Politecnico di Torino

POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

A.A. 2021/2022

Sessione di Laurea Luglio 2022

Tesi di Laurea Magistrale

Machine Learning per la qualità nell'industria 4.0

Relatore

prof. Paolo Garza

Candidato

Omar Burzio

Indice

Capitolo 1

Introduzione

Le tecnologie informatiche necessarie per l'approccio noto come «Qualità 4.0» stanno affermandosi nel mercato a ritmi di crescita sempre più elevati. L'invadenza pervasiva delle tecnologie *Big Data*, sostenute dal ridotto costo del potere computazionale e alimentate da algoritmi di *Machine Learning* e *Deep Learning* via via più sofisticati, sta portando ad una serie di innovazioni che, combinate tra loro, generano effetti sempre più dirompenti.

L'impatto delle tecnologie informatiche sul mondo della produzione è direttamente proporzionale a quanto queste permettano di riprodurre in un ambiente digitale tutto il flusso di informazioni sul prodotto. Dalla fase di ricerca, alla progettazione e all'approvvigionamento, fino alla fase di produzione, distribuzione e servizio post-vendita. La padronanza di tali tecnologie modifica profondamente le regole dello scenario competitivo, dove quell'insieme di *best practices* che pongono l'attenzione sulla buona riuscita e sulla qualità del prodotto rappresentano di certo uno degli aspetti fondanti dell'approccio *Digital Manufacturing*.

Nell'epoca della Industria 4.0 è ormai chiaro come la gestione della qualità non possa più essere considerata in modo analogo a quanto succedeva prima della quarta rivoluzione industriale. I controlli di qualità sono di fatto il principio base del funzionamento delle macchine. Per questo si è arrivati a parlare di *Quality 4.0*, in quanto tutti i processi che

vanno sotto il cappello di «Qualità» tendono ad incorporare, almeno in parte, i concetti cibernetici e di processo che fanno capo alle trasformazioni portate all'industria dai collegamenti tra macchine e computer.

L'approccio tradizionale, che spesso si sostanzia in analisi e misurazioni puntuali della qualità del singolo processo, perde progressivamente di efficacia e lascia spazio ad un modo di considerare la qualità più ampio.

La nuova Qualità 4.0 si caratterizza per essere il principio che informa tutta la raccolta dei dati di produzione. La digitalizzazione dei dati consente di portare i dati diagnostici sulla qualità dal livello della produzione a quello decisionale ed è così che le soluzioni informatiche diventano fondamentali per ottimizzare l'intero sistema di produzione.

Investire per rendere la gestione della qualità efficace ed efficiente significa investire su nuove fonti di vantaggi economici e finanziari.

I vantaggi economici più importanti vengono dall'automazione di processi che, senza un altrettanto adeguato livello di automazione del *Quality Management*, risulterebbero inefficaci. Gli esempi più importanti sono relativi ai processi di gestione delle deviazioni nella qualità dei prodotti (aspetto sul quale ci si è focalizzati in questo elaborato), ai processi di gestione dei reclami, ai processi di *audit*, etc. L'automazione può portare a risparmi concreti sia in termini di tempo speso dagli addetti per gestire un determinato processo (e.g. l'analisi non automatizzata del risultato al termine di un processo di lavorazione di una macchina), che da un punto di vista della eliminazione o riduzione delle perdite economiche legate alla non qualità delle produzioni.

La gestione della Qualità dei fornitori è l'esempio dominante in cui il nuovo approccio alla Qualità mostra tutte le sue potenzialità. L'utilizzo di tecnologie *Cloud Computing* per abilitare la raccolta in tempo reale dei dati di produzione e la loro analisi tramite *Big Data*, così come l'uso di tecnologie intelligenti come il *Machine Learning* e l'Intelligenza Artificiale sta cambiando il modo con cui i produttori e i fornitori parlano di Qualità. Il produttore

riesce ad identificare problemi potenziali di qualità prima del fornitore stesso, con evidenti vantaggi e risparmi in tutta la catena della produzione.

È in questa cornice che si inserisce il lavoro di ricerca, elaborato sull'analisi qualitativa del processo di saldatura delle macchine *Minic* III della *Schunk Sonosystem.*

Lo studio si innesta su uno dei processi produttivi *core* dell'azienda SEWS-CABIND, che opera nel segmento business dell'*automotive* del Sumitomo Electric Group, progettando e producendo cablaggi per la trasmissione di energia elettrica alle varie componenti del veicolo.

L'obiettivo dell'elaborato è dimostrare come l'implementazione a tutto tondo di una «Qualità 4.0», attraverso l'adozione di tecniche di *Machine Learning* e *Deep Learning*, possa portare ad una ottimizzazione del processo di controllo qualità effettuato giornalmente in SEWS-CABIND sulla produzione di cablaggi con saldatura ad ultrasuoni.

Nell'elaborazione dell'analisi empirica ci si è serviti di due algoritmi: l'algoritmo di classificazione noto come *Random Forest*, a cui ha fatto seguito uno studio del data set attraverso la costruzione di una rete neurale artificiale (ANN) multistrato di tipo *feed-forward*.

I risultati hanno dimostrato che entrambi gli algoritmi sono stati in grado di classificare il set di dati a disposizione in modo corretto, costituito da file di log emessi dalle macchine saldatrici al termine di ogni ciclo di lavorazione e memorizzati nel pc con esse integrato.

Essendo gli algoritmi utilizzati per l'analisi di tipo supervisionato, la «correttezza» della classificazione ottenuta va intesa in termini di accuratezza rispetto ai dati presenti nei file di log, riportanti le informazioni sul livello di qualità della saldatura.

È stata dunque dimostrata una sostanziale coincidenza tra i dati dei file di log e l'output del processo di classificazione, ma si noti un aspetto di fondamentale importanza: i file di log contenevano al loro interno le *feature Error Nr* e *Error Text*, indicative della qualità della

saldatura e presenti nel file di log grazie all'intervento manuale dell'operatore eseguito per valutare la qualità della saldatura al termine del processo.
L'analisi empirica elaborata vuole dunque dimostrare come il ricorso ad algoritmi di *Machine Learning* e *Deep Learning* in un tale scenario possa tradursi in una gestione più celere, efficace e senza necessità di intervento manuale sul processo di monitoraggio della qualità nel settore della produzione di cablaggi.

La dissertazione si divide in quattro capitoli. Il primo offre una panoramica sullo sconfinato mondo dell'Intelligenza Artificiale, del *Machine Learning* e del *Deep Learning*, spesso pensati, a torto, come realtà intercambiabili. Focus particolare viene posto sugli algoritmi di classificazione e sulle reti neurali, utilizzati poi nell'analisi empirica. Nel secondo viene presentata la realtà aziendale che ha permesso di elaborare l'analisi su di un caso di business reale, nonché la fase di preparazione ed elaborazione della base dati attraverso l'algoritmo di *Random Forest* e la rete neurale. Il terzo è dedicato all'analisi dei risultati ottenuti con i modelli utilizzati. Seguono le riflessioni a proposito dei possibili sviluppi futuri e le considerazioni finali.

Capitolo 2

Artificial Intelligence, *Machine Learning*, *Deep Learning*: un po' di ordine

Artificial Intelligence, *Machine Learning*, *Deep learning* e Reti Neurali sono termini che spesso il pubblico non specializzato utilizza, a torto, in modo intercambiabile (Figura 2.1).

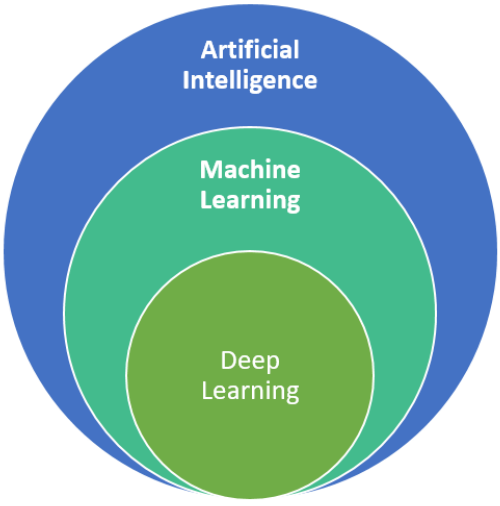

Figura 2.1 Artificial Intelligence, Machine Learning, Deep Learning; *master-iesc-angers.com/artificial-intelligence-machine-learning-and-deep-learning-same-context-different-concepts/*, 24/01/2022.

Il modo più semplice ed efficace per spiegare la differenza, ma soprattutto la condizione di interdipendenza che caratterizza gli elementi di cui sopra è pensare a loro come scatole cinesi. Ciascuno è essenzialmente un componente del termine precedente.

Il *Machine Learning* è un sottocampo dell'*Artificial Intelligence*. Il *Deep Learning* è un sottocampo del *Machine Learning* e le reti neurali costituiscono la spina dorsale degli algoritmi di *Deep Learning*.

Possiamo altresì dire che l'Intelligenza Artificiale sia la disciplina di base e il *Machine Learning* e il *Deep Learning* le tecniche, o meglio, i modelli che ne consentono l'applicazione.

Con il termine generico di *Artificial Intelligence* o Intelligenza Artificiale si fa riferimento a ad un insieme di tecniche e metodi che mirano alla realizzazione di macchine capaci di gestire problemi ed attività tipiche dell'intelligenza umana [1].

Gli studiosi Shai Shalev-Shwartz e Shai Ben-David hanno invece definito il *Machine Learning* come "the automated detection of meaningful patterns in data" [2]. La pratica di *Machine Learning* o Apprendimento Automatico consiste dunque nella creazione di sistemi che apprendono o migliorano le performance in base ai dati utilizzati.

Il *Deep Learning* o Apprendimento Approfondito è il ramo più avanzato del *Machine Learning*. Definito da alcuni come un "*Machine Learning* scalabile" [3], si compone di reti neurali artificiali organizzate in diversi strati, in cui ogni strato calcola i valori per quello successivo. La principale differenza tra i due ambiti risiede per l'appunto nell'autoapprendimento. Mentre il *Deep Learning* combina potenza di calcolo e reti neurali, l'Apprendimento Automatico sfrutta algoritmi fissi, che trattano i dati a disposizione secondo schemi definiti.

L'analisi empirica sviluppata nel presente lavoro di ricerca si è concentrata sull'applicazione di tecniche di *Machine Learning* e *Deep Learning* all'ambito dell'industria

4.0 e di come queste, se opportunamente sviluppate, possano portare a livelli qualitativi di lavoro difficilmente ignorabili.

Prima di entrare nel vivo della trattazione, pare opportuno introdurre al lettore l'architettura all'interno della quale questa si inserisca. La struttura del presente capitolo è tale perché intende guidare il lettore attraverso un percorso che, dalle nozioni più generali in materia di Intelligenza Artificiale, arrivi a spiegare la natura dell'apprendimento di una macchina e come questo avvenga grazie allo sviluppo di algoritmi e all'ausilio di reti neurali, ideate per simulare il funzionamento del cervello umano. Il focus viene posto sugli algoritmi di classificazione e sulle reti neurali, utilizzati nella fase sperimentale.

2.1 L'Intelligenza Artificiale: diversi approcci e un'evoluzione piuttosto recente

Se sull'attributo «artificiale» non è così difficile trovare una comune definizione, sul cosa sia l'intelligenza è un tema ampiamente discusso, che porta con sé soluzioni diverse da disciplina a disciplina.

John McCarthy, professore emerito di Informatica all'Università di Stanford e noto per aver vinto il Premio Turing nel 1971, in un *paper* del 2004 sottolinea che l'Intelligenza Artificiale non si limita alla comprensione degli aspetti prettamente biologici dell'intelligenza umana, ma che può essere considerata come la parte computazionale della capacità di raggiungere obiettivi [4].

L'AI è dunque quel settore che ha come scopo lo studio e la progettazione di macchine che siano in grado di emulare le capacità della mente umana e, attraverso l'esperienza, migliorare.

Marco Somalvico, considerato il padre dell'Intelligenza Artificiale in Italia, la qualifica come la disciplina capace elaborare sistemi con prestazioni che i più definirebbero proprie esclusivamente dell'intelligenza umana [5].

Anche in questo caso, si ritiene che l'IA abbia come scopo quello di riprodurre l'intelligenza umana e che quindi non si debba soltanto limitare a replicarla.

Ancora più recente e completa è la definizione proposta dall'*AI HLEG*[1] in seno alla Conferenza Europea del 2018 dei sistemi di Intelligenza Artificiale, software e hardware progettati dall'uomo che, attraverso regole simboliche o modelli matematici, interpretano ed elaborano le informazioni dall'ambiente circostante per raggiungere un obiettivo prefissato [6].

Considerare esaustivo l'elenco di cui sopra sarebbe riduttivo e non permetterebbe di cogliere la diversità di approcci caratterizzante la storia di questa disciplina. Nel definirla taluni hanno posto l'attenzione ai processi di pensiero e al ragionamento, altri invece al comportamento della macchina.

I successivi due paragrafi vogliono essere un approfondimento sul tema, offrendo una panoramica sulla storia più recente e i campi di applicazione dell'Intelligenza Artificiale.

2.1.1 Cenni storici sull'Intelligenza Artificiale

Sebbene la nascita dell'Intelligenza Artificiale sia riconducibile a studi informatici, viene ad oggi considerata come una disciplina autonoma. Non manca, tuttavia, una forte

[1] L' *AI HLEG* o *High-Level Expert Group on Artificial Intelligence* rappresenta un gruppo di esperti indipendenti di IA costituito dall'Unione Europea nel giugno del 2018. Maggiori informazioni sono disponibili all'indirizzo *https://www.europarl.europa.eu/cmsdata/196377/AI%20HLEG_Ethics%20Guidelines%20for%20Trustworthy%20AI.pdf*, 20/01/2022.

influenza da parte di altre discipline come la matematica, la psicologia, la filosofia e le scienze cognitive.

Non è un caso che la sua nascita venga collegata al matematico, logico, crittografo, filosofo, nonché uno dei padri fondatori dell'Informatica: l'inglese Alan Mathison Turing, che ne parlò per la prima volta in un articolo comparso nel 1950 sulla rivista Mind [7]. Passato alla storia per la sua famosa frase "*Can machines think*?", egli propose quello che sarebbe diventato il primo esperimento mai ideato per la misurazione dell'intelligenza delle macchine: il Test di Turing[2].

I primi lavori di ricerca in ambito IA, si veda il contributo di Warren McCulloch e Walter Pitt, si focalizzarono sulle reti neurali: modelli matematici ispirati alle reti neurali biologiche e composti da neuroni artificiali [8]. Sulla scia di questi studi, nel 1951 il matematico e scienziato americano Marvin Lee Minsky inventò SNARC[3], il primo computer a rete neurale capace di simulare una rete di 40 neuroni [9].

Esiste un evento nella storia dell'IA, diventato famoso per essere ricordato come ufficiale e fondativo: il workshop sulle macchine pensanti organizzato da un allora giovane assistente di matematica del Dartmouth College, John McCarthy. Al consesso parteciparono le menti più brillanti del tempo: lo scienziato Marvin Lee Minsky della Harvard University, l'informatico Nathaniel Rochester di IBM Corporation e l'ingegnere Claude Elwood Shannon dei Bell Telephone Laboratories. "*Every aspect of any other feature of learning or intelligence should be accurately described so that the machine can simulate it*" fu una delle frasi più famose pronunciate durante la conferenza, dove per la prima volta venne usato il termine «*Artificial Intelligence*» [10].

[2] L'esperimento si basava sull'interrogativo per cui una macchia fosse capace o meno di imitare un essere umano in una conversazione.

[3] Acronimo di *Stochastic Neural Analog Reinforcement Computer*.

Fu poi il turno dello psicologo americano Frank Rosenblatt, che nel 1967 costruì *Mark 1 Perceptron*, il primo computer basato su una rete neurale. Una macchina di cinque tonnellate, considerata "the embryo of an electronic computer that will be able to walk, talk, see, write, reproduce itself and be conscious of its existence" [11].

momento di flessione nello sviluppo dell'IA, pubblicarono *Perceptrons*, dimostrando i limiti della struttura *feed-forward* a due strati di Rosenblatt [12].

Il periodo compreso tra gli anni 1974 e 1980 è conosciuto come il «Primo inverno dell'Intelligenza Artificiale». Seguì una breve parentesi in cui fecero la loro comparsa i così detti «Sistemi Esperti» come XCON[4] e le macchine LISP.[5] L'onda dell'entusiasmo non durò molto e nel 1978 ebbe inizio il «Secondo inverno dell'Intelligenza Artificiale». Sarebbe durato fino al 1993.

Negli anni '90 si assistette alla nascita del *World Wide Web* e all'ingresso sul mercato dei processori grafici, le *Graphics Processing Unit*,[6] con una conseguente ampia e rapida diffusione di Internet, consentendo l'accesso a grandi quantità di informazioni e conoscenze e aprendo nuove prospettive per l'IA.

Nel 2006 la professoressa di Informatica dell'Università di Standford Fei-Fei Li contribuì con i suoi studi ad un cambio di paradigma epocale, ipotizzando che il vero limite allo sviluppo dell'Intelligenza Artificiale fosse da ricercare nella quantità di dati a disposizione: una maggiore quantità di dati avrebbe comportato lo sviluppo di modelli sempre più avanzati.

Andrew Ng, professore dell'Università di Standford ha di recente definito l'IA come "*the new electricity*" [13].

[4] Sistema inventato da John McDermott alla Carnegi-Mellon University; acronimo di *expert configure*.

[5] Il linguaggio LISP fu inventato da John McCarthy nel 1958 presso il MIT.

[6] Detti anche *GPU*, sono chip di elaborazione dati molto più veloci delle *CPU* (*Central Processing Unit*), eredità dell'universo del *gaming*, possiedono capacità computazionali molto elevate.

A farla da padrona negli ultimi anni sono stati i potenti sistemi di *Deep Learning*, nati dall'addestramento di reti neurali molto profonde mediante l'ausilio di esempi etichettati. Capaci di riconoscere pressoché ogni cosa con un'accuratezza simile a quella umana, hanno determinato progressi straordinari in campi come il riconoscimento delle immagini e della voce, per citarne alcuni.

Applicazioni come Siri e Alexa si sono fatte strada tra un pubblico entusiasta, lo stesso che ha ipotizzato la venuta di una così detta *superintelligence*. Nick Bostrom nel suo libro ne ha dato una definizione: "any intellect that vastly outperforms the best human brains in practically every field, including scientific creativity, general wisdom, and social skills" [14].

Dall'altro capo del tavolo non mancano però gli scettici e coloro che pongono l'accento sulle pesanti implicazioni etiche e sociali che uno sviluppo incontrollato dell'IA potrebbe avere [15] e chi addirittura auspica un nuovo «inverno».

2.1.2 Impatti e campi di applicazione dell'Intelligenza Artificiale

Che l'Intelligenza Artificiale abbia conosciuto un'espansione senza eguali negli ultimi decenni, arrivando a permeare molti aspetti della vita umana, è materia conosciuta. Tale espansione ha favorito nel tempo il crearsi di posizioni diverse nell'opinione pubblica. Da un lato i sostenitori, intenzionati a sfruttarne le enormi potenzialità e dall'altro i detrattori, i soggetti che più di tutti si trovano a subirne passivamente le conseguenze.

Il 2017 può essere considerato l'anno in cui le potenze mondiali hanno dato il via alla corsa internazionale per lo sviluppo dell'IA [16]. Prima fra tutte la Russia, dove il presidente Vladimir Putin, in un discorso agli studenti in occasione del dibattito internazionale sullo sviluppo dell'IA, ha affermato che "chiunque diventerà il leader in questa sfera, diventerà il governante del mondo" [17]. A seguire l'India, che nello stesso

anno ha istituito una *Task Force* per studiarne gli effetti politici ed economici e la Cina, che si è posta l'obiettivo di diventare una superpotenza nel campo entro il 2030 [18]. [7]

Numerosi studi ed indagini aiutano a capire quale sia l'attuale livello di diffusione ed impiego dell'IA. L'indice noto come *AI Readiness Index 2020* e sviluppato da *Oxford Insights* e dall'*International Research Development Centre* (*IDRC*) per calcolare il livello di preparazione degli stati nel capo dell'IA8 pone in vetta alla classifica gli Stati Uniti, seguiti da Gran Bretagna, Finlandia, Germania e Svezia, per posizionare l'Italia ad un magro trentacinquesimo posto [19].

E ancora, in un'indagine della Commissione Europea fatta su 9640 aziende tra gennaio e marzo del 2020, emerge come nell'Eurozona sia alto il livello di consapevolezza sui benefici apportati dall'IA (78%) e di come circa un 42% delle aziende abbiano adottato almeno una tecnologia IA. La percentuale diminuisce sensibilmente se si parla dell'adozione di almeno due tecnologie (25%). Tra le cause frenanti si annoverano: la difficoltà di reperire forza lavoro specializzata e competente (57%), i costi correlati alla nuova tecnologia (52%) e la difficoltà di adattarvi i processi operativi (49%) [20].

L'Intelligenza Artificiale sta colonizzando quasi tutti i settori dell'economia: *Chatbot*, automobili a guida autonoma, macchine 4.0, tutte avvisaglie di una quarta rivoluzione industriale di cui l'IA ne è il propulsore. Grande diffusione che porta con sé anche critiche e scetticismi in più campi, da quello sociale ed etico, al politico e alla sicurezza.

In un recente rapporto *The Rise of Artificial Intelligence: Future Outlook and Emerging Risks*, la *Allianz Global Corporate & Specialty*[9] identifica rischi e benefici legati alla crescente

[7] Come emerge dal *Next Generation Artificial Intelligence Development Plan*.

[8] L'Indice classifica la capacità dei governi di implementare l'IA nella fornitura dei loro servizi pubblici. Si basa su 33 indicatori applicati a 10 dimensioni e funge da strumento per confrontare lo stato attuale di preparazione in ambito di IA dei governi di tutto il mondo.

[9] *AGCS* è uno dei principali provider di soluzioni assicurative per le aziende a livello globale.

diffusione dell'IA nella società e nell'industria [21]. In campo economico si assisterà ad un aumento del PIL pro capite generato da lavori più performanti. Vi farà però da contraltare l'aumento della disoccupazione, a causa della progressiva sostituzione di lavori a medio - basso reddito. E ancora, non infondate sono le paure che le tecnologie avanzate possano essere utilizzate in campo politico per influenzare l'elettorato o favorire attacchi informatici verso le istituzioni. Pareri più positivi ci sono invece a proposito dei potenziali miglioramenti per quanto riguarda la mobilità (veicoli a guida autonoma), la sanità (analisi avanzata dei dati) e l'ambiente (città intelligenti).

In questa partita pare quasi impossibile schierarsi in modo netto. Stuart Russel, pioniere ed esperto di IA, coglie con estrema maestria che il vero nocciolo della questione risiede nel "problema dell'allineamento del valore", ovvero nell'allineare i valori e gli obiettivi dell'IA con quelli umani [22].

Poiché Google, Facebook e altre aziende stanno attivamente cercando di creare una macchina intelligente, una delle cose che non dobbiamo fare è andare avanti a tutto vapore senza pensare ai rischi potenziali. Se si vuole una intelligenza illimitata, è meglio capire come allineare i computer con i valori e i bisogni umani [23].

Quando le macchine non rispetteranno i nostri stessi valori, allora avranno inizio i problemi. Vanno letti in questa direzione gli accordi *OECD principles on AI* [24], siglati nel 2019 da quarantadue Paesi[10] e il *Global Partnership on Artificial Intelligence* (*GPAI*)[11] del 2020 [25], entrambi ispirati a principi di uno sviluppo etico e responsabile dell'IA.

[10] I Paesi firmatari si impegnano a sviluppare sistemi di IA ispirandosi a principi di sicurezza, equità ed affidabilità.

L'interpretazione vincente dell'IA contemporanea pare esser quella in grado di comprendere e superare i limiti della *Strong Artificial Intelligence*, per arrivare a vestire i panni di una soluzione ibrida capace di risolvere specifici problemi o effettuare ragionamenti dove la mente umana fatica ad arrivare. Non ci si aspetta più che una macchina sia dotata di coscienza o abilità cognitive, ma che sia capace di risolvere in maniera efficiente problemi in campi d'azione differenti.

Un'ottima panoramica sulle attuali soluzioni applicative dell'IA è fornita da uno studio del 2017 della *Forrester Research*, società americana specializzata in ricerche di mercato e consulenza sugli impatti della tecnologia. La Figura 2.2 offre un resoconto sulle prime dieci tecnologie IA, organizzate secondo livello di maturazione e diffusione [26].

[11] Iniziativa internazionale e *multistakeholder* per uno sviluppo ed uso responsabile dell'IA, fondato sul rispetto dei diritti umani ed ispirato a valori come inclusione, diversità, innovazione e crescita economica. Sottoscritto nel 2020 da quattordici governi insieme all'Unione Europea.

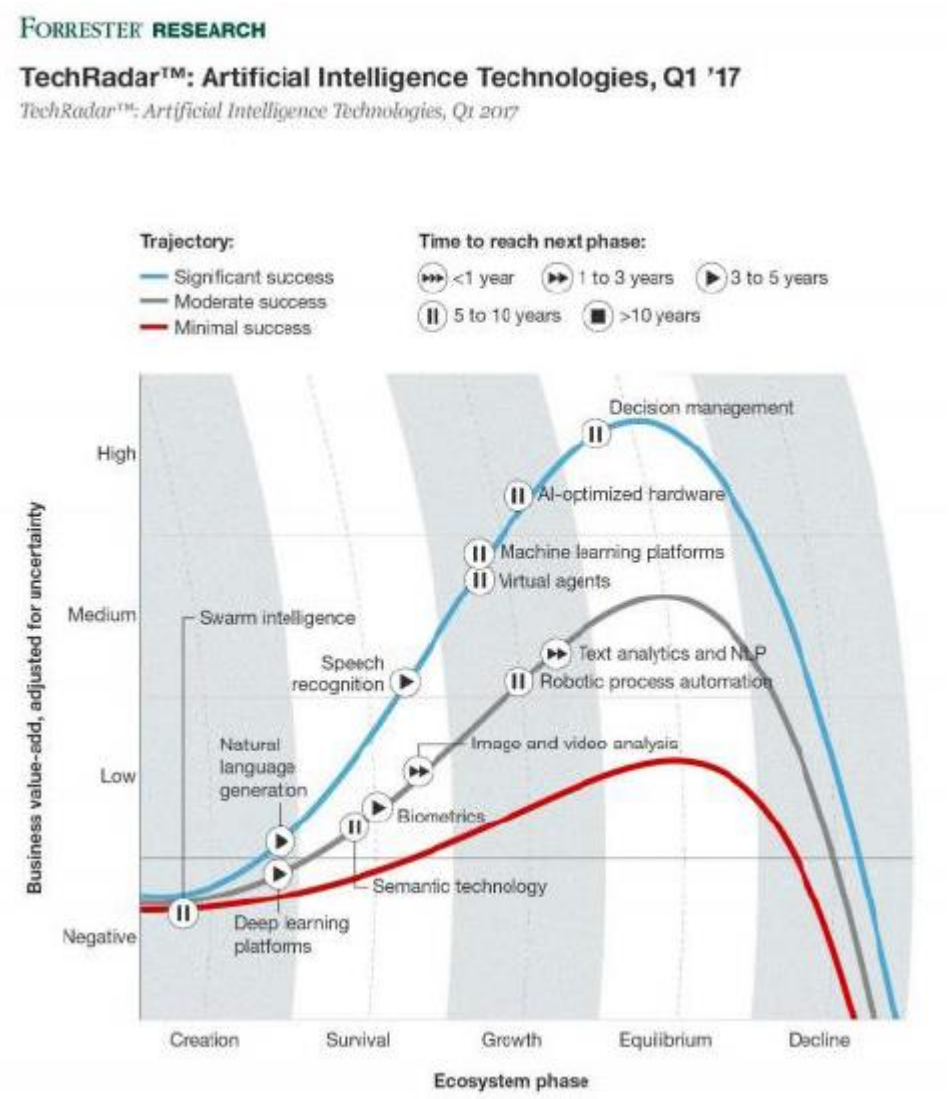

Figura 2.2 Prime dieci tecnologie IA, organizzate secondo livello di maturazione e diffusione; *https://www.forrester.com/report/The-Top-Emerging-Technologies-To-Watch-2017-To-2021/RES133144*, 27/01/2022.

Tra le applicazioni più interessanti dell'IA si citano la *Natural Language Processing* (NLP) o elaborazione del linguaggio naturale, una tecnologia in grado di processare il discorso umano e trasformarlo in testo scritto. Altra tecnologia è la *Speech Recognition*, utilizzata

per trasformare il linguaggio parlato in codice, utile per applicazioni software, assieme alla *Text Analytics*, utilizzata per fare analisi su testi e documenti. E ancora, il *Virtual Agent* o assistente virtuale, largamente utilizzato nell'*Help Desk* di molte aziende sottoforma di *chatbot* o software intelligenti, in grado di sostituirsi all'uomo nello svolgimento di mansioni lunghe e ripetitive (*Process Robotic Automation*). Piattaforme di *Machine Learning* e *Deep Learning* che, tramite algoritmi di apprendimento automatico basati su reti neurali artificiali, vengono impiegate per il riconoscimento e la classificazione di pattern (e.g. riconoscimento di sagome o volti da parte di sistemi di archiviazione di foto come nelle applicazioni di Google Photo o iCloud). Per non parlare della più recente diffusione dell'IA in campo medico o per la gestione automatizza di piattaforme di *trading*, in grado di generare milioni di transazioni al giorno in completa autonomia [27].

Molte realtà stanno investendo ingenti somme di denaro per lo sviluppo di sistemi intelligenti commerciabili. Si citano di seguito alcuni esempi degni di nota.

Nel 2017 l'imprenditore e visionario Elon Musk, con la *Neuralink Corporation*, rende pubblico il suo progetto di connettere computer e cervello umano. Un articolo apparso sul *Wall Street Journal* spiega che l'idea della start-up creata dall'imprenditore sudafricano è di studiare le malattie del cervello [50]. In Italia il supercomputer Leonardo, concepito e gestito dal Cineca, verrà installato nel Tecnopolo di Bologna a fine 2022, proiettando l'Italia verso il calcolo per la ricerca e l'innovazione tecnologica di classe *exascale* [28].

Per quanto riguarda il campo medico, è interessante conoscere Watson, sistema cognitivo di IBM, che ha introdotto nelle corsie degli ospedali l'intelligenza artificiale a supporto dell'insegnamento e della ricerca. Daniela Scaramuccia[12] parla di

[12] *Director Health & Life Science Industries IBM* Italia.

un tutor cognitivo per assistere e potenziare le capacità di medici e studenti. Il tutor offrirà piattaforme di studio personalizzate attraverso la scelta di contenuti, simulazioni, commenti e approfondimenti basati sul livello di conoscenze del singolo studente mediante una semplice interfaccia quale può essere una app [29].

Gli esempi applicativi riportati rappresentano solo una piccolissima parte di quanto oggi sviluppato ed investito nel settore dell'Intelligenza Artificiale. Le previsioni di crescita vanno infatti dai circa 15 miliardi di dollari attuali, agli oltre 35 miliardi nel 2025 (Figura 2.3).

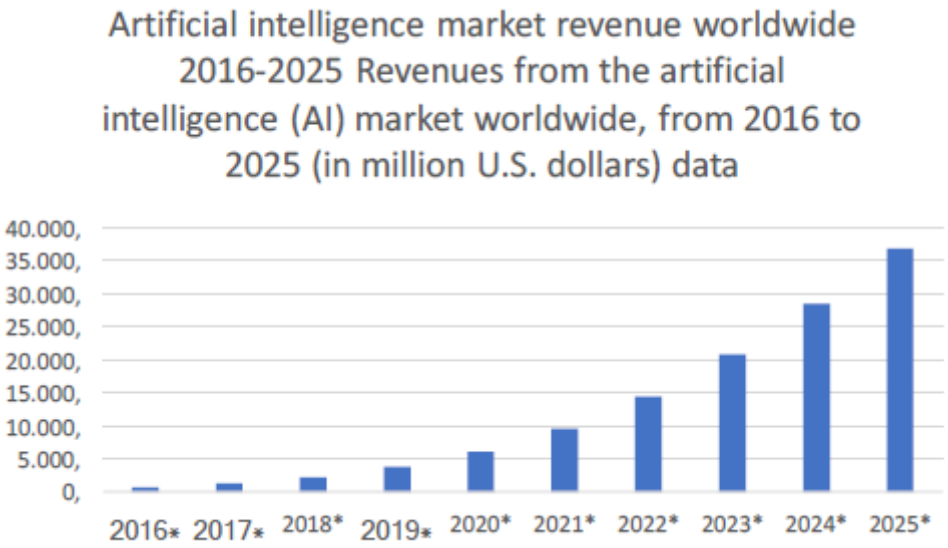

Figura 2.3 Ricavi dell'Intelligenza Artificiale dal 2016 al 2025, espressi in milioni di dollari; *https://www.statista.com/statistics/607716/worldwide-artificial-intelligence-market-revenues/*, 27/01/2022.

2.2 Il *Machine Learning*

Dopo aver introdotto al lettore definizioni e caratteristiche del primo elemento della struttura[13], si intende proseguire con l'analisi dei rimanenti, per giungere al cuore della trattazione avendo ben chiaro il contesto all'interno del quale si inserisce l'oggetto: lo studio e l'analisi qualitativa in ambito industriale effettuata con l'utilizzo di algoritmi di *Machine Learning* e *Deep Learning*.

Nella prima sezione del paragrafo ci si interroga su cosa sia l'apprendimento per una macchina. Fa seguito una breve parentesi sulla genesi e i campi di applicazione del *Machine Learning* ed infine l'intento di approfondire uno degli strumenti attraverso il quale l'apprendimento delle macchine prende forma: gli algoritmi di classificazione, vero focus dell'analisi assieme alle reti neurali.

2.2.1 L'apprendimento per una macchina

Il *Machine Learning* è un campo dell'Informatica che studia i sistemi e gli algoritmi capaci di apprendere direttamente dai dati senza essere stati preventivamente programmati [30].

Il termine *Machine Learning* fu utilizzato per la prima volta nel 1959 da Arthur Lee Samuel, scienziato americano e pioniere nel campo dell'Intelligenza Artificiale [31], per riferirsi a quella branca dell'informatica che utilizza algoritmi che dettano le regole per la lettura e l'interpretazione delle informazioni.

Prima di proseguire nella trattazione bisogna però dare una definizione più precisa di «apprendimento». Si cita a tal proposito il direttore del dipartimento *Machine Learning*

[13] Nei paragrafi precedenti è stato ampiamente esposto il concetto di interdipendenza e struttura a «scatole cinesi» di *Artificial Intelligence*, *Machine Learning* e *Deep learning*.

della *Carnegie Mellon University*, il professor Tom Michael Mitchell: "Un programma apprende da una certa esperienza se, dato un problema con la rispettiva misura della prestazione, la prestazione nel risolvere il problema è migliorata dall'esperienza [32]".

Il compito degli algoritmi di *Machine Learning* si traduce nell'elaborazione di esempi descritti da caratteristiche o *feature* fornite in input sottoforma di vettore di n elementi, dove n è il numero di *feature*. L'abilità dell'algoritmo viene solitamente misurata in termini di performance, che rappresenta la capacità del modello di risolvere un determinato problema.

Nei problemi di classificazione (trattata in modo approfondito nei paragrafi seguenti in quanto di notevole rilevanza per l'analisi) la performance viene solitamente intesa in termini di accuratezza del modello, misurata come la percentuale di campioni correttamente predetti. Un altro parametro di riferimento può essere il tasso di errore, ovvero la percentuale di campioni erroneamente predetti.

Gli algoritmi di *Machine Learning* possono essere categorizzati in base al tipo di esperienza a cui sono sottoposti durante il processo di apprendimento. Si è soliti declinare i paradigmi dell'Apprendimento Automatico come segue:

- supervisionato: si costruisce un modello a partire da dati di addestramento etichettati, in quanto l'obiettivo è elaborare una regola che associ input e output corrispondente;
- non supervisionato: si costruisce un modello a partire da dati di addestramento non etichettati o senza un corrispondente valore di output, in quanto l'obiettivo è identificare dei pattern negli input al fine di riprodurli o prevederli;
- con rinforzo: l'obiettivo è costruire un sistema che, attraverso le interazioni con l'ambiente, migliori le proprie performance.

La categorizzazione di cui sopra va fatta risalire a Samuel, che negli anni '50, basandosi sulla modalità di apprendimento cui possono essere sottoposte le macchine, distinse tra

macchine provviste di esempi completi nell'esecuzione del compito richiesto (apprendimento supervisionato) da software lasciati lavorare «in autonomia» (apprendimento non supervisionato e apprendimento con rinforzo) [33].
Si noti che particolare attenzione viene posta sul primo approccio, poiché utilizzato nella fase sperimentale e che, per dovizia di completezza, si accenna anche gli altri due.

L'apprendimento con supervisione prevede che al modello da allenare venga fornito in input un set di addestramento dove l'etichetta di classe è nota. Considerando y la variabile di output e x il vettore di caratteristiche dei campioni di ingresso, il modello deve capire come predire durante la fase di addestramento y da x, stimando la probabilità $p(y|x)$ [34].
L'obiettivo di questo tipo di apprendimento è di riuscire a costruire modelli in grado di eseguire previsioni in condizioni d'incertezza, in modo tale da riuscire a predire la variabile y nel momento in cui viene fornito un nuovo input [35]. Viene definito apprendimento supervisionato perché l'algoritmo esegue iterativamente delle previsioni che, se errate, vengono corrette da un supervisore, finché il livello di precisione del programma non viene ritenuto sufficiente.
A titolo esemplificativo si ricorda un famoso data set introdotto da Fisher nel 1936 [36] e utilizzato nell'ambito dell'apprendimento automatico come esempio di classificazione statistica [37], l'iris data set. Si tratta di un data set composto da tre classi di cinquanta istanze l'una e quattro variabili per ogni istanza, dove ogni classe si riferisce ad un tipo di pianta di iris e le quattro variabili sono la lunghezza e larghezza del petalo e del sepalo. Annotando ogni specie di iris presente nel data set si può allenare un sistema ad apprendere come classificare le piante di differenti specie basandosi su alcune variabili morfologiche.
Gli algoritmi che fanno uso di apprendimento supervisionato vengono utilizzati in molti settori, da quello medico a quello di identificazione vocale e hanno la capacità di

effettuare ipotesi induttive, ossia ipotesi che possono essere ottenute scansionando una serie di problemi specifici per ottenere una soluzione idonea ad un problema di tipo generale.

In base al tipo di output desiderato è possibile operare un'ulteriore classificazione all'interno degli algoritmi di *Machine Learning* supervisionato in [38]:

- algoritmi di regressione: l'output assume valori continui;
- algoritmi di classificazione: l'output assume valori finiti.

La regressione è una tecnica utilizzata nell'apprendimento con supervisione dove si dispone di un numero di variabili predittive (indipendenti) e una variabile target continua (dipendente) e l'obiettivo è trovare un'eventuale relazione funzionale tra la variabile dipendente e le variabili indipendenti. Un esempio di utilizzo di algoritmi di regressione lo si trova nella predizione del valore del tasso di cambio di una valuta nel futuro, dati i suoi valori in tempi recenti.

Nella classificazione gli output sono divisi in due (classificazione binaria) o più classi (classificazione multi - classe) e il sistema di apprendimento deve produrre un modello che assegni gli input non ancora visti a una o più di queste classi; i valori della risposta prevista sono sempre e solo valori discreti. I filtri antispam rappresentano un classico esempio di classificazione, dove gli input sono le mail e le classi sono «spam» e «non spam».

Il modello elaborato in questa rientra nella categoria dell'apprendimento supervisionato con algoritmi di classificazione.

L'apprendimento non supervisionato prevede che il sistema da allenare sia in possesso degli input ma senza una corrispondenza con gli output. L'obiettivo, in questo caso, è mettere l'agente nella condizione di trovare pattern nascosti o strutture intrinseche nei dati che sono stati forniti. È così possibile fare inferenze dal gruppo di dati, senza avere una possibile risposta esatta e un supervisore che corregga l'eventuale errore. Viene utilizzato quando il problema richiede una quantità enorme di dati non etichettati.

Capire il significato dietro a questi dati richiede algoritmi che classificano i dati in base ai modelli o ai cluster rilevati. L'apprendimento senza supervisione conduce un processo iterativo, analizzando i dati senza intervento umano. Google News è un esempio calzante di apprendimento non supervisionato. Basato sull'analisi di centinaia di migliaia di nuovi contenuti, seleziona e raggruppa tutte le notizie coerenti tra loro. Altro tipico esempio sono i filtri antispam delle caselle elettroniche, dove classificatori di *Machine Learning* basati su *clustering* e associazione vengono utilizzati per identificare mail indesiderate.

Il terzo ed ultimo tipo di apprendimento viene definito *Reinforcement Machine Learning*. È un paradigma di apprendimento automatico basato sull'analisi dei feedback, premi e penalità. È detto anche «meritocratico» perché una ricompensa incoraggia i comportamenti corretti dell'agente.

La principale differenza tra questo paradigma ed il *Machine Learning* supervisionato e non supervisionato sta nel fatto che l'addestramento della macchina non si basa su un data set di dati (i.e. training set), ma alla macchina viene fornito un obiettivo da raggiungere e una funzione premia le azioni che la avvicinano al raggiungimento dell'obiettivo e punisce quelle che la allontanano.

In generale, l'algoritmo di *Reinforcement Learning* modifica il proprio comportamento per ricevere più premi e ridurre al minimo le punizioni. Così facendo crea e perfeziona un modello decisionale o di classificazione.

Questo tipo di apprendimento è tipico delle auto senza pilota, che grazie a un complesso sistema di sensori di supporto sono in grado di percorrere strade riconoscendo eventuali ostacoli e seguendo le indicazioni stradali.

2.2.2 Breve storia e campi di applicazione dell'Apprendimento Automatico

Il lettore può ben immaginare come la storia dell'IA e del *ML* siano strettamente connesse. Si ripercorrono di seguito i momenti di maggior interesse nella storia del *Machine Learning*.

Correva l'anno 1952, quando Arthur Samuel inventò il primo programma in grado di auto apprendere, sfatando il mito per cui un computer fosse solo il grado di eseguire un compito assegnato [39]. Il programma era applicato al gioco della dama ed era in grado di migliorarsi giocando contro se stesso e contro gli avversari, utilizzando la logica dell'apprendimento supervisionato.

Pochi anni dopo, nel 1957, Frank Rosenblatt introdusse le reti neurali artificiali, strettamente collegate con l'Apprendimento Automatico e l'Intelligenza Artificiale [40].

Esattamente dieci anni dopo Cover e Hart crearono il primo programma in grado di riconoscere dei pattern. L'algoritmo alla base del funzionamento del programma si chiamava *nearest neighbour algorithm* ed era in grado di eseguire uno studio comparativo tra la conoscenza pregressa e quanto di nuovo esperienziato: la sua capacità consisteva nel classificare i nuovi oggetti sottopostigli in base al confronto con quanto di più simile avesse in memoria [41].

Un'altra tappa importante per la storia dell'Apprendimento Automatico la si trova agli inizi degli anni '80, quando Gerald DeJong introdusse l'*explanation based learning*[14], una tecnica di apprendimento automatico basato sui domini e sulla generalizzazione di concetti tramite degli esempi pratici [42].

[14] La tecnica EBL o Apprendimento basato sulle spiegazioni, analizza l'ambiente operativo X ed elimina le caratteristiche irrilevanti per concentrare l'analisi soltanto su quelle rilevanti. Infine, generalizza una regola generale di comportamento Y. Una volta trovata la regola generale Y, il programma salva in un database la coppia (X,Y) per evitare di doverla ricalcolare ogni volta.

Fu poi la volta dei sistemi di sintesi vocale[15] basati su reti neurali come NETtalk [43] e la sua versione migliorata NETspeak [44], in grado di convertire un testo in inglese da scritto a vocale dopo essere stati allenati con una base di discorsi informali.

Per tutti gli anni '90 la ricerca continuò ad applicare l'Apprendimento Automatico ai primi programmi di *data mining*[16], *adaptive software*[17] e applicazioni web. Accanto all'apprendimento supervisionato e non supervisionato si posero le basi per un nuovo tipo, quello per rinforzo.

Dagli anni 2000 in poi quasi tutte tra le più grandi società del mondo compresero l'importanza di sviluppare programmi di IA integrati con il *ML* per la gestione dei così detti *Big Data*.

Con il termine *Big Data* si intende quella tecnologia di aggregazione ed elaborazione di una quantità infinita di dati, provenienti da altrettante infinite fonti [45]. La complessità, data dalle dimensioni di queste informazioni, rende impossibile il loro calcolo con le tecniche di elaborazione tradizionali [46].

L'importanza dei *Big Data* nel contesto del *Machine Learning* deriva dal fatto che l'analisi di una così grande mole di dati, che il professor Martin Hilbert stimò nel 2007 essere più di

[15] Text-To-Speech (TTS). Un sistema di sintesi vocale è composto da due parti: una *front-end* e una *back-end*. La parte *front-end* si occupa della conversione del testo in simboli fonetici, mentre la parte *back-end* interpreta i simboli fonetici e li legge, trasformandoli così in voce artificiale.

[16] Il *data mining* è un concetto correlato al ML. Esso introduce tecniche e metodologie con le quali è possibile estrapolare un sapere o una conoscenza da grandi quantità di dati. Tale tecnica ha come obiettivo quello di identificare pattern e regolarità, che consentano poi di ipotizzare e verificare nuove relazioni di tipo causale fra i fenomeni del sistema, o di formulare previsioni di tipo statistico su nuovi dati.

[17] E' un software specializzato progettato per utenti con disabilità fisiche. Questo software di solito funziona su un hardware specializzato. E' anche noto come software «assistivo».

300 exabyte [47], permette di dare loro un senso, scoprendo ed elaborando nuove tendenze e modelli.

Per comprendere la correlazione tra *Machine Learning* e *Big Data* è sufficiente ricordare che questi ultimi rappresentano la fonte tramite cui una macchina «fa esperienza» nel tempo [48]. È ormai noto che per insegnare ad una macchina a sviluppare una propria capacità di apprendimento è necessario fornirle delle informazioni preliminari, attraverso le quali riuscire poi a formulare un «ragionamento» ed elaborare una risposta sotto forma di output. Per adempiere a questo compito il *Machine Learning* utilizza degli algoritmi, che processano le informazioni ricevute e sviluppano nuove capacità.

Sebbene possa sembrare un concetto complesso, le applicazioni di *Machine Learning* sono largamente diffuse e alcune di queste ci accompagnano nella vita di tutti i giorni. Dalle *SERP* (*Search Engine Results Page*), liste di risultati con informazioni attinenti alle ricerche effettuate sul web che i motori di ricerca restituiscono grazie all'utilizzo di algoritmi di *ML* non supervisionato, ai filtri antispam delle mail. Una tecnologia molto simile ma più sofisticata viene impiegata nel settore *finance* per la prevenzione delle frodi, dei furti di dati e di identità. In questo caso gli algoritmi imparano ad agire mettendo in correlazione eventi, abitudini degli utenti, preferenze di spesa, etc. e sulla base di queste informazioni riescono ad identificare eventuali comportamenti anomali.

Larga diffusione nel settore della ricerca scientifica e in campo medico hanno avuto algoritmi ad apprendimento supervisionato, utilizzati per prevenire il verificarsi di epidemie o per effettuare diagnosi di tumori o malattie rare in modo accurato e tempestivo. E ancora, sempre nell'ambito del *Machine Learning* supervisionato, si citano le applicazioni per il riconoscimento vocale o l'identificazione della scrittura manuale. L'apprendimento per rinforzo trova oggi applicazione nello sviluppo delle auto a guida autonoma, che proprio attraverso il *ML* imparano a riconoscere l'ambiente circostante e ad adattare il loro «comportamento» in base alle specifiche situazioni.

Il presente elaborato si focalizza sull'applicazione del *Machine Learning* supervisionato in ambito industriale, oggi largamente utilizzato per ottimizzare i processi di produzione e di vendita. Nella parte dedicata all'analisi empirica viene dettagliato come il *ML* possa trovare applicazione nella così definita *Quality 4.0*, pratica aziendale che si occupa di controllare la qualità del prodotto andando a rilevare i difetti di produzione. Altre applicazioni che rientrano nell'ambito industriale e di business sono la manutenzione predittiva, la previsione delle vendite e il marketing strategico e la previsione dei consumi energetici per la gestione dei costi ad essi correlati.

2.2.3 Gli algoritmi di classificazione

Enunciate le principali differenze tra i modelli di apprendimento e ripercorse le tappe salienti dello sviluppo del *Machine Learning*, riportiamo il focus della trattazione sulla categoria di apprendimento entro la quale rientra il modello utilizzato, ovvero quello supervisionato.

Lo scopo principe di un algoritmo applicato all'apprendimento supervisionato è apprendere la funzione che permette di ottenere da variabili di input X (x_1, x_2, …, x_n) la variabile di output y.

$$y = f(X)$$

L'obiettivo degli algoritmi che affrontano questa tipologia di problemi è dunque riuscire ad approssimare la funzione f in modo tale che, ogni qualvolta arrivi un nuovo dato in input, X riesca a predire la variabile di output y per il data set [49].

Ponendo attenzione alla tipologia di output si nota come gli algoritmi dell'apprendimento supervisionato siano raggruppabili in due categorie: classificazione e regressione.

La prima prevede che l'algoritmo venga addestrato per riuscire a mappare variabili di input con valori di output in qualità di etichette di classi discrete. Il processo di classificazione ha a che fare con problemi dove i dati possono essere divisi in un numero finito di classi, così come affermato da Han, Kamber e Pei: "Classification is a form of data analysis that extracts models describing important data classes. Such models, called classifiers, predict categorical (discrete, unordered) class labels" [50].

Il processo di classificazione può essere suddiviso in tre fasi:

- addestramento: è la fase che permette di creare un modello, che viene addestrato con i dati di input al fine di valutarne la bontà;
- stima dell'accuratezza: viene stimata l'accuratezza del modello utilizzando un insieme di test, dove per accuratezza si intende "the percentage of test set tuples that are correctly classified by the classifier. The associated class label of each test tuple is compared with the learned classifier's class prediction for that tuple" [51];
- utilizzo del modello: il modello elaborato viene utilizzato per classificare record con classe ignota.

A seconda del numero di classi esistono due tipologie di classificazione:

- classificazione binaria: le etichette sono due (Figura 2.4);

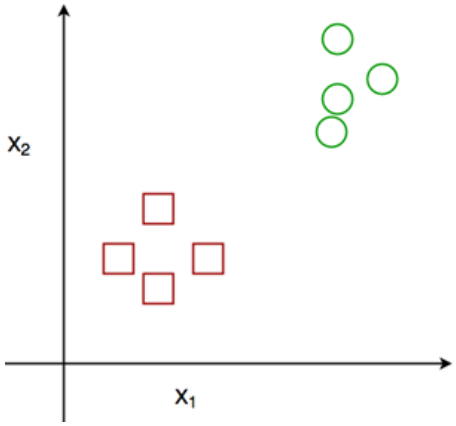

Figura 2.4 Classificazione binaria; *https://www.developersmaggioli.it/blog/apprendimento-supervisionato-classificazione-con-regressione-logistica/*, 28/01/2022.

- classificazione multi - classe: le etichette sono almeno tre (Figura 2.5).

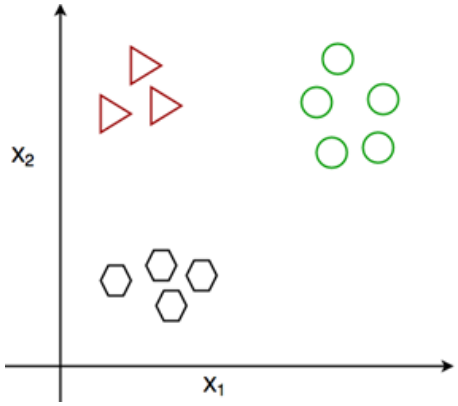

Figura 2.5 Classificazione multi – classe; *https://www.developersmaggioli.it/blog/apprendimento-supervisionato-classificazione-con-regressione-logistica/*, 28/01/2022.

Negli algoritmi di regressione, invece, l'obiettivo della funzione f è quello di ottenere da variabili di input X variabili di output y in qualità di numeri reali continui, dove y è un valore numerico (intero o decimale) e la maggior parte delle volte rappresenta una quantità o una dimensione.

Il problema ivi affrontato è riconducibile ad un processo di classificazione binaria. Più nello specifico, l'algoritmo utilizzato nella fase di sperimentazione è il *Random Forest*, cui si aggiunge un'analisi operata mediante la costruzione di una rete neurale artificiale di tipo *feed-forward*.

Nei paragrafi che seguono vengono analizzate le principali caratteristiche degli algoritmi ad oggi utilizzati per risolvere problemi di classificazione, con un particolare focus sul *Random Forest*.

2.2.3.1 Regressione Logistica

La regressione logistica è un modello di regressione non lineare che può essere usato quando la variabile dipendente y_i è di tipo dicotomico. Si tratta di uno degli algoritmi di *Machine Learning* più utilizzati per la classificazione binaria, ampiamente studiato anche in statistica [52].

Ciò che permette all'algoritmo di essere utilizzato per problemi di classificazione è l'utilizzo di una soglia, utile a classificare gli input con un valore di output piuttosto che un altro, a seconda del fatto che la sua probabilità sia al di sopra o al di sotto della soglia stessa [53].

Nella fase di addestramento l'algoritmo di regressione logistica prende in input n esempi da un insieme di training X. Ogni singolo esempio è composto da m attributi (i.e. variabili indipendenti) e dalla classe corretta di appartenenza.

Durante la fase di addestramento l'algoritmo, appartenente alla tipologia *eager learners*[18], elabora una distribuzione di pesi w, utile a classificare gli esempi con le classi corrette.

[18]In base alla modalità di apprendimento gli algoritmi si dividono in *eager Learners* e *lazy learners*. I primi cominciano ad apprendere sin dalla fase di addestramento. Una volta ottenuto il set di dati di addestramento, l'algoritmo inizia la costruzione del modello, per poi utilizzarlo all'arrivo dei dati di input. Gli algoritmi riconducibili a questa tipologia sono più lenti dei *lazy learners* in fase di apprendimento, ma più veloci nella

In seguito, viene calcolato un valore z, basato sulla combinazione degli input dell'insieme X (x_1, x_2, …, x_n) e dei rispettivi pesi W (w_1, w_2, …, w_n).

$$z = [x_0 \quad \cdots \quad x_n] \begin{bmatrix} w_0 \\ \vdots \\ w_n \end{bmatrix} + b$$

$$z = x_0 w_0 + x_1 w_1 + \cdots + x_n w_n + b$$

È in questo momento che entra in gioco la sigmoide (o funzione logistica), che simulando la funzione di soglia o *threshold*, è in grado di indicare se la variabile ricevuta in ingresso sia 1 o 0, calcolando la probabilità di appartenenza del campione alle classi del modello.
La funzione sigmoide è descritta dalla formula e dal grafico seguono.

$$\varphi(z) = \frac{1}{1 - e^{-z}}$$

fase di test [54]. La maggior parte degli algoritmi di apprendimento automatico appartengono a questa categoria.

I secondi nella fase di addestramento si limitano a memorizzare i dati, per iniziare la modellazione dell'insieme di addestramento solo all'arrivo delle istanze di test [55]. Sono particolarmente utili per set di dati di grandi dimensioni e in continua evoluzione. Se paragonati agli *eager learners*, questi algoritmi sono molto più veloci nella fase di addestramento ma più lenti in quella di predizione. Tra gli altri, rientrano in questa categoria gli algoritmi *k-nearest neighbors* e *Case-based reasoning*.

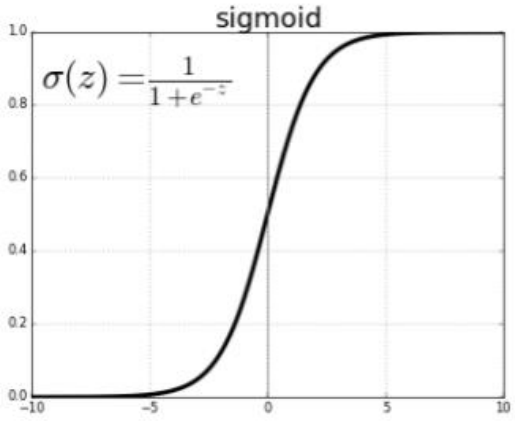

Figura 2.6 La funzione sigmoide; *https://www.developersmaggioli.it/blog/apprendimento-supervisionato-classificazione-con-regressione-logistica/*, 29/01/2022.

Nel contesto della regressione logistica la funzione sigmoide riceve in ingresso il valore z, calcolato attraverso la combinazione lineare $W^T X = w_0+w_1x_1+ ... + w_nx_n$ e lo trasforma in un numero reale compreso nell'intervallo [0,1].

$$f(z) = [0,1]$$

Il numero reale così ottenuto rappresenta la probabilità di appartenenza ad una classe piuttosto che un'altra.

Il risultato della funzione logistica è usato come funzione di attivazione dei nodi della rete neurale.

L'obiettivo della regressione logistica è modellare i pesi in modo tale da massimizzare la probabilità (o minimizzare l'errore) secondo la funzione statistica di verosimiglianza al modello:

$$L_{w,b} = \prod_{i=1,\ldots,N} \varphi\left(z_{(i)}\right)^{y_i} \left(1 - \varphi\left(z_{(i)}\right)\right)^{1-y_i}$$

Lo stesso risultato è ottenibile calcolando la log-probabilità invece della probabilità, come segue:

$$L_{w,b} = \sum_{i=1}^{N} y_i \log \varphi\left(z_{(i)}\right)(x) + (1 - y_i) \log(1 - \varphi\left(1 - z_{(i)}\right))$$

La sommatoria confronta tutte le risposte del modello con le risposte corrette calcolando la log-probabilità. Al termine dell'addestramento l'algoritmo produce un modello, utilizzabile per classificare qualsiasi altro esempio non compreso nel training set.

Tra le caratteristiche che rendono la regressione logistica più utile di altri algoritmi di apprendimento automatico sono la facilità di utilizzo, l'interpretabilità, la scalabilità. Tra le applicazioni odierne della regressione logistica si citano l'utilizzo per la previsione di malattie ed infarti in ambito medico, la previsione di gusti e comportamenti dei consumatori, la previsione di un fallimento di un dato processo o prodotto.

2.2.3.2 *Naïve-Bayes*

L'algoritmo noto come *Naïve-Bayes* affonda le radici negli studi dell'inglese Thomas Bayes sulla probabilità e la teoria delle decisioni [56]. Fa parte dei così detti classificatori

Bayesiani, ovvero metodi statistici di classificazione incrementali[19] in grado di predire la probabilità che una data istanza appartenga ad una certa classe.

Il teorema o formula di Bayes, di cui sotto si riporta la formula, lega la misura di probabilità condizionata di un evento, detta «a posteriori», alla misura di probabilità dello stesso evento, detta «a priori».

$$P(h|d) = \frac{P(d|h) * P(h)}{P(d)}$$

Nell'ambito del *Machine Learning* l'obiettivo è ricercare la miglior ipotesi h a fronte di un dato ricevuto d. In un problema di classificazione l'ipotesi h potrebbe essere tradotta come la classe di output da assegnare ad un dato d ricevuto in input.

Uno dei modi con cui è possibile calcolare l'ipotesi più probabile ricevuto un dato in ingresso è appunto il teorema di Bayes, dove:

- P(h|d) è la probabilità dell'ipotesi h dato d, detta probabilità condizionata o a posteriori di h rispetto a d;
- P(d|h) è la probabilità di d quando l'ipotesi h è verificata, detta probabilità condizionata di d rispetto ad h;
- P(h) è la probabilità che l'ipotesi h sia vera (a prescindere dai dati), detta probabilità a priori di h;
- P(d) è la probabilità che il dato d si verifichi (a prescindere dall'ipotesi), detta probabilità a priori di d.[20]

[19] Ogni istanza dell'insieme di addestramento modifica in maniera incrementale la probabilità che una ipotesi sia corretta.

L'algoritmo *Naïve-Bayes* parte da un'assunzione di base, ovvero l'indipendenza condizionale delle classi. Tale assunto enuncia che l'effetto di un attributo su una data classe deve essere indipendente dai valori degli altri attributi e serve per mitigare le difficoltà computazionali che potrebbero derivare da set di dati con un elevato numero di attributi; da qui la dicitura «naïve».

Partendo dal teorema di Bayes e dato l'assunto appena enunciato, viene di seguito spiegato il funzionamento dell'algoritmo in oggetto.

Sia X una istanza da classificare e C_1, C_2, ..., C_n le possibili classi, i classificatori Bayesiani calcolano $P(C_i \mid X)$ come:

$$P(C_i|X) = \frac{P(X|C_i) * P(C_i)}{P(X)}$$

dove l'obiettivo è quello di massimizzare $P(C_i \mid X)$. La classe C_i per la quale $P(C_i \mid X)$ viene massimizzata è chiamata *maximum posteriori hypothesis* [55].

Analizzando la formula sopra riportata si nota che:

- P(X) è una costante per tutte le classi e quindi non occorre calcolarla;
- $P(C_i)$ è facilmente calcolabile sull'insieme dei dati di addestramento contando la percentuale di istanze di classe C_i sul totale;
- P(X | Ci) sarebbe difficilmente calcolabile in caso di set di dati con un elevato numero di attributi e da qui l'assunto dei classificatori naïve dell'indipendenza condizionale delle classi.

[20] Per quel che riguarda le probabilità a priori, nel caso in cui la variabile sia di tipo discreto, possono essere stimate utilizzando la frequenza campionaria. Se la variabile è di tipo continuo, si parte dal presupposto che la sua distribuzione sia di tipo normale e si utilizza la funzione di densità per il calcolo delle probabilità.

Assumendo che X sia composta dagli attributi $A_1=a_1, a_2, \ldots, A_n = a_n$ otteniamo:

$$P\{C_j \mid A\} = P\{C_j \mid A_1, A_2, \ldots, A_n\} = \frac{P\{A_1, A_2, \ldots, A_n | C_j\} \times P\{C_j\}}{P\{A_1, A_2, \ldots, A_n\}}$$

Ed infine, assumendo verificata l'ipotesi di indipendenza condizionale delle classi, è possibile semplificare il calcolo della probabilità condizionata P{A1, A2, …, An|Cj} come segue:

$$P\{A_1, A_2, \ldots, A_n | C_j\} = \prod_n P\{A_n | C_j\}$$

In conclusione, l'algoritmo determinerà la classe di appartenenza come quella con la probabilità condizionata massima in base agli attributi dei vari elementi, ovvero C_i per cui $P(X|C_i)P(C_i)$ è il massimo.

L'algoritmo *Naïve-Bayes* viene molto apprezzato per la sua semplicità di applicazione e per l'efficienza e la precisione nella risoluzione dei problemi nel caso in cui l'ipotesi di indipendenza condizionale delle classi in input sia verificata. Tuttavia, proprio perché non prende in considerazione la possibile correlazione tra le variabili in input, rischia di fornire un'approssimazione ingenua del problema. A ciò si aggiunge il fatto che lo sviluppo dell'algoritmo necessita di conoscere tutti i dati del problema, in particolar modo tutte le probabilità a priori e condizionate, informazioni quasi sempre difficili e costose da ottenere.

L'algoritmo trova larga applicazione nel settore scientifico come strumento per l'effettuazione di diagnosi, mentre in quello informatico spesso viene utilizzato per classificare documenti testuali.

2.2.3.3 *Support-Vector Machines* (*SVMs*)

Le *Support-Vector Machines* (*SVMs*) sono dei modelli di apprendimento supervisionato associati ad algoritmi di apprendimento per la regressione e la classificazione.

Il primo *paper* ufficiale sulle macchine a vettori di supporto apparve nel 1992 ad opera di Vladimir Vapnik e dei colleghi Bernhard Boser e Isabelle Guyon [56].

In *Data Mining: Concepts and Techniques* Han, Kamber e Pei espongono il modello come una rappresentazione degli esempi come punti nello spazio (Figura 2.7), il cui obiettivo è quello di trovare la retta di separazione delle classi che massimizza il margine tra le classi stesse, identificando il margine come la distanza minima tra la retta e i punti delle due classi [57]. Questo modello permette di ottenere un classificatore binario non probabilistico.

Il calcolo della retta viene fatto utilizzando solo una porzione del data set, ovvero i valori di una classe che più si avvicinano alla retta di separazione, chiamati «vettori di supporto». I vettori di supporto rappresentano i dati dell'insieme di training classificabili con maggiore difficoltà.

La retta di separazione delle due classi è definita «Iperpiano» e in base al numero di *feature* può essere una retta (se le *feature* sono due) o un piano bidimensionale (se le *feature* sono tre), mentre il «margine» rappresenta la distanza tra l'Iperpiano e i vettori di supporto. Se nella regressione logistica i valori soglia sono rappresentati dall'intervallo [0,1], nell'*SVM* sono i valori [1,-1] a rappresentare il margine.

L'obiettivo dell'algoritmo è trovare un piano che massimizzi il margine, ovvero la distanza tra i punti delle due classi.

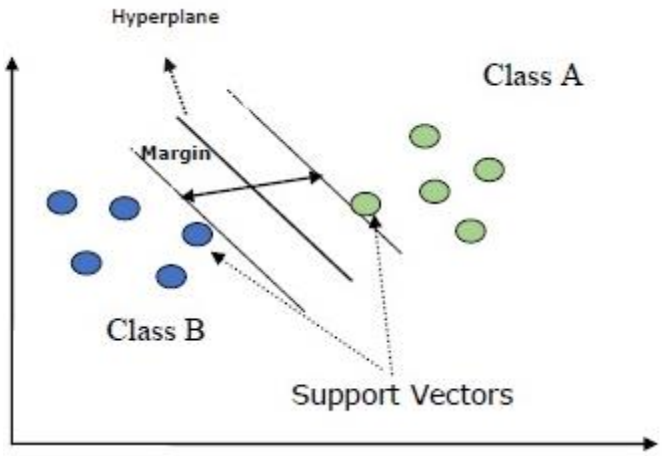

Figura 2.7 Funzionamento del modello *Support-Vector Machines*, con evidenziati i vettori di supporto; *https://www.tutorialspoint.com/scikit_learn/scikit_learn_support_vector_machines.htm*, 8/04/2022.

Durante la fase di addestramento l'algoritmo *SVM* procede al calcolo definitivo dell'Iperpiano, per poi passare, nella fase di test, a categorizzare i nuovi dati di input in base al fatto che si trovino nella parte di piano che identifica gli elementi di una classe piuttosto che l'altra.

Identifichiamo l'iperpiano con l'equazione:

$$W * X + b = 0$$

dove il vettore $W = \{w_1, w_2, \ldots, w_n\}$ è composto da n numeri reali, detti pesi, uno per ogni caratteristica $X = \{x_1, x_2, \ldots, x_n\}$ del set di dati e b è uno scalare.

Se si considerano due attributi in input A_1 e A_2 dove x_1 e x_2 sono rispettivamente i valori degli attributi A_1 e A_2 per X e si considera b come un altro peso w_0, è possibile riscrivere l'equazione come segue:

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

Di conseguenza, ogni punto che si trova al di sopra della retta dell'iperpiano soddisfa l'equazione:

$$w_0 + w_1 x_1 + w_2 x_2 > 0$$

Mentre ogni punto che si trova al di sotto della retta dell'iperpiano soddisfa l'equazione:

$$w_0 + w_1 x_1 + w_2 x_2 > 0$$

È possibile aggiustare i pesi così che gli iperpiani che definiscono i «lati» del margine possano essere scritti come:

$$H1: w_0 + w_1 x_1 + w_2 x_2 \geq 1 \quad \text{per } y_i = +1$$
$$H2: w_0 + w_1 x_1 + w_2 x_2 \leq -1 \quad \text{per } y_i = -1$$

Ne consegue che ogni tupla che si posizioni su o al di sopra di H_1 appartiene alla classe +1, viceversa ogni tupla che si posizioni su o al di sotto di H_2 appartiene alla classe -1.
Combinando le due disequazioni di sopra si ottiene che:

$$y_i(w_0 + w_1 x_1 + w_2 x_2) \geq 1, \forall_i$$

Ricordando che l'obiettivo dell'algoritmo è trovare l'iperpiano di separazione ottimale, ovvero quello con la distanza massima tra le tuple di addestramento più vicine, è possibile ottenere dalla formula sopra riportata la misura del margine «massimo».
La distanza tra l'iperpiano di separazione e qualunque punto su H_1 è $\frac{1}{||W||}$, dove ||w|| è la norma euclidea di W, ovvero $\sqrt{W * W^2}$. Per definizione, questa è uguale alla distanza tra qualunque punto su H_2 e l'iperpiano di separazione. Ne deriva che il margine massimo è uguale a $\frac{2}{||W||}$.

Nonostante appaia lento nella fase di addestramento, questo algoritmo viene largamente utilizzato per la sua accuratezza e la capacità di modellare casi complessi. E' inoltre meno incline di altri al problema dell'*overfitting*. Tra i campi di applicazione si ricordano il riconoscimento di oggetti, della scrittura e quello vocale.

2.2.3.4 *K-Nearest Neighbor* (*K-NN*)

Il *K-Nearest Neighbor* (*K-NN*) fu introdotto per la prima volta negli anni '50 [58]. E' un algoritmo di riconoscimento dei pattern basato sulla vicinanza dei dati, utilizzato per risolvere sia problemi di classificazione che regressione.
I classificatori *nearest-neighbor* si basano sull'apprendimento per analogia e confrontano istanze di test con istanze di addestramento simili ad esse. Ponendo che le istanze di addestramento possiedono n attributi e ognuna di queste rappresenta un punto in uno spazio n-dimensionale, l'obiettivo del classificatore *K-Nearest Neighbor* è di identificare, all'interno dello spazio n-dimensionale, le K istanze di addestramento più vicine alle istanze di test.
La «vicinanza» è solitamente definita in termini di distanza euclidea tra i punti dello spazio n-dimensionale. Dunque, se si identificano due istanze $X_1 = (x_{11}, x_{12}, \ldots, x_{1n})$ e $X_2 = (x_{21}, x_{22}, \ldots, x_{2n})$, il calcolo della distanza sarà:

$$dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}$$

In altre parole, i *K-Nearest Neighbors* di un campione X sono i campioni che hanno le K minori distanze da X [59].

L'algoritmo *K-NN* appartiene alla categoria dei *lazy learners* (Figura 2.8). Durante la fase di addestramento si limita a storicizzare gli elementi del training set, mentre in quella di classificazione calcola la distanza tra gli elementi del training set ed il nuovo elemento. Una volta calcolate le distanze, l'algoritmo seleziona i K elementi più vicini al nuovo elemento e calcola la probabilità del dato di test di appartenere a ognuna delle classi a cui appartengono gli elementi selezionati. Il risultato finale sarà quindi la classe che possiede la probabilità maggiore, ovvero la classe più frequente nei K esempi selezionati.

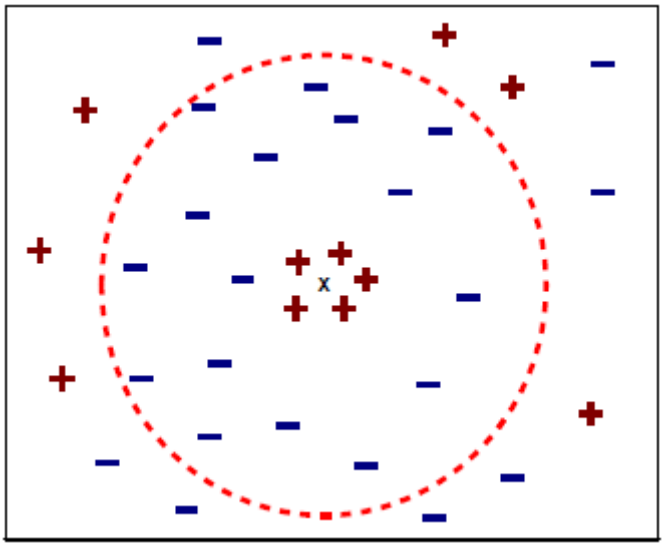

Figura 2.8 Esempio di *K-Nearest Neighbors*;
http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/lguo/knn.html, 10/04/2022.

Molto importante è la corretta determinazione del parametro K in fase di configurazione dell'algoritmo: se troppo piccolo può essere sensibile a dati rumorosi, se troppo grande può rendere costoso e complesso il processo computazionale.

Si tratta di un algoritmo di facile implementazione, ma che può causare problemi non banali in termini di costi e poca accuratezza in caso di data set di grandi dimensioni. Trova applicazione nel settore medico (e.g. misurazione del diabete, prevenzione del cancro al seno) e nei software di filtraggio dei contenuti utilizzati da siti come Amazon e Netflix.

2.2.3.5 *Decision Tree*

Tra gli anni '70 e gli anni '80 l'informatico e studioso di *Machine Learning* John Ross Quinlan sviluppò un algoritmo ad albero di decisione noto come ID3[21] e qualche tempo più tardi presentò il suo successore C4.5; diventato oggi un benchmark con cui vengono spesso confrontati i più recenti algoritmi di apprendimento supervisionato [60]. Negli stessi anni un gruppo di statistici elaborò e pubblicò nel libro *Classification and Regression Trees* (*CART*) il frutto dei loro studi sugli alberi decisionali binari [61].

L'Albero Decisionale è uno degli algoritmi di tipo supervisionato più comuni ed usati in statistica, *Data Mining* e *Machine Learning*. È un modello predittivo adatto ad affrontare sia problemi di classificazione che di regressione (Figura 2.9).

I più noti algoritmi per l'induzione di Alberi di Decisione adottano un approccio *greedy*[22] (i.e. non *backtracking*) e sono costruiti in modo ricorsivo secondo un modello *top-down*,

[21] *Iterative Dichotomiser*.

[22] Traducibile con il termine «avido». Si definisce così perché ad ogni step della costruzione dell'albero la suddivisione è fatta cercando lo *split* migliore dello step e non considerando gli step futuri.

usando una politica nota come *divide et impera*[23]. L'approccio *top-down* inizia con un set di addestramento di tuple e le relative etichette di classe, che, in modo ricorsivo, viene suddiviso in sottoinsiemi più piccoli durante la costruzione dell'albero [62].

Le componenti individuabili in un Albero di Decisione sono:

- i nodi, luoghi dove i dati vengono divisi in base ad una funzione condizionale (test sugli attributi);
- i rami, collegamenti tra i nodi genitori e i nodi figli (risultati del test al nodo padre);
- le foglie, classi o distribuzioni di probabilità per le classi (risultati intermedi o finali).

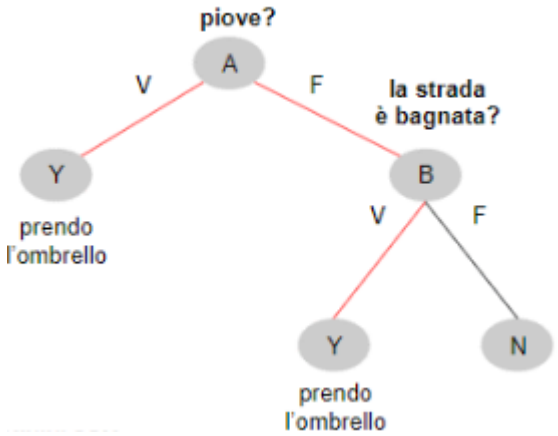

Figura 2.9 Esempio di Albero di Decisone; *https://www.andreaminini.com/ai/machine-learning/alberi-di-decisione*, 10/04/2022.

Il processo di classificazione di un'istanza può essere rappresentato dal seguente flusso:

- si parte dal nodo radice, ovvero dal nodo genitore situato più in alto nella struttura (alla radice sono assegnate tutte le istanze di addestramento);
- si seleziona un attributo;

[23] Dividere il problema in sotto problemi più piccoli.

- si creano tanti nodi (i.e. figli della radice) quanti sono i possibili valori dell'attributo scelto. Ogni istanza di addestramento sarà assegnata al figlio appropriato;
- si procede ricorsivamente usando come radici i nuovi nodi;
- se si raggiunge un nodo foglia viene restituita l'etichetta associata alla foglia, che identifica la decisione finale. In caso contrario, si riparte con la selezione dell'attributo tenendo in considerazione il nodo corrente.

Il processo si blocca quando:

- tutte le istanze di un nodo fanno parte della stessa classe;
- non ci sono più attributi sui quali dividere l'albero.

Si noti che, come alternativa all'assegnazione di una classe ad un nodo, è anche possibile assegnare al nodo la distribuzione di probabilità dell'attributo classe (relativamente alle istanze assegnate al nodo).

Compito dell'algoritmo è la selezione della caratteristica che permette una classificazione migliore degli esempi, ovvero dividendo i campioni di classi eterogenee in sottoinsiemi dove le variabili di output sono omogenee e dove la varianza è ridotta al minimo. Si definisce una metrica per misurare l'impurità.

Sia X un sottoinsieme di un particolare insieme di addestramento con m possibili classi. Per ogni x_i è possibile definire la distribuzione di probabilità $p(x_i) = p_i$, dove X è un data set formato da m classi e p_i è la frequenza relativa della classe i all'interno dell'insieme X.

Data la definizione di X, negli alberi di decisione sono largamente usate le seguenti metriche:

- entropia, dove I_H di X è

$$I_H(X) = \sum_{i=1}^{m} p_i \log_2 p_i$$

- indice di impurità di Gini[24], dove I_G di X è

$$I_G(X) = 1 - \sum_{i=1}^{m} p_i^2$$

- errore di classificazione (dalla teoria bayesiana), dove I_E di X è

$$I_E(X) = 1 - \max p_i$$

La bontà della condizione eseguita è ricavabile dal confronto tra l'impurità misurata da un nodo padre e il nodo figlio. Una maggiore differenza identifica una migliore scelta della condizione. Sia $I_{(.)}$ la metrica che misura l'impurità del nodo, il guadagno Δ, indicatore della qualità della divisione, può essere definito come:

$$\Delta(X) = I(\varepsilon) - \sum_{i=1}^{k} \frac{N(\varepsilon_i)}{N(\varepsilon)} I(\varepsilon_i)$$

Dove $N(\varepsilon)$ e $N(\varepsilon_i)$ rappresentano rispettivamente il numero di campioni nel nodo padre e nel nodo del figlio i-esimo.

Basandosi sull'entropia come metrica di riferimento, allora il guadagno Δ è noto come *Information Gain*[25] [63]. Poiché il metodo utilizza attributi con valori diversi e solito condurre il modello verso il problema di *overfitting*.

[24] Usato dall'algoritmo CART.

[25] Usato dall'algoritmo ID3.

Gli alberi di decisione cercano di selezionare la condizione di test tale da massimizzare il guadagno Δ, equivalente alla minimizzazione della somma pesata delle impurità dei nodi figli.

$$h = \underset{hj}{\operatorname{argmin}} \sum_{i=1}^{k} N(\varepsilon_i) I(\varepsilon_i)$$

Dunque, la miglior domanda $h_j(x)$, ovvero quella che massimizza la scelta dell'attributo, è quella che minimizza tale quantità.

La velocità di implementazione e la semplicità di interpretazione sono tra le caratteristiche che senza dubbio ne hanno facilitato la diffusione, oltre che la elevata capacità di adattamento ai dati di addestramento. Quest'ultima caratteristica rende però gli alberi di decisione piuttosto soggetti al problema di *overfitting*.

È buona prassi applicare agli alberi un algoritmo di raffinamento, detto *pruning*, per ridurre il problema di *overfitting* ed eliminare i rami che rappresentano rumori nei dati o *outliers*.

Gli approcci di *pruning* utilizzati sono il *pre-pruning* e il *post-pruning*. Il *pre-pruning* si limita a fermare la creazione dell'albero sotto determinate condizioni[26] per evitare una eccessiva specializzazione. Non sempre risulta facile scegliere il valore soglia e si rischia di imbattersi in problemi di *early stopping*[27]. La tecnica di *post-pruning* agisce invece su di un albero già creato, eliminando quei rami che non soddisfano alcune condizioni su un validation set precedentemente selezionato. Si tratta del metodo più utilizzato, che prevede tecniche di *subtree-replacement* e *subtree-raising*.

[26] Si divide un nodo solo se esiste un attributo che ha un livello di associazione attributo-classe che supera una soglia prefissata.

[27] Interruzione prematura dell'albero.

Gli alberi di decisione trovano larga diffusione in campo medico per la predisposizione di diagnosi, nel marketing per l'analisi delle campagne promozionali e nel settore bancario per il rilevamento di frodi.

2.2.3.6 *Random Forest Classifier*

Il *Random Forest* è un metodo di apprendimento *ensemble* che opera costruendo una moltitudine di alberi decisionali nella fase di addestramento. Nella statistica e nell'apprendimento automatico i metodi *ensemble* sono noti per servirsi di più algoritmi di apprendimento, cosa che permette di ottenere prestazioni predittive migliori [64]. È questo il classificatore che si è deciso di utilizzare nella fase sperimentale e di seguito appariranno chiare le ragioni (Figura 2.10).

Il *Random Forest* è uno degli algoritmi di tipo supervisionato più usati e versatili del *Machine Learning*, utile a risolvere sia problemi di classificazione che di regressione, la cui prima formulazione risale al 1998 [65]. Fu la ricercatrice americana Ho Tin-Kam a sviluppare il primo algoritmo per foreste casuali, utilizzando il metodo del sottospazio casuale per ridurre la correlazione tra stimatori. Leo Breiman [66] e Adele Cutler ne estesero la formulazione e lo registrarono nel 2006 come marchio.[28]

[28] Numero di registrazione del marchio statunitense 3185828, registrato il 19/12/2006; *https://trademarks.justia.com/786/42/random-78642027.html*, 20/05/2022.

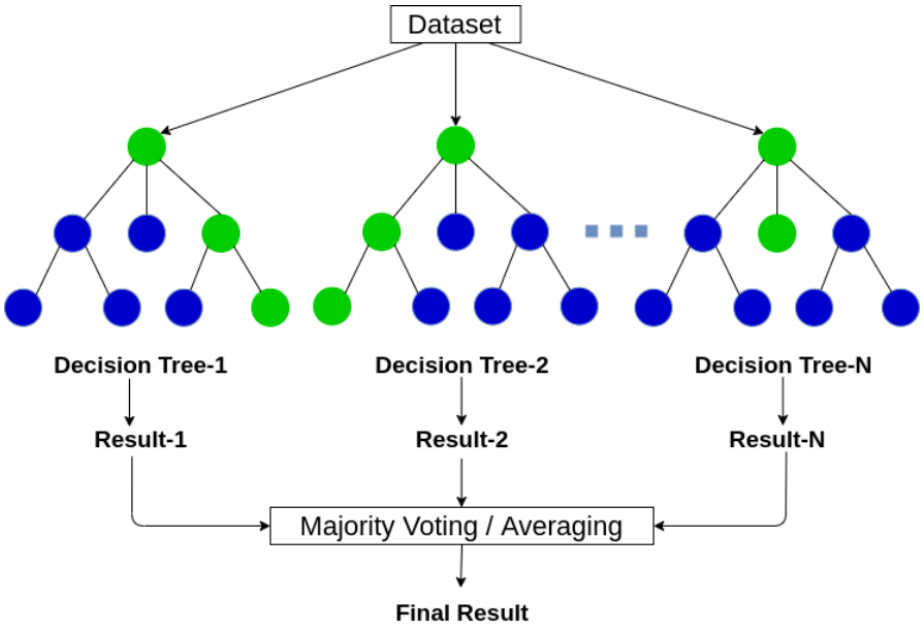

Figura 2.10 Esempio di *Random Forest*; *https://ai-pool.com/a/s/random-forests-understanding*, 20/03/2022.

Alla base della creazione delle foreste casuali vi è il metodo di *bagging* o *bootstrap aggregating*. Si tratta di un metodo che viene solitamente utilizzato per ridurre la varianza di una funzione di previsione stimata e che sembra funzionare bene con procedure ad alta varianza e bassa distorsione. Gli alberi decisionali, dati la capacità di catturare strutture di interazione complesse nei dati e un grado di distorsione relativamente basso (se fatti crescere con sufficiente profondità), paiono proprio i candidati ideali per l'applicazione di questo metodo.

La procedura generale per generare k alberi decisionali, dato un insieme D di addestramento di tuple, è la seguente:

- per ogni iterazione i (i = 1, 2, …,k), un training set D_i di d tuple viene campionato con la sostituzione di D. Ciò significa che ogni D_i è un campione *bootstrap* di D e ne consegue che alcune tuple possano verificarsi più di una volta in D_i , mentre altre possano esserne escluse;

- sia F il numero di attributi da utilizzare per determinare la divisione in ciascun nodo, dove F è molto più piccolo del numero di attributi disponibili;
- per costruire un classificatore dell'albero decisionale M_i si selezioni casualmente, in ogni nodo, gli attributi F come candidati per la divisione nel nodo.

La crescita degli alberi viene alimentata tramite la metodologia CART, secondo la quale gli alberi vengono portati alla dimensione massima e mai potati. Le foreste casuali formate in questo modo, con una selezione di input casuale, sono chiamate *Forest-RI* [67].

Riassumendo quanto fin' ora spiegato, si potrebbe dire che il *Random Forest* è un classificatore *ensemble* che, attraverso il metodo di *bagging*, costruisce una grande raccolta di alberi de-correlati e quindi ne calcola la media.

Durante la fase di training ognuno degli alberi decisionali che compongono il modello viene addestrato utilizzando un sottoinsieme casuale dei dati presenti nel training set. Quindi, ad ogni nodo, viene scelto l'attributo migliore tra un set di attributi selezionati casualmente. La casualità è dunque un fattore centrale quando si ha a che fare con il *Random Forest*, che permette di accrescere la diversità dei vari classificatori e diminuirne la correlazione.

Una volta completata la fase di training, durante la fase di prediction il modello riceve un nuovo dato in ingresso, lo fornisce come input ai vari alberi di decisione che lo compongono ed aspetta che essi abbiano completato il processo di classificazione dell'elemento. Il risultato finale del processo sarà dunque la classe dal maggior numero di alberi.

Che il *Random Forest Classifier* sia uno dei modelli di *Machine Learning* ad oggi più utilizzati e potenti lo confermano gli innumerevoli campi in cui trova applicazione. Tra le caratteristiche che lo rendono così famoso ci sono gli elevati livelli di performance (paragonabili a quelli di un *Decision Tree*), la bassa probabilità di *overfitting* grazie all'elemento della casualità (differenza fondamentale con il *Decision Tree*), l'utilizzo di

algoritmi semplici e la facile interpretazione. Molti lo sconsigliano però nel caso in cui si disponga di poche risorse o di data set piccoli.

Grazie alle sue caratteristiche l'algoritmo trova oggi applicazione in più settori. Da quello bancario, per valutare l'affidabilità creditizia dei potenziali clienti, a quello medico, per effettuare diagnosi su pazienti o definire il dosaggio di medicinali sulla base della storia clinica. Dal settore finanziario, dove gli analisti utilizzano il *Random Forest* per valutare i potenziali mercati azionari, all'*e-commerce*, dove i venditori analizzano gli acquisti passati per studiare le preferenze di vecchi e nuovi clienti.

Per ulteriori dettagli sul classificatore *Random Forest*, di natura più pratica ed applicativa, si rimanda ai capitoli successivi, dedicati all'esposizione dell'analisi empirica effettuata sul processo di saldatura delle macchine saldatrici Schunk Minic III utilizzate in SEWS-CABIND nella produzione dei cablaggi.

2.3 Elementi di *Deep Learning*

Si è dunque giunti all'ultimo tassello della struttura: il *Deep Learning*.

Il *Deep Learning* o Apprendimento Profondo è rappresentato da modelli di apprendimento ispirati alla struttura ed al funzionamento del cervello biologico. Se il *Machine Learning* può essere definito come il metodo che «allena» l'IA, il *Deep Learning* è quello che permette di emulare la mente umana. L'Apprendimento Profondo è un'area del *Machine Learning* che si basa su un apprendimento automatico dai dati di input a più livelli. Ogni livello più profondo prende come input i dati di output del livello precedente, estraendo sempre più informazioni con l'aumentare della profondità [68].

Un aspetto fondamentale che distingue il *Deep Learning* dalle tradizionali tecniche di *Machine Learning* è il livello di influenza che l'intervento umano ha sul risultato finale. Nel *Machine Learning* l'accuratezza e la precisione delle predizioni sono fortemente influenzate

dall'abilità di colui che realizza il modulo di estrazione, mentre *nel Deep Learning* "i livelli di rappresentazione delle *feature* non sono progettati o realizzati direttamente dall'uomo, ma sono appresi automaticamente dai dati" [69].

Il paragrafo sul *Deep Learning* vuole fungere da guida generale sull'argomento, spaziando da minime nozioni sulla storia e i campi di applicazione, ad un approfondimento sulle Reti Neurali e le loro pratiche di addestramento.

2.3.1 Evoluzione ed applicazioni del *Deep Learning*

Il *Deep Learning* costituisce una sotto-area del *Machine Learning* e fa uso di *Deep Neural Network* o Reti Neurali Profonde e di algoritmi per il pre - processamento dei dati. L'Apprendimento Profondo trae ispirazione dalle neuroscienze, ma a differenza del cervello biologico, dove qualsiasi neurone può connettersi a qualsiasi altro neurone sotto alcuni vincoli fisici, le Reti Neurali Artificiali o *Artificial Neural Networks* (ANN) hanno un numero finito di strati e connessioni e una direzione prestabilita della propagazione dell'informazione.

Per molti anni le ANN sono state ignorate sia dai ricercatori che dall'industria a causa degli elevati costi connessi all'utilizzo e all'implementazione. Non a caso i primi sviluppi significativi del *Deep Learning* hanno coinciso con i primi grossi investimenti dei colossi del settore dell'informatica.

Il 2012 è senza dubbio una data degna di nota nella storia del *Deep Learning*, anno in cui al contest *ImageNet* furono presentati i risultati straordinari raggiunti dal professor Geoffrey Hinton dell'Università di Toronto[29]. Un pool di studiosi presentò questa nuova tecnologia

[29] Il gruppo di ricerca guidato da Geoffrey Hinton fu in grado di strutturare gli algoritmi per le ANN su architetture parallele. Il principale risultato fu un notevole incremento del numero di strati, neuroni e

che aveva permesso alla macchina di riconoscere uomini e animali tramite il confronto con milioni di altre immagini, senza alcuna necessità di un intervento umano [70]. Passarono tre anni e nello stesso contest fu Microsoft a farla da padrona, presentando un sistema di *Deep Learning* composto da 152 livelli di astrazione, con un incremento di circa cinque volte superiore al numero di livelli su cui si basava la ricerca precedente.

Il requisito principe per l'addestramento di un modello di *Deep Learning* è quello di avere a disposizione training set molto grandi e questo rende di facile comprensione quanto il recente avvento dei *Big Data* abbia dato un impulso senza precedenti alla sua diffusione. Le ragioni sottostanti la popolarità del *Deep Learning* sono infatti legate in buona sostanza alla diffusione dei *Big Data* e dei sistemi di calcolo parallelo basati su *GPU* (*Graphics Processing Unit*). Avendo come base una quantità massiccia di dati, attraverso l'algoritmo di addestramento, la rete «apprende» come raggiungere gli obiettivi.

Un grande limite connesso agli algoritmi di *Deep Learning* è il loro essere soggetti al così detto *bias*, che li rende sensibili se sottoposti ad etichette create erroneamente. Esempio eclatante fu il flop iniziale di FaceNet,[30] che etichettò alcune facce di individui africani come gorilla [71].

Numerose sono le applicazioni che il *Deep Learning* vanta nella vita di tutti i giorni. Dall'aiuto nelle aziende a migliorare la *customer experience*, ai servizi online di traduzioni automatiche di testi, capaci, tra l'altro, di migliorare continuamente in base ai feedback ricevuti dagli utenti. Dalle auto a guida autonoma addestrate a riconoscere segnali stradali e in grado di azionare automaticamente il meccanismo di frenata, alla *Computer Vision*. La *Computer Vision* è una tecnica dove l'applicazione del *Deep Learning* ha portato a risultati

parametri del modello in generale (anche oltre i 10 milioni di parametri), permettendo alle macchine di computare una quantità massiccia di dati addestrandosi direttamente su di essi.

[30]Sistema di riconoscimento facciale creato da Google.

straordinari in campo medico, fungendo da aiuto nell'analisi di problematiche dei pazienti grazie alla realizzazione di modelli tridimensionali a partire da semplici radiografie. Ed infine numerose tecnologie in grado di analizzare immagini, comprenderne il contenuto e, ad esempio, attribuire didascalie o musiche in linea con il contenuto stesso.

2.3.2 Le reti neurali artificiali

Come ampiamente anticipato, il *Deep Learning* può essere identificato come una branca del *Machine Learning* basata sull'utilizzo di reti neurali profonde, anche conosciute come *Neural Network Multilayers*. Si tratta di sistemi adattivi, in grado di modificare la propria struttura in termini di nodi ed interconnessioni sulla base sia di input interni che esterni.

Le reti neurali artificiali cercano di simulare in toto il funzionamento delle reti neurali biologiche e delle parti che le compongono. Il miglior modo per capire il funzionamento di una rete neurale è capire come funziona un neurone biologico.

Le parti di cui si costituisce un neurone biologico sono [72] (Figura 2.11):

- somi neuronali, la parte centrale del neurone. Il loro compito è quello di ricevere ed elaborare informazioni e, nel caso in cui una certa soglia di attivazione venga superata, generare a loro volta degli impulsi in grado di propagarsi nella rete;
- assoni (o neuriti), che identificano la via di comunicazione in uscita da un neurone. Ogni cellula nervosa ne possiede di norma soltanto uno;
- sinapsi, i siti funzionali ad alta specializzazione nei quali avviene il passaggio delle informazioni fra neuroni. Ognuno di questi ne possiede migliaia. Ciascuna sinapsi termina con un bottone sinaptico;
- dendriti, i filamenti che fuoriescono dal soma. Sono la principale via di comunicazione in ingresso. Il neurone è capace di ricevere segnali attraverso i

propri dendriti, li elabora nel soma e successivamente trasmette il segnale, tramite l'assone, al neurone successivo.

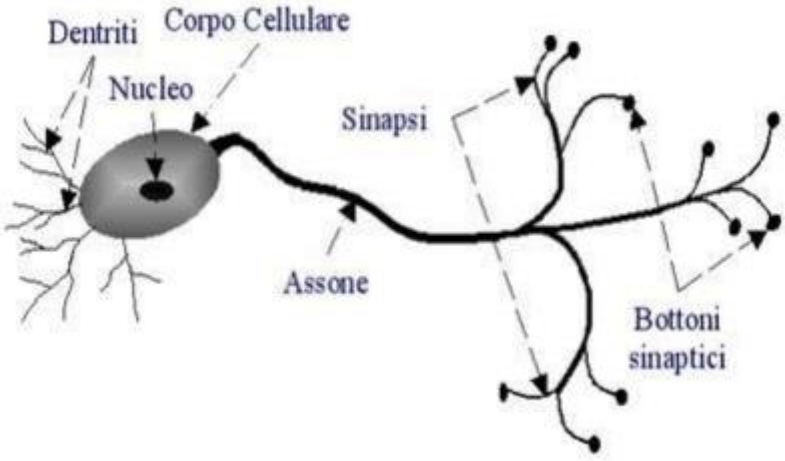

Figura 2.11 Rappresentazione schematica di un neurone biologico in Roberto Prevete, *Il neurone biologico*; *http://www.federica.unina.it/smfn/reti-neurali-e-machinelearning/neurone-biologico/*,13/04/2022.

Le reti neurali si basano sulla simulazione di neuroni artificiali opportunamente collegati, i quali ricevono in ingresso degli stimoli, elaborandoli di conseguenza. Il primo modello di neurone artificiale venne introdotto da Warren McCulloch e Walter Pitts nel 1943 [73] (Figura 2.12).

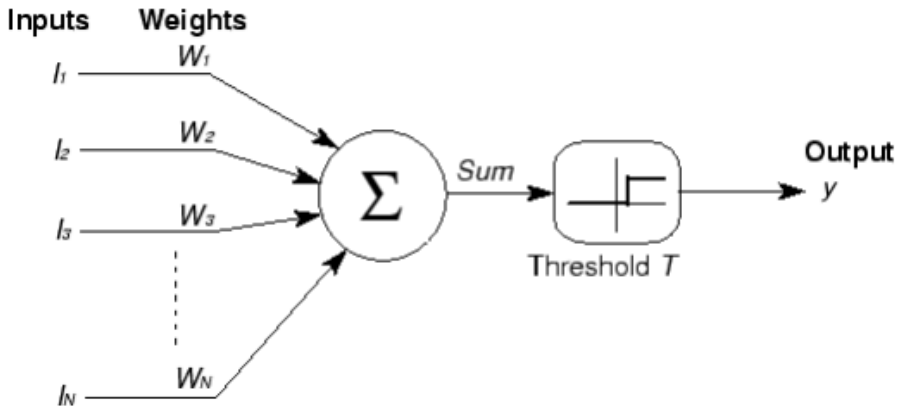

Figura 2.12 Modello di McCulloch e Pitts, Warren S. McCulloch e Walter S. Pitts, *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics, 1943; *https://link.springer.com/article/10.1007/BF02478259*, 13/04/2022.

Nei casi più semplici l'elaborazione prevedeva che i singoli ingressi venissero moltiplicati per un opportuno valore, detto «peso» e il risultato delle moltiplicazioni venisse poi sommato. Se la somma avesse superato una certa soglia, il neurone avrebbe attivato la propria uscita. Si noti che la distribuzione dei valori dei pesi variava in base all'importanza dell'ingresso: un ingresso importante aveva un peso elevato, a differenza di uno meno importante che aveva un valore inferiore. Il modello di McCulloch e Pitts mostrò presto i suoi limiti dati dalla necessità di impostare manualmente pesi e connessioni.

Alla fine degli anni Cinquanta Frank Rosenblatt introdusse una rete composta di unità perfezionate dal modello McCulloch-Pitts: il Percettrone [74]. Il Percettrone nacque dall'unione del modello precedente con la regola di Hebb per l'adattamento dei pesi [75] e l'aggiunta di un ulteriore valore di input a rappresentare il *bias*.[31] La novità principale del modello di Rosenblatt fu la sua capacità di modificare i propri pesi, adattandoli al problema dato senza necessità di intervento manuale.

[31] Da considerarsi come una soglia che influenza ampiamente l'output dell'unità.

La stessa sorte toccò anche al modello di Rosenblatt, criticato nel 1969 da Marvin Minsky e Symour A. Papert per l'incapacità di risolvere problemi non caratterizzati da separabilità lineare delle soluzioni [76]. Seguì un periodo di stallo, noto come «Inverno dell'IA», che si poté considerare concluso con l'invenzione delle reti neurali a più livelli e del famoso Percettrone multistrato, nato dall'interconnessione di input e output di più neuroni artificiali.

Il Percettrone multistrato (MLP), modello utilizzato per elaborare parte dell'analisi empirica di questa tesi, è una vera e propria rete neurale artificiale composta da più percettroni. Esso si compone di un livello di input che riceve il segnale, uno di output, che esegue una previsione o prende una decisione per quanto concerne l'input ed un numero arbitrario di strati «nascosti», il vero motore computazionale della rete (Figura 2.13). Ciascun neurone di un livello è connesso a tutti i neuroni del livello precedente, per tale motivo una rete di questo tipo è anche detta *Fully Connected*.

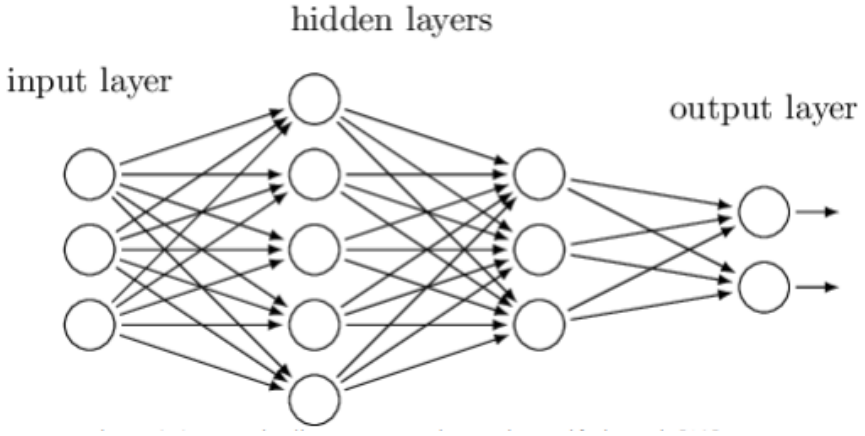

Figura 2.13 Esempio di MLP con due strati nascosti; *https://jsalatas.ictpro.gr/implementation-of-elman-recurrent-neural-network-in-weka/*, 13/04/2022.

In una rete neurale (semplice), alla cui categoria appartiene il MPL, è dunque possibile individuare tre componenti:

- lo strato degli ingressi I (*input layer*) che ha il compito di ricevere ed elaborare i segnali in ingresso adattandoli alle richieste dei neuroni della rete;
- lo strato nascosto H (*hidden layer*) che ha in carica il processo di elaborazione vero e proprio. Gli strati nascosti possono essere più di uno, maggiore è il numero, più intelligente sarà la rete neurale;
- lo strato di uscita O (*output layer*) che raccoglie i risultati dell'elaborazione forniti dallo strato nascosto e li adatta alle richieste del successivo livello-blocco della rete neurale.

Ogni strato della rete neurale contiene uno o più neuroni artificiali, che a loro volta possiedono una o più vie di comunicazioni di ingresso, i cosiddetti dendriti. Il modello di neurone artificiale di base implica una serie di parametri adattivi, denominati pesi, che hanno la funzione di moltiplicatori degli input del neurone. La somma dei pesi moltiplicata per gli input viene chiamata combinazione lineare degli input.

Una volta calcolata la combinazione lineare, il neurone prende la combinazione lineare e la sottopone ad una funzione di attivazione, la quale, insieme alla combinazione lineare, determina l'output del neurone.

Le funzioni di attivazione di cui ci si è serviti per l'analisi empirica sono:

- La funzione logistica o sigmoide

$$g(x) = \frac{1}{1 + e^{-x}}$$

La funzione sigmoidea può essere vista come una versione smussata della funzione di attivazione del percettrone. Anche se è una delle funzioni più usate oggi, non è

esente da problemi. Primo fra tutti vi è quello della scomparsa del gradiente, che si verifica quando il gradiente assume un valore talmente basso che la rete rifiuta di apprendere ulteriormente.

Questa funzione di attivazione è stata utilizzata nel lavoro empirico per gli strati di *input* e quelli nascosti.

Si noti che La funzione di tangente iperbolica (tanh) è una buona alternativa alla sigmoide:

$$g(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

La sua natura è sempre non lineare, ma il suo gradiente è molto più resistente della sigmoide e decidere tra le due dipenderà dalle richieste di robustezza del gradiente stesso. Anche'essa, tuttavia, non è esente dal problema della scomparsa del gradiente;

- la funzione SoftMax

$$g(x)i = \frac{e^{zi}}{\sum_{j=1}^{K} e^{zj}}, \text{ per } i = 1,\ldots, K \text{ e } z = (zi, \ldots, zk) \epsilon Rk$$

Usata per lo più nell'*output layer* e nei problemi di classificazione [77], questa funzione accetta in input un vettore di K numeri reali normalizzandolo in una distribuzione di probabilità composta da K valori di probabilità sugli esponenziali dei valori in input. Ciò garantisce che, dopo averla applicata, i valori saranno sempre compresi nell'intervallo (0,1).

Questa funzione di attivazione è stata utilizzata nel lavoro empirico per gli strati di output.

Ne consegue che la fase di apprendimento all'interno della rete avviene quando i pesi sono adattati in modo da far produrre alla rete gli output corretti. Non di rado le reti neurali sono molto estese e le più grandi contengono centinaia di miliardi di pesi. Ottimizzarli tutti può essere un compito difficile e che richiede altissime capacità di calcolo.

La struttura di una rete neurale può variare molto in termini di complessità. Le reti utilizzate nella parte di analisi empirica sono le cosiddette *Feed-Forward Neural Network* (FFNN), reti in cui le connessioni non formano cicli, ma avanzano in un'unica direzione: dall'ingresso verso l'uscita della rete [78]. Si ricorda che le MLP rientrano appunto nella categoria delle reti neurali *feed-forward* multistrato (Figura 2.14).

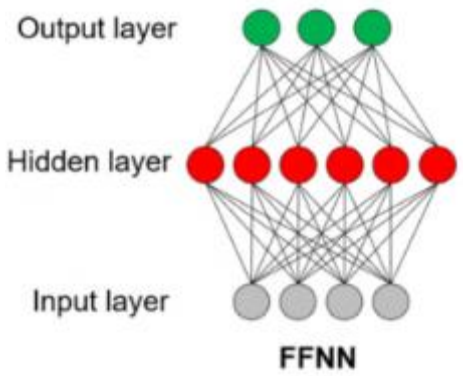

Figura 2.14 Esempio di Rete Neurale *feed-forward* con uno strato nascosto e uno di uscita; *https://www.developersmaggioli.it/blog/le-reti-neurali-ricorrenti/*, 13/04/2022.

Nelle reti *feed-forward* ogni neurone di un livello riceve input solo da neuroni di livelli precedenti e può propagare solamente verso neuroni di livelli successivi. Non sono dunque possibili auto-collegamenti o connessioni con neuroni dello stesso livello.

La funzionalità principale del neurone è quindi quella di propagare il segnale attraverso la rete, con un flusso di informazione dagli ingressi verso le uscite. In un sistema del genere

l'uscita attuale dipende solo ed esclusivamente dall'ingresso attuale; quindi, in sostanza la rete non ha memoria di ingressi avvenuti in tempi precedenti.
Un'altra tipologia di rete neurale è la *Recurrent Neural Network* (RNN), dove sono previste connessioni di feedback verso o neuroni dello stesso livello, o neuroni di livelli precedenti. Una rete neurale ricorrente è una rete in cui esistono dei cicli: i valori di uscita di un layer di livello superiore vengono utilizzati come ingresso per un *layer* di livello inferiore. Le reti appartenenti a questa categoria hanno pertanto una gestione del flusso di informazioni e dell'addestramento molto complicato e che si dipana su più istanti temporali (*unfolding in time*).

2.4 Addestramento di una rete neurale

Per poter utilizzare una rete neurale come classificatore occorre passare per la fase di addestramento della rete stessa. Questa fase consiste nel determinare, avendo a disposizione un training set, i pesi delle connessioni in input in modo tale che sempre di più l'output della rete, a fronte di un certo input, si avvicini a quello desiderato. Per ottenere ciò serve una funzione, detta di costo, che quantifichi la discrepanza tra l'output nell'esempio e quello realmente prodotto dalla rete.
La rete neurale, sia durante la fase di addestramento che in quella di test, ad ogni input che riceve, associa un output y_pred attraverso una funzione di *mapping* f.

$$\mathrm{y_{pred}} = \mathrm{f(x, W)}$$

Dove x rappresenta l'input ricevuto dalla rete, mentre W l'insieme dei pesi utilizzati all'interno della rete.

Per ottenere il valore ottimale occorre modificare i valori di W con lo scopo di minimizzare la cosiddetta funzione di perdita. Questo processo può essere descritto dalla seguente formula:

$$\min\{ J(W) \} = \min\left\{\frac{1}{N}\sum_{i=1}^{N}\left(L_i(f(x_i, W), y_{true})\right) + \lambda R(W)\right\}$$

dove:

- L rappresenta la funzione di perdita calcolata tra la predizione della rete (y_pred) e il relativo output target (y_true);
- R(W) è il termine di regolarizzazione, con il relativo parametro di regolarizzazione λ;
- J(W) esprime la funzione di costo complessiva calcolata sugli N campioni di batch.

La minimizzazione è un processo iterativo in cui durante la prima iterazione vengono inizializzati i pesi con valori casuali ed i dati vengono fatti passare in *forward* lungo gli strati della rete, ottenendo una predizione. La predizione viene poi confrontata con l'output target, calcolando attraverso la funzione di perdita l'errore di predizione. Sulla base della perdita ottenuta, ad ogni iterazione si andrà a propagare in *backward* l'errore, effettuando l'aggiornamento dei pesi seguendo uno dei due approcci:

- approccio batch (adatto in casi di regressione non lineare), dove le modifiche ai pesi vengono apportate una volta che alla rete è stato presentato l'intero insieme degli esempi. L'idea di base è quella di effettuare meno modifiche ma sostanziali;
- approccio sequenziale (adatto in casi di *pattern recognition*), basato su tante piccole modifiche, nel quale i pesi vengono aggiornati dopo la presentazione di ogni singolo pattern. È l'approccio più utilizzato, che permette, fissando il parametro di

apprendimento sufficientemente piccolo e scegliendo in modo casuale gli esempi da presentare alla rete, una vasta esplorazione dello spazio della funzione di costo.

Quello descritto sinora non è altro che il comportamento del più famoso ed utilizzato algoritmo di addestramento di una rete neurale, introdotto nel 1986 da Rumelhart, Hinton e Williams: l'algoritmo della retro-propagazione dell'errore o *Error back-propagation* [79].

L'algoritmo consiste in una tecnica d'apprendimento tramite esempi e si tratta sostanzialmente una generalizzazione dell'algoritmo d'apprendimento del percettrone di Rosenblatt. L'addestramento di un MLP viene solitamente realizzato utilizzando un algoritmo di *back-propagation* che prevede, come anticipato, due fasi: *forward* e *back propagation* [80].

Nella prima i pesi assumono dei valori fissi e vengono calcolate tutte le attivazioni dei neuroni della rete, dal primo *layer* proseguendo fino all'ultimo. Nella seconda il risultato generato dalla rete viene confrontato con quello desiderato e se ne calcola l'errore. L'errore viene così propagato nel senso inverso a quello delle sinapsi, con l'intento di minimizzarlo, modificando i pesi di conseguenza. Alla fine di questa fase comincia una nuova iterazione con la *forward propagation*.

L'algoritmo di apprendimento della *back-propagation* è semplice da implementare e computazionalmente efficiente, in quanto la sua complessità è lineare nei pesi sinaptici della rete. Tuttavia, una delle principali limitazioni dell'algoritmo è che non sempre converge verso un risultato e può essere estremamente lento quando si ha a che fare con reti di grandi dimensioni.

Una serie di caratteristiche fa delle reti neurali artificiali dei sistemi ampiamente utilizzati, tra cui:

- l'elevato parallelismo, che permette di processare grandi quantità di dati in tempi relativamente brevi;

- la tolleranza ai guasti, ovvero la capacità di resilienza nei confronti di eventuali malfunzionamenti di alcune unità;
- la tolleranza al rumore, ovvero la capacità di operare quasi sempre in modo corretto nonostante input imprecisi o incompleti;
- l'evoluzione adattativa, ovvero la capacità di auto-aggiornarsi in caso di modifiche ambientali.

D'altro canto, non vanno però dimenticati il funzionamento a *black* box delle reti neurali, che si accompagna ad elevati livelli di complessità e numerosità dei parametri, l'incapacità di prevedere la lunghezza della fase di addestramento e la necessità di una enorme mole di dati per l'addestramento oltre che una grande potenza computazionale.

Per categorie come il *data mining*, *optimization*, elaborazione di modelli predittivi e simulativi e classificazione, le reti neurali rappresentano tutt'oggi lo strumento migliore per la gestione di analisi e problemi ad esse appartenenti.

Capitolo 3

L'analisi empirica

Il capitolo ha come oggetto la presentazione della realtà aziendale che ha permesso di elaborare l'analisi su di un caso di business reale, nonché la fase di preparazione ed elaborazione della base dati attraverso l'algoritmo di *Random Forest* e la rete neurale.
Dopo una breve presentazione dell'azienda SEWS-CABIND e del Gruppo Sumitomo, al fine di comprendere al meglio le caratteristiche della base dati, viene esposto il processo di produzione del cablaggio e con esso spiegata la struttura ed il funzionamento delle macchine saldatrici ad ultrasuoni *Minic* III della *Schunk Sonosystem*. Seguono i dettagli relativi all'analisi empirica condotta sui file log (i.e. l'output del processo di produzione del cablaggio) mediante l'elaborazione di un algoritmo *Random Forest* e di una rete neurale *feed-forward* a tre strati. Per l'elaborazione di entrambi i modelli ci si è serviti del linguaggio di programmazione Python.

3.1 SEWS-CABIND e la produzione del cablaggio

Il Gruppo Sumitomo è uno dei maggiori gruppi industriali esistenti in Giappone e si occupa di una vasta gamma di settori, con tre società (*Sumitomo Mitsui Banking Corporation*, NEC e *Sumitomo Electric Industries*) inserite dal Financial Times tra le 500 più

importanti del mondo per capitalizzazione. La gestione di ogni società del Gruppo è indipendente, ma le origini ed i valori fondamentali sono comuni e risalenti a Masatomo Sumitomo, un monaco buddista vissuto cinque secoli fa.

SEWS-CABIND è di proprietà della Sumitomo Electric Group (SEG), attraverso le partecipazioni delle due aziende Capogruppo *Sumitomo Wiring System Ltd.* e *Sumitomo Electric Industries Ltd*.

SEWS-CABIND opera nel segmento business dell'*automotive* di SEG, progettando e producendo cablaggi per la trasmissione di energia elettrica alle varie componenti del veicolo.

SEWS-CABIND nasce ufficialmente il 4 maggio del 2001, quando la Società originaria Cabind Automotive S.p.A., operante nel settore dei cablaggi per il mercato «del Bianco» (i.e. lavatrici, frigoriferi, lavastoviglie, etc.), viene acquisita dal Sumitomo Electric Group.

Già dal 1996 la *Sumitomo Wiring Systems Ltd*. aveva iniziato a collaborare con Cabind S.p.A., con un Accordo di Assistenza Tecnica relativo a progetti per il Gruppo Fiat.

La nuova Società nasce quindi da un terreno comune, sviluppatosi in anni di partnership e condivisione di know-how. Il settore cui si rivolge è quello dell'*automotive* e la sede designata Collegno. In breve tempo entrano a far parte della neonata SEWS-CABIND anche le filiali marocchina e polacca di Cabind Automotive S.p.A.

L'obiettivo dell'operazione appare chiaro fin dall'inizio: l'espansione nel settore dei cablaggi (che già vedeva presenti alcune aziende di produzione del Sumitomo Electric Group), per diventare fornitore di aziende automobilistiche in Europa.

Caratteristica distintiva di SEWS-CABIND è la plasmabilità dei processi e una conseguente velocità di risposta a qualsiasi esigenza del cliente. Forza e stabilità finanziaria di una grande realtà si mescolano con la flessibilità e l'adattabilità tipiche della piccola azienda. Questo connubio è il reale valore aggiunto di SEWS-CABIND, ottenuto grazie ad un

concetto fortemente vissuto da ogni livello aziendale: fare proprie le esigenze del cliente, assumendole come un target da raggiungere.

Un sincretismo tra la tipicità della cultura italiana e la mentalità rigorosa del Giappone, tra standardizzazione della produzione e dei workflow produttivi, migliore qualità del prodotto e ambienti industriali più sicuri. Tra i principali clienti si contano Stellantis e CNHi.

3.1.1 Il Gruppo SEWS-CABIND

Partner globale nella fornitura di cablaggi e componenti per il settore automobilistico, si costituisce di otto stabilimenti produttivi e più di 10.000 dipendenti, con tre filiali che gestiscono siti di produzione locali: SEWS-CABIND Maroc S.A.S., con sei stabilimenti ad Ain Harrouda (quartier generale), a Berrechid e Ain Sebaa; SEWS-CABIND Poland sp.z.oo, con due stabilimenti storici a Zywiec e SEWS-CABIND Albania Sh.p.k, dove l'ultimo stabilimento aperto a Bathore Kamez ne rappresenta la sede centrale.

SEWS-CABIND Maroc S.A.S. è specializzata in cavi e manifattura di cablaggi ed è sul mercato dal 2001, anno in cui è stata stabilita come sussidiaria di SEWS-CABIND Italia. Nel 2013 il quartier generale viene spostato da Casablanca alla vicina città di Mohammedia. Oltre allo stabilimento di Ain Harrouda, si contano ad oggi altri quattro siti produttivi, per un totale di 7925 impiegati: Aïn Harrouda con 2525 dipendenti; due in Berrechid rispettivamente con 2865 e 600 dipendenti; Aïn Sebaâ con 1624 dipendenti.

La produzione marocchina è per lo più destinata al cliente Stellantis, al quale la Società si impegna di consegnare i prodotti in un tempo record di circa tre ore.

Degne di nota sono la particolare attenzione data alla forza lavoro femminile, inserita a pieno titolo in un programma di sviluppo e di supporto alla carriera e l'impegno in iniziative di *Corporate Social Responsibility*.

SEWS-CABIND Poland sp.z.oo è specializzata nella produzione di cablaggi ed è sul mercato dal 2001, quando l'azienda è stata stabilita come sussidiaria di SEWS-CABIND Italia. La sede centrale è nella città di Zywiec, un centro pittoresco situato nell'omonima valle, 90 km a sud-ovest di Cracovia. La storia della Società vede l'acquisizione dello stabilimento di Leśnianka nel 2001 e l'ampliamento del sito produttivo di Grunwaldzka nel 2013, al fine di far fronte alla sempre crescente richiesta da parte dei clienti.

L'azienda conta ad oggi due stabilimenti produttivi di cablaggi, entrambi in Zywiec ed impiega circa 1681 risorse, vantando un importante primato a livello regionale ed europeo con la produzione di oltre 900 tipologie differenti di cablaggi prodotti per l'automotive.

Come in ogni azienda di SEWS-CABIND, anche in Polonia è forte l'attenzione dedicata ad iniziative di *Corporate Social Responsibility*, a cui impiegati e relative famiglie prendono parte con entusiasmo. Uno dei progetti più sentiti è il *World clean up* che si tiene ogni anno dal 2012. Altrettanto ricorrenti sono gli *Open days* e i *Family days*.

SEWS-CABIND Albania Sh.p.k è specializzata nella produzione di cablaggi destinati al settore automotive europeo. Fondata come nuova azienda del Sumitomo Electric Group (SEG) nell'agosto del 2019, vede da subito un trend di rapida crescita, divenendo una solida realtà aziendale con circa 1000 dipendenti. Presto si consolida come la prima e la più grande azienda di produzione di cablaggi nell'area di Tirana, orientandosi sempre di più verso un modello di produzione snello e flessibile.

Come tutte le società del gruppo, anche SEWS-CABIND Albania si impegna ad adottarne concretamente i principi, non solo in termini di processi di produzione, ma anche nella creazione di un solido team ispirato dal *Sumitomo Spirit* sin dagli albori [81].

3.1.2 La produzione del cablaggio

Il cablaggio può essere considerato come il sistema nervoso centrale del veicolo, in quanto responsabile del collegamento tra i componenti hardware e le parti elettroniche.
La fase di progettazione del cablaggio vede coinvolte numerose variabili. La prima fra queste è la struttura del veicolo. La varietà del cablaggio dipende *in primis* dalla configurazione e dalle funzioni del veicolo su cui verrà installato e non per ultimo dai componenti elettronici che i cablaggi dovranno collegare e dalle nonché dalla componente *customer-specific*. Ne consegue che questi siano una delle ultime parti a comporre il veicolo ad essere progettate ma tra le prime ad essere costruite nella fase *body-in-white* della catena di montaggio.[32]
Spesso i processi di progettazione più avanzata si servono di strumenti di prototipazione digitale o *Digital Mock-Up* (DMU) che permettono la simulazione di assemblaggi ed interazioni tra i diversi componenti, fungendo da base per lo sviluppo del prodotto e consentendone la verifica già nelle primissime fasi di sviluppo del veicolo.
Il completamento della fase di progettazione porta alla generazione della distinta base (DiBa). La distinta base, detta anche *Bill Of Materials* (*BOM*) è un documento che definisce tutti gli elementi necessari per la realizzazione di un determinato prodotto.[33] Nell'industria delle macchine utensili assume due diverse forme: l'*Engineering BOM* (*EBOM*), una distinta tecnica creata dagli ingegneri durante la fase di progettazione e la *Manufacturing BOM*

[32] Fase della catena di montaggio della macchina che avviene dopo l'unione del telaio e prima della verniciatura e dell'integrazione nella struttura di parti come il motore, i sottoassiemi del telaio o le finiture.

[33] Si tratta di un elenco completo degli assiemi (materiali principali) e dei sottoassiemi (sottocomponenti), nonché delle quantità necessarie di ciascun item corredate dalle istruzioni per il processo di produzione.

(*MBOM*), la distinta base di produzione utilizzata anche per pianificare l'acquisto delle materie prime.

La *BOM* mette dunque in evidenza i componenti ed i cavi necessari alla realizzazione del cablaggio. È sulla base della domanda del cliente e dei tempi di produzione e di consegna dei cablaggi richiesti che il dipartimento della logistica pianifica le consegne del materiale. La fase di produzione di un cablaggio inizia quindi con l'arrivo dei materiali elencati nella sua *BOM*.

La seconda fase è denominata *Incoming Quality Inspection* o *Incoming Quality Check* (IQC). Durante questa fase il dipartimento *Plant Incoming Inspection* ha il compito di verificare che i materiali ricevuti dai fornitori rispondano agli standard richiesti conducendo indagini di laboratorio. Una volta passati i controlli, i cavi ed i componenti vengono spostati nei magazzini in attesa di essere utilizzati nella produzione.

La prima attività eseguita durante la fase di produzione del cablaggio è il taglio dei cavi con apposite *cutting-machines*, così che la lunghezza corrisponda a quanto definito nella fase di progettazione. Segue il procedimento di *crimping*, durante il quale il filo con l'isolamento spelato viene crimpato nella sezione *wire barrel* del terminale, così da garantire il collegamento elettrico e la forza di ritenzione del cavo. Al termine del procedimento, utile per permettere ai cavi di essere utilizzati sia nello scambio di dati che nella trasmissione elettrica, seguono rigidi test di qualità fatti con l'ausilio di appositi dispositivi o piattaforme.

Il passaggio successivo prevede che le estremità del filo opposte a dove è stata eseguita la crimpatura vengano spellate e i nuclei rimangano scoperti, così da servire in seguito per il collegamento di terminali, *connector housings* o moduli.

Segue il procedimento di *twisting* o attorcigliamento dei fili, atto ad eliminare l'interferenza elettromagnetica che i flussi di corrente elettrica per cui vengono utilizzati i cablaggi potrebbero causare.

Una volta eseguito il *twisting* si passa alla fase di saldatura dei cavi con macchine saldatrici ad ultrasuoni, la cui tecnica di saldatura consiste nell'applicazione di una leggera pressione tramite oscillazione meccanica ad alte frequenze. Questa tecnica di giunzione è particolarmente adatta per i metalli non ferrosi come rame, alluminio e rispettive leghe. Si noti che la fase di incollaggio avviene allo stato solido, ovvero senza alcuna fusione delle parti da unire, mediante oscillazioni a bassa pressione e ad alta frequenza, creando in una frazione di secondo una connessione permanente, solida e metallurgicamente pura.
Il processo di stampaggio, che segue la saldatura ad ultrasuoni, consiste nel posizionamento dei cavi in appositi stampi dove viene iniettato un materiale termoplastico fuso, che, una volta raffreddato, si solidificherà fissando meccanicamente i connettori ai cavi. In questo modo cavi e connettori diventano un pezzo unico, con elevate capacità di resistenza a vibrazioni e urti e un ottimo livello di flessibilità.
La produzione del cablaggio si conclude con il rivestimento tramite apposita nastratura, il cui fine è quello di rendere uniti i cavi, proteggerli da agenti atmosferici ed eventuali urti.
I cablaggi vengono poi assemblati, fase in cui viene effettuata anche la verifica che questi soddisfino i requisiti identificati nelle specifiche di progettazione.

3.1.3 La saldatura ad ultrasuoni delle macchine Schunk

Lo studio empirico si basa sull'analisi qualitativa del processo di saldatura ad ultrasuoni delle macchine Minic III della Schunk Sonosystem utilizzate negli stabilimenti SEWS-CABIND (Figura 3.15).
Si tratta di dispositivi che possono essere utilizzati sia per la saldatura cavo-a-cavo, sia per quella cavo-a-terminale e per i quali non è necessario il ricorso a materiale di riempimento. Sono solitamente dotati di collegamento ad un hardware, utilizzato dall'operatore per la definizione dei parametri sulla cui base la macchina eseguisce la saldatura. Generalmente

queste impostazioni vengono settate, testate ed eventqualmente modificate durante la fase di preparazione della macchina, così da prepararla per l'esecuzione effettiva della saldatura.

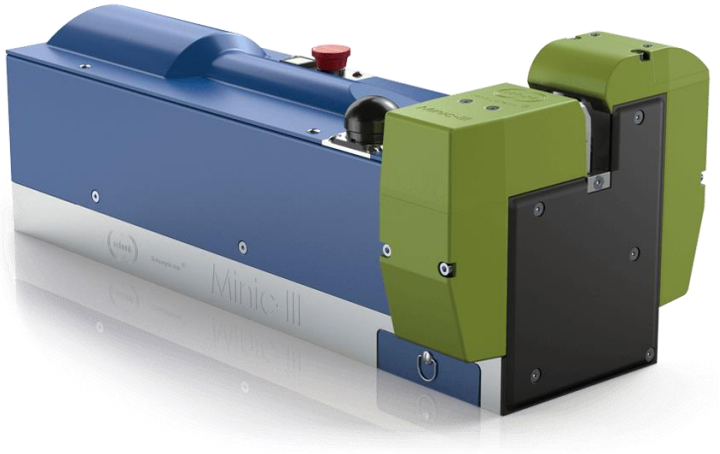

Figura 3.15 Macchina Minic III; *http://www.sews-cabind.com/*, 13/04/2022.

Il procedimento con cui gli operatori eseguono la saldatura, sia effettiva che di prova, può essere riassunto con una sequenza di operazioni:

- l'operatore inserisce il cavo tra l'incudine e la ganascia laterale della macchina (Figura 3.16);

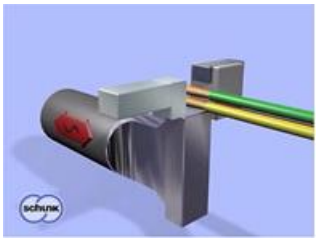

Figura 3.16 Cavo inserito nella ganascia; *http://www.sews-cabind.com/*, 13/04/2022.

- l'operatore fa scorrere la ganascia laterale in modo da bloccare i cavi da saldare tra la ganascia e l'incudine (Figura 3.17);

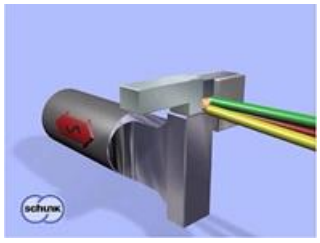

Figura 3.17 Ganascia chiusa, in attesa di avviare la saldatura; *http://www.sews-cabind.com/*, 13/04/2022.

- la macchina esegue la saldatura, impiegando qualche secondo;
- l'operatore fa scorrere la ganascia laterale così da liberare l'insieme dei cavi saldati, che vengono rimossi (Figura 3.18).

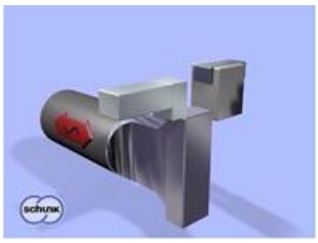

Figura 3.18 Ganascia aperta, cavi rimossi; *http://www.sews-cabind.com/*, 13/04/2022.

La macchina Minic III è composta da un generatore ad ultrasuoni, da un *converter*, da un *booster* e da un sonotrodo, tutti coinvolti nell'esecuzione della saldatura:

- il generatore di ultrasuoni fornisce un'onda elettrica di forma sinusoidale, di frequenza pari a quella di risonanza del trasduttore. Il generatore viene solitamente integrato con un controllore d'ampiezza (assicura che l'ampiezza fornita sia costante) e un controllore di frequenza (assicura che la frequenza sia quella necessaria);
- Il *converter* o trasduttore converte gli impulsi elettrici in movimento meccanico (Figura 3.19);

Figura 3.19 Trasduttore di una saldatrice ad ultrasuoni; *http://www.sews-cabind.com/*, 13/04/2022.

- il *booster* riceve le vibrazioni meccaniche dal trasduttore e le amplifica. Un comune range di amplificazione delle vibrazioni è 1:1 – 1:2,5 (Figura 3.20);

Figura 3.20 Booster di una saldatrice ad ultrasuoni; *http://www.sews-cabind.com/*, 13/04/2022.

- il sonotrodo è il componente che esegue materialmente la saldatura. Può avere diverse forme e dimensioni, a seconda dell'utilizzo e del materiale da saldare. Se il giunto di saldatura è inferiore a 6 mm dalla zona di contatto con il sonotrodo si parla di saldatura a campo vicino, a campo lontano se superiore (Figura 3.21).

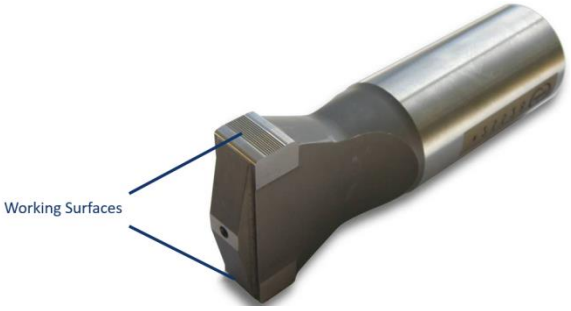

Figura 3.21 Sonotrodo di una saldatrice ad ultrasuoni; *http://www.sews-cabind.com/*, 13/04/2022.

La fase di saldatura vera e propria avviene in pochi secondi, lasso temporale all’interno del quale il sonotrodo esegue diverse operazioni (alcune quasi invisibili all’occhio umano).
Prima di tutto effettua un processo definibile di levigazione, in cui le rugosità e l’ondulazione delle aree di giunzione vengono appianate grazie a forze di taglio oscillanti che frantumano le impurità di disturbo e le eliminano dall’area di contatto. Quest’operazione di convergenza delle due parti continua fino al momento in cui entrano in gioco le forze fisiche di legame. Analizzando la temperatura raggiunta in questa fase, si può osservare che l'attrito e la deformazione plastica provocano un aumento della temperatura nella zona di contatto, che tuttavia non è sufficiente a provocare la fusione dei materiali.
A questo punto le due superfici di contatto si attivano ed inizia lo scambio di elettroni e la conseguente creazione di un forte legame metallico.
Infine, la deformazione plastica delle zone di giunzione porta ad una deformazione elastica del cristallite che innesca una tensione elastica interna nel materiale. La diffusione atomica nel reticolo metallico, favorita dall'aumento della temperatura nella zona di contatto, nonché le sollecitazioni cicliche dei materiali, portano alla riduzione di tale tensione e alla cosiddetta rigenerazione dei cristalli. Il processo di saldatura è giunto a conclusione.

Al termine di ogni ciclo di saldatura le macchine Schunk ad ultrasuoni producono come output una riga di file .log, contenente quello che può essere considerato un resoconto sulla saldatura appena effettuata. Per l’analisi empirica ci si è serviti dei file .log generati da cinque macchine saldatrici SEWS-CABIND in un arco temporale che va da gennaio 2016 ad aprile 2022. Questi dati hanno costituito il data set, opportunamente rielaborato e diviso in training set e test set, su cui si è costruita un’analisi supervisionata per mezzo dell’algoritmo noto come *Random Forest* e di una rete neurale di tipo *feed-forward*.

Il paragrafo successivo intende fornire nozioni basilari a proposito del linguaggio di programmazione utilizzato per l'implementazione dell'algoritmo di apprendimento automatico e delle librerie cui si è deciso di appoggiarsi.

3.2 Python e le librerie utilizzate

Python è il linguaggio di programmazione utilizzato per lo sviluppo dell'analisi empirica.

Si tratta di un linguaggio di programmazione interattivo e orientato agli oggetti, che include moduli, eccezioni, tipizzazione dinamica e classi. E' in grado di supportare altri paradigmi di programmazione oltre quella ad oggetti, come la programmazione funzionale e quella procedurale. Vanta la possibilità di interfacciarsi con diversi tipi di *system call*, librerie e sistemi a finestra ed è estensibile in C o C++. Può anche essere utilizzato come linguaggio di estensione per applicazioni che necessitano di un'interfaccia programmabile. Python è in grado di combinare un potenziale notevole ad una sintassi molto chiara e infine si tratta di un linguaggio «portatile»: funziona su molte varianti di Unix, su MacOS e su Windows [82].

La nascita del linguaggio è attribuibile a Guido van Rossum, programmatore olandese laureatosi all'Università di Amsterdam con un Master in Matematica ed Informatica e conosciuto come il «Benevolent Dictator for Life» di Python.

Il desiderio di rendere Python «facilmente programmabile» è visibile sin dalla versione 2.0 rilasciata nel 2000, tra le cui funzionalità ricordiamo la *list comprehension* e il *garbage collector*.

La *list comprehension*, derivante dai linguaggi *SETL* e *haskell*, è un costrutto sintattico che permette di definire e creare liste in modo conciso e leggibile [83]. Il *garbage collector* riguarda invece la gestione della memoria e rappresenta una tecnica di gestione dinamica, come spiegato da Benjamin Zorn nel suo *The Measured Cost of Conservative Garbage*

Collection [84]. Il *garbage collector* rientra tra i meccanismi di gestione automatica della memoria; è presente in Python sottoforma di modulo, che è possibile disabilitare in caso di necessità.

Le caratteristiche e le funzionalità che definiscono Python lo hanno reso molto popolare. Un'indagine effettuata da StackOverflow nel 2021 rivela che circa il 68% degli sviluppatori di software che ha lavorato in modo approfondito con Python ha mostrato interesse a continuare a sviluppare in questo linguaggio [85].

Volendo riassumere in poche righe le motivazioni del grande successo di Python, citiamo:

- la facilità di apprendimento grazie ad una sintassi semplificata;
- la flessibilità di applicazione in più settori, dallo sviluppo web a sviluppi inerenti *data analytics*, intelligenza artificiale, *machine learning*, *data science* o *data engineering*;
- una community attiva e reattiva, che fornisce a programmatori di tutti i livelli facili accessi a documentazione, guide, tutorial e supporto in caso di problemi o domande.

È ormai universalmente riconosciuto che in ambito *data science* e *data analytics*, con la crescente diffusione di progetti legati al *Machine learning*, *Cloud computing* e *Big data*, Python sia secondo solo al linguaggio *R*.

Le principali librerie di Python utilizzate per lo sviluppo dell'analisi sono:

- *Pandas;*
- *Numpy;*
- *Matplotlib;*
- *Seaborn;*
- *Scikit-learn;*
- *Tensorflow;*
- *Keras.*

La libreria *Pandas* può essere vista come un insieme di strutture dati veloci e flessibili, utilizzabili dai programmatori quando hanno a che fare con dati relazionali o dati etichettati. Si tratta di un potente strumento *open source* per analizzare e manipolare dati. Supporta i *data scientist* in tutto il ciclo di vita del *processing* dei dati [86].

NumPy fornisce oggetti come *array* multidimensionali o derivati (e.g. matrici) e funzioni per agire in modo rapido ed efficace su di essi attraverso operazioni come la manipolazione matematica, logica o di dimensione, oppure operazioni statistiche di base, come la simulazione casuale. Un elemento molto importante del pacchetto è l'oggetto *ndarray*, che incapsulando matrici n-dimensionali di tipi di dato omogenei permette l'esecuzione di operazioni molto performanti.

Matplotlib permette ai programmatori di produrre grafici bidimensionali come istogrammi, spettri di potenza e grafici a dispersione in diversi formati ed ambienti [87].
In particolare, *matplotlib.pyplot* è una raccolta di funzioni che operano come API per apportare modifiche ad una figura come l'aggiunta di etichette [88].

Seaborn è un modulo che si appoggia su *matplotlib*, legato alle strutture dati di *pandas* per creare grafici statistici in Python[89].

Scikit-learn è una libreria che fornisce supporto ai programmatori in problemi relativi al *Machine Learning* (supervisionato e non) durante l'intero ciclo di elaborazione del problema: dal *pre-processing* dei dati, alla scelta dei parametri ottimali da fornire ad un algoritmo di *Machine Learning*, dalla fase di apprendimento (in cui sarà alimentato con i dati del training set), alla sua analisi e valutazione [90]. Mette a disposizione diversi modelli di apprendimento automatico, sia supervisionati che non supervisionati: da classificatori *Random Forest* con il pacchetto *sklearn.ensemble*, ad algoritmi *Nearest Neighbors* attraverso *sklearn.neighbors*, a modelli di clustering come il K-Means tramite *sklearn.cluster*.

Il pacchetto *sklearn.preprocessing* è il pacchetto di *Scikit-learn* che fornisce all'utente le funzionalità per operare nella fase di pre-elaborazione dei dati, mettendo a disposizione funzioni comuni e classi trasformatori per modificare i vettori relativi alle caratteristiche.
Il pacchetto *sklearn.model_selection* mette a disposizione del programmatore, una volta selezionato l'algoritmo di *machine learning*, due differenti approcci di tipo *fit and score* per ottenere gli iper-parametri.[34]
Il modulo *sklearn.metrics*, infine, rende disponibili tool e funzioni per implementare metodi di valutazione della qualità di un modello predittivo.

Tensorflow è una piattaforma end-to-end per il *Machine Learning* che dispone di un insieme completo di strumenti e librerie e permette agli sviluppatori di approfondire l'argomento e di creare e distribuire facilmente applicazioni basate sull'apprendimento automatico

Keras è un'API di *Deep Learning* scritta in Python, in esecuzione sulla piattaforma di *Machine learning TensorFlow*. Le strutture dati alla base di *Keras* sono i *layers* e i modelli.
Il primo è uno degli elementi base che costituiscono le reti neurali in *Keras*, dove ogni strato è caratterizzato da una funzione di computazione *tensor-in-tensor-out* (il cosiddetto metodo *call* del livello) che mappa il tensore. Ogni *layer* presenta una struttura riconducibile ad un *array* multidimensionale con uno stato, memorizzato in variabili *Tensorflow*, detto peso dello strato.
I modelli, invece, possono essere visti come il raggruppamento di più livelli in un oggetto.

[34] Gli iper-parametri sono quei parametri che non vengono calcolati direttamente dai modelli, ma che vengono passati a questi in fase di costruzione.

3.3 L'analisi empirica

Si è giunti al cuore dell'argomentazione. Nel presente paragrafo vengono descritti la struttura dei dati su cui ci si è basati per lo sviluppo dell'analisi, il conseguente processo di *cleaning* con l'eliminazione dei dati sporchi, cui è seguito quello di preelaborazione.
Per assicurarsi che i dati elaborati dal modello predittivo siano attendibili, è necessario correggere l'errore derivante da alcune rilevazioni effettuate dalla macchina durante il processo di saldatura e filtrare le rilevazioni effettuate in processi che generalmente tendono a sporcare i dati.
Una volta ottenuto un data set appropriato, lo si è utilizzato per il processo di *learning*, elaborato attraverso l'utilizzo dell'algoritmo di *Random Forest* e di una rete neurale *feed-forward* a tre strati.

3.3.1 La raccolta dei dati

Il data set utilizzato per l'addestramento del classificatore proviene da file .log generati da cinque macchine saldatrici ad ultrasuoni Minic III utilizzate negli stabilimenti di SEWS-CABIND. L'analisi copre un arco temporale che va da gennaio 2016 ad aprile 2022. I file .log più recenti (i.e. mese corrente e quello precedente) sono di norma archiviati in cartelle condivise direttamente sulle macchine, quelli risalenti ai mesi precedenti si trovano invece in cartelle di backup di un apposito server.

I file presentano una struttura configurabile, che può variare da macchina a macchina. Tutte e cinque le macchine esaminate hanno in comune le seguenti informazioni (reperibili sui manuali delle macchine):

- *Cross-section*, misura della sezione trasversale dell'area da saldare (in mm^2), specificata dall'operatore in fase di configurazione. E' la chiave per fornire i parametri di riferimento da analizzare e può variare tra 0.26 mm^2 e 30 mm^2;

- *Width*, misura della larghezza della giuntura, specificata dall'operatore in fase di configurazione. Può avere un valore compreso tra 1 mm e 15 mm;
- *Pressure*, Indica la pressione che deve essere applicata dal sonotrodo per effettuare la saldatura, specificata dall'operatore in fase di configurazione. Il valore di questo campo può oscillare tra 0 e 6 bar;
- *Amplitude*, ampiezza dell'oscillazione del sonotrodo durante la saldatura in termini di percentuale rispetto alla massima disponibile, variabile da 0 a 100. Il valore è legato ad un parametro specificato dall'operatore in fase di configurazione;
- *Welding Mode*, modalità con cui la macchina ha eseguito la saldatura. Il valore viene specificato dall'operatore in fase di configurazione e le opzioni possibili sono *Energy* e *Delta-H*;
- Energy, uno dei due valori che può assumere il parametro *Welding Mode*. Indica quanta energia dovrà essere applicata durante la saldatura. Il valore viene specificato dall'operatore in fase di configurazione e il massimo che può assumere (dipende dal materiale piezo-elettrico utilizzato) è di solito 4000 W/s;
- *Delta*-H, uno dei due valori che può assumere il parametro *Welding Mode*. Indica la differenza di altezza misurata prima e dopo la saldatura da applicare in fase di saldatura. Il valore viene specificato dall'operatore in fase di configurazione e può variare tra 0 mm e 10 mm;
- *Compacting height,* l'altezza dei cavi da saldare misurata dal sensore prima che venga iniziata la saldatura. L'altezza massima accettata da macchine Schunk Minic III è pari a 10 mm;
- *Compacting height ref.*, l'altezza di riferimento dei cavi da saldare che dovrebbe essere misurata dal sensore prima che venga iniziata la saldatura. Essendo questo campo direttamente collegato al campo precedente, l'altezza massima accettata è pari a 10 mm;

- *Welding height*, l'altezza dei cavi da saldare misurata dal sensore una volta completata la saldatura. La massima altezza prodotta dalle saldatrici ad ultrasuoni è uguale a 10 mm;
- *Welding height ref.*, l'altezza di riferimento dei cavi da saldare che dovrebbe essere misurata dal sensore una volta completata la saldatura. Essendo questo campo direttamente collegato al campo precedente, l'altezza può assumere un valore massimo di 10 mm;
- *Welding time*, il tempo impiegato per effettuare la saldatura misurato da un timer interno alla macchina. Il manuale delle macchine Schunk Minic III non riporta alcuna indicazione sull'intervallo di valori entro i quali può oscillare questo campo, ne consegue che le assunzioni relative a questo campo sono state fatte sulla base dell'analisi dei file di log a disposizione;
- *Welding time ref.*, il tempo di riferimento che dovrebbe impiegare la macchina per effettuare la saldatura. Il manuale delle macchine Schunk Minic III non riporta alcuna indicazione sull'intervallo di valori entro i quali può oscillare questo campo, ne consegue che le assunzioni relative a questo campo sono state fatte sulla base dell'analisi dei file di log a disposizione;
- *Mode*, modalità in cui stava operando la macchina quando è stata eseguita la saldatura. I due possibili valori sono *Setup* e *Production*. Il primo indica che la macchina si trova in fase di preparazione, per cui la saldatura eseguita è una prova utile all'operatore per verificare i parametri impostati, mentre il secondo che la macchina sta eseguendo una saldatura effettiva;
- *Error Nr.*, sulla base dell'indicazione dell'operatore a saldatura conclusa, questo campo assume un valore che rappresenta il codice di errore assegnato per indicare una valutazione della saldatura. Se il valore di questo campo è pari a 200 significa

che la saldatura è stata effettuata in modo corretto, altri valori indicano che si è verificato qualche problema in fase di saldatura;

- *Error Text*, descrizione *user-friendly* del valore indicato nel campo *Error Nr*.

Il contenitore scelto per memorizzare i dati utilizzati per l'analisi è un *dataframe*, ovvero la struttura dati principale della libreria *pandas*. Si tratta di una tabella bidimensionale di dimensioni variabili e tipi di dato potenzialmente eterogenei. Utilizzando la combinazione delle librerie *os* per navigare all'interno dei folder dove sono memorizzati i file e della funzione *read_csv* del modulo *pandas*, che restituisce un *dataframe* basato sul contenuto del file con una struttura simile ad un file csv*,* è stata ottenuta la struttura dati popolata, pronta per essere analizzata.

Più nello specifico, alla funzione è stato passato il *path* di un file csv temporaneo, creato sulla struttura di un file di log, oltre a tre indicazioni che forniscono informazioni sulla struttura del file csv:

- il parametro *quoting* impostato su *QUOTE_NONE* istruisce il *reader* utilizzato dalla funzione a non eseguire una particolare elaborazione ai caratteri riconducibili a virgolette;
- il parametro *sep*, impostato su ';' indica che il carattere utilizzato nel file ricevuto in ingresso per separare tra di loro i campi è il punto e virgola;
- il parametro *encoding* fornisce al *reader* del file l'informazione che nel file è stata utilizzata la codifica standard di Python chiamata *latin-1*, detta anche *iso-8859-1*.

3.3.2 *Data cleaning*

Una volta reperita la base dati, il processo di «pulizia» rappresenta una delle principali tecniche propedeutiche alla fase di preelaborazione dei dati. Con il termine generico di *Data cleaning* si identificano una serie di interventi sulla base dati come

l'aggiunta di valori mancanti, l'aggiustamento dei dati rumorosi, l'identificazione e eliminazione degli *outliers*, la soluzione delle inconsistenze.

I file di output delle saldature utilizzati per la costruzione e la valutazione del modello di *Machine Learning* presentano un discreto grado di completezza in tutti i campi oggetto dell'analisi, ma è stato comunque necessario intervenire sulla base dati per renderla adatta all'elaborazione.

La prima problematica affrontata è stata la presenza di valori nulli o mancanti. Si è deciso di gestirla in maniera differente a seconda del campo dove questa si è verificata.

I possibili approcci quando si hanno dati con valori mancanti sono: ignorare le istanze dove si verifica la problematica, oppure sostituire il valore mancante con una generica costante o con media, mediana o il valore più probabile.

La prima soluzione è comunemente adottata quando il valore manca per l'etichetta di classe. Si tratta di un metodo non molto efficace, ma che si rivela il preferibile quando il record contiene diversi attributi con valori mancanti. Si è deciso di adottare questa soluzione quando il problema si è verificato nella colonna obiettivo o in colonne che potevano causare il rischio di perdita di consistenza delle informazioni riportate dal record, ovvero:

- *Error Nr*. (colonna obiettivo), questa è la caratteristica che ricopre il ruolo di etichetta e l'assenza di questa informazione non permetterebbe di capire se il record di file log faccia riferimento ad una saldatura effettuata in modo corretto o meno;
- *Welding Mode* è il campo che indica la modalità di lavoro della macchina (a energia o a differenza di altezza) e la mancanza del dato, imputabile esclusivamente ad un malfunzionamento della macchina in fase di scrittura del file di log, rende inconsistente l'intero record;

- *Mode* è il campo che permette all'operatore di definire se si tratta di una saldatura di prova o effettiva e la mancanza del dato rende inconsistente l'intero record;

Si è poi deciso di intervenire sui valori mancanti o inconsistenti sostituendoli con una costante nelle colonne *Compacting height*, *Welding height* e *Welding Time*. La presenza di valori vuoti o popolati con il carattere «-» deriva dal fatto che l'operatore, in fase di configurazione della macchina, può decidere di non imputare un valore per i parametri *Compacting height*, *Welding height* e *Welding Time*. In caso di campi con valore non popolato o popolato con il carattere «-», il default imputato è 0.

La seconda problematica da gestire riguarda il fatto che le rilevazioni relative alle saldature effettuate in fase di preparazione delle macchine (i.e. *Mode = Setup*) potrebbero sporcare l'analisi, in quanto prove eseguite dall'operatore per trovare i parametri ottimali da utilizzare per le saldature effettive. L'ultimo step riconducibile alla correzione degli errori, per avere un insieme di dati consistenti da fornire in input all'algoritmo di classificazione, è stato dunque applicare un filtro al fine di eliminare la porzione di dati relativi alla fase di preparazione della macchina.

3.3.3 *Data pre-processing*

La preelaborazione dei dati, intesa come la somma dei passaggi come la standardizzazione, la normalizzazione, l'estrazione di funzionalità e la riduzione delle dimensioni, è necessaria per ottenere una migliore classificazione dei dati. Lo scopo del *data pre-processing* è trovare l'insieme di funzionalità in grado di fornire il maggior numero di informazioni e migliorare le prestazioni del classificatore.

La standardizzazione, detta anche *feature scaling*, è un processo utilizzato per normalizzare l'intervallo in cui operano caratteristiche o variabili, considerate in modo indipendente. La normalizzazione delle caratteristiche è invece necessaria per eliminare l'effetto di diverse

caratteristiche quantitative misurate su scale diverse. L'estrazione delle caratteristiche serve ad ottenere nuove variabili, calcolate a partire dai dati grezzi, al fine di ottenere una classificazione coerente o di evidenziare caratteristiche dell'insieme di dati utilizzato. Questo procedimento è molto critico quando si opera con classificatori, dal momento che le prestazioni della classificazione potrebbero essere degradate se le caratteristiche non sono designate correttamente. L'ultimo passaggio consiste nella riduzione di dimensione o *feature selection*. L'obiettivo di questa fase è limitare il numero delle caratteristiche al minimo indispensabile, ma che allo stesso tempo sia in grado di realizzare la massima capacità di generalizzazione [91].

3.3.3.1 *Feature scaling*

Il procedimento di standardizzazione delle caratteristiche è stato eseguito secondo due modalità, a seconda della tipologia di caratteristica valutata: nel caso di caratteristiche di tipo numerico con distribuzione continua è stata utilizzato un *min max scaler* (così da limitare l'intervallo di distribuzione tra 0 e 1), mentre per variabili di tipo discreto, numeriche o alfanumeriche un *label encoder*.

Per il *min-max scaler* sono stati valutati sia la soluzione proposta dalla libreria *preprocessing* del modulo *scikit-learn*, sia una soluzione custom ricavata dal *map* della caratteristica grazie al metodo *apply*, reso disponibile dall'oggetto *dataframe* della libreria *pandas*. Essendo che il valore minimo e quello massimo nella soluzione *scikit-learn* vengono calcolati automaticamente basandosi sull'intervallo entro cui varia la caratteristica esaminata, è stata preferita la soluzione custom, in quanto generalmente il range utilizzato non ricopre tutto il disponibile e la stessa caratteristica potrebbe non essere scalata in modo non uniforme in caso di range differenti.

La standardizzazione utilizzata per questo gruppo di variabili può essere descritta dalla seguente formula:

$$x_{std} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Dove x_{min} e x_{max} indicano rispettivamente il valore minimo ed il valore massimo della caratteristica. Per popolare le variabili x_{min} e x_{max} si è fatto riferimento ai valori indicati nel manuale delle macchine saldatrici Shunk Minic III quando indicati, in caso contrario si è fatto ricorso al valore minimo e massimo ricavati dai valori assunti dalla variabile nei file di log (applicando un margine di errore):

- *Cross-section* da manuale può variare tra 0.26 mm^2 e 30 mm^2. Sono stati utilizzati 0 mm^2 e 30 mm^2 rispettivamente come valore minimo e massimo;
- *Width* da manuale può variare tra 1 mm e 15 mm. È stato usato 0 mm – 15 mm come range di partenza applicato al relativo *min-max scaler*;
- *Pressure* può oscillare tra 0 e 6 bar. I valori 0 e 6 bar sono stati considerati rispettivamente come valore minimo e valore massimo;
- *Amplitude* può assumere valori compresi tra 0 e 100 (campo percentuale). E' stato usato 0– 100 come range di partenza applicato al relativo *min-max scaler*;
- Energy presenta un range di oscillazione che dipende dal materiale piezo-elettrico utilizzato. Può essere generalizzato come 0 W/s - 4000 W/s;
- *Delta-H* può variare tra 0 mm e 10mm. Sono stati utilizzati 0 mm e 10 mm rispettivamente come valore minimo e massimo del *min-max scaler*;
- *Compacting height*, l'altezza massima accettata da macchine Schunk Minic III è pari a 10 mm. È stato usato 0 mm – 10 mm come range;

- *Compacting height ref.,* può assumere gli stessi valori del campo *Compacting height*. È stato usato 0 mm – 10 mm come range;
- *Welding height*, l'altezza massima accettata da macchine Schunk Minic III è pari a 10 mm. E' stato usato 0 mm – 10 mm come range;
- *Welding height ref.* può assumere gli stessi valori del campo *Welding height*. E' stato usato 0 mm – 10 mm come range;
- *Welding time* non presenta alcuna indicazione nel manuale a proposito di un intervallo di valori entro cui può oscillare. Dai file di log emerge che il valore massimo assunto è 988 ms e quello minimo 0 ms. Mantenendo un sufficiente margine di errore è stato utilizzato l'intervallo 0 ms – 4000 ms come range di partenza applicato al relativo *min-max scaler*;
- *Welding time ref.* non presenta alcuna indicazione nel manuale a proposito di un intervallo di valori entro cui può oscillare. Può assumere gli stessi valori del campo *Welding time* e quindi è stato utilizzato l'intervallo 0 ms – 4000 ms come range;
- *Compacting Height Diff Perc* può assumere valori compresi tra 0 e 100 (campo percentuale). E' stato usato 0– 100 come range di partenza applicato al relativo *min-max scaler*;
- *Weld. Height Diff Perc* può assumere valori compresi tra 0 e 100 (campo percentuale calcolato). Sono stati utilizzati 0 mm e 10 mm rispettivamente come valore minimo e massimo del *min-max scaler*;
- *Weld. Time Diff Perc* può assumere valori compresi tra 0 e 100 (campo percentuale calcolato). Sono stati utilizzati 0 mm e 10 mm rispettivamente come valore minimo e massimo del *min-max scaler*.

Le due caratteristiche coinvolte nella normalizzazione attraverso il procedimento di *label encoding* sono:

- *Welding Mod*, che può assumere i valori *Energy* e *Delta-H* a seconda dell'impostazione dell'operatore. I due valori sono stati rispettivamente mappati in 1 e 0;
- *Mode*, che può assumere i valori *Setup* o *Production*. Dal momento che in fase di *data cleaning* tutti i record dove questa caratteristica non era uguale a *Production* sono stati scartati (perché contenenti dati non consistenti), questa caratteristica assume un solo valore che viene mappato con 1.

La normalizzazione eseguita sulla variabile di studio, l'*Error Number*, può essere ricondotta al *label enconding*, anche se con qualche differenza rispetto ai campi *Welding Mode* e *Mode*.

Partendo dal presupposto che l'obiettivo dell'algoritmo sviluppato è verificare la correttezza o meno della saldatura effettuata dalla macchina, dall'analisi dei file di log emerge come la variabile *Error Number* può assumere una serie di valori (Tabella 3.1).

Codice errore	Descrizione errore
200	Saldatura corretta
208, 221	Saldatura errata per differenza tra altezza cavo da saldare e quella impostata nei parametri
209	Saldatura errata per differenza tra altezza cavo saldato e quella impostata nei parametri
210	Saldatura errata perché eseguita in un tempo superiore a quello impostato
222	Saldatura errata perché l'altezza della saldatura richiesta eccede il limite massimo
78, 201, 202, 204, 211, 212, 213, 224, 399, 535, 540, 541, 816	Saldatura errata per problemi generici

Tabella 3.1 Possibili valori assunti dal campo *Error Nr.* con le rispettive descrizioni, elaborazione personale.

A questo proposito si è deciso di eseguire:

- prima il *mapping* della variabile *Error Number* con un valore binario, valorizzandola con 1 in caso di correttezza della saldatura, con 0 in tutti gli altri casi ed effettuare l'analisi su questo presupposto (Tabella 3.2);

Codice errore	Descrizione errore	*Mapping* codice errore
200	Saldatura corretta	1
208, 221	Saldatura errata per differenza tra altezza cavo da saldare e quella impostata nei parametri	0
209	Saldatura errata per differenza tra altezza cavo saldato e quella impostata nei parametri	0
210	Saldatura errata perché eseguita in un tempo superiore a quello impostato	0
222	Saldatura errata perché l'altezza della saldatura richiesta eccede il limite massimo	0
78, 201, 202, 204, 211, 212, 213, 224, 399, 535, 540, 541, 816	Saldatura errata per problemi generici	0

Tabella 3.2 Mappatura del campo *Error Nr.* in un valore binario, elaborazione personale.

- in seguito, il *mapping* della variabile *Error Number* con più di due valori per vedere se un maggiore dettaglio applicato alla variabile studio si sostanzi in una peggiore capacità di classificazione da parte dei modelli (Tabella 3.3).

Codice errore	Descrizione errore	*Mapping* codice errore
200	Saldatura corretta	1
208, 221	Saldatura errata per differenza tra altezza cavo da saldare e quella impostata nei parametri	2
209	Saldatura errata per differenza tra altezza cavo saldato e quella impostata nei parametri	3
210	Saldatura errata perché eseguita in un tempo superiore a quello impostato	4
222	Saldatura errata perché l'altezza della saldatura richiesta eccede il limite massimo	5
78, 201, 202, 204, 211, 212, 213, 224, 399, 535, 540, 541, 816	Saldatura errata per problemi generici	0

Tabella 3.3 Mappatura dei valori di *Error Nr.* in una multi - classe, elaborazione personale.

3.3.3.2 *Feature extraction*

Una volta completata la normalizzazione dei dati, sono state utilizzate le caratteristiche del *dataframe* per aggiungere tre campi calcolati e mettere in evidenza il rapporto tra le misure rilevate dai sensori dalla macchina e quelle di riferimento impostate dall'operatore.

L'operazione è stata eseguita usando il metodo *apply* dell'oggetto *data frame*, passando come parametri la funzione da eseguire e l'impostazione *axis = 1* per indicare che la funzione deve essere eseguita per ogni riga del *dataframe*.

A tale scopo sono stati aggiunti i campi:

- *Compacting Height Diff Perc*, che assume due diversi valori a seconda del fatto che l'operatore abbia impostato o meno l'altezza di riferimento dei cavi da saldare. Nel caso in cui l'altezza di riferimento non sia stata configurata, situazione riconoscibile dal campo *Compacting Height Ref* pari a 0, il campo calcolato vale 0, come se fosse quello aspettato, in modo tale da non introdurre penalità. Altrimenti questo campo assume il valore della percentuale del modulo inteso come la differenza tra la *Compacting Height* e la *Compacting Height Ref* ;

$$Comp.Height\ Diff\ Perc = \begin{cases} se\ Comp.Height\ Ref. = 0 & \rightarrow & 0 \\ altrimenti & \rightarrow & \left| \frac{Comp.Height - Comp.Height\ Ref.}{Comp.Height\ Ref.} \right| \end{cases}$$

- Weld*. Height Diff Perc*, che assume due diversi valori a seconda del fatto che l'operatore abbia impostato o meno l'altezza di riferimento dei cavi al termine della saldatura. Nel caso in cui l'altezza di riferimento non sia stata configurata, situazione riconoscibile dal campo *Weld. Height Ref* pari a 0, il campo calcolato vale 0, come se

fosse quello aspettato, in modo tale da non introdurre penalità. Altrimenti, questo campo assume il valore della percentuale del modulo della differenza tra la *Weld. Height* e la *Weld. Height Ref* rispetto a quest'ultima;

$$Weld.Height\ Diff\ Perc = \begin{cases} se\ Weld.Height\ Ref. = 0 & \rightarrow & 0 \\ altrimenti & \rightarrow & \left|\frac{Weld.Height - Weld.Height\ Ref.}{Weld.Height\ Ref.}\right| \end{cases}$$

- *Weld. Time Diff Perc*, che assume due diversi valori a seconda del fatto che l'operatore abbia impostato o meno il tempo di riferimento che la macchina deve impiegare per eseguire la saldatura. Nel caso in cui la misura del tempo di riferimento non sia stata configurata, situazione riconoscibile dal campo *Weld. Time Ref* pari a 0, il campo calcolato vale 0, come se fosse quello aspettato, in modo tale da non introdurre penalità. Altrimenti, questo campo assume il valore della percentuale del modulo della differenza tra la *Weld. Time* e la *Weld. Time Ref* rispetto a quest'ultima.

$$Weld.Time\ Diff\ Perc = \begin{cases} se\ Weld.Time\ Ref. = 0 & \rightarrow & 0 \\ altrimenti & \rightarrow & \left|\frac{Weld.Time - Weld.Time\ Ref.}{Weld.Time\ Ref.}\right| \end{cases}$$

3.3.3.3 *Feature selection*

La selezione delle funzionalità, come parte della strategia di preelaborazione dei dati, si è rivelata piuttosto efficace, soprattutto in caso di data set con dimensioni elevate. Gli obiettivi della *feature selection* includono: la creazione di modelli più semplici e

comprensibili, il miglioramento delle prestazioni e la preparazione di dati chiari e comprensibili [92].

Possiamo definire formalmente la *Feature Selection* (FS) come segue: sia e_i un'istanza $e_i = (e_{i1}, ..., e_{in}, e_{iy})$, dove e_{ir} corrisponde al r-esimo valore della caratteristica dell'i-esimo campione ed e_{iy} corrisponde al valore della classe di output Y. Assumiamo un training set D con m esempi, le cui istanze e_i sono formate da un insieme X di n caratteristiche e da un insieme di test D_t. $S_\theta \subseteq X$ è definito come il sottoinsieme di funzionalità selezionate prodotte da un algoritmo FS [93].

È possibile identificare tre diversi approcci sui quali si basano gli algoritmi di FS:

- *filtering approaches*, metodi statistici che classificano le caratteristiche in base a criteri specifici. Quando l'insieme delle funzionalità più classificate viene utilizzato per la previsione della classe, le prestazioni della classificazione sono spesso inferiori rispetto a quando vengono utilizzate tutte le funzionalità. Tra questi algoritmi troviamo *ANOVA*, *Pearson correlation*, *variance thresholding*;
- modelli *wrapper*x, dove il processo di selezione è legato alla performance di uno specifico modello di classificazione e la FS è realizzata utilizzando alcuni metodi di ottimizzazione e strategie di ricerca. Una variante molto utilizzata dei modelli *wrapper* è quella randomizzata, che si basa su strategie di ricerca come gli algoritmi genetici (GA), *hill climbing* e *simulated annealing*. SI tratta di un procedimento molto impegnativo dal punto di vista computazionale e che può portare alla selezione di funzionalità distorte rispetto al classificatore;
- metodi FS *embedded*, che incorporano il procedimento di selezione all'interno del processo di sviluppo del modello di apprendimento automatico. Questa tipologia di ricerca dei sottoinsiemi di funzionalità è un processo di ottimizzazione combinato che comprende sia la selezione di un insieme ottimizzato di funzionalità, sia la scelta

dei parametri di ottimizzazione del modello. Ne sono un esempio gli algoritmi *Lasso*, *Ridge*, *Decision Tree* e *Random Forest*.

Per analizzare l'importanza delle caratteristiche riguardo la correlazione con la variabile di output sono stati utilizzati tre algoritmi: 2 di tipo *embedded*, ovvero *LASSO* e il *Random Forest* e uno di tipo *filtering*, come *ANOVA*.

LASSO, acronimo di *least absolute shrinkage and selection operator*, è un metodo che si basa su una regressione lineare avente predittori x_{ij} standardizzati e valori di risposta y_i centrati con i = 1, 2, …, N e j = 1, 2, …, p. L'obiettivo di questo metodo è trovare l'insieme di valori $\beta = \{\beta_j\}$ tali da minimizzare la seguente sommatoria:

$$\sum_{i=1}^{N}\left(y_i - \sum_{j} x_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

Dove λ è un parametro di regolarizzazione non negativo, mentre il secondo termine può essere definito come *l1 penalty* ed è molto importante. L'approccio della penalizzazione *l1* è chiamato anche ricerca delle basi nell'elaborazione del segnale.

LASSO agisce riducendo continuamente i coefficienti verso lo 0 all'aumentare di λ e accade che alcuni coefficienti, se λ è sufficientemente grande, sono ridotti esattamente a 0. In più, restringendo il set di variabili, spesso si è in grado di ottenere un miglioramento dell'accuratezza della previsione, grazie ad un compromesso tra *bias* e varianza.

Utilizzando i concetti di Jianqing Fan e Runze Li in *Variable Selection via Nonconcave PenalizedLikelihood and its Oracle Properties*, possiamo definire la procedura utilizzata per dedurre i coefficienti β_j una *oracle procedure*, in quanto denota le seguenti proprietà [94]:

- seleziona le variabili principali da utilizzare per il modello $\{j: \beta_j \neq 0\}$;
- produce il tasso di stima migliore una volta trovato il corretto *subset* di variabili [95].

ANOVA, acronimo di *Analysis of Variance*, è una tecnica statistica utilizzata per l'analisi della variazione in una variabile di risposta. Si tratta di una variabile casuale continua, misurata in condizioni definite da fattori discreti definibili come variabili di classificazione. *ANOVA* viene comunemente utilizzata per testare l'uguaglianza tra mezzi confrontando la varianza tra i gruppi rispetto alla varianza all'interno dei gruppi, ovvero il cosiddetto errore casuale [96].

Un altro campo di applicazione è nel calcolo della *feature importance*, utile per la valutazione delle caratteristiche nel procedimento di *feature selection* in ambito di *Machine Learning*. In particolare, sfruttando uno degli obiettivi tipici dell'*ANOVA*, ovvero confrontare i mezzi delle variabili di risposta per diverse combinazioni delle variabili di classificazione, viene utilizzato per decidere se una caratteristica mostra una differenza significativa tra due o più classi [97].

La selezione delle caratteristiche basata sul classificatore *Random Forest* sfrutta i punteggi di importanza forniti dal modello. Questo metodo *ensemble* di alberi decisionali, basandosi o sull'impurità di Gini o sull'entropia, implicitamente esegue una selezione delle caratteristiche utilizzando un piccolo sottoinsieme di «variabili forti» solo per la classificazione, fornendo la cosiddetta *feature importance* [98].

Nell'analisi si è deciso di utilizzare come metrica l'entropia:

$$E = -\sum_{i=1}^{n} p(i) \log_2 p(i)$$

dove p_j rappresenta la probabilità della j-esima classe. Questo punteggio di importanza delle caratteristiche fornisce una classifica relativa delle caratteristiche spettrali ed è un sottoprodotto nell'addestramento del classificatore *Random Forest*. In ogni nodo τ all'interno degli alberi binari T della foresta casuale, viene ricercata la divisione ottimale

utilizzando l'entropia i(τ) per misurare quanto bene una divisione di potenziale sta separando i campioni delle due classi in un particolare nodo.

I risultati ottenuti dall'analisi della *features importance* sul data set trovano una rappresentazione grafica nelle figure 3.22 e 3.23. Il primo grafico riporta le caratteristiche scelte dai metodi *LASSO* e *ANOVA* (Figura 3.22), il secondo espone la *features importance* ricavata attraverso il *Random Forest* (Figura 3.23).

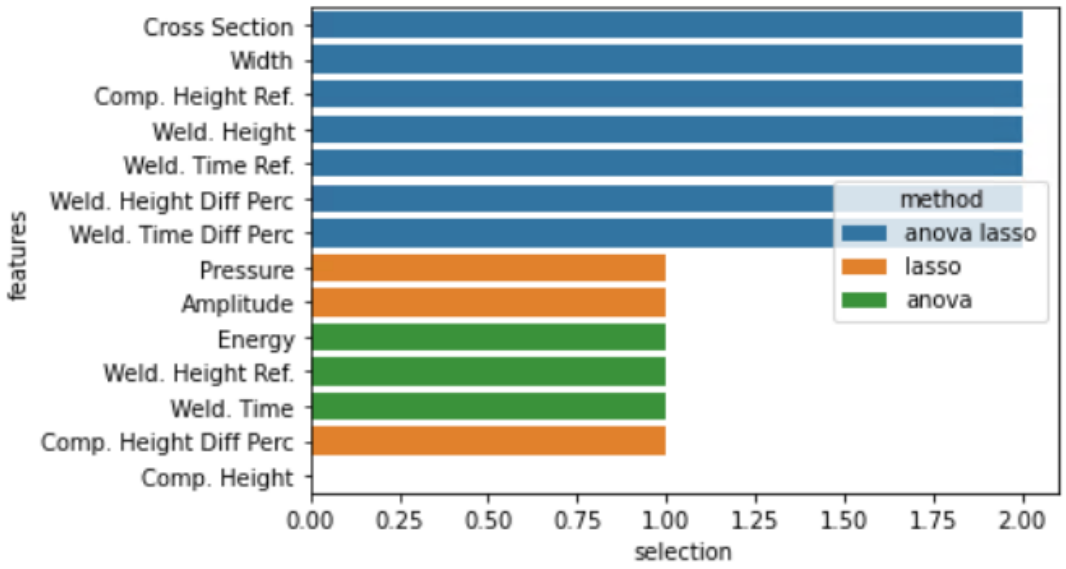

Figura 3.22 Selezione delle caratteristiche secondo LASSO e ANOVA, elaborazione personale con Matplotlib.

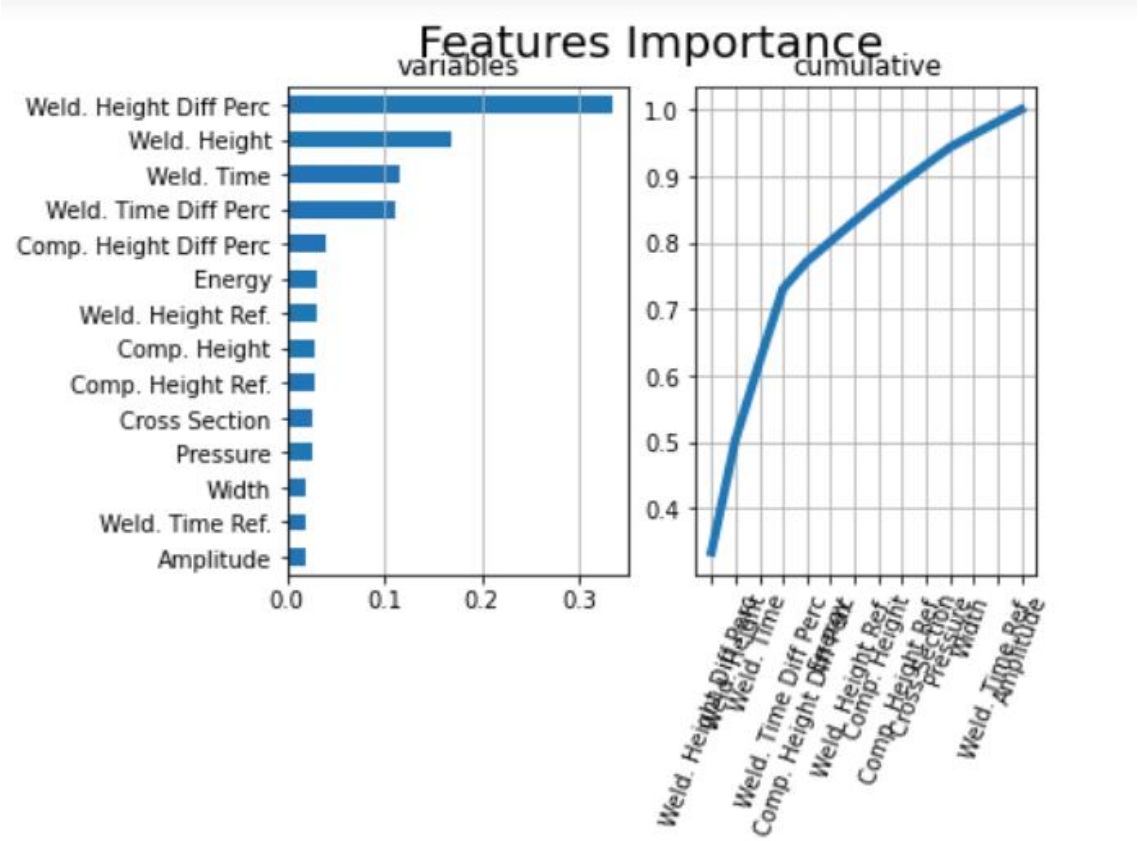

Figura 3.23 Analisi della features importance con classificatore Random Forest, elaborazione personale con Matplotlib.

A fronte dei risultati cui si è giunti, il data set da sottoporre al classificatore è stato limitato alle seguenti sette caratteristiche:

- *Cross Section*;
- *Comp. Height Diff Perc.*;
- *Weld. Time Diff Perc.*;
- *Weld. Height Diff Perc.*;
- *Weld. Height*;
- *Weld. Height Ref.*;

- *Weld. Time.*

3.3.4 *Learning*

Terminato il processo di preparazione dei dati, si è proceduto con l'analisi vera e propria.

Nell'elaborazione dell'analisi empirica ci si è serviti di due algoritmi: l'algoritmo di classificazione noto come *Random Forest*, a cui ha fatto seguito uno studio del data set attraverso la costruzione di una rete neurale artificiale (ANN) multistrato di tipo *feed-forward*.

L'obiettivo della classificazione è definire se i due modelli utilizzati, usando come input i valori relativi alle sette caratteristiche selezionate nella fase di *feature selection*, sono stati in grado di indicare se la saldatura è stata eseguita in modo corretto o meno sulla base di quanto contenuto nei file di log.

Al fine di dare un maggior livello di dettaglio sulla tipologia di errore che ha causato la non correttezza della saldatura, si è deciso di mappare in un primo momento la variabile *Error Number* con un valore binario, per poi proseguire con un *mapping* a sei valori.

Il training set utilizzato per la fase di addestramento si costituisce di un totale di 762.746 saldature e copre l'arco temprale che va da gennaio 2016 a dicembre dello stesso anno. Il set di dati utilizzato per la fase di test consta di 3.700.000 saldature eseguite da gennaio 2017 ad aprile 2022.

I risultati dell'analisi empirica sono trattati in modo approfondito nel capitolo successivo, vediamo ora come è stata affrontata la classificazione del data set con l'algoritmo *Random Forest* e la rete neurale.

3.3.4.1 Elaborazione con *Random Forest*

Come primo modello di elaborazione è stato scelto un classificatore *Random Forest*, famoso tra i modelli di tipologia *ensemble*. La classificazione *ensemble* è un approccio di *data mining* che utilizza più classificatori per fornire l'etichetta di classe ad un nuovo oggetto ancora non classificato. L'approccio su cui si basa, detto anche *Arbitrary Forest*, mette insieme alcuni alberi scelti in maniera casuale e unisce le loro previsioni facendone una media.

Si tratta di un approccio molto apprezzato dalla comunità di ricerca per le elevate prestazioni, che si accompagnano ad un ottimo grado di precisione. Il *Random Forest classifier* consta di una combinazione di alberi decisionali classificatori, dove ognuno dei quali viene generato utilizzando un vettore casuale campionato indipendentemente dal vettore di input e dove ogni albero esprime un voto unitario al fine di avere come risultato della classificazione di un vettore di input quella con il maggior numero di voti [98]. Il *Random Forest* è dunque costituito da alberi costruiti in modo pseudo - casuale mantenendo un'elevata accuratezza dei dati di addestramento e migliorando la precisione nella generalizzazione all'aumentare della complessità.

Dal momento che la previsione viene calcolata in base al voto della maggioranza degli alberi decisionali del modello, la funzione di discriminazione può essere definita come segue:

$$H(x) = argmax_y \sum_{i=1}^{k} I(h_i(X, \theta_k) = Y)$$

Dove I indica la funzione indicatore, h è la decisione del singolo albero e Y è la variabile di output. $Argmax_y$ indica il valore Y quando viene massimizzato $\sum_{i=1}^{k} I(h_i(X, \theta_k) = Y)$.

Per ogni nuovo set di addestramento generato per far crescere l'albero, un terzo dei dati viene scelto in maniera casuale ed escluso: sono i campioni chiamati *out-of-bag* (OOB). I dati rimanenti, denominati *in-the-bag*, vengono usati per costruire l'albero decisionale. Si noti che i campioni OOB possono essere utilizzati per valutare le performance del modello [99].

È tramite il processo di *bagging* che il classificatore è in grado di ridurre l'errore di generalizzazione e la correlazione tra gli alberi. Questo risultato viene raggiunto facendo crescere l'albero utilizzando la miglior suddivisione di un sottoinsieme casuale di caratteristiche in input invece di utilizzare le variabili del migliore *split*. Ciò permette di generare gli alberi senza effettuare il *pruning*, riducendo di molto le tempistiche del modello. È questo uno dei grossi vantaggi dei classificatori *Random Forest* rispetto agli alberi decisionali: anche senza la potatura degli alberi l'errore di generalizzazione converge ed il problema dell'*overfitting* si risolve per «la legge dei grandi numeri».

Come sottolineato da Breiman "la casualità utilizzata nella costruzione degli alberi deve mirare a una bassa correlazione ρ, mantenendo comunque una forza ragionevole" [100]. Questo compromesso può essere raggiunto mediante l'utilizzo di iper-parametri.

Il primo parametro considerato è il *min_samples_split*, che indica il numero di campioni necessari per dividere un nodo interno. Un valore di questo parametro inferiore porta ad avere alberi decisionali più profondi, con un numero di *split* necessari per arrivare ai nodi terminali maggiore.

In numerosi pacchetti software standard il valore predefinito per la classificazione è 1, per la regressione 5. Come dimostrato prima da Díaz-Uriarte e De Andres nel 2006 in *Gene selection and classification of microarray data using random fores* [101] e poi nel 2011 da Goldstein in *Random forests for genetic association studies*, il parametro svolge una funzione apprezzabile, conducendo a buoni risultati [102]. Nel 2004 Segal ha provato come un aumento della quantità di rumore porti ad avere una dimensione dei nodi ottimale più

alta [103]. Più nel dettaglio, specialmente in caso data set di grandi dimensioni, i tempi di elaborazione necessari diminuiscono in modo esponenziale all'aumentare del valore di questo parametro. Quindi, l'obiettivo è trovare il giusto compromesso tra buone prestazioni e performance ragionevoli.

Il secondo parametro preso in considerazione è il *max_depth* che permette di controllare la profondità massima degli alberi decisionali che compongono il classificatore *Random Forest*. Il valore di default è *None* e sta a significare che la profondità degli alberi aumenta finché tutti i nodi terminali sono puri o fino a quando tutti questi contengono meno campioni del parametro *min_samples_split*.

Il parametro *n_estimators* ha lo scopo di impostare il numero di alberi decisionali che comporranno il classificatore. Se da un lato occorre avere un numero sufficiente di alberi per garantire predizioni più chiare e quindi convergenza, dall'altro va tenuto in conto il tempo di esecuzione necessario (maggiore per un numero di alberi decisionali più elevato), anche se, grazie alla programmazione concorrente e all'indipendenza degli alberi, è possibile addestrarli in parallelo, limitandone la crescita. Una particolarità, dimostrata empiricamente da Oshiro prima e da Probst e Boulesteix poi, è che per data set molto grandi, la curva di convergenza cresce molto in fretta per i primi cento alberi, per poi tendere a stabilizzarsi [104].

Il numero di caratteristiche da considerare quando si cerca il miglior *split* nei nodi è settabile attraverso il parametro *max_features*. Valori comuni per questo parametro sono *sqrt*, *log2* e *None*, che indicano rispettivamente un numero di caratteristiche pari alla radice quadrata del numero totale di caratteristiche, al logaritmo in base 2 di questo, oppure uguale al numero totale di caratteristiche.

Il parametro *criterion* è stato impiegato per impostare la funzione usata a valutare la qualità della suddivisione. I più noti sono Gini e l'entropia. L'impurità di Gini indica la probabilità di

classificare erroneamente un'osservazione. Più basso è il Gini, migliore è la divisione e con essa minore è la probabilità di errata classificazione. La formula dell'Indice di Gini è:

$$G = \sum_{i=1}^{n} p(i) * (1 - p(i))$$

dove n è il numero di classi e p(i) la probabilità di scegliere un elemento della classe i.
Durante la fase di addestramento di un albero decisionale, la divisione migliore viene scelta massimizzando il cosiddetto *Gini Gain*, calcolato sottraendo le impurità pesate dei rami dall'impurità originale.
L'entropia è una metrica che misura il disordine, l'impurità e quindi l'incertezza di un gruppo di osservazioni. Ne consegue che suddivisioni migliori avvengono per valori di entropia minori. La formula dell'entropia è:

$$E = -\sum_{i=1}^{n} p(i) \log_2 p(i)$$

dove n è pari al numero di classi e p(i) la probabilità di scegliere un elemento della classe i. Il valore massimo si raggiunge quando la probabilità delle classi è uguali, mentre un nodo puro è caratterizzato dal valore minimo, ossia 0.

Per trovare la combinazione di parametri ottimale si è ricorsi al *Random Search*, un algoritmo utilizzato per affrontare problemi generici di ottimizzazione, descrivibile attraverso la formula:

$$min_{x \in S} f(x)$$

dove x è un vettore di n variabili decisionali, S è una regione ammissibile n-dimensionale non vuota e f è una funzione a valori reali definita su S. L'obiettivo è trovare un valore per x contenuto in S che minimizza f. L'algoritmo è descrivibile attraverso una sequenza di iterazioni $\{X_i\}$ con le iterazioni i = 0,1, … che possono dipendere da punti precedenti e da parametri algoritmici. Le combinazioni di variabili utilizzate nelle iterazioni sono casuali, riflettendo la natura probabilistica dell'algoritmo:

- step 0, dove vengono inizializzati i parametri θ_0, i punti iniziali $X_0 \subset S$ e l'indice di iterazione i = 0;
- step 1, dove vengono generati un insieme di punti candidati $Vi_{+1} \subset S$ basandosi su un apposito generatore e la relativa distribuzione campionaria;
- step 2, dove viene aggiornato X_{i+1} in base ai punti candidati V_{i+1}, alle iterazioni precedenti ed ai parametri dell'algoritmo. Assieme a questo vengono anche aggiornati i parametri dell'algoritmo $\theta\ i_{+1}$;
- step 3, dove se viene soddisfatto uno *stopping criterion*, allora si interrompe. Altrimenti incrementa l'indice di iterazione i e torna allo step 1 [105].

Per l'applicazione dell'algoritmo *Random Search* è stata utilizzata la classe *RandomizedSearchCV* del pacchetto *sklearn.model_selection*. La classe, implementando una metodologia *fit and score*, esegue una ricerca casuale sugli iper-parametri di un modello di *Machine Learning* e cerca di ottimizzarli utilizzando la validazione incrociata. La *cross-validation* è un metodo statistico di valutazione e confronto usato dagli algoritmi di apprendimento che funziona dividendo i dati in due porzioni: una usata per la fase di apprendimento, l'altra per la convalida del modello.

Secondo il metodo della *cross-validation* i set di addestramento e convalida devono incrociarsi in round successivi in modo tale che ogni punto dati abbia la possibilità di essere convalidato. La forma di base della convalida incrociata è la *k-fold cross-validation*, dove i dati vengono prima di tutto suddivisi in k segmenti o porzioni di dimensioni uguali. In

seguito, vengono eseguite k iterazioni di traning e validazione in modo tale che in ogni iterazione venga presa in considerazione una porzione differente per l'operazione di validazione, mentre le restanti k-1 vengono utilizzate per l'addestramento.

La classe *RandomizedSearchCV* utilizzata per l'applicazione dell'algoritmo, a differenza della *GridSearchCV*, non prova tutte le combinazioni di valori ma si concentra su di un numero fisso imputato con il parametro *n_iter*.

Per ognuno dei parametri sopra indicati è stato passato all'algoritmo l'elenco di possibili valori come segue:

- *Criterion*: [Entropy, Gini];
- *N_estimators*: [10, 25, 50, 75, 100, 150];
- *Max_depth*: [None, 1, 3, 5, 6];
- *Min_samples_split*: [1, 2, 4, 5, 7];
- *Max_features*: [auto, sqrt, log2].

Nella classificazione di tipo binario (i.e. con *mapping* binario della variabile *Error Nr.*) l'esecuzione della *RandomizedSearchCV* con i parametri sopra indicati, *n_iter* impostato a 25 e in input il training set costituito dai dati sulle saldature dell'anno 2016 ha restituito un modello *Random Forest* addestrato con i seguenti parametri, considerati la combinazione migliore dall'algoritmo:

- *Criterion*: Gini;
- *N_estimators*: 25;
- *Max_depth*: 5;
- *Min_samples_split*: 4;
- *Max_features*: sqrt.

Nella classificazione di tipo multi-classe (i.e. con *mapping* non binario della variabile *Error Nr.*) l'esecuzione della *RandomizedSearchCV* ha restituito un modello *Random Forest* addestrato con i seguenti parametri, considerati la combinazione migliore dall'algoritmo:

- *Criterion*: Entropy;
- *N_estimators*: 100;
- *Max_depth*: None;
- *Min_samples_split*: 7;
- *Max_features*: auto.

3.3.4.2 Elaborazione con rete neurale

Alla classificazione del data set con l'algoritmo di *Random Forest* ha fatto seguito un'analisi attraverso la costruzione di una rete neurale artificiale (ANN) multistrato di tipo *feed-forward*.

La rete neurale profonda di tipo *feed-forward* è anche conosciuta come *multilayer perceptron* (MLP) ed è composta da almeno tre strati di neuroni artificiali: uno di input, uno di output e uno o più strati nascosti. Il flusso di informazioni che compongono questo modello è unidirezionale (dall'input verso l'output) e i neuroni di uno strato possono essere interconnessi solo con neuroni dello strato successivo.

Si tratta di un modello di *Deep Learning* che ha da tempo trovato ampi spazi d'impiego nella risoluzione di numerosi problemi ingegneristici di intelligenza artificiale.

Mentre il numero di neuroni dei livelli di input (i.e. numero di caratteristiche del problema) e di output (i.e. numero di classi che l'output può assumere) è fisso, la scelta del numero di livelli nascosti e dei relativi neuroni è a carico del programmatore e concorre a definire l'architettura della rete. L'obiettivo è quindi comporre l'architettura in modo tale da ottimizzare la risoluzione del problema, oltre ad avere una buona generalizzazione del problema di classificazione affrontato.

Le connessioni tra i neuroni sono caratterizzate da pesi w. Assumendo come $w_{k,j}^{i}$ il peso che caratterizza la connessione tra il neurone artificiale k del livello nascosto i e il neurone j

del livello i + 1, come X = (x_0, x_1, …, x_{n0}) i neuroni del livello di input e f la funzione di attivazione, si può definire l'output dei neuroni h_i^j del primo livello nascosto come il risultato della seguente funzione:

$$h_i^j = f\left(\sum_{k=1}^{n_{i-1}} w_{k,j}^0 x_k\right) \quad j = 1, \dots, n$$

Per i successivi strati nascosti avremo come output dei neuroni h_i^j

$$h_i^j = f\left(\sum_{k=1}^{n_{i-1}} w_{k,j}^{i-1} h_{i-1}^k\right) \quad j = 1, \dots, n_i$$

Infine, assumendo che la rete abbia N strati nascosti, l'output di un neurone dello strato di output è pari a:

$$y_i = f\left(\sum_{k=1}^{n_N} w_{k,j}^N h_N^k\right) \quad j = 1, \dots, n_i$$

Dal momento che l'output della rete neurale è Y = (y_1, …, y_j, …, y_{N+1}) = F (W, X), si può affermare che l'output della rete è funzione degli input X e dei pesi delle connessioni all'interno della rete neurale.

Si definisce fase di apprendimento di un MLP quel processo dove, sfruttando la *back-propagation* dell'errore legato alla differenza dell'output fornito dalla rete e l'output aspettato, vengono modificati i pesi con lo scopo di minimizzare l'errore [106].

Le variabili che concorrono alla definizione dell'architettura della rete sono la selezione del numero di strati nascosti e la selezione del numero di neuroni artificiali per strato nascosto. Si noti che, sebbene questi livelli non interagiscano direttamente con l'esterno, possono influenzare notevolmente le prestazioni del modello.

Molti dei problemi affrontati con tecniche di *Deep Learning* (simili a quello trattato) mostrano come l'utilizzo di un solo strato nascosto sia sufficiente. L'utilizzo di due livelli nascosti raramente migliora il modello, potrebbe anzi rivelarsi controproducente e portare al verificarsi di un maggior rischio di convergenza ai minimi locali. Per il problema affrontato è sufficiente un *hidden layer*.

La scelta del numero di neuroni all'interno dello strato nascosto deve essere anch'essa ponderata. L'utilizzo di un numero di neuroni non sufficiente potrebbe portare al verificarsi di *underfitting*, ovvero quel problema di apprendimento che si verifica quando la classificazione si basa su pochi parametri soffrendo di un'eccessiva discrepanza (*high bias*). Parimenti, l'utilizzo di troppi neuroni negli strati nascosti potrebbe causare *overfitting*, portando ad un'elevata variabilità della classificazione causata da un modello troppo complesso ed eccessivamente sensibile ai dati di training (*high variance*).[35]

Esistono molti metodi basati su regole empiriche per determinare il numero corretto di neuroni da utilizzare negli strati nascosti:

- il numero di neuroni nascosti dovrebbe essere compreso tra la dimensione del livello di input e la dimensione del livello di output;
- il numero di neuroni nascosti dovrebbe essere pari ai due terzi della somma tra il numero di neuroni dello strato di input e quelli dello strato di output;

[35]In una situazione di *overfitting* la rete neurale possiede un'eccessiva capacità di processare informazioni, tanto che il numero limitato di informazioni contenute nel training set non è sufficiente per addestrare tutti i neuroni degli strati artificiali.

- il numero di neuroni nascosti dovrebbe essere inferiore a due volte la dimensione del livello di input.

Queste regole offrono un valido spunto nel processo di selezione del numero di neuroni, il cui valore ottimale dipende da una serie di variabili come il numero di unità negli strati di input e output, il numero di casi per l'addestramento, la quantità di rumore tra i valori di output, la complessità della funzione o della classificazione da far apprendere al modello, l'algoritmo di apprendimento scelto. In definitiva, la selezione dell'architettura della rete neurale si riduce ad un processo di tentativi ed errori.

Nel modello binario (i.e. con *mapping* binario della variabile *Error Nr.*) si è deciso di strutturare il livello di input con sette neuroni, pari al numero di caratteristiche del data frame. Per quanto riguarda il livello di output si è deciso di utilizzare due neuroni e la funzione di attivazione nota come *softmax*. Si noti che, trattandosi di una classificazione di tipo binaria, nella definizione del numero di neuroni per lo strato di output, è stato possibile scegliere tra due opzioni (indifferenti ai fini delle prestazioni del modello):

- utilizzare un solo neurone e il sigmoide come funzione di attivazione;
- utilizzare due neuroni e la *softmax* funzione di attivazione.

Per definire il numero di neuroni dello strato nascosto sono state valutate le seguenti opzioni:

- sette neuroni, pari a due terzi della somma dei neuroni negli strati di input e output;
- sei neuroni;
- cinque neuroni.

È emerso che, in termini di accuratezza, nessuna delle tre ipotesi fosse preferibile all'altra (tutte quante hanno dimostrato di funzionare correttamente), ma che dal punto di vista della funzione di perdita, il modello con cinque neuroni fosse quello da prediligere. Si è dunque deciso di utilizzare uno strato nascosto con cinque neuroni.

Per il modello multiclasse (i.e. con *mapping* non binario della variabile *Error Nr.*) si è deciso di utilizzare sette neuroni per lo strato di input, sei neuroni con la funzione di attivazione *softmax* per quello di output e sette neuroni per quello nascosto.

Una volta definita l'architettura del MLP, occorre selezionare gli iper-parametri, ovvero la *batch size* ed il numero di epoche da usare per addestrare il modello. Il primo indica il numero di campioni nel training set da utilizzare per il calcolo del gradiente, il secondo il numero di volte in cui l'algoritmo di apprendimento lavora sull'intero set di addestramento.

Le possibili opzioni per la scelta della *batch size*, che danno la possibilità al programmatore di scegliere il giusto compromesso tra velocità ed accuratezza, sono:

- *Full-Batch Gradient Descent;*
- *Mini-Batch Gradient Descent;*
- *Stochastic Gradient Descent.*

Nel *Full-Batch Gradient Descent* il batch comprende tutti i campioni del training set e per completare un'epoca è necessaria una iterazione. Nel caso in cui si abbia a che fare con set di dati utilizzati per l'addestramento composti da un numero elevato di campioni, la fase di aggiornamento potrebbe risultare molto onerosa, dal momento che il gradiente deve essere valutato per ogni somma.

Nel *Mini-Batch Gradient Descent* il calcolo del gradiente viene eseguito prima di aggiornare il parametro solo per un sottoinsieme dei campioni di addestramento. Da ciò deriva la necessità di avere un numero di iterazioni per completare un'epoca pari al numero di sottoinsiemi necessari per completare il calcolo per tutti i campioni del training set.

Lo *Stochastic Gradient Descent* (SGD) è considerabile come un caso particolare del *Mini-Batch Gradient Descent*, dove il sottoinsieme è composto da un solo campione. Per ognuno dei campioni viene calcolato il gradiente e viene aggiornato il parametro, facendo sì che

siano necessarie n iterazioni per completare un'epoca, con n pari al numero di campioni del training set.

Il programmatore deve quindi scegliere il giusto compromesso tra velocità di aggiornamento del parametro, direzione di convergenza e capacità di memoria. La discesa del gradiente in modalità *Full-Batch* tende a convergere più lentamente degli altri due poiché il gradiente deve essere aggiornato per tutti i campioni prima dell'aggiornamento. Per la direzione di convergenza, invece, maggiore è la quantità di informazioni utilizzata nella fase di aggiornamento, più diretta sarà la convergenza della discesa del gradiente. In questo caso il *Full-Batch* converge in maniera più diretta rispetto al *Mini-Batch* e all'*SGD*. La quantità di memoria utilizzata per il calcolo è direttamente proporzionale alla grandezza del *batch* utilizzato: maggiore sarà il *batch*, maggiore sarà la capacità di memoria richiesta.

Generalmente si opta per un *Mini-Batch*, in quanto il *Full-Batch*, soprattutto per training set di grandi dimensioni come quello di cui si dispone, è molto oneroso in termini di quantità di memoria richiesta e lento come velocità di aggiornamento del parametro, mentre l'SGD ha una convergenza troppo fluttuata.

La scelta del numero di epoche deve anch'essa essere fatta in maniera ponderata, dal momento che un numero troppo esiguo potrebbe portare la rete a non apprendere in modo adeguato, mentre un numero troppo elevato potrebbe causare *overfitting*, portando la rete ad immagazzinare così tante informazioni da adattarsi eccessivamente al set di dati e non riuscire a classificare correttamente altri data set.

Per il *batch size* si è scelto di utilizzare 64 campioni, mentre il numero di epoche da ventuno (pari a tre volte il numero delle colonne) è stato abbassato a 18 per evitare *overfitting*, dato che nelle ultime tre epoche non si sono verificati margini di miglioramento.

Gli ultimi due parametri impostati sono stati la funzione di perdita e l'algoritmo di ottimizzazione.

La funzione di perdita scelta è stata la *Categorical Crossentropy*, che calcola la perdita di *crossentropy* tra le etichette e le previsioni ed è consigliata quando sono presenti due o più classi di etichette [107].

Per il secondo iper-parametro è stato scelto *Adam*, un metodo che permette di eseguire un'ottimizzazione stocastica efficiente, che richiede solo gradienti del primo ordine e poca memoria. Il metodo calcola i *learning rate* adattivi individuali per parametri diversi dalle stime dei primi e secondi momenti dei gradienti. Il nome deriva dall'acronimo di *adaptive moment estimation* e ne indica l'elaborazione, ovvero la stima del momento adattativo. Questo metodo, oltre ad essere semplice da implementare ed efficiente dal punto di vista computazionale, è invariante al ridimensionamento diagonale dei gradienti ed è anche adatto per problemi di grandi dimensioni in termini di dati e/o parametri [108].

I risultati hanno dimostrato che entrambi gli algoritmi sono stati in grado di classificare il data set a disposizione in modo corretto.

Essendo gli algoritmi utilizzati per l'analisi di tipo supervisionato, la «correttezza» della classificazione ottenuta va intesa in termini di accuratezza rispetto ai dati presenti nei file di log, riportanti le informazioni sul livello di qualità della saldatura.

La percentuale di accuratezza di entrambi i modelli si aggira intorno al 100% in caso di *mapping* binario della variabile *Error Nr.*, al

E' stata dunque dimostrata una sostanziale coincidenza tra i dati dei file di log e l'output del processo di classificazione, ma si noti un aspetto di fondamentale importanza: i file di log contenevano al loro interno le *feature Error Nr* e *Error Text*, indicative della qualità della saldatura e presenti nel file di log grazie all'intervento manuale dell'operatore al termine del processo.

L'argomentazione dettagliata dei risultati ottenuti viene trattata nel capitolo successivo.

Capitolo 4

L’analisi dei risultati

Il processo di classificazione aiuta con la categorizzazione del set di dati in diverse classi. Un modello di apprendimento automatico permette di: inquadrare il problema, raccogliere i dati, aggiungere le variabili, addestrare il modello e misurarne le prestazioni. La parte forse più importante di qualsiasi modello di *Machine Learning* corrisponde all’ultimo termine, ovvero all’avere contezza del livello di bontà ed accuratezza del modello elaborato. Per avere informazioni a proposito della bontà del modello è necessario disporre di determinati parametri che ne definiscano la qualità.

Il capitolo è dedicato all’analisi delle prestazioni dei due modelli utilizzati per mezzo della matrice di confusione e degli indici che da essa emanano. La tecnica della matrice di confusione aiuta a misurare le prestazioni di un classificatore nell’ambito dell’apprendimento automatico, fornendo una visione olistica delle prestazioni del modello.

4.1 Elaborazione del test set

Per la fase di test e la contestuale analisi dei modelli addestrati sono stati utilizzati i file di log relativi alle saldature eseguite da gennaio 2017 ad aprile 2022. Come ampiamente anticipato, si tratta di file emessi dalle macchine saldatrici ad ultrasuoni Minic

III e depositati sul file system locale. Qui vi rimangono per i primi due mesi, fino a quando una procedura schedulata li sposta in una cartella di backup di un apposito server e li organizza in base alla macchina di provenienza.

I file di log sono da considerarsi l'output del processo di saldatura ad ultrasuoni operato dalle macchine Minic III della Schunk Sonosystem e si costituiscono di un insieme di valori relativi a parametri impostati dall'operatore in preparazione della fase di saldatura (e mantenuti durante il ciclo di produzione) e di rilevazioni eseguite dai sensori della macchina prima, durante e dopo la saldatura.

Il training set utilizzato per la fase di addestramento si costituisce di un totale di 762.746 saldature e copre l'arco temprale che va da gennaio 2016 a dicembre dello stesso anno. Il set di dati utilizzato per la fase di test consta 3.700.000 saldature eseguite da gennaio 2017 ad aprile 2022.

La preparazione del test set, parimenti al data set di addestramento, ha previsto i seguenti passaggi (per i dettagli sull'esecuzione di ogni step si rimanda al capitolo "L'analisi empirica"):

- *data cleaning*, ovvero sostituzione dei valori vuoti o inconsistenti con la costante 0 ed eliminazione delle saldature di prova (i.e. parametro *Mode* = *Setup*);
- *feature scaling*, ovvero standardizzazione delle caratteristiche con il *min max scaler* per quelle di tipo numerico con distribuzione continua e con il *label encoder* per quelle numeriche o alfanumeriche;
- *feature extraction*, ovvero l'aggiunta dei tre campi calcolati *Compacting Height Diff Perc*, Weld*. Height Diff Perc* e *Weld. Time Diff Perc*;
- *feature selection*, ovvero il data set da sottoporre al classificatore è stato limitato alle sette caratteristiche Cross Section, Comp. Height Diff Perc.; Weld. Time Diff Perc.; Weld. Height Diff Perc.; Weld. Height; Weld. Height Ref.; Weld. Time.

In seguito, i dati di test sono stati organizzati in *data frame pandas* differenti. Per l’analisi delle performance dei due modelli è stato creato un data set contenente tutte le saldature da gennaio 2017 ad aprile 2022, mentre per lo studio sull’andamento e sulla qualità delle saldature si è deciso di dividere lo stesso data set in data set più piccoli (con dati semestrali ed annuali).
I set di dati così originati sono stati utilizzati come input del classificatore *Random Forest* e del *Multi-Layer Perceptron* ad uno strato nascosto, al fine di paragonare i risultati ottenuti nella fase di test con quella di addestramento.

4.2 Come valutare un modello di apprendimento automatico

L'apprendimento automatico è un campo dell'Intelligenza Artificiale che si occupa della costruzione di sistemi informatici adattativi, ovvero in grado di migliorare le proprie prestazioni sulla base dell’esperienza effettuata. Il *Machine Learning* è la disciplina che studia la generalizzazione, la costruzione e l’analisi di algoritmi in grado di generalizzare.
La classificazione è una delle pratiche più comuni dell'apprendimento automatico ed ha la particolarità di avere classi, intese come funzioni target, in qualità di variabili di tipo discreto. Il compito è quello di realizzare un modello sulla base delle caratteristiche degli oggetti la cui etichetta è nota in anticipo, mediante il quale verrà eseguita la classificazione di nuovi oggetti. In un problema di classificazione il numero delle classi è noto in anticipo e limitato [109].

Siccome per lo stesso problema e lo stesso insieme di dati di addestramento possono essere prodotti modelli differenti, è emersa la necessità di avere strumenti utili a valutarne la qualità. La corretta valutazione dei modelli appresi è una delle questioni più importanti nel riconoscimento dei modelli.

La valutazione di un modello viene di solito fatta sulla base di una serie di metriche. Le metriche comunemente utilizzate per la valutazione di problemi di classificazione sono raggruppabili in tre famiglie:

- metriche basate su una soglia e su una comprensione qualitativa dell'errore come l'accuratezza, la precisione della macro-media (sia aritmetica che geometrica), la F-*score* o F-*measure* media e la statistica Kappa o indice di concordanza.

  Queste misure vengono utilizzate quando l'obiettivo è ridurre al minimo il numero di errori. Si rivelano più appropriate per set di dati bilanciati o non bilanciati, per il rilevamento di segnali o guasti e per attività di recupero di informazioni;
- metriche basate su una comprensione probabilistica dell'errore come la misurazione della deviazione dalla probabilità reale come media assoluta dell'errore, l'errore quadratico medio (punteggio Brier), la funzione di perdita o *Log Loss* (entropia incrociata), il tasso di probabilità (rango) e la calibrazione.

  Queste misure sono particolarmente utili quando si vuole valutare l'affidabilità dei classificatori avendo una misura del fallimento e il grado di probabilità con cui è stata selezionata la classe errata;
- metriche basate sulla bontà della classificazione eseguita dal modello sugli esempi come l'AUC, che per due classi equivale alla statistica di Mann–Whitney–Wilcoxon ed è strettamente correlata al concetto di separabilità.

  Si tratta di metriche ampiamente utilizzate in campi come la progettazione di campagne di mailing, la gestione delle relazioni con i clienti (CRM), la creazione di sistemi di raccomandazione, il rilevamento delle frodi, i filtri antispam, etc. Situazioni dove i classificatori vengono utilizzati per selezionare le migliori n istanze di un insieme di dati o quando una buona distinzione delle classi è fondamentale [110].

Il problema affrontato dai due modelli utilizzati è una classificazione di tipo binario, cui è stata paragonata anche una multi-classe, dove l'obiettivo è constatare se la saldatura eseguita dalla macchina saldatrice ad ultrasuoni sia stata fatta correttamente o meno. Essendo gli algoritmi utilizzati di tipo supervisionato, la «correttezza» della classificazione ottenuta va intesa in termini di accuratezza rispetto ai dati presenti nei file di log, riportanti le informazioni sul livello di qualità della saldatura. Questa prima analisi è stata sviluppata attraverso una matrice di confusione e i relativi indici, per poi soffermarsi sulle evidenze emerse dalla *ROC curve* e dall'*AUC curve*. Si è deciso di mettere a confronto i due classificatori nel contesto di una classificazione binaria, per poi valutare i risultati in caso di *mapping* multi-classe dell'etichetta di classe.

4.2.1 Matrice di confusione con output binario

Uno dei concetti chiave nelle prestazioni di classificazione è la matrice di confusione (matrice di errore AKA), che è una visualizzazione tabulare delle previsioni del modello rispetto alle etichette di verità fondamentale. Ogni riga della matrice di confusione rappresenta le istanze in una classe prevista e ogni colonna rappresenta le istanze in una classe effettiva.

Si tratta di una tabella a doppia entrata, che svolge la funzione di raccogliere e mettere a confronto i dettagli delle classificazioni previste ed effettive eseguite da un algoritmo di classificazione per valutarne le prestazioni.

Il paragrafo espone i risultati ottenuti dai modelli *Random Forest* e *MLP* in caso di *mapping* binario della variabile *Error Nr*.

In caso di classificatori binari la matrice prevede la costruzione sulla base di due classi (Tabella 4.4):

		Classificazione predetta	
		Saldatura errata (0)	Saldatura corretta (1)
Classificazione effettiva	Saldatura errata (0)	TN	FP
	Saldatura corretta (1)	FN	TP

Tabella 4.4 Matrice di confusione con output binario, elaborazione personale.

Dove, in base al contesto di utilizzo:

- TN rappresenta il numero di previsioni corrette dove la saldatura è stata predetta come errata ed effettivamente si è rivelata tale;
- FP rappresenta il numero di previsioni errate dove la saldatura è stata predetta come corretta, mentre questa è stata effettuata in modo errato;
- FN rappresenta il numero di previsioni errate dove la saldatura è stata predetta come errata, mentre questa è stata effettuata correttamente;
- TP rappresenta il numero di previsioni corrette dove la saldatura è stata predetta come corretta ed effettivamente si è rivelata tale.

Partendo dalla matrice di confusione è possibile ricavare degli indici in grado di fornire informazioni relative all'accuratezza dei modelli [111].

Per ottenere sia la matrice di confusione dei modelli *Random Forest* e MLP è stata utilizzata la funzione *confusion_matrix* della libreria *metrics* del pacchetto *sci-kit learn* di Python.

La funzione riceve come parametri due *array* monodimensionali con il valore della variabile di output. In questo caso i possibili valori sono due:

- 0 per la saldatura errata;
- 1 per la saldatura corretta.

La funzione restituisce quindi una matrice di confusione 2x2 C_{ij}, dove i è l'etichetta effettiva e j quella prodotta.

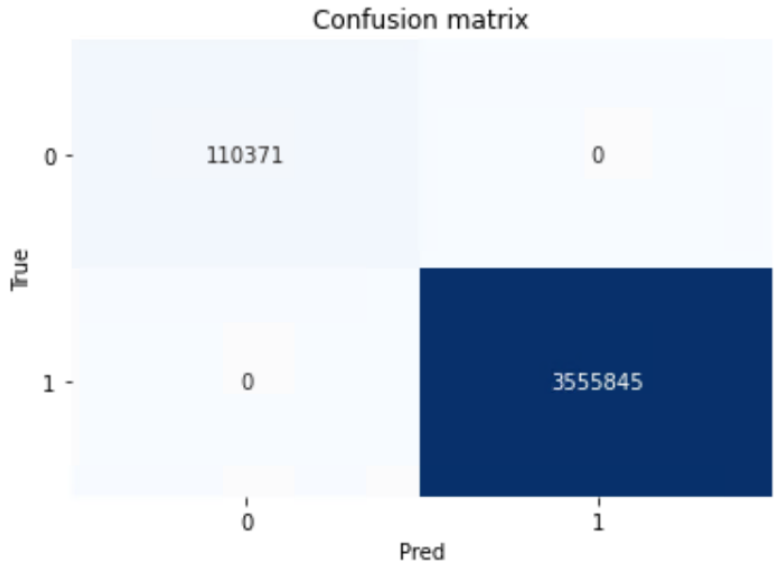

Figura 4.24 Matrice di confusione con classificatore *Random Forest* e output binario, elaborazione personale con scikit-learn.

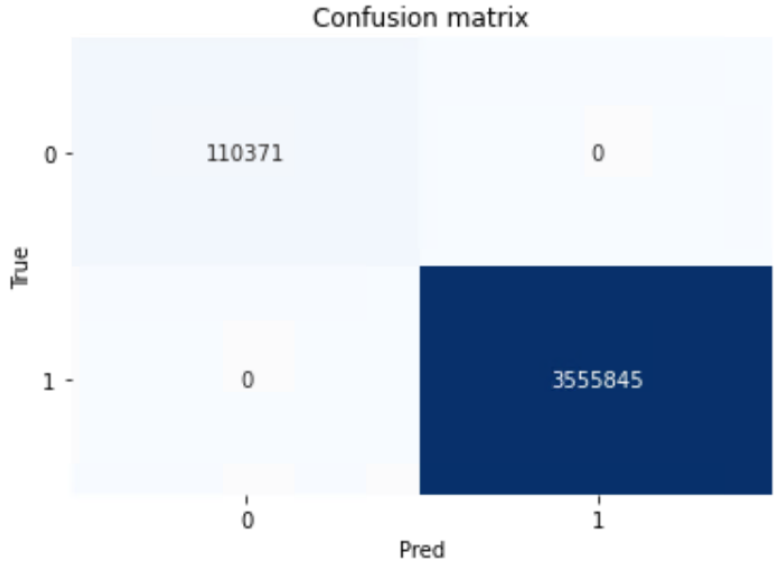

Figura 4.25 Matrice di confusione con classificatore *Multi-Layer Perceptron* e output binario, elaborazione personale con scikit-learn.

Le Figure 4.24 e 4.25 mostrano rispettivamente la matrice di confusione relativa al modello *Random Forest* e alla rete neurale. La metrica ottenuta è identica per i due algoritmi e riporta:

- 110.371 campioni veri negativi, ovvero saldature effettivamente errate e predette come tali;
- 0 campioni *false positive*, ovvero saldature effettivamente negative e classificate come positive;
- 0 campioni false negative, ovvero saldature effettivamente positive ma predette come positive;
- 3.555.845 campioni veri positivi, corrispondenti a saldature effettivamente corrette e classificate come corrette dal modello.

4.2.1.1 Accuratezza

Nella valutazione dei modelli di classificazione uno dei concetti base è il confronto tra correttezza ed errori. Se l'applicazione dei modelli di classificazione ad un caso selezionato porta alla previsione di classi differenti dagli esempi di etichette effettive, ci si trova di fronte ad un errore nella classificazione. Si noti che uno dei limiti di questa metrica risiede nel fatto che venga dato lo stesso peso a tutti i tipi di errore (e.g. falso positivo e falso negativo).

Questo approccio si basa sull'accuratezza come misura per valutare la qualità della modello classificatore e fa affidamento ad una misura che può essere definita come il rapporto tra il numero di esempi correttamente classificati in rapporto al numero totale di esempi classificati:

$$Accuratezza = \frac{numero\ di\ esempi\ classificati\ correttamente}{numero\ totale\ di\ casi}$$

Nonostante si tratti di uno dei metodi di valutazione più comuni per l'analisi della performance di modelli di classificazione, l'accuratezza fornisce una valutazione limitata e viziata da due variabili: non viene fatta distinzione tra tipologia di errori differenti e la forte dipendenza dalla distribuzione delle classi nei set di dati [112].
L'accuratezza (AC) è pari al rapporto tra il numero totale di previsioni eseguite correttamente e l numero totale di previsioni effettuate:

$$AC = \frac{TP + TN}{TP\ +\ FP\ +\ TN + FN}$$

Per l'elaborazione dell'AC ci si è serviti della libreria *metrics* del pacchetto *sci-kit learn*. La funzione utilizzata è l'*accuracy_score*. Questa, ricevendo come parametri due *array* monodimensionali con i valori aspettati e quelli predetti, restituisce una percentuale che indica la quantità di campioni classificati correttamente.
Per entrambi i classificatori, *Random Forest* e *Multi Layer Perceptron*, il valore restituito dalla funzione è pari a 1 e indica che, come già si evince guardando le due matrici di confusione, tutti i campioni del data set di test sono stati classificati correttamente.

4.2.1.2 *Recall*

Il *Recall*, chiamata in psicologia Richiamo o Sensibilità, è la proporzione di casi effettivamente positivi che sono previsti correttamente come positivi. Misura il tasso di copertura dei casi realmente positivi da parte della regola +P dei casi previsti come positivi.

La caratteristica principale è quella di riflettere quanti dei casi rilevanti il +P vengono evidenziati dalla regola.
Si tratta di un indice che ha riscosso poco successo nel campo del *Machine Learning* e della linguistica computazionale, anche se importanti riscontri ha avuto nel cosiddetto *Word Alignment.*[36] Nell'ambito nella curva di ROC questo indice è anche conosciuto come TPR o *true positive rate* ed è matematicamente definito come il numero di veri positivi (TP) diviso per il numero di veri positivi (TP) più il numero di falsi negativi (FN) [113]:

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

Il *recall* è stato calcolato mediante la funzione *recall_score*, messa a disposizione dalla libreria *sklearn.metrics*. Sulla base dei due *array* monodimensionali con le etichette reali delle saldature e con quelle ottenute dai due classificatori la funzione ha restituito il valore 1 sia per il modello *Random Forest*, che per la rete neurale, evidenziando la capacità di entrambi i modelli di predire correttamente tutte le saldature del data set di test eseguite dalla macchina Schunk.

4.2.1.3 Precisione

La precisione o *confidence* è, a differenza del *Recall*, la metrica su cui si focalizzano generalmente *machine learning*, *data mining* e *information retrival*. Viene invece completamente ignorato nell'analisi di ROC. Questo l'indice rappresenta la capacità di un modello di classificazione di identificare solo i dati rilevanti, ovvero la capacità di

[36] Ovvero la corrispondenza tra parole nella lingua di origine e quelle di arrivo in una coppia di frasi che sono una la traduzione dell'altra.

identificare correttamente i casi veri tra quelli classificati come tali. Matematicamente evidenzia il numero di veri positivi rispetto al numero di record classificati come veri, ovvero la somma tra i veri positivi più il numero di falsi positivi [114]:

$$Precisione = \frac{TP}{TP + FP}$$

La precisione può essere anche chiamata *true positive accuracy* (TPA), essendo una misura di accuratezza relativa ai previsti positivi in contrasto con il tasso di scoperta degli effettivamente positivi (TPR).
La precisione è stata ricavata applicando la funzione *precision_score* della libreria *metrics* di *sci-kit learn* alle etichette di entrambi i classificatori. Il risultato ottenuto è stato 1 in entrambi i casi, dimostrandone la capacità di predire correttamente la validità della saldatura.

4.2.1.4 *F-measure*

Un'altra misura in grado di fornire informazioni riguardo l'accuratezza dei risultati è l'F-*meausre* o F1 *score*. Questa metrica è calcolata combinando il *Recall* e la Precisione.
Questa metrica, basata sulla media armonica dei due indici *Recall* e precisione, ha acquisito un certo grado di popolarità nell'ambito del *Machine Learning* già dagli anni '90. Oggigiorno è spesso utilizzata nel campo dell'*information retrieval* e per valutare la performance di modelli di classificazione. Altra applicazione dove ha trovato ampiamente spazio è la *natural language processing*, dove viene utilizzato per valutare la *Named Entity Recognition*

Classification (NERC).[37] Nel *Machine Learning* gli vengono preferite altre metriche come il coefficiente di correlazione di Matthews o il kappa di Cohen, in quanto considerano anche i valori negativi.

L'F1 *score* può variare tra zero e uno (valori alti indicano performance di classificazione migliori):

$$F - measure = \frac{2 * Recall * Precision}{Recall + Precision} = \frac{2 * TP}{2 * TP + FP + FN}$$

La metrica è un caso particolare della chiamata F_β-score o F_β-measure, corrispondente alla media armonica ponderata tra *Recall* e Precisione, con il coefficiente β pari a 1. Il valore generico può quindi essere ricondotto alla seguente formula:

$$F_\beta - measure = (1+\beta)^2 \frac{Recall * Precision}{(\beta^2\, Precision) + Recall} = \frac{(1+\beta)^2\, TP}{(1+\beta)^2 * TP + \beta^2 * FN + FP}$$

Poiché sia l'*F-measure* che l'F_β*-measure* considerano solo tre dei quattro elementi della matrice di confusione e non utilizzano nel calcolo le saldature correttamente predette come errate, si ha che due ipotetici classificatori che si differenziano solo per il *true negative rate* avrebbero lo stesso *F1-score*. Per questo motivo è stato introdotto l'*Adjusted F-Measure* (AGF):

[37] Processo di riconoscimento di unità di informazioni come nomi, inclusi nomi di persone, organizzazioni e località ed espressioni numeriche (e.g. ora, data, denaro ed espressioni percentuali) da testo non strutturato.

$$AGF = \sqrt{F_2 * InvF_{0.5}}$$

Dove F_2 è pari all' *F_β-measure* con β pari 2 e $InvF_{0.5}$ è uguale all'*F_β-measure* con β pari 0.5 basato sulla matrice di confusione ottenuta invertendo le etichette vero e falso [115].
Nell'analisi si è deciso di calcolare il punteggio più comune di questa «famiglia» di metriche, ovvero l'*f-measure*. E' stato ricavato come output della funzione *metrics.f1_score*, messa a disposizione dal pacchetto *sci-kit learn*. La funzione, confrontando un *array* monodimensionale con le variabili di output reali dei campioni con quello con le etichette ricavate dalla classificazione eseguita con *Random Forest* prima e rete neurale poi, ha restituito 1 come indice in entrambi i casi.

4.2.2 *ROC curve*

La curva *Receiver Operating Characteristic* (ROC) è uno schema grafico ampiamente utilizzato per indagare la relazione tra sensibilità e specificità di un classificatore binario.
La sensibilità, intesa come il tasso di veri positivi, misura la proporzione di positivi correttamente classificati. Il tasso di specificità, ovvero il tasso di veri negativi, misura la proporzione di negativi correttamente classificati.
Convenzionalmente il tasso vero positivo (TPR) viene tracciato rispetto al tasso falso positivo (FPR), che equivale a uno meno il tasso di vero negativo. Se un classificatore emette un punteggio proporzionale alla sua convinzione che un'istanza appartenga alla classe positiva, abbassando la soglia di decisione (oltre la quale si ritiene che l'istanza appartenga alla classe positiva), aumenteranno sia il *true positive rate* che il *false positive rate*. Variando la soglia di decisione dal suo valore massimo a quello minimo si ottiene una curva lineare a tratti da (0,0) a (1,1) tale per cui ogni segmento ha una pendenza non negativa.

La ROC *curve* viene utilizzata per affrontare problemi come la determinazione di una soglia di decisione che riduca al minimo il tasso di errore o il costo della classificazione sbagliata sotto determinate classi e distribuzioni di costi, oppure per identificare le regioni in cui un classificatore supera un altro, trovare le regioni in cui un classificatore ha prestazioni peggiori del caso e ottenere stime calibrate della classe a posteriori.

L'analisi ROC affonda le sue radici nella teoria della detenzione del segnale, detta anche teoria della rilevazione. Egan prima [116] e Swets et al. nel 2000 descrivono come abbia trovato impiego nella rappresentazione del compromesso tra *hit rate* (i.e. rapporto tra il numero di *hit* rispetto alla somma di *hit* e *miss*) e *false alarm rate* (i.e. proporzione tra *false alarm* e la somma di questi con i *corrective negative*) di classificatori [117].

L'analisi ROC ha visto presto espandere il suo campo di applicazione: dalla visualizzazione e analisi del comportamento dei sistemi diagnostici [118], ai test diagnostici nel processo decisionale medico [119].

Nel 1989 Kent Spackman fu uno dei primi ad adottare l'analisi ROC nell'ambito dell'apprendimento automatico per la valutazione e il confronto di algoritmi di *Machine Learning* [120].

Negli ultimi anni si è assistito ad un uso significativo della metrica ROC proprio nell'ambito del *Machine Learning*, dal momento che metriche semplici come l'accuratezza o altre metriche di valutazione delle performance si sono rivelate poco significative. Inoltre, essendo un metodo di rappresentazione grafica delle prestazioni, si presta particolarmente in caso di domini con distribuzione asimmetrica di classi e costi di errore di classificazione diseguali.

I grafici ROC sono di tipo bidimensionale, in cui il *true positive rate* viene tracciato sull'asse Y ed il *false positive rate* sull'asse X con l'obiettivo di mostrare i relativi compromessi tra benefici (veri positivi) e costi (falsi positivi).

Si dice classificatore discreto un classificatore che emette solamente un’etichetta di classe, producendo una coppia TPR-FPR corrispondente ad un singolo punto del grafico. Per analizzare i diversi punti nello spazio ROC (i.e. diverse classificazioni discrete) è stata presa in considerazione la Figura 3 sottostante.

Il punto in basso a sinistra nello spazio ROC, identificato dalle coordinate (0,0), rappresenta la strategia di non emettere mai una classificazione positiva. Un classificatore che si posiziona in questa punto commette n errori falsi positivi senza mai ottenere risultati veri positivi.

La strategia opposta, ovvero quella di emettere incondizionatamente classificazioni positive, è rappresentata dal punto in alto a destra, con X e Y entrambi pari a 1.

Il punto con coordinate (0;1), ovvero il punto D, rappresenta la classificazione perfetta. In generale, un punto nello spazio ROC è migliore di un altro se si trova più a nord-ovest, in quanto caratterizzato da un TPR maggiore e da un FPR inferiore.

I classificatori che appaiono sul lato sinistro di un grafico ROC (i.e. vicino all'asse X) possono essere definiti «conservativi» e tendono ad effettuare classificazioni positive solo con prove evidenti, limitando gli errori falsi positivi, ma ad avere al contempo bassi tassi positivi veri. I classificatori in alto a destra di un grafico ROC possono essere pensati come «liberali», poiché propendono a rendere positive classificazioni con prove deboli, classificando correttamente tutti i campioni effettivamente positivi. In questo caso però spesso si hanno tassi elevati di falsi positivi.

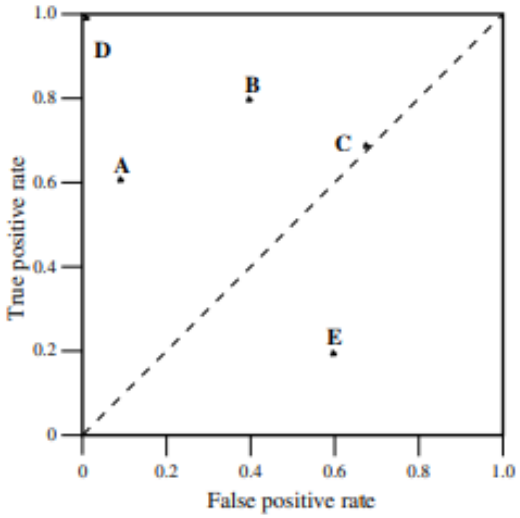

Figura 4.26 Grafico ROC base con alcuni esempi di classificatori discreti, *https://www.sciencedirect.com/science/article/abs/pii/S016786550500303X*, 05/06/2022.

La retta tratteggiata nel grafico della figura 4.26 e identificata dall'equazione y=x rappresenta la strategia di previsione casuale di una classe da parte di un algoritmo classificatore. Ad esempio, se questo è in grado di classificare in modo casuale come positivi la metà dei casi, ci si può aspettare che sia in grado di predire in modo corretto metà dei campioni positivi e metà di quelli negativi, producendo il punto con coordinate (0,5;0,5) nello spazio ROC. Allo stesso tempo, se il classificatore casuale è in grado di classificare il 90% dei casi come positivi, si può supporre che il tasso di positivi veri sia il 90%, così come il tasso dei falsi positivi, identificando il punto (0.9;0.9) sul grafico.
In conclusione, un classificatore casuale (e.g. il classificatore discreto contrassegnato dalla lettera C) produce un punto nello spazio ROC che scorre sulla diagonale rappresentabile dall'equazione y = x in base alla frequenza con cui classifica un campione come positivo.

In base al modo in cui eseguono la previsione, è possibile raggruppare i classificatori in due diversi insiemi. I classificatori discreti (e.g. alberi decisionali) sono modelli progettati per fornire una decisione precisa, con output pari a 0 o 1. Quando vengono utilizzati per

insiemi di test producono una singola matrice di confusione e sul grafico ROC identificano un singolo punto. I classificatori probabilistici (e.g. reti neurali) generano invece una probabilità, un punteggio direttamente proporzionale a quanto un'istanza possa appartenere ad una classe di output. Questi possono essere utilizzati in combinazione con una soglia per ottenere un classificatore discreto, ottenendo un 1 se la probabilità o il punteggio sono maggiori della soglia, 0 viceversa. Ne consegue che, variando il valore della soglia, è possibile ottenere come risultato un grafico che rappresenta la curva di ROC relativa al classificatore.

L'ampio impiego dell'analisi ROC nel contesto dell'apprendimento automatico deriva da:

- la capacità di fornire informazioni sull'accuratezza dei punteggi prodotti dal classificatore nel discriminare istanze positive da quelle negative;
- la dipendenza dal *true positive rate* e dal *false positive rate*, insensibili ai cambiamenti di distribuzione delle classi nel data set. Si tratta di una caratteristica fondamentale, che contraddistingue questa metrica da altre come l'accuratezza, la *F-measure* e la precisione.

Per tracciare la curva di ROC sono state utilizzate la libreria *metrics* del pacchetto *scikit-learn* e la libreria *matplotlib*. Ci si è serviti della funzione *roc_curve*, messa a disposizione dalla prima libreria, passando come parametri i due *array* con i valori di output effettivi e quelli generati dal modello classificatore. Questa funzione, in base ai due array monodimensionali, genera le coordinate della curva di ROC. Le coordinate ottenute sono state poi passate come parametro alla funzione *plot* di *matplotlib*, che ha generato i grafici delle Figure 4.27 e 4.28

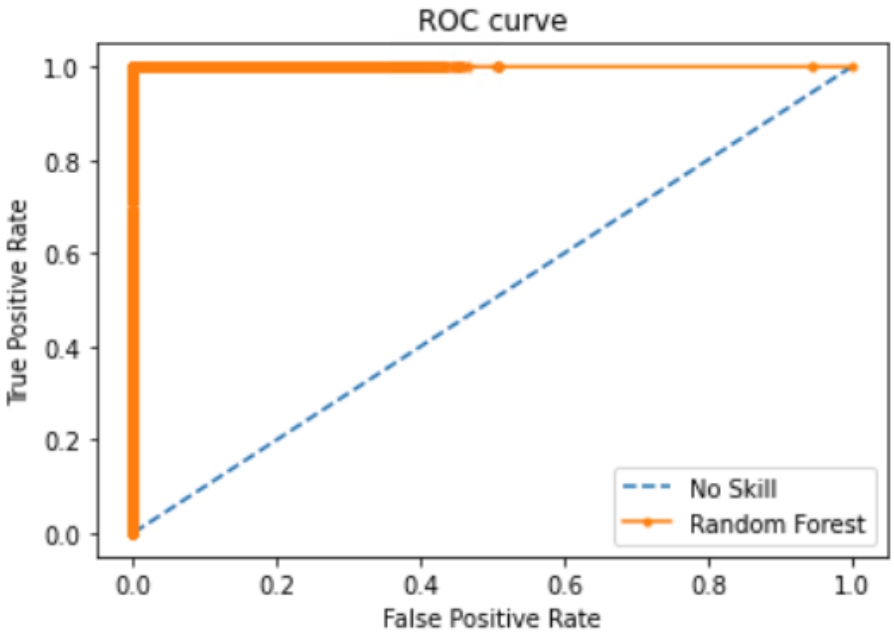

Figura 4.27 *ROC curve* relativa al modello *Random Forest* con output binario, elaborazione personale.

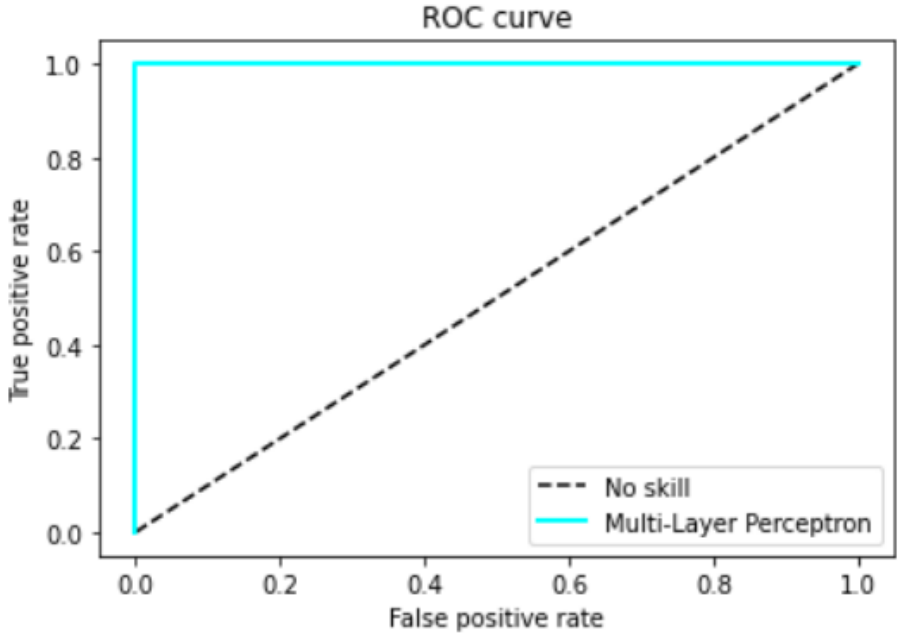

Figura 4.28 *ROC curve* relativa alla rete neurale con output binario, elaborazione personale.

I grafici relativi alle figure 4.27 e 4.28 mettono a confronto la curva ROC di un classificatore ideale di tipo casuale (linea tratteggiata) con la curva ROC generata dal *Random Forest* e dal *Multi-Layer Perceptron* (linee continue). Si può notare che l'andamento della ROC *curve* dei classificatori utilizzati sia sostanzialmente identico, denotando un eccellente lavoro di predizione svolto da entrambi i modelli di *Machine Learning*.

4.2.2.1 *AUC*

Spesso, quando occorre confrontare l'analisi dei risultati di algoritmi differenti, è preferibile una metrica di tipo numerico piuttosto che grafico. Un'importante statistica associata all'analisi ROC è la cosiddetta *Area Under ROC Curve* (AUC). Essendo che la curva di ROC si sviluppa all'interno del quadrato dell'unità, il valore AUC è compreso tra 0 e 1, estremi inclusi.

Quando il valore dell'AUC è pari a 1, significa che il classificatore è in grado di distinguere perfettamente tutti i campioni positivi da quelli negativi. Quando il valore è pari a 0, invece, significa che il classificatore ha predetto tutti i campioni effettivamente negativi come positivi e viceversa. Quando l'AUC ha un valore compreso tra 0,5 e 1, allora c'è un'alta probabilità che il classificatore sia in grado di distinguere correttamente le istanze positive dai campioni negativi. Generalmente accade quando il classificatore tende a rilevare un maggior numero di veri positivi e veri negativi rispetto a falsi positivi e falsi negativi.

Un AUC pari a 0,5 può identificare diversi scenari:

- il classificatore assegna lo stesso punteggio a tutti gli esempi di test, positivi o negativi, e quindi la curva ROC è la diagonale ascendente;
- le distribuzioni dei punteggi per classe sono simili e ciò porta ad avere una curva ROC vicina (ma non identica) alla diagonale ascendente;

- il classificatore assegna ad una metà dei campioni di una particolare classe i punteggi più alti e all'altra metà i punteggi più bassi.

La qualità di un modello, misurata sulla base delle performance attraverso la curva di ROC e l'indice AUC trova una rappresentazione grafica nella Figura 4.29.

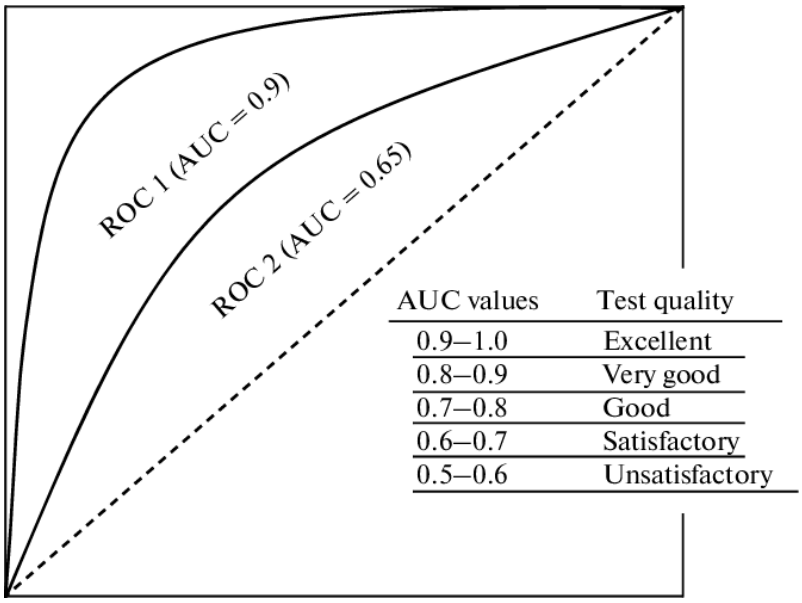

Figura 4.29 Analisi della qualità di un classificatore con la curva di ROC e l'indice AUC; *https://www.researchgate.net/figure/An-example-of-ROC-curves-with-good-AUC-09-and-satisfactory-AUC-065-parameters_fig2_276079439*, 09/06/2022.

Statisticamente l'indice AUC può essere interpretato come la probabilità di un campione effettivamente positivo di ottenere un punteggio maggiore rispetto ad un esempio negativo. Questo indice risulta essere una versione normalizzata del test "somma dei ranghi" di Wilcoxon-Mann-Whitney usato per verificare l'ipotesi nulla.[38] In conclusione,

[38] Il test viene utilizzato per verificare l'ipotesi nulla, ovvero non ci sia differenza o relazione tra due campioni di misurazioni ordinali tratti da un'unica distribuzione.

l'indice AUC può essere interpretato come un indice di separabilità tra campioni effettivamente veri e campioni falsi [121].

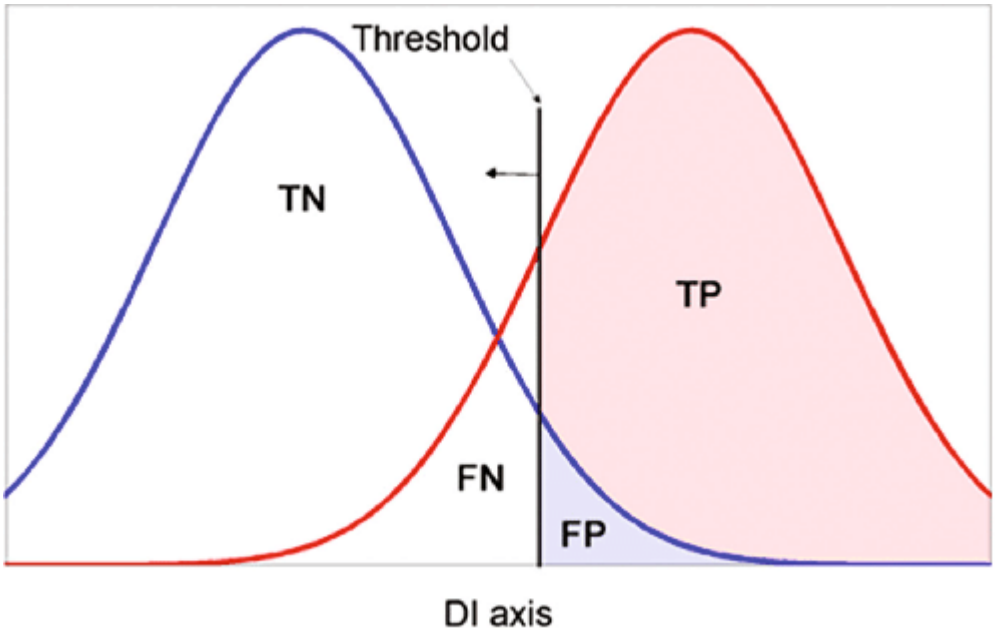

Figura 4.30 Interpretazione grafica dell'indice AUC; *https://www.researchgate.net/figure/ROC-curve-construction-Null-Normal-condition-and-Alternative-Abnormal-condition-PDFs_fig3_320384373*, 07/06/2022.

Facendo riferimento alla Figura 4.30 e analizzando i possibili valori assunti dall'indice AUC si evince che:

- se vale 1 non c'è sovrapposizione tra la curva che rappresenta i campioni positivi e quella dei campioni negativi perché il grado di separabilità che caratterizza il classificatore è quello ideale. Il modello è stato in grado di distinguere correttamente tutti i campioni veri da quelli falsi per cui si hanno solo casi *true positive* e *true negative* (senza FN e FP);

- se vale 0,7 significa che la probabilità di separare correttamente i veri dai falsi è del 70% e si verifica una sovrapposizione tra le due curve;
- se vale 0,5 le due curve si sovrappongono, come se il modello non avesse capacità di distinguere correttamente classi positive da quelle negative;
- se vale 0 le due curve sono diametralmente opposte al caso con AUC uguale ad 1 e questo significa che il classificatore ha predetto i campioni effettivamente veri come falsi e viceversa.

Per il calcolo di questo indice è stata utilizzata la funzione *roc_auc_score,* appartenente alla libreria *metrics* del pacchetto *scikit-learn* di Python. La funzione riceve come parametri l'*array* con le etichette effettive del data set di test e l'*array* con i valori predetti per calcolare e restituire l'AUC *score* relativo. Il risultato ottenuto, passando prima gli output del *Random Forest classifier* e poi quelli della rete neurale come valori predetti è 1 in entrambi i casi. Ciò significa che i due modelli addestrati sono stati in grado di distinguere le saldature corrette da quelle errate e classificare tutti i campioni ricevuti in ingresso.

4.2.3 *Random Forest* vs *Multi-Layer Perceptron*

Una volta calcolati gli indici per entrambi i modelli di classificazione, si è proceduto con il confronto dei risultati, così da istituire un paragone in termini di analisi delle performance.

Metrica	*Random Forest*	*MLP*
Accuratezza	1	1
Recall	1	1
Precisione	1	1
F-*measure*	1	1
AUC	1	1

Tabella 4.5 Paragone della performance di *Random Forest* e MLP con output binario, elaborazione personale.

Tutte le metriche riportano lo stesso risultato per entrambi i classificatori. Si può affermare che le performance eseguite dai due classificatori si sono rivelate identiche in termini di:

- Accuratezza, in quanto entrambi sono stati in grado di classificare correttamente tutti i campioni del data set di test;
- *Recall*, dato che sia il classificatore RF che l'MLP hanno predetto correttamente tutti i campioni positivi;
- Precisione, dal momento che tutti i campioni predetti come positivi erano effettivamente tali;
- *F-measure*, in quanto per entrambi i classificatori gli indici di *recall* e precisione sono stati i migliori possibili;
- AUC *score*, in quanto entrambi i classificatori sono stati in grado di eseguire classificazioni perfette.

4.3 Cosa succede con un output multi - classe

L'inaspettata quanto insolita percentuale di accuratezza nella classificazione, riscontrata in caso di etichetta di classe binaria, ha portato ad introdurre un maggiore livello di complessità nell'analisi.

Si è deciso di mappare l'etichetta di classe *Error Number* come segue, introducendo un ulteriore dettaglio a proposito della motivazione dell'errore:

- 1 se saldatura corretta;
- 2 se saldatura errata per differenza tra altezza cavo da saldare e quella impostata nei parametri;
- 3 se saldatura errata per differenza tra altezza cavo saldato e quella impostata nei parametri;
- se saldatura errata perché eseguita in un tempo superiore a quello impostato;
- se saldatura errata perché l'altezza della saldatura richiesta eccede il limite massimo;
- 0 se saldatura errata per problemi generici.

I risultati della classificazione con *Random Forest* e *MLP* mostrano come all'aumentare delle variabili di output e alla diminuzione dei casi per ogni classe con *Error Nr.* $\neq 1$, diminuisca l'accuratezza della classificazione, pur mantenendosi su percentuali molto elevate.

Entrambe le matrici di confusione mostrano come i classificatori abbiano incontrato difficoltà nel classificare correttamente alcuni valori, in particolare: *Error Nr.* = 4 e 0 per il *Random Forest* (Figura 4.31); *Error Nr.* = 4 per il *MLP* (Figura 4.32).

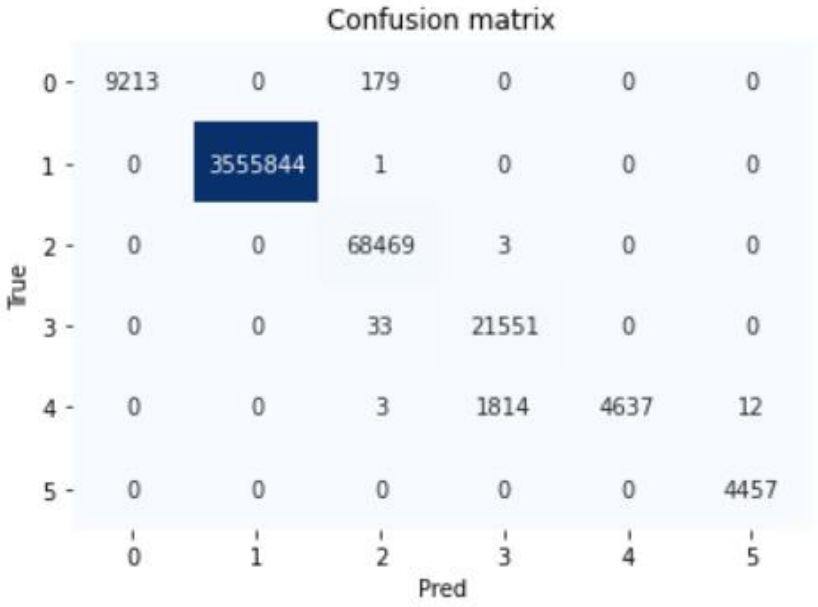

Figura 4.31 Matrice di confusione con classificatore *Random Forest* e output multi - classe, elaborazione personale con scikit-learn.

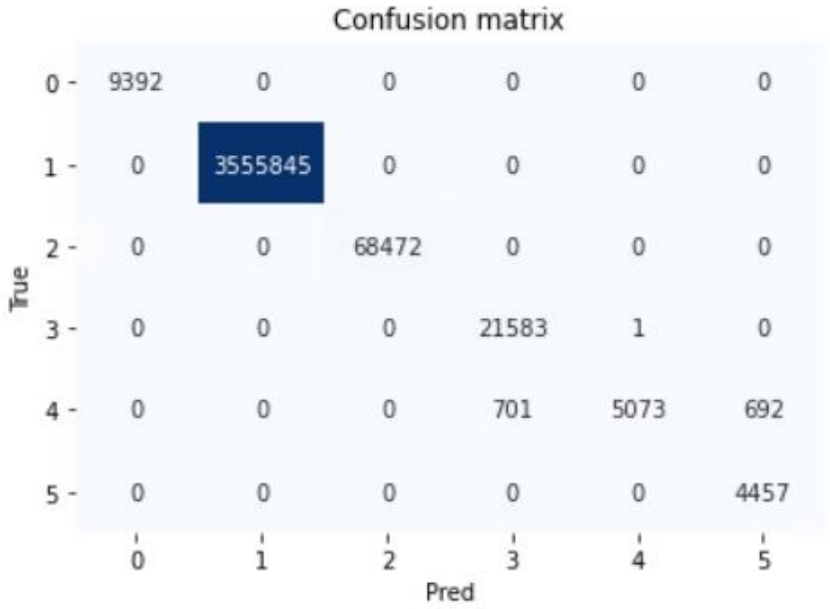

Figura 4.32 Paragone della performance di *Random Forest* e MLP, elaborazione personale con scikit-learn.

Il paragone istituito tra i due modelli mostra come gli indici di performance nel caso di un output multi - classe non abbiano più tutti un valore pari a 1 e come la capacità dei classificatori sia diminuita rispetto all'ipotesi di classificatore binario (Tabella 4.6).

Metrica		*Random Forest*	MLP
Accuratezza		0,999442	0,9996
Recall	*Micro average*	0,999442	0,9962
	Macro average	0,949417	0,964087
Precisione	*Micro average*	0,999442	0,9962
	Macro average	0,986069	0,972325
F-*measure*	*Micro average*	0,999442	0,9962
	Macro average	0,963592	0,965194
AUC	*Micro average*	0,999999	0,99999
	Macro average	0,999983	0,999903

Tabella 4.6 Paragone della performance di *Random Forest* e *MLP* con output multi - classe, elaborazione personale.

Nonostante quello che emerga dalle metriche faccia pensare che la rete neurale sia migliore al *Random Forest* in termini di performance, dall'analisi *ROC* e *AUC* emerge che il RF (Figura 4.33) possiede una capacità di generalizzare la classificazione leggermente superiore alla rete neurale (Figura 4.34).

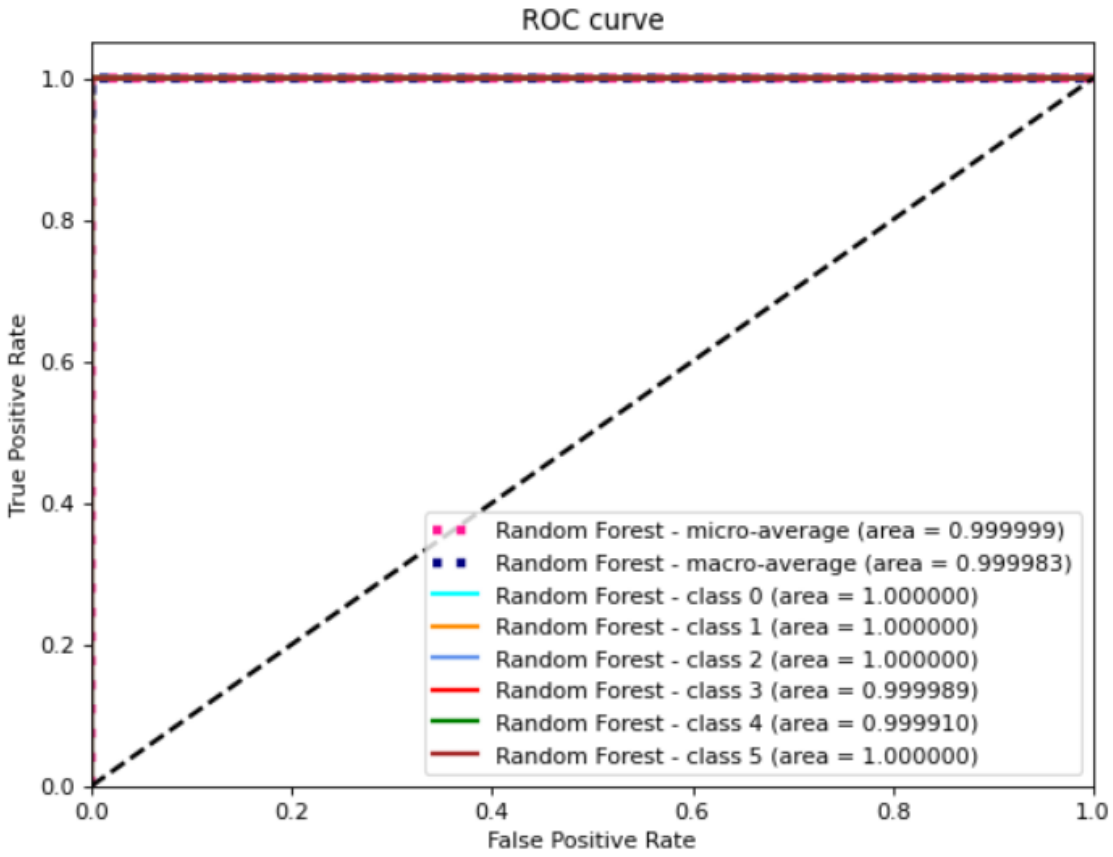

Figura 4.33 *ROC curve* relativa al modello *Random Forest* con output multi - classe, elaborazione personale.

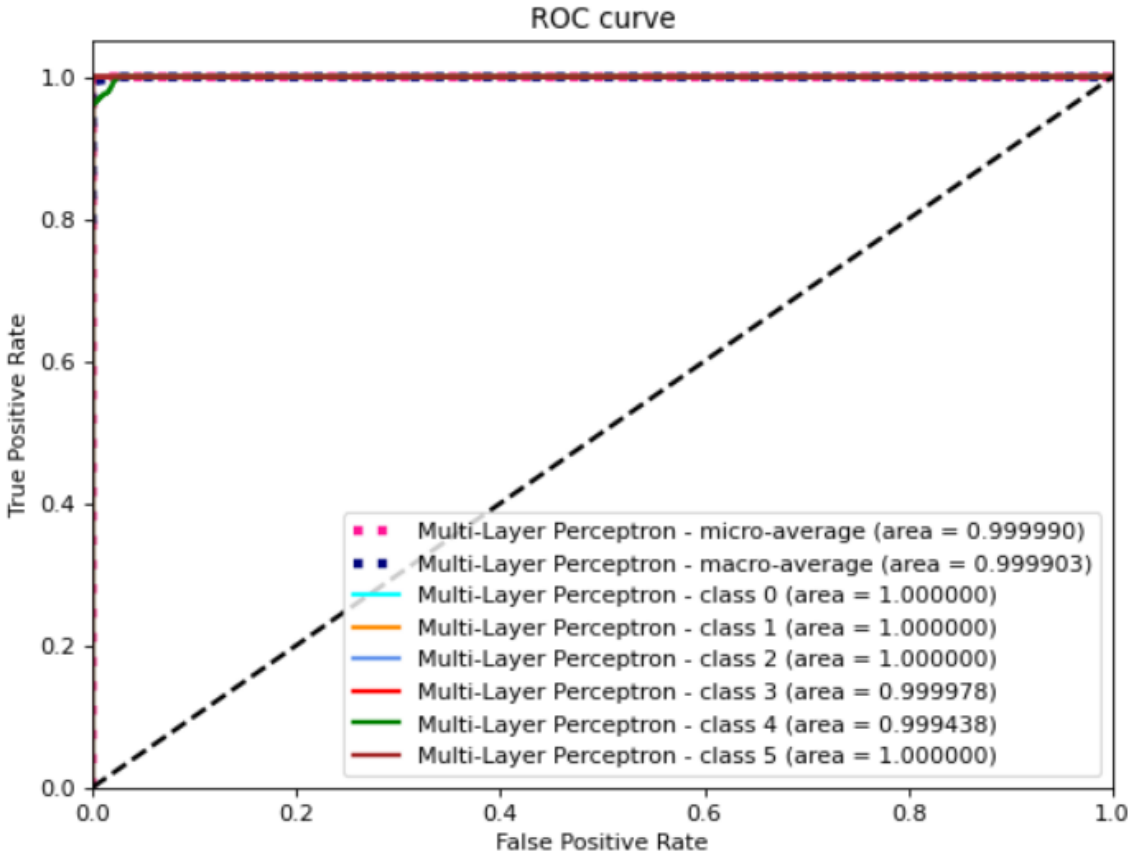

Figura 4.34 *ROC curve* relativa alla rete neurale con output multi - classe, elaborazione personale.

Capitolo 5

Conclusioni e sviluppi futuri

L'avvento delle nuove tecnologie che ad intermittenza, ormai da secoli, rivoluzionano il mondo dell'industria rende necessario ripensare al tema della gestione della Qualità. Gli strumenti ad oggi disponibili consentono di cambiare radicalmente il modo di fare industria e di produrre e comportano cambiamenti anche per il mondo della Qualità, dove la gestione del dato assume un ruolo ancora più importante.

Investire sulla Qualità significa investire su nuove fonti di vantaggi economici. I più importanti vengono dall'automazione di processi che, senza un altrettanto adeguato livello di automazione del *Quality Management*, risulterebbero inefficaci. Tra gli esempi più rilevanti ci sono i processi di gestione delle deviazioni nella qualità dei prodotti, dove automatizzare significa ridurre il tempo speso dagli addetti per gestire un determinato processo e ridurre o eliminare le perdite economiche legate alla non qualità delle produzioni.

Nonostante i recenti progressi, il ritardo è ancora importante per molte aziende italiane, dove parlare di *Quality 4.0* appare ancora una scelta azzardata, ma dove le basi per uno sviluppo in questa direzione si possono chiaramente intravedere.

La realtà produttiva di SEWS-CABIND rappresenta un caso di studio esemplare, dove lo studio qualitativo di uno dei processi produttivi *core* ha reso possibile constatare lo *status quo*, per poi identificarne gli auspicabili sviluppi futuri.

È stato dimostrato che attraverso l'adozione di tecniche di *Machine Learning* e *Deep Learning* è possibile ottimizzare il processo di controllo qualità effettuato giornalmente in SEWS-CABIND sulla produzione di cablaggi con saldatura ad ultrasuoni.

Entrambi i modelli utilizzati, l'algoritmo di *Random Forest* e la rete neurale artificiale multistrato di tipo *feed-forward*, sono stati in grado di classificare correttamente il data set che gli è stato sottoposto nella forma di file log emessi dalle saldatrici Minic III opportunamente rielaborati. Nel contesto di una classificazione di tipo supervisionato è stato dimostrato che i modelli, a partire da un *training set* epurato delle *feature* relative alla qualità della saldatura (i.e. *Error Nr.*; *Error Text*), hanno prodotto le stesse informazioni sul livello qualitativo presenti nei file di log integrali, dove i valori relativi alle variabili *Error Nr* e *Error Text* vengono imputati manualmente dall'operatore al termine di ogni ciclo di saldatura.

Gli algoritmi si sono dimostrati piuttosto performanti dal punto di vista di più metriche: Accuratezza, *Recall*, Precisione, *F-measure*, Coefficiente di correlazione di Matthews, Kappa di Cohen, *ROC* e *AUC curve*.

La prima importante osservazione che si deduce è che il ricorso ad algoritmi di *Machine Learning* e *Deep Learning* in un *business scenario* di questo tipo potrebbe tradursi in una gestione del monitoraggio sulle saldature più celere e privo di intervento manuale da parte dell'operatore.

Una possibile estensione del lavoro di analisi fatto potrebbe poi guardare all'implementazione di algoritmi con il compito di fornire direttive sui parametri di impostazione ottimali in base ai quali eseguire le saldature effettive, riducendo di molto le

tempistiche della fase preparatoria, limitando il numero di prove da effettuare e con esse gli scarti prodotti.
E perché non estendere poi il raggio di analisi ad altre macchine attivamente coinvolte nel processo di produzione del cablaggio: le così dette *cutting-machines* Komax, responsabili della fase di taglio ad alta precisione dei cavi e anch'esse dotate di software ed emissione di file log come output del processo.

5.1 Ricerca dei parametri ottimali per le macchine Schunk

Tornando al processo di saldatura eseguito dalle macchine Schunk Minic III, si ricorda che l'operatore prima di tutto imposta manualmente i parametri sulla macchina, per poi proseguire con la fase di preparazione. Terminata questa fase, la macchina procede con l'effettuare le saldature effettive.
La fase di preparazione della macchina è deputata alla ricerca dei parametri ottimali, così che questa sia poi in grado di eseguire correttamente le saldature durante il ciclo di produzione. E' durante questa fase che i parametri impostati dall'operatore vengono testati più e più volte con l'esecuzione di saldature di prova, fino alla definizione dei parametri che soddisfano tutti i requisiti.
Tali parametri possono essere divisi in due gruppi:

- le indicazioni da fornire al sonotrodo e la parte della macchina attiva durante l'esecuzione della saldatura. Nel primo gruppo troviamo l'energia che dovrà applicare il sonotrodo durante l'esecuzione della saldatura, oppure quanto dovrà oscillare in termini di ampiezza durante in questa operazione;
- le misure di riferimento. Al secondo gruppo appartengono parametri come l'altezza di riferimento dei cavi da saldare, quella dei cavi una volta conclusa la saldatura e il tempo di riferimento da impiegare per eseguire una saldatura.

Una possibile analisi alternativa a quella eseguita sarebbe implementare un algoritmo di apprendimento automatico in grado di fornire informazioni riguardo a quali parametri sarebbero preferibili o meno. Il modello sarebbe in grado di fornire informazioni utili all'operatore per affrontare una fase di preparazione della macchina in modo più efficiente, limitando il numero di prove da effettuare e gli scarti prodotti.

5.2 Komax *machines* e ottimizzazione dei parametri

Le considerazioni sugli sviluppi futuri potrebbero essere estese anche ad altre tipologie di macchine coinvolte nel processo di produzione del cablaggio in SEWS-CABIND: le *cutting-machines* Komax. Queste macchine sono coinvolte nel primo step di produzione del cablaggio, ovvero il taglio dei cavi in base alle lunghezze definite in fase di progettazione.

Così come per le macchine saldatrici, anche la fase di lavorazione delle macchine Komax prevede uno step di preparazione antecedente alla produzione vera e propria. Nella fase di preparazione l'operatore alterna due operazioni: la modifica dei parametri usati dalla macchina per eseguire il taglio e la prova di questi parametri.

Le macchine Komax emettono anch'esse un file di log come output della lavorazione eseguita. Seppur di formato leggermente diverso rispetto ai file log delle Minic III, contengono informazioni a proposito dei valori relativi ai parametri impostati dall'operatore, l'inizio o la fine di una fase di preparazione della macchina e l'inizio o la fine di una fase di lavorazione effettiva.

Appare chiaro come anche in questo caso si potrebbe pensare all'implementazione di un algoritmo di apprendimento automatico in grado di fornire i parametri ottimali da impostare per affrontare il processo di taglio dei cavi.

Tempi prolungati dettati dall'impostazione manuale dei parametri e da test ripetuti e un elevato quantitativo di scarto lascerebbero il posto ad una fase di preparazione della macchina molto più efficiente o quasi del tutto assente, con tempistiche, prove e scarti sensibilmente ridotti.

5.3 Manutenzione predittiva e macchine Schunk

Le riflessioni a proposito di eventuali sviluppi futuri potrebbero infine approdare al tema della manutenzione predittiva. Considerata una manutenzione fondata sulle stime dello stato di degrado di un elemento, si basa sulle condizioni effettive delle apparecchiature per prevedere quando sarà necessaria la manutenzione o la sostituzione.
È un tipo di manutenzione che segue la salute, lo stato e le prestazioni di un asset in tempo reale. Mira a ridurre i guasti costosi e inaspettati e offre al produttore l'opportunità di pianificare la manutenzione intorno al proprio programma di produzione. Attraverso la combinazione di dati in tempo reale ottenuti tramite *l'Industrial Internet of Things* (IIoT), la manutenzione predittiva analizza continuamente lo stato delle attrezzature durante lo svolgimento di normali operazioni, per ridurre la probabilità di guasti imprevisti della macchina.
La *ratio* che sottostà a questo tipo di «manutenzione 4.0» è quella di individuare uno o più parametri che vengono misurati ed elaborati utilizzando approcci di apprendimento automatico.

La rapida diffusione dell'Industria 4.0 ha portato un utilizzo sempre più diffuso di sensori nell'ambito manifatturiero che, combinati con modelli di apprendimento automatico, rappresentano la spina dorsale della nuova manutenzione 4.0.
L'attuale tipo di gestione della manutenzione effettuata in SEWS-CABIND sulle macchine saldatrici, basata sulla sostituzione del sonotrodo a fronte di un numero definito a priori di

saldature, potrebbe lasciare spazio ad una visione automatizzata e migliorata del processo. L'adozione di una manutenzione predittiva a tutto tondo prevedrebbe l'applicazione al sonotrodo di analizzatori frequenza, anch'essi integrati nel processo di emissione di dati nel file di output, in grado fornirne informazioni a proposito dello stato di salute del sonotrodo in base alla frequenza di oscillazioni. I dati così ottenuti nei file di log potrebbero poi costituire il data set per algoritmi di apprendimento automatico capaci di analizzare *real-time* lo stato di salute della macchina e prevederne eventuali guasti futuri.

La scelta del componente delle macchine saldatrici Minic III cui applicare un analizzatore di frequenza ricadrebbe sul sonotrodo poiché considerato quello più delicato e maggiormente esposto al rischio di usura. Il dispositivo agisce attivamente durante tutto il processo di saldatura, «sfregando» contro il materiale da saldare. Va da sé che l'azione ripetuta per centinaia di migliaia di esecuzioni sia fonte di deterioramento e conseguente malfunzionamento o rottura. Al momento questo rischio viene gestito tramite la definizione a priori di due soglie:

- un numero di saldature (10.000) dopo il quale il sonotrodo viene girato;
- un numero di saldature (20.000) dopo il quale il sonotrodo viene cambiato.

In ottica di miglioramento ed automazione della gestione appena esposta, sarebbe necessario avere nei file di log valori che permettano di individuare lo stato di salute del sonotrodo (cosa che ora non avviene). Con l'applicazione di analizzatori di frequnza sarebbe possibile eseguire un'analisi preventiva dello stato di salute del sonotrodo, sfruttando la caratteristica per cui la frequenza di oscillazione è influita dallo stato di salute di questo componente.

Infine, implementare il sensore in modo tale che aggiunga tali valori nel file di log, per poi utilizzarli come data set per un modello di apprendimento automatico in grado di elaborare analisi relative allo stato di salute del sonotrodo in base anche alla frequenza di oscillazione misurata dal sensore.

L'approccio tradizionale legato alla analisi e alla misurazione puntuale della qualità del singolo processo perde progressivamente di efficacia e lascia spazio ad un modo di considerare la qualità più ampio. Questo significa l'emergere e il confrontarsi con dinamiche più complesse non solo nel campo della Qualità, ma in tutta la realtà aziendale, dove la gestione dei diversi aspetti diventa ogni giorno più interconnessa.

Non solo l'approccio Qualità 4.0 può essere un driver di velocità per i benefici che comporta. E' tutto il movimento digitale collegato alle tecnologie *Industry 4.0* che porterà con sé lo sviluppo delle moderne metodiche della Qualità, in un circolo virtuoso che è destinato a fornire accelerazione lungo tutta la catena del valore.

Bibliografia e sitografia

[1] Tullio De Mauro, *Grande dizionario italiano dell'uso*, Torino: UTET, 2000.

[2] Shai Shalev-Shwartz e Shai Ben-David, *UNDERSTANDING MACHINE LEARNING From Theory to Algorithms*, New York: Cambridge University Press, 2014.

[3] Lex Friedman, *Deep Learning Basics* (lezione introduttiva al MIT), Stati Uniti, 2019. Disponibile all'indirizzo: *deeplearning.mit.edu*, 24/01/2022.

[4] John McCarthy, *WHAT IS ARTIFICIAL INTELLIGENCE?*, Stanford: Stanford University Press, 2004.

[5] Marco Somalvico, *Intelligenza artificiale*, *Scienza & vita nuova*: Hewlett Packard, 1987.

[6] Communication from the Commission to the European Parliament, the European Council, the Council, the European Economic and Social Committee and the Committee of the Regions on Artificial Intelligence for Europe, Brussels, 25.4.2018 COM(2018) 237 final. Disponibile all'indirizzo: *eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52018DC0237*, 24/01/2022.

[7] Alan M. Turing, *Computing Machinery and Intelligence*, in Mind, 1950.

[8] Warren S. McCulloch e Walter S. Pitts, *A logical calculus of the ideas immanent in nervous activity, in Bulletin of Mathematical Biophysics*, 1943. Disponibile all'indirizzo: *https://link.springer.com/article/10.1007/BF02478259*, 02/05/2022.

[9] David E. Rumelhart, Geoffrey E. Hinton e Ronald J. Williams, *Learning representations by back-propagating errors*, in Nature 323, 1986.

[10] John McCarthy, *WHAT IS ARTIFICIAL INTELLIGENCE?*, Stanford: Stanford University Press, 2004.

[11] Frank Rosenblatt, *Principles of neurodynamics perceptions and the theory of brain mechanisms*, New York: Cornell Aeronautical Laboratory, 1961.

[12] Marvin L. Minsky e Seymour A. Papert, *Perceptrons, An Introduction to Computational Geometry*, Cambridge: The MIT Press, 1969.

[13] Definizione dell'IA secondo Andrew Ng, professore dell'Università di Stanford. Disponibile all'indirizzo: *www.gsb.stanford.edu/insights/andrew-ng-why-ai-new-electricity*, 26/01/2022.

[14] Nick Bostrom, *Superintelligence, Paths, Dangers, Strategies*, OUP Oxford, 2012.

[15] *THE BELMONT REPORT*, 1979. Disponibile all'indirizzo: *www.hhs.gov/ohrp/sites/default/files/the-belmont-report-508c_FINAL.pdf*, 26/01/2022.

[16] Erika Buzzo, *Intelligenza Artificiale nel 2017: un anno importante per l'AI*. Disponibile all'indirizzo: *namu.io/intelligenza-artificiale-nel-2017/*, 26/01/2022.

[17] Discorso di Vladimir Putin sull'IA. Disponibile all'indirizzo: *https://www.smartiani.com/news/vladimir-putin-ai-mondo-4075*, 26/01/2022.

[18] Department of International Cooperation Ministry of Science and Technology (MOST), P.R.China, *Next Generation Artificial Intelligence Development Plan*. Disponibile all'indirizzo: *https://www.mfa.gov.cn/ce/cefi//eng/kxjs/P020171025789108009001.pdf*, 26/01/2022.

[19] Oxford Insights, *AI Readiness Index 2020*. Disponibile all'indirizzo: *https://www.oxfordinsights.com/government-ai-readiness-index-2020*, 26/01/2022.

[20] Snezha Kazakova e Allison Dunne Daan Bijwaard, *European enterprise survey on the use of technologies based on artificial intelligence*, Lussemburg: Publications Office of the European Union, 2020. Disponibile all'indirizzo*: https://op.europa.eu/en/publication-detail/-/publication/f089bbae-f0b0-11ea-991b-01aa75ed71a1*, 26/01/2022.

[21] *THE RISE OF ARTIFICIAL INTELLIGENCE: FUTURE OUTLOOK AND EMERGING RISKS*. Disponibile all'indirizzo: *http://www.agcs.allianz.com/insights/white-papers-andcase-studies/artificial-intelligence/*, 26/01/2022.

[22] Stuart Russell, *Of Myths and Moonshine*, contributo alla conversazione *The Myth of AI*. Disponibile all'indirizzo: *https://www.edge.org/conversation/the-myth-of-ai#26015*, 26/01/2022.

[23] *Dove va l'intelligenza artificiale*: MIT Technology Review, 2015. Disponibile all'indirizzo: *http://www.linkiesta.it/it/article/2015/02/21/dove-va-lintelligenzaartificiale/24781/*, 26/01/2022.

[24] Accordo internazionale *OECD principles on AI*. Disponibile all'indirizzo: *https://www.oecd.org/science/forty-two-countries-adopt-new-oecd-principles-on-artificial-intelligence.htm*, 27/01/2022.

[25] Accordo internazionale *Global Partnership on Artificial Intelligence* (GPAI). Disponibile all'indirizzo: *https://www.gov.uk/government/publications/joint-statement-from-founding-members-of-the-global-partnership-on-artificial-intelligence/joint-statement-from-founding-members-of-the-global-partnership-on-artificial-intelligence*, 27/01/2022.

[26] Analisi della Forrester Research sull'IA, Forbes. Disponibile all'indirizzo: *https://www.forrester.com/report/The-Top-Emerging-Technologies-To-Watch-2017-To-2021/RES133144*, 27/01/2022.

[27] Gli obiettivi della Neuralink Corporation. Disponibile all'indirizzo: *https://www.wsj.com/articles/elon-musk-launches-neuralink-to-connect-brains-with-computers-1490642652*, 27/01/222.

[28] Il supercomputer Leonardo. Disponibile all'indirizzo: *https://www.cineca.it/temi-caldi/Leonardo*, 27/01/222.

[29] Daniela Scaramuccia su Watson di IBM. Disponibile all'indirizzo: *https://formiche.net/2017/04/watson-ibm-medicina/*, 27/01/222.

[30] Terence Milles, *Machine Learning Vs. Artificial Intelligence: How Are They Different?* In Forbes, 2018. Disponibile all'indirizzo:

https://www.forbes.com/sites/forbestechcouncil/2018/07/11/machine-learning-vs-artificial-intelligence-how-are-they-different/?sh=6d8d39173521, 01/02/2022.

[31] Arthur L. Samuel, Some *Studies in Machine Learning Using the Game of Checkers*, in IBM Journal of Research and Development, 1959.

[32] Tom M. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[33] Arthur L. Samuel, *Some Studies in Machine Learning Using the Game of Checkers*, in IBM Journal of Research and Development, 1959.

[34] Yann LeCun, Yoshua Bengio e Geoffrey Hinton, *Deep learning*, in Nature, 2015. Disponibile all'indirizzo: *https://www.nature.com/articles/nature14539*, 10/03/2022.

[35] Richard O. Duda, Peter E. Hart e David G. Stork, *Pattern classification*, New York: John Wiley & Sons INC, 2012.

[36] Ronald A. Fisher, *The use of multiple measurements in taxonomic problems*, in Annual Eugenic, 1936.

[37] Andrew J. Frank e Arthur Asuncion, *UCI machine learning repository*, University of California, Irvine, 2010. Disponibile all'indirizzo: *http://archive.ics.uci.edu/ml*, 10/03/2022.

[38] Christopher M. Bishop, *Pattern recognition and machine learning*, Berlino: Springer, 2006.

[39] Arthur L. Samuel, *Some Studies in Machine Learning Using the Game of Checkers*, in IBM Journal of Research and Development, 1959.

[40] Frank Rosenblatt, *Principles of neurodynamics perceptions and the theory of brain mechanisms*, New York: Cornell Aeronautical Laboratory, 1961.

[41] Thomas M. Cover e Peter E. Hart, *Nearest Neighbor Pattern Classification*, in IEEE Transactions on Information Theory, 1967.

[42] James G. DeJong, *Generalizations based on explanations*, in IJCAI'81 the seventh international joint conference on artificial intelligence, 1981.

[43] Terrence J. Sejnowski e Charles R. Rosenberg, *NETtalk: a parallel network that learns to read aloud*, in The Johns Hopkins University Electrical Engineering and Computer

Science Technical Report, 1986. Disponibile all'indirizzo: *https://papers.cnl.salk.edu/PDFs/NETtalk_%20A%20Parallel%20Network%20That%20Learns%20to%20Read%20Aloud%201988-3562.pdf*, 06/02/2022.

[44] Neil McCulloch, Mark Bedworth e John Bridle, *NETspeak — A re-implementation of NETtalk*, in Computer Speech & Language, 1987.

[45] Alessandro Rezzani, *Big Data. Architettura, tecnologia e metodi per l'utilizzo di grandi basi di dati*, Santarcangelo di Romagna: Maggioli Editore, 2013.

[46] Xue-Wen Chen e Xiaotong Lin, *Big Data Deep Learning: Challenges and Perspectives*, in IEEE, 2014. Disponibile all'indirizzo: *https://ieeexplore.ieee.org/abstract/document/6817512/authors#authors*, 07/02/202.

[47] Martin Hilbert e Priscila López, *The World's Technological Capacity to Store, Communicate, and Compute Information*, in Science, 2011.

[48] Viktor Mayer-Schönberger e Kenneth Cukier, *Big Data: A Revolution that Will Transform how We Live, Work, and Think*, New York: Houghton Mifflin Harcourt Publishing Company, 2013.

[49] Jaiwei Han, Micheline Kamber e Jian Pei, *Data Mining: Concepts and Techniques. Third edition*, Waltham: Morgan Kaufmann Publishers, 2012.

[50] *Ibid*.

[51] *Ibid*.

[52] Geoffrey J. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*, New York: Wiley, 1992.

[53] Tom M. Mitchell, *Generative and discriminative classifiers: naive bayes and logistic regression*, In Machine learning: ECML, Berlino: 2005.

[54] Iris Hendrickx e Antal Van den Bosch, *Hybrid Algorithms with Instance-Based Classification*, in Machine Learning: ECML, Berlino: 2005.

[55] David W. Aha, *Lazy Learning*, Berlino: Springer, 1997.

[56] Mr. Bayes e Mr. Price, *An Essay towards solving a Problem in the Doctrine of Chances*. *By the late Rev. Mr. Bayes, communicated by Mr. Price, in a letter to John Canton, M. A. and F. R. S.*, Philosophical Transactions of the Royal Society of London, 1763. Disponibile all'indirizzo: *https://web.archive.org/web/20110410085940/http://www.stat.ucla.edu/history/essay.pdf* , 10/04/2022.

[57] Jaiwei Han, Micheline Kamber e Jian Pei, *Data Mining: Concepts and Techniques. Third edition*, Waltham: Morgan Kaufmann Publishers, 2012.

[58] Vladimir Vapnik, Bernhard Boser e Isabelle Guyon, *A training algorithm for optimal margin classifiers in Proceedings of the fifth annual workshop on Computational learning theory*, 1992. Disponibile all'indirizzo: *https://dl.acm.org/doi/10.1145/130385.130401*, 14/02/2022.

[59] Jaiwei Han, Micheline Kamber e Jian Pei, *Data Mining: Concepts and Techniques. Third edition*, Waltham: Morgan Kaufmann Publishers, 2012.

[60] John R. Quinlan, *Induction of Decision Trees*, in Machine Learning, 1986.

[61] Leo Breiman, Jerome H. Friedman, Richard Olshen, Charles J. Stone, *Classification and Regression Trees*, 1983. Disponibile all'indirizzo: *https://www.taylorfrancis.com/books/mono/10.1201/9781315139470/classification-regression-trees-leo-breiman-jerome-friedman-richard-olshen-charles-stone*, 10/04/2022.

[62] Jaiwei Han, Micheline Kamber e Jian Pei, Data *Mining: Concepts and Techniques. Third edition*, Waltham: Morgan Kaufmann Publishers, 2012.

[63] Pang-Ning Tan, Michael Steinbach e Vipin Kumar, *Introduction to Data Mining*, Boston: Addison-Wesley Longman Publishing Co, 2006.

[64] David Opitz e Richard Maclin, *Popular ensemble methods: An empirical study*, in Journal of Artificial Intelligence Research, 1999. Disponibile all'indirizzo: *https://jair.org/index.php/jair/article/view/10239*, 11/04/2022.

[65] Tim Kam Ho, *The Random Subspace Method for Constructing Decision Forests*, in Pattern Analysis and Machine Intelligence, 1998.

[66] Leo Breiman, *Random Forests*, in Machine Learning, 2001. Disponibile all'indirizzo: *https://link.springer.com/article/10.1023/A:1010933404324*, 11/04/2022.

[67] Jaiwei Han, Micheline Kamber e Jian Pei, D*ata Mining: Concepts and Techniques. Third edition*, Waltham: Morgan Kaufmann Publishers, 2012.

[68] Yann LeCun, Yoshua Bengio e Geoffrey Hinton, *Deep learning*, in Nature, 2015. Disponibile all'indirizzo: *https://www.nature.com/articles/nature14539*, 11/04/2022.

[69] Ibid.

[70] Alex Krizhevsky, Ilya Sutskever e Geoffrey E. Hinton, *Imagenet classification with deep convolutional neural networks*, in Advances in Neural Information Processing Systems, 2012.

[71] FaceNet etichetta alcune facce di individui africani come gorilla. Disponibile all'indirizzo: *https://www.ilfattoquotidiano.it/2018/01/12/google-foto-se-si-cercano-gorilla-o-scimmia-nessun-risultato-censurate-perche-mostravano-immagini-di-neri/4089722/*, 11/04/2022.

[72] Roberto Prevete, *Il neurone biologico*. Disponibile all'indirizzo: *http://www.federica.unina.it/smfn/reti-neurali-e-machinelearning/neurone-biologico/*, 12/04/2022.

[73] Warren S. McCulloch e Walter S. Pitts, *A logical calculus of the ideas immanent in nervous activity*: Bulletin of Mathematical Biophysics, 1943. Disponibile all'indirizzo: *https://link.springer.com/article/10.1007/BF02478259*, 13/04/2022.

[74] Frank Rosenblatt, *Principles of neurodynamics perceptions and the theory of brain mechanisms*, New York: Cornell Aeronautical Laboratory, 1961.

[75] Donald O. Hebb, *The organization of behavior; a neuropsychological theory*, New York: Wiley, 1949.

[76] Marvin L. Minsky e Seymour A. Papert, *Perceptrons, An Introduction to Computational Geometry*, Cambridge: The MIT Press, 1969.

[77] Richard S. Sutton e Andrew G. Barto, *2.3 Softmax Action Selection*, *in Reinforcement Learning: An Introduction*, Cambridge: The MIT Press, 1998.

[78] Simon Haykin, FEEDFORWARD NEURAL NETWORKS: AN INTRODUCTION. Disponibile all'indirizzo: *https://media.wiley.com/product_data/excerpt/19/04713491/0471349119.pdf*, 12/04/2022.

[79] David E. Rumelhart, Geoffrey E. Hinton e Ronald J. Williams, *Learning representations by back-propagating errors*, in Nature 323, 1986. Disponibile all'indirizzo: *https://www.nature.com/articles/323533a0*, 16/04/2022.

[80] Paul J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science*, Harvard University, 1974. Disponibile all'indirizzo: *https://www.researchgate.net/publication/279233597_Beyond_Regression_New_Tools_for_Prediction_and_Analysis_in_the_Behavioral_Science_Thesis_Ph_D_Appl_Math_Harvard_University*, 12/04/2022.

[81] L'azienda SEWS-CABIND e il Gruppo Sumitomo. Disponibile all'indirizzo: *http://www.sews-cabind.com/*, 10/5/2022.

[82] Python. Disponibile all'indirizzo: *https://docs.python.org/3/faq/general.html#what-is-python*, 10/5/2022.

[83] Python e la *list comprehension*. Disponibile all'indirizzo: *https://docs.python.org/3/tutorial/datastructures.html*, 10/5/2022.

[84] Benjamin Zorn, *The Measured Cost of Conservative Garbage Collection Software: Practice and Experience*, University of Colorado Boulder: Department of Computer Science, 1993.

[85] Blog *Stackoverflow*. Disponibile all'indirizzo: *https://stackoverflow.blog/2021/07/14/getting-started-with-python/*, 10/5/2022.

[86] Librerie Python. Disponibile all'indirizzo: *https://pandas.pydata.org/docs/getting_started/overview.html*, 13/5/2022.

[87] Librerie Python. Disponibile all'indirizzo*: https://matplotlib.org/2.2.5/index.html*, 13/5/2022.

[88] Librerie Python. Disponibile all'indirizzo: *https://matplotlib.org/stable/tutorials/introductory/pyplot.html*, 13/5/2022.

[89] Librerie Python. Disponibile all'indirizzo: *http://seaborn.pydata.org/introduction.html*, 13/5/2022.

[90] Librerie Python. Disponibile all'indirizzo: *https://scikit-learn.org/stable/getting_started.html*, 13/5/2022.

[91] Abdulhamit Subasi, *Practical Machine Learning for Data Analysis Using Python*, 2020. Disponibile all'indirizzo: *https://www.researchgate.net/publication/342599493_Practical_Machine_Learning_for_Data_Analysis_Using_Python*, 13/5/2022.

[92] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang e Huan Liu, *Feature Selection: A Data Perspective*, 2016. Disponibile all'indirizzo: *https://arxiv.org/abs/1601.07996*, 13/5/2022.

[93] S. Ramírez-Gallego et al., *An Information Theory-Based Feature Selection Framework for Big Data Under Apache Spark*, in IEEE Transactions on Systems, Man and Cybernetics: Systems, 2018. Disponibile all'indirizzo: *https://ieeexplore.ieee.org/document/7970198/authors#authors*, 14/5/2022.

[94] Jianqing Fan e Runze Li, *Variable Selection via Nonconcave PenalizedLikelihood and its Oracle Properties*, 2001. Disponibile all'indirizzo: *https://pennstate.pure.elsevier.com/en/publications/variable-selection-via-nonconcave-penalized-likelihood-and-its-or*, 14/5/2022.

[95] Hui Zou, *The Adaptive Lasso and Its Oracle Properties*, 2012. Disponibile all'indirizzo: *https://www.tandfonline.com/doi/abs/10.1198/016214506000000735*, 14/5/2022.

[96] Martin G. Larson, *Analysis of Variance*, 2008. Disponibile all'indirizzo: *https://www.ahajournals.org/doi/full/10.1161/CIRCULATIONAHA.107.654335*, 14/5/2022.

[97] Mahdi Bejani, Davood Gharavian e Nasrolah Charkari, *Audiovisual emotion recognition using ANOVA feature selection method and multi-classifier neural networks*, 2014. Disponibile all'indirizzo: *https://link.springer.com/article/10.1007/s00521-012-1228-3*, 14/5/2022.

[98] Bjoern H. Menze, B. Michael Kelm, Ralf Masuch, Uwe Himmelreich, Peter Bachert, Wolfgang Petrich e Fred A. Hamprecht, *A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data*, 2009. Disponibile all'indirizzo: *https://europepmc.org/article/PMC/2724423*, 16/5/2022.

[99] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, Charles J. Stone, *Classification and Regression Trees*, 1984. Disponibile all'indirizzo: *https://www.semanticscholar.org/paper/Classification-and-Regression-Trees-Breiman-Friedman/8017699564136f93af21575810d557dba1ee6fc6*, 16/5/2022.

[100] *Ibid*.

[101] Díaz-Uriarte e De Andres, *Gene selection and classification of microarray data using random forest*, 2006. Disponibile all'indirizzo: *https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-7-3*, 17/5/2022.

[102] Benjamin A. Goldstein, Eric C Polley e Farren B. S. Briggs, *Random forests for genetic association studies*, 2011. Disponibile all'indirizzo: *https://pubmed.ncbi.nlm.nih.gov/22889876/#affiliation-1*, 17/5/2022.

[103] Michael Segal, *Machine Learning Benchmarks and Random Forest Regression*, 2004. Disponibile all'indirizzo: *https://www.semanticscholar.org/paper/Machine-Learning-*

Benchmarks-and-Random-Forest-Segal/d91ea30f4f9de817b29bb4ece00f43cb971822b4, 18/5/2022.

[104] Philipp Probst, Marvin Wright e Anne-Laure Boulesteix, *Hyperparameters and Tuning Strategies for Random Forest*, 2019. Disponibile all'indirizzo: *https://arxiv.org/abs/1804.03515*, 18/5/2022.

[105] Zelda B. Zabinsky, *Random Search Algorithms*, 2010. Disponibile all'indirizzo: *https://www.semanticscholar.org/paper/Random-Search-Algorithms-Zabinsky/184ec80f16524f8a6c5c432aa35d07d717e71a82*, 18/5/2022.

[106] Hassan Ramchoun, Mohammed Amine Janati Idrissi, Youssef Ghanou e Mohamed Ettaouil , *Multilayer Perceptron: Architecture Optimization and Training*, 2016. Disponibile all'indirizzo: *https://www.semanticscholar.org/paper/Multilayer-Perceptron%3A-Architecture-Optimization-Ramchoun-Idrissi/404179608460b02dd43baf308e00acd454b81d79*, 18/5/2022.

[107] Shaohong Tian, Xianfeng Zhang, Jie Tian e Quan Sun, *Random Forest Classification of Wetland Landcovers from Multi-Sensor Data in the Arid Region of Xinjiang, China*, 2016. Disponibile all'indirizzo: *https://www.mdpi.com/2072-4292/8/11/954*, 18/5/2022.

[108] Diederik P. Kingma e Jimmy Lei Ba, *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION*, 2014. Disponibile all'indirizzo: *https://www.semanticscholar.org/paper/Adam%3A-A-Method-for-Stochastic-Optimization-Kingma-Ba/a6cb366736791bcccc5c8639de5a8f9636bf87e8*, 18/5/2022.

[109] Jasmina Dj. Novakovic, Alempije Veljovic, Sinisa S. Ili, Zeljko Papi e Milica Tomovic, *Evaluation of Classification Models in Machine Learning*, 2017. Disponibile all'indirizzo: *https://www.scirp.org/reference/ReferencesPapers.aspx?ReferenceID=2544204*, 1/06/2022.

[110] C. Ferri, J. Hernández-Orallo e R. Modroiu, *An experimental comparison of performance measures for classification*, 2009. Disponibile all'indirizzo: *https://www.sciencedirect.com/science/article/abs/pii/S0167865508002687*, 1/06/2022.

[111] Jasmina Dj. Novakovic, Alempije Veljovic, Sinisa S. Ili, Zeljko Papi e Milica Tomovic, *Evaluation of Classification Models in Machine Learning*, 2017. Disponibile all'indirizzo: *https://www.scirp.org/reference/ReferencesPapers.aspx?ReferenceID=2544204*, 1/06/2022.

[112] A. K. Santra e C. Josephine Christ, *Genetic Algorithm and Confusion Matrix for Document Clustering*, 2012. Disponibile all'indirizzo *https://www.researchgate.net/publication/265119309_Genetic_Algorithm_and_Confusion_Matrix_for_Document_Clustering*, 1/06/2022.

[113] David M. V. Powers, *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*, 2011. Disponibile all'indirizzo: *https://www.researchgate.net/publication/313610493_Evaluation_From_precision_recall_and_fmeasure_to_roc_informedness_markedness_and_correlation*, 1/06/2022.

[114] David M. V. Powers, *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*, 2020. Disponibile all'indirizzo: *https://arxiv.org/abs/2010.16061*, 2/06/2022.

[115] Pierre Baldi, Søren Brunak, Yves Chauvin, Claus A. F. Andersen, Henrik Nielsen, *Assessing the accuracy of prediction algorithms for classification: an overview*, 2000. Disponibile all'indirizzo: *https://academic.oup.com/bioinformatics/article/16/5/412/192336?login=false*, 2/06/2022.

[116] James P. Egan, *Signal detection theory and ROC analysis, Series in Cognition and Perception*, Academic Press, 1975.

[117] John A. Swets, Robin M. Dawes e John Monahan, *Psychological Science Can Improve Diagnostic Decisions*, 2000. Disponibile all'indirizzo: *https://www.jstor.org/stable/40062277?addFooter=false*, 3/06/2022.

[118] John A. Swets, *Measuring the accuracy of diagnostic systems*, 1988. Disponibile all'indirizzo: *https://www.scirp.org/(S(i43dyn45teexjx455qlt3d2q))/reference/referencespapers.aspx?referenceid=1104099*, 4/06/2022.

[119] Kelly H. Zou, A James O'Malley e Laura Mauri, *Receiver-operating characteristic analysis for evaluating diagnostic tests and predictive models*, 2007. Disponibile all'indirizzo: *https://pubmed.ncbi.nlm.nih.gov/17283280/*, 4/06/2022.

[120] Kent A. Spackman, *SIGNAL DETECTION THEORY: VALUABLE TOOLS FOR EVALUATING INDUCTIVE LEARNING*, 1989. Disponibile all'indirizzo: *https://www.sciencedirect.com/science/article/pii/B9781558600362500473*, 4/06/2022.

[121] Wilcoxon-Mann-Whitney, test somma dei ranghi. Disponibile all'indirizzo: *https://www.britannica.com/science/Mann-Whitney-Wilcoxon-test*, 4/06/2022.