[**KDD (KNOWLEDGE DISCOVERY IN DATABASE)** 2](#_Toc7536936)

[**Data Mining** 7](#_Toc7536937)

[**2.1.1** **Tipologie di dati e attributi delle collezioni di dati** 9](#_Toc7536938)

[**2.2** **Classificazione** 9](#_Toc7536939)

[**2.2.1** **Preparazione dei dati** 10](#_Toc7536940)

[**2.2.2** **Valutazione della qualità di un classificatore** 11](#_Toc7536941)

[**2.3** **Descrizione e analisi delle tipologie degli algoritmi di classificazione** 12](#_Toc7536942)

[**2.3.1** **Alberi di decisione** 12](#_Toc7536943)

[**2.3.1.1** **Problematiche di costruzione di un modello: overfitting e underfitting** 14](#_Toc7536944)

[**2.3.1.2** **Funzionamento della classificazione basata sull’albero di** 14](#_Toc7536945)

[**2.3.1.3** **Indici di qualità dell’attributo di split** 15](#_Toc7536946)

[**2.3.1.4** **Qualità della classificazione basata sugli alberi di decisione** 17](#_Toc7536947)

[**2.3.2** **Classificazione basata su regole** 17](#_Toc7536948)

[**2.3.2.1** **Proprietà e problematiche delle regole** 18](#_Toc7536949)

[**2.3.2.2** **Costruzione delle regole** 18](#_Toc7536950)

[**2.3.3** **Classificazione per regole di associazione** 20](#_Toc7536951)

[**2.3.3.1 Qualità della classificazione per regole associative** 21](#_Toc7536952)

[**2.3.4** **Reti neurali** 21](#_Toc7536953)

[**2.3.4.1** **Qualità della classificazione tramite rete neurale** 22](#_Toc7536954)

[**2.3.5** **Classificazione Bayesiana** 23](#_Toc7536955)

[**2.3.6** **Support vector machine** 24](#_Toc7536956)

[**2.3.7.1** **Tipologie di support vector machine** 25](#_Toc7536957)

[**2.3.7.2** **Funzionamento di una SVM** 26](#_Toc7536958)

[**2.3.7.3** **Qualità della classificazione tramite SVM** 27](#_Toc7536959)

[**2.3.7** **K-Nearest Neighbours** 27](#_Toc7536960)

[**2.3.8.1** **Calcolo della distanza tra i vicini** 27](#_Toc7536961)

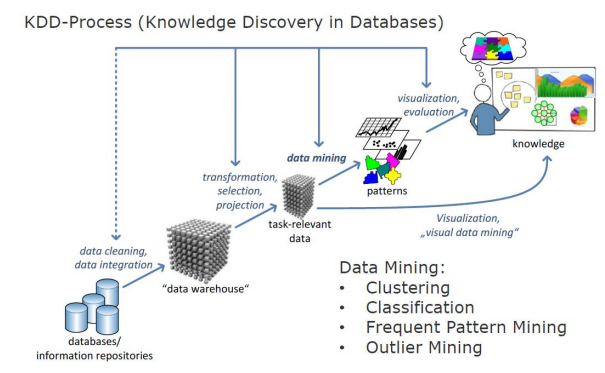
[**2.3.8.2** **Selezione del parametro k** 28](#_Toc7536962)

[**2.3.8.3** **Funzionamento del classificatore k-NN** 28](#_Toc7536963)

[**2.3.8.4** **Qualità di un classificatore k-NN** 29](#_Toc7536964)

## **KDD (KNOWLEDGE DISCOVERY IN DATABASE)**

Il KDD è una procedura interattiva e iterativa [1.2], che cerca di estrarre dai dati informazioni implicite, sconosciute a priori e potenzialmente utili.



Andiamo ad analizzare ora le singole fasi:

* *Identificazione degli obiettivi*: l'oggetto di questa fase è l'individuazione dell'ambito di applicazione in cui deve essere considerato il KDD, inoltre bisogna individuare gli obiettivi da perseguire. Si tratta forse della fase più difficile sia in termini di allocazione risorse sia perché devono essere determinate, in modo preciso, le misure del successo e i criteri per misurare successi e fallimenti. Si può fare una lista solo parziale dei molteplici aspetti che vanno presi in considerazione, alcuni sono il costo stimato del progetto e la scelta degli strumenti di data mining da utilizzare.
* *ETL( Extraction, Trasformation, Loading)*
* *Selezione*: In questa fase deve essere selezionato l'insieme iniziale dei dati, che devono essere sottoposti ad analisi. I dati grezzi vengono segmentati e selezionati secondo alcuni criteri al fine di pervenire ad un sottoinsieme di dati, che rappresentano il nostro target data o dati obiettivo. Infatti è abbastanza evidente che un database può contenere diverse informazioni inutili per il caso di studio in esame. I dati target sono estratti da risorse primarie quali un data warehouse, uno o più database transazionali o uno o più flat file. Se i dati originali sono collocati in un flat file, la creazione del target risulta molto semplice. I sistemi di gestione dei database immagazzinano e manipolano dati transazionali, ciò consente ai sistemi informatici,relativi a tali sistemi, di fare aggiornamenti e di estrarre informazioni in modo rapido. Ciò è dovuto alla strutturazione dei dati tramite modelli relazionali, il cui scopo è ridurre la ridondanza dei dati, tramite la decomposizione di singole tabelle in più strutture relazionali, ed accelerare l'accesso alle informazioni. Del resto lo scopo del DM è proprio utilizzare la ridondanza dei dati per reperire “conoscenza", ecco perché è necessario ricomporre le strutture relazionali. Si intuisce quindi che è stretto il legame tra DM e Data Warehouse, il cui scopo è proprio quello di mettere insieme i dati, e non scomporli, al fine di sfruttarne la ridondanza. Spesso è anche necessario mettere insieme informazioni estratte da più fonti, cosa che può rendere complessa la fase di selezione in quanto bisogna trasformare i dati in modo da assicurare l'omogeneità in quanto, ad esempio, la codifica dei dati deve essere uguale per tutti i record dei dati target, altrimenti l'analisi risulta di scarsa utilità.
* *Preelaborazione*: Generalmente il target data disponibile non deve essere analizzato interamente ma basta estrarne un campione opportuno, eseguendo poi un'analisi su base campionaria. Inoltre i dati devono essere preprocessati, cioè “puliti", trattando in maniera opportuna i dati anomali e mancanti. Vanno individuati i valori errati delle variabili; trovare gli errori nei dati categorici diventa un problema quando si analizzano dataset molto grandi. I dati vanno anche semplificati; queste tecniche di data smoothing sono mirate alla riduzione del numero di valori per una variabile numerica. Alcuni classificatori, come le reti neurali, utilizzano funzioni che effettuano la semplificazione durante il processo di classificazione, eseguendo così un data smoothing interno. Due semplici tecniche di semplificazione sono il calcolo e l'arrotondamento dei valori medi. Lo smoothing del valore medio è appropriato quando si usa un classificatore che non supporta i dati numerici e vogliamo avere un'informazione grossolana sui valori delle variabili numeriche; in tal caso tutti i valori delle variabili numeriche sono sostituiti con la media della classe. La presenza di dati mancanti può essere affrontata in diversi modi; in genere la mancanza di valori è indicativa di informazioni perse. Alcune tecniche di data mining sono in grado di trattare direttamente i valori mancanti tuttavia molti classificatori richiedono che tutte le variabili abbiano dei valori per cui si devono scartare tutti i record contenenti valori mancanti oppure, per valori reali, sostituire i valori che mancano con la media della classe oppure sostituirli con i valori rilevati per altre osservazione molto simili.
* *Trasformazione*: I dati, per essere utilizzati, spesso devono essere trasformati; questa fase può assumere varie forme e può essere necessaria per varie ragioni. Si possono convertire tipi di dati in altri o definirne di nuovi, ottenuti attraverso l'uso di operazioni matematiche e logiche sulle variabili, eseguire delle normalizzazioni (scalamento decimale, normalizzazione min-max o con lo z-score) o addirittura eliminare delle variabili. In genere infatti gli algoritmi di DM non lavorano in modo efficiente se i dati contengono una grande quantità di variabili che non sono in grado di prevedere la classe di appartenenza; si rende quindi utile una ricerca ed una successiva eliminazione delle variabili ridondanti e “inutili" per il problema in questione. A volte le variabili con poco potere previsivo possono essere combinate con altre per formare nuove variabili con un alto grado di capacità previsiva.
* *Data mining*: Ai dati trasformati vengono applicate una serie di tecniche in modo da poterne ricavare dell'informazione non banale o scontata. Sono gli obiettivi che si vogliono raggiungere a dare un'indicazione sul tipo di tecnica/algoritmo che deve essere applicata.
* *Interpretazione e valutazione*: Scopo di questa fase è determinare la validità del modello ottenuto con il DM; in sintesi non basta interpretare i risultati ma bisogna capire in che misura questo modello o risultato possa essere utile. Questo può essere fatto in vari modi sia attraverso un'analisi statistica che euristica o sperimentale.
* *Data Visualization*: L'ultimo obiettivo consiste nell'utilizzare ciò che è stato appreso, creando un report o un rapporto tecnico su ciò che è stato scoperto e cercando di capire in che modo sfruttare ciò che è stato scoperto.

Si capisce bene quindi che il processo di estrazione della conoscenza è lungo e piuttosto articolato, sono fondamentali le scelte che si fanno per il trattamento di anomalie o errori nei dati e l'identificazione chiara degli obiettivi che si vogliono perseguire.

2.1 Strategie del data mining

Le strategie di DM possono essere suddivise in strategie supervisionate e non supervisionate. L'apprendimento supervisionato costruisce modelli tramite

l'utilizzo di attributi di input (variabili indipendenti) per predire i valori degli attributi di output (variabili dipendenti). La maggior parte degli algoritmi di tipo supervisionato ammette un solo attributo di output, solo pochi strumenti consentono di specificare più attributi di output. Quando l'apprendimento non è supervisionato, non esiste un attributo di output; di conseguenza, tutti gli attributi utilizzati per costruire i modelli sono variabili indipendenti. Le strategie di apprendimento supervisionato possono ulteriormente classificate sia in funzione del fatto che gli attributi si output siano discreti o categorici, sia in base al fatto che i modelli siano progettati per determinare una condizione corrente o per predire una situazione futura.

Le principali strategie di apprendimento supervisionato sono la classificazione,

la stima e la previsione. La prima è caratterizzata da una variabile di output di tipo categorico e l'enfasi è riposta sulla costruzione di modelli in grado di assegnare nuovi casi a una di un insieme di classi ben definite. La seconda, analogamente alla classificazione, ha come scopo la determinazione di un valore per un attributo di output incognito; tuttavia gli attributi di uscita sono numerici, anziché categorici. Lo scopo di un modello di previsione, diversamente dai primi due, è determinare il comportamento futuro invece di quello attuale. Qella clusterizzazione non supervisionata invece, non esiste una variabile dipendente che guida il processo di apprendimento ma è il programma di apprendimento che costruisce una struttura di conoscenze utilizzando alcune misure della qualità dei cluster per raggruppare i casi in due o più classi. Un obiettivo primario di tale strategia è scoprire le strutture concettuali dei dati, le applicazioni più comuni della clusterizzazione non supervisionata includono:

• Determinare se i dati presentano relazioni significative nella forma dei concetti.

• Valutare le prestazioni di un modello di apprendimento supervisionato.

• Determinare l'insieme ideale di attributi di input per l'apprendimento supervisionato.

• Identificare gli outlier.

La cosa più interessante è proprio l'impiego della clusterizzazione non supervisionata per stabilire se un modello ottenuto con strategie supervisionate avrà buone o scarse prestazioni. Qualora si presenti la seconda eventualità, una soluzione potrebbe essere quella di rivedere gli attributi e i casi scelti per creare il modello supervisionato. In effetti, è possibile scegliere gli attributi migliori per un modello supervisionato applicando ripetutamente la clusterizzazione con attributi alternativi. In questo modo, è possibile determinare quegli attributi che sono in grado di distinguere meglio le classi note presenti nei dati.

Sfortunatamente, anche quando si hanno a disposizione pochi attributi, l'applicazione di questa tecnica potrebbe risultare difficile da implementare. La clusterizzazione non supervisionata può aiutare anche a individuare alcuni casi anomali presenti nei dati, tali casi sono detti outlier e possono rivestire una grande importanza per cui dovrebbero sempre essere identificati. Uel data mining, infatti, gli outlier potrebbero essere proprio quei casi che stiamo cercando di identificare; l'obiettivo di un'analisi di DM può essere quello di riuscire a identificare delle situazioni anomale.

Infine all'interno delle strategie di DM è fondamentale la marhet bashet analysis, il cui scopo è ricercare relazioni interessanti tra i prodotti delle vendite al dettaglio. I risultati di tale analisi aiutano i commercianti a ideare le campagne promozionali, a disporre i prodotti negli scaffali, a presentarli nei cataloghi e a sviluppare strategie di marheting incrociato.

**Data Mining**

Con il termine Data Mining si identificano un insieme di tecniche di analisi che permettono un processo automatico di estrazione di informazioni utili, chiamati **pattern**, da collezioni di dati (o **dataset**). Queste tecniche di estrazione automatica si basano sull’utilizzo di diversi algoritmi. [1]



Figura 2.1 Processo di estrazione di informazioni utili dai dati

Per poter arrivare all’estrazione di pattern è necessario seguire un processo diviso in varie fasi, chiamato **knowledge discovery process**, che inizialmente parte dalla selezione dei dati da sottoporre ad analisi dalle diverse fonti disponibili, come ad esempio dei dati presi dalle basi di dati. Successivamente si passa in una fase di fondamentale importanza per l’accuratezza finale dei pattern estratti, la fase di **preprocessing**. Infatti, questa fase permette di migliorare la qualità dei dati ad esempio risolvendo delle inconsistenze che questi possono presentare, rimuovendo eventuali dati anomali (**outliers**) non utili all’analisi, così chiamati poiché si differenziano dal resto dei dati all’interno del dataset, e risolvendo eventuali conflitti presenti tra i dati dopo l’integrazione delle diverse fonti. Questa fase di preprocessing è seguita da una di trasformazione che consente di preparare i dati per alla fase successiva di Data Mining, grazie alla quale, come precedentemente detto, è possibile estrarre i pattern dai dati applicando degli specifici algoritmi. Infine, dopo una fase di interpretazione e valutazione dei pattern estratti, si passa alla fase finale chiamata **knowledge**, in cui i dati finali delle analisi vengono presentati.

Attualmente, i contesti di utilizzo delle tecniche di Data Mining sono diversi: alcuni

esempi sono l’analisi di dati estratti dai social network, la comprensione del

comportamento di utenti in siti Web, come ad esempio gli e-commerce, l’analisi dei testi (**text mining**), che permette la conoscenza di informazioni utili estratte dai documenti, l’analisi di dati clinici, la fornitura di servizi alle persone attraverso l’uso della loro posizione e vari altri contesti.

* + 1. **Tipologie di dati e attributi delle collezioni di dati**

Esistono diverse tipologie di collezione di dati tra cui comunemente si trovano quelle costituite da dati caratterizzati da diversi attributi. In una collezione di dati di questo tipo, ogni dato presenta i valori per ogni attributo. Ogni attributo ne rappresenta una determinata caratteristica e ne esistono diverse tipologie tra cui troviamo quelli:

* + - * Nominali: presentano un valore che corrisponde ad una caratteristica (es. i colori)
      * Ordinali: i valori di un attributo di questo tipo sono ordinati tra loro
      * Numerici (o contigui): presentano valori numerici interi reali
      * Binari: gli attributi di questo tipo sono costituiti da due soli valori

Nelle collezioni di dati, può capitare che non tutti i dati presentano tutti i valori per tutti gli attributi che li caratterizzano. Infatti, spesso nei dataset è possibile identificare dei dati che presentano i cosiddetti **missing value**, valori mancanti per un determinato attributo. Una collezione di dati caratterizzata da questi valori mancanti può influenzare la qualità delle analisi che verranno effettuate su di essa; infatti, durante il processo di analisi o viene eliminato l’intero dato che li contiene oppure il dato viene mantenuto, ma i valori mancanti vengono sostituiti da altri valori.

* 1. **Classificazione**

Nel Data Mining, un processo di classificazione consiste nell’assegnazione di etichette di classe ai dati non ancora etichettati. Dato un insieme di classi, individuate per mezzo di **etichette di classe**, e una collezione di dati precedentemente classificati, l’obiettivo è quello di generare un **modello di classificazione** che consenta di assegnare le corrette etichette di classe ai dati non ancora etichettati. Un modello di classificazione viene generato utilizzando uno specifico **algoritmo di classificazione** a cui in input vanno dei **dati di training** (o **training set**) che consistono in una collezione di dati, chiamati **istanze**, costituiti da un insieme di valori per un insieme finito attributi e da un valore per l’attributo di classe. Dall’analisi dei dati di training viene generato il modello di classificazione e questo conclude la prima fase del processo di classificazione. Successivamente, utilizzando questo modello di classificazione è possibile assegnare ai dati non ancora etichettati, denominati **dati di test** (o **test set**), le rispettive etichette di classe. Tutto il procedimento spiegato è mostrato nello schema della Figura 2.2:

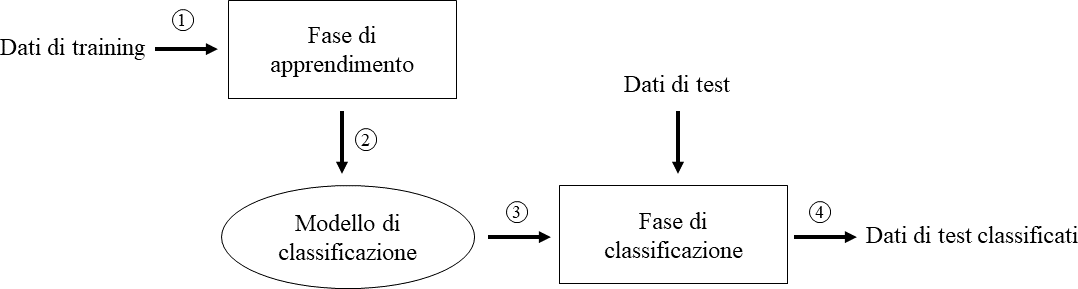


Figura 2.2 Schema classificazione

Dalla Figura 2.2 si evince che dopo una fase di apprendimento utilizzando come input le istanze di training viene generato un modello di classificazione. Questo prende in ingresso le istanze di test e viene avviata la fase di classificazione. Alla fine di tutto il processo di analisi tutte le istanze di test vengono etichettate tramite un’etichetta di classe.

La classificazione è divisa in due tipologie: **supervised** e **unsupervised** (o **clustering**). Nella classificazione di tipo supervised l’insieme delle possibili etichette di classi è conosciuta a priori, quindi, nel processo di classificazione, alle istanze di test viene assegnata un valore di etichetta di classe tra quelle indicate a priori. Invece, nella classificazione di tipo unsupervised non ci sono etichette di classe note a priori, ma solo dopo aver effettuato la classificazione, dividendo i dati in gruppi secondo attributi simili, è possibile dare un nome ai diversi gruppi generati e creare così le etichette di classe. Quest’ultima è meglio nota sotto il nome di **clustering**.

* + 1. **Preparazione dei dati**

Per poter generare un modello di classificazione dotato di una buona qualità nel classificare le istanze di test, occorre effettuare una fase di preprocessing della collezione di dati in input. Infatti, i dati che vengono solitamente analizzati provengono da diverse fonti e possono quindi presentare ridondanze, dati anomali o mancanti, rumori o conflitti. Per limitare l’influenza di tutte queste problematiche sul risultato finale della classificazione occorre utilizzare delle tecniche di preprocessing.

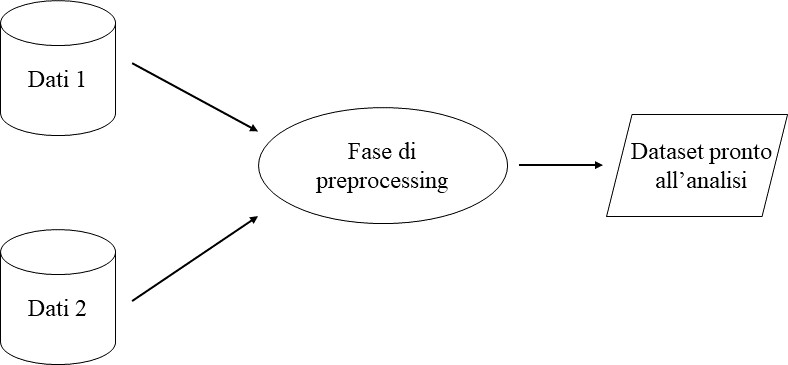


Figura 2.3 Schema preprocessing

Nella Figura 2.3 è mostrato uno schema in cui i dati vengono prelevati da due diverse fonti indicate come “Dati 1” e “Dati 2” e, dopo una fase di preprocessing, vengono generata una collezione di dati pronti per la fase successiva del processo di analisi.

* + 1. **Valutazione della qualità di un classificatore**

L’obiettivo finale della classificazione è quello di etichettare le istanze di test grazie all’utilizzo del modello di classificazione creato dai dati di training. Quindi, occorre valutare il comportamento di un classificatore utilizzando alcune misure di qualità. Solitamente, la qualità di un classificatore viene valutata in termini di:

* + - * **Accuratezza**: indica la qualità della classificazione dei dati non etichettati; è definita come la percentuale di dati correttamente classificati rispetto al numero totale di dati sottoposti a classificazione;
      * **Efficienza**: tempo di generazione del modello di classificazione utilizzando i dati di training e tempo di classificazione dei dati di test;
      * **Scalabilità**: come si comporta un classificatore al variare della dimensione e del numero di attributi del training set;
      * **Robustezza**: indica quanto un classificatore è in grado di essere robusto in presenza di collezioni di dati contenenti missing value o **rumori**, definiti come istanze di training etichettate in modo errato;
      * **Interpretabilità**: livello di interpretabilità del modello di classificazione.
  1. **Descrizione e analisi delle tipologie degli algoritmi di classificazione**

Nello stato dell’arte esistono diverse tecniche di classificazione di tipo supervised, tra le quali sono presenti: alberi di decisione, regole di classificazione, regole di associazione, reti neurali, naїve bayes e reti bayesiane, k-nearest neighbours (k-NN), support vector machine (SVM). Nei paragrafi successivi verranno descritte e analizzate.

* + 1. **Alberi di decisione**

La tecnica di classificazione basata sull’albero di decisione prevede la costruzione di un

albero che presenta tre principali elementi: i **nodi**, gli **archi** e le **foglie**. [2]

Un nodo è costituito da un **attributo di split**, attributo scelto tra quelli dei dati di training secondo una strategia di tipo greedy, miglior attributo locale in grado di produrre una distribuzione omogenea dei dati di training in accordo con le etichette di classe. Ogni **arco** consente la connessione tra un nodo padre e un nodo figlio e presenta uno specifico valore dell’attributo di split presente nel nodo padre. Ogni **foglia**, invece, rappresenta un’etichetta di classe.

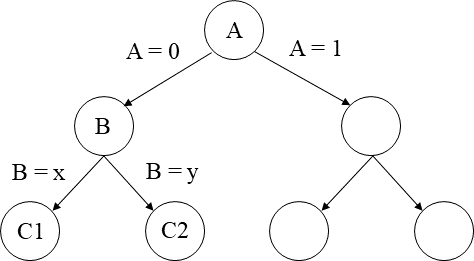


Figura 2.4 Albero di decisione

Nella Figura 2.4 è mostrato un esempio di albero di decisione in cui il nodo radice rappresenta l’attributo A dell’insieme di dati di training e i due archi uscenti dal questo nodo presentano rispettivamente uno tra i possibili valori che l’attributo A può assumere. A seconda del valore che un’istanza di test presenta per l’attributo A verrà percorso l’arco di sinistra o quello di destra dell’albero di decisione durante la fase di classificazione. Inoltre, percorrendo il ramo di sinistra si incontra il nodo figlio B del nodo radice che a sua volta genera due nodi foglia che costituiscono due valori possibili valori di etichette di classe: C1 e C2.

Ogni nodo dell’albero di decisione è una condizione da testare per ogni istanza dei dati di test e può presentare una struttura di due tipi: **multi-way split**, che consente di generare un arco per ogni valore distinto dell’attributo scelto per lo split o **binary split** che permette di dividere i valori dell’attributo di split soltanto in due sottoinsiemi. Nella Figura 2.5 sono mostrati le due possibili strutture di un nodo.

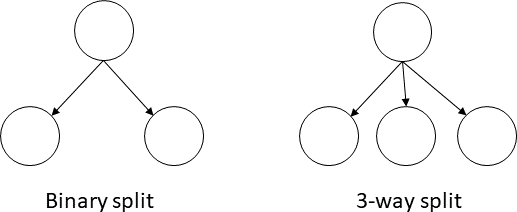


Figura 2.5 Strutture di un nodo

Anche se un attributo può assumere più di due valori, ma si sceglie di effettuare un split di tipo binario, i valori possono essere raggruppati in diversi modi per poter consentire lo split binario.

La costruzione del modello di classificazione ad albero differisce a seconda dell’algoritmo utilizzato e ne esistono diversi tra cui: algoritmo di Hunt, ID3, C4.5, C5.0, SLIQ, SPRINT, CART.

La costruzione di un albero di decisione avviene grazie all’utilizzo della collezione dei dati di training. Inizialmente, analizzando i dati di training, viene selezionato il migliore attributo di split per il nodo radice, utilizzando degli indici di qualità che saranno introdotti nel Paragrafo 2.3.1.4. In seguito, lo split viene eseguito sull’attributo scelto e i dati vengono divisi in un numero di sottoinsiemi pari al numero di archi creati dallo split.

Questo procedimento viene ripetuto in maniera sui vari sottoinsiemi dei dati di training che si vengono a creare dagli split, fino a quando non viene raggiunta una **condizione di fine** dell’algoritmo utilizzato; ad esempio fino a quando non ci sono più dati da separare. A questo punto la generazione dell’albero di decisione è completa.

* + - 1. **Problematiche di costruzione di un modello: overfitting e underfitting**

La costruzione di un modello di classificazione ad albero di decisione può portare a delle problematiche note. Una tra queste è il fenomeno dell’**overfitting** che si viene a creare quando il modello di classificazione è troppo specializzato per i dati di training. Questo porta alla creazione di un albero di decisione complesso che presenta un numero elevato di nodi. All’aumentare del numero di nodi il valore di accuratezza della classificazione aumenta se si usano come dati di test le stesse istanze di training. Contrariamente, utilizzando delle istanze diverse come dati di test, il valore di accuratezza diminuisce all’aumentare del numero di nodi. Il fenomeno di overfitting può essere in parte risolto utilizzando le seguenti tecniche di potatura dell’albero di decisione:

* + - * + **Pre-pruning**: in una tecnica di questo tipo si parte dall’idea di interrompere la costruzione dell’albero di decisione quando si raggiungono determinate condizioni. Normalmente un algoritmo di costruzione dell’albero di decisione si arresta quando viene raggiunta una condizione di fine, come ad esempio quando tutte le istanze nei vari sottoinsiemi appartengo alla stessa classe. In una tecnica di pre-pruning si cerca di scegliere una soglia in modo tale da interrompere la costruzione dell’albero quando il numero di istanze dei vari sottoinsiemi si trova al di sotto di tale soglia scelta;
        + **Post-pruning**: considerando questa tipologia di soluzione, invece, l’albero di decisione viene interamente costruito e poi viene potato tenendo conto del miglioramento delle prestazioni dopo le eventuali potature dei nodi apportate.

Utilizzando in maniera eccessiva queste tecniche utili alla riduzione dell’overfitting si può incorrere ad un’altra problematica, l’**underfitting**. Infatti, si parla di underfitting quando il modello di classificazione è stato generato in maniera troppo semplicistica ed il processo di classificazione sia dei dati di training che del test set porta in entrambi i casi ad un errore di classificazione elevato, quindi ad una bassa accuratezza nella classificazione.

* + - 1. **Funzionamento della classificazione basata sull’albero di**

**decisione**

Per effettuare una classificazione utilizzando un albero di decisione occorre innanzitutto una fase di creazione del modello utilizzando le istanze dei dati di training. Successivamente è possibile classificare le istanze di test grazie al modello di classificazione precedentemente creato. Un esempio di funzionamento della classificazione basata sull’albero di decisione è il seguente:

Dato il modello di classificazione ad albero di decisione di esempio e le due istanze di test mostrate in Figura 2.6:

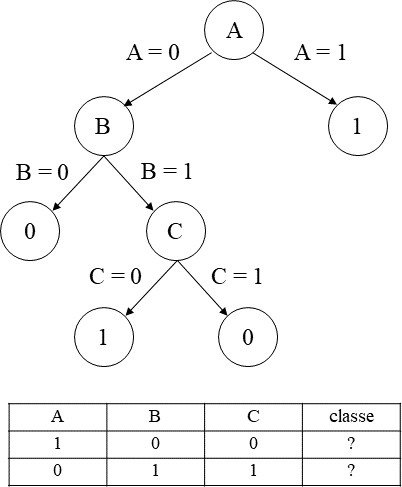


Figura 2.6 Esempio di classificazione basata sull’albero di decisione

Le due istanze di test mostrate in Figura 2.6 sono costituite da quattro attributi “A”, “B”, “C”, “classe”. Durante la fase di classificazione, la prima istanza di test, dato che per l’attributo A presenta un valore uguale a 1, viene direttamente classificata con l’etichetta di classe 1, perché percorre l’arco di destra uscente dal nodo radice. Quindi, assumerà un valore 1 per l’attributo “classe”. Invece, considerando la seconda istanza da etichettare, percorre l’arco sinistro che collega il nodo radice A con il nodo che rappresenta l’attributo di split B. In questo nodo viene testato il valore che assume l’istanza per l’attributo B che in questo caso è pari a 1, quindi non si raggiunge ancora un nodo foglia ed è necessario un ulteriore test sull’attributo C. Dopo questo ultimo test, l’istanza viene classificata con l’etichetta di classe 0, poiché il suo valore per l’attributo C è 1 e viene attraversato l’arco destro per raggiungere il nodo foglia. Il suo attributo “classe”, quindi, assumerà il valore 0.

* + - 1. **Indici di qualità dell’attributo di split**

La selezione dell’attributo di split è un’operazione critica, in quanto bisogna scegliere tra tutti gli attributi quelli che consentono di produrre una distribuzione omogenea dei dati di training secondo le etichette di classe. Per valutare la qualità dello split occorre una **misura di impurità**, definita come la frequenza delle etichette di classe in quel determinato split. Un split presenta un elevato livello di qualità, quindi distribuzione omogenea in accordo con le etichette di classe, quando più basso è il suo livello di impurità. Esistono diverse misure dell’impurità di uno split, ognuna di queste viene utilizzata in base dell’algoritmo di costruzione dell’albero di decisione scelto. Tra gli

indici che misurano l’impurità di uno split sono presenti: Gini index, entropia ed errore di classificazione.

Il **Gini index** misura il livello di non omogeneità della distribuzione dei dati di training secondo le etichette di classe ed è un numero compreso tra 0 e 1. Più basso è il valore di questo indice più la distribuzione dei dati è omogenea, quindi indica un basso livello di impurità e conseguentemente un’alta qualità dello split. L’indice di Gini per un dato nodo

𝑛 è definito come:

𝐺𝐼𝑁𝐼(𝑛) = 1 − ∑[𝑝(𝑗|𝑛)]2

𝑗

dove 𝑝(𝑗|𝑛) è la frequenza della classe 𝑗 al nodo 𝑛.

Il valore massimo indica che tutte le istanze sono equamente distribuite tra tutte le classi e comporta un alto livello di impurità, invece, il valore minimo corrisponde ad un basso livello di impurità poiché tutte le istanze appartengono ad un'unica classe.

Quando un nodo *x* su cui occorre effettuare uno split genera *k* archi, quindi *k* nodi figli, la qualità dello split è calcolata:

𝐺𝐼𝑁𝐼𝑠𝑝𝑙𝑖𝑡

𝑘

𝑛𝑖

= ∑

𝑛

𝑖=1

𝐺𝐼𝑁𝐼 (𝑖)

dove 𝑛𝑖 è il numero di record al nodo figlio 𝑖 e 𝑛 è il numero di record del nodo *x* scelto.

Questo indice è utilizzato da algoritmi per la costruzione del modello basato sull’albero

di decisione come CART, SLIQ e SPRINT.

L’**entropia**, come l’indice di Gini, misura il livello di impurità di un nodo, ma consente di raggiungere il valore più basso più velocemente se più nodi figli appartengono alla stessa classe. Ad un dato nodo *n*, l’entropia è definita come:

𝐸𝑁𝑇𝑅𝑂𝑃𝐼𝐴(𝑛) = − ∑ 𝑝(𝑗|𝑡)log 𝑝(𝑗|𝑛)

𝑗

dove 𝑝(𝑗|𝑡) è la frequenza della classe 𝑗 al nodo 𝑛*.*

Come nell’indice di Gini, un valore elevato dell’entropia corrisponde ad un alto livello di

impurità, invece, uno basso indica un basso livello di impurità.

Dato un nodo padre 𝑝, il cui split genera 𝑘 partizioni, si sceglie lo split che massimizza il seguente valore di guadagno, chiamato **information gain**:

𝑘

𝐺𝐴𝐼𝑁𝑠𝑝𝑙𝑖t

= 𝐸𝑁𝑇𝑅𝑂𝑃𝐼𝐴(𝑝) − (∑ 𝑛𝑖

𝑛

𝑖=1

𝐸𝑁𝑇𝑅𝑂𝑃𝐼𝐴(𝑖))

dove 𝑛𝑖 è il numero di record nella partizione 𝑖.

Questo indice di guadagno viene utilizzato da algoritmi come ID3 e C4.5, ma presenta lo svantaggio di tendere a preferire la creazione di un numero elevato di partizioni piccole ma con un basso livello di impurità. Per superare questo svantaggio è stato definito un altro guadagno, il **gain ratio**, definito come:

𝐺𝐴𝐼𝑁𝑟𝑎𝑡𝑖𝑜 =

𝐺𝐴𝐼𝑁𝑠𝑝𝑙𝑖𝑡

− ∑𝑘 𝑛𝑖 log 𝑛𝑖

𝑖=1 𝑛 𝑛

dove il denominatore consente di evitare la creazione di tante piccole partizioni. L’**errore di classificazione** misura il numero di record erroneamente classificati ad un certo nodo 𝑛*,* ed è definito come:

𝐸𝑅𝑅𝑂𝑅𝐸(𝑛) = 1 − max 𝑝(𝑖|𝑛)

dove 𝑝(𝑖|𝑡) indica la frequenza della classe 𝑖 al nodo 𝑛.

Secondo l’errore di classificazione uno split su un nodo è di qualità maggiore quando il numero di errori minore per quel nodo. Come le due misure di impurità descritte in precedenza, un elevato valore di questo indice indica un numero elevato di errori, quindi i record sono distribuiti in modo equo tra le classi ed il valore di impurità è alto. Un valore basso dell’errore di classificazione, invece, indica un basso livello di impurità.

* + - 1. **Qualità della classificazione basata sugli alberi di decisione**

Come discusso nel Paragrafo 2.2.2, la valutazione della qualità delle tecniche di classificazione viene effettuata in base a determinate caratteristiche come accuratezza, efficienza, scalabilità, interpretabilità e robustezza. La tecnica di classificazione basata sugli alberi di decisione presenta, come tutte le altre tecniche, dei vantaggi e degli svantaggi. Tra i vantaggi ci sono:

* + - * + Efficienza: rapida costruzione del modello di classificazione, poiché non esplora tutte le possibilità ma opera secondo un approccio greedy (ottimo locale per ogni nodo dell’albero). Inoltre presenta una veloce classificazione di istanze di test, dato che occorre prendere un’istanza e propagarla nel modello ad albero per effettuarne la classificazione.
        + Interpretabilità: modello facile da interpretare se è di piccole dimensioni
        + Accuratezza: quando il problema di classificazione è semplice, l’accuratezza è

comparabile ad altri metodi di classificazione

* + - * + Scalabilità

Uno svantaggio, invece, è il basso livello di robustezza, visto che in presenza di missing value l’accuratezza potrebbe diminuire.

* + 1. **Classificazione basata su regole**

La classificazione basata su regole si basa sulla costruzione e sull’utilizzo di regole costituite da una o più **condizioni** e da un’etichetta di classe, chiamata anche **conseguente della regola**. [2] Un esempio di una regola di classificazione è mostrato di seguito:

(Condizioni) -> etichetta di classe

Le condizioni di una regola corrispondono ad uno o più attributi scelti tra quelli presenti tra i dati di training associati ai corrispondenti valori. Su questi attributi si effettua il test e sono tutti posti tra loro in una condizione di AND logico. Se tutte le condizioni di una determinata regola sono soddisfatte, allora viene assegnata l’etichetta di classe di quella regola all’istanza su cui viene effettuato il test.

Un esempio di regole di associazione è il seguente, in cui sono mostrate due diverse regole R1 e R2:

R1: *attributo-1 = A AND attributo-2 = A -> etichetta di classe = 0*

R2: *attributo-1 = A AND attributo-2 = B -> etichetta di classe = 1*

Un esempio di applicazione di queste regole può essere effettuato considerando la seguente istanza di test:

|  |  |  |
| --- | --- | --- |
| **attributo-1** | **attributo-2** | **etichetta di classe** |
| A | A | ? |

Questa istanza soddisfa la regola di classificazione R1, quindi l’etichetta di classe

assegnata sarà 0.

* + - 1. **Proprietà e problematiche delle regole**

Le regole possono presentare le due proprietà seguenti:

* + - * + Mutualmente esclusive: quando ogni istanza soddisfa una e una sola regola
        + Esaustive: ogni istanza soddisfa almeno una regola

Solitamente le regole sono sia mutualmente esclusive che esaustive, ma nel caso in cui una di queste proprietà non fosse soddisfatta si verrebbero a creare delle problematiche per la classificazione dei dati di test. Infatti, se le regole non fossero mutualmente esclusive, più di una regola soddisferebbe una determinata istanza di test; in questo caso si presenta il problema di quale regola occorre scegliere. Esistono diverse strategie da poter adottare in questi casi, tra cui è possibile utilizzare una soluzione chiamata **ordered rule set**, che consente di ordinare le regole di classificazione in base ad una priorità. Quindi, quando un’istanza di test soddisfa due o più regole verrà considerata la regola a priorità più elevata e all’istanza verrà assegnata l’etichetta di classe di quella regola. Se, invece, le regole non fossero esaustive, un’istanza potrebbe non soddisfare alcuna regola presente nell’insieme delle regole create. Una soluzione a questo problema è l’utilizzo di un’etichetta di classe di default in modo che se nessuna regola fosse soddisfatta verrebbe assegnata questa particolare etichetta di classe a quell’istanza di test.

* + - 1. **Costruzione delle regole**

Le regole di classificazione possono essere costruite secondo due differenti tecniche: metodo diretto e indiretto (o induttivo).

Utilizzando un **metodo diretto**, le regole vengono generate direttamente a partire dalla collezione dei dati di training. Tra gli algoritmi che utilizzano questo metodo ci sono RIPPER e CN2. Per la costruzione delle regole, utilizzando questo metodo, le istanze di training vengono analizzate ad una ad una e viene generata una regola. Quando viene creata una regola, l’istanza da cui viene generata viene eliminata. Dopo la generazione di tutte le regole ne viene fatta un’ottimizzazione.

Invece, utilizzando un metodo di tipo **indiretto** (o **induttivo**), le regole vengono estratte da un esistente modello di classificazione, generato utilizzando ad esempio un algoritmo classificazione basato sull’ albero di decisione. L’algoritmo maggiormente utilizzato per la generazione delle regole secondo questa tecnica è il C4.5rules. Considerando quindi un modello di classificazione ad albero di decisione, in questa tipologia di estrazione delle regole, per ogni possibile percorso esistente tra il nodo radice e le foglie dell’albero viene creata una nuova regola. Come condizioni per una determinata regola vengono utilizzati tutti gli attributi di split attraversati in un percorso con i rispettivi valori, posti in AND logico tra loro. Come etichetta di classe della regola viene preso il valore del nodo foglia associato al percorso attraversato.

Considerando il modello di classificazione ad albero di decisione in Figura 2.7 si può presentare un esempio di generazione delle regole che utilizza una tecnica di costruzione indiretta:

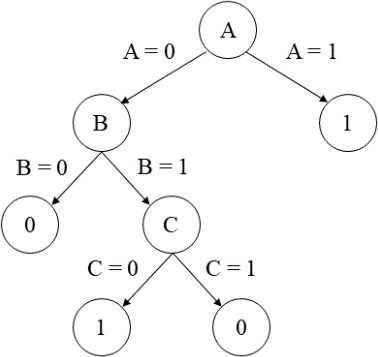


Figura 2.7 Albero di decisione da cui derivare le regole

Dall’albero di decisione è possibile generare quattro diverse regole, corrispondenti a tutti i percorsi esistenti tra il nodo radice e i nodi foglia. Le regole che verranno generate sono le seguenti:

R1: *A = 1 -> 1*

R2: *A = 0 AND B = 0 -> 0*

R3: *A = 0 AND B = 1 AND C = 0 -> 1*

R4: *A = 0 AND B = 1 AND C = 1 -> 0*

* + 1. **Classificazione per regole di associazione**

La classificazione associativa permette la classificazione di dati utilizzando regole estratte da coppie attributo-valore più frequenti, denominato **frequent itemset**, dalla collezione di dati da analizzare. [2] La creazione di un classificatore di questo tipo si basa su due principali fasi: ricerca delle coppie attributo-valore, chiamati **item**, più frequenti e successiva generazione delle regole di associazione. In un primo momento, si scorre tutta la collezione di dati considerando tutti gli item più frequenti, poi tra questi se ne selezionano soltanto un sottoinsieme che soddisfano dei criteri di qualità ed infine si generano le regole di associazione. I criteri di qualità che vengono utilizzati sono: la **confidenza** e il **supporto**.

La confidenza tra due valori degli attributi A e B è definita come la probabilità di trovare

B in un’istanza avendo già trovato A.

Invece, il supporto è definito come il rapporto tra il numero di istanze che contengono sia A che B e il numero totale delle istanze nella collezione di dati. Questi parametri di qualità vengono calcolati per ogni coppia attributo-valore frequente e si selezionano soltanto quelle coppie che hanno, per questi parametri, dei valori al di sopra o almeno pari ad una determinata soglia definita. In questo modo, si considerano soltanto un sottoinsieme di coppie attributo-valore da cui si generano le regole di associazioni utilizzate per effettuare la classificazione. Per la selezione delle coppie attributo-valore più frequenti si parte dall’idea che data una collezione di dati da analizzare di dimensione *N* ne derivano *2N* itemset candidati. Per poter generare il frequent itemset si può adottare un approccio di tipo brute force, ma non è efficiente, quindi si cerca di adottare delle tecniche che consentano di ridurre il numero di itemset candidati alla creazione del frequent item set

finale. Per effettuare questa riduzione si utilizza l’algoritmo **FP-growth**, che sfrutta una

rappresentazione ad albero dell’insieme di dati da analizzare chiamato **FP-tree**.

**2.3.3.1 Qualità della classificazione per regole associative**

La classificazione basata su regole associative permette di ottenere un elevato livello di accuratezza, in quanto si considerano le correlazioni tra i vari attributi della collezione di dati da analizzare. Inoltre, questa tecnica di classificazione è caratterizzata da un’alta scalabilità ed interpretabilità. Un classificatore di questo tipo presenta un’elevata robustezza, poiché le regole di associazione vengono create anche in mancanza di alcuni valori. Di contro, è poco scalabile su istanze che presentano un elevato numero di attributi ed è caratterizzato da una bassa efficienza; infatti, la generazione di buone regole associative è lunga e complessa poiché dipende dalla soglia di supporto scelta.

* + 1. **Reti neurali**

Le reti neurali sono una tecnica di classificazione la cui struttura è composta da un insieme di nodi interconnessi tra loro che ne costituiscono l’unità computazionale. [2] All’interno di una rete neurale sono sempre presenti due tipologie di nodi: **nodi di input**, prendono in ingresso le istanze sia per la fase di apprendimento della rete che per quella di classificazione, e **nodi di output** che assegnano la classe all’istanza sotto analisi. Oltre a questi due tipi di nodi possono essere presenti anche degli altri, chiamati **nodi nascosti**, organizzati in vari livelli. Un modello di rete neurale è mostrato in Figura 2.8:

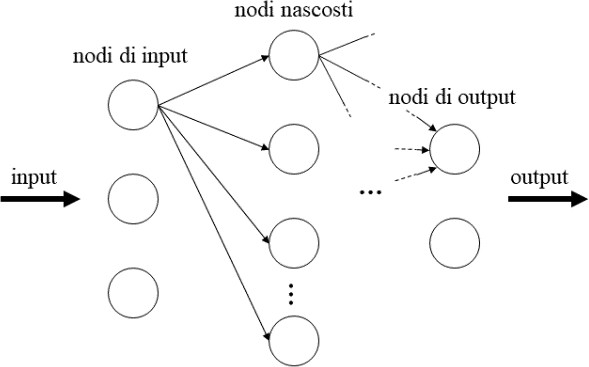


Figura 2.8 Modello di una rete neurale

In definitiva, una rete neurale è composta da vari livelli: da un primo livello costituito da un insieme di nodi di input, da uno o più livelli di insiemi di nodi nascosti e da un ultimo livello composto da un insieme di nodi di output.

In un modello a rete neurale sono associate altre due caratteristiche: pesi e offset. Ad ogni connessione tra i vari nodi che formano una rete neurale è associato un **peso** che inizialmente assume un valore casuale, ma durante la fase di apprendimento della rete viene continuamente aggiornato a seguito di un errore di classificazione di un’istanza rispetto al valore di classe previsto. Inoltre, ad ogni nodo è associato anche un valore di **offset**.

In una rete neurale gli input inviati ai nodi di input sono costituiti dal valore degli attributi dell’istanza che deve essere analizzata. L’output di questo primo livello della rete rimane invariato, poiché in uscita dai nodi di input sono presenti gli stessi valori che vengono forniti in input. In ogni nodo appartenente ai livelli successivi al primo, nodi nascosti e nodi di output, avviene l’effettiva computazione. Infatti, gli input di questi livelli corrispondono agli output dei livelli precedenti in cui però bisogna considerare il peso associato al collegamento tra i due nodi ed un valore caratteristico del nodo, l’offset. Considerando un nodo 𝑛 tra i nodi nascosti o tra quelli di output, il suo input 𝐼𝑛 è dato dalla seguente relazione:

𝐼𝑛 = ∑ 𝑤𝑖,𝑛 𝑂𝑖 + 𝑜𝑓𝑓𝑠𝑒𝑡𝑛

𝑖

dove 𝑤𝑖,𝑛 è il peso del collegamento tra il nodo 𝑖 del livello precedente e il nodo 𝑛 preso in considerazione, 𝑂𝑖 è l’output del nodo 𝑖 del livello precedente e 𝑜𝑓𝑓𝑠𝑒𝑡𝑛 è l’offset associato al nodo 𝑛 considerato.

Ogni nodo, inoltre, applica poi una **funzione di attivazione** sul valore che riceve in input ed invia il suo output al livello successivo. Infine, quando viene generato l’output dai nodi di output, se durante la fase di apprendimento si verifica un errore tra il valore della classe calcolato e quello previsto per un’istanza, viene calcolato l’errore da ogni nodo di output e viene propagato ai livelli precedenti, dove vengono sistemati i valori dei pesi e degli offset di tutti i nodi di tutti i livelli che costituiscono la rete neurale.

* + - 1. **Qualità della classificazione tramite rete neurale**

Tra tutte le diverse tecniche di classificazione, le reti neurali sono tra quelle con il più alto livello di accuratezza. Per raggiungere tale livello di accuratezza occorre un tempo di training della rete neurale lungo e una complessa configurazione di tutti i parametri. Quindi, i vantaggi di una classificazione tramite rete neurale sono un’elevata accuratezza della classificazione ed un’alta robustezza ai rumori e ai missing value, poiché i dati vengono propagati più volte nella rete ed ogni iterazione questi vengono riassorbiti.

Inoltre, questa tecnica di classificazione consente di effettuare la regressione, in quanto è in grado di generare output sia di tipo discreto che continuo. Di contro, gli svantaggi sono bassa efficienza, poiché è necessario un tempo di apprendimento lungo ed una complessa configurazione dato che occorre scegliere in modo accurato parametri, e la non interpretabilità del modello, poiché sono presenti soltanto un insieme di pesi ed un offset per ogni nodo. La tecnica di classificazione basata su rete neurale, data la non interpretabilità del modello, viene utilizzata quando si vuole raggiungere un’elevata accuratezza, ma non è necessario sapere come è stata presa la decisione della classificazione.

* + 1. **Classificazione Bayesiana**

Un classificatore Bayesiano consente la classificazione di collezioni di dati utilizzando dei valori di probabilità ricavati grazie all'applicazione del **teorema di Bayes**. [2] Il teorema di Bayes permette di calcolare la probabilità che una data istanza appartenga ad una determinata classe, indicata come 𝑃(𝐼|𝐶), dove 𝐼 indica l'istanza considerata e 𝐶 è il valore di una determinata classe. Tra tutti i valori di probabilità che vengono calcolati per una determinata istanza da classificare viene preso il valore massimo; questo indica che quell'istanza ha maggiore probabilità di appartenere a quella classe scelta in confronto a tutte gli altri possibili valori delle classi. Dal teorema di Bayes la probabilità 𝑃(𝐼|𝐶) è data dalla formula seguente:

𝑃(𝐶|𝐼) =

𝑃(𝐼|𝐶)𝑃(𝐶)

𝑃(𝐼)

dove 𝑃(𝐼) è un valore costante per tutti i valori di C, quindi è trascurabile, 𝑃(𝐶) è chiamata **probabilità a priori** e 𝑃(𝐶|𝐼) è chiamata **probabilità a posteriori**. Il calcolo di 𝑃(𝐼|𝐶) è effettuato grazie all'assunzione dell'**ipotesi Naїve** con la quale, data un’etichetta di classe, gli attributi che compongono l'istanza sono considerati come statisticamente indipendenti tra loro. Tuttavia, questa ipotesi non è sempre vera ed è la ragione per cui ne influenza l'accuratezza della classificazione. Quindi, è possibile effettuare il calcolo della probabilità considerando la seguente assunzione:

𝑘

𝑃(𝐼|𝐶) = ∏ 𝑃(𝑎𝑖|𝐶)

𝑖=1

Dove *k* è il numero di attributi dell’istanza e 𝑎𝑖 è il valore dell’attributo 𝑖-esimo.

In definitiva, per l'assegnazione dell'etichetta di classe adb un'istanza occorre il calcolo di tutte le probabilità per cui quell'istanza appartenga alle varie classi 𝑃(𝐶𝑖|𝐼) e la successiva selezione del valore dell'etichetta di classe che presenta il valore di probabilità più alto.

* + 1. **Support vector machine**

Una tecnica di classificazione basata su **support vector machine (SVM)** consente di classificare collezioni di dati sia lineari che non lineari. [2] Una SVM rappresenta tutte le istanze della collezione dei dati di training su un piano formato da un numero di assi (dimensioni) pari al numero degli attributi che costituiscono le istanze. Ad esempio se un’istanza è costituita da tre attributi, allora i dati di training saranno rappresentati su un piano tridimensionale. Le tre principali caratteristiche di un classificatore SVM sono: **linee** o **iperpiani**, a seconda se il classificatore presenta, rispettivamente, un grafico bidimensionale o n-dimensionale, **margini** e **vettori di supporto**. Una linea o un iperpiano costituisce un “confine” che permette di classificare le istanze appartenenti a classi diverse dividendole tra loro. Un margine è una distanza tra le due istanze di classi diverse più vicine. Invece, i vettori di supporto corrispondono alle istanze più difficili da classificare per una SVM, poiché sono quelle che si trovano all’interno dei margini di un iperpiano.

* + - 1. **Tipologie di support vector machine**

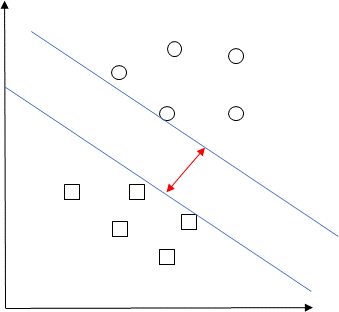
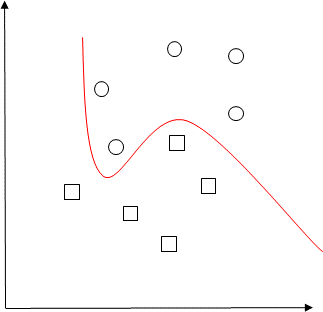
Una SVM permette sia la classificazione collezioni di dati linearmente separabili tra le varie classi sia quelle le cui istanze sono separabili in maniera non lineare. Le istanze di una collezione di dati sono linearmente separabili se, dopo averle rappresentate su un piano, sono separabili da una o più linee rette. Nella Figura 2.10 è rappresentato un esempio di grafico bidimensionale in cui sono raffigurate delle istanze di training linearmente separabili.

Figura 2.10 SVM lineare Figura 2.11 SVM non lineare

I dati di training raffigurati nella Figura 2.10 sono costituiti da dieci istanze appartenenti a due classi differenti. Queste istanze sono separabili da linee rette, quindi risultano separabili in maniera lineare.

Di contro, quando le istanze di una collezione di dati non sono separabili da linee rette, sono separabili in maniera non lineare. Un esempio è mostrato nella Figura 2.11 in cui non è possibile trovare una linea retta che separi le istanze rappresentate appartenenti a due classi differenti.

Le istanze di training di esempio mostrate nella Figura 2.11 sono separabili dalla linea rossa, ma dato che questa non è una retta, le istanze non sono separabili in maniera lineare.

Una SVM si applica in maniera differente a seconda della tipologia delle istanze di una collezione di dati da classificare, linearmente separabili e non.

In presenza di istanze separabili in maniera lineare, occorre trovare tra tutte le rette o tra tutti gli iperpiani che le separano tra le diverse classi quelle che massimizzano il valore del margine. Infatti, viene selezionata la retta o l’iperpiano con valore di margine massimo, poiché consente di minimizzare l’errore di classificazione. Un esempio che rappresenta due iperpiani caratterizzati due diversi valori di margine è mostrato nella Figura 2.12:

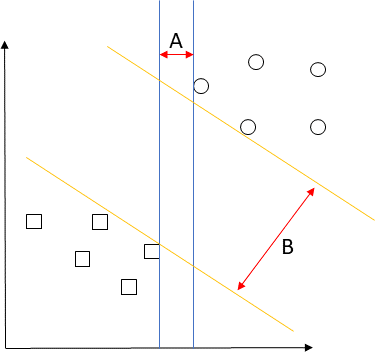


Figura 2.12 Esempio SVM lineare

Come si evince dalla Figura 2.12, i due iperpiani presentano margini di diversa grandezza ed in questo caso verrà selezionato l’iperpiano con margine B > A.

Per quanto riguarda collezioni di dati separabili in maniera non lineare il procedimento di classificazione è più complesso, in quanto occorre operare in due fasi separate ed estendere il comportamento del precedente approccio. Nella prima fase le istanze della collezione di dati vengono mappate su uno spazio dimensionale più grande in modo da renderle separabili in maniera lineare. Infine, nella seconda fase è possibile utilizzare l’approccio precedente di ricerca una linea o un iperpiano che massimizza la grandezza del margine, dato che adesso le istanze sono linearmente separabili.

* + - 1. **Qualità della classificazione tramite SVM**

Un classificatore basato su support vector machine tra i vantaggi presenta la capacità di risolvere problemi di classificazione difficili e non lineari e di garantirne un alto livello di accuratezza di classificazione. Per quanto riguarda la risoluzione di problemi semplici, invece, l’accuratezza è paragonabile a quella di una tecnica di classificazione basata su un albero di decisione. Tra gli svantaggi sono presenti l’elevato tempo di creazione del modello, inferiore a quello impiegato da una rete neurale, e la non interpretabilità del modello.

* + 1. **K-Nearest Neighbours**

La tecnica di classificazione di tipo **k-Nearest Neighbour (k-NN)** non prevede la creazione di un modello di classificazione per etichettare le istanze di test, ma conserva in un grafico le istanze di training e successivamente, utilizzando proprio questo grafico, riesce ad effettuare la classificazione delle istanze di test. [2] Il grafico utilizzato per il salvataggio delle istanze di training è un grafico *n*-dimensionale, in cui *n* corrisponde al numero di attributi che caratterizzano le istanze appartenenti alla collezione dei dati di training. Nel grafico le istanze vengono rappresentate come dei punti. La classificazione, secondo questa tecnica, avviene rappresentando l’istanza di test nel grafico e scegliendo i k punti più vicini a questa. All’istanza di test viene assegnata il valore della classe più presente tra quelle delle k istanze di training più vicine.

* + - 1. **Calcolo della distanza tra i vicini**

Per effettuare in modo corretto il calcolo della distanza occorre, innanzitutto, normalizzare il valore degli attributi, diversamente si potrebbero preferire degli attributi rispetto ad altri. Il calcolo della distanza tra attributi con valori numerici viene effettuato utilizzando il **teorema della distanza Euclidea tra due punti** enunciato dalla formula seguente:

𝑑(𝑥, 𝑦) = 2 ∑(𝑥 − 𝑦 )2

√ 𝑖 𝑖

𝑖

dove 𝑑(𝑥, 𝑦) è il valore della distanza tra due punti 𝑥 e 𝑦.

Invece, per il calcolo di attributi con valori non numerici, ad esempio gli attributi di tipo categorico, si procede in un altro modo: considerate due istanze, se i due valori di un attributo coincidono, allora sarà dato un valore di distanza 0, altrimenti sarà assegnato il valore della distanza massimo preso dall’intervallo di normalizzazione per quell’attributo.

* + - 1. **Selezione del parametro k**

Nella classificazione di tipo k-Nearest Neighbour è di fondamentale importanza la giusta selezione del parametro k, poiché ne influenza l’accuratezza della classificazione. Se si scegliesse un valore di k troppo basso, la classificazione potrebbe essere influenzata da istanze di rumore.

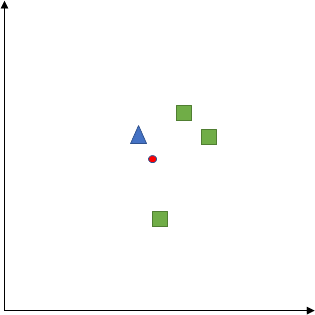


Figura 2.13 Classificatore k-NN

Nella Figura 2.13 è mostrato un grafico bidimensionale di esempio in cui sono rappresentate quattro istanze di training, una blu e tre verdi, e un’istanza di test rossa. Se venisse scelto un valore del parametro k = 1, l’istanza di test verrebbe etichettata con la classe blu erroneamente, perché per k = 3 o k = 4 sarebbe stata selezionata l’etichetta di classe corretta, la verde.

Di contro, un valore di k troppo grande potrebbe portare all’inclusione di punti

appartenenti ad altre classi con la possibilità di provocare un errore di classificazione.

In definitiva, la selezione del parametro k gioca un ruolo chiave nella tecnica di classificazione k-NN.

* + - 1. **Funzionamento del classificatore k-NN**

La tecnica di classificazione k-NN prevede l’iniziale rappresentazione tramite punti di tutte le istanze di training su un grafico il cui numero di assi è pari al numero di attributi delle istanze della collezione dei dati di training. Infatti, per la fase di apprendimento, questa tecnica di classificazione deve soltanto memorizzare in maniera efficiente i dati di training per la successiva lettura. In seguito, viene scelto il valore del parametro k. Quando occorre classificare un’istanza di test, anch’essa viene rappresentata su questo grafico e ne viene calcolata la distanza tra il punto corrispondente all’istanza di test con tutti i punti rappresentanti le istanze di training. Tra tutti questi valori di distanza vengono selezionati i k valori più bassi. In aggiunta al calcolo della distanza, nella selezione delle

k istanze, viene utilizzato un valore di peso inversamente proporzionale al quadrato della distanza, in modo da dare maggiore peso alle istanze di training più vicine al punto che rappresenta l’istanza di test. Dopo aver selezionato le k istanze di training più vicine, viene scelto il valore di classe più presente tra le k istanze selezionate e viene assegnato all’istanza di test.

* + - 1. **Qualità di un classificatore k-NN**

Una classificazione effettuata tramite la tecnica k-NN presenta un buon valore del livello accuratezza e consente la gestione anche di casi particolari di una collezione di dati, poiché conserva tutti i dati di training. Quindi, conservando tutto il training set non fa alcuna astrazione delle istanze presenti ed in questo modo riesce a considerare ogni caso particolare. Come un classificatore bayesiano, anche la tecnica di classificazione k-NN presenta la caratteristica di essere incrementale, poiché in presenza di un nuovo dato etichettato basta aggiungerlo tra i dati di training precedentemente presenti nel grafico. Uno degli svantaggi di questa tecnica di classificazione è l’elevato tempo di classificazione, in quanto occorre calcolare tutte le distanze tra i punti del training set e tutti i nuovi punti che devono essere etichettati. Maggiore è la cardinalità del training set, maggiore sarà questo tempo di classificazione. Quindi, un classificatore di questo tipo va utilizzato per la risoluzione di problemi di classificazione in cui non occorre velocità. Inoltre, un altro svantaggio è la non interpretabilità, in quando questo classificatore non presenta alcun modello, ma soltanto tutte le istanze del training set.