Alma Mater Studiorum
Università di Bologna
Campus di Cesena

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea in Ingegneria e Scienze Informatiche

COMPARAZIONE DI ALGORITMI DI MACHINE LEARNING TRADIZIONALI PER IL RICONOSCIMENTO DELLE EMOZIONI VOCALI

Relatore:
Dr. Giovanni Delnevo

Presentata da:
Falconi Eleonora

Correlatore:
Dr.ssa Chiara Ceccarini

Sessione II
Anno Accademico 2023-2024

A chi mi ha sostenuta
e avuto fiducia in me

Introduzione

Il linguaggio parlato è un modo naturale di comunicare tra gli esseri umani ed è anche un segnale complesso che trasporta una grande quantità di informazioni come pensieri, sentimenti, stati d'animo del parlante. Le macchine stanno diventando sempre più importanti in vari scenari per facilitare la vita quotidiana, l'industria, l'istruzione, l'intrattenimento e l'occupazione in una serie di campi. La capacità del computer di riconoscere e reagire alle intenzioni dell'uomo in tempo reale è necessaria per l'interazione uomo-macchina [1]. Il presente volume di tesi, costruito sulle solide basi del progetto di tesi del Dott. Alessandro Zanzi, ha lo scopo di illustrare in modo approfondito i risultati ottenuti dalla comparazione di diversi algoritmi di Machine Learning, una branca fondamentale dell'intelligenza artificiale, per il riconoscimento delle emozioni nel parlato, noto come Speech Emotion Recognition (SER). Questo campo di ricerca mira a identificare e classificare automaticamente le emozioni umane espresse vocalmente, un aspetto che può ampliare le potenzialità delle interazioni uomo-macchina, rendendole più empatiche e sensibili agli stati d'animo degli utenti. L'obiettivo principale di questo lavoro è quello di consolidare e arricchire le conoscenze nel campo del SER, fornendo una base solida per comprendere e valutare le metodologie più diffuse nel Machine Learning, applicate specificamente al problema del riconoscimento delle emozioni vocali. A tal fine, sono state adottate le tecniche più consolidate e riconosciute, come la Cross-validation e la Grid Search, che consentono rispettivamente di valutare l'efficacia dei modelli su dati non visti e di ottimizzare i parametri dei modelli in modo sistematico. In partico-

lare, sono stati utilizzati i principali algoritmi applicati in questo settore, allo scopo di addestrare accuratamente i modelli e analizzare le loro prestazioni nella classificazione delle emozioni all'interno del dataset selezionato per l'addestramento. Questo approccio permette di identificare i punti di forza e le limitazioni di ciascun algoritmo, fornendo una panoramica completa che possa costituire un riferimento utile per futuri sviluppi e applicazioni nel campo del SER.
Il volume di tesi è stato suddiviso in questo modo:

- Il primo capitolo introduce i concetti fondamentali del campo di studio del SER, offrendo una panoramica completa sulle basi teoriche e sui recenti progressi in questo ambito. La sezione iniziale sottolinea l'importanza del SER e le sue potenziali applicazioni nel mondo reale, evidenziando l'impatto che questa tecnologia può avere in diversi settori. Successivamente vengono descritti i vari modelli psicologici e le categorie delle principali emozioni. Infine, viene fornita una descrizione dettagliata delle principali tipologie di caratteristiche audio estraibili, spiegandone l'utilità e il ruolo cruciale nelle applicazioni di analisi del suono.

- Nel secondo capitolo vengono presentate le tecnologie adottate per la preparazione del dataset e l'addestramento dei modelli. Viene fornita una panoramica delle principali librerie utilizzate, distinguendo tra quelle impiegate per la parallelizzazione e serializzazione del codice, come Joblib e Pickle, e quelle dedicate all'estrazione delle feature dal dataset e all'addestramento dei modelli, come Librosa e Scikit-learn. Successivamente, viene descritto il database RAVDESS utilizzato per la creazione del dataset e successivamente analizzato i dati, fase essenziale per preparare i dati agli esperimenti. Infine, si illustrano le diverse tecniche di addestramento dei modelli, mirate a ottimizzare le prestazioni e ottenere i migliori risultati.

- Nel terzo e ultimo capitolo vengono analizzati e discussi i risultati ottenuti nelle diverse fasi degli esperimenti. Nella prima parte si descrive come il dataset è stato suddiviso in training e test set e quali tecniche di data augmentation sono state impiegate per incrementarne le dimensioni di esso. In seguito, vengono presentati i risultati ottenuti durante le varie fasi del processo di addestramento, mettendo a confronto le prestazioni dei modelli per evidenziare il miglioramento ottenuto grazie all'implementazione della Grid Search rispetto alla Cross-validation di default. Le metriche utilizzate come parametri di valutazione sono state fondamentali in questo progetto di tesi, consentendo di confrontare i risultati delle diverse fasi e trarre delle conclusioni finali.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Contesto applicativo

In questo capitolo vengono presentati i fondamenti teorici e le motivazioni alla base dello sviluppo di questo progetto di tesi, fornendo una panoramica delle principali tappe del percorso di ricerca. Si analizza il ruolo delle emozioni nella vita quotidiana e le innovazioni tecnologiche nel campo dell'*intelligenza artificiale* (AI) e degli algoritmi di *Machine Learning* (ML) per il riconoscimento delle emozioni nel parlato, che hanno reso possibile l'interpretazione delle emozioni espresse attraverso la voce da parte di macchine e sistemi.

1.1 Cos è lo Speech Emotion Recognition?

Il *parlato* è uno dei mezzi più significativi e naturali attraverso cui gli esseri umani comunicano le proprie emozioni, stati cognitivi e intenzioni. Negli ultimi decenni, il riconoscimento delle emozioni nel parlato ha attirato molta attenzione nell'ambito dell'informatica incentrata sull'uomo, poiché rappresenta un aspetto fondamentale per comprendere il comportamento emotivo umano. Le crescenti applicazioni dello *speech emotion recognition* (SER), in particolare quelle legate all'interazione uomo-computer e al *computing affettivo*, lo rendono un componente centrale della prossima generazione di sistemi informatici, dove un'interfaccia naturale tra uomo e macchina con-

sentirà l'automazione di servizi che richiedono una buona comprensione dello stato emotivo dell'utente [6]. I sistemi di riconoscimento delle emozioni nel parlato sono metodologie che analizzano e classificano i segnali vocali per identificare le emozioni in essi presenti. Il SER è un campo di studio che si occupa di dedurre le emozioni umane dai segnali vocali, questi sistemi, spesso considerati problemi di classificazione o regressione, si concentrano sull'identificazione dell'input vocale come appartenente a diverse categorie emotive. Attualmente, la ricerca è focalizzata sull'identificazione di combinazioni ottimali di classificatori, mirate a incrementare l'efficacia del riconoscimento emozionale in applicazioni reali [5].

1.1.1 Definizione e Origine del SER

Il SER è un'area di studio emergente nel campo dell'elaborazione del parlato. L'obiettivo del SER è quello di prevedere uno stato affettivo all'interno del discorso, in termini di categorie di emozioni, dimensioni e attributi emotivi o entrambi [7]. Un sistema SER mira a identificare le diverse emozioni della persona che sta parlando, estraendo e classificando le caratteristiche principali da un segnale vocale *pre-elaborato*. Tuttavia, il modo in cui esseri umani e macchine riconoscono e associano gli aspetti emotivi dei segnali vocali è notevolmente diverso sia dal punto di vista quantitativo che qualitativo. Questo infatti può rappresentare una grande sfida per quanto riguarda integrare conoscenze provenienti da campi interdisciplinari diversi, in particolare per quanto riguarda il riconoscimento delle emozioni nel parlato, la psicologia applicata e l'interfaccia *uomo-computer* [8]. Però possiamo constatare che la realizzazione di una macchina che riconosca le emozioni dal parlato non rappresenta un concetto che risale a pochi anni fa. Infatti, le prime ricerche in questo ambito risalgono alla metà degli anni '80, quando furono utilizzate per la prima volta proprietà statistiche di specifiche caratteristiche acustiche. Circa un decennio più tardi, l'evoluzione delle architetture informatiche rese possibile l'implementazione di algoritmi più complessi finalizzati al riconoscimento delle emozioni. Inolte, le crescenti esigenze di mercato per

servizi automatici incentivarono ulteriormente lo sviluppo di questo settore, espandendolo ai più disparati campi [9].

1.1.2 Importanza e Applicazione

Il riconoscimento delle emozioni nel parlato (SER) è emerso come un argomento di ricerca di grande interesse, con applicazioni in diversi settori. Può essere utilizzato nel campo della sanità intelligente, dove lo stato emotivo attuale del paziente viene rilevato attraverso la voce, permettendo così l'erogazione di servizi appropriati. Il SER è utile ad esempio durante la guida, poiché i sistemi SER integrati nel veicolo possono rilevare lo stato emotivo del conducente per garantire misure di sicurezza adeguate, nel rilevamento di false informazioni nel campo delle indagini criminali, nelle diagnosi e nel monitoraggio medico. Oltre a questi ci sono tanti altri settori in cui può avere una reale utilità, e alcuni di questi sono elencati nella Tabella 1.1 [5].

Aree di Applicazione	**Applicazioni Specifiche**
Investigazione	Rilevazione delle bugie
Sanità	Strumento diagnostico, identificazione della depressione, sistemi e-health
Interazione Uomo-Macchina	Risposte nei call center, assistenza robotica, robot sociali, sistemi di assistenza, sistemi conversazionali intelligenti
Educazione	Analisi della salute mentale e del benessere in classe, insegnamento online, sistemi di apprendimento sensibili alle emozioni
Assistenza Intelligente	Pubblicità digitale, giochi online, assistenti smart per la casa
Turismo	Sistemi di raccomandazione per il turismo
Intrattenimento	Miglioramenti nelle performance teatrali e nelle interazioni
Trasporti	Valutazione dello stato emotivo del conducente

Tabella 1.1: Aree di applicazione e applicazioni specifiche del SER [5]

1.2 Le Emozioni nel Parlato

Per implementare con successo un sistema di riconoscimento delle emozioni vocali è necessario definire e modellare accuratamente l'emozione. Secondo *Plutchik*, più di novanta definizioni di emozione sono state proposte nel ventesimo secolo. Le emozioni sono stati psicologici complessi composti da diversi componenti come ad esempio l'esperienza personale, le reazioni fisiologiche, comportamentali e comunicative. In psicologia, le teorie delle emozioni sono raggruppate in quattro principali tradizioni, ognuna delle quali formula diverse ipotesi di base su ciò che è centrale nella natura delle emozioni. Quindi possiamo dire che esistono quattro prospettive che si concentrano su diversi aspetti delle emozioni, i *darwiniani* si occupano dell'organizzazione evolutiva delle emozioni, i *jamesiani* privilegiano l'organizzazione corporea delle emozioni. I teorici dell'emozione cognitiva, invece, sono interessati all'organizzazione psicologica delle emozioni, mentre i costruttivisti sociali si concentrano sull'organizzazione *socio-psicologica* e *sociologica* delle emozioni. Nel campo del riconoscimento delle emozioni nel parlato, diversi gruppi di ricerca utilizzano solitamente differenti stati emotivi. Considerando l'accordo generale nelle varie tradizioni appena citate, secondo cui alcune emozioni complete sono più fondamentali di altre, potrebbe essere preferibile concentrarsi su quelle, che possono essere definite come segue [10]:

- Felicità: la felicità è uno stato emotivo o affettivo caratterizzato da sensazioni di gioia, piacere e soddisfazione.
- Rabbia: la rabbia è spesso una risposta alla percezione di una minaccia dovuta a un conflitto fisico, ingiustizia, negligenza, umiliazione o tradimento.
- Paura: la paura è uno stato emotivo che si esprime quando si avverte un pericolo, una risposta naturale a un particolare stimolo negativo.
- Tristezza: la tristezza è uno stato d'animo che manifesta una sensazione di svantaggio e di perdita. Questa emozione è considerata un

sentimento opposto alla felicità.

- Neutralità: la neutralità è uno stato emotivo, che comprende sonnolenza, calma, rilassamento, soddisfazione, appagamento, ecc.

1.2.1 Modelli Psicologici delle emozioni

Esistono diversi criteri con cui classificare le numerose concezioni attuali dell'emozione, nella discussione che segue, i modelli attualmente utilizzati sono classificati in quattro categorie per evidenziare i principi che sembrano comuni agli approcci prospettici. Sebbene vi sia ovviamente una certa diversità tra i modelli all'interno di ciascuna categoria, si suggerisce che la varietà tra categorie sia significativamente maggiore rispetto alla varianza all'interno della stessa categoria. Va notato che sia la categorizzazione che l'etichettatura sono il risultato dell'analisi personale dell'autore della maggior parte del lavoro teorico e potrebbero non essere condivise da altri teorici.

Modelli dimensionali

Modelli unidimensionali I sostenitori dei modelli unidimensionali ritengono che una singola dimensione sia sufficiente per distinguere gli stati emotivi. Questa dimensione può essere *l'attivazione/eccitazione* o la *valenza*. L'attivazione misura il grado di eccitazione, da basso ad alto, mentre la valenza varia tra emozioni positive e negative. La valenza, che va dal polo negativo (sgradevole) al polo positivo (piacevole), è spesso considerata il principale criterio per la differenziazione emotiva, poiché riflette anche i due orientamenti comportamentali fondamentali: approccio ed evitamento.

Modelli multidimensionali Una delle prime proposte per un sistema multidimensionale fu avanzata da Wundt (1905) [11], il quale sosteneva l'uso di metodi sia introspettivi che sperimentali, utilizzando misurazioni fisiologiche per studiare il sentimento emotivo. Secondo lui la natura dello stato emotivo era determinata dalla sua posizione su tre dimensioni in-

dipendenti: piacevolezza-sgradevolezza, riposo-attivazione e rilassamento-attenzione. Questo approccio influenzò la psicologia delle emozioni, e Schlosberg (1954) [11] introdusse un modello tridimensionale per studiare le espressioni facciali. Successivamente, Russell e Plutchik svilupparono modelli bidimensionali basati su valenza e attivazione, organizzando le emozioni in uno schema circolare (circumplex) per rappresentare graficamente somiglianze e differenze tra le emozioni in termini di vicinanza spaziale.

Modelli delle emozioni discrete

Modelli di circuito I modelli a circuito sono quelli che adottano un approccio neuropsicologico all'emozione e suggeriscono che il numero di emozioni fondamentali e la loro differenziazione sono determinati da circuiti neurali sviluppati evolutivamente. Panksepp [11] sostiene l'esistenza di quattro circuiti fondamentali, o "sistemi" di comando emotivo, che dovrebbero produrre sequenze comportamentali ben organizzate, suscitate dalla stimolazione neurale: la rabbia, la paura, l'aspettativa e il panico. Tuttavia, ci si aspetta che varie interazioni tra questi sistemi possano portare a "stati emotivi di secondo ordine", costituiti da attività combinate tra i sistemi primari.

Modelli delle emozioni di base Le teorie delle emozioni di base suggeriscono l'esistenza di un numero limitato di emozioni fondamentali, come rabbia, paura, gioia, tristezza e disgusto, sviluppatesi come strategie adattive nel corso dell'evoluzione. Queste emozioni, in genere tra 7 e 14, presentano specifiche condizioni di innesco e schemi di reazione fisiologica, espressiva ed emotiva. Plutchik [11] ha identificato un insieme di emozioni di base correlate a classi di motivazione fondamentali, continuando la tradizione etologica. I modelli attuali derivano dal lavoro di Darwin su espressione e universalità delle emozioni, ampliato da Tomkins [11], che ha proposto le emozioni di base come programmi neuromotori stabili attivati automaticamente da condizioni specifiche.

Modelli orientati al significato

Modelli lessicali La struttura dei campi semantici dei termini emotivi è stata spesso utilizzata come base per la costruzione di modelli nella psicologia delle emozioni. L'ipotesi di base è che la saggezza del linguaggio aiuti in qualche modo il teorico a scoprire la struttura sottostante di un fenomeno psicologico. Sebbene non ci sia una corrispondenza perfetta tra il lessico delle emozioni e i processi psicofisiologici inconsci, questo approccio è stato adottato da vari studiosi. Oatley e Johnson-Laird [11](1987) si sono concentrati sulle strutture degli obiettivi, mentre Ortony, Clore e Collins [11] hanno analizzato la struttura semantica del lessico emotivo. Shaver ha utilizzato l'analisi dei cluster per creare alberi di termini emotivi con diversi gradi di generalità. Questi studi cercano di comprendere sia come le persone etichettano gli stati emotivi sia i meccanismi emotivi sottostanti.

Modelli costruttivi sociali I modelli costruttivi sociale sostengono che il significato dell'emozione è generalmente costituito da comportamenti e valori determinati a livello socio-culturale. Sebbene i sostenitori di questo approccio non neghino la presenza di componenti di reazione psicobiologica nell'emozione, considerano queste ultime secondarie rispetto al significato conferito dal contesto socio-culturale, sia in termini di interpretazione della situazione scatenante sia per quanto riguarda il ruolo della reazione emotiva nel processo di comprensione e interazione sociale della persona.

Modelli componenziali

I teorici dei modelli componenziali sostengono che le emozioni emergano da una valutazione cognitiva delle situazioni ed eventi precedenti e che il pattern delle reazioni nei diversi domini di risposta quali fisiologia, espressione, tendenze all'azione e sensazione, sia determinato dall'esito di questo processo di valutazione. Lazarus insieme ad Arnold [11], colui che creò uno dei modelli più stringenti in questo ambito, è stato un pioniere nell'introduzione del concetto di valutazione soggettiva che tiene conto del significato che un

evento assume per l'individuo e della sua capacità di farvi fronte, sulla natura dell'emozione che ne deriva. [11].

1.2.2 Categorizzazione e tassonomie

Le emozioni non solo possono essere distinte da altri fenomeni affettivi, ma all'interno della categoria stessa di "emozione" esistono diverse sottocategorie proposte. A quanto ne sappiamo nessuna tassonomia delle emozioni ha ancora raggiunto un consenso unanime, tuttavia alcune categorie sono considerate utili a livello concettuale. Le tassonomie delle emozioni si basano su vari criteri, e le categorie spesso si sovrappongono perciò non devono essere interpretate come esclusive tra loro, ma piuttosto come diverse modalità di categorizzazione utilizzate nelle diverse tradizioni di ricerca. Infatti, una singola emozione come ad esempio la rabbia può rientrare in più di una categoria.

Emozioni di base Le emozioni di base rappresentano una categoria ampiamente utilizzata nella ricerca attuale in *neuroscienze affettive*. Questo concetto si può definire simile a quello delle emozioni "primarie", "discrete" o "fondamentali" e si basa sull'idea che esista un piccolo gruppo di emozioni, solitamente tra 2 e 10, considerate più elementari rispetto ad altre. Tra le emozioni comunemente ritenute di base troviamo: rabbia, disgusto, paura, gioia, tristezza e sorpresa. Secondo questa teoria, l'aggettivo "di base" viene utilizzato per esprimere tre postulati ideati dallo psicologo Paul Ekman [12]:

1. "Esistono una serie di emozioni distinte che differiscono l'una dall'altra in modi importanti"

2. "L'evoluzione ha svolto un ruolo importante nel plasmare sia le caratteristiche uniche e comuni che queste emozioni mostrano, sia la loro funzione attuale"

3. " Le emozioni non basilari sono costituite da miscele di emozioni basilari"

La maggior parte del lavoro nelle neuroscienze affettive nell'ultimo decennio è concentrata nella ricerca di sistemi cerebrali specifici dedicati a ciascuna emozione di base, utilizzando come prova sia dissociazioni neuropsicologiche doppie sia risultati di neuroimaging.

Emozioni autoriflessive (o autocoscienti) Le *emozioni autoriflessive*, come vergogna, imbarazzo, senso di colpa e orgoglio, si distinguono per il coinvolgimento del sé, piuttosto che per una preoccupazione di sopravvivenza. Note anche come *emozioni autocoscienti* o *emozioni morali*, sono sempre più studiate nelle neuroscienze affettive, dato il crescente interesse per le emozioni non basilari e il sé. Le neuroscienze possono aiutare a chiarire le differenze tra queste emozioni, come vergogna e senso di colpa, fornendo evidenze empiriche utili ai dibattiti su somiglianze e differenze.

Emozioni positive vs negative La distinzione tra emozioni "positive" e "negative" si basa spesso sulla valenza, ossia sulla percezione soggettiva di piacere o disagio. Tuttavia, questa distinzione non riguarda solo le sensazioni, ma anche gli eventi scatenanti, classificati come positivi o negativi in base alla loro piacevolezza o capacità di raggiungere obiettivi. Alcuni studiosi evidenziano che le valutazioni degli eventi possono essere ambivalenti, portando a provare sia piacere che dispiacere contemporaneamente.

Emozioni sociali Le emozioni sociali, come vergogna, invidia, senso di colpa e gratitudine, sono suscitate da situazioni sociali, spesso in presenza di altri individui reali o immaginari, e servono a regolare il comportamento o raggiungere obiettivi sociali. La neuroscienza sociale si concentra su come le emozioni sono modulate dal contesto sociale. Ad esempio, la paura, pur non essendo un'emozione sociale, può essere influenzata dalle valutazioni degli altri. Gli studi che hanno confrontato il modo in cui il cervello calcola le informazioni emotive sociali rispetto a quelle non sociali suggeriscono che alcune regioni critiche per l'elaborazione delle emozioni sono coinvolte anche nell'elaborazione della rilevanza sociale.

Emozioni morali Le emozioni morali sono quelle emozioni suscitate da valutazioni morali, come ad le norme morali, gli obblighi morali, il bene e il male morale, i valori morali e le virtù morali. Sono state proposte diverse classificazioni di questa tipologia di emozioni, per esempio secondo Haidt, si possono distinguere quattro tipi di emozioni morali [12]:

1. Quelle legate alla coscienza di sé, come la vergogna e senso di colpa
2. Quelle che condannano gli altri, quali disprezzo, rabbia e disgusto
3. Quelle suscitate dalla sofferenza degli altri, come la compassione
4. Quelle che lodano gli altri, come la gratitudine e l'elevazione

1.2.3 Caratteristiche Vocali e Parametri

Il suono è classificato in suoni *vocali* e *non vocali*. I suoni vocali vengono prodotti grazie alla vibrazione delle corde vocali. Queste vibrazioni creano delle periodicità nel segnale vocale, che danno origine alla *frequenza fondamentale* (F_0) o *altezza tonale*. In termini semplici, viene chiamata la frequenza di *cutoff* alla quale le corde vocali tendono a vibrare. F_0 è inversamente proporzionale alla distanza tra le periodicità, il che rende il suono vocale con frequenza più alta o più bassa. D'altra parte, nel parlato non vocale non avviene alcuna vibrazione delle corde vocali, quindi non ci sono periodicità, perciò può essere modellato come un segnale di rumore filtrato. Pertanto la curva dell'altezza tonale esiste solo nei segmenti vocali.

Intonazione

Il *pitch* è una variabile che cambia continuamente durante l'eloquio, a seconda di fattori come emozioni, età e genere. Questo fenomeno viene chiamato *intonazione* (l'andamento ascendente e discendente del pitch) o la presenza di un pitch alto o basso. L'intonazione è la principale caratteristica vocale che ci aiuta a distinguere tra emozioni e identificare differenze tra fra-

si dichiarative e interrogative, nonché individuare le parole rilevanti in una frase.

Durata

Il secondo parametro più importante della prosodia è la *durata*, che conferisce naturalezza al parlato. La durata dei *fonemi* dipende da molteplici fattori, come la loro identità fonetica, i fonemi vicini, il livello di stress, e la loro posizione nella frase o nella parola. Inoltre, la durata delle parole varia in base alla loro priorità all'interno della frase: le parole significative o enfatizzate tendono ad avere una durata maggiore.

Articolazione

L'articolazione non solo aiuta l'ascoltatore a percepire la parola, ma facilita anche la comprensione. La *velocità* o il *tasso di movimento* degli articolatori, ossia le parti del sistema vocale che partecipano alla produzione dei suoni, è definita come velocità di articolazione. Questa velocità varia in funzione delle emozioni, dell'età, della lingua e del genere. Lo studio della velocità di articolazione aiuta a comprendere le emozioni e la prosodia del parlato.

Intensità

L'intensità di un segnale vocale definisce la quantità di energia con cui viene prodotto il parlato. Anche l'intensità varia significativamente a seconda delle emozioni. L'intensità è utilizzata per codificare l'informazione prosodica. Fisiologicamente, le variazioni nell'intensità sono parzialmente correlate alle variazioni della F_0 per gli accenti naturali, ma recenti studi hanno dimostrato che le variazioni di intensità sono indipendenti dalle variazioni di F_0. L'intensità vocale viene misurata su unità prosodiche a breve e lungo termine.

Vibrazione

Le vibrazioni delle corde vocali causano vibrazioni anche nel tratto vocale. La frequenza a cui vibra il tratto vocale è chiamata *frequenza di risonanza* o *formanti*. I formanti vengono misurati attraverso il contenuto in frequenza dei suoni vocali, che contengono le informazioni necessarie all'orecchio umano per distinguere tra i vari suoni delle vocali.

Epoch

L'*Epoch* è definito come l'impulso significativo di eccitazione del sistema del tratto vocale durante il parlato, questo avviene quando le corde vocali si chiudono rapidamente e creano un'esplosione di pressione dell'aria che genera un impulso energetico. Questi momenti si verificano attorno agli GCI (glottal closure instants). I suoni non vocali non presentano epoch poiché non ci sono vibrazioni delle corde vocali, e quindi non c'è una forza di eccitazione impulsiva. La modifica della prosodia è possibile rilevando la posizione degli epoch nel parlato espressivo o emozionale. [13]

1.3 Tecniche di estrazione delle Feature

Le caratteristiche di un campione vocale sono cruciali nel riconoscimento delle emozioni, poiché una combinazione ben progettata migliora significativamente il tasso di riconoscimento. Il parlato, un segnale continuo e variabile, veicola informazioni ed emozioni permettendo l'estrazione di caratteristiche globali o locali in base all'approccio. Le *caratteristiche globali* (o *a lungo termine*) includono statistiche generali come media, valori minimi e massimi e deviazione standard. Le caratteristiche locali (o *a breve termine*) catturano le dinamiche temporali, avvicinandosi a stati *stazionari* per cogliere meglio le emozioni che variano nel tempo. Per esempio, la rabbia è spesso più evidente all'inizio di un enunciato, mentre la sorpresa si manifesta di solito alla fine. Queste caratteristiche locali e globali dei sistemi SER vengono analizzate in quattro categorie principali:

- *Caratteristiche della qualità vocale*
- *Caratteristiche spettrali*
- *Caratteristiche prosodiche*
- *Caratteristiche basate sull'operatore di energia di Teager (TEO)*

Le caratteristiche prosodiche e spettrali sono quelle più comunemente utilizzate nei sistemi SER. Alcune delle caratteristiche vengono elencate in categorie diverse da vari studi a seconda dell'approccio adottato. Le caratteristiche TEO sono specificamente progettate per riconoscere lo stress e la rabbia [14].

1.3.1 Feature qualitative

La qualità del suono nel riconoscimento delle emozioni nel parlato ha un impatto significativo. Molti ricercatori sostengono che l'eccellenza nella qualità del segnale sia influenzata dagli stati emotivi. Sebbene la qualità del suono sia un indicatore utile dell'emozione, non è considerata una caratteristica standard nel riconoscimento delle emozioni nel parlato. Di conseguenza, i risultati spesso non riescono a identificare correttamente la descrizione della qualità del suono nelle varie emozioni, come eccitazione, rabbia e paura, portando a disaccordi tra i ricercatori. A tal proposito, Scherer ha suggerito che un suono intenso può derivare dalla rabbia, dalla felicità o dalla paura, con il risultato di una qualità sonora spesso disturbante. Le caratteristiche più comuni del suono utilizzate nel riconoscimento delle emozioni nel parlato sono lo shimmer e il jitter.

Jitter

Il jitter si riferisce a una deviazione a breve termine della frequenza fondamentale del parlato. Alcuni ricercatori hanno osservato il segnale vocale su un oscilloscopio, notando che nessun periodo di questo segnale è identico e che la frequenza è pari alle variazioni che definiscono il jitter.

$$\text{Jitter}(i) = \frac{|F_0(i) - F_0(i-1)|}{\text{mean}\,\{F_0(i) \,|\, i = 2, 3, \ldots, n\}} \tag{1.1}$$

La formula 1.1 esprime la variazione relativa della frequenza fondamentale tra cicli successivi rispetto alla media, ed è utilizzato per identificare fluttuazioni nella stabilità della voce.

Shimmer

Lo shimmer rappresenta una deviazione a breve termine nell'ampiezza. Questa caratteristica riflette le variazioni di energia nel parlato, indicando così diversi livelli di attivazione.

Lo shimmer viene calcolato tramite la radice quadrata dell'energia media quadratica (RMS). L'energia RMS è ottenuta dall'equazione (1.2), in cui $E(i)$ rappresenta l'energia del frame i e $S(k)$ è il valore dei campioni del segnale vocale in questo frame dopo l'applicazione del windowing.

$$E(i) = \sqrt{\frac{1}{K} \sum_{k=1}^{K} S^2(k)} \tag{1.2}$$

$E(i)$ rappresenta l'ampiezza del segnale vocale al ciclo i e $S(k)$ è il valore del segnale al campione k, e K è il numero totale di campioni in un ciclo. Ora possiamo calcolare lo shimmer come nell'equazione 1.5 [2].

$$\text{Shimmer}(i) = \frac{|E(i) - E(i-1)|}{\text{mean}\,\{E(i) \,|\, i = 2, 3, \ldots, n\}} \tag{1.3}$$

Questa formula misura la variazione relativa dell'ampiezza tra due cicli consecutivi, $E(i)$ ed $E(i-1)$. Il numeratore calcola la differenza assoluta tra le ampiezze dei cicli successivi, indicando la variabilità dell'intensità. Il denominatore normalizza questa variazione rispetto alla media delle ampiezze su tutti i cicli i per ottenere una misura normalizzata della fluttuazione dell'intensità.

1.3.2 Feature spettrali

Le proprietà spettrali del segnale vocale vengono estratte per arricchire le caratteristiche fondamentali, offrendo una rappresentazione complementare. Queste caratteristiche includono il contenuto in frequenza del parlato, che presenta variazioni distintive a seconda delle emozioni espresse. Le principali caratteristiche spettrali impiegate nel riconoscimento delle emozioni comprendono: MFCC, LPC, LPCC, GFCC e PLP.

Coefficiente Cepstrale in Frequenza Mel

L'MFCC è una caratteristica fondamentale del parlato che cattura informazioni rilevanti dai segnali vocali e fornisce una descrizione concisa dello spettro del segnale. Il calcolo degli MFCC prevede la suddivisione dei segnali vocali in segmenti temporali, e per ciascun segmento viene generato un vettore MFCC. Quindi un vettore MFCC, di dimensione d, viene impiegato come input per il sistema diagnostico per ogni campione, dove d rappresenta il numero di coefficienti MFCC. I coefficienti MFCC si basano su caratteristiche dell'orecchio umano nella percezione e comprensione del parlato, ciò ha reso i coefficienti potenti in tutte le aree di elaborazione e riconoscimento del parlato. Di seguito verrano descritte le fasi di estrazione dell'MFCC da un segnale audio, come mostrato in Figura 1.1:

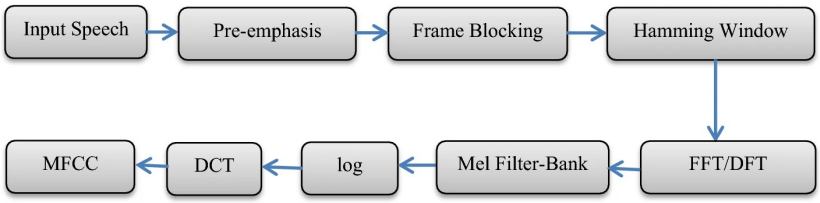

Figura 1.1: Diagramma a blocchi del processo di estrazione dell'MFCC [2]

Pre-emphasis la pre-enfasi è un processo di filtraggio di alta precisione per amplificare l'energia delle frequenze più alte. Questo filtro può essere applicato sia nel dominio della frequenza che in quello del tempo. Per quanto

riguarda il dominio temporale, il filtro può essere definito tramite la seguente equazione (1.6):

$$y_n = x_n - \alpha x_{n-1}, \quad 0.9 \leq \alpha \leq 1.0 \tag{1.4}$$

Il valore corrente y_n è dato dalla differenza tra il valore attuale del segnale x_n e una frazione α del valore precedente x_{n-1}. Il parametro α assume un valore di 0.95, la funzione di trasferimento può essere espressa come segue:

$$H(z) = 1 - \alpha z^{-1} \tag{1.5}$$

L'equazione 1.5 è il calcolo di un filtro a risposta infinita all'impulso (IIR) di primo ordine. Il fattore z^{-1} indica un ritardo di un campione. In sostanza, questa funzione di trasferimento definisce come il filtro modifica l'input in base ai valori precedenti, con α che controlla il peso assegnato ai campioni precedenti.

Framing durante la fase di frammentazione avviene la segmentazione del campione vocale in fotogrammi da 20-40 ms. Questo passaggio è necessario poichè il segnale vocale varia nel tempo, ma su brevi intervalli può essere considerato stazionario, ciò permette una più accurata analisi delle caratteristiche del segnale.

Windowing La finestra di Hamming è utilizzata per prelevare i campioni dal segnale e successivamente moltiplicarli per la funzione di finestratura. La finestra di Hamming è calcolata attraverso l'equazione seguente:

$$w[n] = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{L}\right) & 0 \leq n \leq L-1 \\ 0 & \text{altrimenti} \end{cases} \tag{1.6}$$

Dove la "finestratura" è applicata ai campioni $w[n]$ del segnale all'interno dell'intervallo $0 \leq n \leq L-1$, con una riduzione graduale dei valori ai bordi, mentre al di fuori di questo intervallo la finestra assume valore zero.

Discrete Fourier Transform(DFT) La trasformata discreta di Fourier (DFT) è una tecnica fondamentale nell'analisi dei segnali digitali. Essa permette di trasformare un segnale campionato nel dominio del tempo in un insieme di componenti frequenziali, consentendo di analizzare le frequenze presenti nel

segnale originale. La DFT prende come input una sequenza di campioni di un segnale e restituisce un'altra sequenza che rappresenta le ampiezze e le fasi delle frequenze che compongono quel segnale. In altre parole, essa scompone un segnale complesso in una somma di onde sinusoidali di varie frequenze, facilitando l'identificazione delle frequenze dominanti e la loro intensità.

$$H_k = \sum_{n=0}^{N-1} h_n e^{-i\frac{2\pi kn}{N}}, \quad k = 0, \ldots, N-1 \tag{1.7}$$

Nell'equazione 1.7, N è la lunghezza della sequenza e k è l'indice delle componenti di frequenza. La DFT converte la sequenza temporale h_n in una rappresentazione spettrale H_k, che descrive la distribuzione delle frequenze nella sequenza originale.

Mel scale filter bank Il Mel Scale Filter Bank è un insieme di filtri triangolari applicati a uno spettrogramma per convertire le frequenze lineari di un segnale audio in una scala Mel, che riflette meglio la percezione uditiva umana. I filtri sono distribuiti in modo più denso nelle basse frequenze, dove l'orecchio umano è più sensibile, e meno denso nelle alte frequenze. Ogni filtro calcola l'energia del segnale in una banda di frequenze specifica. Questo processo è fondamentale per l'estrazione delle caratteristiche vocali, poichè catturano informazioni rilevanti per il riconoscimento del parlato e delle emozioni. E' definito come segue:

$$\text{mel}(f) = 1127 \ln\left(1 + \frac{f}{700}\right) \tag{1.8}$$

Nell'eq. 1.8, f è la frequenza in Hertz, mentre m è la frequenza corrispondente nella scala Mel. La trasformazione è logaritmica, poiché l'orecchio umano percepisce meglio le differenze nelle basse frequenze rispetto alle alte. La costante 700 e la base logaritmica riflettono questa caratteristica percettiva.

Discrete Cosine Transform (DCT) La trasformata discreta coseno è un'operazione matematica utilizzata per trasformare un segnale discreto nel dominio del tempo in un dominio delle frequenze. Essa rappresenta il segnale come una somma di coseni con frequenze diverse, consentendo di separare le

informazioni a bassa frequenza da quelle ad alta frequenza. La DCT è particolarmente efficace per la compressione dei dati e l'analisi spettrale, poiché concentra la maggior parte dell'energia del segnale nelle prime componenti, riducendo così la dimensionalità dei dati. La DCT viene applicata per ottenere i coefficienti cepstrali non correlati attraverso l'Eq.1.11 [2].

$$g\left(n:u\right)=\left(\frac{2}{m}\right)^{0.5}\sum_{i=0}^{M-1}\left\{\frac{1}{3}\log\left(\overline{y}\left(n:i\right)\right)\cos\left[\frac{\pi u}{2M}\left(2i-1\right)\right]\right\} \tag{1.9}$$

Dove, il termine $\left(\frac{2}{m}\right)^{0.5}$ funge da fattore di normalizzazione, mentre la somma $\sum_{i=0}^{M-1}$ considera M componenti del segnale. Il termine $\overline{y}(n:i)$ rappresenta una trasformazione del segnale, mentre l'uso del logaritmo log viene utilizzato per ridurre la variabilità di un insieme di valori che possono coprire un intervallo molto grande. La funzione coseno modula il segnale permettendo di analizzare frequenze specifiche.

Mel-Spectrogram

Il Mel Spectrogram o anche chiamato *spettro di Mel* è una rappresentazione delle frequenze di un segnale audio, sviluppata per riflettere meglio la percezione uditiva umana. A differenza di una scala lineare, le frequenze sonore prodotte dalla voce umana non seguono un andamento lineare, e per questo viene utilizzata la scala Mel. Questa scala trasforma le frequenze lineari, mantenendole tali sotto 1 kHz, mentre sopra questa soglia viene applicata una scala logaritmica. Il risultato del calcolo del *Mel-frequency spectrum* è una distribuzione delle frequenze con campioni equilibrati, dove ogni campione rappresenta una banda di frequenze significativa per l'orecchio umano. La *frequenza mel* è definita così:

$$\mathrm{Mel}\left(f\right)=2595\log_{10}\left(1+\frac{f}{700}\right) \tag{1.10}$$

Chroma

Il chroma è una tecnica di estrazione delle caratteristiche che si concentra sui toni musicali presenti in file audio, questa caratteristica fornisce una distribuzione delle variazioni tonali nell'audio sotto forma di un semplice vettore. Il risultato dell'analisi chroma è un cromagramma, che viene costruito in base a 12 livelli di tonalità. L'utilizzo della chroma è utile per riconoscere l'altezza tonale (alta o bassa) della voce dell'attore in un file audio, poiché il tono del parlato può aiutare a riconoscere specifiche emozioni [15].

Spectral contrast

Il *contrasto spettrale* di un suono misura la differenza di energia tra i *picchi*(valori più alti) e le *valli*(valori più bassi), all'interno di queste bande di frequenza. Nell'estrazione delle caratteristiche, il campione audio viene prima suddiviso in segmenti tramite una finestra di analisi di 200 ms con una sovrapposizione di 100 ms. Poi per ogni segmento, viene eseguita la trasformata di Fourier per ottenere le componenti spettrali, che vengono poi suddivise in sei sottobande. Per garantire la stabilità della caratteristica del contrasto spettrale, la forza dei picchi e delle valli spettrali è stimata come valore medio in un piccolo intorno dei valori massimi e minimi.

Tonnetz

Il *tonnetz* è uno spazio delle altezze definito dalla rete di relazioni tra le altezze musicali nell'intonazione naturale. Le relazioni armoniche vicine vengono modellate come distanze brevi su un piano euclideo infinito, mentre gli accordi diventano strutture geometriche sul piano. Le tonalità invece vengono definite da regioni nella rete armonica.

Linear Prediction Coefficient (LPC)

I *Coefficienti di predizione lineare* sono uno degli strumenti più potenti nell'elaborazione del parlato. L'idea generale di questa analisi è che ogni

campione di un segnale audio può essere descritto come un'equazione lineare in termini di ingressi e uscite del campione precedente. Il metodo utilizzato per modellare il condotto vocale umano consiste nell'usare un filtro chiamato *filtro sorgente*. L'LPC rappresenta le caratteristiche del segnale desiderato. Di solito vengono utilizzati per codificare il segnale audio, poiché la maggior parte delle caratteristiche del segnale dell'oggetto selezionato vengono sfruttate. Questi coefficienti pratici sono simili alla previsione di un campione a partire dai campioni audio precedenti; ad esempio, avendo 14 segnali audio prototipo, è possibile prevedere il campione successivo utilizzando questi coefficienti.

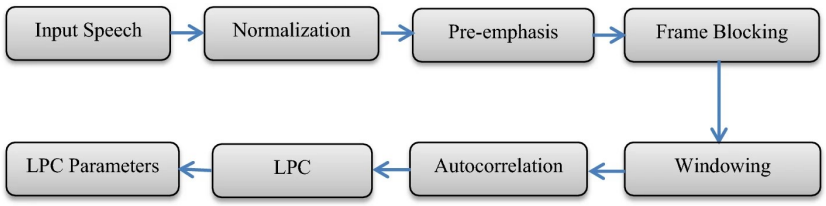

Figura 1.2: Diagramma a blocchi della generazione dell'LPC [2]

Linear Prediction Cepstral Coefficient (LPCC)

Il *coefficiente cepstrale di predizione lineare* è stato ampiamente utilizzato per la sua affidabilità e robustezza rispetto all'LPC. Gli LPCC sono calcolati tramite il metodo dell'autocorrelazione a partire dai coefficienti LPC, come illustrato in Figura 1.3, e catturano informazioni esplicite delle caratteristiche vocali. Rispetto all'LPC, l'LPCC riflette meglio l'andamento del logaritmo dello spettro vocale. La loro stima avviene in modo ricorsivo, utilizzando tutti i coefficienti successivi del filtro. Prima dell'elaborazione, il segnale viene normalizzato per ridurre l'errore causato dal rumore additivo, poiché l'LPCC è più sensibile al rumore rispetto all'MFCC. Tuttavia, la vulnerabilità al rumore rappresenta un limite significativo.

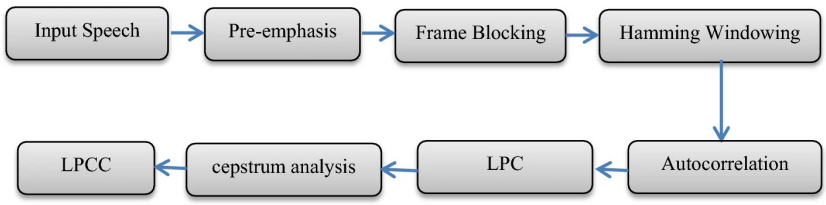

Figura 1.3: Diagramma a blocchi della generazione dell'LPCC [2]

Gamma tone Frequency Cepstral Coefficient (GFCC)

I *coefficienti cepstrali* della frequenza *gamma tone* sono caratteristiche uditive basate su un insieme di banchi di filtri "gamma tone". I coefficienti GFCC vengono calcolati utilizzando una tecnica simile a quella degli MFCC, applicando però i filtri gamma tone sulle energie delle diverse sotto-bande al posto del Mel-filter bank. Nella Figura 1.4 viene raffigurato il diagramma completo della generazione del GFCC da un segnale audio. Un filtro gamma

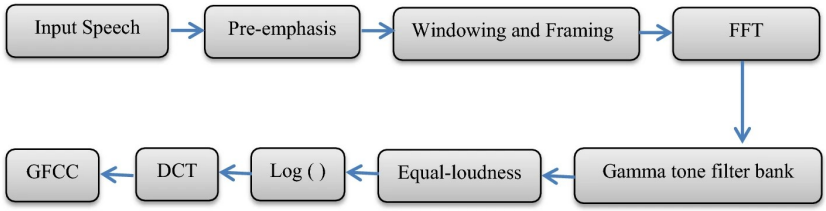

Figura 1.4: Diagramma a blocchi della generazione del GFCC [2]

tone, in base alla frequenza centrale, può essere definito come:

$$g\left(t\right) = at^{n-1}e^{-2\pi bt}\cos(2\pi f_c t + \varphi) \tag{1.11}$$

φ è la *fase*, ma solitamente viene impostata a zero. La costante a controlla il guadagno, mentre il valore di n è generalmente inferiore a 4. Il fattore b è definito come segue:

$$b = 25.17\left(\frac{4.37 f_c}{1000} + 1\right) \tag{1.12}$$

Successivamente, viene applicata la Trasformata Coseno Discreta (DCT), come già fatto nell'operazione per gli MFCC e secondo l'equazione precedente (Eq. 1.11).

I valori tipici di u variano da 0 a 31, e vengono selezionati i primi 12 elementi per una caratteristica GFCC a 12 dimensioni, indicate come $g(n:0), g(n:1), \ldots, g(n:11)$ (Eq.1.13). $g(n)$ rappresenta un insieme di caratteristiche o parametri estratti in un determinato istante n, rendendo il vettore $g(n)$ utile per analisi successive:

$$g\left(n\right) = \left[g\left(n:0\right), \ldots, g\left(n:11\right)\right]^T \tag{1.13}$$

Perceptual Linear Prediction (PLP)

Il metodo di calcolo dei coefficienti della *predizione lineare percettiva* deriva dalla percezione umana. Questa tecnica tiene conto del fatto che la risoluzione in frequenza e la sensibilità ai cambiamenti di frequenza nell'orecchio umano variano a seconda delle frequenze. Inoltre, la sensibilità dell'orecchio all'intensità sonora varia in base alle frequenze: la sensazione percepita è proporzionale alla radice cubica dell'energia del suono. La tecnica PLP si basa sullo spettro a breve termine, ma come altre tecniche di questo tipo, è sensibile agli effetti dei canali di telecomunicazione sui valori spettrali a breve termine. La Figura 1.5 mostra un diagramma PLP per l'analisi del parlato precedentemente proposto.

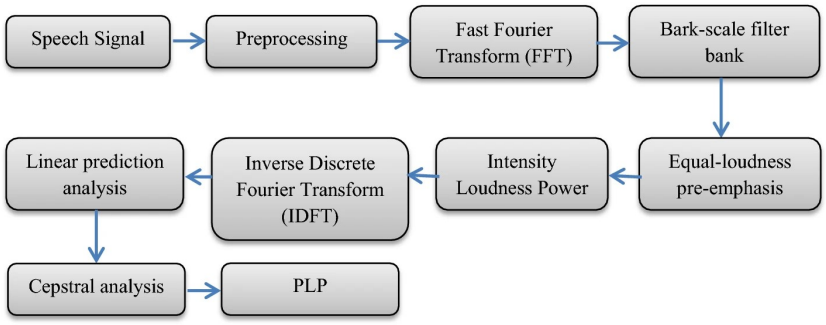

Figura 1.5: Diagramma a blocchi della generazione del PLP [2]

1.3.3 Feature Prosodiche

Le caratteristiche prosodiche sono tra le più comuni per il riconoscimento delle emozioni nel parlato. Sebbene richiedano un minor carico computazionale rispetto alle caratteristiche spettrali, sono spesso influenzate da fattori ambientali e utilizzate insieme a parametri spettrali per migliorare l'accuratezza. Tra le caratteristiche prosodiche principali vi sono energia, tasso di zero-crossing e frequenza del pitch f_0, che esprimono in modo significativo le emozioni del parlante. Le caratteristiche prosodiche per l'espressione pongono l'accento sulle affermazioni e spesso riguardano l'ampiezza del segnale vocale e le variazioni di frequenza del segnale audio. Queste caratteristiche includono informazioni sul ritmo e il tono del discorso e si basano su tempo, frequenza del pitch ed energia. Alcune delle loro caratteristiche generali includono [2]:

- **Tono medio:** valore medio di f_0 (frequenza fondamentale) per le dichiarazioni.
- **Intervallo di dominio:** intervallo tra i domini di f_0 (frequenza fondamentale).
- **Degradazione finale:** diminuzione della pendenza di f_0 (frequenza fondamentale) alla fine del taglio finale o in un incremento crescente.
- **Linea di riferimento:** valore fisso di f_0 (frequenza fondamentale) dopo aver raggiunto i toni più alti e più bassi.

Energy

L'energia è la caratteristica più importante dei segnali vocali, che definisce i confini tra il parlato e il silenzio. L'energia di ciascun fotogramma, in base alla relazione che segue, indica la diversità della gamma dei segnali vocali a breve termine. Il contenuto del segnale a diverse distanze indica l'intensità

del suono percepito dall'orecchio umano.

$$E_n = \sum_{m=-\infty}^{\infty} [x(m)W(n-m)]^2 \tag{1.14}$$

La formula rappresenta l'energia E_n di un segnale $x(m)$ filtrato da una finestra $W(n-m)$, dove n è l'indice temporale corrente e m è l'indice della sommatoria. L'energia è calcolata come la somma dei quadrati dei valori filtrati su tutti i possibili valori di m, dall'infinito negativo all'infinito positivo.

Zero-crossing rate

Il tasso di attraversamento dello zero nei segnali audio può essere impiegato per distinguere le porzioni di silenzio dalle parti con dialogo. Il tasso di attraversamento dello zero in un frame contenente n campioni è determinato dall'equazione (1.15)

$$\text{ZCR} = \frac{1}{N} \sum_{n=0}^{N-1} \frac{|\text{sgn}\,[x(n)] - \text{sgn}\,[x(n-1)]|}{2} \tag{1.15}$$

In particolare, $x(n)$ è il segnale audio campionato, e N è il numero totale di campioni considerati. La funzione sgn(x) restituisce il segno di x, che è 1 se $x > 0$, -1 se $x < 0$, e 0 se $x = 0$. La formula conta il numero di transizioni tra il segno positivo e negativo del segnale, normalizzandolo rispetto al numero totale di campioni N. Un valore elevato di ZCR può indicare un segnale più "frastagliato", mentre un valore basso può suggerire una maggiore continuità nel segnale.
Nei segnali discreti, l'attraversamento dello zero si verifica quando campioni consecutivi presentano segni algebrici opposti e può essere determinato utilizzando la formula (1.16). La sequenza dei campioni $x(n)$ e la funzione segno, *sgn*, sono definiti come segue:

$$\text{sgn}\,[x(n)] = \begin{cases} -1, & x(n) \leq 0 \\ 1, & x(n) > 0 \end{cases} \tag{1.16}$$

La funzione 1.16 restituisce -1 se $x(n)$ è minore o uguale a zero, e 1 se $x(n)$ è maggiore di zero.

Frequency of Pitch

La distanza temporale tra due vibrazioni sonore intense consecutive, nota come periodo di intonazione, e il numero di vibrazioni per unità di tempo definiscono la frequenza fondamentale. Il pitch del segnale può essere determinato calcolando l'*autocorrelazione*, la funzione di autocorrelazione per un segnale circolare è data dalla seguente espressione:

$$r(\eta) = \lim_{M\to\infty} \frac{1}{2N+1} \sum_{n=-M}^{M} X(n)x(n+\eta) \tag{1.17}$$

L'equazione 1.17 esprime la correlazione tra due segnali $X(n)$ e $x(n+\eta)$, dove η rappresenta un ritardo. La correlazione è calcolata come il limite, quando M tende all'infinito, della media dei prodotti dei valori dei segnali $X(n)$ e $x(n+\eta)$ su un intervallo di $2M+1$ campioni, centrato su $n=0$.
L'informazione periodica relativa al pitch del parlato è principalmente determinata dalla frequenza del pitch. Più alta è la frequenza del pitch, più acuto sarà il suono, mentre una frequenza più bassa corrisponde a un suono più grave. L'analisi dell'autocorrelazione è uno dei metodi tradizionali per stimare la frequenza del pitch nel parlato. Questa frequenza, indicata come F_0, si aggira intorno ai 50-250 Hz negli uomini, 150-450 Hz nelle donne e 300-700 Hz nei bambini [2].

1.3.4 Feature dell'operatore energetico Teager

Il *Teager Energy Operator* (TEO) è un operatore introdotto da Teager e Kaise [14] per analizzare il parlato e rilevare lo stress attraverso un processo non lineare di interazione tra vortici e flusso d'aria nel tratto vocale umano. Secondo Teager, lo stress influisce sulla tensione muscolare di chi sta parlando, modificando il flusso d'aria durante la produzione dei suoni vocalizzati. L'operatore sviluppato da Teager per misurare l'energia del segnale vocale

attraverso questo processo non lineare è stato documentato da Kaiser, dove Ψ rappresenta l'operatore di energia di Teager e $x(n)$ è il segnale vocale campionato. L'operazione è definita come segue:

$$\Psi[x(n)] = x^2(n) - x(n+1) \cdot x(n-1) \tag{1.18}$$

La formula esprime una funzione $\Psi[x(n)]$ che calcola un valore basato sull'input $x(n)$, che rappresenta il segnale a un determinato istante di tempo n. Il primo termine, $x^2(n)$, rappresenta il quadrato del valore del segnale in n, evidenziando l'importanza della magnitudine del segnale. Il secondo termine, $x(n+1) \cdot x(n-1)$, rappresenta il prodotto dei valori del segnale nei punti temporali immediatamente successivi e precedenti a n.
Zhou et al. [14] hanno introdotto tre nuove caratteristiche basate sul TEO per l'analisi dello stress nel parlato:

- TEO-FM-Var (frequency modulation),
- TEO-Auto-Env (normalized TEO auto-correlation envelope area),
- TEO-CB-Auto-Env (critical band based TEO auto-correlation envelope area).

Tra queste, la caratteristica *TEO-CB-Auto-Env* si è dimostrata particolarmente efficace nel rilevare lo stress rispetto a caratteristiche tradizionali come pitch e MFCC (Mel-Frequency Cepstral Coefficients). Questo approccio è stato testato nel contesto di classificazioni di stress su dataset come SUSAS, dimostrando che TEO-CB-Auto-Env è in grado di superare i metodi classici nelle condizioni di stress vocale [14].

1.3.5 Approcci di Estrazione Basati su Deep Learning

Le reti *neurali profonde*, ispirate al cervello umano, sono fondamentali nel riconoscimento delle emozioni da discors, ma la variabilità emotiva in diverse situazioni rappresenta una sfida. Le *reti neurali convoluzionali* (CNN), utilizzate principalmente nella visione artificiale, hanno dimostrato grande

efficacia. Uno studio ha ottenuto una precisione dell'84,3% con una CNN profonda, composta da tre strati convoluzionali e tre completamente connessi, superando modelli precedenti. Nel SER, sono comuni anche le *reti neurali profonde* (DNN), le RNN e gli *autoencoder*, ciascuno con vantaggi specifici. Nella Figura 1.6 troviamo il confronto tra i processi dei due differenti approcci di estrazione. La difficoltà maggiore nell'analisi delle emozioni sta nell'incertezza e nell'ambiguità delle caratteristiche rilevanti, che dipendono da vari fattori come frasi, interlocutore, velocità e modalità del discorso. Di seguito sono riportati i tipi di reti neurali profonde utilizzati per il riconoscimento delle emozioni dal parlato:

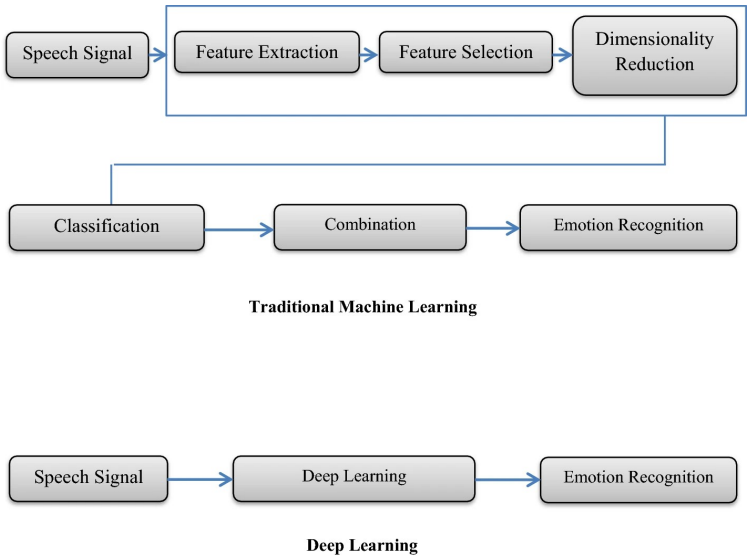

Figura 1.6: I componenti principali del sistema di riconoscimento delle emozioni [2].

Rete neurale a convoluzione

La Convolutional Neural Network(CNN) è una rete neurale profonda che si allena in due fasi: *feedforward* e *back-propagation*. Nel feedforward, il segnale di input (ad esempio, un segnale audio intrusivo) attraversa i vari strati della rete, dove avvengono operazioni di convoluzione e la rete calcola

l'output. L'errore rispetto alla risposta corretta viene poi calcolato, e nella fase di back-propagation i parametri vengono aggiornati per ridurre questo errore, il processo viene ripetuto per molte iterazioni. Una CNN è composta da tre tipi principali di strati:

- *Livello di convoluzione*: che elabora i dati in forma tridimensionale
- *Livello di pooling*: che riduce la dimensione spaziale per diminuire i parametri e i calcoli
- *Livello completamente connesso*: dove tutti i neuroni dell'ultimo strato sono collegati tra loro.

La figura 1.7 illustra il modello della rete neurale convoluzionale.

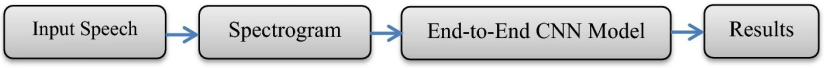

Figura 1.7: Modello di reti neurali convoluzionali [2].

Multi-layer perceptron

La Multilayer Perceptron (MLP) è una rete neurale con una struttura multilivello a carattere multidimensionale, progettata per gestire i dati in modo complesso. Può essere descritta come una rete con un livello di input, un livello di output e più livelli nascosti intermedi. Seguendo una tecnica chiamata "gerarchia delle caratteristiche", ogni livello svolge compiti specifici di organizzazione e analisi, come si può vedere nella Figura 1.8 l'organizzazione del modello MLP. Le MLP trovano applicazione in particolare per il trattamento di dati non strutturati o non etichettati. In uno studio [16] è stato proposto un database personalizzato per il riconoscimento delle emozioni, ottimizzando la rete per quattro emozioni, è stato raggiunto un tasso di riconoscimento del 97,1%, mentre per tre emozioni il tasso è stato del 96,4%. Nell'esperimento è stata considerata solo la caratteristica MFCC (Mel-Frequency Cepstral Coefficient).

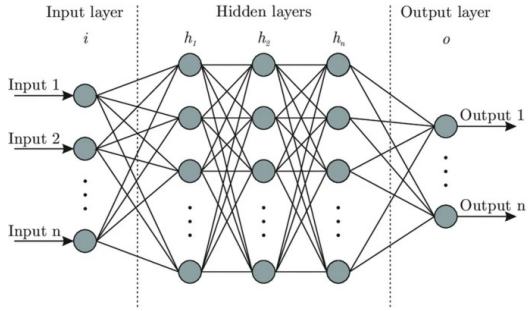

Figura 1.8: Struttura delle MLP [2].

Rete neurale ricorrente

La *Recurrent Neural Network* (RNN) è una rete neurale specializzata nel trattamento di dati sequenziali, dove input e output sono interconnessi per prevedere stati futuri. Tuttavia, come le CNN, le RNN hanno prestazioni limitate per pochi passi di *back-propagation* e necessitano di memoria per gestire le informazioni sequenziali. Il principale limite delle RNN è la scomparsa del gradiente, dove i gradienti si riducono esponenzialmente durante l'addestramento, portando alla perdita di informazioni iniziali.

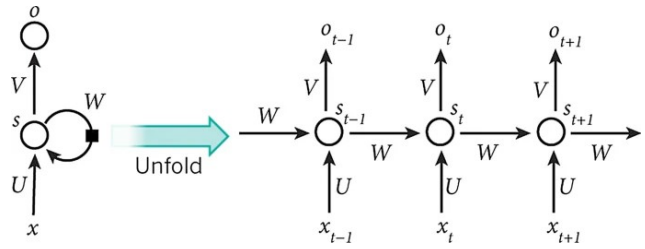

Figura 1.9: Struttura delle reti neurali ricorrenti [2].

Rete a credenze profonde

La *Deep Belief Network* (DBN) è un modello generativo che combina connessioni dirette e indirette tra variabili, con uno strato visibile e più strati nascosti. A differenza delle reti feedforward, le DBN impiegano variabili *stocastiche binarie* come unità nascoste e usano la *back propagation* per risol-

vere problemi di localizzazione e lentezza. Grazie alla capacità di apprendere parametri complessi, le DBN sono usate nel riconoscimento delle emozioni vocali e prevengono la non linearità tra i livelli. La Figura 1.10 mostra la struttura di una DBN con quattro strati nascosti e uno visibile.

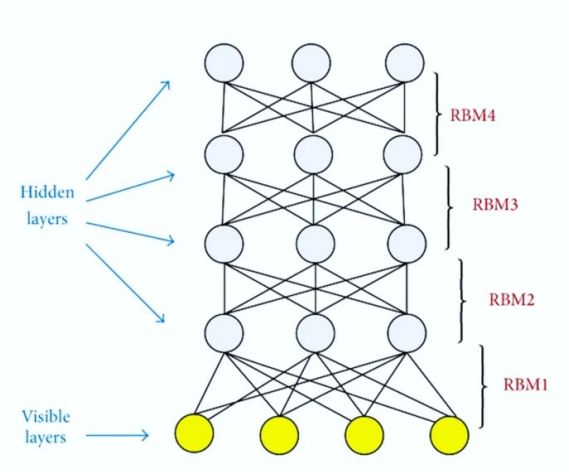

Figura 1.10: Struttura delle reti di credenza profonde [2].

Macchina di Boltzmann profonda

La *Deep Boltzmann Machine* (DBM) è un modello generativo probabilistico con unità binarie stocastiche, composto da unità visibili e più livelli nascosti collegati simmetricamente, come illustrato nella Figura 1.11. Le unità di ogni livello sono indipendenti tra loro ma dipendono dai livelli adiacenti. Utilizzata nel riconoscimento delle emozioni vocali, la DBM gestisce efficacemente la non linearità tra i livelli e impiega metodi di *back propagation* per l'addestramento. Il pre-addestramento non supervisionato permette di apprendere dai grandi database non etichettati, sebbene l'inferenza sia limitata a un passaggio verso l'alto.

Memoria a Lungo-Breve Termine

La *Long Short-Term Memory* (LSTM) affronta il problema del *vanishing gradient* con tre porte (forget, input, output) e uno stato di cella. La porta di

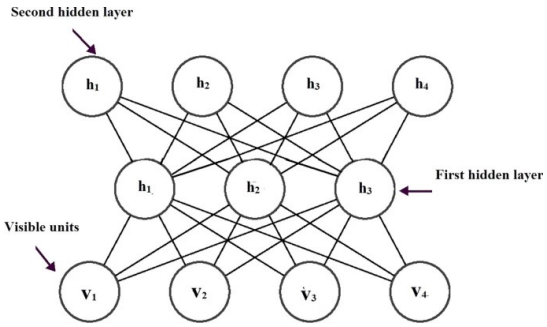

Figura 1.11: Struttura della macchina di Boltzmann profonda [2].

input decide quali nuove informazioni memorizzare, la forget cosa scartare, e l'output cosa restituire dallo stato di cella. Questo meccanismo permette di gestire dipendenze a lungo termine, come illustrato nella Figura 1.7, può dimenticare e richiamare le informazioni nello stato della cella tramite delle porte e mettere in relazione le conoscenze passate con quelle presenti. Kaya et al. hanno applicato LSTM-RNN al riconoscimento emotivo acustico *cross-corpus* e *cross-task*, classificando emozioni a livello di enunciato tramite previsioni di valenza e "arousal" a livello di frame. Hanno combinato queste previsioni con un sistema SVM utilizzando una fusione ponderata dei risultati, dimostrando l'efficacia del metodo per il riconoscimento delle emozioni acustiche in modalità continua e a livello di enunciato. La Figura 1.12 mostra la struttura principale dell'LSTM [13].

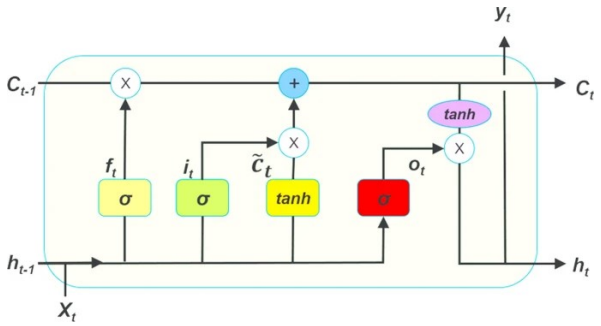

Figura 1.12: Struttura LSTM [2].

Capitolo 2

Le tecnologie utilizzate

Questo capitolo descrive le tecnologie impiegate per sviluppare il progetto di tesi, il quale è stato implementato interamente in *Python*. Inoltre, è stato utilizzato *Jupyter Notebook*, che si può definire come un'applicazione Web open source che permette di creare e condividere documenti testuali interattivi, contenenti oggetti quali equazioni, grafici e codice sorgente eseguibile [17]. Nella prima parte, verranno illustrate le principali librerie Python utilizzate per l'elaborazione e l'analisi dei dati, fondamentali per l'implementazione del progetto in questo ambito. Successivamente, verrà trattato il processo di sviluppo del modello, partendo dalle fasi di pre-processing dei dati fino all'addestramento e ottimizzazione degli algoritmi. Infine, saranno esaminati gli algoritmi di *Machine Learning* utilizzati, sui quali è stato testato il modello e confrontato per valutarne l'efficacia.

2.1 Librerie utilizzate

La libreria standard di Python è un insieme di moduli predefiniti che vengono forniti con ogni installazione di Python. Questa libreria offre una vasta gamma di funzionalità che coprono molte esigenze comuni nello sviluppo di software. Utilizzando la libreria standard, è possibile accedere a strumenti e funzioni pronti all'uso per risparmiare tempo e sforzi nella scrittura del

codice. In questo progetto di tesi sono state utilizzate librerie come Numpy, Pandas e Matplotlib per gestire la parte più matematica e grafica, mentre Librosa, Sklearn e Os per la preparazione dei dati. Invece le librerie come Pickle e Joblib sono servite per facilitare il lavoro e velocizzare l'addestramento dei modelli.

2.1.1 Numpy, Pandas e Matplotlib

Queste librerie sono state utilizzate soprattutto per la fase di pre-processing e analisi dei dati, per creare grafici e trasformare i campioni audio in *array*.

Numpy

NumPy è il pacchetto fondamentale per il calcolo scientifico in Python. È una libreria che fornisce un oggetto array multidimensionale, vari oggetti derivati (come array mascherati e matrici), e un insieme di routine per operazioni rapide sugli array. Una delle sue caratteristiche più potenti è l'uso di *broadcasting*, una tecnica che consente di applicare operazioni tra array di diverse forme senza dover esplicitamente ridimensionarli. Inoltre, NumPy funge da base per molte altre librerie di calcolo scientifico e machine learning in Python, come SciPy, pandas, e TensorFlow [18].

Pandas

Pandas è una libreria *open-source* di Python utilizzata per la manipolazione e l'analisi dei dati. È particolarmente utile per operazioni su dati strutturati, come tabelle e serie temporali. Pandas fornisce due principali strutture dati: *dataFrame* (per tabelle bidimensionali, simili a fogli di calcolo) e *series* (per array unidimensionali). Queste strutture consentono operazioni di filtraggio, aggregazione, ordinamento, raggruppamento e manipolazione dei dati in modo efficiente [19].

Matplotlib

Matplotlib è una delle librerie di visualizzazione dei dati più utilizzate in Python. Questa libreria è stata creata da John Hunter [20] per la creazione di grafici 2D e 3D nel linguaggio di programmazione Python, che produce figure di qualità scientifica adatte per pubblicazioni in una varietà di formati per la stampa e in ambienti interattivi su diverse piattaforme. Matplotlib è progettato con la filosofia che l'utente dovrebbe essere in grado di creare grafici semplici con pochi comandi. Matplotlib cerca di rendere facili le operazioni semplici e possibili quelle complesse [20].

2.1.2 Os, Sklearn e Librosa

Trattandosi di un progetto di Machine Learning, è necessario avvalersi delle librerie che permettono l'utilizzo e l'implementazione dei vari algoritmi di ML. Inoltre, Os e Librosa hanno avuto un ruolo fondamentale nella preparazione del dataset e nell'estrazione delle feature nei vari campioni audio.

Os

La libreria *Operating Systems* di Python fornisce funzioni per interagire con il sistema operativo. Consente di eseguire operazioni come la gestione dei file e delle directory, l'interazione con variabili d'ambiente, la gestione dei processi e la modifica dei permessi di file. La libreria è utile per scrivere script portabili, poiché supporta funzionalità specifiche del sistema operativo, mantenendo un'interfaccia comune per Windows, macOS e Unix. Le funzioni principali includono la gestione di percorsi di file (`os.path`), la manipolazione delle directory (`os.mkdir, os.rmdir`), e l'esecuzione di comandi di sistema (`os.system`). Nel progetto di tesi viene utilizzato per navigare all'interno del percorso dove sono contenuti i file audio da utilizzare.

Sklearn

Il progetto Scikit-learn, lanciato per la prima volta dal data scientist David Cournapeau [21] nel 2007, richiede il supporto di altri pacchetti come *NumPy* e *SciPy*, che sono un framework *open-source* sviluppato specificamente per applicazioni di ML nel linguaggio Python. Come molti altri progetti open source, *Scikit-learn* è attualmente mantenuto principalmente da membri della comunità. Le funzioni di base di Scikit-learn sono suddivise in sei parti: classificazione, regressione, clustering, riduzione dimensionale, selezione del modello e preprocessing dei dati. È importante notare che l'implementazione di MLP in Scikit-learn non è adatta per problemi su larga scala, poiché Scikit-learn non supporta il deep learning e l'accelerazione della GPU. Il maggiore vantaggio di Scikit-learn risiede nel suo algoritmo di regressione, che copre quasi tutte le esigenze degli sviluppatori e cosa più importante, Scikit-learn fornisce anche un semplice e utile riferimento ai casi d'uso per ciascun algoritmo. Questa libreria offre tutte le implementazioni modulari degli algoritmi di ML e i dati possono essere applicati direttamente al modello per ottenere il risultato desiderato. Di seguito, vengono elencati i vantaggi di Scikit-learn [21]:

1. Modelli selezionati e di alta qualità.
2. Copre la maggior parte dei compiti di machine learning.
3. Facilità di utilizzo, richiedendo solo un numero ridotto di pacchetti Python.

Librosa

Librosa è un pacchetto Python per l'analisi della musica e dell'audio, che offre una vasta gamma di strumenti per gestire i file audio, utilizzata soprattutto nell'ambito musicale, del *data scientist* e del ML. Librosa facilita il lavoro con i file audio grazie a una serie completa di funzioni, come la preelaborazione dei dati audio, estrazione delle caratteristiche, visualizzazione e

analisi, oltre a tecniche avanzate come la classificazione dei generi musicali e la separazione delle sorgenti audio. Librosa è costruita su NumPy e SciPy, fondamentali per il calcolo scientifico in Python, e offre moduli per diverse funzionalità:

- **Core**: funzioni principali di Librosa, incluse quelle per il caricamento di file audio, il campionamento e lo stretching temporale.
- **Estrazione delle caratteristiche**: estrazione di caratteristiche audio come mel spectrogram, contrasto spettrale, caratteristiche cromatiche, tasso di attraversamento dello zero e centroide temporale.
- **Visualizzazione**: funzioni per la visualizzazione di forme d'onda, spettrogrammi e altre rappresentazioni visive dell'audio.
- **Effetti**: funzioni per la manipolazione audio, come lo spostamento temporale e di tonalità, riduzione del rumore e segmentazione dell'audio.
- **Tecniche avanzate**: tecniche avanzate come la classificazione dei generi musicali, il riconoscimento delle emozioni vocali e la separazione delle sorgenti audio [22].

2.1.3 Pickle e Joblib

Le librerie pickle e joblib sono utili in questo progetto di tesi poiché permettono di salvare e caricare facilmente modelli addestrati, ottimizzando il riutilizzo e riducendo il tempo di esecuzione, con joblib particolarmente efficiente per la serializzazione di grandi dataset e l'esecuzione parallela.

Pickle

Il modulo *pickle* implementa un algoritmo di base per serializzare e deserializzare una struttura di oggetti Python. La "pickling" è il processo mediante il quale una gerarchia di oggetti Python viene convertita in un flusso

di byte, mentre la "unpickling" è l'operazione inversa, mediante la quale un flusso di byte viene riconvertito in una gerarchia di oggetti. Il modulo pickle può serializzare i tipi di base in Python, inclusi numeri, stringhe, tuple, liste e dizionari che contengono solo oggetti "picklable". Tuttavia, tale modulo non può serializzare funzioni definite dall'utente e dati delle classi, ma solo il loro nome. Per ogni tipo in Python, abbiamo definito un tag a un byte per identificare il tipo e usiamo strumenti corrispondenti (moduli, pacchetti o funzioni esistenti o nuovi) per serializzare o deserializzare. I passaggi per serializzare sono i seguenti:

1. Leggere il tipo dell'oggetto per determinare il pickler corrispondente.
2. Utilizzare la funzione dumps del pickler corrispondente per serializzarlo in byte grezzi.
3. Inserire un tag a un byte prima dei byte [23].

Esiste una tabella tag-to-pickler per registrare le mappature di tag/tipi e pickler. Gli utenti possono modificare la tabella per rendere questo modulo più adatto alle loro applicazioni.

Joblib

Joblib è una libreria Python utilizzata per eseguire compiti computazionalmente intensivi in parallelo. Fornisce un insieme di funzioni per eseguire operazioni in parallelo su grandi set di dati e per memorizzare nella cache i risultati di funzioni costose in termini di calcolo. Joblib è particolarmente utile per i modelli di machine learning perché consente di salvare lo stato del calcolo e riprendere il lavoro successivamente o su una macchina diversa. Rispetto ad altre tecniche di memorizzazione e caricamento dei modelli di machine learning, l'utilizzo di Joblib offre diversi vantaggi. Poiché i dati vengono memorizzati come stringhe di byte anziché come oggetti, è possibile salvare le informazioni in modo rapido e semplice, occupando meno spazio rispetto al tradizionale pickling. Inoltre, Joblib corregge automaticamente

gli errori durante la lettura o la scrittura dei file, rendendolo più affidabile rispetto al pickling manuale [24].

2.2 Raccolta dati

Il RAVDESS [25] è un database multimodale validato di parlato e canto emozionale. Il database è bilanciato per genere e comprende 24 attori professionisti che pronunciano frasi lessicalmente identiche con un accento neutro del Nord America. Il parlato include espressioni di calma, felicità, tristezza, rabbia, paura, sorpresa e disgusto, mentre il canto contiene emozioni di calma, felicità, tristezza, rabbia e paura. Ogni espressione è prodotta a due livelli di intensità emotiva, con un'espressione neutra aggiuntiva. Il set di 7356 registrazioni è stato valutato ciascuno 10 volte in termini di validità emotiva, intensità e genuinità. Le valutazioni sono state fornite da 247 individui, rappresentativi di partecipanti non addestrati provenienti dal Nord America. Un ulteriore gruppo di 72 partecipanti ha fornito dati di *test-retest*. Sono stati riportati alti livelli di validità emotiva e affidabilità nei test-retest. Vengono presentate misure corrette di accuratezza e compositi di "qualità" per supportare i ricercatori nella selezione degli stimoli.

2.2.1 Database RAVDESS

Il dataset utilizzato per l'esperimento è una sottoporzione del Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS), denominata RAVDESS Emotional Speech Audio. A differenza del dataset completo RAVDESS, che include audio e video di parlato e musica per un totale di quasi 25 GB, la versione qui utilizzata comprende esclusivamente audio parlato, con una dimensione di 450 MB. Questo dataset contiene 1440 file in formato WAV, suddivisi in 60 tracce per ciascuno dei 24 attori professionisti coinvolti (12 uomini e 12 donne). In ogni traccia, un attore pronuncia una frase con accento neutro del Nord America, esprimendo una specifica emozione e un differente livello di intensità. Le frasi contenute nei file audio sono due:

"Kids are talking by the door" e "Dogs are sitting by the door". Le emozioni espresse sono 8: calma, felicità, tristezza, rabbia, paura, disgusto, sorpresa e un'emozione neutra, la quale è associata a un unico livello di intensità. Le altre emozioni, invece, sono espresse con due livelli di intensità (normale e forte). Ogni combinazione di frase, emozione e intensità è ripetuta due volte, per un totale di 60 combinazioni disponibili in doppia ripetizione. Le informazioni relative alle combinazioni di emozioni, frasi e livelli di intensità sono codificate nei nomi dei file attraverso indici specifici, il cui significato è disponibile nella pagina Kaggle del dataset [25].

2.2.2 Analisi dei dati

Il dataset è stato sottoposto a una fase di pre-processing durante la quale, partendo dai titoli dei file audio, è stata costruita una tabella contenente tutte le informazioni necessarie per l'analisi, l'estrazione delle caratteristiche e la successiva fase di classificazione. Per l'estrazione delle serie temporali dalle tracce audio è stata utilizzata la funzione `load` di Librosa per caricare i file audio in formato *array*, mantenendo la frequenza di campionamento predefinita di 22050 Hz. Questo approccio preserva la risoluzione temporale delle tracce, ciò significa che ogni secondo di audio è rappresentato da 22050 valori in virgola mobile. Considerando che la durata media delle tracce audio è di circa 4 secondi, ciascun vettore dei campioni contiene mediamente 88200 valori. Per analizzare e visualizzare le caratteristiche del segnale audio, sono state utilizzate diverse tecniche di rappresentazione grafica che mostrino la forma d'onda e lo spettrogramma:

- La **forma d'onda** è stata visualizzata con `librosa.display.waveshow()`, mostrando l'ampiezza del segnale nel tempo. Questo tipo di rappresentazione consente di osservare le variazioni di intensità sonora durante la traccia audio, come si vede in Figura 2.1.

- E' stato calcolato lo **spettrogramma di Mel** utilizzando

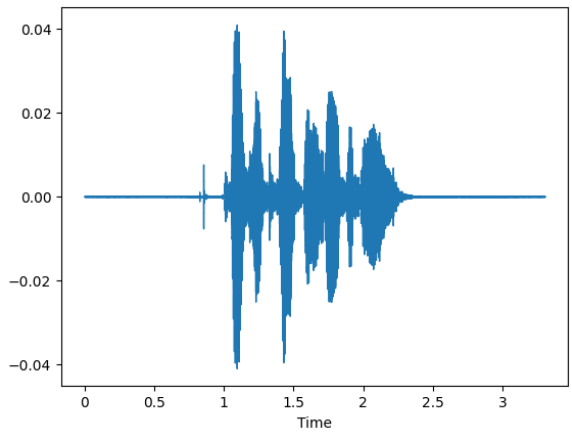

Figura 2.1: Rappresentazione della forma d'onda di una traccia del dataset

`librosa.feature.melspectrogram()`, che fornisce una rappresentazione temporale della distribuzione delle frequenze, mettendo in evidenza l'energia presente nelle diverse bande di frequenza nel corso del tempo, come si può notare in Fig. 2.2. Lo spettrogramma è stato visualizzato con `librosa.display.specshow()`, utilizzando una scala di frequenze Mel per migliorare l'interpretazione delle informazioni spettrali.

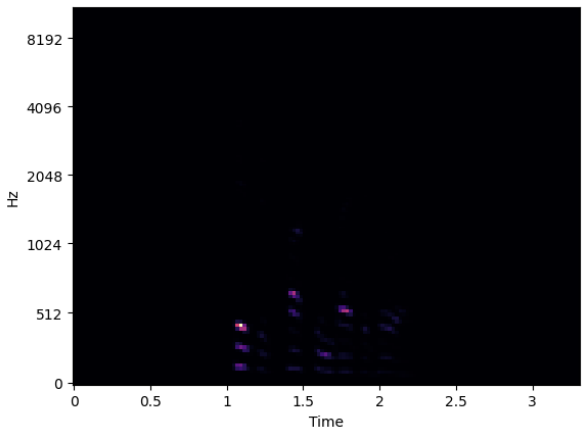

Figura 2.2: Rappresentazione dello spettrogramma di una traccia del dataset

Queste visualizzazioni permettono di analizzare il contenuto audio sia nel dominio temporale (forma d'onda) che nel dominio della frequenza (spettrogramma di Mel), fornendo informazioni preziose per l'estrazione delle caratteristiche e la classificazione delle emozioni nelle tracce audio.

2.3 Pipeline di Sviluppo del Modello

La creazione di questo progetto di tesi ha dovuto affrontare un lungo processo, a partire dall'incremento degli elementi nel dataset tramite le tecniche di data augmentation, fino ad arrivare alla tecniche di addestramento, fondamentali per questo progetto.

2.3.1 Preprocessing dei Dati

Prima di poter lavorare sul dataset è stato necessario una rielaborazione dei dati, quale il data augmentation, che ha permesso di aver a disposizione una mole di dati maggiore rispetto a quella precedente su cui fare addestramento.

Data Augmentation

Due problemi comuni dei database vocali per il SER sono:

1. la quantità di dati audio non è abbastanza grande, soprattutto per l'addestramento di un classificatore machine learning.

2. i dati delle diverse classi di emozioni sono spesso sbilanciati.

Questi due problemi influenzano notevolmente le prestazioni di classificazione delle emozioni. Per risolvere questi problemi, viene utilizzata una strategia di incremento dei dati. L'incremento dei dati è progettato per generare più dati per aumentare la quantità di dati e per bilanciare i dati in diverse classi di emozioni. Questo metodo può ridurre l'over-fitting dei modelli e quindi migliorare la generalizzazione del modello [26].

Per incrementare la dimensione del dataset nel progetto di tesi, sono state utilizzate diverse tecniche di data augmentation. In particolare, si è fatto ampio uso delle funzionalità offerte dalla libreria Librosa, che consentono di applicare filtri specifici e modificare le caratteristiche degli audio, e delle funzioni standard di NumPy, impiegate per manipolare direttamente i valori delle serie temporali estratte, introducendo effetti generali. Il processo di augmentation ha permesso di portare la dimensione complessiva del dataset da 1440 elementi a 4320.

Di seguito vengono descritte le tecniche di augmentation adottate nel corso dell'esperimento.

- `noise` viene sommato del rumore bianco al segnale audio originale. Questo può rendere il modello più robusto a variazioni nel rumore di fondo, simulando scenari realistici in cui l'audio potrebbe contenere rumori indesiderati. Nel codice, viene aggiunto un valore costante di rumore (`0.1*np.ones(len(audio))`) a ogni campione audio.

- `time_stretch` modifica la velocità di riproduzione dell'audio senza alterarne il pitch. Nel codice, il parametro `rate=1.2` indica che l'audio viene riprodotto al 120% della velocità originale, il che accorcia la durata dell'audio.

- `pitch_shift` consente di cambiare l'altezza del suono (pitch) senza alterarne la durata. Nel codice, l'audio viene spostato di due semitoni verso l'alto `(n_steps=2)`. Questo può simulare variazioni nella voce o nel tono dell'oratore.

2.3.2 Tecniche di Addestramento

Le tecniche di addestramento giocano un ruolo cruciale nell'ottimizzazione dei modelli di riconoscimento delle emozioni. In questo contesto, la cross-validation permette di valutare le prestazioni in modo affidabile, il tuning dei parametri consente di migliorare l'accuratezza e la GridSearch automa-

tizza la ricerca dei parametri ottimali, massimizzando l'efficacia dei modelli di classificazione.

Cross-Validation

La *validazione incrociata* è l'applicazione della suddivisione dei dati per stimare la capacità predittiva di uno o più modelli candidati. La validazione incrociata funziona adattando ciascun modello a un sottoinsieme dei dati disponibili (il set di addestramento) e poi confrontando le capacità predittive dei modelli (perdita) sulla porzione rimanente dei dati (il set di test). Per migliorare la stima, la procedura di suddivisione viene solitamente ripetuta selezionando in modo sistematico diversi sottoinsiemi di dati e riassumendo le prestazioni predittive complessive tra le iterazioni. Esistono molte varianti della *Cross Validation*, ognuna caratterizzata da diversi schemi di suddivisione dei dati. Uno schema comunemente utilizzato è il *k-fold*, in cui i dati disponibili vengono suddivisi in k sottoinsiemi di dimensioni uguali ("fold"), generando k coppie distinte di set di addestramento o test, ottenute rimuovendo uno dei fold alla volta (il set di test) dall'intero insieme di dati. Questo schema può essere applicato una sola volta, per una singola suddivisione iniziale (k-fold ordinario), oppure può essere ripetuto molte volte per suddivisioni differenti (k-fold ripetuto). Nel progetto di Tesi, per il caso suddividiamo i dati del training set (composto da 20 attori) utilizzando una k-fold Cross-Validation con 10 fold. In ogni ciclo, selezioniamo una coppia di attori (un maschio e una femmina) che viene usata come set di test, mentre i rimanenti attori vengono utilizzati per addestrare il modello. Un aspetto vantaggioso della convalida incrociata è che permette di sfruttare tutti i dati disponibili sia per l'addestramento sia per i test, invece di limitarsi a una singola suddivisione dei dati. Questo approccio porta a una valutazione più affidabile delle prestazioni del modello e a una stima più precisa delle sue performance previste su dati non ancora osservati [27].

Tuning degli Iperparametri

L'ottimizzazione degli iperparametri dovrebbe essere considerata come un ciclo esterno formale nel processo di apprendimento. Nel caso più generale, tale ottimizzazione dovrebbe includere una scelta di *budgeting* su quante risorse di calcolo (CPU) destinare all'esplorazione degli iperparametri e quante risorse destinare alla valutazione di ciascuna configurazione di iperparametri. Una strategia semplice per l'ottimizzazione degli iperparametri è un approccio *greedy*: indagare il vicinato locale di una data configurazione di iperparametri, variando un iperparametro alla volta e misurando come cambia la prestazione. L'unica informazione ottenuta con questa analisi è come diversi valori di iperparametri si comportano nel contesto di una singola istanza degli altri iperparametri.

Grid Search

La strategia più comunemente utilizzata per l'ottimizzazione degli iperparametri è una combinazione di Grid Search (GS) e ottimizzazione manuale. Ci sono diverse ragioni per cui la ricerca manuale e la Grid Search prevalgono come stato dell'arte nonostante decenni di ricerca sull'ottimizzazione globale:

- L'ottimizzazione manuale fornisce ai ricercatori un certo grado di comprensione del processo di ottimizzazione.
- Non ci sono sovraccosti tecnici o barriere per l'ottimizzazione manuale.
- Nella Grid Search la parallelizzazione e l'implementazione sono semplici.
- La Grid Search (con accesso a un cluster di calcolo) solitamente trova una soluzione migliore rispetto all'ottimizzazione manuale pura e sequenziale nello stesso tempo.
- La Grid Search è affidabile in spazi a bassa dimensionalità (es., 1-d, 2-d). Ad esempio, la Grid Search è relativamente efficiente per ottimizzare i due iperparametri standard degli SVM.

La GS soffre della "maledizione della dimensionalità" perché il numero di combinazioni di valori cresce esponenzialmente con il numero di iperparametri. Pertanto, la GS non è consigliata per l'ottimizzazione di molti iperparametri. In questo progetto di tesi è stata utilizzata per valutatare l'accuratezza media e l'F1-score su 10 fold di cross validation [28].

2.4 Algoritmi di Machine Learning per il SER

Il Machine Learning è la capacità delle macchine di apprendere, dove una macchina è costruita utilizzando determinati algoritmi attraverso i quali può prendere decisioni autonome e fornire risultati all'utente. Fondamentalmente, è considerato un sotto-campo dell'AI. Oggi il Machine Learning viene utilizzato per la classificazione di dati complessi e il processo decisionale. In termini semplici, consiste nello sviluppo di algoritmi che permettono al sistema di apprendere e prendere decisioni necessarie. I metodi e i compiti del Machine Learning sono generalmente suddivisi in tre categorie principali:

1. **Apprendimento supervisionato** In questo tipo di apprendimento il sistema riceve un campione di input e viene mappato con l'output. In questo tipo di apprendimento, ogni esempio è una coppia costituita da un oggetto in ingresso (fondamentalmente un vettore) e da un valore di uscita desiderato (segnale di supervisione). Un algoritmo di apprendimento supervisionato analizza e studia i dati di addestramento e produce una funzione dedotta che può essere utilizzata per mappare nuovi esempi. Gli approcci per l'Apprendimento Supervisionato includono Random Forest, Support Vector Machines, Decision Trees, ecc.

2. **Apprendimento non supervisionato** In questo tipo di apprendimento, al sistema vengono forniti alcuni input di esempio, ma non è presente alcun output. Poiché non c'è un output desiderato, la categorizzazione viene effettuata in modo che l'algoritmo possa differenziare correttamente tra i set di dati. Si tratta di definire una funzione per descrivere una struttura nascosta a partire da dati non etichettati.

3. **Apprendimento per rinforzo** Il reinforcement learning è un sottodominio del machine learning ispirato alla psicologia comportamentale, che si occupa di come gli agenti software dovrebbero agire in un ambiente per massimizzare una certa nozione di ricompensa cumulativa. Viene studiato e utilizzato in molte teorie, come la teoria dei giochi, la teoria del controllo, la ricerca operativa, la teoria dell'informazione, ecc [29].

2.4.1 Alberi di Decisione e Tecniche di Ensemble

Un albero decisionale è lo strumento più utilizzato per il processo decisionale. I metodi *ensemble* e gli alberi decisionali sono strettamente collegati nell'apprendimento automatico: gli alberi decisionali vengono spesso utilizzati come modelli di base all'interno degli approcci ensemble per migliorare le prestazioni complessive. Gli ensemble combinano i risultati di più alberi decisionali per ridurre errori di varianza e *bias*, aumentando così la capacità di generalizzazione del modello.

Decision Tree

L'albero decisionale è un tipo comune di algoritmo di apprendimento automatico, il cui scopo è creare un modello che predice il valore di una variabile target apprendendo semplici regole decisionali dedotte dalle caratteristiche dei dati. Il modello ha una struttura ad albero che contiene un nodo radice, diversi nodi interni e diversi nodi foglia. Il nodo radice contiene tutti i campioni, quindi suddivide i dati in nodi interni figli in base all'attributo ottimale selezionato secondo un determinato criterio. Ogni nodo interno continua a suddividere, fino a quando la maggior parte dei dati viene classificata correttamente [30]. Per comprendere in seguito il modello Random Forest, si può iniziare descrivendo il modello ad albero decisionale. In generale, lo sviluppo di un albero decisionale coinvolge tre passaggi:

1. Determinare la variabile target e le variabili attributo. Come mostrato nella Figura 2.3, tutti i dati si concentrano nel nodo radice prima della suddivisione.

2. Il *dataset* viene suddiviso in nodi figli applicando un algoritmo di suddivisione (ad esempio l'algoritmo ID3) alle variabili attributo.

3. Ogni nodo figlio verrà trattato come nodo genitore per una successiva suddivisione [31].

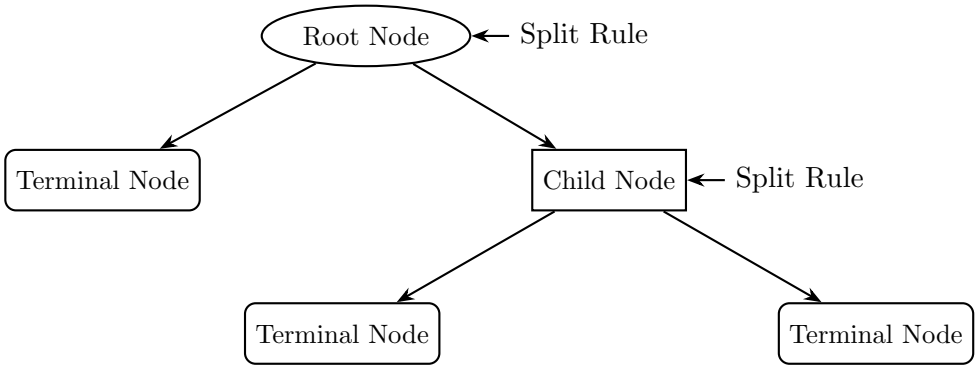

Figura 2.3: Struttura di un albero decisionale

Metodo ensamble

L'*ensemble learning* è una classe di metodi di apprendimento automatico che addestra più modelli per risolvere lo stesso problema e li combina per ottenere un nuovo modello migliore. Quando si applicano i metodi di ensemble, i modelli base possono essere diversi (come alberi decisionali, reti neurali, macchine a vettori di supporto o altri algoritmi di apprendimento) oppure essere gli stessi modelli ma addestrati con dati o parametri diversi. Anziché selezionare il singolo modello migliore, i risultati finali dell'ensemble possono essere determinati utilizzando metodi di media, votazione o stacking. Questi processi aumentano la diversificazione dei modelli di ensemble e riducono il rischio di overfitting [30]. Si distinguono generalmente due famiglie di metodi di ensemble:

- Nei **metodi di bagging** come RF, il principio guida è quello di costruire diversi apprendimenti di base in parallelo e poi formare un risultato finale facendo la media di queste previsioni.
- Nei **metodi di boosting** come XGBoost, lo scopo è quello di combinare diversi apprendenti di base deboli per produrre un apprendente potente.

Random Forest

Il modello di *Random Forest* (RF) è un metodo di apprendimento *ensemble* composto da numerosi alberi decisionali, che utilizza il *bootstrapping* (campionamento casuale con re-immissione), la selezione casuale di sottoinsiemi di caratteristiche e il voto medio per effettuare previsioni. RF rappresenta una versione potenziata del metodo standard dell'albero decisionale, poiché le prestazioni di molti deboli apprenditori, ovvero un singolo albero decisionale, possono essere migliorate attraverso uno schema di voto. La procedura di addestramento di RF consiste nel creare innanzitutto un nuovo set di dati di addestramento mediante campionamento bootstrapping per ogni albero nella foresta; successivamente, per ottenere la divisione ottimale, si fa crescere un albero cercando un sottoinsieme casuale di caratteristiche di input in ogni nodo. Le previsioni risultanti da ogni albero vengono infine aggregate per produrre una previsione finale (Fig. 2.4), mediante media dei valori di tutti gli alberi (*regressione*) o voto di maggioranza (*classificazione*). Le Random Forest (RF) superano i problemi di overfitting ottenendo la previsione finale aggregata dalle singole previsioni di ciascun albero decisionale, e offrono un'elevata accuratezza grazie alla riduzione della varianza. Le RF riducono la varianza di un singolo albero decisionale, portando a previsioni migliori su nuovi dati. Inoltre, le RF sono molto flessibili, poiché non richiedono pre-elaborazione dei dati, come la scalatura o l'assunzione di linearità o altre ipotesi sui dati. Tuttavia, il principale svantaggio delle RF è la complessità, che richiede risorse computazionali elevate e rende difficile interpretare intuitivamente la relazione tra variabili dipendenti e predittive [32].

Il modello di Random Forest è costruito principalmente su due metodi:

1. **Bagging** è un processo che prevede l'estrazione di campioni di bootstrapping e poi aggregare i modelli appresi su ciascun campione.

2. **Sottospazio casuale** implica la selezione casuale delle variabili da usare per suddividere i nodi di ciascun albero decisionale nel modello RF [31].

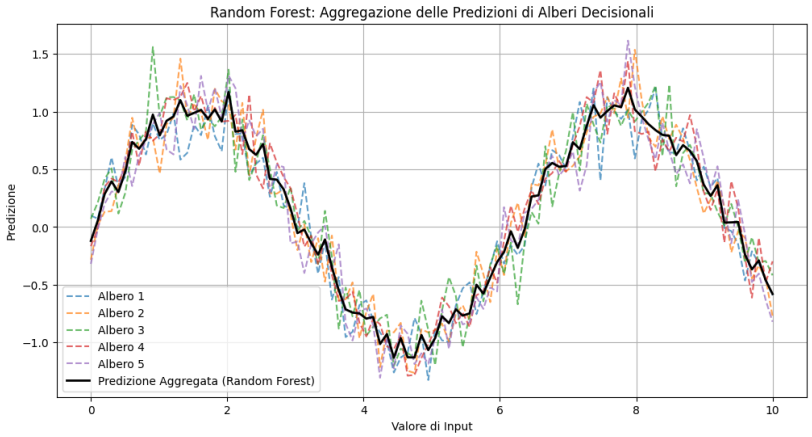

Figura 2.4: Predizione di un modello RF a cinque alberi decisionali.

Gradient Boosting

Il *Gradient Boosting* è un altro tipo di algoritmo di machine learning supervisionato e di tipo ensemble, utilizzabile sia per problemi di classificazione che di regressione. Il motivo principale per cui algoritmi come random forest, extra trees, gradient boosting, ecc., vengono definiti ensemble è che il modello finale viene generato a partire da molti modelli individuali. Di conseguenza, i casi più complessi diventano il focus durante l'addestramento. Un addestramento sequenziale dei modelli utilizzando il gradient boosting minimizzerà gradualmente una funzione di perdita. La minimizzazione della funzione di perdita avviene in modo simile a un modello di rete neurale artificiale, in cui i pesi vengono ottimizzati. Nel gradient boosting, dopo aver

costruito i weak learners, le predizioni vengono confrontate con i valori reali. La differenza tra le *predizioni* e i *valori effettivi* rappresenta il tasso di errore del modello. Questo tasso di errore può essere ora utilizzato per calcolare il gradiente, che è essenzialmente la derivata parziale della funzione di perdita. Il gradiente viene usato per determinare la direzione in cui devono cambiare i parametri del modello per ridurre l'errore nel successivo ciclo di addestramento. A differenza di un modello di rete neurale, in cui la funzione obiettivo è minimizzare una funzione di perdita in un singolo modello, nel gradient boosting le predizioni di più modelli vengono combinate(vedi Figura 2.5) [3].

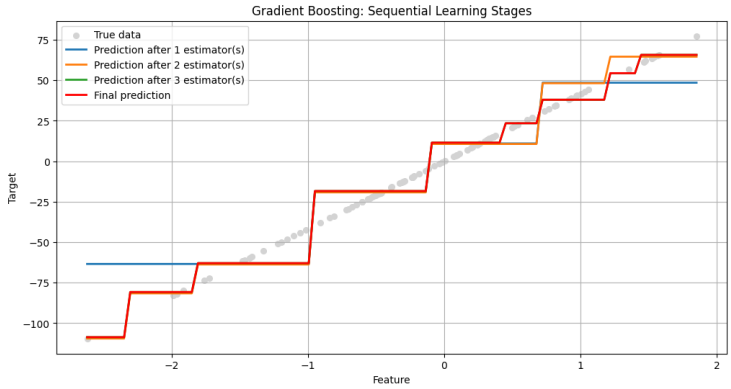

Figura 2.5: Rappresentazione del processo di apprendimento sequenziale nel Gradient Boosting

Extreme Gradient Boosting

L'*Extreme Gradient Boosting* (XGBoost) è un algoritmo di ML basato su alberi decisionali potenziati dal gradient boosting. XGBoost, uno degli algoritmi di apprendimento comunitario di nuova generazione, aumenta la precisione complessiva del modello prevenendo il problema dell'overfitting durante l'addestramento. Come mostrato in Fig. 2.6, XGBoost costruisce alberi successivi per ridurre progressivamente l'errore residuo, con ogni nuovo albero che corregge l'errore del modello precedente. Il principale motivo del

successo di questo metodo è la funzione obiettivo che utilizza nel processo di apprendimento. La funzione obiettivo comprende la funzione di perdita e la regolarizzazione. La funzione di perdita calcola la differenza tra ciascun valore predetto dal modello e il suo valore effettivo. Il termine di regolarizzazione controlla la complessità del modello, eliminando il problema dell'overfitting. Devan e Khare [33] hanno utilizzato XGBoost per la selezione delle caratteristiche nella rilevazione delle intrusioni e hanno confrontato le loro prestazioni di classificazione in base a diversi numeri di caratteristiche. Hanno sviluppato un modello di classificazione con reti neurali profonde (DNN) e confrontato i risultati con metodi di machine learning classici. Dhaliwal et al. hanno sviluppato un modello di classificazione utilizzando XGBoost per aumentare l'efficacia dei sistemi di rilevamento delle intrusioni, confrontando i loro risultati con quelli di studi che impiegano diversi metodi di machine learning [33].

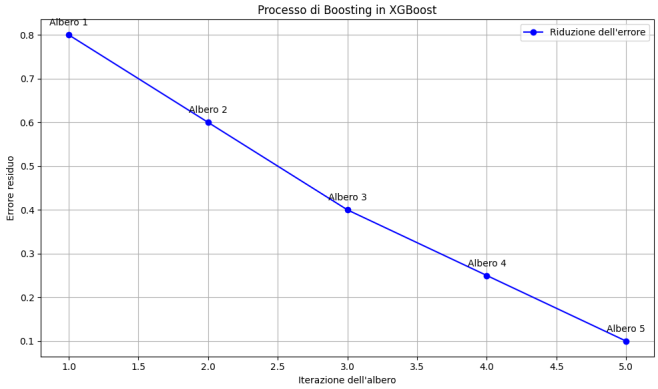

Figura 2.6: Processo di Boosting in XGBoost

2.4.2 Modelli di Regressione

I *modelli di regressione* sono tecniche di apprendimento supervisionato utilizzate per prevedere il valore di una variabile dipendente (o target) in base a una o più variabili indipendenti (o caratteristiche). L'obiettivo è trovare una funzione che mappi l'input al valore di output con un certo livello di

accuratezza. Nei modelli di regressione, l'output previsto è una variabile continua, piuttosto che categoriale come nei problemi di classificazione [34].

Logistic regression

La regressione logistica è un altro potente algoritmo di machine learning supervisionato utilizzato per problemi di classificazione binaria quando il target è categorico. Il modo migliore per comprendere la regressione logistica è pensare a essa come una regressione lineare, ma per problemi di classificazione. La regressione logistica utilizza essenzialmente una funzione logistica, definita qui di seguito (2.1), per modellare una variabile di output binaria. La differenza principale tra regressione lineare e regressione logistica è che l'intervallo della regressione logistica è limitato tra 0 e 1. Inoltre, a differenza della regressione lineare, la regressione logistica non richiede una relazione lineare tra le variabili di input e di output, grazie all'applicazione di una trasformazione logaritmica non lineare al rapporto di probabilità.

$$\text{Logistic function} = \frac{1}{1 + e^{-x}} \tag{2.1}$$

Nell'equazione della funzione logistica, x è la variabile di input. Alimentiamo valori da -20 a 20 nella funzione logistica. Come illustrato nella Fig. 2.7, gli input sono stati trasferiti nell'intervallo tra 0 e 1.

A differenza della regressione lineare, dove la MSE(Mean-Squared Error) o la RMSE(Root-Mean-Squared Error) viene utilizzata come funzione di perdita, la regressione logistica utilizza una funzione di perdita nota come "massima verosimiglianza (MLE)", che è una probabilità condizionata. Se la probabilità è superiore a 0,5, le predizioni saranno classificate come classe 0. Altrimenti, sarà assegnata la classe 1 [3].

2.4.3 Reti Neurali

Le *Neural Networks* (NN) sono modelli computazionali ispirati alla struttura neurale, ai processi di elaborazione e alla capacità di apprendimento del cervello umano, seppure su scale molto più ridotte. Questa tecnica è

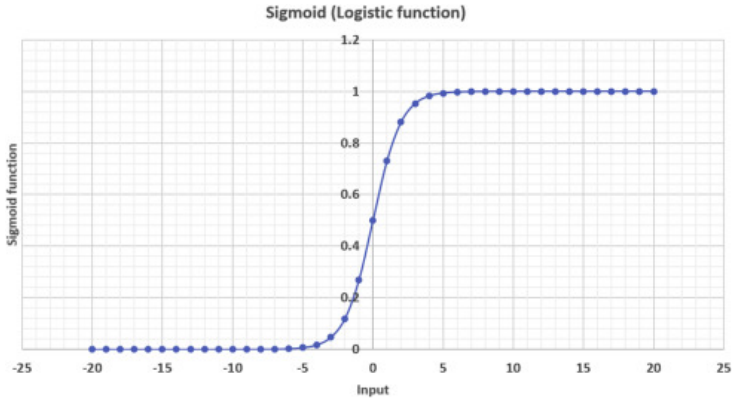

Figura 2.7: Regressione logistica applicata a un intervallo compreso tra -20 e 20 [3].

particolarmente adatta per affrontare problemi in cui le relazioni sono non lineari o altamente dinamiche. Grazie alla loro capacità di catturare una vasta gamma di relazioni, le reti neurali permettono di modellare in modo rapido e relativamente semplice fenomeni complessi che altrimenti sarebbero difficili o impossibili da spiegare con altri metodi [35].

Multi-Layer Perceptron (MLP)

Il *perceptron multilivello* (MLP) è un complemento della rete neurale feed-forward. Esso è composto da tre strati: uno strato di input, uno strato nascosto e uno strato di output, come mostrato nella Fig. 2.8. Lo strato di input accetta il segnale da elaborare. Lo strato di output è responsabile di funzioni come la classificazione e la previsione. Il meccanismo di connessione diretta dell'MLP è costituito da una serie infinita di strati nascosti situati tra gli strati di output e di input. I dati vengono trasmessi in un percorso diretto dallo strato di input allo strato di output nell'MLP, equivalente a una rete feed-forward. La tecnica di apprendimento *back-propagation* viene utilizzata per addestrare tutti i nodi nell'MLP. Gli MLP possono risolvere problemi che non sono separabili linearmente e sono strutturati per approssimare ogni funzione continua [36]. Le seguenti sono le computazioni eseguite da ciascun

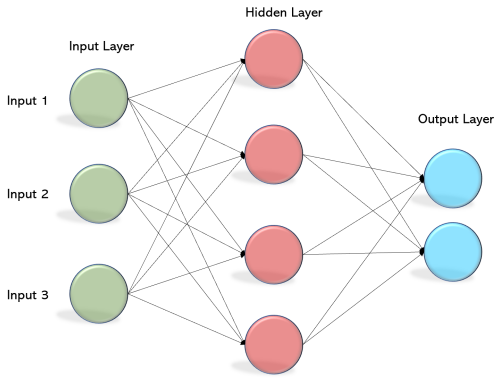

Figura 2.8: La rappresentazione del perceptron multilivello [4].

neurone nei livelli nascosti e di output:

$$h(r) = \Phi(r) = \rho(V_1 g(r) + c_1) \tag{2.2}$$

$$g(r) = \phi(V_2 h(r) + c_2) \tag{2.3}$$

con i vettori di bias c_1, c_2; le matrici di pesi V_1, V_2; e le funzioni di attivazione ρ e Φ. I parametri da apprendere sono l'insieme $\theta = \{V_1, V_2, c_1, c_2\}$. Scelte distintive per la funzione di attivazione comprendono la funzione tangente iperbolica, dove

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.4}$$

oppure la funzione sigmoide logistica, con

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}. \tag{2.5}$$

2.4.4 Support Vector Machines

L'SVM è un altro tipo di algoritmo di apprendimento supervisionato che può essere utilizzato sia per problemi di classificazione che di regressione. Diversamente da molti algoritmi di apprendimento automatico, in cui la funzione obiettivo è minimizzare una funzione di costo, l'obiettivo nell'SVM è

massimizzare il margine tra i support vectors attraverso un pianoforte separatore. In Fig. 2.9, un iperpiano separatore è tracciato per separare le istanze blu da quelle rosse. Questo iperpiano è disegnato in modo tale da massimizzare il margine su entrambi i lati. Si noti che deve essere tracciata la più ampia estensione possibile per separare le istanze rosse e blu. Le istanze blu e rosse più vicine all'iperpiano separatore sono chiamate support vectors, ed è per questo che questo algoritmo è chiamato Support Vector Machine. La distanza tra le linee tratteggiate blu e rosse rispetto all'iperpiano separatore è chiamata "margine". L'obiettivo è massimizzare i margini tra i vettori di supporto. Questa illustrazione è per uno spazio bidimensionale e l'iperpiano avrà dimensione n-1 per uno spazio n-dimensionale [3].

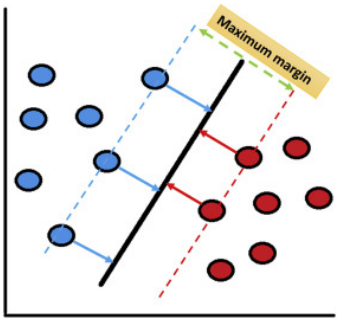

Figura 2.9: Support Vector Machines per l'illustrazione della classificazione [3].

Support Vector Classification (SVC)

Il *classificatore a vettori di supporto* (SVC) è un metodo di apprendimento supervisionato basato sulla statistica. Esso offre eccellenti prestazioni anche con piccoli dataset, in contrasto con gli algoritmi di intelligenza artificiale convenzionali che generalmente richiedono una grande quantità di dati di addestramento. Inoltre, gli algoritmi tradizionali possono incontrare difficoltà come rimanere intrappolati in un minimo locale, avere una basso tasso di convergenza o essere soggetti a overfitting. Queste difficoltà, tuttavia, non si presentano quando si utilizza l'algoritmo SVC. Il kernel nell'SVC

gioca un ruolo cruciale nel migliorare le sue capacità, infatti, converte i dati di input in uno spazio a dimensioni superiori con l'obiettivo di identificare più facilmente l'iperpiano separatore, come mostrato in Figura 2.10. Questo consente all'algoritmo di gestire relazioni complesse e non lineari tra i punti dati, rendendolo versatile in diverse applicazioni reali. La funzione principale dell'algoritmo SVM è descritta nell'Eq. 2.6, dove si mira a identificare la linea separatrice più distante tra le diverse classi [37].

$$f(x) = (w, x) + b \tag{2.6}$$

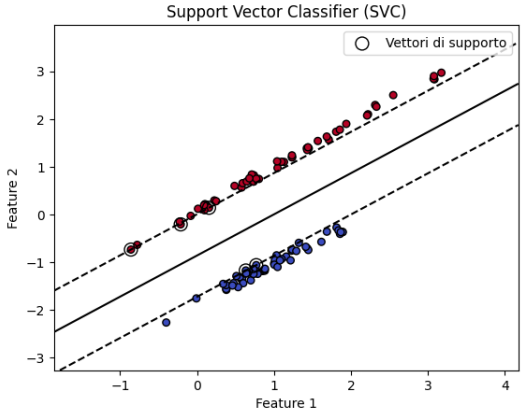

Figura 2.10: Classificazione eseguita da SVC su un dataset bidimensionale

Capitolo 3

Sviluppo progettuale

Nell'ultimo capitolo di questo volume di tesi verranno mostrati i risultati ottenuti dai diversi esperimenti sostenuti e precedentemente descritti. In seguito alla descrizione del contesto dell'esperimento e delle tecniche impiegate per raggiungere i risultati ottenuti tramite le tecnologie sopra descritte, si analizzeranno gli esiti raggiunti riguardanti ognuna delle tipologie di modelli usati. Nella prima sezione verrà descritto il Dataset utilizzato, la metodologia con cui sono stati separati i dati e la Data augmentation che è stata effettuata per migliorare l'addestramento. Nella seconda sezione verrano trattati invece i vari esperimenti che sono stati condotti e i modelli utilizzati per ottenere il miglior risultato. Infine, nell'ultima sezione verranno descritti e messi a confronto i diversi modelli utilizzati e le loro prestazioni.

3.1 Dataset

Il dataset utilizzato è stato un elemento cruciale in questo progetto di tesi in quanto ha permesso di testare efficacemente i modelli sviluppati e di valutare le loro prestazioni. La qualità e la quantità dei dati raccolti hanno contribuito significativamente alla robustezza delle analisi effettuate, consentendo di esaminare con precisione l'accuratezza e la capacità di generalizzazione dei modelli nel riconoscimento delle emozioni. Senza un dataset

adeguato, sarebbe stato impossibile validare i risultati ottenuti e ottenere una comprensione approfondita delle performance dei vari approcci sperimentati.

3.1.1 Split dei dati

La separazione dei dati è un passaggio cruciale nel processo di sviluppo di modelli di Machine Learning, poiché consente di valutare le loro prestazioni. Nella pratica, i dati vengono solitamente suddivisi in tre subset:

- **Training set:** utilizzato per addestrare il modello.
- **Validation set:** valuta le prestazioni del modello durante la fase di addestramento, aiutando a regolare gli iperparametri e a prevenire l'*over-fitting*.
- **Test set:** fornisce una valutazione corretta delle prestazioni del modello su dati non visti.

Il *training set*, o set di addestramento, è la porzione di dati utilizzata per addestrare il modello. Il modello deve osservare e imparare dal set di addestramento, ottimizzando i suoi parametri. Il *validation set*, o set di validazione, è un insieme di dati di esempi utilizzati per modificare i parametri del processo di apprendimento. Questo insieme di dati ha lo scopo di classificare l'accuratezza del modello e può aiutare nella selezione del modello. I dati devono essere suddivisi in modo che i set di dati possano avere un'elevata quantità di dati di addestramento [38]. Infatti, in questo progetto di tesi il dataset è stato suddiviso principalmente in 2 parti con un rapporto di 80:20, dove 80 rappresenta il dataset di addestramento e 20 il dataset di test. Il principale motivo di questa decisione è stato il *principio di Pareto*, secondo il quale l'80% dei risultati è determinato dal 20% delle cause. Dato che disponiamo di un dataset abbastanza consistente a livello quantitativo, molte ricerche e sperimentazioni indicano che, per dataset di grandi dimensioni, l'80:20 è il miglior rapporto di suddivisione per l'addestramento e il test, quindi è stato deciso di adottarlo. Nello specifico, il dataset contiene 1.440 righe, mentre

per il test set è stata presa la decisione di scegliere quattro attori dividendo equamente maschi e femmine, per un totale di 240 righe (Fig. 3.1). Dato che per il training set sono stati selezionati venti attori e sapendo che il set è composto da 1.200 righe (Fig. 3.2), possiamo calcolare la percentuale di righe nel training set rispetto al totale, che sarà pari all'80% circa (Eq. 3.1). Mentre se vogliamo calcolare la percentuale di righe nel test set rispetto al totale sarà pari al 20% circa (Eq. 3.2), quindi è stato rispettato il principio di Pareto.

$$\text{Percentuale training set} = \frac{1200}{1440} \times 100 = 83.33\% \tag{3.1}$$

$$\text{Percentuale test set} = \frac{240}{1440} \times 100 = 16.67\% \tag{3.2}$$

Gli attori sono stati selezionati sia casualmente sia con una scelta mirata, destinando quattro di essi alla fase di verifica e i restanti venti all'addestramento. Durante la Cross Validation, i dati di training sono stati suddivisi ulteriormente in set di training e validation, distribuendo i campioni tra dieci coppie di attori. Un altro aspetto rilevante è la casualità nella suddivisione dei dati: campionare in modo casuale ciascun set aiuta a garantire l'affidabilità dei risultati, evitando che essi siano influenzati dai campioni specifici utilizzati.

3.1.2 Data augmentation

Come descritto ampiamente nel capitolo precedente, diverse tecniche di data augmentation sono state adottate per ampliare la dimensione del dataset, portandolo da 1.440 a circa 4.320 elementi, quasi quadruplicandone la quantità iniziale. Questo risultato è stato raggiunto principalmente grazie alla libreria Python Librosa, che ha consentito di modificare le caratteristiche audio dei campioni vocali e applicare vari filtri per aumentarne la varietà. L'utilizzo del data augmentation si è rivelato fondamentale per questo progetto, poiché un maggior numero di dati permette al modello di apprendere

	Emotions	Actor	Chroma_1	Chroma_2	Chroma_3	Chroma_4	Chroma_5	Chroma_6	Chroma_7	Chroma_8	...
0	neutral	1	0.548701	0.565829	0.553265	0.552938	0.556421	0.551393	0.554142	0.550156	...
1	neutral	1	0.546703	0.571287	0.566894	0.552942	0.564013	0.573600	0.551956	0.547391	...
2	neutral	1	0.550520	0.565476	0.563730	0.564877	0.558413	0.542510	0.546161	0.539936	...
3	neutral	1	0.533803	0.530319	0.525897	0.539019	0.555649	0.547618	0.525339	0.529758	...
4	calm	1	0.547657	0.582334	0.594180	0.585837	0.588524	0.597594	0.586218	0.575669	...
...	...	...	...	...	...	...	...	...	...	...	...
1435	surprised	24	0.474954	0.435510	0.405085	0.406706	0.427041	0.432501	0.446725	0.446329	...
1436	surprised	24	0.577005	0.551714	0.518492	0.511875	0.503541	0.481119	0.488970	0.487261	...
1437	surprised	24	0.558421	0.513652	0.478237	0.464434	0.481322	0.505518	0.485552	0.466317	...
1438	surprised	24	0.494204	0.479549	0.473162	0.457562	0.471886	0.498949	0.509475	0.463188	...
1439	surprised	24	0.543803	0.495370	0.483270	0.471041	0.459251	0.476948	0.491497	0.493417	...

240 rows × 175 columns

Figura 3.1: Tabella raffigurante parte del test set

una più ampia gamma di variazioni emotive, riducendo il rischio di overfitting. Aumentare la diversità del dataset con campioni leggermente modificati aiuta a rafforzare la capacità del modello di generalizzare a nuovi dati, migliorando la robustezza e l'affidabilità delle previsioni emotive. In questo progetto di tesi nello specifico per ampiare il dataset sono stati applicati dei rumori bianchi al segnale audio, time-stretch che ha modificato la velocità di riproduzione dell'audio e infine pitch-shift che ha alterato l'altezza del suono. Nel Listato 3.1 sottostante troverete descritti i vari passaggi che sono stati effettuati per atture questo processo. Inizialmente definiamo la cartella d'origine `input_folder` da cui prendere i file audio originali e quella di output `output_folder` in cui salvare i campioni modificati, poi nel caso la cartella di output non fosse già stata creata, effettua questo passaggio. I due cicli `for` scorrono ricorsivamente tutti i file di `input_folder` per identificare tutti quelli con estensione `.wav`, sui quali si applicherà data augmentation. Poi utilizzando `Librosa`, il file audio viene caricato in memoria come array audio con la *frequenza di campionamento* `sr`. Dopodiché vengono applicate le tecniche di data augmentation quali, aggiunta di rumore bianco uniforme al segnale audio originale, modifica della velocità del campione con un `rate=1.2`, che aumenta la velocità del 20%, mantenendo invariata la tonalità

	Emotions	Actor	Chroma_1	Chroma_2	Chroma_3	Chroma_4	Chroma_5	Chroma_6	Chroma_7	Chroma_8	...
60	neutral	2	0.449431	0.431913	0.422490	0.454994	0.449320	0.466227	0.483222	0.501104	...
61	neutral	2	0.486225	0.485757	0.481704	0.490454	0.504468	0.513672	0.523946	0.543750	...
62	neutral	2	0.525115	0.512116	0.539546	0.533606	0.552692	0.572779	0.585125	0.575789	...
63	neutral	2	0.479317	0.478523	0.474578	0.484419	0.529331	0.550069	0.539958	0.514924	...
64	calm	2	0.566381	0.508163	0.506020	0.521153	0.540124	0.567265	0.549716	0.549326	...
...	...	...	...	...	...	...	...	...	...	...	...
1375	surprised	23	0.502991	0.522106	0.548523	0.567805	0.586772	0.576823	0.551229	0.530235	...
1376	surprised	23	0.507962	0.505127	0.529315	0.554288	0.607912	0.626283	0.578955	0.536716	...
1377	surprised	23	0.511948	0.510118	0.524634	0.555045	0.590097	0.575568	0.530507	0.515312	...
1378	surprised	23	0.437533	0.451178	0.460808	0.477158	0.512867	0.547356	0.496269	0.425153	...
1379	surprised	23	0.495292	0.530780	0.559124	0.578081	0.589402	0.568826	0.526266	0.496273	...

1200 rows × 175 columns

Figura 3.2: Tabella raffigurante parte del training set

e infine shift della tonalità, che ha aumentato la tonalità dell'audio di due semitoni. Infine, avviene il salvataggio dei file modificati che verranno organizzati in `output_folder` nello stesso modo in cui si trovavano nella cartella di input, con l'aggiunta di un prefisso nel nome che indica il tipo di *augmentation* applicata.

```
# Set the paths to the input and output folders
input_folder = '...\\Audio_Speech_Actors_01-24'
output_folder = '...\\Audio_Speech_Actors_01-24_augmented'
# Create the output folder if it doesn't exist
if not os.path.exists(output_folder):
    os.mkdir(output_folder)

augmented_file_count = 0

for root, dirs, files in os.walk(input_folder):
    for file in files:
        if file.endswith('.wav'):
            # Load the audio file
            audio, sr = librosa.load(os.path.join(root, file))
            noisy_audio = audio + 0.1 * np.ones(len(audio))
            stretched_audio = librosa.effects.time_stretch(audio, rate=1.2)
            pitch_shifted_audio = librosa.effects.pitch_shift(audio, sr=sr,
                n_steps=2)
            # Get the relative path of the input file
            rel_path = os.path.relpath(root, input_folder)
```

```
20              # Create the corresponding output folder structure
21              out_path = os.path.join(output_folder, rel_path)
22              os.makedirs(out_path, exist_ok=True)
23              # Save augmented audio files in the output folder
24              sf.write(os.path.join(out_path, 'noisy_' + file), noisy_audio,
                    sr)
25              sf.write(os.path.join(out_path, 'stretched_' + file),
                    stretched_audio, sr)
26              sf.write(os.path.join(out_path, 'pitchshifted_' + file),
                    pitch_shifted_audio, sr)
```

Listing 3.1: Codice Python per Data augmentation

3.2 Impostazione degli esperimenti

In questa sezione verranno descritte le varie fasi del processo di addestramento dei modelli, a partire dalle metriche utilizzate per la valutazione, la cross-validation senza la scelta dei parametri e infine la grid search che ci ha permesso di ottenere migliori risultati grazie al tuning degli iper-parametri.

3.2.1 Metriche usate

In questa sottosezione vengono presentate le principali metriche utilizzate per valutare le prestazioni dei modelli sviluppati: l'accuracy, l'F1-score e la matrice di confusione. L'accuracy fornisce una misura generale della percentuale di previsioni corrette effettuate dal modello. Tuttavia, nei casi in cui i dati sono sbilanciati tra classi, questa metrica può risultare insufficiente. Per questo motivo, è stata introdotta l'F1-score, una metrica che bilancia precision e recall, rendendola particolarmente utile per valutare modelli in presenza di classi minoritarie. Infine, la matrice di confusione offre una panoramica dettagliata delle prestazioni del modello per ciascuna classe, evidenziando il numero di previsioni corrette e gli errori di classificazione.

F1-score

L'*F1-score* può essere interpretato come una media armonica di *precision* e *recall*, in cui l'F1 score raggiunge il suo valore migliore a 1 quando precision e recall sono "perfetti", ovvero quando non ci sono né falsi positivi né falsi negativi. Al contrario il valore peggiore è 0 quando non ci sono veri positivi o quando ci sono solo errori. Il contributo relativo di precision e recall al calcolo dell'F1 score è uguale. La formula per l'F1 score è:

$$\mathrm{F1} = \frac{2 \cdot \mathrm{TP}}{2 \cdot \mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{3.3}$$

I principali elementi della formula sono *True Positives (TP)*, *False Negatives (FN)* e *False Positives (FP)*. I True Positives (TP) rappresentano i casi correttamente identificati come positivi dal modello. Ad esempio, se il modello è progettato per rilevare emozioni positive e classifica correttamente un'emozione come positiva, si tratta di un vero positivo. Al contrario, i False Negatives (FN) sono i casi che erano positivi, ma il modello ha classificato come negativi; questo accade, ad esempio, quando un'emozione positiva viene erroneamente classificata come negativa, causando una "perdita" di un caso positivo. Infine, i False Positives (FP) sono i casi negativi che il modello ha classificato erroneamente come positivi. Ad esempio, se il modello rileva un'emozione positiva in un caso che non lo è, aggiunge in modo errato un caso positivo, creando un falso positivo[39].

Accuracy

L'*accuracy* di un modello si basa sulle previsioni corrette effettuate per le classi di un documento. I file di addestramento sono raggruppati e quindi verificati rispetto agli algoritmi per predire l'accuratezza. L' 80% del set di dati viene utilizzato per generare il classificatore, mentre il 20% viene confrontato con il classificatore per misurare l'efficacia della previsione dei risultati. L'accuratezza è il numero di previsioni corrette rispetto a tutte le

previsioni effettuate. Si calcola dividendo il numero di previsioni corrette per il numero totale di previsioni (Eq. 3.4).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{3.4}$$

Questo valore fornisce una misura semplice e diretta della capacità del modello di classificare correttamente i dati. In aggiunta, rispetto a F1-score abbiamo i *True Negatives*, che rappresentano le istanze negative previste correttamente come negative. Una consideazione per quanto riguarda i dati sbilanciati, l'accuratezza potrebbe non riflettere effettivamente la capacità del modello di distinguere le classi, poiché il modello potrebbe avere alte prestazioni solo nella classe maggioritaria. In questi casi, metriche come la matrice di confusione, precision, recall e F1-score offrono un quadro più completo [40].

Confusion Matrix

La *matrice di confusione* è una tabella incrociata utilizzata per rappresentare i parametri di base su cui è possibile ottenere le prestazioni predittive di un modello di classificazione. Le prestazioni di ogni modello di classificazione possono essere dedotte in base a quanto esso classifica correttamente i casi o commette errori di classificazione. Gli elementi n_{ij} nella matrice di confusione (dove i è l'indice della riga e j è l'indice della colonna) indicano i casi appartenenti alla classe i che sono stati classificati come j. Di conseguenza, gli elementi nella diagonale (n_{ii}) rappresentano i casi classificati correttamente, mentre gli elementi fuori dalla diagonale sono quelli classificati erroneamente. Il numero totale di casi è rappresentato nell' Eq. 3.5 [41]:

$$N = \sum_{i=1}^{M} \sum_{j=1}^{M} n_{ij} \tag{3.5}$$

In questo progetto di tesi la matrice di confusione è stata utilizzata per valutare le prestazioni di ciascun modello di classificazione sui dati di test. Quindi la matrice di confusione consente di analizzare le classificazioni cor-

rette e errate per ciascuna classe, rendendo visibili i tipi di errore commessi dai modelli e fornendo un quadro più dettagliato delle loro prestazioni su ogni classe di output. Per brevità non verrano mostrate tutte le matrici di confusione create per ogni modello, ma è stata creata una *heatmap* in modo da avere una rappresentazione visiva delle accuratezze per ciascuna classe nei diversi modelli, permettendo un confronto diretto delle prestazioni di ogni modello per ogni classe di emozione(Fig. 3.3).

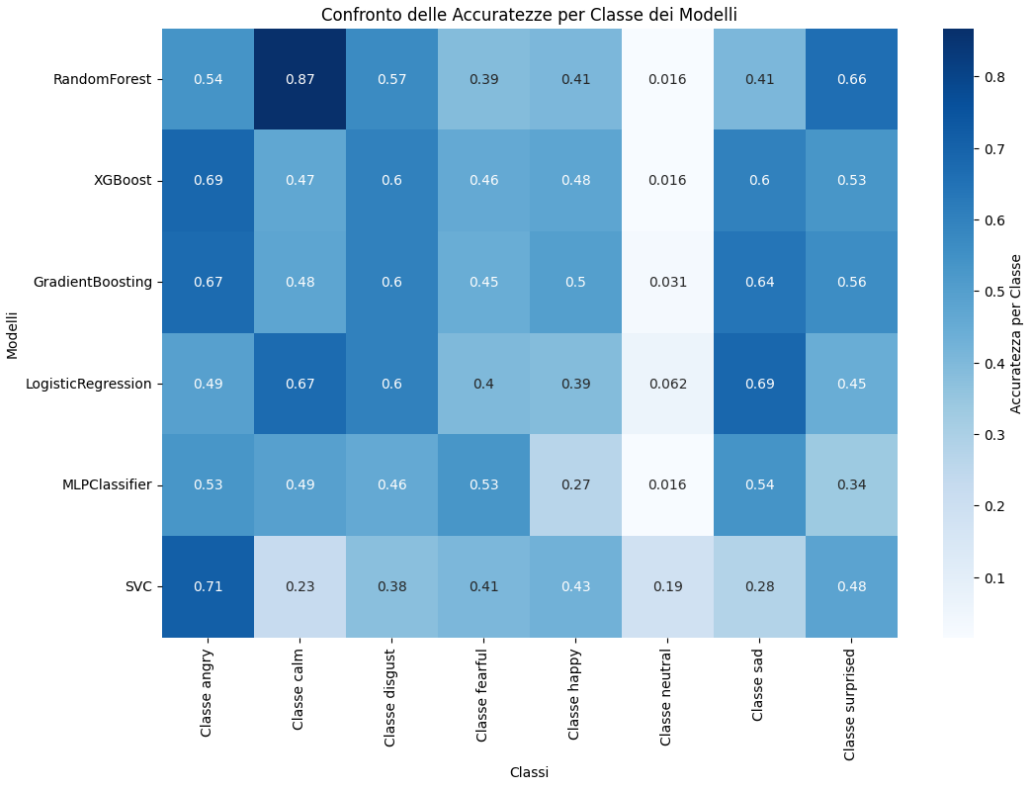

Figura 3.3: Heatmap dell'accuracy dei modelli a confronto

I colori più scuri indicano un'accuratezza più alta, suggerendo che il modello ha riconosciuto con maggiore precisione le emozioni corrispondenti. Dall'analisi della heatmap, emerge che i vari modelli si comportano in modo diverso a seconda delle emozioni. Ad esempio, il modello MLP si distingue per l'elevata accuratezza nel riconoscere l'emozione *angry* (81%) e *disgust*

(75%), ma ha una prestazione quasi nulla per la classe *calm* (3,1%). Questo suggerisce che l'MLP ha capacità di classificazione molto specifiche per alcune emozioni, ma fatica a generalizzare su altre.

Al contrario, il modello Gradient Boosting mostra una distribuzione di accuratezza più bilanciata, con valori attorno a 66% per *angry*, *disgust*, e *happy* e il 50% per *sad*. Tuttavia, anche il Gradient Boosting incontra difficoltà nel riconoscere la classe *fearful*, per la quale tutti i modelli mostrano un'accuratezza pari a zero, indicando che nessuno dei modelli è stato in grado di classificare correttamente questa emozione. Questo potrebbe essere dovuto a una bassa rappresentazione di questa classe nel dataset o alla somiglianza di questa emozione con altre, rendendola più difficile da riconoscere.

Il modello Logistic Regression ottiene prestazioni modeste, con accuracy attorno al 66% per le emozioni *angry* e disgust, ma valori inferiori per altre emozioni come *calm* e *sad*. XGBoost e Random Forest mostrano valori simili, mantenendo una buona accuratezza per emozioni come *disgust* e *happy* ma presentando anch'essi difficoltà con *calm* e *fearful*. Infine, l'SVC offre prestazioni accettabili, soprattutto per la classe *disgust*, ma con accuratezze variabili sulle altre emozioni. L'assenza di classificazioni corrette per la classe fearful evidenzia un problema comune a tutti i modelli, che potrebbe indicare la necessità di rivedere la rappresentazione di questa classe nel dataset o di esplorare tecniche di bilanciamento per migliorare le prestazioni. Inoltre, le prestazioni divergenti tra le varie classi suggeriscono che alcuni modelli hanno punti di forza su emozioni specifiche ma non sono in grado di garantire una precisione uniforme su tutte le categorie.

3.2.2 Cross-validation

Per quanto riguarda la parte degli esperimenti effettuati, siamo partiti con una cross-validation con 10 fold, nella quale il training set è stato suddiviso nuovamente in modo da prendere ad ogni iterazione un fold composto da due attori di ambo i sessi come test set e i restanti 18 (9 donne e 9 uomini) come training set. Nel codice sottostante (Listing 3.2) è stata implementata

una funzione che accetta come parametro un modello, che sarà il modello da addestrare e valutare. Il ciclo `for` viene eseguito 10 volte (una per ogni fold) e ad ogni iterazione prende due attori di entrambi i sessi che utilizza come fold di test. Poi viene fatta una selezione in cui dentro a `cross_test_set` ci saranno le righe del training set per cui l'attore è uno dei due scelti per il fold corrente, o maschio o femmina, e `cross_training_set` conterrà tutte le righe rimanenti. Dopodiché vengono separate le etichette della classe `Emotions` inserite dentro a `y_train` e `y_test`. Mentre `X2d_train` e `X2d_test` contengono le feature da utilizzare per il modello. Infine, il modello viene addestrato sul fold di training, costituito dagli attori non selezionati per il fold di test e poi vengono calcolate le metriche descritte nella sezione precedente, quali Accuracy e F1-score.

Risultati

Per ottenere una visione completa del processo di addestramento, è stata effettuata una cross-validation generica utilizzando i parametri di deafult in ogni modello, quindi `random_state` e `n_estimators=50` per chi li possedeva, in modo da poter poi confrontare i risultati con la cross validation con il tuning degli iper-parametri.

	RF	**XGB**	**GB**	**LR**	**MLP**	**SVC**
Accuratezza	0.3700	0.4096	0.4079	0.3733	0.3367	0.3817
F1-score	0.3311	0.3925	0.3853	0.3351	0.2981	0.3326

Tabella 3.1: Confronto delle performance dei modelli in termini di accuratezza e F1-score

La Tabella 3.1 mostra un confronto delle prestazioni di diversi modelli di machine learning utilizzati per il riconoscimento delle emozioni vocali, valutati in termini di accuratezza e F1-score. In questo caso, il modello con la migliore accuratezza è XGBoost, con un valore di 0.4096, seguito dal Gradient Boosting con un'accuratezza simile di 0.4079. Questi risultati suggeriscono

che i modelli basati su boosting tendono a funzionare meglio rispetto ad altri algoritmi su questo dataset. L'F1-score, che bilancia precision e recall, è particolarmente rilevante nei contesti in cui il dataset è sbilanciato, come nel riconoscimento delle emozioni, dove alcune classi possono essere più rappresentate di altre. Anche qui, i modelli XGBoost e Gradient Boosting ottengono i migliori risultati, con F1-score pari a 0.3925 e 0.3853 rispettivamente, il che indica una maggiore capacità di gestire correttamente le classi minoritarie. Al contrario, il modello MLP ottiene il valore più basso di F1-score, pari a 0.2981, suggerendo una minore affidabilità nel rilevamento accurato delle diverse emozioni.

```
def cross_validation(model):
    model_accuracy = []
    model_f1 = []
    for i in range(10):
        m = training_actors_m[i]
        f = training_actors_f[i]
        cross_test_set = training_set.loc[training_set['Actor'].isin([m, f])
            ]
        cross_training_set = training_set.drop(cross_test_set.index)
        y_train = cross_training_set["Emotions"]
        X2d_train = cross_training_set[dataset.columns[6:]]
        y_test = cross_test_set["Emotions"]
        X2d_test = cross_test_set[dataset.columns[6:]]
        # fit the classifier
        model.fit(X2d_train, y_train)
        # compute the score and record it
        model_accuracy.append(model.score(X2d_test, y_test))
        y_pred = model.predict(X2d_test)
        model_f1.append(f1_score(y_test, y_pred, average="macro"))
    accuracy.append(model_accuracy)
    f1.append(model_f1)
```

Listing 3.2: Codice Python per Cross validation

3.2.3 Grid Search

La *Grid Search* è una tecnica di ottimizzazione che effettua una ricerca esaustiva su una griglia di iperparamtri per trovare la combinazione di

parametri che massimizza l'accuratezza di un modello di classificazione. La ricerca viene effettuata in maniera parallela in questo caso e ogni combinazione di parametri è valutata tramite una procedura di cross-validation. Ora verrà analizzata nel dettaglio la funzione Grid Search implementata manualmente. Il codice nel Listato 3.3 contiene due funzioni, `save_model_params` accetta due parametri: `filename`, che è il nome del file in cui salvare i dati, e `data`, che rappresenta i parametri del modello che si desidera memorizzare. La funzione apre il file specificato in modalità scrittura binaria ('wb') e utilizza `pickle.dump(data, f)` per serializzare e salvare i dati nel file. Grazie all'uso del contesto with, il file viene chiuso automaticamente al termine dell'operazione. La funzione `load_model_params`, invece, serve per recuperare i parametri salvati.

```
def save_model_params(filename, data):
    with open(filename, 'wb') as f:
        pickle.dump(data, f)
def load_model_params(filename):
    if os.path.exists(filename):
        with open(filename, 'rb') as f:
            return pickle.load(f)
    return None
```

Listing 3.3: Codice Python per Grid Search

Alla funzione Grid Search verranno passati i seguenti parametri come nel Listing 3.4:

- `model`: il modello di classificazione di cui ottimizzare i parametri;
- `param_grid`: un dizionario con i parametri e i rispettivi valori su cui fare la ricerca;
- `save_file`: il percorso di un file per poter salvare i risultati intermedi.

Dopodiché per ottimizzare la ricerca sono state usate delle librerie specifiche, descritte nel capitolo precedente, per ridurre i tempi di esecuzione. Quindi con `load_model_params` si cerca di caricare i migliori risultati precedentemente salvati e poi si controlla se esistono già dei risultati salvati. In quel caso

vengono recuperati `best_score` e `best_params` per evitare di ripetere le valutazioni delle combinazioni già eseguite. Infine, con `product(*param_values)` vengono generate tutte le possibili combinazioni dei valori di parametri, che saranno poi salvate nella lista `param_combinations`. Ogni combinazione rappresenta un insieme unico di parametri da testare nel modello.

```
def GridSearch(model, param_grid, save_file):
    best_score = 0.0
    best_params = {}
    saved_data = load_model_params(save_file)
    if saved_data:
        print("Caricamento dei risultati intermedi...")
        best_score, best_params = saved_data
    param_names = param_grid.keys()
    param_values = param_grid.values()
    param_combinations = list(product(*param_values))
```

Listing 3.4: Preparazione dei dati

Proseguendo con il codice, nel listing 3.5, troviamo la funzione `evaluate_combination`, in cui abbiamo `param_combination` che associa a ciascun nome di parametro il suo valore e questa combinazione di parametri sarà poi usata per creare e testare il modello. Dopodiché viene creato un sottoinsieme del dataset, `cross_test_set`, contente esclusivamente i dati dei due attori (maschio e femmina) selezionati in una determinata iterazione del ciclo, che verrà poi usato come test set in quella specifica iterazione. Mentre a `cross_training_set` verrà assegnato il set di addestramento creato eliminando dal training set i dati degli attori selezionati per il test set. `y_train` e `y_test` contengono le etichette di classe (emozioni) rispettivamente per il set di addestramento e quello di test, ottenute selezionando la colonna `Emotions` di `cross_training_set` e `cross_test_set`.

```
def evaluate_combination(params):
        param_combination = dict(zip(param_names, params))
        model_accuracy = []
        model_f1 = []
        for i in range(10):
            m = training_actors_m[i]
```

```
            f = training_actors_f[i]
            cross_test_set = training_set.loc[training_set['Actor'].isin([m,
                f])]
            cross_training_set = training_set.drop(cross_test_set.index)
            y_train = cross_training_set["Emotions"]
            x_train = cross_training_set[dataset.columns[6:]]
            y_test = cross_test_set["Emotions"]
            x_test = cross_test_set[dataset.columns[6:]]
```

Listing 3.5: Cross validation

Nel listing 3.6 viene creata una nuova istanza del modello alla quale vogliamo passare come parametri la coppia chiave-valore in `param_combination` e dopodiché con la funzione `fit` viene allenato il modello sui dati di training con lo scopo di apprendere la relazione tra le caratteristiche contenute in `X2d_train` e le etichette in `y_train`. Il codice in questo punto è stato ripreso dalla sezione della cross validation (Listing. 3.2) con qualche aggiunta. In seguito, viene calcolata l'accuratezza del modello, ovvero la percentuale di previsioni corrette rispetto al totale delle previsioni.
Nella lista `model_accuracy` verranno aggiunti tutti i valori numerici restituiti dalla funzione `score`, che rappresentano l'accuratezza del modello sui dati di test. Per quanto riguarda la predizione viene utilizzato `predict`, il quale usa il modello per prevedere le etichette dei dati di test e produrre una lista che verrà utilizzato nel calcolo di *F1*. La metrica *F1 score* viene calcolata tra le etichette previste (`y_pred`) e quelle reali (`y_test`), e l'utilizzo di `average="macro"` fa sì che il calcolo prenda la media semplice degli F1 score di ciascuna classe, dando lo stesso peso a ciascuna classe.

```
# Crea una nuova istanza del modello
model_instance = model.__class__(**param_combination)
# Fai il fitting del modello
model_instance.fit(X2d_train, y_train)
# Registra i risultati
model_accuracy.append(model_instance.score(X2d_test, y_test))
y_pred = model_instance.predict(X2d_test)
model_f1.append(f1_score(y_test, y_pred, average="macro"))
```

Listing 3.6: Gestione e creazione istanza del modello

Come si può osservare nel listato 3.7, viene calcolata la media dell'accuratezza, che corrispondee alla percentuale di previsioni corrette, attraverso tutte le iterazioni di valutazione del modello. Quindi dividendo la somma delle acuratezze per il numero totale di valori otteniamo `avg_accuracy`. Stessa cosa viene fatta anche con la metrica F1, dividendo la somma per il numero di valori otterremmo la media F1 score. Questa media è utile per comprendere la qualità delle previsioni del modello, bilanciando la precisione e la sensibilità per ogni classe in maniera equilibrata. Infine questa parte di codice ritorna la configurazione dei parametri e le due medie delle metriche, che rappresentano le performance complessive del modello per questa particolare combinazione di parametri.

```
avg_accuracy = sum(model_accuracy) / len(model_accuracy)
avg_f1 = sum(model_f1) / len(model_f1)
return param_combination, avg_accuracy, avg_f1
```

Listing 3.7: Calcolo dell'accuracy e F1-score

La libreria Joblib, come spiegato ampiamente nello scorso capitolo, è stata utilizzata per velocizzare l'esecuzione del codice. In questo caso (listing 3.8) si utilizza la funzione `Parallel` che permette di eseguire operazioni in parallelo creando un oggetto che coordina l'esecuzione di una lista di operazioni in parallelo. Innanzitutto utilizzando `n_jobs=-1` verranno impiegati tutti i processori disponibili del computer e grazie all'uso di `delayed`, un wrapper, verrà ritardata l'esecuzione di una funzione con i parametri specificati. In questo modo la funzione della libreria Joblib può eseguire questa chiamata quando un processore diventa disponibile.

```
results = Parallel(n_jobs=-1)(delayed(evaluate_combination)(params) for
    params in param_combinations)
```

Listing 3.8: Parallelizzazione della valutazione delle combinazioni

Viene poi effettuato un controllo, nel caso in cui l'accuratezza media è maggiore di `best_score`, variabile che mantiene traccia del valore più alto di

accuratezza media trovato fino a quel punto, allora verrà aggiornata con il nuovo valore (Listing 3.9). Mentre in `best_params` viene aggiornato con la configurazione di parametri attuale, l'insieme di questi dati viene salvato nel file `save_file`.

```
for param_combination, avg_accuracy, avg_f1 in results:
        if avg_accuracy > best_score:
            best_score = avg_accuracy
            best_params = param_combination
    # Salva il miglior risultato raggiunto finora
        save_model_params(save_file, (best_score, best_params))
        print(f"Parameters: {param_combination}")
        print(f"Average Accuracy: {avg_accuracy}")
        print(f"Average F1 Score: {avg_f1}")
        print("----------------------")

    print("Best parameters:")
    print(best_params)
    print(f"Best Accuracy: {best_score}")
    print("----------------------")
```

Listing 3.9: Stampa dei risultati dell'addestramento

3.3 Esito della valutazione

Dopo aver descritto il processo di Data augmentation applicato al dataset, la procedura di cross-validation e la Grid Search per ottimizzare la combinazione degli iperparametri, è giunto il momento di analizzare i risultati e scoprire quale modello ha ottenuto le migliori prestazioni.

3.3.1 Prestazioni dei modelli

A seguito della valutazione dei modelli attraverso diversi metodi, l'ultima fase è stata quella della Grid Search che ci ha permesso di ottenere i risultati migliori testando tutte le diverse combinazioni degli iperparametri scelti. Come si può vedere nella Fig. 3.4 sono rappresentate le metriche di Accuracy e F1 calcolate per ogni modello.

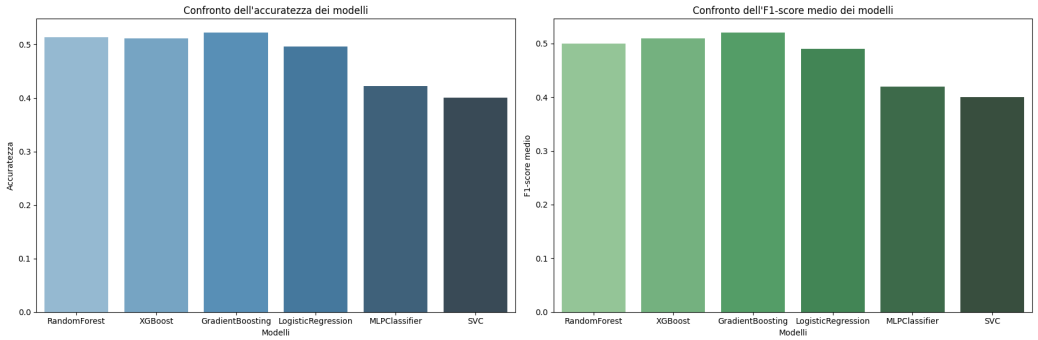

Figura 3.4: Accuracy e F1 a confronto su tutti i modelli

Random Forest

Per ottimizzare le prestazioni del modello Random Forest, sono stati condotti esperimenti utilizzando la grid search per individuare la combinazione ideale di iperparametri. La grid search consente di esplorare sistematicamente un insieme predefinito di valori per ciascun iperparametro, valutando le performance del modello per ogni combinazione attraverso il set di validazione. In questo modo, è possibile identificare la configurazione che bilancia al meglio complessità e capacità di generalizzazione. Gli iperparametri selezionati per la Random Forest sono i seguenti:

- `n_estimators:` il numero di alberi, con valori testati di 200, 400 e 600. Un numero maggiore di alberi tende a migliorare la robustezza del modello.

- `max_depth` la profondità massima di ciascun albero, con valori di 10, 15, 20 e nessun limite. La profondità massima aiuta a controllare l'overfitting.

- `min_samples_split` il numero minimo di campioni richiesti per dividere un nodo, impostato su 2 e 5. Parametri più alti possono prevenire l'overfitting.

- `min_samples_leaf` il numero minimo di campioni richiesti per essere una foglia, con valori di 1 e 2. Anche questo parametro contribuisce a controllare l'overfitting.

Iperparametro	Valore
`n_estimators`	600
`max_depth`	20
`min_samples_split`	5
`min_samples_leaf`	1

Tabella 3.2: Miglior combinazione di iperparametri per Random Forest

La miglior combinazione di iperparametri per il modello Random Forest ha raggiunto un'accuratezza del 41,17%, evidenziando un giusto bilancio tra complessità del modello e capacità di generalizzazione. Con `n_estimators` impostato a 600, il modello è in grado di sfruttare un numero elevato di alberi per catturare meglio la variabilità del dataset, migliorando la generalizzazione e aumentando la stabilità del risultato, sebbene ciò comporti un incremento nel tempo di calcolo. La profondità massima di 20 permette al modello di esplorare pattern più complessi. Inoltre, un valore di `min_samples_split` pari a 5 contribuisce a limitare la crescita del numero di nodi e a migliorare la robustezza del modello, richiedendo almeno 5 campioni per effettuare una divisione. Infine, `min_samples_leaf` impostato a 1 consente al modello di mantenere un buon livello di dettaglio nella classificazione, poiché permette che ciascun nodo foglia contenga un singolo campione.

Gradient Boosting

Per ottimizzare le prestazioni del modello di Gradient Boosting, sono stati condotti esperimenti utilizzando la tecnica di Grid Search, che permette di esplorare sistematicamente diverse combinazioni di iperparametri al fine di identificare quella che massimizza l'accuratezza del modello. Gli iperparametri influenzano direttamente la complessità, la velocità di apprendimento

e la capacità di generalizzazione del modello. L'obiettivo di questa fase è quindi trovare una configurazione che bilanci l'accuratezza e la capacità del modello di evitare l'overfitting.

- `n_estimators` i numeri di alberi, testati con 100, 200 e 300.
- `max_depth` la profondità massima, limitata a 2 e 3, per mantenere modelli meno complessi e prevenire l'overfitting.
- `subsample` la frazione di campioni da utilizzare per addestrare ciascun albero, con valori di 0.6, 0.7 e 0.8.
- `learning_rate` il tasso di apprendimento, impostato a 0.05 e 0.1, che controlla l'influenza di ogni albero nel modello finale.

Iperparametro	**Valore**
`n_estimators`	200
`max_depth`	3
`subsample`	0.8
`learning_rate`	0.05

Tabella 3.3: Miglior combinazione di iperparametri per Gradient Boosting

La combinazione ottimale di iperparametri per il modello di Gradient Boosting, che ha ottenuto la migliore accuratezza di 0.436, presenta alcune caratteristiche specifiche (Tab. 3.2). In particolare, un numero di *estimators* pari a 200 assicura un bilanciamento tra capacità di apprendimento e overfitting, mentre una profondità massima di 3 mantiene la semplicità dei singoli alberi, migliorando la generalizzazione del modello. La scelta di un tasso di apprendimento di 0.05 rallenta il processo di apprendimento, permettendo al modello di adattarsi gradualmente ai dati e riducendo il rischio di sovradattamento. Infine, un *subsample* di 0.8 indica che ogni albero viene addestrato su un sottoinsieme casuale dell'80% dei dati di training, introducendo ulteriore variabilità e favorendo la robustezza del modello.

Extreme Gradient Boosting

Per migliorare le prestazioni del modello di Extreme Gradient Boosting, è stata condotta una serie di esperimenti utilizzando un processo che ha permesso di testare diverse combinazioni di parametri rilevanti per il modello, con l'obiettivo di trovare l'equilibrio ideale tra complessità, capacità di generalizzazione e accuratezza. La grid search è stata impostata su un insieme selezionato di valori per parametri come il numero di alberi (n_estimators), la profondità massima degli alberi (max_depth) e la percentuale di colonne campionate per ogni albero (colsample_bytree). I risultati ottenuti sono stati valutati in base all'accuratezza e alla capacità del modello di evitare il sovradattamento, assicurando una buona generalizzazione sui dati di test.

- `n_estimators` il numero di alberi da costruire, con valori di 200, 300 e 400, simile a Random Forest.
- `max_depth` la massima profondità degli alberi, testata con valori di 3 e 6, per controllare la complessità del modello.
- `colsample_bytree` la frazione di colonne da campionare per costruire ciascun albero, con valori di 0.5, 0.7 e 1.0, per gestire la diversità degli alberi e migliorare la generalizzazione.

Iperparametro	**Valore**
`n_estimators`	400
`max_depth`	6
`colsample_bytree`	0.7

Tabella 3.4: Miglior combinazione di iperparametri per XGBoost

Questa tabella riassume la miglior combinazione di iperparametri trovata per il modello XGBoost, che ha raggiunto un'accuratezza ottimale di 0.4354. Gli iperparametri selezionati includono *n_estimators*, *max_depth*, e *colsample_bytree*, che hanno avuto un impatto significativo sulle prestazioni

del modello. Nello specifico, il parametro *n_estimators* è stato impostato a 400, indicando il numero di alberi che il modello costruisce durante l'allenamento. Un numero più elevato di alberi consente al modello di catturare meglio le complessità del set di dati, ma può incrementare anche il rischio di overfitting. Il parametro *max_depth*, impostato a 6, definisce la profondità massima di ogni albero. Una profondità maggiore permette al modello di apprendere relazioni più complesse, ma un valore troppo alto può aumentare il rischio di overfitting. Infine, *colsample_bytree* è stato impostato a 0.7, limitando la frazione di caratteristiche utilizzate in ogni albero e introducendo diversità tra gli alberi, il che riduce la varianza complessiva del modello e migliora la generalizzazione.

Logistic Regression

Per ottimizzare le prestazioni del modello di Regressione Logistica, è stata condotta una grid search sui principali iperparametri, come il parametro di regolarizzazione `C` e il metodo di ottimizzazione `solver`. La grid search consente di esplorare combinazioni di valori iperparametrici predefiniti, al fine di individuare la configurazione ottimale che massimizza l'accuratezza di classificazione sul set di validazione. In particolare, si è ritenuto utile testare valori di `C` che variano tra 0.01 e 10, per valutare l'effetto della regolarizzazione sul modello, e diversi solver che influenzano le tecniche di ottimizzazione per garantire una convergenza efficiente. Di seguito, vengono riportati i risultati della migliore configurazione trovata per la Regressione Logistica.

- `C:` il parametro di regolarizzazione, testato a 0.01, 0.1, 1 e 10. Valori più bassi di C aumentano la regolarizzazione.
- `solver:` il metodo di ottimizzazione, scelto tra "lbfgs", "liblinear" e "saga", per determinare la tecnica di apprendimento.

La miglior combinazione di iperparametri trovata per il modello di Logistic Regression, con un'accuratezza massima di 0.3908, include i valori specificati nella tabella sopra (Tab. 3.4). L'iperparametro `C` è impostato a 0.01,

Iperparametro	Valore
`C`	0.01
`solver`	liblinear

Tabella 3.5: Miglior combinazione di iperparametri per Logistic Regression

un valore basso che tende a ridurre l'intensità della regularizzazione. Ciò permette al modello di evitare overfitting nei casi in cui le caratteristiche sono numerose e correlate, favorendo una soluzione più regolare. Il parametro `solver` è stato impostato su *liblinear*, particolarmente adatto per dataset piccoli o per problemi di classificazione binaria e multiclasse ridotti, migliorando l'efficienza di ottimizzazione per questo modello lineare.

Multi-Layer Perceptron

Per l'addestramento del modello Multi-Layer Perceptron (MLP), abbiamo esplorato diverse configurazioni di iperparametri utilizzando una grid search. Questo approccio ci ha permesso di identificare la combinazione di parametri ottimale per massimizzare le prestazioni del modello, mantenendo un equilibrio tra accuratezza e complessità. In particolare, la ricerca ha interessato la struttura della rete (numero e dimensioni dei livelli nascosti), il termine di regolarizzazione (alpha) e il tasso di apprendimento iniziale (learning_rate_init). L'obiettivo era ottenere un modello capace di apprendere pattern complessi senza incorrere in overfitting, garantendo stabilità nel processo di ottimizzazione e favorendo una buona generalizzazione dei risultati

- `hidden_layer_sizes` la struttura delle dimensioni dei livelli nascosti, testando configurazioni come (50, 50), (50, 50, 50) e (100,).
- `alpha` il termine di regolarizzazione, testato a 0.0001, 0.001 e 0.01.
- `learning_rate_init`: il tasso di apprendimento iniziale, con valori di 0.001 e 0.01.

Iperparametro	Valore
`hidden_layer_sizes`	(50, 50, 50)
`alpha`	0.01
`learning_rate_init`	0.001

Tabella 3.6: Miglior combinazione di iperparametri per MLP

La combinazione di iperparametri riportata nella tabella 3.5 ha ottenuto la migliore accuratezza per il modello MLP nel nostro esperimento, raggiungendo un valore massimo di 0.4108. La configurazione della rete include tre strati nascosti, ciascuno composto da 50 neuroni, scelti per bilanciare la capacità di apprendimento della rete e limitare il rischio di overfitting. Il parametro alpha è impostato su 0.01, contribuendo a regolare la penalizzazione per la complessità della rete e a evitare il sovradimensionamento; valori più alti di alpha infatti limitano l'overfitting, ma possono influire negativamente sull'apprendimento di pattern complessi. Il tasso di apprendimento iniziale, `learning_rate_init`, è pari a 0.001, il che consente una convergenza stabile durante l'ottimizzazione e riduce il rischio di oscillazioni nel processo di addestramento. In generale, questa configurazione rappresenta un buon equilibrio tra accuratezza e complessità del modello, anche se l'accuratezza ottenuta suggerisce che potrebbe essere necessaria una configurazione più avanzata o una diversa architettura per ottenere prestazioni migliori.

Support Vector Classification

Il modello SVC è stato selezionato per la sua efficacia nel trattare con problemi di classificazione e la sua capacità di generalizzare bene anche in contesti con dati limitati o complessi. Abbiamo utilizzato la tecnica della grid search per esplorare diverse combinazioni di iperparametri e identificare la configurazione ottimale. La grid search consente di testare esaustivamente un insieme definito di valori per ciascun iperparametro, con l'obiettivo di individuare quella combinazione che massimizza le prestazioni del modello sul set di training.

- `C` il parametro di regolarizzazione, con valori di 0.1, 1 e 10.
- `kernel` il tipo di kernel da utilizzare, con opzioni "linear" e "rbf".
- `gamma` il parametro gamma per il kernel, testato come "scale" e "auto".

Iperparametro	Valore
`C`	0.1
`kernel`	linear
`gamma`	scale

Tabella 3.7: Miglior combinazione di iperparametri per SVC

Per l'SVC, la migliore combinazione di iperparametri ha incluso un valore di `C` pari a 0.1, un kernel `lineare`, e gamma impostato su scale. Un valore di `C` più basso come 0.1 implica una maggiore penalizzazione per gli errori, contribuendo a un modello più semplice e meno incline all'overfitting. La scelta di un kernel lineare è adatta in questo contesto, suggerendo che i dati sono separabili in modo lineare nello spazio delle caratteristiche. Infine, con gamma impostato su scale, il modello utilizza un'influenza proporzionata al numero di feature, il che aiuta ad adattarsi meglio a dataset con distribuzioni complesse. Questa configurazione ha prodotto un'accuracy del 40.25%, il che rappresenta il miglior risultato ottenuto per la SVC in termini di capacità di classificazione rispetto ad altre combinazioni provate. Tuttavia, il punteggio indica ancora un margine di miglioramento, suggerendo possibili approfondimenti su parametri aggiuntivi o altre tecniche di pre-elaborazione.

Risultati

La Tabella 3.7 riporta l'accuratezza ottenuta dai diversi modelli sul set di test. Tra i modelli considerati utilizzando il dataset originario, Gradient Boosting ha raggiunto la migliore accuratezza con un valore di 0,4358, seguito da XGBoost con un'accuratezza di 0,4317. Anche se l'accuratezza varia di poco tra i vari modelli, il Logistic Regression ha ottenuto il punteggio più

basso (0,3908), suggerendo una minore capacità di generalizzazione rispetto agli altri algoritmi. Random Forest e MLP hanno mostrato prestazioni intermedie, mentre SVC ha raggiunto un'accuratezza di 0,4025. Questi risultati evidenziano come i modelli di tipo boosting tendano a ottenere prestazioni migliori rispetto agli altri approcci utilizzati.

	RF	**XGB**	**GB**	**LR**	**MLP**	**SVC**
Original	0,4117	0,4317	**0,4358**	0,3908	0,4108	0,4025
Augmented	0,4117	0.4354	**0,4358**	0,3908	0,4108	0,4025

Tabella 3.8: Accuratezza dei modelli con dataset originale e aumentato

Inoltre, è stato fatto un confronto tra l'addestramento dei modelli con il dataset originale e dataset aumentato, come si può vedere nella Tabella 3.8, e possiamo dire che la maggior parte degli algoritmi hanno mantenuto la stessa accuratezza, ad eccezione di XGB che ha mostrato un miglioramento significativo, passando da 0,4317 a 0,4354. Gradient Boosting si conferma il modello con le prestazioni migliori, evidenziando la sua robustezza e stabilità su dataset con e senza augmentazione.

	RF	**XGB**	**GB**	**LR**	**MLP**	**SVC**
Cross validation	0.3700	**0.4096**	0.4079	0.3733	0.3367	0.3817
Grid Search	0.4117	0.4354	**0.4358**	0.3908	0.4108	0.4025

Tabella 3.9: Confronto delle performance dei modelli in termini di accuratezza per cross validation e grid search

In generale, i valori di accuratezza ottenuti tramite Grid Search risultano superiori rispetto a quelli ottenuti con Cross Validation con parametri di default per ciascun modello (Tab. 3.9). Ad esempio, il modello XGB mostra un incremento di accuratezza da 0.4096 a 0.4354, mentre il modello

GB passa da 0.4079 a 0.4358. In particolare, il modello MLP trae vantaggio dal tuning, mostrando un aumento significativo dell'accuratezza da 0.3367 a 0.4108. Questo risultato suggerisce che la sua performance dipende fortemente dalla corretta selezione dei parametri. In conclusione, i risultati ottenuti con Grid Search, grazie all'ottimizzazione degli iper-parametri, sono nettamente superiori rispetto a quelli della Cross Validation con parametri di default per tutti i modelli testati.

Conclusioni

Lo Speech Emotion Recognition ha un grande potenziale di utilità nella vita quotidiana, con applicazioni che potrebbero apportare benefici in diversi ambiti. La capacità di riconoscere le emozioni dal parlato potrebbe rendere le interazioni con le tecnologie più naturali ed empatiche, migliorando l'esperienza utente in numerosi settori. Inoltre, l'uso del SER può essere determinante in contesti dove la componente emotiva riveste un ruolo cruciale, contribuendo a una maggiore adattabilità e personalizzazione dei servizi. L'integrazione di questa tecnologia potrebbe trasformare l'interazione uomo-macchina, ampliando le sue applicazioni in modo sempre più pervasivo e utile nella vita di tutti i giorni. In questo volume di tesi, l'argomento del Riconoscimento delle Emozioni Vocali è stato introdotto inizialmente con una trattazione teorica, mirata a creare una solida base di comprensione per tutto il processo. Questo approccio ha permesso di chiarire i concetti fondamentali, per poi passare alla parte più pratica, relativa alla configurazione degli esperimenti. L'obiettivo principale di questo lavoro era confrontare i risultati ottenuti da tecniche di Machine Learning avanzate con quelli di esperimenti preesistenti, così da valutare i vantaggi degli approcci moderni. I risultati degli esperimenti hanno confermato un netto miglioramento delle performance rispetto alle tecniche basilari, con il Gradient Boosting che si è rivelato l'algoritmo più performante. Tuttavia, è importante ricordare che questo progetto rappresenta solo l'inizio di un percorso di ricerca: ulteriori sviluppi potrebbero includere la valutazione di nuove combinazioni di parametri, con l'obiettivo di ottimizzare ulteriormente l'accuratezza dei modelli e spinger-

si verso risultati ancora migliori in futuro. Di seguito verranno elencati i possibili sviluppi futuri:

- Integrazione di Reti Neurali Recurrenti (RNN) e Transformer utilizzando modelli di deep learning, particolarmente indicati per l'elaborazione di sequenze audio, per migliorare la comprensione della dinamica temporale dell'espressione emotiva nel parlato.

- Uso di modelli pre-addestrati e Transfer Learning come *BERT* o *wav2vec*, che potrebbero migliorare il riconoscimento delle emozioni vocali attraverso il trasferimento di conoscenze già apprese su dataset di grandi dimensioni.

- Analisi multimodale integrando dati vocali con altre modalità, come l'analisi delle espressioni facciali o dei gesti, per migliorare l'accuratezza del riconoscimento delle emozioni attraverso una visione più completa degli stati emotivi.

- Integrazione in sistemi educativi e di supporto emotivo, come il monitoraggio dell'umore degli studenti durante le lezioni o dei pazienti in terapia, permettendo un feedback in tempo reale.

- Analisi delle emozioni in Streaming Audio sviluppando un sistema in grado di riconoscere le emozioni vocali in flussi audio continui, come podcast o video in diretta, per adattare dinamicamente i servizi in base alle emozioni rilevate.

- Ampliamento della varietà di dataset integrando e confrontando dataset da lingue e culture diverse per migliorare la generalizzazione dei modelli SER. Esplorare l'integrazione di dataset come IEMOCAP, Emo-DB e altri, mirati a lingue specifiche.

Bibliografia

[1] K. Ezzameli and H. Mahersia. Emotion recognition from unimodal to multimodal analysis: A review. *Information Fusion*, 99:101847, 2023. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2023.101847. URL https://www.sciencedirect.com/science/article/pii/S156625352300163X.

[2] Mohammed Jawad Al-Dujaili and Abbas Ebrahimi-Moghadam. Speech emotion recognition: A comprehensive survey. *Wireless Personal Communications*, 129(4):2525–2561, April 2023. ISSN 1572-834X. doi: 10.1007/s11277-023-10244-3. URL https://doi.org/10.1007/s11277-023-10244-3.

[3] Hoss Belyadi and Alireza Haghighat. Chapter 5 - supervised learning. In Hoss Belyadi and Alireza Haghighat, editors, *Machine Learning Guide for Oil and Gas Using Python*, pages 169–295. Gulf Professional Publishing, 2021. ISBN 978-0-12-821929-4. doi: https://doi.org/10.1016/B978-0-12-821929-4.00004-4. URL https://www.sciencedirect.com/science/article/pii/B9780128219294000044.

[4] A. Tamouridou and altri. Multilayer perceptron, 2018. URL https://www.example.com.

[5] Swapna Mol George and P. Muhamed Ilyas. A review on speech emotion recognition: A survey, recent advances, challenges, and the influence of noise. *Neurocomputing*, 568:127015, 2024. ISSN 0925-2312. doi:

https://doi.org/10.1016/j.neucom.2023.127015. URL https://www.sciencedirect.com/science/article/pii/S0925231223011384.

[6] Qirong Mao, Ming Dong, Zhengwei Huang, and Yongzhao Zhan. Learning salient features for speech emotion recognition using convolutional neural networks. *IEEE Transactions on Multimedia*, 16(8):2203–2213, 2014. doi: 10.1109/TMM.2014.2360798.

[7] Bagus Tris Atmaja, Akira Sasou, and Masato Akagi. Speech emotion and naturalness recognitions with multitask and single-task learnings. *IEEE Access*, 10:72381–72387, 2022. doi: 10.1109/ACCESS.2022.3189481.

[8] Taiba Majid Wani, Teddy Surya Gunawan, Syed Asif Ahmad Qadri, Mira Kartiwi, and Eliathamby Ambikairajah. A comprehensive review of speech emotion recognition systems. *IEEE Access*, 9:47795–47814, 2021. doi: 10.1109/ACCESS.2021.3068045.

[9] Dimitrios Ververidis and Constantine Kotropoulos. Emotional speech recognition: Resources, features, and methods. *Speech Communication*, 48(9):1162–1181, 2006. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2006.04.003. URL https://www.sciencedirect.com/science/article/pii/S0167639306000422.

[10] Jia Rong, Gang Li, and Yi-Ping Phoebe Chen. Acoustic feature selection for automatic emotion recognition from speech. *Information Processing Management*, 45(3):315–328, 2009. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2008.09.003. URL https://www.sciencedirect.com/science/article/pii/S0306457308000885.

[11] Joan C. Borod, editor. *The Neuropsychology of Emotion*. 1993.

[12] David Sander. *Models of Emotion: The Affective Neuroscience Approach*, page 5–54. Cambridge University Press, 2013. URL https://www.cambridge.org/core/books/abs/

cambridge-handbook-of-human-affective-\neuroscience/models-of-emotion/506FDA85ACCDC3617684DCA93724CC3C.

[13] Sippee Bharadwaj and Purnendu Bikash Acharjee. Exploring human voice prosodic features and the interaction between the excitation signal and vocal tract for assamese speech. *International Journal of Speech Technology*, 26(1):77–93, March 2023. doi: 10.1007/s10772-021-09946-5. URL https://doi.org/10.1007/s10772-021-09946-5.

[14] Mehmet Berkehan Akçay and Kaya Oğuz. Speech emotion recognition: Emotional models, databases, features, preprocessing methods, supporting modalities, and classifiers. *Speech Communication*, 116:56–76, 2020. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2019.12.001. URL https://www.sciencedirect.com/science/article/pii/S0167639319302262.

[15] Ayush Shah, Manasi Kattel, Araju Nepal, and D. Shrestha. Chroma feature extraction. In *Proceedings of the International Conference on Signal Processing and Communications(SPCOM)*, 01 2019. URL https://www.researchgate.net/publication/330796993_Chroma_Feature_Extraction.

[16] Muhammad Fahreza Alghifari, Teddy Surya Gunawan, and Mira Kartiwi. Speech emotion recognition using deep feedforward neural network. *Indonesian Journal of Electrical Engineering and Computer Science*, 10 (2):554–561, 2018. ISSN 2502-4760.

[17] Jupyter. URL https://jupyter.org/.

[18] NumPy Developers. Numpy documentation, 2024. URL https://numpy.org/doc/stable/.

[19] pandas development team. About pandas, 2024. URL https://pandas.pydata.org/about/index.html.

[20] Niyazi Ari and Makhamadsulton Ustazhanov. Matplotlib in python. In *2014 11th International Conference on Electronics, Computer and Computation (ICECCO)*, pages 1–6, 2014. doi: 10.1109/ICECCO.2014.6997585.

[21] Zhaobin Wang, Ke Liu, Jian Li, Ying Zhu, and Yaonan Zhang. Various frameworks and libraries of machine learning and deep learning: A survey. *Archives of Computational Methods in Engineering*, 31(1): 1–24, 2024. ISSN 1886-1784. doi: 10.1007/s11831-018-09312-w. URL https://doi.org/10.1007/s11831-018-09312-w.

[22] Analytics Vidhya. Hands-on guide to librosa for handling audio files, 2024. URL https://www.analyticsvidhya.com/blog/2024/01/hands-on-guide-to-librosa-for-handling-audio-files/.

[23] Zhaomeng Zhu, Gongxuan Zhang, Yongping Zhang, Jian Guo, and Naixue Xiong. Briareus: Accelerating python applications with cloud. In *2013 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum*, pages 1449–1456, 2013. doi: 10.1109/IPDPSW.2013.161. URL https://ieeexplore.ieee.org/document/6651038.

[24] Analytics Vidhya. How to save and load machine learning models in python using joblib library, 2023. URL https://www.analyticsvidhya.com/blog/2023/02/how-to-save-and-load-machine-learning\-models-in-python-using-joblib-library/.

[25] Steven R. Livingstone and Frank A. Russo. Ravdess emotional speech audio. https://www.kaggle.com/datasets/uwrfkaggler/ravdess-emotional-speech-audio, 2018.

[26] Samaneh Madanian, Talen Chen, Olayinka Adeleye, John Michael Templeton, Christian Poellabauer, Dave Parry, and Sandra L. Schneider. Speech emotion recognition using machine learning — a systematic review. *Intelligent Systems with Applications*, 20:200266,

2023. ISSN 2667-3053. doi: https://doi.org/10.1016/j.iswa.2023.200266. URL https://www.sciencedirect.com/science/article/pii/S2667305323000911.

[27] Guoqiang Zhang, Michael Y. Hu, B Eddy Patuwo, and Daniel C. Indro. Artificial neural networks in bankruptcy prediction: General framework and cross-validation analysis. *European Journal of Operational Research*, 116(1):16–32, 1999. ISSN 0377-2217. doi: https://doi.org/10.1016/S0377-2217(98)00051-4. URL https://www.sciencedirect.com/science/article/pii/S0377221798000514.

[28] Răzvan Andonie. Hyperparameter optimization in learning systems. *Journal of Membrane Computing*, 1(4):279–291, 2019. doi: 10.1007/s41965-019-00023-0. URL https://doi.org/10.1007/s41965-019-00023-0.

[29] Madan Somvanshi, Pranjali Chavan, Shital Tambade, and S. V. Shinde. A review of machine learning techniques using decision tree and support vector machine. In *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*, pages 1–7, 2016. doi: 10.1109/ICCUBEA.2016.7860040.

[30] Y. Wang, Z. Pan, J. Zheng, L. Qian, and M. Li. A hybrid ensemble method for pulsar candidate classification. *Astrophysics and Space Science*, 364(8):139, Aug 2019. doi: 10.1007/s10509-019-3602-4. URL https://doi.org/10.1007/s10509-019-3602-4.

[31] Elnaz Kabir, Seth Guikema, and Brian Kane. Statistical modeling of tree failures during storms. *Reliability Engineering System Safety*, 177:68–79, 2018. ISSN 0951-8320. doi: https://doi.org/10.1016/j.ress.2018.04.026. URL https://www.sciencedirect.com/science/article/pii/S095183201730707X.

[32] Myung-Jin Jun. A comparison of a gradient boosting decision tree, random forests, and artificial neural networks to model urban land

use changes: the case of the seoul metropolitan area. *International Journal of Geographical Information Science*, 35(11):2149–2167, 2021. doi: 10.1080/13658816.2021.1887490. URL `https://doi.org/10.1080/13658816.2021.1887490`.

[33] Huseyin Ahmetoglu and Resul Das. A comprehensive review on detection of cyber-attacks: Data sets, methods, challenges, and future research directions. *Internet of Things*, 20:100615, 2022. ISSN 2542-6605. doi: https://doi.org/10.1016/j.iot.2022.100615. URL `https://www.sciencedirect.com/science/article/pii/S254266052200097X`.

[34] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[35] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. A review of supervised machine learning algorithms. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1310–1315, 2016.

[36] S. Abinaya and M.K. Kavitha Devi. Chapter 12 - enhancing crop productivity through autoencoder-based disease detection and context-aware remedy recommendation system. In Mohammad Ayoub Khan, Rijwan Khan, and Mohammad Aslam Ansari, editors, *Application of Machine Learning in Agriculture*, pages 239–262. Academic Press, 2022. ISBN 978-0-323-90550-3. doi: https://doi.org/10.1016/B978-0-323-90550-3.00014-X. URL `https://www.sciencedirect.com/science/article/pii/B978032390550300014X`.

[37] Muhammad Aasim, Ramazan Katırcı, Alpaslan Şevket Acar, and Seyid Amjad Ali. A comparative and practical approach using quantum machine learning (qml) and support vector classifier (svc) for light emitting diodes mediated in vitro micropropagation of black mulberry (morus nigra l.). *Industrial Crops and Products*, 213:118397, 2024. ISSN 0926-6690. doi: https://doi.org/10.1016/j.indcrop.2024.

118397. URL https://www.sciencedirect.com/science/article/pii/S0926669024003741.

[38] TechTarget Staff. Data splitting: Definition and how it works, 2023. URL https://www.techtarget.com/searchenterpriseai/definition/data-splitting.

[39] Scikit-learn. f1_score, 2024. URL https://scikit-learn.org/1.5/modules/generated/sklearn.metrics.f1_score.html.

[40] IBM. Understanding model accuracy, 2023. URL https://www.ibm.com/docs/en/datacap/9.1.9?topic=project-understanding-model-accuracy.

[41] Pablo Diez. Chapter 1 - introduction. In Pablo Diez, editor, *Smart Wheelchairs and Brain-Computer Interfaces*, pages 1–21. Academic Press, 2018. ISBN 978-0-12-812892-3. doi: https://doi.org/10.1016/B978-0-12-812892-3.00001-7. URL https://www.sciencedirect.com/science/article/pii/B9780128128923000017.

[42] S. Sharanyaa, Tini J Mercy, and Samyukthaa V.G. Emotion recognition using speech processing. In *2023 3rd International Conference on Intelligent Technologies (CONIT)*, pages 1–5, 2023. doi: 10.1109/CONIT59222.2023.10205935.

[43] Yuto. Wave analytics method. what is spectral contrast? *Zenn*, 2024. doi: 10.1109/ACCESS.2022.3189481. URL https://zenn.dev/yuto_mo/articles/7413ca2ed4eb5f.

[44] Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. Music type classification by spectral contrast feature. In *Proceedings. IEEE International Conference on Multimedia and Expo*, volume 1, pages 113–116 vol.1, 2002. doi: 10.1109/ICME.2002.1035731.

[45] URL https://huggingface.co/learn/audio-course/chapter1/audio_data.

[46] Juan Pablo Bello. Chroma and tonality. *EL9173 Selected Topics in Signal Processing: Audio Content Analysis*. URL `https://s18798.pcdn.co/jpbello/wp-content/uploads/sites/1691/2018/01/6-tonality.pdf`.

[47] Leena Mary and B. Yegnanarayana. Extraction and representation of prosodic features for language and speaker recognition. *Speech Communication*, 50(10):782–796, 2008. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2008.04.010. URL `https://www.sciencedirect.com/science/article/pii/S0167639308000587`.

[48] Saikat Basu, Jaybrata Chakraborty, Arnab Bag, and Md. Aftabuddin. A review on emotion recognition using speech. In *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 109–114, 2017. doi: 10.1109/ICICCT.2017.7975169.

[49] Arundhati Navada, Aamir Nizam Ansari, Siddharth Patil, and Balwant A. Sonkamble. Overview of use of decision tree algorithms in machine learning. In *2011 IEEE Control and System Graduate Research Colloquium*, pages 37–42, 2011. doi: 10.1109/ICSGRC.2011.5991826.

Ringraziamenti

Desidero ringraziare la mia famiglia e chi mi è stato sempre vicino, per il costante supporto e per l'incoraggiamento che mi ha permesso di affrontare con determinazione questo percorso. Grazie per la vostra presenza e per aver sempre creduto nelle mie capacità.