UNIONE EUROPEA
Fondo Sociale Europeo
Investiamo nel tuo futuro

pon SPAO
SISTEMI DI POLITICHE ATTIVE PER L'OCCUPAZIONE

ANPAL
Agenzia Nazionale Politiche Attive del Lavoro

INAPP
ISTITUTO NAZIONALE PER L'ANALISI DELLE POLITICHE PUBBLICHE

Breve disamina degli algoritmi di intelligenza artificiale (aspetti tecnici e metodologici)

Novembre 2022

L'Istituto nazionale per l'analisi delle politiche pubbliche (INAPP) è un ente pubblico di ricerca che si occupa di analisi, monitoraggio e valutazione delle politiche del lavoro, delle politiche dell'istruzione e della formazione, delle politiche sociali e, in generale, di tutte le politiche economiche che hanno effetti sul mercato del lavoro. Nato il 1° dicembre 2016 a seguito della trasformazione dell'Isfol e vigilato dal Ministero del Lavoro e delle politiche sociali, l'Ente ha un ruolo strategico - stabilito dal decreto legislativo 14 settembre 2015, n. 150 - nel nuovo sistema di governance delle politiche sociali e del lavoro del Paese.

L'Inapp fa parte del Sistema statistico nazionale (SISTAN) e collabora con le istituzioni europee. Da gennaio 2018 è Organismo Intermedio del PON Sistemi di Politiche Attive per l'Occupazione (SPAO) per svolgere attività di assistenza metodologica e scientifica per le azioni di sistema del Fondo sociale europeo ed è Agenzia nazionale del programma comunitario Erasmus+ per l'ambito istruzione e formazione professionale. È l'ente nazionale all'interno del consorzio europeo ERIC-ESS che conduce l'indagine European Social Survey.

INAPP
Corso d'Italia, 33
00198 Roma
Tel. + 39 06854471
www.inapp.org

Il presente rapporto è stato redatto dall'Inapp in qualità di Organismo Intermedio del PON SPAO FSE 2014-2020, Azione 10.4.11.1, Ambito di attività 4 (Struttura Lavoro e Professioni, diretta da Paolo Severati).

Autore del testo: Saverio Lovergine

Correzione delle bozze, editing grafico e impaginazione a cura di *Valentina Orienti*

Testo chiuso a ottobre 2022
Pubblicato a novembre 2022

Indice

Introduzione

Negli ultimi anni l'interesse per l'Intelligenza artificiale (IA) è cresciuto in diversi settori economici e ambiti di policy. Il settore dell'IA si sta spostando verso la creazione di sistemi intelligenti in grado di collaborare efficacemente con gli esseri umani, inclusi alcuni modi creativi per lo sviluppo di modalità interattive e scalabili (Amir *et al.* 2016).

Haenlein e Kaplan (2019) definiscono l'IA come la capacità di un sistema di interpretare correttamente i dati esterni, di imparare da tali dati e di utilizzare le conoscenze per raggiungere obiettivi e compiti specifici attraverso un adattamento flessibile, senza l'intervento diretto dell'essere umano.

Fino all'avvento degli algoritmi di Machine learning (ML), gli studi sull'IA si basavano su una struttura descritta come *hardcoded* (ivi 2019), definita da regole e istruzioni codificate.

In seguito, all'interno dell'IA si sono sviluppati sistemi di ML, e come sottoinsieme di quest'ultimo, sistemi di Deep learning (DL), che possono essere descritti come un insieme di algoritmi e di metodi di programmazione, che sfruttano la potenza di elaborazione dei computer attuali per processare in modo efficiente grandi set di dati (Saunderson e Bruchansky 2019).

Il ML è una branca dell'IA che si occupa della progettazione e dello studio di sistemi che apprendono conoscenza attraverso l'analisi di grandi quantità di dati in modo da riconoscere automaticamente modelli complessi, ed essere in grado di prendere decisioni, o, categorizzare dei dati. Quindi, un algoritmo di ML è capace di apprendere dai dati.

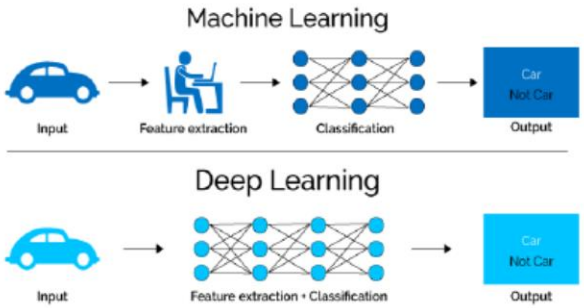
Il DL, sottocategoria del ML, è caratterizzato dalla creazione di un modello di apprendimento automatico a più strati, nel quale i livelli più profondi acquisiscono in input le uscite dei livelli precedenti, trasformandoli e astraendoli sempre di più. Sebbene l'algoritmo di DL esegua anche l'apprendimento basato sui dati, il processo di apprendimento non funziona su un singolo modello matematico, come negli algoritmi di ML standard, ma su calcoli basati su diagrammi di rete espressi come una rete neurale (Lecun *et al.* 2015).

La discriminante tra ML e DL[1] riguarda la modalità di apprendimento di ciascun algoritmo: il primo, (classico, o 'non profondo') dipende in misura maggiore dall'intervento umano per apprendere; il secondo, invece, automatizza gran parte del processo relativo all'estrazione di

[1] Con il 'Machine learning', la macchina è in grado di effettuare ragionamenti di carattere deduttivo, creando generalizzazioni a partire dal caso concreto. Con il 'deep learning' , invece, vengono affrontati problemi maggiormente complessi, in genere attraverso l'inserimento di più strati nascosti nell'ambito delle reti neurali.

caratteristiche, eliminando parte dell'intervento umano e consentendo l'uso di dataset più grandi. Gli esperti determinano il set di caratteristiche, per comprendere le differenze tra i dati di input, richiedendo dati più strutturati per l'apprendimento.

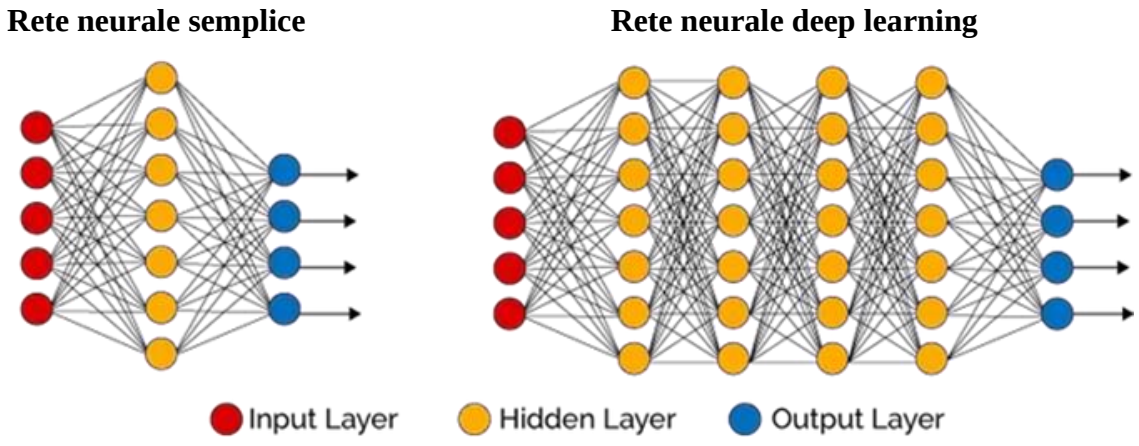
Figura 1. Differenze tra i processi di ML e DL[2]



Fonte: https://bit.ly/3NUNf7y

Una delle tecniche con cui può essere effettuato il ML è costituita dalla creazione di reti neurali, una tecnologia che tende a 'replicare' il funzionamento del cervello umano.
Una rete neurale (o rete neurale artificiale) è composta da livelli di nodi che contengono un livello di input, uno o più livelli nascosti, e un livello di output.
Come si può notare dalla figura 2, l'architettura della rete neurale artificiale prevede che i neuroni siano disposti su diversi livelli, rendendo necessaria la disposizione del numero dei livelli (layer) e del numero di neuroni per ogni livello.

Figura 2. Struttura di una rete neurale artificiale



Fonte: Tosi (2020)

Ciascun nodo si connette ad un altro associando un peso ed una soglia. Se l'output di qualsiasi singolo nodo è al di sopra della soglia specificata, tale nodo è attivato ed invia i dati al successivo livello di rete. In caso contrario non si passa alcun dato al livello successivo di rete. Quindi, una rete neurale è un insieme interconnesso di funzioni elementari in cui gli output sono rappresentati dagli input delle successive funzioni.

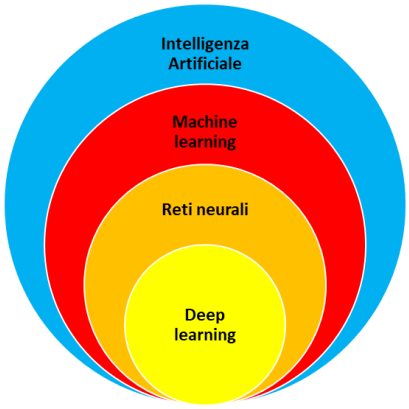
[2] Per la differenza tra Machine learning e Deep Learning si veda, tra le altre, la definizione al link https://bit.ly/3NUNf7y.

Una rete neurale che consta solo di due o tre livelli è una rete neurale di base; mentre se ha più di tre livelli è considerata un algoritmo di DL, o una rete neurale profonda.

Una rete neurale si addestra ponendo in input una serie di esempi che costituiscono il dataset di training. Il raffronto tra le risposte fornite dai campioni di training e quelle attese determina l'errore della rete. Le risposte fornite dalla rete per i campioni di addestramento vengono confrontate con le risposte. La procedura è ripetuta fino a quando le uscite della rete non forniscono un errore al di sotto di una soglia prestabilita. In conclusione, le reti neurali si affidano ai dati di addestramento per imparare a migliorare la loro performance.

Detto ciò, ogni algoritmo di DL è un algoritmo di ML, ma non tutti gli algoritmi di ML sono algoritmi di DL, come si evince dalla figura 3, che rappresenta la relazione tra IA, ML, Rete neurale e DL.

Figura 3. Relazione tra IA, ML, Rete neurale e DL



Fonte: Autore (2022)

1. Il machine learning e gli algoritmi di apprendimento

La definizione di ML più accreditata dalla comunità scientifica è quella fornita nel 1997 da Tom Mitchell, direttore del dipartimento Machine Learning della Carnegie Mellon University: "Un programma apprende da una certa esperienza E se nel rispetto di una classe di compiti T, con una misura della prestazione P, la prestazione P misurata nello svolgere il compito T è migliorata dall'esperienza E".

Quindi, l'importanza degli algoritmi di ML è dovuta dalla loro capacità di fornire alla macchina una conoscenza di base e una conoscenza più elaborata, create grazie a 'esperienze' (azioni, corrette o errate che siano), che permettono di svolgere determinati compiti, migliorando le proprie capacità, risposte e funzioni.

La forza dell'apprendimento automatico rispetto ad altre forme di analisi è nella sua capacità di scoprire intuizioni nascoste e prevedere i risultati di input futuri non visibili (attività di generalizzazione[3]). A differenza degli algoritmi iterativi, in cui le operazioni sono esplicitamente dichiarate, gli algoritmi di ML prendono in prestito concetti dalla teoria della probabilità per selezionare, valutare e migliorare i modelli statistici.

Dal punto di vista teorico, rispetto ad un approccio tradizionale, quello di ML si identifica in un sistema[4] costituito di due fasi: la prima, di allenamento, in cui la macchina apprende a partire da esempi; la seconda, di funzionamento (o generalizzazione), successiva alla prima, che permette di generalizzare e gestire nuovi dati nello stesso dominio applicativo.

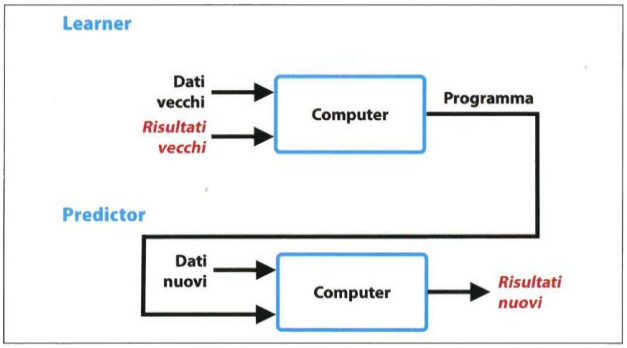
In altre parole, si chiede alla macchina di comprendere il legame che unisce dati e risultati. Come evidenziato nella figura 4, in una prima fase (learner), la macchina esegue l'attività di apprendimento usando un algoritmo di ML (algoritmo di learning); nella seconda (predictor), sfrutta quanto ha imparato nella fase di learner su nuovi dati di input[5]. Infatti, in tale approccio cambia il flusso di processo: l'algoritmo di ML 'impara' per arrivare ad un risultato e restituisce tale processo in uscita (De Mauro 2020, 66).

[3] Per 'generalizzazione' si intende l'abilità di un algoritmo di performare ottimamente su input mai osservati.

[4] Gli elementi principali che definiscono un sistema che apprende sono:
- algoritmo (insieme di regole, basate su metodi statistici, per estrarre dai dati degli schemi ricorrenti);
- modello (rappresentazione di un ambito reale tramite un algoritmo costruito con opportuni criteri);
- dataset di learner (insieme dei dati che permettono, grazie all'algoritmo, la costruzione del modello);
- dataset di test (insieme dei dati che, a seguito della fase di *learner*, sono utilizzati per valutare la qualità del modello).

[5] Ad esempio, per apprendere a identificare l'immagine di un cavallo il programma deve ricevere moltissime immagini di tale animale (dati di *learning*). Per migliorare la precisione dell'algoritmo sono necessari ulteriori dati di *feedback*.

Figura 4. Predizione tramite ML



Fonte: De Mauro (2020)

Tale approccio di ML, rispetto ad un approccio tradizionale, ha i seguenti vantaggi:

1. riutilizza il programma generato dal computer in modo che possa 'girare' su nuovi dataset, al fine di ottenere nuovi risultati (una macchina che 'impara' è capace di sfruttare tale processo e applicarlo su casi futuri);
2. identifica delle connessioni (pattern) estremamente complesse tra dati e risultati (grazie alla possibilità di processare più dati, inseriti più volte fino a quando non 'impara').

Quindi, l'efficacia di tale approccio deriva dalla possibilità che l'algoritmo di ML evidenzi le suddette connessioni tra dati e risultati, con la possibilità che l'apprendimento automatico possa proporre pattern inaspettati[6].

Premesso ciò, i data scientist fanno uso di svariati algoritmi di ML per individuare pattern all'interno dei loro dati che forniscano spunti interessanti per supportare decisioni strategiche di vario tipo.

1.1 Tecniche di apprendimento

Nel ML i sistemi fanno uso di diverse tecniche per apprendere che differiscono tra loro sia per le modalità con cui agiscono per raggiungere il loro obiettivo, sia per la tipologia dei dati forniti in input. Gli algoritmi di ML sono tipicamente classificati, in base alla natura dell'apprendimento, in quattro principali categorie:

a. apprendimento supervisionato
b. apprendimento non supervisionato
c. apprendimento semi-supervisionato
d. apprendimento per rinforzo.

[6] Una delle questioni più pressanti che gli esperti di IA stanno affrontando è il cd. problema del *black box*, ovvero l'impossibilità di capire perché una IA sia giunta a una determinata conclusione. Ciò vale in particolare per gli algoritmi di DL, e per gli apprendimenti diversi da quello supervisionato.

A. Apprendimento supervisionato (supervised learning)

L'apprendimento supervisionato è la tecnica più utilizzata (e studiata) tra quelle su menzionate. Il nome di questa classe di algoritmi deriva dal ruolo 'guida', o meglio da 'supervisore', svolto dal data scientist nel processo di apprendimento.

In tale apprendimento il sistema è istruito conoscendo già che (corretti) valori di output corrispondono a ciascun input. Nello specifico, i dati sono classificati ed etichettati, ovvero sono forniti sia di un dataset in input (*training set),* corredati da tutti gli attributi utili al sistema, sia di un valore di output (*target value* o *label*), che può essere discreto o continuo. Questo valore di output determina la tipologia di problema da risolvere: se il valore a disposizione è reale si ha un problema di 'regressione', se il valore è discreto si ha un problema di 'classificazione'.

L'apprendimento supervisionato è utilizzato principalmente nelle classificazioni, ossia nell'applicazione di una certa etichetta abbinata ad un certo dato di input.

L'obiettivo finale è quello di produrre un'ipotesi induttiva, ovvero generare un modello di risoluzione per problemi generali basato su proprietà conosciute apprese dai dati di input.

Il meccanismo di apprendimento supervisionato impiega l'algoritmo per l'allenamento delle reti neurali artificiali usato in combinazione con un metodo di ottimizzazione, al quale si affianca l'esperienza dell'operatore che istruisce la rete. Il motivo risiede nella difficoltà di definire un rapporto adeguato fra le dimensioni del *learning set*, della rete e l'abilità a generalizzare.

D'altronde, un numero eccessivo di parametri in ingresso e l'elevata capacità di elaborazione, rendono difficile alla rete neurale di imparare a generalizzare; mentre un *learning set* con variabili di input scarsi, non permette alla rete di avere sufficienti parametri per imparare a generalizzare.

Tale tecnica è utile soprattutto per risolvere problemi di valori mancanti nel dataset. Con questo approccio è possibile allenare il sistema con dataset completi. Una volta allenato il sistema su di un numero esauriente di casi, è possibile testarlo su dataset con dati mancanti, per verificare come il sistema li completa. Pertanto, i dataset completi sono utilizzati per l'allenamento della rete, mentre quelli incompleti sono utilizzati per verificare se la rete li completa.

L'apprendimento supervisionato è impiegato per i problemi di classificazione (es. profilazione dei clienti) o per problemi di regressione (es. previsioni meteo).

Gli altri approcci, riportati di seguito, diversamente dall'apprendimento supervisionato, sono invece più orientati alla ricerca o al miglioramento delle relazioni nel set di dati.

B. Apprendimento non supervisionato (unsupervised learning)

L'apprendimento non supervisionato, o approccio duale, possiede caratteristiche e algoritmi estremamente differenti rispetto a quelli dell'apprendimento supervisionato. Tale approccio è più vicino a quella che molti autori considerano la 'vera' IA, ovvero l'idea che un computer sia in grado di imparare a identificare processi e *pattern* di diversi livelli di complessità anche senza alcun supporto da parte di esseri umani, come è necessario, invece, nell'apprendimento

supervisionato.

Infatti, la macchina deve riconoscere alcuni aspetti della struttura dei dati in input senza alcuna indicazione riguardante il risultato in output; pertanto, i metodi di classificazione e regressione precedentemente analizzati sono difficilmente applicabili. In questo caso non ci sono risultati passati da fornire alla macchina come riferimento, ma essa, attraverso determinate tecniche, cerca le relazioni tra i vari dati, ed eventualmente li suddivide in categorie, senza nessuna supervisione da parte dell'umano. Infatti, nessuna etichetta è data all'algoritmo come indicazione su cosa cercare, e l'unico metodo a disposizione per definire un modello generale è confrontare i dati in input fra loro, cercando analogie e differenze, e riorganizzandoli sulla base di ragionamenti e previsioni sui dati successivi.

Lo scopo dell'apprendimento non supervisionato è quello di risalire, attraverso l'utilizzo di algoritmi, a *pattern* sconosciuti in una fase precedente all'interno del dataset senza ricevere alcun 'aiuto' da dati pregressi di riferimento.

Tuttavia, questi *pattern* sono solo delle approssimazioni rispetto ai possibili risultati ottenibili attraverso l'apprendimento supervisionato. Una volta che il sistema trova le relazioni tra dati, li suddivide in categorie; in seguito spetta all'utilizzatore usarli e/o etichettarli appropriatamente. Questo approccio può essere un obiettivo di per sé (ad es. per scoprire modelli nascosti nei dati) o un mezzo verso un fine (in funzione dell'apprendimento); ed è usato, si ribadisce, quando esiste la forte convinzione, se non la certezza, che ci siano relazioni interne tra i dati.

C. Apprendimento semi supervisionato (semi-supervised learning)

Nell'apprendimento semi-supervisionato, il dataset è parzialmente etichettato[7], ed è prevalentemente utilizzato per ottimizzare il processo (regola) di classificazione di tale set di dati (*labelling*). Quest'ultimo caso si presenta molto frequentemente, soprattutto quando l'etichettatura dei dati risulta molto costosa, oppure quando si lavora su un flusso costante di dati.

I modelli semi-supervisionati[8] mirano a definire alcune ipotesi (presupposti) sui dati al fine di giustificare l'utilizzo di una serie di dati etichettati per trarre conclusioni sui punti dati non etichettati (spesso maggiori rispetto ai primi).

Alcuni metodi semi-supervisionati comuni sono le macchine vettoriali di supporto trasversali e i metodi basati su grafici (ad es. la propagazione delle etichette).

[7] Tale apprendimento è utilizzato nel caso in cui solo una minoranza dei dati presenta etichette e un valore di output.

[8] Tali modelli possono essere raggruppati in tre categorie:

a. presupposto di continuità (si presume che i punti dati 'vicini' tra loro abbiano maggiori probabilità di avere un'etichetta comune);
b. ipotesi del *cluster* (si presume che i dati costruiscono *cluster* discreti e che i punti nello stesso *cluster* abbiano maggiori probabilità di condividere un'etichetta);
c. presupposto molteplice (si presume che i dati si trovino approssimativamente in uno spazio di dimensioni inferiori rispetto allo spazio di input).

D. Apprendimento rinforzato (reinforcement learning)

Nell'apprendimento rinforzato l'obiettivo è permettere alla macchina di interagire con l'ambiente circostante, dal quale riceve stimoli a seconda della correttezza della scelta compiuta, e/o di imparare anche dagli errori che derivano da questa continua interazione. In questo processo l'agente che apprende deve valutare l'azione più opportuna da compiere tra un insieme di esse, valutando la miglior ricompensa (*reward*) tra quelle effettuate nell'ambiente specifico (*environment*). Quindi, lo scopo è apprendere l'azione ottimale in ciascuno stato, in modo da massimizzare la somma dei *reward* ottenuti nel lungo periodo.

Un punto di forza dell'apprendimento rinforzato infatti è la creazione di un agente che agisce partendo anche senza una conoscenza iniziale del 'mondo reale', ovvero anche se non si forniscono esempi o simulazioni degli stati dell'ambiente, in quanto essi saranno appresi in autonomia dal sistema. In altre parole, l'apprendimento rinforzato consente a un agente di apprendere superando i limiti della conoscenza umana, scoprendo scenari di comportamento dell'ambiente imprevedibili a priori. In virtù di ciò, tale apprendimento consente il vantaggio di risparmiare tempo in pre-addestramento di un sistema, di superare il livello fornito da un simulatore del mondo reale che non sarà mai esaustivo, in quanto l'apprendimento è continuo e non legato ai dati, ma migliora con il funzionamento del sistema nel mondo reale.

Questa caratteristica rende l'apprendimento rinforzato molto interessante e adatto per gestire sistemi complessi e, ove è necessario, per prendere delle decisioni per far evolvere il sistema con una serie di scelte che spesso sono sconosciute o difficili da definire.

Ampio utilizzo di tale apprendimento sono la finanza (per le strategie di trading), il controllo della qualità, la gestione dei servizi di consegna prodotti ai clienti, l'ottimizzazione delle catene di fornitura in aziende logistiche, l'assistenza al parcheggio di auto ecc.

Google, ad esempio, utilizza l'apprendimento rinforzato per controllare gli impianti di climatizzazione nei data center. L'apprendimento per rinforzo è utilizzato anche nel controllo di sistemi complessi (sistemi di trasporto intelligenti, *smart grid* [9] ecc.).

1.2 Principali metodi di elaborazione dei dati

In questo paragrafo si riporta una breve panoramica, senza la pretesa di esaustività, delle tecniche cardine di ML e delle principali problematiche che si presentano quando ci si approccia a contesti che richiedono il loro utilizzo, in modo tale da fornire una visione generale sia della diversità degli approcci, sia dei vari contesti di applicazione.

[9] Reti elettriche intelligenti.

Apprendimento supervisionato

Come già riportato, nel caso dell'apprendimento supervisionato si possono avere due tipi di approcci:

- Classificazione: nel caso di classificatori (o descrittori), gli input sono suddivisi in due o più classi, e l'algoritmo deve produrre un modello che assegni i nuovi elementi testati ad una o a più di queste classi (*multi-label classification*).
- Regressione: nel caso della regressione, gli output assegnati sono risultati numerici, invece che descrittori.

Detto ciò, con lo scopo di fornire una visione generale della diversità e degli svariati contesti di utilizzo, si propone un elenco dei principali algoritmi usati come classificatori, anche se in molti casi alcuni di essi sono validi anche per la regressione.

Il modello più semplice di classificazione di oggetti finiti è il *Decision Tree*. La sua funzione è generare un albero decisionale che si sviluppa attraverso due passaggi: la fase di *building* e la fase di *pruning*. La prima, accresce di dimensioni l'albero istanziando archi e nodi. Questi ultimi definiscono le regole di *splitting* che identificano le classi omogenee in termini di output. La fase di *pruning*, invece, si realizza attraverso l'utilizzo di tecniche specifiche che eliminano i rami che non apportano un valore sufficientemente informativo all'albero. Esistono numerosi algoritmi basati su alberi decisionali ognuno dei quali adotta differenti tecniche in fase di *building* e *pruning* (es. ID3, CART, CHAID, MARS).

K-nearest neighbors (KNN) è un algoritmo di apprendimento automatico supervisionato utilizzato per risolvere problemi sia di classificazione che di regressione. Il classificatore KNN è anche un algoritmo di apprendimento non parametrico, basato su istanze[10]. Pur essendo il più semplice e il più veloce tra gli algoritmi di ML, non è adatto a dataset di grandi dimensioni, in quanto richiede molte risorse computazionali e di memoria. L'algoritmo lavora su k esempi per volta, e cerca i k elementi più somiglianti[11] – elementi vicini, prossimi o meno distanti – al campione da classificare; l'output che si assegna è l'etichetta che compare più volte tra i k esempi. Quindi, l'algoritmo KNN essenzialmente valuta la somiglianza degli elementi più vicini e classifica tali elementi. Lo scopo di questo algoritmo, quindi, è quello di predire la classe di una nuova istanza, basandosi sulla posizione dei dati del dataset. Pertanto, per identificare i *data points* più vicini alla nuova istanza l'algoritmo fissa un parametro k che determina il numero di punti più vicini alla nuova istanza da prendere in considerazione.

Anche *Support Vector Machine* è un algoritmo di apprendimento supervisionato, utile sia per la classificazione di *pattern* che per la regressione. Tale algoritmo è molto efficiente su dataset a molte dimensioni. Inoltre, esso ottimizza l'uso di memoria perché di tutto il dataset apprende i cosiddetti vettori di supporto, un sottoinsieme di esempi di addestramento che identifica

[10] L'apprendimento basato su istanze non avviene tramite l'addestramento esplicito di un modello, ma attraverso la scelta di memorizzare le istanze di addestramento che rappresentano la 'conoscenza' utile alla predizione.
[11] La somiglianza è definita in base a una metrica di distanza tra due punti di dati.

l'iperpiano separatore migliore, tra tutti i possibili, che separano i dati in classi. Se l'insieme dei dati di addestramento ha N dimensioni viene prodotto un iperpiano di N-1 dimensioni che separa le classi. Nel caso in cui non esistesse un iperpiano in grado di dividere le classi di dati, l'algoritmo utilizzerà una mappatura non lineare per trasformare i dati in una dimensione superiore e cercare nuovamente un modo di dividere linearmente le classi di dati. L'algoritmo risulta molto efficace per modelli non lineari tramite l'uso dei *Kernel*[12]. Tale tecnica è molto utilizzata nelle applicazioni di riconoscimento vocale e di immagini. Esiste una versione di tale algoritmo per effettuare la regressione, chiamata *Support Vector Regression*, in cui l'idea principale è sempre quella di minimizzare l'errore.

Random Forest, ugualmente, è uno strumento che ha apportato notevoli migliorie nell'accuratezza dell'apprendimento per la classificazione e la regressione, e rientra nella sfera dell'*ensemble learning*[13]. Basato su alberi decisionali, l'algoritmo costruisce più alberi su sottoinsiemi del dataset di addestramento selezionati casualmente, aggregando le previsioni di ciascun albero per scegliere in seguito la previsione migliore. Tale algoritmo permette anche di valutare l'importanza delle caratteristiche, ovvero tra i dati di input è possibile fare un elenco in ordine di importanza per la determinazione dell'output. Anche per la regressione (in questo caso si parla di *Regression forest*) si costruiscono una moltitudine di alberi decisionali e come output si ottiene la previsione di ogni albero. L'algoritmo funziona bene sia per la classificazione binaria che per quella multi-classe. Inoltre, esso non richiede operazioni di normalizzazione del dataset, e la presenza di *outliers* non impatta sul risultato. La comprensione dei risultati è molto semplice e intuitiva, in quanto il grafico è simile a un *flow chart* le cui foglie contengono i dati di una sola classe. Nel processare i dati, l'algoritmo avanza in modo che ad ogni step essi sono separati in base alla caratteristica che produce il miglior risultato (o migliore separazione). L'algoritmo procede ricorsivamente generando una sequenza di chiamate fino alla classificazione di tutti i dati (condizione di terminazione). Infatti, se tale processo dell'albero non è limitato è facile incorrere nell'*overfitting*[14], ovvero ci si imbatte in un modello troppo preciso sui dati in addestramento, ma poco efficace su dati 'nuovi'.

Naive Bayes è un algoritmo veloce ed efficiente soprattutto su dataset piccoli. Basato sul teorema

[12] Trasformazioni delle caratteristiche di input in spazi con un maggior numero di dimensioni in modo da separare le classi sempre con un iperpiano, ovviamente di dimensioni maggiori

[13] Gli algoritmi *ensemble* sono costituiti da più classificatori di base combinati tra loro in modo tale che una combinazione di classificatori fornisca performance migliori rispetto al singolo classificatore. Rispetto alla modalità di aggregazione, i metodi di tipo *ensemble* sono divisi in macrocategorie, tra cui *bagging*, *voting* e *boosting*. Il *bagging* consiste nel campionare N volte il dataset iniziale e addestrare N volte lo stesso algoritmo, in modo da aggregare i risultati facendo la media degli output, nel caso di regressione, e ricavando la categoria con maggior frequenza, nel caso di classificazione. Si tratta di un metodo che aumenta i dati di addestramento per diminuire la varianza nelle previsioni. Il *Voting*, versione 'semplificata' del Bagging, permette di unire i risultati di diverse categorie di classificatori (il Bagging duplica invece sempre la stessa). Il *boosting* è un processo sequenziale in cui ad ogni step si migliora il modello precedente correggendone gli errori. Esempi di *ensemble* di questo tipo sono AdaBoost, Gradient Boosting, XGBoost e CatBoost.

[14] *L'overfitting* rappresenta il rischio di sovradattamento durante il processo di apprendimento induttivo. Ciò accade quando l'algoritmo individua delle false regolarità, ovvero rumori che sporcano i dati fuorviando l'apprendimento induttivo della macchina.

di Bayes[15], l'algoritmo lavora sulle probabilità condizionate delle caratteristiche in input (soprattutto quando la dimensione degli input è elevata), ovvero l'algoritmo fa delle assunzioni sulla distribuzione dei dati che può essere Gaussiana, Bernoulli o Multinomiale[16]. *Bernoulli Naive Bayes* è utile quando il dataset è composto da dati in forma binaria (del tipo vero o falso, spam o non spam, sì o no, 0 o 1 ecc.); mentre *Naive Bayes Multinomiale* è molto usato per la classificazione multi-classe e offre buone performance su grandi dataset. Utilizzabile su dati discreti o continui, tale algoritmo è impiegato generalmente per la classificazione di documenti, anche se può essere applicato su modelli di *sentiment analysis* e di classificazione di profili di persone sulla base dei propri *post* sui *social network*.

Continuando nella presentazione degli algoritmi nell'ambito dell'apprendimento supervisionato, nel caso in cui si disponga di una correlazione molto buona tra la variabile da predire e un'altra variabile nel dataset, la regressione potrebbe rappresentare un buon modello. Quindi, con questa tecnica si cerca una relazione lineare tra una o più variabili indipendenti di input e una variabile di output; nel caso di una variabile indipendente e un output, il risultato è una retta nel piano che interpola i punti costituiti dal dataset di addestramento.

Nella regressione, dovendo prevedere un valore continuo, si decide un margine di tolleranza dell'errore. A differenza dei modelli lineari ove si approssimano i valori con una retta che risulta molto meno precisa, nel caso di modelli non lineari si può ricorrere alla regressione polinomiale per minimizzare l'errore, ottenendo quindi una funzione curva che approssima meglio i dati.

Il modello più noto di tale apprendimento è la regressione lineare, ovvero una derivazione lineare della relazione tra un valore di risposta e una o più variabili. Ciò consente, in casi semplici, di estrapolare un modello di comportamento della funzione che può essere esteso nel dominio delle variabili utilizzate, eventualmente anticipando il comportamento di un sistema sulla base del comportamento precedentemente registrato[17].

La 'regressione logistica' è uno dei classificatori più efficienti, soprattutto con un output binario. In tale caso, l'output è un valore compreso tra 0 e 1 che rappresenta la probabilità stimata di un'etichetta; inoltre, attraverso tale modello si ottiene la classe di appartenenza del dato e la stima della probabilità di appartenenza a tale classe.

'Discesa del gradiente' è un algoritmo lineare che cerca il minimo della funzione di costo calcolando il gradiente (che in una dimensione è la derivata prima della funzione). Sebbene possa essere utilizzato come algoritmo a sé stante, è impiegato con maggior successo come tecnica di ottimizzazione in altri algoritmi supervisionati e nelle reti neurali.

Le 'reti neurali' possono essere utilizzate anche per la regressione. Infatti, ponendo un unico

[15] Il teorema di Bayes consente di calcolare e aggiornare gli indici di probabilità a seguito dell'acquisizione di nuovi dati dai sistemi analizzati. Tale teorema definisce la probabilità condizionata (o a posteriori) di un evento rispetto ad un altro.

[16] Tali metodi possono ottenere prestazioni di previsione elevate purché l'assunzione di indipendenza tra le variabili misurate sia corretta.

[17] Alla regressione lineare si aggiungono dei metodi di regolarizzazione che introducono delle penalizzazioni nel calcolo dell'errore (metodi Ridge e LASSO).

neurone nel livello di output è possibile allenare una rete neurale a predire valori continui. In questo caso, per misurare l'errore della rete si utilizza lo scarto quadratico medio, con l'obiettivo di minimizzarlo in modo da ridurre al minimo l'errore.

Apprendimento non supervisionato

I metodi di risoluzione dei problemi dell'apprendimento non supervisionato sono suddivisi in tre grandi gruppi: *clustering*, *pattern recognition* e *associazione*.

La finalità del *clustering* è di individuare delle strutture all'interno di un set di dati non classificato. Per fare ciò, gli algoritmi di *clustering*, utilizzando diverse tecniche, raggruppano in alcuni *cluster* i dati con caratteristiche simili. I risultati ottenuti, spesso, sovrastimano le somiglianze tra i gruppi e non gestiscono i *data points* individualmente; pertanto, tali risultati devono essere ritenuti approssimativi.

L'algoritmo più diffuso, performante, e semplice da utilizzare è il *K-means clustering,* che consente ai dati di appartenere a più di un singolo *cluster* alla volta[18]. Esso si basa sui cd. 'centroidi'[19], ossia dei punti appartenenti allo spazio dei *feature* che media le distanze tra tutti i dati appartenenti al *cluster* ad esso associato. Pur rappresentando il baricentro del *cluster*, in generale, in virtù delle sue caratteristiche, esso non è uno dei punti del dataset. Quando si configura un modello di *clustering*[20] usando il metodo *K-means*, è necessario specificare un numero di destinazione k che indica il numero di centroidi desiderati nel modello. La difficoltà maggiore è rappresentata, appunto, dalla scelta a priori del numero di *cluster*, motivo per il quale si devono realizzare diversi tentativi prima di stabilire il numero corretto. Infatti, la qualità dell'algoritmo e della scelta del numero di *cluster* è valutata a posteriori, principalmente in due modalità: la 'valutazione del gomito' (il punto in cui cambia la curva di un parametro legato all'errore quadratico); la valutazione del cd. 'coefficiente di silhouette' (legata alla coesione calcolata come distanza tra i dati dello stesso *cluster* e la separazione come distanza media tra dati di un *cluster* e quelli del suo vicino più prossimo). Il vantaggio principale dell'uso di tale tecnica è che i dati non sono forzatamente inseriti all'interno di un *cluster*; pertanto, talvolta questo algoritmo risulta più efficace nel fornire una convergenza più robusta verso la soluzione finale. *Gaussian Mixture model* (una generalizzazione dell'algoritmo *K-Means*) fornisce una maggiore flessibilità nella dimensione e nella forma dei *cluster*. Questi raggruppamenti sono utili per esplorare i dati, identificare le anomalie nei dati e, infine, per eseguire delle stime.

Gli algoritmi appartenenti alle tecniche di *Pattern Recognition* analizzano i dataset e ricercano eventuali regolarità nei dati che li compongono. Le fasi di sviluppo dell'algoritmo sono due: una, 'esplorativa', ove gli algoritmi ricercano le regolarità; l'altra, 'descrittiva', ove gli algoritmi

[18] Un *cluster* è un insieme di osservazioni che condividono caratteristiche simili.
[19] Il 'centroide' è un punto rappresentativo di ogni *cluster*.
[20] Per questi motivi, il *clustering* viene spesso usato nelle fasi iniziali delle attività di ML, per esplorare i dati e individuare correlazioni impreviste.

categorizzano i *pattern* individuati nella fase precedente.

'Apriori' e 'ECLAT' sono gli algoritmi principali di tale gruppo. Essi hanno l'obiettivo di generare degli *itemset* frequenti di oggetti, ovvero di individuare degli insiemi di oggetti appartenenti ad un dataset di uso frequente. *Join* e *prune* sono le fasi che servono per determinare quale insieme di oggetti è il più frequente. Nella prima fase (k) i dati appartenenti al dataset sono raggruppati all'interno di vari *itemset*; nella seconda, l'utilizzo di un valore di soglia minima selezionerà gli *itemset* frequenti. Una volta terminata la selezione si avvierà nuovamente la fase *join* inizializzando degli *itemset* composti da k+1 elementi. In questo modo si partirà con degli *itemset* contenenti un singolo oggetto del dataset e, a seguito di varie iterazioni, alla fine del processo si otterrà un singolo *itemset* contenente i dati più frequenti. Nonostante i due algoritmi svolgano la stessa funzione, le tecniche applicate per arrivare al risultato finale sono differenti:

- 'Apriori' nella fase *join* effettua una scansione dell'intero dataset per generare gli *itemset*;
- 'ECLAT' genera *itemset* utilizzando i risultati ottenuti dalla fase di *prune* precedente (se presente), ciò consente ad 'ECLAT' di essere generalmente più veloce e di usare meno memoria.

Le tecniche di 'Associazione' sono spesso impiegate in presenza di grandi dataset dai quali si deducono regole generali che descrivono le relazioni. I metodi generalmente utilizzati per misurare l'associazione tra due elementi sono:

- *Support,* che calcola la frequenza della presenza di una determinata caratteristica all'interno di un dato, tra tutti i dati apparenti al dataset;
- *Confidence,* che analizza la probabilità condizionata (il grado di probabilità che una data caratteristica sia presente se all'interno del dato analizzato si ha un'altra caratteristica specifica);
- *Lift,* che calcola la frequenza di una probabilità condizionata.

Principal component analysis (PCA) è uno degli algoritmi di associazione che consente di sintetizzare i dataset utilizzando un numero ridotto di dimensioni. Per raggiungere tale scopo si effettua una trasformazione ortogonale delle coordinate dello spazio originale, al fine di formare un nuovo insieme di variabili linearmente non correlate, chiamate componenti principali. A tali componenti corrispondono degli autovettori della matrice di covarianza dei dati. In altre parole, si può pensare a PCA come un cambio di base. La trasformazione produce un insieme di vettori di base, un sottoinsieme dei quali è in grado di coprire un sottospazio lineare all'interno dello spazio originale che rappresenta quasi totalmente il raggruppamento dei dati. Tuttavia, non tutti i dati sono facilmente riassunti in un sottospazio lineare. Nei problemi di classificazione, ad esempio, ci sono molte fonti di dati che non sono separabili linearmente. In questo caso è possibile invocare il 'trucco del kernel', che permette di applicare la PCA a set di dati non lineari. Nella finanza quantitativa, PCA è spesso usata per l'analisi di fattori, in quanto permette l'osservazione di un gran numero di titoli correlati attraverso il tentativo di ridurne la dimensione osservando un insieme più piccolo di *variabili latenti* non osservate e non correlate.

Apprendimento semi-supervisionato

L'apprendimento semi-supervisionato, come detto, è un ibrido dei due precedenti apprendimenti. Infatti, il *learning set* è parzialmente etichettato o riceve informazioni correttive dall'esterno. Detto algoritmo è anche chiamato apprendimento auto-supervisionato in quanto le reti si autogestiscono.

Principale algoritmo semi-supervisionato sono le reti *generative avversarie* (o 'antagoniste generative') che hanno bisogno di moltissimi dati in entrata. L'algoritmo si compone di due reti neurali avversarie che interagiscono tra loro. La prima, 'generatore', opera in modo che i dati siano quanto più simili a quelli etichettati; la seconda, 'discriminatore', deve apprendere il modo di come distinguere i dati etichettati di input da quelli concepiti dal generatore. Tale processo finisce quando il discriminatore non è più in grado di distinguere i dati generati da quelli reali, ovvero quando il generatore ha imparato a stimare correttamente i dati.

A ciò si aggiungono anche gli algoritmi di *clustering semi-supervisionati* che, a differenza di altri che dividono in gruppi i dati a seconda delle loro somiglianze, sfruttano, se disponibili, delle informazioni aggiuntive su una piccola porzione di dati per ottenere risultati migliori. Tali informazioni possono essere espresse sotto forma di etichetta di classe, o sotto forma di vincoli a coppie. Nello specifico si avranno due liste di vincoli: quella che si basa sui dati che devono finire nello stesso *cluster*, chiamati *must-link;* e quella che si basa sui dati che non devono finire nello stesso *cluster*, chiamati *cannot-link*. Quando si usa un algoritmo di *clustering standard*, i *cluster* generati si basano solo sulla distribuzione dei dati nello spazio. Grazie all'uso degli algoritmi semi-supervisionati si può affinare il raggruppamento mediante regole semantiche, anziché topologiche.

'MPCK-Means', variante del K-Means, raggruppa i dati in modo che la varianza totale all'interno di ogni *cluster* sia resa minima. L'informazione ausiliaria, che rende tale l'algoritmo semi-supervisionato, è espressa mediante vincoli a coppie. Per supportare i vincoli a coppie è necessario modificare la funzione obiettivo del K-Means, associando una penalità ad ogni vincolo non rispettato. MPCK-Means è in grado anche di imparare una metrica per ogni *cluster*, in modo che ognuno possa avere una forma diversa. Inoltre, visto che tale algoritmo parametrizza una metrica diversa per ogni *cluster*, esso è anche in grado di fare il *clustering* di una collezione di dati nella quale i diversi *cluster* hanno una forma diversa tra di loro. Questo rappresenta un vantaggio rispetto alla versione originale di K-Means. Infine, l'algoritmo al crescere del numero di vincoli aumenta la qualità del *clustering*.

Apprendimento rinforzato

Tra gli algoritmi utilizzati in tale apprendimento c'è il *Markov Decision Processes,* utile a definire l'interazione tra l'agente che apprende e il suo ambiente, in termini di stati, azioni e ricompense. *Markov Decision Process* permette la formalizzazione classica di decisioni sequenziali (ad es.

causa ed effetto), in cui le azioni non influenzano solo la singola ricompensa ma anche gli stati successivi e le future ricompense.

La 'Programmazione Dinamica' è un algoritmo che richiede alti sforzi computazionali, e calcola le policy ottimali partendo da un modello dell'ambiente descritto da un *Markov Decision Process*. Con metodi iterativi sono valutate le funzioni di policy e di ricompensa partendo da ogni stato del sistema.

'Monte Carlo', invece, è un algoritmo che apprende da un'esperienza che non richiede una conoscenza pregressa della dinamica dell'ambiente, ma si basa sul campionare e media la risposta di ogni coppia azione-stato. Un vantaggio di questo metodo è offrire risultati accurati nella valutazione di un sottoinsieme di stati selezionati che possono essere utilizzati con ambienti simulati.

I metodi 'Temporal Difference' combinano i due metodi precedenti (Monte Carlo e Programmazione Dinamica). Essi, infatti, apprendono direttamente dall'esperienza senza un modello di riferimento alla dinamica dell'ambiente (come Monte Carlo); e aggiornano le stime su valori di ricompensa già appresi senza attendere il risultato finale dato dalla transizione di stato dell'ambiente (come Programmazione Dinamica).

L'algoritmo che si rifà al metodo 'Q-Learning'[21] consente di apprendere la strategia ottimale anche se le azioni sono scelte in modo casuale o esplorativo. Il metodo si basa sulla massima ricompensa ricevuta per un particolare insieme di azioni nello stato considerato. Nello specifico, si considerano gli stati e le ricompense precedenti per le azioni sugli stati a istanti susseguenti.

Con SARSA (State action reward state action), versione modificata dell'algoritmo Q-Learning, l'agente apprende le azioni da compiere in base all'insieme di azioni nello stato attuale, senza che si prendano in considerazione gli stati e le ricompense precedenti all'istante attuale. Diversamente dal *Q-Learning* non è necessario utilizzare la massima ricompensa per aggiornare la strategia di utilizzo.

[21] Il nome deriva dalla funzione Q, che calcola il beneficio atteso di un'azione nello stato considerato.

Conclusioni

Per i data scientist la scelta dell'algoritmo più utile e funzionale all'obiettivo prefissato non è determinata a priori dal tipo di applicazione, ma spesso è fatta a seguito di diversi tentativi con più modelli di apprendimento. Tale scelta ottimale non segue regole prefissate, in quanto ogni algoritmo ha le proprie caratteristiche e le proprie specifiche, sulle quali i *data scientist* svolgono le proprie analisi. In aggiunta, tale scelta, a parità di caratteristiche e specifiche degli algoritmi, è legata anche all'intuito, se non all'esperienza pregressa dell'operatore, in quanto il ML non è una scienza esatta.

Ad esempio, il livello di interpretabilità dei risultati della fase di *learning*, offre la possibilità di spiegare il rapporto tra dati e risultati (ad es. nel caso di apprendimento supervisionato), o di comprendere le strutture dei rapporti tra dati (ad es. nel caso di apprendimento non supervisionato). In altri casi, il livello di interpretabilità è tale che non si comprende il suddetto rapporto, e bisogna accontentarsi del risultato senza capire come si è arrivati a ciò (*black box*).

Infatti, in questo tipo di operazioni non è raro il trade-off tra l'aumento del livello di interpretabilità di un algoritmo e la diminuzione dell'accuratezza dei suoi risultati, compresi, forse, solo dall'operatore che sceglie quale tecnica utilizzare.

In questo percorso di *data trasformation* propendere verso una direzione o l'altra dipende da variabili quali: dinamiche di condotta delle attività umane, e dalla cultura e dalla tipologia di un'organizzazione (ad es. differenza tra l'applicazione in ambito pubblico o in ambito privato).

In considerazione di ciò, si riportano due esempi che possono spiegare meglio quanto su riportato, ovvero sulla possibilità di abbinare i metodi utilizzati alla natura dell'apprendimento e, naturalmente, all'output desiderato e agli scenari di utilizzo.

Nel primo, De Mauro (2020, 70) riporta una panoramica sintetica degli algoritmi più noti suddivisi nei tre tipi di apprendimento con alcune considerazioni sul livello di interpretabilità.

Figura 5. Selezione di algoritmi di ML organizzati per scenari di utilizzo e tipo di apprendimento

Tipo di apprendimento	Scenario di utilizzo	Esempio di algoritmo	Interpretabilità
Supervied learning (riconosco connessioni che legano i dati in input a una colonna obiettivo)	Regressione (l'obiettivo è un numero)	Linear/logistic regression	Media
		Decision Tree	Alta
	Classificazione (l'obiettivo è una categoria)	Random forest	Bassa
		Neural network	Bassa
Unsupervised learning (riconosco strutture di dati)	Clustering "voglio raggruppare in gruppi omogenei"	K-means	Alta
		Hierarchical clustering	Alta
		Laten Dirichlet Allocation (LDA)	Media
	Riduzione della dimensionalità "voglio ridurre il numero di colonne"	Principal component analysis (PCA)	Bassa
		Singular value decomposition	Bassa
Reinforcement learning (faccio tentativo e imparo dagli errori)	Agente autonomo "voglio imparare a rispondere in base a come varia l'ambiente"	Q-learning	Bassa
		Monte Carlo	Bassa

Fonte: De Mauro (2020)

Nel secondo esempio, si riportano i risultati di un'analisi della letteratura costruita su 120 principali contributi metodologici dell'IA attraverso il ML. Lo studio analizza le principali sfide, tendenze, approcci tecnologici e metodi di ML tra gli anni 2017 e 2021. La struttura concettuale comprende 12 categorie, quattro gruppi (apprendimento supervisionato, apprendimento non supervisionato, apprendimento semi-supervisionato e apprendimento rinforzato) e otto sottogruppi.

Figura 6. Selezione di algoritmi di ML organizzati per scenari di utilizzo e tipo di apprendimento

Tipo di apprendimento	Scenario di utilizzo	Metodologia IA usata	N
Supervised learning	Regression	Artificial Neural Networks (ANN)	2
		Statistical algorithm	1
		Logistic regression	1
		Regression tree	1
	Classification	ANN	13
		Support vector machines (SVM)	3
		Random Forest	2
		Reduced complexity algorithm	1
		Bayesian methods	1
		Decision Tree	1
		Manifold learning	1
Unsupervised learning	Clustering	ANN	11
		K-means	6
		Hierarchical clustering	3
		Proven predictive coding	1
		Hidden Markov model	1
		Graphic schema theory	1
	Dimension reduction	ANN	2
		Geographically weighted regression	1
		Bayesian methods	1
		Principal component analysis	1
Semi-supervised learning	Inductive	ANN	8
		SVM	3
		Graphic schema theory	2
		Bayesian methods	1
		K nearest neighbor	1
		Virtual adversarial training	1
		Gaussian micture model	1
		Fuzzy C-Means	1
		Multiple learning	1
		Mixmatch algorithm	1
	Transductive	ANN	3
		SOGFSE algorithm	1
		Bayesian methods	1
		Gaussian micture model	1
		K nearest neighbor	1
Reinforcement learning	Control	ANN	9
		Direct policy search	3
		Markov's decision processes	2
		Dynamic scheduling	1
		Deep reinforcement learning	1
		Ad-Hoc Techniques	
		Learning the time difference	
	Classification	ANN	6
		Multiple learning	1

Fonte: Serey *et al.* (2021)

Bibliografia

Amir O., Kamar E., Kolobov A., Grosz B.J. (2016), Interactive Teaching Strategies for Agent Training, in Kambhampati S. (ed.), *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16),* pp.804, Palo Alto, California, USA, International Joint Conferences on Artificial Intelligence

De Mauro A. (2020), *Big Data per il Business*, Adria (RO), Edizioni Apogeo

Haenlein M., Kaplan A. (2019), A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence, *Sage Journal,* 61, n.4

Lecun Y., Bengio Y., Hinton G. (2015), Deep learning, *Nature*, 521, pp.436-444

Saunderson S., Bruchansky C. (2019), Evolve or die The Future of Digital Communities, *The Disconnect,* n. 3

Serey J., Quezada L., Alfaro M., Fuertes G., Vargas M., Ternero R., Sabattin J., Duran C., Gutierrez S. (2021), Artificial Intelligence Methodologies for Data Management, *Symmetry*, 13, n.11, art. n.2040

Tosi T. (2020), Qual è la differenza tra machine learning, deep learning e reti neurali?, *SpremuteDigitali.com* <https://bit.ly/3UVpXk3>


INAPP
ISTITUTO NAZIONALE PER L'ANALISI
DELLE POLITICHE PUBBLICHE