t , sceglie un'azione corrente a_t , e la esegue. L'ambiente risponde dando all'agente una ricompensa $r_t = r(s_t, a_t)$ e producendo lo stato successivo $s_{t+1} = \delta(s_t, a_t)$. Qui le funzioni δ e r fanno parte dell'ambiente e non sono necessariamente note all'agente.

In un MDP, le funzioni $\delta(s_t, a_t)$ e $r(s_t, a_t)$ dipendono solo dallo stato e dall'azione correnti e non da stati o azioni precedenti. In generale, δ e r possono essere funzioni non deterministiche

il compito dell'agente è apprendere una politica, $\pi : S \rightarrow A$, per selezionare la sua prossima azione a , in base allo stato attuale osservato s_t ; ovvero, $\pi(s_t) = a_t$. Per specificare quale policy vogliamo che l'agente impari in modo da produrre la massima ricompensa cumulativa.



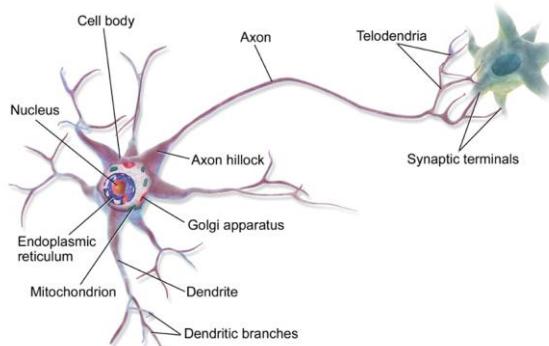
RETI NEURALI

Le Reti Neurali Artificiali (RNA) sono un tipo di modello di Machine Learning che tenta di simulare il comportamento dei sistemi neurali biologici presenti nel Il **neurone** è una cellula che costituisce l'elemento base del nostro sistema nervoso. È responsabile della trasmissione delle informazioni nel cervello, nel midollo spinale e in tutto il corpo, attraverso impulsi elettrici e segnali chimici.. La sua struttura di un neurone è formata da tre parti principali: i **dendriti**, il **corpo cellulare**, e l'**assone**.

I **dendriti** sono rami che si estendono dal corpo del neurone. Il loro compito è quello di ricevere messaggi da altri neuroni. Quando un neurone riceve un segnale, questo arriva prima ai dendriti, che lo raccolgono come se fossero antenne e lo trasmettono verso il centro della cellula.

Il **corpo cellulare**, situato tra i dendriti e l'assone, è il cuore del neurone. Qui si trovano il nucleo e tutti gli organelli necessari alla sopravvivenza e al funzionamento della cellula. È proprio nel corpo cellulare che avviene l'elaborazione iniziale del segnale ricevuto.

Una volta che il segnale è abbastanza forte, il neurone lo trasmette lungo l'**assone**, che trasporta l'impulso nervoso verso la sua destinazione.



Sulla base del neurone umano è stato creato il neurone artificiale che non è altro che un modello matematico basato su grafi orientati.

Questi modelli neurali possono essere:

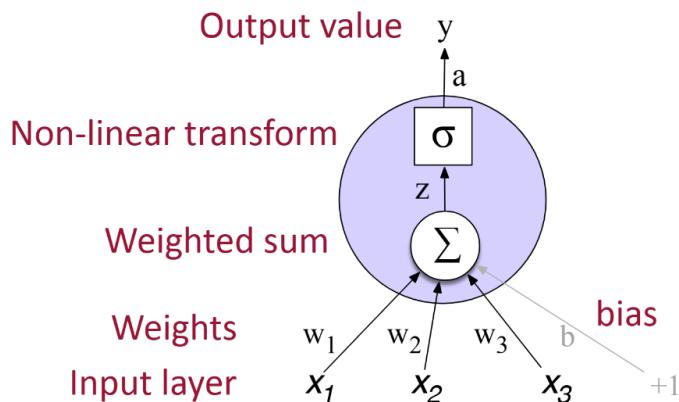
- **shallow**, che sono formati da un solo strato di neuroni;
- **deep** che sono formati da molti, se non moltissimi strati di neuroni.

Percettrone

Uno dei primi modelli fu quello del **percettrone**, una rete neurale formata da un solo neurone. Questo riceve **N input**, ognuno con un **peso**, più un **bias**, un termine noto. Questi vengono tutti moltiplicati con i loro pesi e sommati fra loro; questa sommatoria poi viene modificata da una funzione non lineare, **funzione di attivazione**, che modifica la sommatoria e produce un output solo se si supera un certo valore di threshold. Questa simula la funzione di decisione del

neurone umano, quindi modifica gli input ricevuti. Si sceglie una funzione non lineare perché questa permette di modellare molti più problemi e con molto più precisione rispetto ad una funzione lineare.

Il **bias** è un **valore costante** che viene aggiunto alla somma ponderata degli input di un neurone. Serve a **spostare la funzione di attivazione** verso destra o verso sinistra, permette al modello di adattarsi meglio ai dati, anche quando tutti gli input sono zero. Aumenta la flessibilità del modello, migliorando la capacità di apprendere relazioni complesse.



I pesi non vengono assegnati a caso ma vengono determinati tramite il processo di addestramento del modello neurale. La fase di addestramento viene eseguita analizzando sequenzialmente le osservazioni nel set di addestramento e modificando i pesi associati ai collegamenti a ogni iterazione. Inizialmente si associano dei pesi casuali e poi sulla base dell'output si modificano cambiando anche la topologia della rete neurale, o nel caso il numero dei layer. L'addestramento di una RNA può essere un processo richiedente tempo, soprattutto quando il numero di nodi nascosti è elevato.

Topologia delle reti

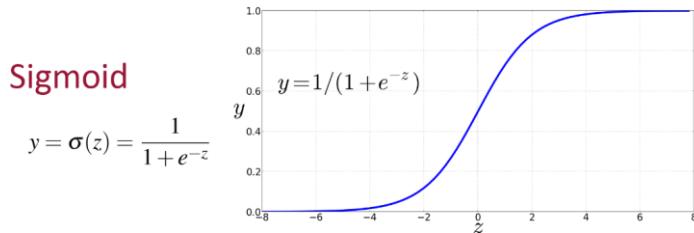
Sulle base dei nodi della rete si possono avere infinite configurazioni, topologie di rete. Maggiore sono il numero di nodi, maggiori sono anche il numero di layer e più pattern fra i dati si possono cogliere.

Al di là del costo economico, i nodi e le connessioni occupano spazio in memoria. Il fatto di avere tanti i nodi, influenza il numero di bit che usiamo per la funzione di attivazione.

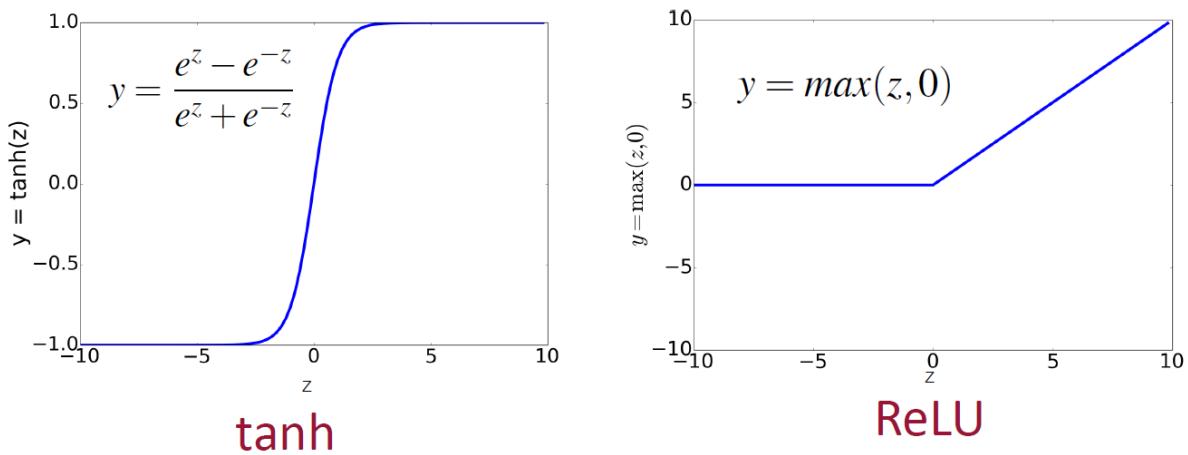
Funzioni di attivazione

Sulla base del problema che dobbiamo modellare sceglieremo una certa funzione di attivazione. Nel primo modello del percettrone, la funzione di attivazione era un funzione a gradino, ma questa aveva il problema di essere non derivabile q

questo era un problema essendo che le reti neurali funzionano tramite esse. Si è scelta la **sigmoide** per il suo andamento simile, ma con il vantaggio di essere derivabile.



Oppure un'alternativa è la **tanh** che l'ulteriore vantaggio di essere normalizzata [-1;1]. La **ReLU** una funzione che introduce non linearità nel modello, consentendo alla rete di apprendere e rappresentare relazioni più complesse nei dati.



Rectified Linear Unit

Le **squashing function** sono funzioni di attivazione che "schiacciano" i valori di input all'interno di un intervallo limitato. Sono fondamentali nelle reti neurali perché:

- Rendono l'output prevedibile e gestibile, limitandolo a un certo range.
- Stabilizzano l'addestramento, evitando valori troppo grandi o troppo piccoli che possono causare errori numerici.
- Facilitano l'interpretazione dell'output, ad esempio come **probabilità**.

Sulla base del problema che dobbiamo modellare dobbiamo, si deve scegliere una **decision surface**, un iperpiano che separa le varie classi del problema. L'iperpiano è un **manifold** cioè uno spazio topologico che localmente è uno spazio euclideo.

Dal matematiche all'italiano, questo vuol dire che un manifold non è altro che un poliedro e che ogni faccia di esso è uno spazio topologico che separa la classe da un'altra.



Ciò che fa la funzione di attivazione è imparare la decision surface e per comprendere la il problema deve essere linearmente separabile.

Teorema Separabilità lineare

Se il problema è **linearmente separabile**, allora l'algoritmo del percettrone **converge in un numero finito di iterazioni** a una soluzione che **separa perfettamente** le classi nel training set, indipendentemente dal valore del learning rate (purché costante e positivo).

Tipi di rete neurale

Il **Multilayer Perceptron (MLP)** è un tipo di rete neurale definito da un grafo stratificato con tre strati:

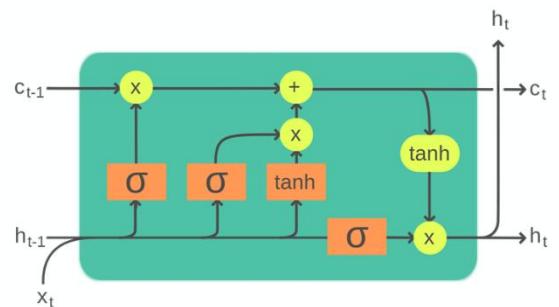
- **Strato di Input:** Questo è il primo strato e corrisponde al numero di attributi di input. Tipicamente, ogni attributo numerico o binario è rappresentato da un singolo nodo.
- **Strati Nascosti (o Strato Nascosto):** Questi sono gli strati intermedi tra lo strato di input e lo strato di output. Ogni nodo nascosto opera sui segnali ricevuti dallo strato precedente e produce un valore di attivazione che viene trasmesso allo strato successivo. Se visti dall'alto, gli strati nascosti sono oscurati dallo strato di output. Uno strato nascosto converte gli input originali in una combinazione di livello superiore di tali input, fungendo da meccanismo di estrazione delle caratteristiche.
- **Strato di Output:** Questo è l'ultimo strato e produce le previsioni. Per la classificazione binaria, lo strato di output contiene un singolo nodo che rappresenta l'etichetta di classe binaria. Per un dominio con m classi, lo strato di output può consistere in un neurone per ciascuna classe.

Le **reti neurali profonde** sono semplicemente

reti neurali su larga scala che possono contenere molti strati di neuroni.

Contrastando con le reti neurali "shallow" (poco profonde) che coinvolgono solo un piccolo numero di strati nascosti, le reti neurali profonde sono in grado di rappresentare caratteristiche a più livelli di

astrazione e spesso richiedono un numero significativamente inferiore di nodi per strato per raggiungere prestazioni di generalizzazione simili. Le **Recurrent Neural Networks (RNN)**, a differenza degli MLP, permettono connessioni di feedback e di conseguenza di avere memoria per elaborare input che si basano anche su quelli precedenti. Queste hanno memoria a breve termine, ma apportando modifiche al nodo possiamo realizzare, le **Long Short-Term**



Memory (LSTM) in cui i nodi raggruppano diverse funzioni di attivazione e per poter produrre l'output, tengono conto anche dello stato e dell'output del nodo precedente.

Il flusso di propagazione dei dati è **feed forward**, cioè va sempre in avanti, dagli strati di input a quelli nascosti (se presenti) e infine allo strato di output.

Per i problemi di classificazione binaria, la funzione di attivazione che viene usata è la **funzione logistica**, che trasforma l'output in una probabilità. Ma in una rete che contiene più output, quindi una classificazione multiclasse si usa la **funzione logistica Multimodale o Softmax** che assegna delle probabilità sulla possibilità di avere quel sentimento. La somma delle probabilità fra tutte le classi obv deve essere sempre 1.

For a vector z of dimensionality k , the softmax is:

$$\text{softmax}(z) = \left[\frac{\exp(z_1)}{\sum_{i=1}^k \exp(z_i)}, \frac{\exp(z_2)}{\sum_{i=1}^k \exp(z_i)}, \dots, \frac{\exp(z_k)}{\sum_{i=1}^k \exp(z_i)} \right]$$

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^k \exp(z_j)} \quad 1 \leq i \leq k$$

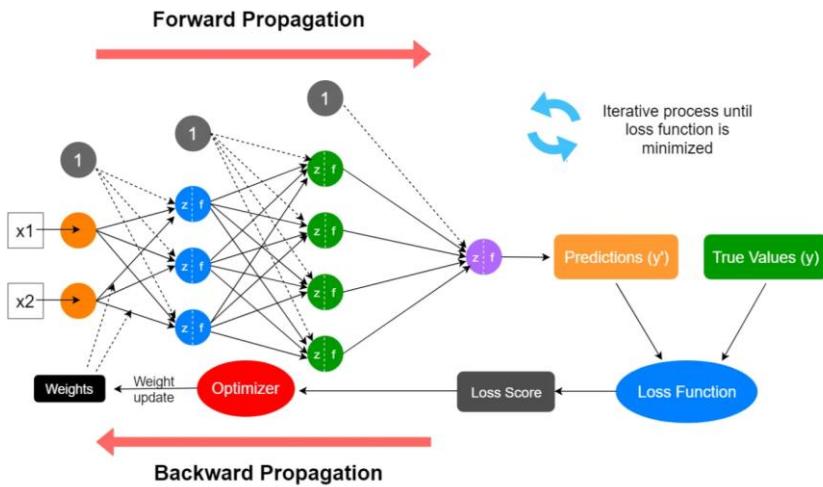
TRAINING

Lo scopo principale del training è regolare i pesi e i bias all'interno della rete neurale. Questo viene fatto in modo che l'output della rete per ciascun input del set di addestramento sia **sufficientemente vicino al valore target corrispondente**. L'obiettivo è quindi **ottimizzare la performance della rete**, che di solito significa **minimizzare una funzione di errore o di perdita**. Le reti neurali apprendono in un processo supervisionato. Ciò significa che il processo di apprendimento è **induttivo**.

Il training è un processo iterativo.

1. inizializzazione dei pesi a piccoli numeri casuali.
2. Gli esempi di addestramento vengono presentati alla rete, uno alla volta eseguendo una propagazione in avanti per calcolare l'output temporaneo della rete.
3. L'**output calcolato** viene confrontato con il **target desiderato**, e la **discrepanza** viene calcolata.
4. Questo errore viene quindi utilizzato per **modificare i pesi** al fine di **ridurre la discrepanza**.
5. Quando l'ultimo esempio di addestramento è stato presentato, si completa una **epoch**.

Questo processo viene ripetuto per **diverse epoch**. Il training continua fino a quando non viene soddisfatto un **criterio di arresto**.



Miglioramento dell'apprendimento

La **backpropagation** è il meccanismo che permette alla rete di imparare dai propri errori, modificando i pesi in modo da migliorare le previsioni nel tempo. Dopo che l'output è stato prodotto questo viene confrontato il **valore target** che indica la classe corretta. Si determina per ogni uscita la perdita tra il valore ottenuto e quello target tramite una **funzione di loss**, che rappresenta la funzione dell'errore e che deve essere minimizzata per creare un modello decente. Si determina per ogni nodo la **responsabilità** dei neuroni dello strato di output, che presenta la misura in cui l'attività di quel neurone ha influenzato l'errore finale della rete. È utile per identificare quali neuroni e connessioni devono essere modificati per ridurre l'errore complessivo. L'obiettivo principale dell'addestramento è **minimizzare una funzione loss**.

Tipi di Funzioni di Loss

Mean Square Error – MSE $\text{Loss}(y_k, \hat{y}_k) = (y_k - \hat{y}_k)^2$, dove y_k è l'etichetta vera e \hat{y}_k è l'output della rete. L'errore totale E sull'intero set di addestramento è la somma delle perdite individuali

Cross Entropy La cross entropy si usa per misurare la differenza tra due distribuzioni di probabilità ed è utile quando vogliamo confrontare una distribuzione attesa con una distribuzione predetta.

È il numero medio di bit necessari per identificare un evento quando si utilizza una codifica basata su una distribuzione q , anziché quella vera p . Quando addestriamo una rete neurale, utilizziamo la cross entropy per confrontare le previsioni del modello (q_i) con i valori attesi (p_i).

L'obiettivo è minimizzare la cross entropy, cioè far sì che le probabilità predette dal modello siano il più simile possibile a quelle attese.

- Cross entropy is defined as:

$$H(p, q) = - \sum_{i=1}^c p_i \cdot \log(q_i)$$

- If we consider y and \hat{y} the cross-entropy (to be minimized) will be given by:

$$H(y, \hat{y}) = - \sum_{i=1}^c y_i \cdot \log(\hat{y}_i)$$

- Finally, for a 2 class classification where $y_2 = 1 - y_1$:

$$H(y, \hat{y}) = - y \cdot \log(\hat{y}) - (1 - y) \cdot \log(1 - \hat{y})$$

Questa funzione di perdita ha un vantaggio significativo quando utilizzata con funzioni di attivazione sigmoide nello strato di output, poiché semplifica il calcolo dei gradienti, rendendoli più efficaci nel promuovere l'apprendimento. La funzione di loss dipende dai pesi e dal bias.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^m L_{\text{CE}}(f(x^{(i)}; \theta), y^{(i)})$$

- θ : insieme dei **pesi e bias** della rete neurale (cioè i parametri da apprendere).
- $\hat{\theta}$: rappresenta i **parametri ottimali** che minimizzano la funzione di perdita.
- m : numero di esempi nel dataset di addestramento.
- $x^{(i)}$: input dell'i-esimo esempio.
- $y^{(i)}$: etichetta corretta (target) dell'i-esimo esempio.
- $f(x^{(i)}; \theta)$: output della rete neurale per l'input $x^{(i)}$ usando i parametri θ
- L_{CE} : funzione di **loss cross-entropy** usata spesso nella classificazione.

La funzione di loss per la regressione logistica è una funzione convessa quindi sappiamo che ha solo un minimo. La minimizzazione della funzione di loss viene solitamente realizzata utilizzando un **algoritmo di ottimizzazione** basato sul **gradiente** perché partendo da un qualunque punto della funzione, sicuramente arriviamo al minimo. Ma per le reti neurali no. 😊 te pareva...

Usiamo il **Gradient Descent**, capace d'individuare il valore minimo di una **cost function**, permettendo di migliorare le previsioni. Calcoliamo la $\partial L / \partial w$ derivata

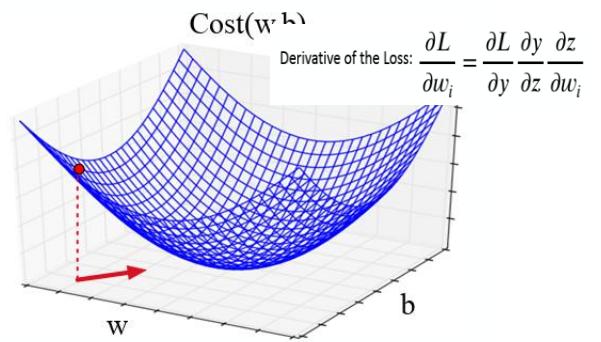
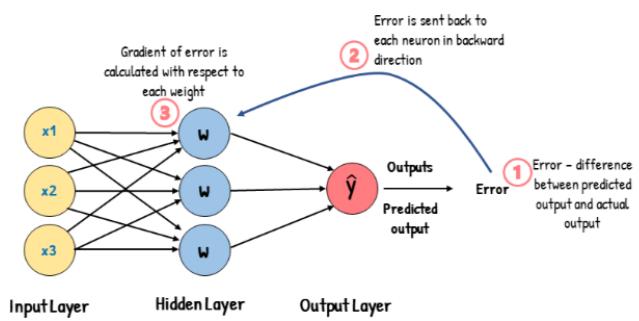
parziale della loss rispetto al peso e aggiorniamo i pesi **nella direzione opposta al gradiente**.

$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x; w), y)$$

Per capire di quanto deve essere aggiornato il peso ci si basa sul **learning rate η** , un hyperparameter controlla **quanto velocemente** il modello **aggiorna i pesi** durante l'addestramento, sulla base dell'errore commesso.

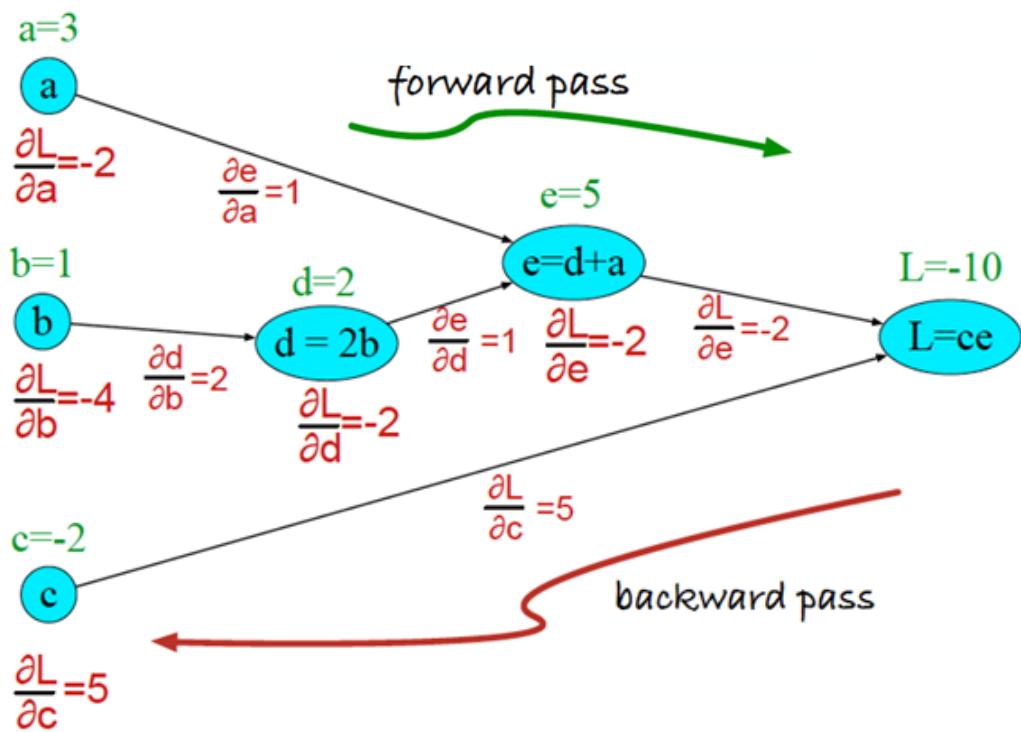
- Per η troppo piccolo, il tempo di **convergenza** è grande, quindi la determinazione del punto di minimo impiega parecchie risorse computazionali.
- Per η grande, potrebbe fallire la **convergenza**, superando il minimo, o arrivando persino a divergere mancando la determinazione della soluzione.

Backpropagation



Lo stesso ragionamento lo si può fare a N dimensioni, ma con il problema che il gradiente ora è un vettore di derivate e questo può essere un problema per il fatto che le derivate, per essere calcolate sfruttano la chain rule, e può risultare computazionalmente complesso calcolarlo. Ecco perché conviene usare una funzione che sia facilmente derivabile. Una soluzione può essere usare **computation graph**, che ci aiuta a rappresentare e gestire i passaggi necessari per calcolare un'espressione matematica complessa.

Ogni operazione è rappresentata come un nodo del grafo e le connessioni indicano come i risultati di alcune operazioni vengono usati come input per altre. Questo grafo ci permette di scomporre anche espressioni complesse in operazioni atomiche.



Soluzione: error backpropagation, un caso speciale di backward Differentiation



STEP7 - Ottimizzare il modello e valutarne l'accuratezza

La valutazione di un modello di Machine Learning (ML) è utile per comprendere le prestazioni, confrontarlo con altri modelli e determinare se è adatto per l'uso previsto. Ha due obiettivi principali:

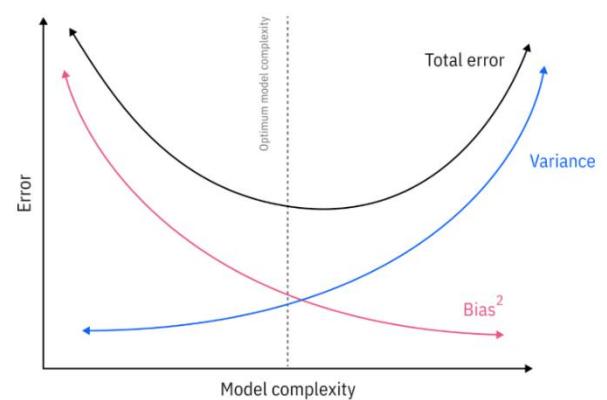
- **Stimare l'accuratezza della generalizzazione:** Valutare quanto bene il modello si comporta su dati non visti, al di fuori dell'insieme di addestramento.
- **Confrontare diversi modelli:** Determinare quale modello offre le migliori prestazioni per un dato problema.



GENERALIZZAZIONE

Il concetto di **generalizzazione** si riferisce alla capacità di un modello di prevedere correttamente su nuovi esempi o istanze che non sono stati visti durante l'addestramento. L'obiettivo fondamentale dell'apprendimento automatico è indurre funzioni **generali** da esempi specifici di addestramento. La capacità di generalizzazione di un modello è strettamente legata al trade-off tra bias ed errore di test.

- Un modello con un **bias elevato** è eccessivamente semplificato e potrebbe non riuscire a catturare la struttura sottostante dei dati (underfitting), compromettendo la generalizzazione.
- Modelli con un **bias basso** sono più complessi, ma con set di addestramento piccoli possono soffrire di alta varianza e portare all'overfitting, che è l'incapacità di generalizzare a nuovi dati.



L'errore di generalizzazione (**EMSE: Expected Mean Squared Error**) si può scomporre in tre componenti fondamentali:

$$E[\text{Errore}] = \text{Bias}^2 + \text{Varianza} + \text{Rumore}$$

Bias: è l'errore dovuto alle assunzioni errate o semplificative del modello. È alto se il modello è troppo semplice .

Varianza: è la sensibilità del modello ai dati di addestramento. Se cambia molto da un dataset all'altro, ha alta varianza (es. overfitting).

Rumore: è l'errore intrinseco e non eliminabile nei dati, dovuto a imprecisioni o casualità.

L'obiettivo principale del machine learning è di ridurre l'errore di generalizzazione del modello. Per farlo, **è necessario trovare il giusto equilibrio tra bias e varianza (Bias-Variance Trade-off)**

La scelta del giusto modello dipende dalle esigenze del problema.

1. **Valutare la complessità del modello:** modelli semplici come le regressioni lineari hanno un alto bias e bassa varianza, mentre i modelli complessi come le reti neurali hanno un basso bias e alta varianza.
2. **Dimensione del dataset:** aumentando la dimensione del dataset, è possibile ridurre la varianza del modello. Infatti, con più dati di addestramento, il modello avrà maggiori informazioni per generalizzare e ridurre l'adattamento ai dati di addestramento.
3. **Regolarizzazione:** la regolarizzazione è una tecnica utilizzata per controllare la complessità del modello. Ad esempio, la regolarizzazione L1 e L2 possono aiutare a ridurre la varianza del modello.
4. **Selezione delle feature:** la selezione delle feature è un'altra tecnica utilizzata per controllare la complessità del modello. Rimuovere le feature non rilevanti o ridondanti può aiutare a ridurre la varianza del modello.

Il **bootstrap sampling** è una tecnica statistica di campionamento con ripetizione. per stimare la variabilità di una statistica (media, varianza, mediana, ecc.) senza fare assunzioni forti sulla distribuzione dei dati. Il bootstrap viene usato per creare molteplici varianti artificiali di un dataset, a partire da un unico campione osservato. La caratteristica distintiva del bootstrap è che, per creare un nuovo dataset, si estrae casualmente (con rimpiazzo) dal dataset originale lo stesso numero di osservazioni.

Supponiamo di avere un dataset D con n osservazioni.

1. Estrazione con ripetizione: si selezionano n elementi da D, scegliendo ogni volta un elemento a caso, ma rimettendolo dentro. Quindi un'osservazione può essere scelta più volte, mentre un'altra può non essere scelta mai.
2. Questo processo genera un nuovo dataset Db, chiamato bootstrap sample.
3. Si possono generare molti bootstrap sample (D1,D2,...,Db) ripetendo questo procedimento più volte.

Una proprietà interessante è che circa il 63% delle osservazioni originali comparirà, in media, almeno una volta in ciascun campione bootstrap, mentre il resto delle osservazioni (circa il 37%) non verrà selezionato. Queste osservazioni non incluse sono chiamate out-of-bag samples (*OOB*), e hanno un ruolo molto importante, per esempio nella stima dell'errore nei metodi ensemble come il **bagging (Bootstrap Aggregating)**. L'idea alla base del bagging è che invece di addestrare un singolo modello su un unico dataset, si addestrano **molti modelli**, ognuno su una **variante diversa** dello stesso dataset, ottenuta attraverso una tecnica chiamata **bootstrap**.

Ogni campione bootstrap viene usato per addestrare un **modello indipendente**. Alla fine, quando arriva il momento di fare una predizione, si aggregano tutte le predizioni individuali:

- **Nel caso della classificazione**, si utilizza il **voto di maggioranza**: la classe più predetta è quella scelta.
- **Nel caso della regressione**, si calcola invece la **media** delle predizioni.

Il bagging si basa su un principio statistico: la media di molte stime indipendenti è meno variabile di una singola stima. Se i modelli che si addestrano sono abbastanza diversi tra loro, ma non troppo sbagliati individualmente, la loro combinazione riduce significativamente l'errore totale, in particolare la varianza, lasciando quasi invariato il bias. Questo rende il bagging ideale per modelli che tendono a **overfittare**. Un'applicazione famosa del bagging è il metodo delle **Random Forest**, che estende il bagging introducendo anche una selezione casuale delle feature in fase di addestramento degli alberi, rendendoli ancora più diversificati tra loro.

Naturalmente, il bagging ha anche dei limiti, non aiuta molto quando il problema principale è l'alta bias: se il modello base è troppo semplice per rappresentare correttamente il fenomeno, il bagging non potrà compensare questa mancanza. Inoltre, può risultare **computazionalmente costoso**, perché richiede l'addestramento di molti modelli.

Cross-Validation

La cross-validation è una tecnica statistica che consiste nel dividere il dataset in più parti (detti *fold*) e nel valutare il modello su dati che non ha mai visto durante l'addestramento, ripetendo questa procedura più volte. Questo insieme simula l'arrivo di nuovi dati reali, permettendo una stima non ottimistica dell'errore di generalizzazione del modello in produzione. L'utilizzo corretto di un test set separato è essenziale per evitare il cosiddetto **data leakage** e garantire una valutazione attendibile. **Data leakage** è un problema che si verifica quando informazioni del dataset di test o dati futuri "trapelano" nel processo di addestramento o selezione del modello, causando una stima troppo ottimistica delle prestazioni. In pratica, il modello "vede" dati che dovrebbe invece ignorare, quindi impara da informazioni che non saranno disponibili quando sarà usato su dati nuovi. Per evitarlo, una parte dell'insieme di addestramento può essere ulteriormente suddivisa in un **insieme di validazione**. Questo insieme viene utilizzato per la **selezione del modello** e per l'**ottimizzazione degli iperparametri**. L'errore sull'insieme di validazione (errval) è un indicatore migliore delle prestazioni di generalizzazione rispetto all'errore sull'insieme di addestramento.

Holdout Method: Il dataset viene diviso in due insiemi disgiunti: un **insieme di addestramento** (utilizzato per la selezione del modello) e un **insieme di test** (utilizzato per stimare il tasso di errore di generalizzazione, errtest).

K-fold Cross-Validation: Per ottenere una stima più robusta delle prestazioni, si utilizza la **cross-validation**. Una tecnica comune è la **k-fold cross-validation**, in cui i dati vengono divisi in k "pieghe" (fold). Il modello viene addestrato su $k-1$ pieghe e valutato sulla piega rimanente. Questo processo viene ripetuto k volte, con ogni piega utilizzata una volta come insieme di test. Le prestazioni vengono quindi aggregate (ad esempio, calcolando la media dell'accuratezza su tutte le pieghe, ACVA). Una forma estrema di cross-validation è il **leave-one-out**, in cui k è uguale al numero di punti dati, con ogni punto utilizzato per il test una sola volta.

Nested Cross-Validation: Quando sono presenti iperparametri da sintonizzare, si può utilizzare la **nested cross-validation**, che include un ciclo interno di cross-validation per la selezione degli iperparametri e un ciclo esterno per la valutazione del modello con gli iperparametri ottimali.

Metriche di valutazione per modelli di classificazione

Per valutare i modelli di classificazione, vengono utilizzate diverse metriche basate la principale è l'**accuracy** che ci indica quante volte il nostro modello ha correttamente classificato un item nel nostro dataset rispetto al totale.

$$\text{accuracy} = \frac{\text{numero_risposte_corrette}}{\text{numero_risposte_totali}}$$

Non ci permette di comprendere il **contesto** nel quale stiamo operando. Da sola non va bene, conviene accompagnarla insieme alla **matrice di confusione**, le cui dimensioni dipendono dal numero di feature che abbiamo.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

True Positive TP → Il modello prevede positivo e in effetti è positivo.

False Positive FP → Il modello prevede positivo ma è negativo (errore tipo I). Falso allarme 😊

True Negative TN → Il modello prevede negativo e in effetti è negativo.

False Negative FN → Il modello prevede negativo ma è positivo (errore tipo II). Mancata rilevazione. 🙄

Dalla matrice di confusione si possono calcolare varie metriche.

$$\text{accuracy} = \frac{TN + TP}{TN + TP + FP + FN}$$

La **precision** indica quante classi classificate come positive che sono effettivamente positive. È sensibile allo sbilanciamento delle classi.

$$\text{Precision} = \frac{TP}{TP + FP}$$

La **recall (o Sensibilità)** indica fra tutte le istanze effettivamente positive, quante il modello ha individuato correttamente. È sensibile allo sbilanciamento delle classi.

$$\text{Recall} = \frac{TP}{TP + FN}$$

La **specificità** indica fra le istanze negative quante ne sono state correttamente identificate come negative.

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Quando abbiamo più feature calcoliamo queste metriche per tutte. Un classificatore sarà buono se le metriche di tutte le feature che consideriamo sono buone

L'**F1-score** è la media armonica tra Precision e Recall. Serve per trovare un equilibrio tra questi due valori.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN}$$

L'**F β -score** è una generalizzazione dell'F1-score, calcolata tramite media armonica ponderata che permette di dare maggiore importanza a Precision o Recall a seconda delle necessità del problema, utilizzando un parametro β .

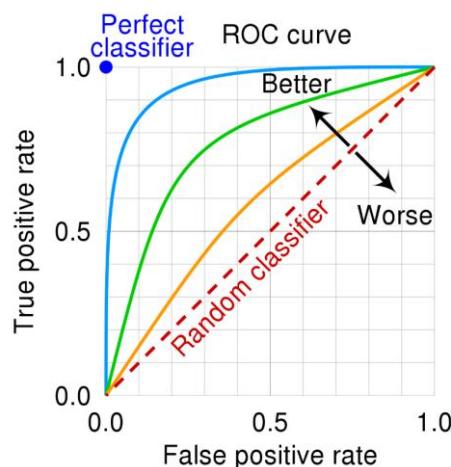
$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}} = (1 + \beta^2) \cdot \frac{2TP}{2TP + \beta^2 \cdot FP + FN}$$

Se $\beta > 1$, dà più peso al Recall (importante se i falsi negativi sono più gravi).

Se $\beta < 1$, dà più peso alla Precision (utile quando i falsi positivi sono più problematici).

Se $\beta = 1$, il risultato è l'F1-score, che bilancia Precision e Recall in modo uguale.

ROC Curve e AUC: La Curva Caratteristica Operativa del Ricevitore (ROC) traccia il tasso di veri positivi (TPR) contro il tasso di falsi positivi (FPR) a diverse soglie di classificazione. L'Area Sotto la Curva ROC (AUC) fornisce una misura aggregata delle prestazioni del classificatore. Si cerca di massimizzare i TP (recall) e di minimizzare gli FP (precision)



Metriche di valutazione per modelli di regressione

Per i modelli di regressione, le prestazioni vengono valutate utilizzando l'analisi dei **residui** cioè la differenza tra i valori osservati e quelli previsti e metriche come:

MSE - Mean Squared Error: La media dei quadrati dei residui.

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

Valori più bassi indicano un modello che si adatta meglio ai dati. È molto sensibile agli outliers.

MAE - Mean Absolute Error: La media dei valori assoluti dei residui.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

più robusto rispetto al MSE agli outliers perché non eleva al quadrato l'errore. Anche qui, valori più bassi indicano migliori prestazioni

MAPE - Mean Absolute Percent Error misura l'errore in percentuale rispetto ai valori osservati.

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

È utile quando vuoi capire l'errore relativo, ma non va usato se i valori osservati possono essere zero (o molto vicini a zero) perché diverge.

R-squared (Coefficiente di Determinazione): La proporzione della varianza nella variabile dipendente che è prevedibile dalla variabile indipendente.

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

Va da 0 a 1 (o anche negativo se il modello è peggiore di una media costante). Un valore più alto indica un modello migliore.

F-statistic: Una misura complessiva di quanto la varianza spiegata dal modello superi la varianza residua.

$$F = \frac{\frac{SS_{\text{reg}}}{p}}{\frac{SS_{\text{res}}}{n-p-1}} = \frac{\text{Varianza spiegata dal modello}/p}{(\text{Varianza residua})/(n-p-1)}$$

Un valore alto di F e un corrispondente p-value basso indicano che almeno uno dei predittori è significativo.

dove:

- n = numero di osservazioni
- p = numero di predittori (variabili indipendenti)
- y_i = valore osservato
- \hat{y}_i = valore predetto dal modello
- \bar{y} = media dei valori osservati
- SS_{reg} = somma dei quadrati della regressione (spiegata)
- SS_{res} = somma dei quadrati dei residui (non spiegata)

Valutazione di modelli di clustering

La valutazione di modelli di clustering è meno diretta a causa dell'assenza di un attributo target. Si utilizzano misure di **coesione**, quanto sono simili gli oggetti all'interno di un cluster e **separazione** quanto sono distinti i diversi cluster.

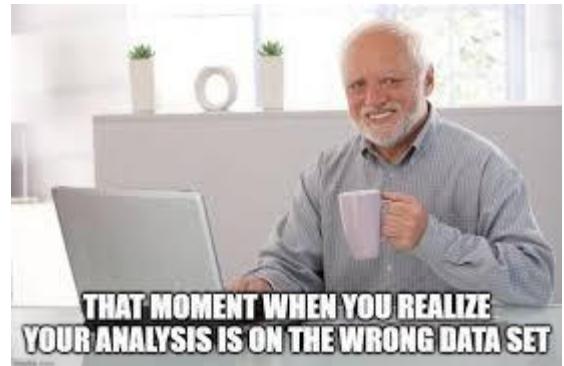
STEP8 - Effettuare previsioni su dati che il modello non ha mai visto

Dopo aver addestrato e valutato il modello, è fondamentale testarlo su dati nuovi, mai utilizzati durante le fasi di training o validazione. Questo step serve per verificare la **capacità di generalizzazione** del modello, ovvero quanto bene riesce a fare previsioni su dati reali, fuori dal campione su cui è stato costruito.

A questo scopo, si utilizza il **test set**, un insieme di dati tenuto da parte fin dall'inizio e completamente invisibile al modello fino a questo punto.

L'analisi delle previsioni su questo set consente di calcolare metriche finali di performance e rilevare eventuali problemi di overfitting.

Se i risultati sono soddisfacenti anche sul test set, allora il modello è pronto per essere **distribuito** e utilizzato in ambienti reali. In caso contrario, potrebbe essere necessario rivedere l'intero processo, a partire dalla raccolta dati o dalla scelta del modello.



CONFRONTO DEI MODELLI

Una volta costruito il modello, dobbiamo vedere se è davvero il migliore che possiamo ottenere, allora confrontiamo i modelli.

Diversi algoritmi possono dare risultati molto diversi sullo stesso problema, e non esiste un modello migliore in assoluto.

Come si fa un confronto tra modelli?

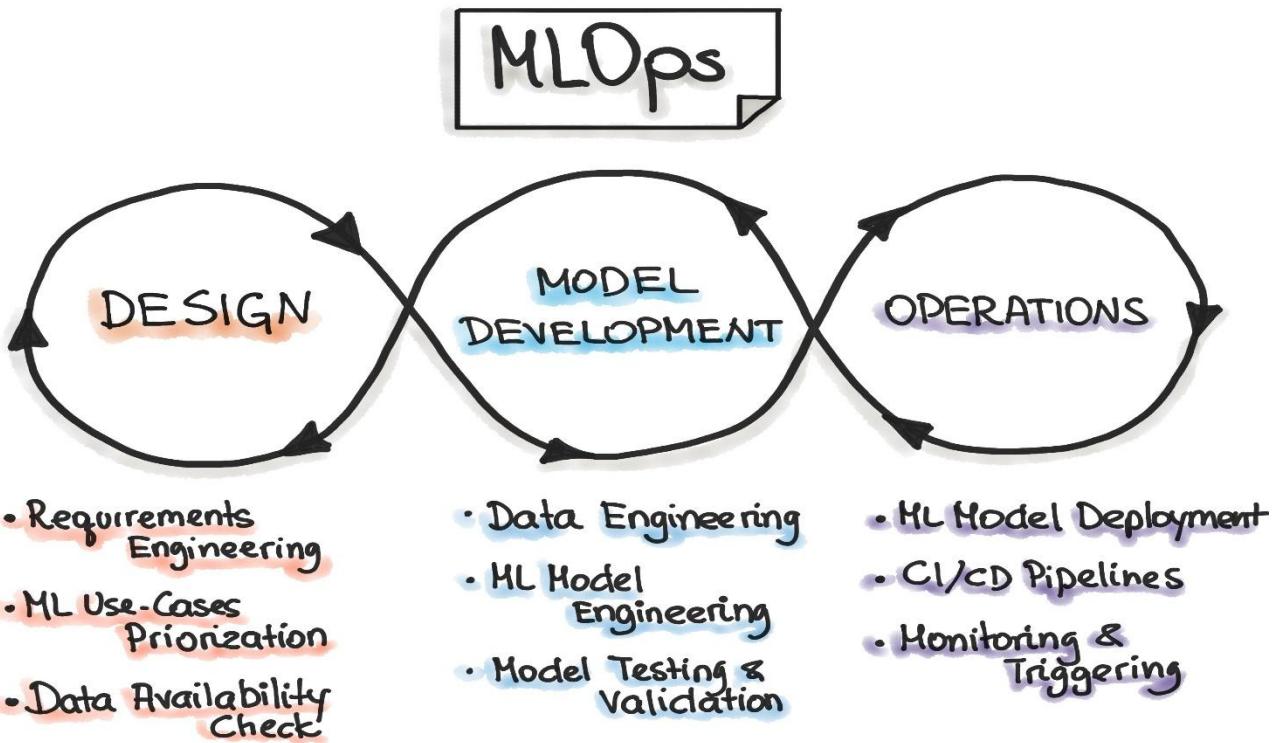
1. **Usiamo le stesse condizioni** Tutti i modelli devono essere testati sugli **stessi dati**, utilizzando le stesse **feature** e lo stesso schema di valutazione
2. **Guardiamo le metriche** Ogni modello produce **delle metriche di valutazione** e a seconda del modello che usiamo, confrontiamo le metriche dei vari algoritmi
3. **Verifichiamo la stabilità** Un modello non deve funzionare bene solo per caso. In questo modo possiamo capire se un modello è veramente robusto o se ha solo avuto "fortuna" con un particolare set di dati.

Spesso il modello più preciso non è quello più utile. Ci sono altri fattori da considerare:

- **Interpretabilità:** in certi ambiti, è fondamentale capire **perché** il modello prende una certa decisione. Un modello semplice come la regressione

logistica può essere preferito a una rete neurale proprio per questo motivo.

- **Efficienza:** alcuni modelli richiedono molto tempo o risorse per essere addestrati.
- **Generalizzazione:** il modello deve saper gestire bene anche nuovi dati, non solo quelli usati per addestrarlo. Se un modello è molto preciso sul training set ma crolla sul test set, non è una buona scelta.



MLOps Machine Learning Operations è un processo che aiuta le organizzazioni e i leader aziendali a generare valore a lungo termine e a ridurre i rischi associati alla data science, al machine learning e alle iniziative di intelligenza artificiale (AI) **standardizzando e la semplificando della gestione del ciclo di vita del machine learning**. La gestione del rischio dei modelli sta diventando sempre più importante a livello aziendale, soprattutto con la crescente **decision automation**.

MLOps trae ispirazione dal concetto di **DevOps**, che semplifica le pratiche di modifica e aggiornamento del software, condividendo principi come **l'automazione robusta, la fiducia e la collaborazione tra i team**, il ciclo di vita end-to-end del servizio (**build, test, release**), e la priorità alla **continuous delivery e all'alta qualità**. Una differenza fondamentale è che l'implementazione di codice software in produzione è diversa dall'implementazione di modelli di machine learning, poiché i modelli sono influenzati da dati in continua evoluzione.

Oltre a DevOps, esiste anche **DataOps**, focalizzato sulla fornitura di dati pronti per l'uso aziendale, con un'attenzione particolare alla qualità dei dati e alla gestione dei metadati.

MLOps è essenziale per qualsiasi team che abbia anche un solo modello in produzione, poiché è fondamentale il **monitoraggio continuo delle prestazioni e l'adeguamento dei modelli**. Permettendo operazioni sicure e affidabili, **MLOps è fondamentale per mitigare i rischi** indotti dall'uso di modelli ML. Senza infrastrutture MLOps, spingere modelli in produzione è rischioso, poiché

spesso **le prestazioni di un modello possono essere valutate completamente solo nell'ambiente di produzione**. Il monitoraggio è essenziale per garantire che il modello si comporti come previsto e per avere una conoscenza precisa di quanto ampiamente venga utilizzato.

MLOps è anche fondamentale per strategie trasparenti di machine learning, consentendo al management di comprendere quali modelli sono in produzione e il loro impatto sul business. Può anche fornire una comprensione approfondita della **data pipeline** dietro questi modelli.

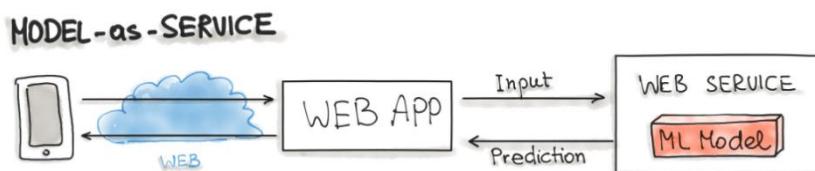
MLOps comprende cinque componenti chiave

- Lo **sviluppo del modello** comporta la definizione degli obiettivi aziendali, l'analisi dei dati, l'ingegnerizzazione delle feature, l'addestramento e la valutazione.
- La **distribuzione** riguarda la produzione e l'implementazione dei modelli, affrontando sfide tecniche come la creazione di artefatti ML e la scelta di strategie di deployment.
- Il **monitoraggio** è cruciale per garantire che i modelli continuino a funzionare bene nel tempo, affrontando preoccupazioni di DevOps, data scientist e business.
- L'**iterazione** implica lo sviluppo e l'implementazione di versioni migliorate dei modelli.
- La **governance** mira a garantire che l'azienda rispetti le proprie responsabilità nei confronti di tutti gli stakeholder, inclusi aspetti finanziari, legali ed etici, con un forte desiderio di equità. Le iniziative di governance in MLOps rientrano in due categorie principali: **data governance**, garantire un uso e una gestione appropriati dei dati e **process governance**, uso di processi ben definiti per garantire che tutte le considerazioni di governance siano affrontate nel punto corretto del ciclo di vita del modello.

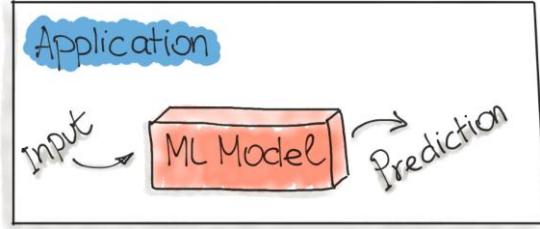
Il deployment del modello si riferisce al processo di **distribuzione di un modello di machine learning in un ambiente di produzione**. Questo è lo stadio finale della consegna di un progetto di ML.

I **Model Serving Patterns** si riferiscono ai diversi modi in cui un modello di machine learning può essere integrato in un sistema software per renderlo disponibile per la generazione di predizioni in un ambiente di produzione.

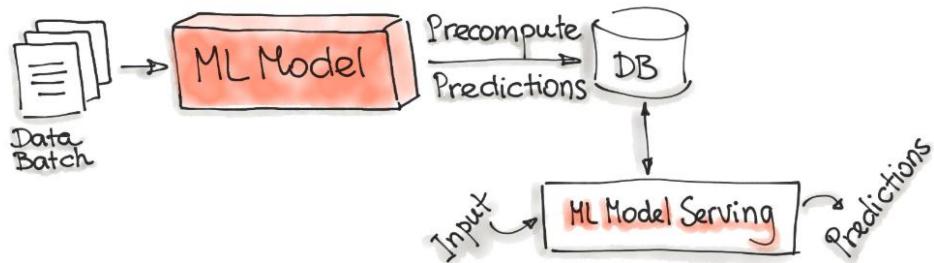
Model-as-Service: Questo è un pattern comune per encapsulare un modello ML come un servizio indipendente. Il modello ML e l'interprete necessario per la sua esecuzione vengono racchiusi all'interno di un **servizio web dedicato**. Le applicazioni possono interagire con questo servizio tramite **API REST**.



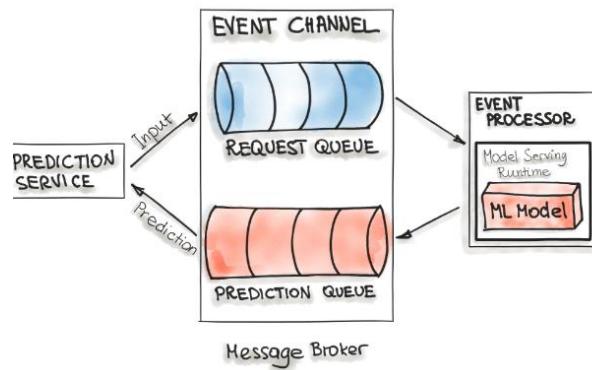
Model-as-Dependency: un modello ML impacchettato è considerato come una dipendenza all'interno dell'applicazione software. L'applicazione consuma il modello ML come una convenzionale dipendenza (ad esempio, un file .jar) invocando il metodo di predizione e passando i valori necessari. Il valore di ritorno di tale esecuzione del metodo è una predizione eseguita dal modello ML precedentemente addestrato.



Precompute: il modello ML è già addestrato per pre-calcolare le predizioni per un batch di dati in entrata. Le predizioni risultanti vengono memorizzate in un database. Quindi, per qualsiasi richiesta in ingresso, il sistema interroga il database per ottenere il risultato della predizione.



Model-on-Demand: il modello ML è una dipendenza disponibile a runtime. Il ciclo di rilascio è indipendente ed è pubblicato separatamente. L'architettura **message-broker** è tipicamente utilizzata per tale serving di modelli on-demand. Questa architettura contiene un componente broker e un processore di eventi. Il broker contiene code di messaggi. Un processo scrive richieste di predizione nella coda di input. Il processore di eventi contiene il runtime del model serving e il modello ML, legge le richieste dalla coda, esegue le predizioni e scrive i risultati nella coda di output, che vengono poi inviati al servizio che ha iniziato la richiesta di predizione.



I modelli vengono impacchettati come servizi distribuibili con varie strategie:

- **Containerizzazione:** La containerizzazione con la quale il modello, con tutte le sue dipendenze e al codice per l'inferenza, viene impacchettato in un container. La funzionalità del modello è quindi accessibile tramite un'**API REST**.
- **Funzioni Serverless:** Vari fornitori di cloud offrono piattaforme di machine learning dove è possibile distribuire i modelli. Per distribuire un modello come funzione serverless, il codice dell'applicazione e le dipendenze vengono impacchettati in file .zip con una singola funzione di entry point gestita dai provider cloud.