Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria del Cinema

e dei Mezzi di Comunicazione

Classificazione dell'efficacia della comunicazione paraverbale tramite parametri acustici del segnale vocale

Relatore:

Prof. Alessio Carullo

Corelatore:

Prof.ssa Arianna Astolfi

Candidato:

Giacomo Eandi

Anno Accademico 2018/2019

Col tono giusto si può dire tutto, col tono sbagliato nulla:

l'unica difficoltà consiste nel trovare il tono.

(George Bernard Shaw).

Abstract

L'attività di tesi si colloca nell'ambito di una ricerca mirata alla valutazione oggettiva dell'efficacia della comunicazione paraverbale in ambito relazionale, così da creare uno strumento adeguato attraverso cui si possa valutare, ad esempio, una performance attoriale.

Lo studio mira a classificare le registrazioni, effettuate da soggetti professionisti (attori), all'interno di classi definite come *forme della voce*. Le forme della voce sono un nuovo paradigma di analisi degli stati emozionali, sono cinque e si distinguono in Piatta, Tonda, Quadra, Triangolare e Spigolosa.

Grazie alla collaborazione di 13 attori (7 uomini e 6 donne), in precedenti lavori, è stato possibile ottenere 65 registrazioni le quali sono state validate da esperti di comunicazione paraverbale che si sono occupati di collocarle all'interno della corretta classe di appartenenza. Parallelamente, sono stati individuati ed estratti, dalle registrazioni, 20 parametri che permettono di identificare le variazioni nel segnale vocale.

Da questa base è cominciata la ricerca di cui si parla in questa tesi. È stata, in primis, svolta un'analisi riguardante la correlazione sia tra i parametri estratti e la forma della voce, sia tra i parametri stessi al fine di individuare quelli migliori per la classificazione e rimuovere quelli ridondanti. Avvalendosi dei parametri a più elevata correlazione significativa ($P-value < 0.05$) con la forma della voce, si è cominciato a verificare l'accuratezza restituita dai vari modelli di classificazione utilizzando l'applicazione Classification Learner di Matlab ™. Il lavoro di analisi dell'accuratezza è stato svolto parallelamente su 3 set di dati: il primo composto

da tutti gli attori, il secondo dai soli uomini e il terzo dalle sole donne. La divisione è stata fatta per identificare quali fossero i parametri necessari alla classificazione a seconda del sesso dell'attore partecipante.

Condurre l'analisi sulle tre differenti tipologie di dati ha permesso, inoltre, di individuare i migliori modelli di classificazione a seconda della tipologia di dati utilizzati.

Successivamente all'analisi delle accuratezze, si è aperta una fase di studio sulle complessità computazionali degli algoritmi utilizzati per estrarre i parametri per la classificazione. Unendo i risultati ottenuti con le complessità trovate in letteratura per i modelli di classificazione, si è potuto arrivare ad una stima del costo dell'intero processo, dall'estrazione dei parametri al loro utilizzo per la classificazione.

Complessivamente, per tutti i gruppi di dati trattati, è stata ottenuta una buona precisione di classificazione con percentuali al di sopra del 70% per i principali modelli di cui si è eseguito l'addestramento.

Tuttavia, resta da tenere in considerazione il fatto che in tutta l'analisi, a causa della carenza di dati, non è stato possibile introdurre un metodo di validazione dei modelli. Le accuratezze ottenute sono, perciò, da intendersi come un semplice riferimento di quelle che possono essere le prestazioni di un algoritmo rispetto a tutti gli altri e non come un indicatore preciso di quella che può essere la sua accuratezza in assoluto per i dati trattati.

L'introduzione di un metodo di validazione, conseguente all'estensione del numero di rilevazioni utilizzate, sarà sicuramente il punto di partenza dei futuri lavori riguardanti la classificazione basata sulle forme della voce.

Indice

CAPITOLO 1

Introduzione

Mai come in questi ultimi anni il comunicare ha assunto un ruolo di primo piano nella vita di tutti i giorni e di tutti gli esseri umani. Ognuno in modo diverso, ma comunque qualsiasi persona sulla Terra, comunica con gli altri creando un flusso imponente ed ininterrotto di informazioni trasmesse da un emittente ad un ricevente più o meno consapevole. Un esempio eclatante di questo incessante comunicare è il fatto che ogni minuto su una piattaforma come YouTube vengano caricate 500 ore di video e ogni giorno viene guardato oltre 1 miliardo di ore di contenuti audiovisivi. Numeri impressionanti che danno l'idea di come comunicare sia diventato facile e di come le persone abbiano voglia di farlo in modo sempre più immediato e facile. Quello di YouTube è solo un esempio di come la comunicazione si sia trasformata nel corso degli anni passando da una comunicazione basata maggiormente sulle parole scritte o stampate, e quindi di tipo più tattile, ad una più basata sui contenuti multimediali di vario tipo e, quindi, in grado di coinvolgere sensi come l'udito e la vista.

Ma quante di queste informazioni sono utili? Quante sono quelle trasmesse in modo efficace? E soprattutto, quante sono quelle che verranno effettivamente ricordate perché si distinguono dal resto del "brusio"?

Ecco, quindi, che in questo panorama diventa fondamentale riuscire a trasmettere i propri messaggi in modo coerente, chiaro, funzionale e, soprattutto, efficace. Se si riesce a raggiungere questa padronanza nella comunicazione si entra in possesso di un'arma di una potenza inaudita, specialmente se si crea un rapporto di empatia tra mittente e ricevente.

Come si può facilmente immaginare, però, padroneggiare la comunicazione non è assolutamente qualcosa alla portata di tutti data la complessità della materia trattata e la sua composizione articolata. La maggior parte di chi comunica (e quindi di qualsiasi essere vivente, come sostiene Watzlawick[1]) si concentra molto più sul piano verbale trascurando quello non verbale e quello paraverbale. Un errore imperdonabile che porta inevitabilmente a scontrarsi con fallimenti comunicativi anche importanti in determinate situazioni dove è fondamentale il modo in cui si comunica con chi ci sta di fronte: si pensi ad esempio ad un oratore di una conferenza TED dove si ha un tempo limitato per trasmettere dei concetti che potrebbero cambiare la vita degli spettatori. Verso la fine degli anni '60 Albert Mehrabian, psicologo statunitense, cercò di dimostrare la portata di questo errore. L'americano condusse delle ricerche riguardanti l'importanza di diversi aspetti della comunicazione arrivando ad un risultato piuttosto sconvolgente nonostante la ap-

parente banalità della scoperta. Scoprì, infatti, che quando si comunica, in particolare quando si cerca di trasmettere sentimenti ed emozioni, le parole (il piano verbale) contano solo per il 7%, la comunicazione paraverbale influisce per il 38% mentre quella non verbale influisce per il restante 55%.

Questo risultato si può considerare come uno dei cardini della ricerca di cui si tratta nel presente elaborato. Non ci si è occupati di comunicazione non verbale, è stato analizzato invece tutto quello che riguarda il piano paraverbale da un punto di vista puramente oggettivo per stabilire una valutazione scientifica dell'efficacia di questo tipo di comunicazione ed una correlazione tra la valutazione effettuata da esperti e quanto visto attraverso i parametri individuati. Lo studio della comunicazione paraverbale viene effettuato attraverso l'uso delle forme della voce, un nuovo modello di valutazione dello stato d'animo, proposto dall'Accademia Interago di Roma. In precedenti studi sono stati individuati dei parametri per lo categorizzazione dei file audio prodotti dalle persone coinvolte nella ricerca. Sono stati coinvolti 13 attori che hanno prestato la loro voce per interpretare le diverse forme vocali. Questi dati, validati da esperti di comunicazione paraverbale, sono stati utilizzati per individuare i modelli di Machine Learning a maggior accuratezza per la classificazione dei campioni audio. Individuati i migliori modelli si è cercato di tracciare un andamento delle loro prestazioni a seconda dei parametri utilizzati così da individuare differenti scenari di applicazione.

1.1 Le forme della voce

Grazie agli studi effettuati dall'Accademia Interago di Roma sono state individuate cinque forme della voce, un concetto nuovo nel campo dello studio emozionale che permette di oltrepassare l'idea di legame tra una singola emozione ed un determinato tono vocale. Le forme della voce possono essere viste come cinque classi emozionali le quali comprendono, ognuna, analoghi stati emotivi al loro interno. Come detto, non ci si avvale più della classica distinzione in emozioni espresse dal parlante ma si va a considerare la voce in modo molto diverso: diventa il tramite per esprimere uno stato emotivo più complesso, caratterizzato da più ampie sfumature e sottocategorie; non è più il prodotto di un' emozione ma è il mezzo con cui si trasmette una sensazione.

Ecco quindi che possiamo identificare le cinque forme vocali etichettandole come segue[2]

- **Piatta**: Monotona e costante. Caratterizzata da scarsi cambi in intensità, frequenza e ritmo. Il risultato è un tono noioso e incapace di catturare l'attenzione di chi ascolta.
- **Tonda**: Tono colloquiale ed empatico. Usata specialmente per trasferire informazioni o sensazioni personali.
- **Quadra**: Usata da chi vuole comunicare un alto grado di professionalità ed autorevolezza. È il caso, ad esempio, degli insegnanti o di chi fa i voice-over nei documentari.
- **Triangolare**: Evidenzia un alto grado di coinvolgimento nella comunicazione. L'obiettivo principale di chi usa questa forma vocale è quello di trasmettere energia, interesse e positività.
- **Spigolosa**: Usata per accusare o trasmettere aggressività nei confronti di chi ascolta.

Le cinque forme vocali possono essere ulteriormente suddivise in due macrocategorie: funzionali e disfunzionali. All'interno della classe delle forme funzionali ricadono le forme tonda, triangolare e quadra quelle cioè che sono efficaci da un punto di vista relazionale. Nella classe delle forme disfunzionali sono collocate le forme piatta e spigolosa, le quali sono considerate come poco efficaci da un punto di vista relazionale. La classificazione della voce in forme si rivela efficace in quanto è in grado di coprire interamente lo spettro emotivo umano con solo 5 differenti classi senza dover utilizzare una classe per ogni emozione, operazione che sarebbe, palesemente, impossibile da effettuare.

1.2 Obiettivo degli studi

L'attività di tesi si colloca nell'ambito di una ricerca mirata alla valutazione oggettiva dell'efficacia della comunicazione paraverbale in ambito relazionale, così da creare uno strumento adeguato attraverso cui si possa valutare, ad esempio, una performance attoriale.
L'obiettivo principale che ci si pone è quello di trovare, quindi, uno o più algoritmi di Machine Learning in grado di predire, con un'accuratezza sufficientemente elevata, la forma della voce a cui un determinato campione audio appartiene; nel fare questa classificazione non si vuole, però,

sacrificare la velocità di calcolo in fase sia di estrazione delle features (parametri) che in fase di classificazione dei campioni inseriti come input.
Questo bisogno di avere alte prestazioni con una bassa richiesta computazionale deriva dalla volontà di implementare successivamente questi algoritmi, in primis, all'interno di una webapp e, parallelamente, all'interno di un dispositivo stand-alone, il quale non per forza sarà dotato di un processore ad alte prestazioni.
Per raggiungere tale scopo si parte dalla ricerca in letteratura di quello che è lo stato dell'arte della tecnologia utilizzata applicando poi tali conoscenze al problema in esame. Analizzate le possibilità offerte dagli algoritmi si cerca di ridurre al minimo il costo in termini di risorse del processo di classificazione agendo sulla scelta dei parametri utilizzati e sull'ottimizzazione degli algoritmi.
Ci si è posti questo obiettivo in quanto la necessità pratica è quella di arrivare a fornire agli attori (o a chiunque voglia utilizzare questa risorsa) uno strumento in grado di stimare con una buona percentuale di accuratezza la loro performance a livello emotivo indirizzandoli verso una corretta interpretazione del ruolo a loro assegnato.
Per avere un buon dataset di partenza è stato fondamentale il supporto di Interago Academy di Roma, la quale ha offerto supporto umano e intellettuale alla ricerca fornendo il talento e l'esperienza di attori professionisti. Con la loro collaborazione è stato possibile avere un dataset piuttosto bilanciato per procedere con il training dei vari algoritmi esistenti.

1.3 Contesto applicativo

Come detto in precedenza, la ricerca è partita grazie al supporto di Interago e alla volontà di sviluppare un software di supporto per chi voglia cimentarsi con la recitazione ed avere un feedback indicativo della qualità della sua performance. Ovviamente il software non è indicato solo per chi lavora nel mondo del cinema o del teatro; potrebbe, per esempio, tornare utile a chi, di professione, comunica con altri (professori, speakers, conferenzieri, venditori, etc) o chi semplicemente voglia avere un riscontro sulla propria efficacia comunicativa e relazionale. Per questo motivo si vuole implementare il modello di classificazione all'interno di una webapp mettendola, quindi, a disposizione di tutti quelli che navighino il sito su cui risiede. Parallelamente a questa implementazione web, si vuole installarla all'interno di dispositivi più compatti dotati di microfono, processore e memoria, in grado di elaborare i dati per estrarre le features desiderate e dare un feedback uti-

lizzando il modello di classificazione dei dati. Per poter inserire il software su tale hardware è necessario avere un algoritmo performante che richieda il minor numero possibile di operazioni e il minor quantitativo di memoria mantenendo la capacità di processare correttamente i dati che arrivano in input al sistema.

CAPITOLO 2

Lo stato dell'arte

Per lo studio dei modelli e delle accuratezze da questi restituite ci si è avvalsi della letteratura già esistente in ambito di studio delle emozioni e in ambito Machine Learning.

2.1 Lo studio delle emozioni

Fin dalla nascita della psicologia lo studio delle emozioni ha sempre destato interesse tra i ricercatori impegnati nel trovare una correlazione tra lo stato emotivo di un paziente e il suo stato fisiologico. Numerosi studiosi si sono concentrati sul ruolo che la voce ha nel trasmettere emozioni identificando in essa un mezzo affidabile per la comunicazione degli stati emozionali.[3]. Se si guarda solo al piano verbale della comunicazione quest'assunzione può non sembrare così scontata ma, se si estende l'osservazione anche al piano non verbale e a quello paraverbale, ecco che diventa chiaro che uno strumento come la voce diventi importante per trasmettere emozioni attraverso la modulazione della stessa. Si è quindi spostato il focus sull'analisi del segnale vocale al fine di identificare dei parametri che potessero essere correlati con le emozioni trasmesse dal parlante. I parametri base che, solitamente, venivano usati sono intonazione, frequenza, durata e potenza [4, 3] ovvero i, cosiddetti, *parametri prosodici*. Tuttavia, con il proseguire degli studi, si è notato che questi indicatori da soli non erano più sufficienti per la corretta distinzione delle emozioni in quanto presentavano somiglianze troppo marcate per classi diverse (impedivano di avere una distinzione netta) e sovente si presentava il bisogno di conoscere i valori normali dei parametri per analizzarne la variazione.[5]
Sono stati, quindi, introdotti dei nuovi parametri legati allo spettro del segnale vocale ottenendo così una migliore separazione tra le diverse classi di emozioni[3] riuscendo così, anche, a slegarsi dalla necessità di analizzare la variazione dei valori "base". Questi nuovi parametri verranno spiegati in dettaglio nella sezione successiva.
Un utilizzo analogo al modo in cui le *features* vengono utilizzate in questa ricerca è stato fatto in [6] dove si è cercato di mettere in relazione la percezione qualitativa delle voci di alcuni speaker radiofonici con dei parametri oggettivi del segnale vocale. In questo studio si è notata una correlazione piuttosto marcata tra quella che veniva percepita come una voce "buona" per la radio e determinati valori nei parametri indicando, quindi, una possibile oggettivazione del concetto soggettivo di *piacevolezza* di una voce. Lo studio fatto in [6] è di grande interesse nel caso di cui si discute in questa tesi in quanto pone l'accento esclusivamente sulla comunicazione paraverbale,

data la natura non visiva del mezzo radiofonico, e per il fatto che i parametri utilizzati per l'analisi sono sostanzialmente gli stessi introdotti per le forme della voce (CPPS, F0, tempo di pausa e di fonazione).

2.2 I parametri utilizzati

Per una corretta analisi del segnale vocale non è sufficiente analizzare gli *indicatori prosodici* [3]. Serve un più ampio numero di parametri, diversi per qualità e tipologia, che siano in grado di andare ad aggiungere informazioni utili alla ricerca e producano risultati utilizzabili da un algoritmo.

In questa ricerca i parametri utilizzati sono 20, divisi per provenienza

- Frequenza
  - Media della frequenza fondamentale ($F0_{mean}$)
  - Deviazione standard della frequenza fondamentale ($F0_{std}$)
- Intensità (RMS)
  - Media dell'intensità
  - Deviazione standard dell'intensità
- Fonazione
  - Media del tempo di fonazione ($voiced_{mean}$)
  - Moda del tempo di fonazione ($voiced_{mode}$)
  - Deviazione standard del tempo di fonazione ($voiced_{std}$)
  - Media del tempo di silenzio ($unvoiced_{mean}$)
  - Moda del tempo di fonazione ($unvoiced_{mode}$)
  - Deviazione standard del tempo di silenzio ($unvoiced_{std}$)
  - Tempo di fonazione in percentuale (Dt %)
- CPPS

- Media del CPPS ($CPPS_{mean}$)
- Mediana del CPPS ($CPPS_{median}$)
- Moda del CPPS ($CPPS_{mode}$)
- Deviazione standard del CPPS ($CPPS_{std}$)
- Range del CPPS ($CPPS_{range}$)
- Quinto percentile del CPPS ($CPPS_{5prc}$)
- Novantacinquesimo percentile del CPPS ($CPPS_{95prc}$)
- Skewness del CPPS ($CPPS_{skew}$)
- Kurtosis del CPPS ($CPPS_{kurt}$)

Seguendo la procedura indicata in [2] il primo step dell'estrazione dei parametri è fare il *down-sampling* del segnale vocale a 22050 Hz (ovverò la metà del sample rate di registrazione, 44100 Hz); fatto questo i samples sono raggruppati in frames da 1024 punti (che corrispondono a circa 46 ms) su cui si calcolano RMS, HNR e F0 (pitch). Questi tre dati servono per classificare un frame come *voiced* o *unvoiced*.
Un frame è voiced se viene soddisfatta la seguente condizione[2]

$$\left[RMS_{frame} > \frac{RMS_{aver}}{2}\right] \text{AND} \left[HNR_{frame} > 0\text{dB}\right] \text{AND} \left[\frac{|FO_{frame} - F0_{frame-1}|}{F0_{frame-1}} < 0.5\right] \tag{2.1}$$

dove RMS_{aver} è la media su tutta la registrazione del RMS calcolato per ogni frame.
Per il calcolo del tempo di fonazione in percentuale si utilizza, invece

$$\text{Dt\%} = 100 \cdot \frac{n_{voiced}}{n_{voiced} + n_{unvoiced}} \tag{2.2}$$

dove n rappresenta il numero di frame classificati come voiced o unvoiced.

Tra i parametri sopra elencati una particolare importanza la riveste il CPPS, sigla che sta per *Cepstral Peak Prominence Smoothed* ed è la vesione "smoothed" del *CPP*, un indicatore introdotto in [7] definito come *misura acustica della qualità della voce e una delle più robuste e promettenti misurazioni acustiche della gravità della disfonia* [8]. Grazie al CPP i frame vocali sono analizzati nel dominio del *cepstrum* (in quefrenza) che, in modo più formale, è definito come segue da Bogert et al. in [9].

Si supponga di avere un segnale con eco rappresentato da

$$x(t) = s(t) + \alpha s(t - \tau) \tag{2.3}$$

lo spettro di tale segnale è definito come

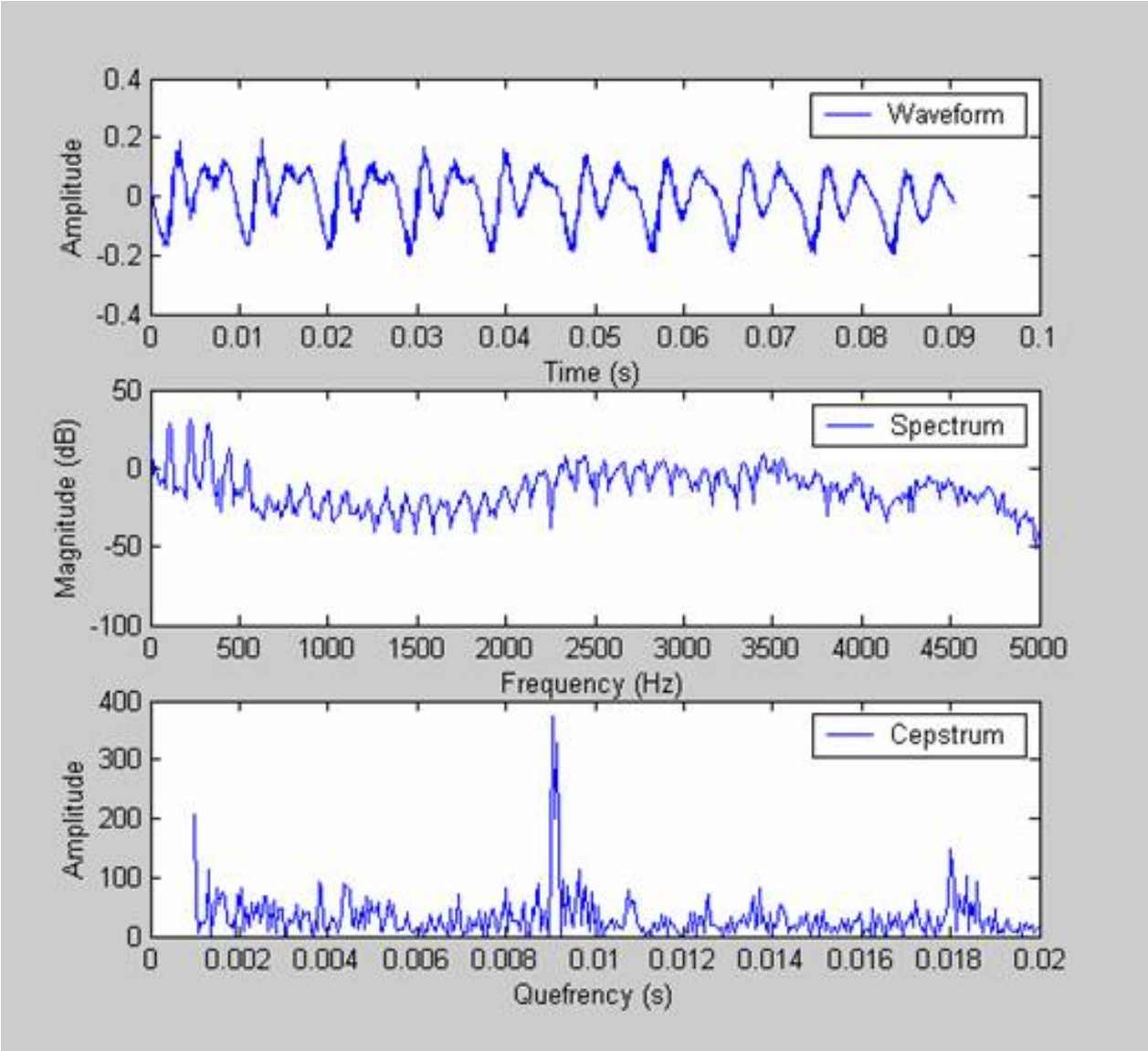
$$|X(f)|^2 = |S(f)|^2[1 + \alpha^2 + 2\alpha\cos(2\pi f\tau)] \tag{2.4}$$

da cui si vede che lo spettro è formato da un inviluppo modulato da una funzione periodica. Facendo il logaritmo di 2.4 si può convertire il prodotto in una somma arrivando alla forma

$$|C(f)|^2 = \log|X(f)|^2 = \log|S(f)|^2 + \log[1 + \alpha^2 + 2\alpha\cos(2\pi f\tau)] \tag{2.5}$$

di cui in Figura 2.1 si può vedere un esempio grafico

Figura 2.1: Esempio di Cepstrum[10]



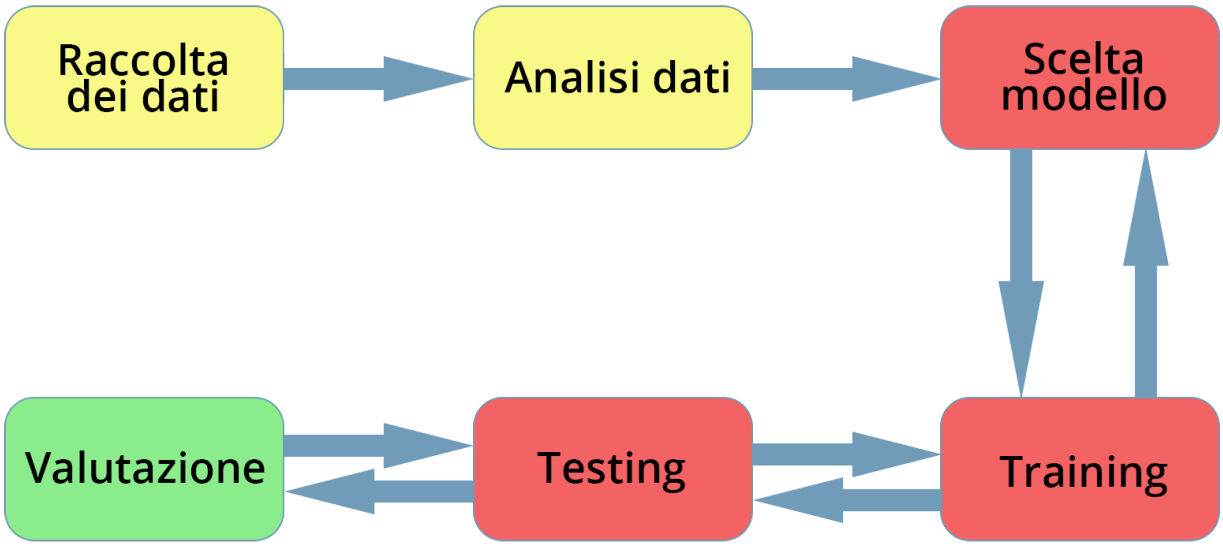
Ottenuta la distribuzione del CPPS sono stati calcolati, successivamente, sulla stessa tutti i parametri sopra elencati (media, moda, mediana, deviazione standard, etc)

2.3 Machine Learning e classificazione

Il Machine Learning (ML) è un'evoluzione degli studi fatti durante lo sviluppo delle teorie sul computational learning e sul pattern recognition[11]. Con ML si ha uno strumento efficace per le analisi dei dati potendo predire delle tendenze nei dati stessi. L'obiettivo del Machine Learning è

quello di creare algoritmi che siano in grado di apprendere come classificare nuovi dati sulla base di dati precedentemente avuti in ingresso e di correggere, se necessario, tali predizioni attraverso meccanismi di feedback. Vengono quindi sviluppati degli algoritmi in grado di tener conto di queste variazioni nei dati. In figura 2.2 è riportato un semplice flowchart del procedimento standard per l'implementazione del Machine Learning. Dopo la prima fase iniziale di setup ed analisi dei dati si passa a reiterare il processo di training e testing/validazione del modello fino a raggiungere dei risultati in linea con quanto ci si è prefissati in partenza.

Figura 2.2: Machine Learning Flowchart



Esistono due categorie principali di algoritmi di ML, le quali si distinguono per la tipologia di dati in ingresso.[12]

- **Supervised Machine Learning**: per i dati utilizzati dal modello le features sono già estratte; c'è stato quindi un precedente trattamento dei dati.
- **Unsupervised Machine Learning**: i dati non sono ancora ne etichettati ne categorizzati. Il modello agisce senza training precedente e tenta di cercare pattern ed estrae features opportune

Gli algoritmi di Machine Learning si possono, altresì, dividere sulla base dell'obiettivo che perseguono; abbiamo quindi tre tipologie di algoritmi

- **Classificazione**: i dati in ingresso sono assegnati a classi distinte (output discreto).

- **Regressione**: i dati in ingresso vanno assegnati a un output numerico continuo.
- **Clustering**: i dati vanno divisi in gruppi ma, a differenza della classificazione, le classi non sono note in partenza.

In questa tesi vengono usati algoritmi di *classificazione* e un Machine Learning di tipo *supervised*. I predittori (features) per i campioni audio sono estratti ”al di fuori” dell'algoritmo di Machine Learning e ogni campione deve essere assegnato ad una delle 5 classi (forme della voce). Per una discussione più approfondita di quali siano i principali algoritmi di Machine Learning si rimanda all'Appendice 2.

CAPITOLO 3

Metodi e procedimenti

In questo capitolo verranno esposti quali sono i metodi e i procedimenti utilizzati al fine di raggiungere l'obiettivo di cui si parla nel Capitolo 1, ovvero l'analizzare e classificare le registrazioni usando il paradigma della forma della voce al fine di valutarne l'efficacia in termini di comunicazione paraverbale.
Il procedimento di scelta, scrittura dell'algoritmo ed estrazione delle features per ogni campione vocale in nostro possesso, è omesso in quanto esula dagli scopi di questa tesi e, soprattutto, perché è stato effettuato a conclusione di precedenti studi di cui si parla nell'apposita Appendice B. Il lavoro effettuato in quel contesto viene considerato come punto di partenza per la ricerca che si svolge nell'ambito di questa tesi.

Il dataset principale che è stato utilizzato è stato creato grazie al supporto di 13 attori professionisti i quali, secondo precise indicazioni fornite loro, hanno interpretato, ognuno, le 5 forme vocali descritte nella sezione 1.1. Si era in possesso, quindi, di 65 file audio. Ognuna di queste registrazioni è stata validata da un team di esperti in comunicazione paraverbale i quali, dopo averle analizzate, le hanno assegnate alle cinque forme vocali di appartenenza. Questo è stato un passaggio chiave in quanto ha permesso di avere un dataset in cui ogni campione era già classificato e poteva, quindi, far parte del set per l'addestramento. Gli esperti hanno confermato che ognuno degli attori aveva interpretato correttamente le forme della voce richieste. Da qui si è potuto cominciare con le considerazioni che hanno portato ai risultati di questa tesi.
Gli attori erano 7 maschi e 6 femmine, numeri che permettono di considerare il set come bilanciato per quanto riguarda il sesso. Per questo motivo è stato deciso di lavorare su 3 fronti in modo parallelo: in primis è stato considerato il dataset complessivo (uomini e donne, 13 attori, 65 file audio), poi si è andati a dividere il set sulla base del genere ottenendo, così, due set differenti. Il primo con 7 attori e 35 file audio e il secondo con 6 attrici e 30 file audio.

La ricerca del miglior modello è stata portata avanti utilizzando l'applicazione *Classification Learner* di Matlab™, un tool che fornisce tutti i principali modelli di Machine Learning per la classificazione di dati, consente di farne il training e, successivamente, permette di esportare il modello per riutilizzarlo esternamente all'applicazione. All'interno dell'applicazione sono presenti, inoltre, degli strumenti per l'analisi dei dati e della bontà della classificazione.
Il workflow seguito per i tre dataset sopra citati è il medesimo: per prima cosa si analizza la

correlazione tra le features e le forme vocali, sulla base di quest'analisi si va a ridurre il numero di predittori utilizzati cercando di lasciare solo quelli strettamente necessari alla classificazione. Trovato il numero minimo di features quel che si è fatto è testare differenti algoritmi al fine di trovare quello in grado di dare la miglior accuratezza.

Finita questa prima riduzione basata sull'analisi della correlazione tra forma della voce e predittori quel che si è sperimentato è la riduzione delle features basandosi sulla loro correlazione reciproca: ci si è avvalsi, nuovamente, della tabella di correlazione andando, ad ogni iterazione, ad eleggere la coppia di features a *maggior correlazione reciproca* ed eliminando quella a *minor correlazione con la forma vocale.*

Parte I

Analisi delle accuratezze

3.1 Dataset complessivo

3.1.1 Analisi dei dati e scelta delle features

All'interno del dataset sono contenuti tutti i 13 attori che hanno prestato la loro voce: 7 uomini e 6 donne.

Utilizzando la funzione *corr* di Matlab™è stato possibile trovare le matrici contenenti i *coefficienti di correlazione* e i *p-value* per i parametri estratti. Di default Matlab™utilizza i *coefficienti di correlazione lineare di Pearson* per il calcolo della correlazione, per il caso in esame va bene utilizzare tali coefficienti e quindi non sono state fatte modifiche ai parametri della funzione. Come output vengono restituite due matrici: la prima contenente i valori dei coefficienti di correlazione per ogni coppia di parametri, la seconda contiene i p-value per ogni coppia di parametri.

In Tabella 3.1 sono riportati i valori ottenuti per la correlazione tra le features e la forma della voce. Nella prima colonna sono contenuti i coefficienti, nella seconda i p-value. Le righe colorate di verde contengono i predittori che hanno la maggior correlazione significativa ($|Correlazione| > 0.25$ e $p - value < 0.05$) con la forma della voce. Il valore della correlazione scelto come "buona correlazione" non è così elevato come sarebbe opportuno supporre; questa scelta è stata fatta per compensare il fatto che non si hanno dataset numerosi e per avere una panoramica migliore del comportamento degli algoritmi al diminuire del numero di predittori utilizzati. Rimane invariato invece il valore limite del *p-value* ($(p - value < 0.05)$.

Tabella 3.1: Coefficienti di correlazione (secondo Pearson) tra forma della voce e features

Parametro	Correlazione	p-value
Gender	0.000	1.000
$CPPS_{mean}$	0.0541	0.6685
$CPPS_{median}$	0.05535	0.6615
$CPPS_{mode}$	0.17671	0.1591
$CPPS_{std}$	-0.2953	0.0169
$CPPS_{range}$	-0.0271	0.8305
$CPPS_{5prc}$	0.26162	0.03523
$CPPS_{95prc}$	-0.0863	0.4944
$CPPS_{skew}$	-0.1181	0.3490
$CPPS_{kurt}$	-0.051	0.6871

Parametro	Correlazione	p-value
$voiced_{mean}$	-0.6018	$1.1415e-07$
$voiced_{mode}$	-0.1673	0.1829
$voiced_{std}$	-0.5706	$6.98e-07$
$unvoiced_{mean}$	-0.5664	$8.73e-07$
$unvoiced_{mode}$	0.0856	0.4977
$unvoiced_{std}$	-0.65	$4.65e-09$
$F0_{mean}$	0.6038	$1.01e-07$
$F0_{std}$	0.44574	0.0002
RMS_{mean}	-0.0691	0.5846
RMS_{std}	0.0548	0.6646
Dt%	0.2848	0.02147

Osservando i coefficienti di correlazione e i relativi p-value si nota che i predittori maggiormente correlati significativamente con la forma della voce sono 9: $CPPS_{std}$, $CPPS_{5prc}$, $voiced_{mean}$, $voiced_{std}$, $unvoiced_{mean}$, $unvoiced_{std}$, $F0_{mean}$, $F0_{std}$, Dt%.

3.1.2 Training iniziale degli algoritmi

Individuato il numero massimo di features migliori a livello teorico si è poi proceduto ad utilizzare tali predittori per il training di tutti gli algoritmi messi a disposizione dall'applicazione *Classification Learner* così da avere un punto di partenza per indirizzare il successivo lavoro.
In Tabella 3.2 sono riportati i risultati del training sopra descritto. Sono evidenziati, con il colore verde, gli algoritmi che hanno restituito un accuratezza migliore rispetto a tutti gli altri. Durante questa fase sono stati scartati tutti quei modelli con accuratezza del 100% in quanto non è stato

introdotto un metodo di validazione del modello e tali risultati non si possono considerare, quindi, attendibili. Gli algoritmi qui evidenziati sono quelli su cui ci si è concentrati maggiormente nelle fasi seguenti di questa ricerca.

Tabella 3.2: Modelli addestrati con 9 feature e relativa accuratezza

Modello	Accuratezza
Fine Tree	76.9 %
Medium Tree	76.9 %
Coarse Tree	60.0 %
Linear Discriminant	61.5 %
Quadratic Discriminant	90.8 %
Linear SVM	56.9 %
Quadratic SVM	84.6 %
Cubic SVM	98.5 %
Fine Gaussian SVM	100 %
Medium Gaussian SVM	75.4 %
Coarse Gaussian SVM	50.8 %

Modello	Accuratezza
Fine KNN	100 %
Medium KNN	44.6 %
Coarse KNN	20.0 %
Cosine KNN	38.5 %
Cubic KNN	43.1 %
Weighted KNN	100 %
Boosted Trees	100 %
Bagged Trees	100 %
Subspace Discriminant	60.0 %
Subspace KNN	100 %
RUSBoosted Trees	92.3 %

Uno dei principali motivi per cui, senza l'utilizzo di un metodo di validazione si ha un'accuratezza del 100% è l'*overfitting*. L'overfitting si ha quando un modello è troppo basato sui dati in esame e li rappresenta, quindi, in modo eccessivamente accurato. Avere questo tipo di problema conduce all'impossibilità di scalare il modello su dati diversi da quelli utilizzati per l'addestramento, scenario che, soprattutto nel caso di cui si parla in questo elaborato, non è auspicabile. Opposto all'overfitting si può avere il problema dell'*underfitting* che rappresenta la situazione diametralmente opposta, quella, cioè, dove non si utilizzano sufficienti parametri e si hanno quindi basse performance di predizione.

Fatta questa prima scrematura si è poi andati ad effettuare il procedimento di progressiva eli-

minazione di cui si parla nell'introduzione di questo capitolo andando, così, ad individuare un andamento progressivo dell'accuratezza restituita dagli algoritmi.

3.1.3 Correlazione tra features e forma della voce

Si è poi proceduto, per il dataset completo, ad analizzare il comportamento dei principali algoritmi al diminuire del numero di features. La riduzione è stata effettuata basandosi sulla correlazione tra predittori e forma della voce. Ad ogni iterazione si è eliminata la feature a minor correlazione con la forma vocale. La prima colonna rappresenta il test fatto utilizzando tutte le features disponibili per verificare il grado di variazione in accuratezza rispetto all'utilizzare solo quelle ad elevata correlazione. Si riporta un elenco delle features utilizzate ad ogni training dei modelli.

- **20 features**: utilizzate tutte
- **9 features**: utilizzate $Cpps_{std}$, $Cpps_5$, $Voiced_{mean}$, $Voiced_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$, $f0_{std}$, $dt(\%)$
- **8 features**: utilizzate $Cpps_5$, $Voiced_{mean}$, $Voiced_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$, $f0_{std}$, $dt(\%)$
- **7 features**: utilizzate $Voiced_{mean}$, $Voiced_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$, $f0_{std}$, $dt(\%)$
- **6 features**: utilizzate $Voiced_{mean}$, $Voiced_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$, $f0_{std}$
- **5 features**: utilizzate $Voiced_{mean}$, $Voiced_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$
- **4 features**: utilizzate $Voiced_{mean}$, $Voiced_{std}$, Unv_{std}, $f0_{mean}$
- **3 features**: utilizzate $Voiced_{mean}$, Unv_{std}, $f0_{mean}$
- **2 features**: utilizzate Unv_{std}, $f0_{mean}$
- **1 features**: utilizzata Unv_{std}

In Tabella 3.3 si riporta il comportamento dei principali algoritmi per quanto riguarda l'accuratezza restituita al diminuire del numero di features utilizzate. Le colonne rappresentano il numero di predittori utilizzati mentre le righe indicano i modelli di cui è stato eseguito il training.

Tabella 3.3: Modelli addestrati al decrescere del numero di features - Correlazione tra features e forma della voce

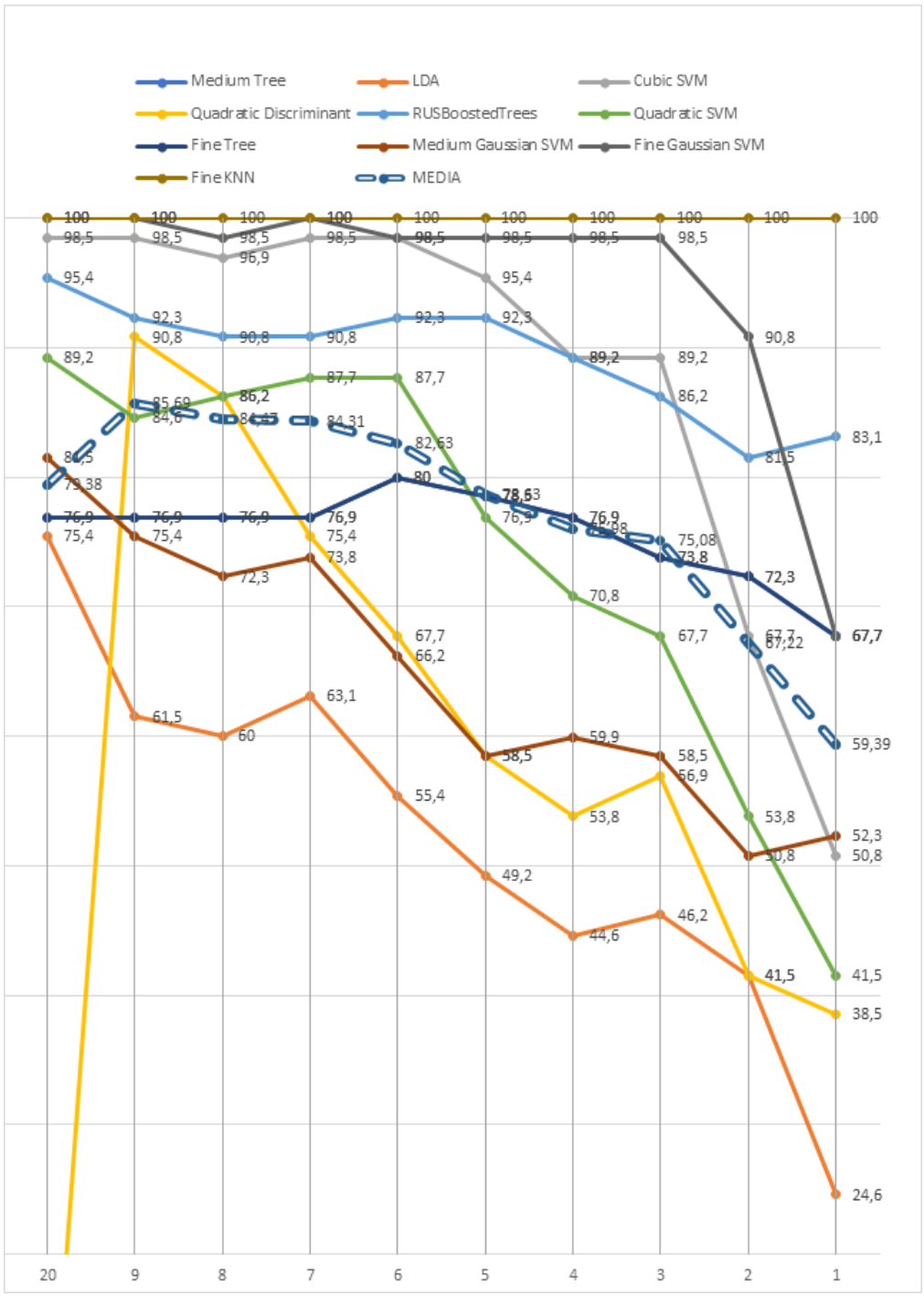
Modello	20	9	8	7	6	5	4	3	2	1
Medium tree	76.9	76.9	76.9	76.9	80	78.5	76.9	73.8	72.3	67.7
Linear Discriminant	75.4	75.4	61.5	60	55.4	49.2	44.6	46.2	41.5	24.6
Cubic SVM	98.5	98.5	96.9	98.5	98.5	95.4	89.2	89.2	67.7	50.8
Quadratic Discriminant	––	90.8	86.2	75.4	67.7	58.5	53.8	56.9	41.5	38.5
RUSBoosted Trees	95.4	92.3	90.8	90.8	92.3	92.3	89.2	86.2	81.5	83.1
Quadratic SVM	89.2	84.6	86.2	87.7	87.7	76.9	70.8	67.7	53.8	41.5
Fine Tree	76.9	76.9	76.9	76.9	80	78.5	76.9	73.8	72.3	67.7
Medium Gaussian SVM	81.5	75.4	72.3	73.8	66.2	58.5	59.9	58.5	50.8	52.3
Fine Gaussian SVM	100	100	98.5	100	98.5	98.5	98.5	98.5	90.8	67.7
Fine KNN	100	100	100	100	100	100	100	100	100	100

In Figura 3.1 è riportato il grafico dell'andamento dell'accuratezza al variare del numero di features utilizzate per la predizione.

3.1.4 Correlazione reciproca tra features

Basandosi sulla correlazione reciproca tra le features estratte si è andati a ridurre il numero utilizzato e, ad ogni riduzione, si è eseguito il training dei principali algoritmi. I predittori da eliminare

Figura 3.1: Accuratezza al variare del numero di features - Correlazione tra features e forma della voce. La linea tratteggiata rappresenta l'andamento medio degli algoritmi



sono stati scelti secondo il seguente criterio: viene individuata la coppia a maggior correlazione reciproca e tra i componenti della coppia viene eliminato quello con la più bassa correlazione con la forma della voce. Così facendo si riesce a mantenere un buon rapporto di correlazione tra features e forma della voce andando ad eliminare le informazioni ridondanti conservando solo i predittori indipendenti.

In Tabella 3.4 vengono riportate le accuratezze ottenute in fase di training riducendo il numero di predittori secondo il metodo sopra esposto. In Figura 3.2 è rappresentato l'andamento dell'accuratezza al variare del numero dei predittori utilizzate per la predizione. Ad ogni iterazione le features utilizzate sono le seguenti:

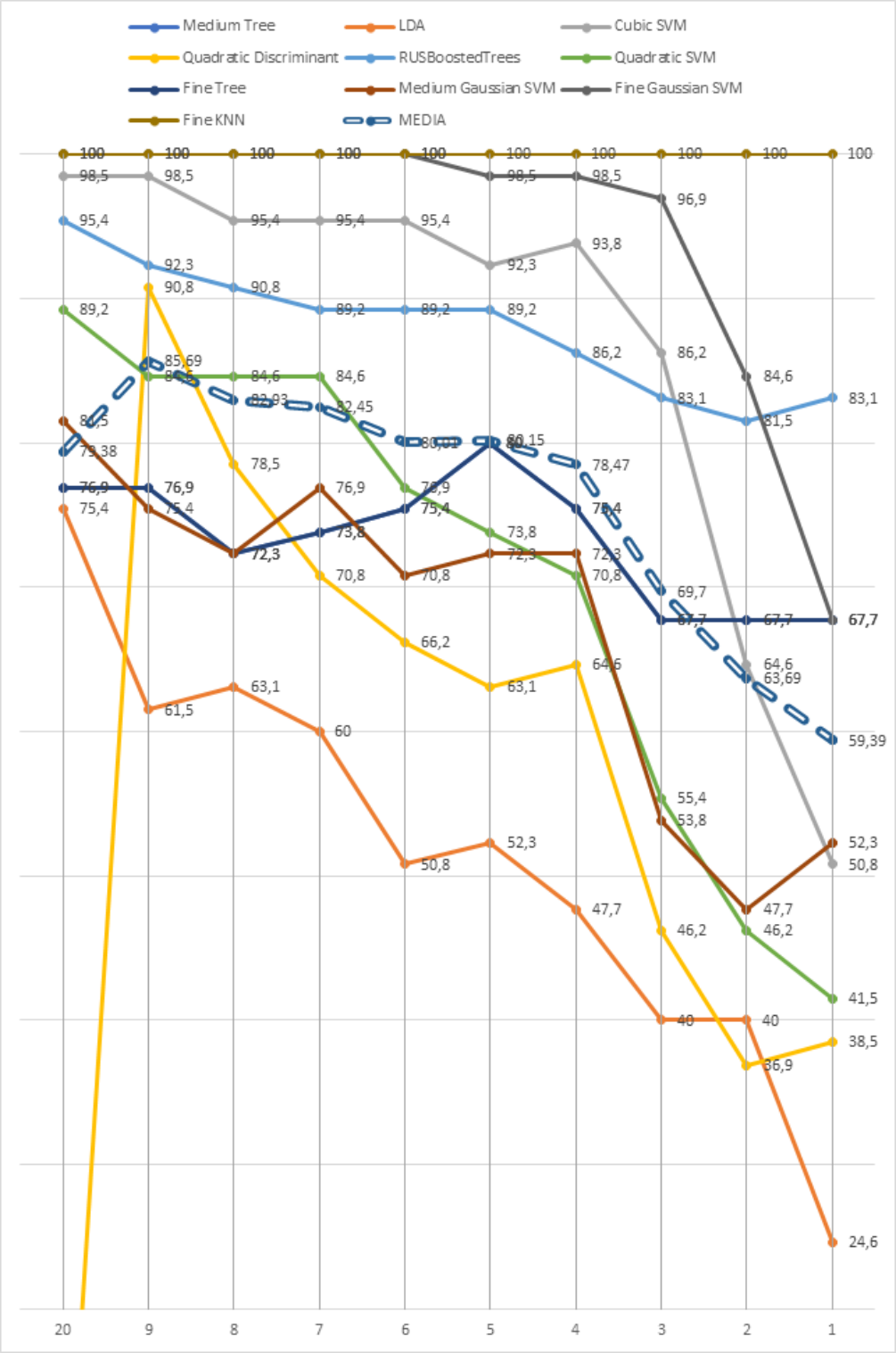
- **20 features**: utilizzate tutte
- **9 features**: utilizzate $Cpps_{std}$, $Cpps_5$, $Voiced_{mean}$, $Voiced_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$, $f0_{std}$, $dt(\%)$
- **8 features**: utilizzate $Cpps_{std}$,$Cpps_5$, $Voiced_{mean}$, Unv_{mean}, Unv_{std}, $f0_{mean}$, $f0_{std}$, $dt(\%)$
- **7 features**: utilizzate $Cpps_{std}$,$Cpps_5$, $Voiced_{mean}$, Unv_{std}, $f0_{mean}$, $f0_{std}$, $dt(\%)$
- **6 features**: utilizzate $Cpps_{std}$,$Cpps_5$, $Voiced_{mean}$, Unv_{std}, $f0_{mean}$, $f0_{std}$
- **5 features**: utilizzate $Cpps_{std}$,$Cpps_5$, Unv_{std}, $f0_{mean}$, $f0_{std}$
- **4 features**: utilizzate $Cpps_{std}$,$Cpps_5$, Unv_{std}, $f0_{std}$
- **3 features**: utilizzate $Cpps_{std}$,$Cpps_5$, Unv_{std}
- **2 features**: utilizzate $Cpps_{std}$, Unv_{std}
- **1 features**: utilizzata Unv_{std}

In Tabella 3.4 sono tabulati i valori di accuratezza al variare del numero di predittori utilizzati per il training, mentre in Figura 3.2 si può visualizzare l'andamento dell'accuratezza man mano che diminuiscono le features utilizzate.

Tabella 3.4: Modelli addestrati al decrescere del numero di features - Correlazione reciproca tra le features

Modello	20	9	8	7	6	5	4	3	2	1
Medium tree	76.9	76.9	72.3	73.8	75.4	80	75.4	67.7	67.7	67.7
Linear Discriminant	75.4	61.5	63.1	60	50.8	52.3	47.7	40	40	24.6
Cubic SVM	98.5	98.5	95.4	95.4	95.4	92.3	93.8	86.2	64.6	50.8
Quadratic Discriminant	––	90.8	78.5	70.8	66.2	63.1	64.6	46.2	36.9	38.5
RUSBoosted Trees	95.4	92.3	90.8	89.2	89.2	89.2	86.2	83.1	81.5	83.1
Quadratic SVM	89.2	84.6	84.6	84.6	76.9	73.8	70.8	55.4	46.2	41.5
Fine Tree	76.9	76.9	72.3	73.8	75.4	80	75.4	67.7	67.7	67.7
Medium Gaussian SVM	81.5	75.4	72.3	76.9	70.8	72.3	72.3	53.8	47.7	52.3
Fine Gaussian SVM	100	100	100	100	100	98.5	98.5	96.9	84.6	67.7
Fine KNN	100	100	100	100	100	100	100	100	100	100

Figura 3.2: Accuratezza al variare del numero di features - Correlazione reciproca tra le features. La linea tratteggiata rappresenta l'andamento medio degli algoritmi



3.2 Dataset Maschile

3.2.1 Analisi dei dati e scelta delle features

Nel dataset sono contenuti i parametri estratti dalle registrazioni dei soli attori maschi. 7 attori hanno interpretato le 5 forme vocali per un totale di 35 file audio. Questo dataset è stato ottenuto per estrazione da quello complessivo quindi viene conservata la stessa struttura dei parametri.

Come fatto in precedenza si procede al calcolo della *matrice di correlazione* e dei *p-value* tra i parametri estratti e la forma vocale. In questo caso il parametro *Gender* non viene preso in considerazione essendo il dataset esclusivamente maschile. I risultati ottenuti utilizzando la funzione *corr* di Matlab ™sono riportati in Tabella 3.5 e, come per il dataset complessivo, sono evidenziati in verde i parametri che verranno presi come punto di partenza per l'analisi. Anche in questo caso il valore della correlazione scelto come "buona correlazione" ($|Correlazione| > 0.25$) è inferiore a quello che solitamente si utilizza in letteratura ($|Correlazione| > 0.4$) per il fatto che non si hanno dataset numerosi e per avere una panoramica migliore del comportamento degli algoritmi al diminuire del numero di predittori utilizzati; rimane invariato invece il valore limite del *p-value* (($p - value < 0.05$).

Tabella 3.5: Coefficienti di correlazione tra forma della voce e features - Dataset maschile

Parametro	Correlazione	p-value
$CPPS_{mean}$	0.2872	0.0943
$CPPS_{median}$	0.2977	0.0824
$CPPS_{mode}$	0.3433	0.04346
$CPPS_{std}$	-0.1581	0.3645
$CPPS_{range}$	0.0904	0.6054
$CPPS_{5prc}$	0.5078	0.0018
$CPPS_{95prc}$	0.1145	0.5125
$CPPS_{skew}$	-0.3451	0.0423
$CPPS_{kurt}$	-0.051	0.7691

Parametro	Correlazione	p-value
$voiced_{mean}$	-0.7239	$8.88e-07$
$voiced_{mode}$	-0.2404	0.1642
$voiced_{std}$	-0.7143	$1.44e-06$
$unvoiced_{mean}$	-0.6321	$4.64e-05$
$unvoiced_{mode}$	-0.2425	0.1604
$unvoiced_{std}$	-0.6765	$8.14e-06$
$F0_{mean}$	0.799	$8.74e-09$
$F0_{std}$	0.3237	0.0578
RMS_{mean}	-0.0667	0.7038
RMS_{std}	0.0983	0.5742
Dt%	0.2291	0.1855

Come si evince da Tabella 3.5 le features scelte sono:
$CPPS_{mode}$, $CPPS_{5prc}$, $CPPS_{skew}$, $voiced_{mean}$, $voiced_{std}$, $unvoiced_{mean}$, $unvoiced_{std}$, $F0_{mean}$

3.2.2 Training iniziale degli algoritmi

Prendendo in considerazione le 8 features selezionate col metodo della correlazione è stato fatto il training di tutti gli algoritmi disponibili all'interno dell'applicazione *Classification Learner* di Matlab ™ i cui risultati sono riportati in Tabella 3.6. In verde sono evidenziati i modelli che hanno restituito una miglior accuratezza rispetto alla media generale e che, quindi, rappresentano una scelta ottimale. Sono stati scartati tutti gli algoritmi che hanno prodotto valori di accuratezza pari al 100 % in quanto non è stato introdotto alcun metodo di validazione del modello e, quindi,

i risultati non sono attendibili. In questa sessione di training un caso particolare è rappresentato dal modello *Quadratic Discriminant* il quale produce un fallimento in fase di training legato ad una cattiva interpretazione della varianza tra le classi da parte dell'algoritmo utilizzato per l'apprendimento.

Tabella 3.6: Modelli addestrati con 8 feature e relativa accuratezza - Dataset Maschile

Modello	Accuratezza
Fine Tree	80.0 %
Medium Tree	80.0 %
Coarse Tree	77.1 %
Linear Discriminant	85.7 %
Quadratic Discriminant	−−
Linear SVM	77.1 %
Quadratic SVM	94.3 %
Cubic SVM	100.0 %
Fine Gaussian SVM	100.0 %
Medium Gaussian SVM	85.7 %
Coarse Gaussian SVM	57.1 %

Modello	Accuratezza
Fine KNN	100.0 %
Medium KNN	54.3 %
Coarse KNN	20.0 %
Cosine KNN	48.6 %
Cubic KNN	60.0 %
Weighted KNN	100.0 %
Boosted Trees	20.0 %
Bagged Trees	100.0 %
Subspace Discriminant	74.3 %
Subspace KNN	100.0 %
RUSBoosted Trees	20.0 %

3.2.3 Correlazione tra features e forma della voce - Dataset Maschile

Individuati gli algoritmi che, utilizzando le features indicate come migliori dall'analisi della correlazione, danno le accuratezze più elevate si è proceduto con lo studio del comportamento dei suddetti algoritmi al variare del numero di predittori utilizzati in fase di training. Il procedimento di rimozione delle features è lo stesso che si è utilizzato per il dataset completo; anche in questo caso viene indicata l'accuratezza restituita utilizzando tutte le features a disposizione. Si riporta un elenco delle features utilizzate ad ogni training dei modelli.

- **20 features**: utilizzate tutte
- **8 features**: utilizzate $Cpps_{mode}$, $Cpps_5$, $Cpps_{skew}$, $Voiced_{mean}$, $Voiced_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$
- **7 features**: utilizzate $Cpps_5$, $Cpps_{skew}$, $Voiced_{mean}$, $Voiced_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$
- **6 features**: utilizzate $Cpps_5$, $Voiced_{mean}$, $Voiced_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$
- **5 features**: utilizzate $Voiced_{mean}$, $Voiced_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$
- **4 features**: utilizzate $Voiced_{mean}$, $Voiced_{std}$, Unv_{std}, $f0_{mean}$
- **3 features**: utilizzate $Voiced_{mean}$, $Voiced_{std}$, $f0_{mean}$
- **2 features**: utilizzate $Voiced_{mean}$, $f0_{mean}$
- **1 features**: utilizzata $f0_{mean}$

Scegliendo di ridurre le feature secondo l'ordine sopra elencato si sono ottenute le accuratezze riportate in Tabella 3.7

Tabella 3.7: Modelli addestrati al decrescere del numero di features - Correlazione tra features e forma della voce per dataset Maschile

Modello	20	8	7	6	5	4	3	2	1
Fine tree	82.9	80	80	80	77.1	74.3	71.4	68.6	60
Medium tree	82.9	80	80	80	77.1	74.3	71.4	68.6	60
Coarse tree	77.1	77.1	77.1	77.1	74.3	65.7	71.4	65.7	57.1
Medium Gaussian SVM	100	85.7	82.9	85.7	88.6	85.7	77.1	68.6	48.6
Quadratic SVM	94.3	94.3	91.4	94.3	91.4	91.4	82.9	74.3	48.6
Linear discriminant	94.3	85.7	77.1	77.1	71.4	71.4	65.7	57.1	51.4
Linear SVM	80	77.1	71.4	71.4	74.3	71.4	62.9	60	54.3
Subspace Discriminant	82.9	77.1	74.3	71.4	68.6	65.7	60	54.3	51.4

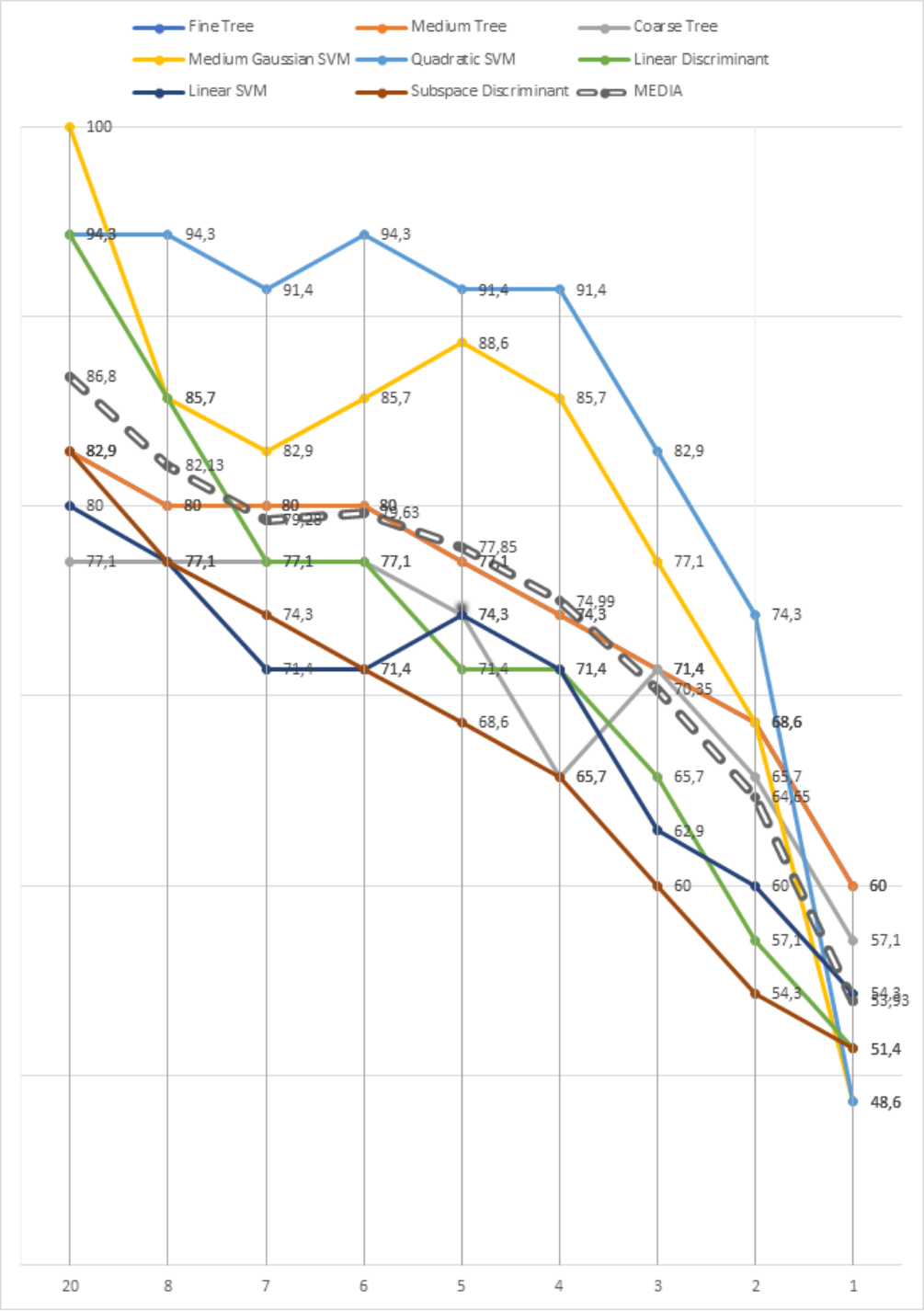
Si riporta in Figura 3.3 l'andamento dell'accuratezza per gli algoritmi selezionati al variare del numero di features utilizzate in fase di training.

3.2.4 Correlazione reciproca tra features - Dataset Maschile

Utilizzando nuovamente un processo analogo a quanto fatto per il dataset completo si è andati a ridurre le features sulla base della loro correlazione reciproca. In 3.8 sono riportati i risultati ottenuti nelle differenti sessioni di training con differenti numeri di features, mentre in Figura 3.4 è disponibile un grafico dell'andamento dei risultati di predizione.Si riporta, qui di seguito, un elenco delle features utilizzate ad ogni training dei modelli.

- **20 features**: utilizzate tutte
- **8 features**: utilizzate $Cpps_{mode}, Cpps_{5}, Cpps_{skew}, Voiced_{mean}, Voiced_{std}, Unv_{mean}, Unv_{std}, f0_{mean}$

Figura 3.3: Accuratezza al variare del numero di features - Correlazione tra features e forma vocale per dataset maschile. La linea tratteggiata rappresenta l'andamento medio degli algoritmi

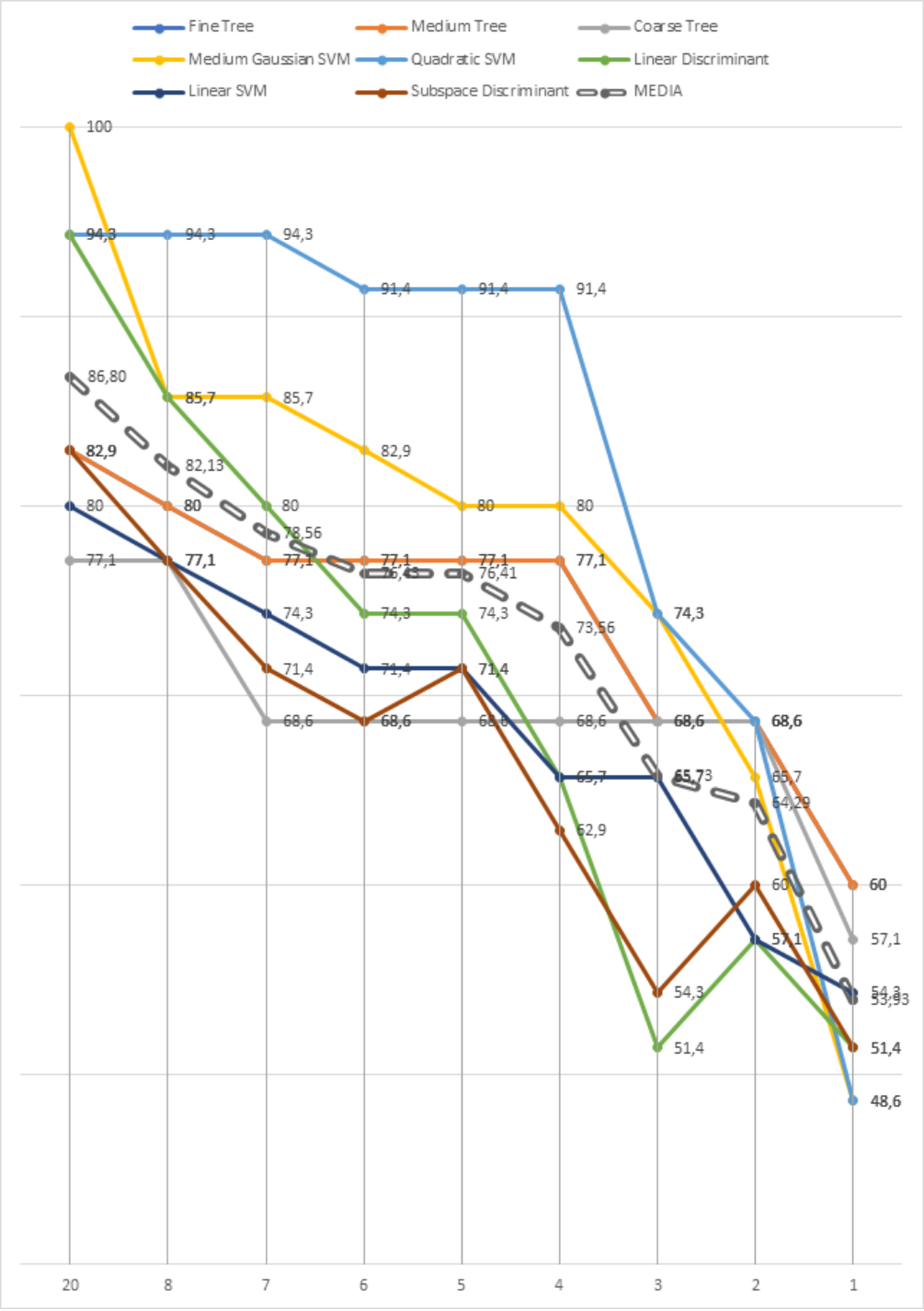


- **7 features**: utilizzate $Cpps_{mode}$, $Cpps_5$, $Cpps_{skew}$, $Voiced_{mean}$, $Voiced_{std}$, Unv_{std}, $f0_{mean}$
- **6 features**: utilizzate $Cpps_{mode}$, $Cpps_5$, $Cpps_{skew}$, $Voiced_{mean}$, Unv_{std}, $f0_{mean}$
- **5 features**: utilizzate $Cpps_{mode}$, $Cpps_5$, $Voiced_{mean}$, Unv_{std}, $f0_{mean}$
- **4 features**: utilizzate $Cpps_5$, $Voiced_{mean}$, Unv_{std}, $f0_{mean}$
- **3 features**: utilizzate $Cpps_5$, $Voiced_{mean}$, $f0_{mean}$
- **2 features**: utilizzate $Cpps_5$, $f0_{mean}$
- **1 features**: utilizzata $f0_{mean}$

Tabella 3.8: Modelli addestrati al decrescere del numero di features - Correlazione reciproca tra features per dataset Maschile. La linea tratteggiata rappresenta l'andamento medio degli algoritmi

Modello	20	8	7	6	5	4	3	2	1
Fine tree	82.9	80	77.1	77.1	77.1	77.1	68.6	68.6	60
Medium tree	82.9	80	77.1	77.1	77.1	77.1	68.6	68.6	60
Coarse tree	77.1	77.1	68.6	68.6	68.6	68.6	68.6	68.6	57.1
Medium Gaussian SVM	100	85.7	85.7	82.9	80	80	74.3	65.7	48.6
Quadratic SVM	94.3	94.3	94.3	91.4	91.4	91.4	74.3	68.6	48.6
Linear discriminant	94.3	85.7	80	74.3	74.3	65.7	51.4	57.1	51.4
Linear SVM	80	77.1	74.3	71.4	71.4	65.7	65.7	57.1	54.3
Subspace Discriminant	82.9	77.1	71.4	68.6	71.4	62.9	54.3	60	51.4

Figura 3.4: Accuratezza al variare del numero di features - Correlazione reciproca tra features per dataset maschile



3.3 Dataset Femminile

3.3.1 Analisi dei dati e scelta delle features

Nel dataset sono contenuti i parametri estratti dalle registrazioni delle sole donne. 6 attrici hanno interpretato le 5 forme vocali per un totale di 30 file audio. Questo dataset è stato ottenuto per estrazione da quello complessivo quindi viene conservata la stessa struttura dei parametri.

Come per i set precedenti si è calcolata la *matrice di correlazione* e dei *p-value* tra i parametri estratti e la forma vocale. Come nel caso del dataset maschile il parametro *Gender* non viene preso in considerazione essendo il dataset composto da sole done. I risultati ottenuti utilizzando la funzione *corr* di Matlab ™sono riportati in Tabella 3.5 e, come già fatto per gli altri dataset, in verde si trovano i parametri presi come punto di partenza per l'analisi. Per i valori di riferimento di *correlazione* e *p-value* sono state fatte le stesse considerazioni precedenti.

Tabella 3.9: Coefficienti di correlazione tra forma della voce e features - Dataset Femminile

Parametro	Correlazione	p-value
$CPPS_{mean}$	-0.19881	0.2922
$CPPS_{median}$	-0.19411	0.3040
$CPPS_{mode}$	-0.0227	0.9052
$CPPS_{std}$	-0.4601	0.0105
$CPPS_{range}$	-0.2022	0.2838
$CPPS_{5prc}$	0.0146	0.9390
$CPPS_{95prc}$	-0.3441	0.0626
$CPPS_{skew}$	0.0938	0.6219
$CPPS_{kurt}$	-0.0519	0.7854

Parametro	Correlazione	p-value
$voiced_{mean}$	-0.4539	0.0117
$voiced_{mode}$	-0.0395	0.8359
$voiced_{std}$	-0.3975	0.0296
$unvoiced_{mean}$	-0.5128	0.0038
$unvoiced_{mode}$	0.2259	0.2301
$unvoiced_{std}$	-0.6891	$2.55e-05$
$F0_{mean}$	0.7503	$1.8e-06$
$F0_{std}$	0.5907	0.0006
RMS_{mean}	-0.0719	0.7055
RMS_{std}	0.0169	0.9293
Dt%	0.3849	0.0357

Come riportato in Tabella 3.9 le features scelte sono 8:
$CPPS_{std}$, $voiced_{mean}$, $voiced_{std}$, $unvoiced_{mean}$, $unvoiced_{std}$, $F0_{mean}$, $F0_{std}$, Dt%.

A queste, che sono evidenziate in verde chiaro, c'è da aggiungere $Cpps_{95prc}$, evidenziata in verde più scuro. Si è scelto di provare ad eseguire un training dei modelli aggiungendo questa feature in quanto il suo p-value eccede di poco i limiti che ci si era prefissati ($p - value < 0.05$).

3.3.2 Training iniziale degli algoritmi

Utilizzando le 8 features elencate in precedenza si è iniziato a fare varie iterazioni del processo di training dei modelli a disposizione. I risultati sono riportati in Tabella 3.10. Sono stati scartati tutti gli algoritmi che hanno prodotto valori di accuratezza pari al 100 % in quanto non è stato

introdotto alcun metodo di validazione del modello e, quindi, i risultati non sono attendibili. In questa sessione di training un caso particolare è rappresentato dal modello *Quadratic Discriminant* il quale produce un fallimento in fase di training legato ad una cattiva interpretazione della varianza tra le classi da parte dell'algoritmo utilizzato per l'apprendimento.

Tabella 3.10: Modelli addestrati con 8 features e relativa accuratezza - Dataset Femminile

Modello	Accuratezza
Fine Tree	80.0 %
Medium Tree	80.0 %
Coarse Tree	73.3 %
Linear Discriminant	66.7 %
Quadratic Discriminant	−−
Linear SVM	70 %
Quadratic SVM	83.3 %
Cubic SVM	93.3 %
Fine Gaussian SVM	100.0 %
Medium Gaussian SVM	80.0 %
Coarse Gaussian SVM	56.7 %

Modello	Accuratezza (in %)
Fine KNN	100.0 %
Medium KNN	46.7 %
Coarse KNN	20.0 %
Cosine KNN	40.0 %
Cubic KNN	56.7 %
Weighted KNN	100.0 %
Boosted Trees	20.0 %
Bagged Trees	100.0 %
Subspace Discriminant	60.0 %
Subspace KNN	100.0 %
RUSBoosted Trees	20.0 %

3.3.3 Correlazione tra features e forma della voce - Dataset Femminile

Come fatto in precedenza anche si è poi proceduto ad effettuare varie iterazioni del processo di training così da riuscire a tracciare un grafico dell'andamento in funzione della variazione del numero di predittori utilizzati (riportato in Figura 3.5.

Il processo di rimozione è identico a quello utilizzato nei due casi precedenti con l'unica differenza che, in questo caso, si è effettuata una prima iterazione andando ad aggiungere una feature. Si riporta un elenco delle features utilizzate ad ogni training dei modelli e in Tabella 3.11 sono elencate le relative accuratezze ottenute.

- **20 features**: utilizzate tutte
- **9 features**: utilizzate $Cpps_{std}$, $Cpps_{95prc}$, $Voiced_{mean}$, $Voiced_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$, $f0_{std}$,Dt(%)
- **8 features**: utilizzate $Cpps_{std}$, $Voiced_{mean}$, $Voiced_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$, $f0_{std}$, Dt(%)
- **7 features**: utilizzate $Cpps_{std}$, $Voiced_{mean}$, $Voiced_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$, $f0_{std}$
- **6 features**: utilizzate $Cpps_{std}$, $Voiced_{mean}$, Unv_{mean}, Unv_{std}, $f0_{mean}$, $f0_{std}$
- **5 features**: utilizzate $Cpps_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$, $f0_{std}$
- **4 features**: utilizzate Unv_{mean}, Unv_{std}, $f0_{mean}$, $f0_{std}$
- **3 features**: utilizzate Unv_{std}, $f0_{mean}$, $f0_{std}$
- **2 features**: utilizzate Unv_{std}, $f0_{mean}$
- **1 features**: utilizzata $f0_{mean}$

Tabella 3.11: Modelli addestrati al decrescere del numero di features - Correlazione tra features e forma della voce per dataset Femminile

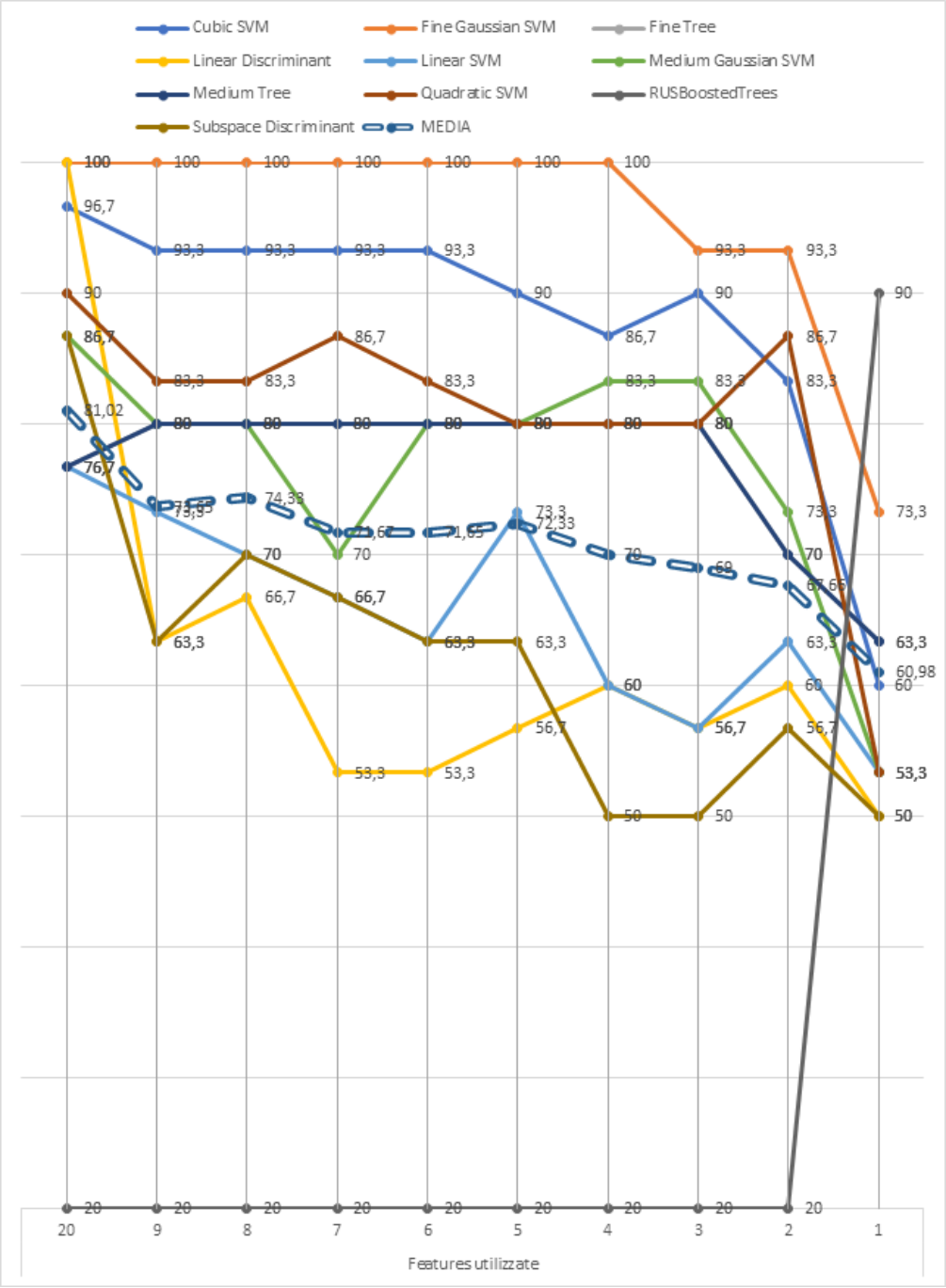
Modello	20	9	8	7	6	5	4	3	2	1
Cubic SVM	96.7	93.3	93.3	93.3	93.3	90	86.7	90	83.3	60
Fine Gaussian SVM	100	100	100	100	100	100	100	93.3	93.3	73.3
Fine tree	76.7	80	80	80	80	80	80	80	70	63.3
Linear Discriminant	100	63.3	66.7	53.3	53.3	56.7	60.0	56.7	60.0	50.0
Linear SVM	76.7	73.3	70.0	66.7	63.3	73.3	60.0	56.7	63.3	53.3
Medium Gaussian SVM	86.7	80.0	80.0	70.0	80.0	80.0	83.3	83.3	73.3	53.3
Medium tree	76.7	80	80	80	80	80	80	80	70	63.3
Quadratic SVM	90	83.3	83.3	86.7	83.3	80.0	80.0	80.0	86.7	53.3
RUSBoosted trees	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	90.0
Subspace Discriminant	86.7	63.3	70	66.7	63.3	63.3	50.0	50.0	56.7	50.0

3.3.4 Correlazione reciproca tra features - Dataset Femminile

Analogamente a quanto visto nelle sezioni precedenti (per il set maschile e per quello completo) si riportano in Tabella 3.12 i risultati ottenuti nelle differenti sessioni di training con differenti numeri di features, mentre in Figura 3.6 si può vedere un grafico dell'andamento dei risultati di predizione. Qui di seguito, un elenco delle features utilizzate ad ogni training dei modelli.

- **20 features**: utilizzate tutte

Figura 3.5: Accuratezza al variare del numero di features - Correlazione tra features e forma vocale per dataset femminile. La linea tratteggiata rappresenta l'andamento medio degli algoritmi

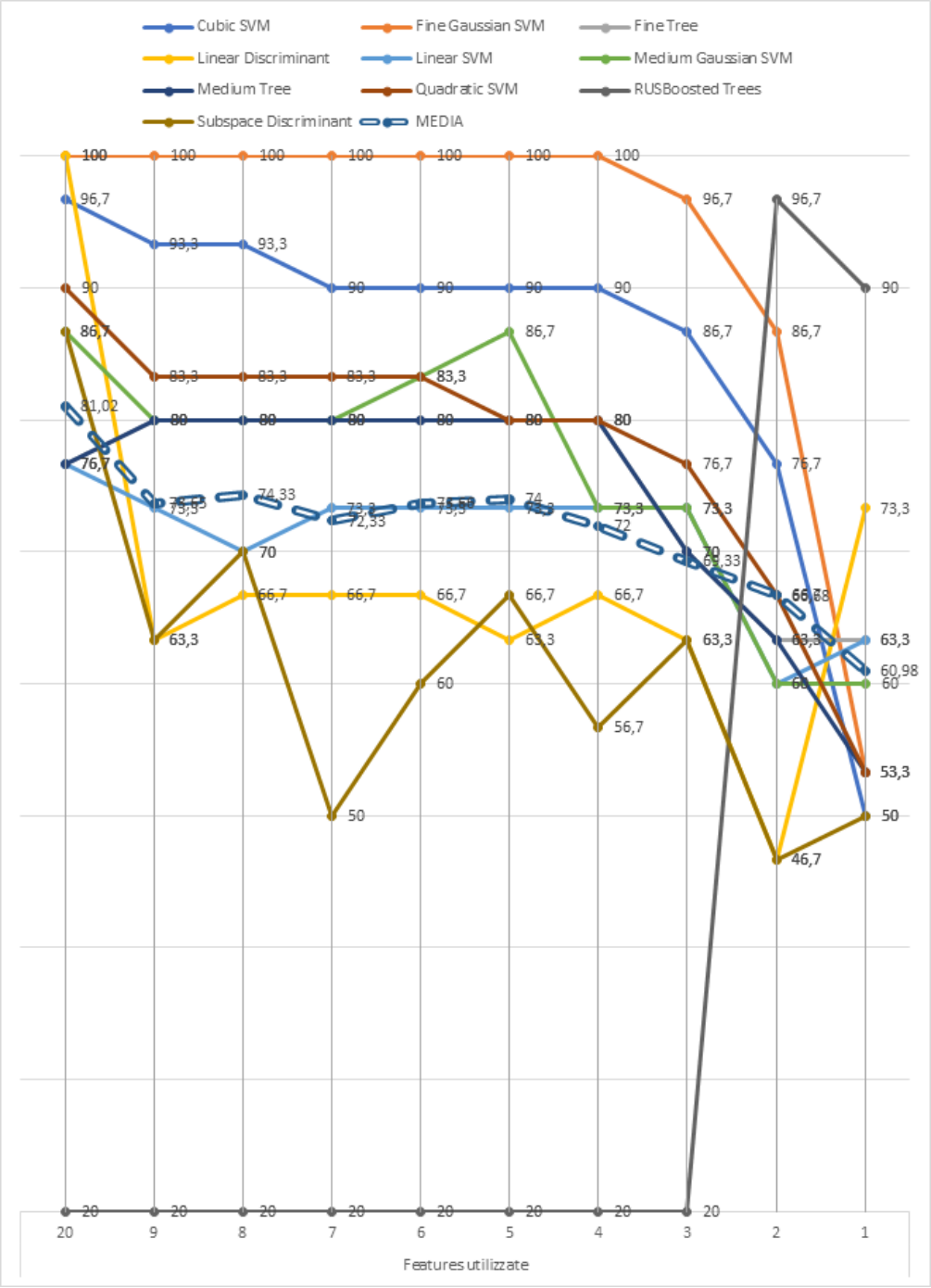


- **9 features**: utilizzate $Cpps_{std}$, $Cpps_{95prc}$, $Voiced_{mean}$, $Voiced_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$, $f0_{std}$, Dt(%)
- **8 features**: utilizzate $Cpps_{std}$, $Voiced_{mean}$, $Voiced_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$, $f0_{std}$, Dt(%)
- **7 features**: utilizzate $Cpps_{std}$, $Voiced_{mean}$, Unv_{mean}, Unv_{std}, $f0_{mean}$, $f0_{std}$, Dt(%)
- **6 features**: utilizzate $Cpps_{std}$, $Voiced_{mean}$, Unv_{std}, $f0_{mean}$, $f0_{std}$, Dt(%)
- **5 features**: utilizzate $Cpps_{std}$, Unv_{std}, $f0_{mean}$, $f0_{std}$, Dt(%)
- **4 features**: utilizzate $Cpps_{std}$, Unv_{std}, $f0_{mean}$, Dt(%)
- **3 features**: utilizzate $Cpps_{std}$, Unv_{std}, $f0_{mean}$
- **2 features**: utilizzate $Cpps_{std}$, $f0_{mean}$
- **1 features**: utilizzata $f0_{mean}$

Tabella 3.12: Modelli addestrati al decrescere del numero di features - Correlazione reciproca tra features per dataset Femminile

Modello	20	9	8	7	6	5	4	3	2	1
Cubic SVM	96.7	93.3	93.3	90.0	90.0	90.0	90.0	86.7	76.7	60
Fine Gaussian SVM	100	100	100	100	100	100	100	96.7	86.7	73.3
Fine tree	76.7	80	80	80	80	80	80	70	63.3	63.3
Linear Discriminant	100	63.3	66.7	66.7	66.7	63.3	66.7	63.3	46.7	50.0
Linear SVM	76.7	73.3	70.0	73.3	73.3	73.3	73.3	73.3	60.0	53.3
Medium Gaussian SVM	86.7	80.0	80.0	80.0	83.3	86.7	73.3	73.3	60.0	53.3
Medium tree	76.7	80.0	80.0	80.0	80.0	80.0	80.0	70.0	63.3	63.3
Quadratic SVM	90	83.3	83.3	83.3	83.3	80.0	80.0	76.7	66.7	63.3
RUSBoosted trees	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	96.7	90.0
Subspace Discriminant	86.7	63.3	70.0	50.0	60.0	66.7	56.7	63.3	46.7	50.0

Figura 3.6: Accuratezza al variare del numero di features - Correlazione reciproca tra features per dataset Femminile. La linea tratteggiata rappresenta l'andamento medio degli algoritmi



Parte II

Analisi delle complessità computazionali

Fino ad ora ci si è occupati esclusivamente dell'accuratezza restituita dagli algoritmi testati tralasciando l'analisi delle risorse richieste dai diversi modelli. Lo scopo di questa sezione è quello di identificare quale sia la complessità computazione degli algoritmi usati per la predizione dell'appartenenza dei campioni alle diverse classi e quella degli algoritmi di estrazione delle features. Si è analizzato questo aspetto degli algoritmi per poter effettuare una scelta più precisa di quali siano i migliori modelli e parametri per una futura implementazione del lavoro svolto in quanto un minor numero di parametri strettamente necessari alla classificazione implica una complessità inferiore per l'estrazione.
Quanto osservato per la complessità computazionale durante la fase di estrazione dei parametri necessari alla classificazione verrà poi messo in relazione con la complessità richiesta dagli algoritmi di predizione così da ottenere una panoramica completa delle risorse richieste da tutto il processo di estrazione dei predittori e il relativo utilizzo per la classificazione dei campioni.
Il computo della complessità computazionale e dei tempi di esecuzione degli algoritmi è stato effettuato sia su computer classici sia su un dispositivo creato appositamente per l'analisi del segnale vocale e per l'estrazione dei parametri ad esso relativi (Vocal Holter Med). Il Vocal Holter Med non è stato creato durante gli studi sulla forma della voce ma durante quelli di cui si parla nell'Appendice C, ovvero quelli sui malati di Parkinson.

La differenziazione tra i diversi dispositivi è stata necessaria per avere una miglior stima della fattibilità degli obiettivi di cui si parla all'interno del Capitolo 1. Grazie ad essa è stato possibile avere una panoramica del comportamento dei dispositivi durante l'estrazione dei parametri e il loro utilizzo; è stato inoltre possibile definire funzioni di approssimazione della variazione dei tempi di esecuzione al variare della dimensione dei file utilizzati.

3.4 Test su computer

Per procedere con l'analisi delle risorse richieste durante l'esecuzione su computer si è adottato un approccio di tipo sperimentale volto a individuare la spesa media per ciascun algoritmo, ottenuta testandoli su differenti macchine. Per effettuare i test sono stati utilizzati cinque file audio della durata di 1 secondo ciascuno; ogni file rappresentava una diversa forma della voce. In primis è stato necessario identificare le operazioni "base" svolte da ogni algoritmo. Con *operazione base* si

intende un operazione che non può essere ulteriormente scomponibile in sotto-operazioni: alcuni esempi di operazione base (o semplice) sono la somma, la sottrazione, il calcolo della lunghezza di un vettore o il calcolo di un logaritmo.
Individuate le operazioni componenti ogni algoritmo, per ognuno di essi è stato effettuato un semplice conto delle occorrenze di tali operazioni trovandone il totale necessario per processare i diversi file. Dato che il conto è stato fatto differenziando a seconda della forma della voce analizzata si è poi fatta la media del numero di operazioni necessarie al processo ottenendo, quindi, un numero che fosse un indicazione piuttosto affidabile delle operazioni richieste e, quindi, della spesa media in termini di risorse richieste.
Trovato il numero medio è stata effettuata una ricerca in letteratura della complessità computazionale delle operazioni base identificando, così, le operazioni di maggior impatto sull'esecuzione degli algoritmi. Grazie a questa ricerca è stato possibile produrre una stima del costo computazionale per ogni algoritmo.

Gli script utilizzati per estrarre i parametri sono riportati in Tabella 3.13. Lo script chiamato *CPPS* contiene al suo interno il processo di estrazione di anche tutti gli altri parametri non legati al CPPS; inoltre, è una versione ottimizzata degli altri script. Il confrontare questo script con gli altri è stato fatto per avere una stima del cambiamento prestazionale che si ottiene con un algoritmo ottimizzato rispetto ad uno che non lo è. C'è, tuttavia, la necessità di fare un appunto a tale analisi: i quattro script necessari per l'estrazione delle features derivano da precedenti lavori svolti sull'estrazione dei parametri e si trovano a dover rifare delle operazioni già presenti in altri algoritmi a loro analoghi, situazione che li porta ad avere un tempo di esecuzione totale ovviamente più elevato se comparato con lo script CPPS il quale riutilizza le variabili che calcola una volta sola. Ad esempio, ognuno dei quattro script deve effettuare l'apertura e lettura del file audio da analizzare occupando ogni volta il processore per rifare un'operazione che nello script CPPS.m viene fatta unicamente all'inizio.

Tabella 3.13: Algoritmi utilizzati per l'estrazione delle features

Nome	Descrizione
f0_rms.m	Estrazione dei parametri legati a frequenza e intensità del segnale vocale.
voice_unvoice.m	Identificazione dei frame di voce e di silenzio e relativa estrazione dei parametri
voicedp.m	Creazione di file audio senza frame di silenzio
cppsvoice.m	Estrazione dei parametri legati al CPPS
CPPS	Estrazione dei parametri legati al CPPS. Script maggiormente ottimizzato.

L'analisi è stata effettuata su 3 macchine differenti, riportate in Tabella 3.14. Sono stati utilizzati due Laptop e un Pc Desktop a differenti prestazioni così da avere una panoramica del comportamento al variare dell' hardware e del software a disposizione. Sulle tre macchine era installato il Software Matlab ™ all'interno del quale venivano utilizzati gli algoritmi elencati in Tabella 3.13. Per il conteggio delle operazioni è stato utilizzato un metodo molto semplice in cui sono state create delle variabili dedicate al conteggio di ogni operazione che, ad ogni occorrenza, venivano aumentate di un'unità per la variabile rappresentante l'operazione eseguita. Per il calcolo del tempo di esecuzione, invece, sono stati utilizzati i comandi *tic* e *toc* di Matlab ™ i quali permettono di ottenere il tempo necessario all'esecuzione del codice contenuto tra le due parole.

Tabella 3.14: Macchine utilizzate per il testing della complessità computazionale

Caratteristiche	Macchina 1	Macchina 2	Macchina 3
Tipologia	Laptop	Laptop	Desktop
Processore	Intel I7-4500U	Intel I5-4210U	Intel I7-4790
Clock rate	1,80 GHz	1,70 GHz	3,60 GHz
Core/Thread	2 core, 4 thread	2 core, 4 thread	4 core, 8 thread
RAM	12 Gb	4 Gb	8 Gb
Architettura	64 bit	64 bit	64 bit

3.4.1 Testing degli algoritmi - Algoritmo unico

L'algoritmo qui analizzato è l'algoritmo CPPS.m il quale, come detto, permette l'estrazione di tutte le features necessarie. Si è iniziato andando a trovare il numero di operazioni necessarie per processare i file audio. Il conteggio è stato effettuato per ogni forma vocale e i risultati sono riportati in Tabella 3.4.1. Le prime cinque colonne indicano le forme della voce; l'ultima colonna, invece, contiene il numero medio di volte in cui l'operazione viene eseguita.

Tabella 3.15: Numero di operazioni richieste per l'estrazione delle features dalle 5 forme vocali. Pt: piatta, Ton:tonda, Quad: quadra, Tri: triangolare, Spi: spigolosa

Operazione	Pt	Ton	Quad	Tri	Spi	Media
Lettura file XLS	1	1	1	1	1	1
Comparazione stringhe	1	1	1	1	1	1
Concatenazione stringhe	4	4	4	4	4	4

Lettura file audio	1	1	1	1	1	1
Dimensioni matrici/array	923	668	695	814	716	763.2
OR logico	2	2	2	2	2	2
Sotto campionamento	1	1	1	1	1	1
Trasposizione matrice	849	668	695	814	716	763.2
Calcolo RMS	475	475	475	475	475	475
Valore assoluto	1112	712	746	1169	737	895.2
Massimo	1114	858	782	1171	741	933.2
Logaritmo	1112	712	746	1169	737	895.2
Aggiunta rumore Gaussiano	1	1	1	1	1	1
Somma matrice	2	2	2	2	2	2
Floor	745	384	485	783	497	578.8
Arrotonda	3	3	3	3	3	3
Media	476	476	476	476	476	476
op. puntoPunto	818	549	531	781	505	636.8
Somma vettore	918	830	763	862	730	820.6
Hamming	370	189	238	389	245	286.2

FFT	740	378	476	778	490	572.4
Controllo NAN	0	0	0	0	0	0
Minimo	371	190	239	390	246	287.2
Percentile	6	6	6	6	6	6
Filtraggio	2	2	2	2	2	2
Pseudoinversione	472	472	472	472	472	472
Sorting	4	4	4	4	4	4
Moda	4	4	4	4	4	4
Deviazione standard	4	4	4	4	4	4
Range	4	4	4	4	4	4
5° percentile	4	4	4	4	4	4
95° percentile	4	4	4	4	4	4
Skewness	4	4	4	4	4	4
Kurtosis	4	4	4	4	4	4
Mediana	4	4	4	4	4	4
Vettore caratteri	1	1	1	1	1	1
Divisioni	$84.4e06$	$75.78e06$	$61.21e06$	$88.75e06$	$55.98e06$	$73.23e06$
Somme	$168e06$	$151e06$	$122e06$	$177e06$	$112e06$	$146e06$
AND logico	1852	1019	1207	1947	1227	1450.4
Trova	3	3	3	3	3	3

Trovato il numero delle singole operazioni, per le principali (quelle fatte più volte) si va a individuare, cercando in letteratura, il costo computazionale. I risultati sono mostrati in Tabella 3.16

Tabella 3.16: Complessità computazionale delle principali operazioni effettuate. N=dimensione vettore/numero righe. M=numero colonne

Operazione	Numero medio	Costo
Dimensioni matrici/array	763.2	O(1) o $O(N)$
Trasposizione matrice	763.2	$O(N \cdot M)$
Calcolo RMS	475	$O(N)$
Valore assoluto	895.2	$O(1)$
Massimo	933.2	$O(N)$
Logaritmo	895.2	$O(1)$
Floor	578.8	$O(1)$
Media	476	$O(N)$

Operazione	Numero medio	Costo
operazione punto punto	636.8	$O(N)$
FFT	572.4	$O(NlogN)$
Minimo	287.2	$O(N)$
Somma vettore	820.6	$O(N)$
Pseudoinversione	472	$O(N^3)$
Divisioni	$73.23e06$	$O(1)$
Somme	$146e06$	$O(1)$
AND logico	1450.4	$O(1)$

Il costo computazionale di questo algoritmo si può approssimare con $O(N^3)$. Indipendentemente dal numero di operazioni effettuate la notazione degli *O Grande* permette di trovare un limite massimo a cui l'algoritmo tende per quanto riguarda la complessità.

Il numero di operazioni è indipendente dal tipo di macchina su cui viene eseguito l'algoritmo. Per mettere in relazione il tempo necessario ad una macchina per eseguire uno script e le sue caratteristiche hardware e software sono stati eseguiti differenti test sui computer indicati in Tabella 3.14. Si riportano in Tabella 3.17 i risultati evidenziando per ogni forma della voce il tempo necessario

all'estrazione dei parametri e indicando, infine, un tempo medio per l'esecuzione.

Tabella 3.17: Tempo (espresso in secondi) necessario per estrazione delle features con algoritmo CPPS.m. Pt: piatta, Ton:tonda, Quad: quadra, Tri: triangolare, Spi: spigolosa

ID Macchina	Pt	Ton	Quad	Tri	Spi	Media
Macchina 1	0.124967	0.111714	0.111479	0.117128	0.112	0.115458
Macchina 2	0.131249	0.113648	0.119319	0.133571	0.162219	0.132001
Macchina 3	0.082483	0.079823	0.078399	0.073203	0.076514	0.078084

3.4.2 Testing degli algoritmi - Algoritmi estrazione separati

Per gli altri algoritmi di estrazione delle features si è seguito lo stesso approccio usato per quello "unico" andando prima a conteggiare il numero di operazioni necessarie e poi ad individuare il tempo necessario all'esecuzione dei differenti algoritmi sulle macchine utilizzate.

f0_rms.m

Tabella 3.18: Operazioni richieste per l'estrazione delle features con f0_rms.m. Pt: piatta, Ton: tonda, Quad: quadra, Tri: triangolare, Spi: spigolosa

Operazione	Pt	Ton	Quad	Tri	Spi	Media
Somma	9119116	10091736	7661822	14225690	8147044	9849081.6
Concatenazione stringhe	1	1	1	1	1	1
Lettura file audio	1	1	1	1	1	1
Massimo Minimo	254	200	138	292	194	215.6

Logaritmo	150	166	126	234	134	162
Valore assoluto	75	83	63	117	67	81
Calcolo RMS	99	115	191	160	128	138.6
Decimate	1	1	1	1	1	1
Divisioni	9117421	10089707	7658517	14222864	8144838	9846669.4
Lunghezza vettore	226	215	260	306	225	246.4
Floor	172	85	46	131	110	108.8
Media	100	116	192	161	129	139.6
operazione punto punto	98	114	190	159	127	137.6
Trova	2	2	2	2	2	2
Percentile	4	4	4	4	4	4
Deviazione standard	2	2	2	2	2	2

Tabella 3.20: Tempo (espresso in secondi) necessario per estrazione delle features con f0_rms.m. Pt: piatta, Ton:tonda, Quad: quadra, Tri: triangolare, Spi: spigolosa

ID Macchina	Pt	Ton	Quad	Tri	Spi	Media
Macchina 1	0.696866	0.766684	0.806903	0.824385	0.834897	0.785947
Macchina 2	0.793779	0.680632	0.685221	0.970427	0.790064	0.784023
Macchina 3	0.453901	0.388172	0.467398	0.429559	0.465281	0.440862

Tabella 3.19: Costo computazionale delle principali operazioni richieste da f0_rms.m.

Operazione	Numero medio	Costo
Somma	9849081.6	$O(1)$
Massimo Minimo	215.6	$O(N)$
Logaritmo	162	$O(1)$
Calcolo RMS	138.6	$O(N)$

Operazione	Numero medio	Costo
Divisioni	9846669.4	$O(1)$
Lunghezza vettore	246.4	$O(N)$ o $O(1)$
Floor	108.8	$O(1)$
Media	139.6	$O(N)$
operazione punto punto	137.6	$O(N)$

voice_unvoice.m

Tabella 3.21: Numero di operazioni richieste per l'estrazione delle features con algoritmo voice_unvoice.m. Pt: piatta, Ton:tonda, Quad: quadra, Tri: triangolare, Spi: spigolosa

Operazione	Pt	Ton	Quad	Tri	Spi	Media
Somma	206	203	200	206	199	202.8
Concatenazione stringhe	1	1	1	1	1	1
Lettura file audio	1	1	1	1	1	1
Trasposizione matrice	1	1	1	1	1	1
Lunghezza vettore	3	3	3	3	3	3
Divisioni	47	47	47	47	47	47

Moltiplicazioni	126	126	126	126	126	126
Calcolo RMS	21	21	21	21	21	21
Media	2	2	2	2	2	2
Operazione punto punto	2	2	2	2	2	2
Deviazione standard	2	2	2	2	2	2
Moda	2	2	2	2	2	2

Tabella 3.22: Costo computazionale delle principali operazioni dell'algoritmo voice_unvoice.m. N=lunghezza vettore

Operazione	Numero medio	Costo
Somma	202.8	$O(1)$
Concatenazione stringhe	1	$O(N^2)$
Trasposizione matrice	1	$O(N^2)$
Lunghezza vettore	3	$O(N)$
Divisioni	47	$O(1)$
Moltiplicazioni	126	$O(1)$

Tabella 3.23: Tempo (espresso in secondi) necessario per estrazione delle features con algoritmo voice_unvoice.m.. Pt: piatta, Ton:tonda, Quad: quadra, Tri: triangolare, Spi: spigolosa

ID Macchina	Pt	Ton	Quad	Tri	Spi	Media
Macchina 1	0.197786	0.227903	0.243415	0.197248	0.194055	0.212081
Macchina 2	0.506500	0.345453	0.252213	0.226905	0.207451	0.307704
Macchina 3	0.144170	0.181501	0.140457	0.135085	0.144609	0.149164

voicedp.m

Tabella 3.24: Numero di operazioni richieste per l'estrazione delle features con algoritmo voicedp.m. Pt: piatta, Ton:tonda, Quad: quadra, Tri: triangolare, Spi: spigolosa

Operazione	Pt	Ton	Quad	Tri	Spi	Media
Lettura file audio	1	1	1	1	1	1
Decimate	1	1	1	1	1	1
Divisioni	91	91	91	91	91	91
Lunghezza vettore	3	3	3	3	3	3
Calcolo RMS	21	21	21	21	21	21
Somma	84	84	84	84	84	84
Media	2	2	2	2	2	2
Massimo	1	1	1	1	1	1

Scrittura file audio	1	1	1	1	1	1

Tabella 3.25: Costo computazionale delle principali operazioni richieste dall'algoritmo voicedp.m. N=lunghezza vettore

Operazione	Numero medio	Costo
Divisioni	91	$O(1)$
Calcolo RMS	21	$O(N)$
Somma	84	$O(1)$

Tabella 3.26: Tempo (espresso in secondi) necessario per la rimozione dei frame di silenzio. Pt: piatta, Ton:tonda, Quad: quadra, Tri: triangolare, Spi: spigolosa

ID Macchina	Pt	Ton	Quad	Tri	Spi	Media
Macchina 1	0.718821	0.433889	0.436958	0.455427	0.465165	0.502052
Macchina 2	0.602329	0.481278	0.478853	0.517951	0.558197	0.527722
Macchina 3	0.290587	0.311262	0.288163	0.290284	0.281554	0.527722

CPPS_opt.m

Tabella 3.27: Numero di operazioni richieste per l'estrazione delle features con algoritmo CPPS_opt.m. Pt: piatta, Ton:tonda, Quad: quadra, Tri: triangolare, Spi: spigolosa

Operazione	Pt	Ton	Quad	Tri	Spi	Media
Lettura file audio	1	1	1	1	1	1
Trasposizione matrice	1955	1955	1955	1955	1955	1955
Lunghezza vettore	1	1	1	1	1	1
Divisioni	5395	5395	5395	5395	5395	5395
Floor	2	2	2	2	2	2
Arrotonda	3	3	3	3	3	3
Somma	4904	4904	4904	4904	4904	4904
Logaritmo	980	980	980	980	980	980
Valore assoluto	980	980	980	980	980	980
Hamming	980	980	980	980	980	980
Operazioni punto punto	982	982	982	982	982	982
FFT	490	490	490	490	490	490
Media	1	1	1	1	1	1
Filtraggio	2	2	2	2	2	2
Dimensioni vettore	2	2	2	2	2	2
Massimo	1	1	1	1	1	1
Minimo	1	1	1	1	1	1
Moda	1	1	1	1	1	1

Mediana	1	1	1	1	1	1
Deviazione standard	1	1	1	1	1	1
Range	1	1	1	1	1	1
5° percentile	1	1	1	1	1	1
95° percentile	1	1	1	1	1	1
Skewness	1	1	1	1	1	1
Kurtosis	1	1	1	1	1	1

Tabella 3.28: Costo computazionale delle principali operazioni richieste dall'algoritmo CPPS_opt.m. N=lunghezza vettore

Operazione	Numero medio	Costo
Trasposizione matrice	1955	$O(N^3)$
Divisioni	5395	$O(1)$
Somma	4904	$O(1)$
Logaritmo	980	$O(1)$
Valore assoluto	980	$O(1)$
Hamming	980	$O(N)$
Operazioni punto punto	982	$O(N)$
FFT	490	$O(NlogN)$

Tabella 3.29: Tempo (espresso in secondi) necessario per l'estrazione delle features con algoritmo CPPS_opt.m. Pt: piatta, Ton:tonda, Quad: quadra, Tri: triangolare, Spi: spigolosa

ID Macchina	Pt	Ton	Quad	Tri	Spi	Media
Macchina 1	1.559164	1.062917	1.001100	0.874944	0.883578	1.076341
Macchina 2	1.094740	0.943856	0.985086	0.987061	1.029477	1.008044
Macchina 3	0.495076	0.474984	0.478627	0.473598	0.471315	0.47872

Fatti questi test sugli script per l'estrazione separata delle features, per avere un'analisi più precisa del comportamento degli algoritmi è stata condotta anche una sperimentazione utilizzando i file generati dall'algoritmo *voicedp.m*, ovvero quelli a cui sono stati eliminati i frame di silenzio. L'idea alla base di questo test è quella di individuare come varia il tempo impiegato se i file contengono un numero minore di frame da analizzare. In tal modo si può modellare meglio quella che è la reale applicazione degli algoritmi: prima si individuano i frame di voce e quelli di silenzio e poi si vanno ad effettuare i calcoli dei parametri solo per quelli che contengono del parlato.

Sono state utilizzate, nuovamente, le tre macchine indicate in Tabella 3.14, seguendo lo stesso procedimento di calcolo del tempo necessario per ognuna delle cinque forme della voce.

Tabella 3.30: Tempo (espresso in secondi) necessario per l'esecuzione degli algoritmi su Macchina 1 dopo VoiceDP.m. Pt: piatta, Ton:tonda, Quad: quadra, Tri: triangolare, Spi: spigolosa

Algoritmo	Pt	Ton	Quad	Tri	Spi	Media
F0_rms.m	0.565109	0.639153	0.595969	0.706912	0.817626	0.664954
VoiceUnvoice.m	0.182744	0.208677	0.1813	0.199966	0.237176	0.201973
CppsVoice_opt.m	0.832640	0.919687	0.823945	0.833380	0.952868	0.872504

Tabella 3.31: Tempo (espresso in secondi) necessario per l'esecuzione degli algoritmi su Macchina 2 dopo VoiceDP.m. Pt: piatta, Ton:tonda, Quad: quadra, Tri: triangolare, Spi: spigolosa

Algoritmo	Pt	Ton	Quad	Tri	Spi	Media
F0_rms.m	0.623354	0.637396	0.666718	0.635759	0.824497	0.677545
VoiceUnvoice.m	0.222100	0.225143	0.218232	0.234417	0.278904	0.235739
CppsVoice_opt.m	0.928886	0.994941	0.935344	0.927268	0.936729	0.932634

Tabella 3.32: Tempo (espresso in secondi) necessario per l'esecuzione degli algoritmi su Macchina 3 dopo VoiceDP.m. Pt: piatta, Ton:tonda, Quad: quadra, Tri: triangolare, Spi: spigolosa

Algoritmo	Pt	Ton	Quad	Tri	Spi	Media
F0_rms.m	0.347378	0.356954	0.367547	0.386153	0.367672	0.365141
VoiceUnvoice.m	0.136224	0.133115	0.134380	0.131999	0.131088	0.133361
CppsVoice_opt.m	0.478639	0.475314	0.473310	0.462795	0.464778	0.470967

Analizzando ed unendo tutti questi risultati, è stato possibile individuare quale, tra quelli proposti sia il miglior algoritmo per l'estrazione delle feature. Per la decisione ci si è basati su alcuni parametri accessori che verrano esposti più in dettaglio nel Capitolo successivo.

3.4.3 Tempi di esecuzione delle singole operazioni

Progredendo nell'analisi della complessità computazionale degli algoritmi quello che è stato fatto è una valutazione del tempo necessario al completamento di singole operazioni, le quali corrispondono a quelle evidenziate come più frequenti durante la fase di conteggio delle stesse. Per fare questo test la procedura seguita è la seguente:

1. Eseguire all'interno di un loop una determinata operazione N volte

2. Trovare il tempo necessario per eseguire questa operazione N volte
3. Individuare il tempo necessario ad eseguire il ciclo ”vuoto” N volte
4. Sottrarre il tempo impiegato per l’esecuzione del ciclo a quello impiegato per l’esecuzione dell’operazione N volte
5. Dividere il risultato trovato per N così da avere una stima del tempo impiegato per una singola operazione.

Sempre prendendo come riferimento le macchine indicate in Tabella 3.14 è stato eseguito questo test i cui risultati sono riportati in Tabella 3.33.

Tabella 3.33: Tempo per l’esecuzione di singole operazioni. Tempo 1: macchina 1; Tempo 2: macchina 2; Tempo 3: macchina 3

Operazione	Tempo 1 (ns)	Tempo 2 (ns)	Tempo 3 (ns)
Somma	0.47470	0.5616	0.2933
Sottrazione	0.74290	2.241	0.0333
Divisione	2.68	2.63	0.4101
Moltiplicazione	3.10	1.22	0.0018
Massimo	43.244	638.09	24.75
Trasposta	517.65	588.52	334.72
Operazione punto punto (10 elementi)	102.11	90.95	39.55
Logaritmo	39.289	70.17	30.60
Valore assoluto	2.4062	3.129	1.4348

Verificare il tempo di esecuzione di una singola operazione per ognuna delle macchine utilizzate durante il test permette di unire questi risultati con le complessità computazionali di ogni operazione ed avere, quindi, una stima della risposta della macchina a seconda di quale operazione viene

effettuata. Questi risultati vanno, però, considerati nel più ampio panorama di esecuzione degli script all'interno dei quali non solo queste operazioni "principali" contribuiscono all'aumento del tempo di esecuzione totale.

3.5 Test su Vocal Holter Med

Per avere un riferimento differente dall'architettura di un computer e dalla sua complessità sono stati eseguiti dei test sul dispositivo Vocal Holter Med (VHM).
Tale dispositivo, progettato durante precedenti studi svolti nei laboratori del Politecnico di Torino, permette di registrare un segnale vocale utilizzando un microfono a contatto e di elaborarlo successivamente per estrarre i parametri di interesse.
All'interno del VHM è installata una scheda del tipo NanoPi Duo prodotta da FriendlyElec; sulla scheda è montato il processore Allwinner H2+, Quad-core Cortex-A7 dotato di 4 core e 8 thread il quale, stando alla scheda fornita dal produttore, ha una frequenza di clock massima pari a 1.296 GHz.

Come per le analisi fatte sui tre computer, anche in questo caso non si può avere una stima precisa dei tempi di calcolo necessari per completare le differenti operazioni in quanto il processore utilizza il parallelismo per svolgere più velocemente i calcoli richiesti; inoltre, durante l'elaborazione dei dati, vengono svolte operazioni "accessorie" che richiedono l'utilizzo del processore anche durante i calcoli di nostro interesse. In questo modo il tempo di calcolo va, anche se di poco, ad aumentare e non si ha la possibilità di verificare quanto siano influenti le altre operazioni sull'esecuzione del codice principale in quanto non è possibile arrestare completamente le operazioni "accessorie" e stimare esclusivamente il tempo di calcolo dell'algoritmo di nostro interesse.
L'analisi che è stata fatta è, quindi, una stima del tempo necessario al device VHM per elaborare i file in input secondo differenti parametri di elaborazione, esposti in dettaglio più avanti.

Le misure sul VHM sono state condotte con una metodologia leggermente diversa rispetto a quella utilizzata per le 3 macchine.
Per rappresentare il caso peggiore è stato registrato un segnale sempre "sopra soglia" (senza istanti di silenzio) il quale veniva poi diviso in frame da 46 ms ciascuno. Successivamente sono stati eseguiti calcoli sul file stesso cambiando, in un caso la dimensione della finestra utilizzata (si prendevano in considerazione più o meno frame) e, nell'altro caso, la tipologia di calcolo effettuato (quali e quanti parametri venivano richiesti). Confrontando i risultati ottenuti si riesce a capire quale sia l'influenza del cambiare operazioni e quantità di dati sul tempo di esecuzione da parte

Tabella 3.34: Tempi per calcolo distribuzione CPPS e altri parametri

Numero frame	Tempo (s)
1 frame	13.2
2 frame	17.0
4 frame	24.36
8 frame	38.9

del VHM.

3.5.1 Estrazione CPPS e altri parametri

Il primo test è stato condotto estraendo i parametri legati a frequenza, intensità e distribuzione del CPPS per un numero di frame crescente. Sono state effettuate le misurazioni utilizzando quantità di frame crescenti pari a 1,2,4 e 8 frame. I risultati sono riportati in Tabella 3.34. In Figura 3.7 viene visualizzato l'andamento del tempo di esecuzione; sull'asse X c'è il numero di frame analizzati, sull'asse Y ci sono i relativi tempi ottenuti (espressi in secondi). Da tale grafico si nota la linearità del tempo di esecuzione all'aumentare della quantità di frame utilizzati.

3.5.2 Estrazione CPPS

Questo test è stato fatto richiedendo l'estrazione esclusivamente della distribuzione del CPPS. Il numero di frame utilizzati è pari a 1,2,4,8 e 16. I risultati sono riportati in tabella 3.35. In figura 3.8 viene visualizzato l'andamento del tempo di esecuzione; sull'asse delle ascisse c'è il numero di frame analizzati, sull'asse delle ordinate i relativi tempi ottenuti nelle diverse misurazioni (espressi in secondi).

In entrambi i test effettuati si nota come il tempo necessario all'esecuzione cresca linearmente al crescere dei frame utilizzati.
Questo risultato, come verrà descritto più in dettaglio all'interno del Capitolo 4, ha permesso di arrivare a conclusioni riguardanti il tempo di calcolo necessario per un numero di frame maggiore

Figura 3.7: Rappresentazione grafica dell'andamento del tempo di esecuzione estraendo tutte i parametri con VHM per numero di frame crescente. La linea tratteggiata rappresenta l'approssimazione calcolata.

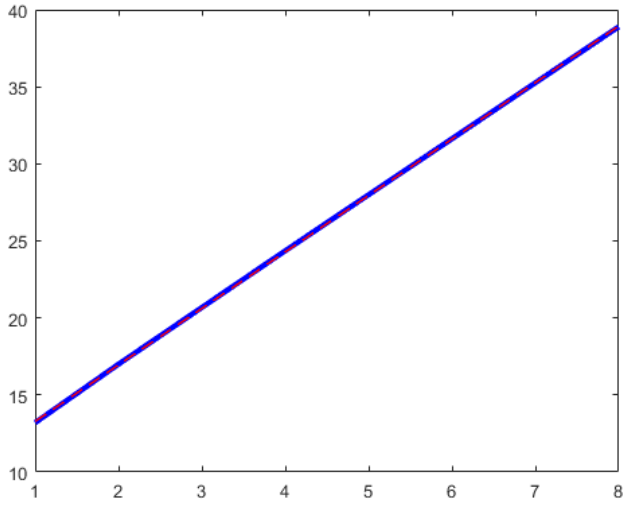


Figura 3.8: Rappresentazione grafica dell'andamento del tempo di esecuzione per l'estrazione del solo CPPS con VHM per numero di frame crescentee. La linea tratteggiata rappresenta l'approssimazione calcolata.

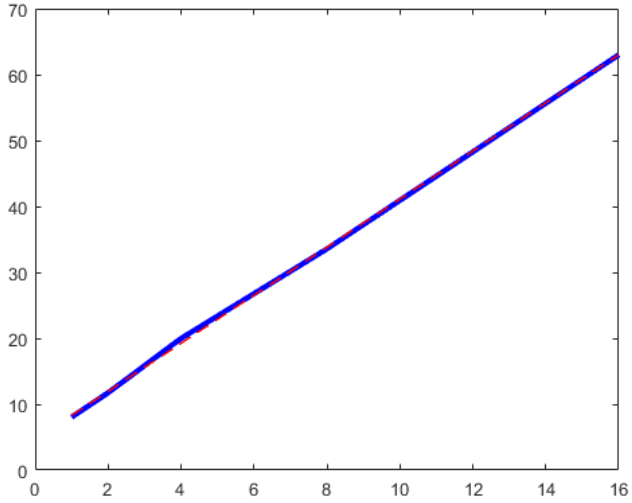


Tabella 3.35: Tempi per calcolo distribuzione CPPS

Numero frame	Tempo (s)
1 frame	8.04
2 frame	11.734
4 frame	19.978
8 frame	33.56
16 frame	63

a quelli analizzati servendosi di apposite funzioni matematiche per l'approssimazione senza la necessità di effettuare il test con file di dimensioni elevate.

3.5.3 Estrazione parametri su 256 frame

In conclusione, è stata ancora effettuato un altro tipo di analisi sul VHM: si è reiterato il processo di estrazione della distribuzione del CPPS utilizzando una singola finestra composta da 265 frames (corrispondenti a circa 12 secondi,ovvero la totalità del file audio registrato). Così facendo si ottiene un singolo valore di CPPS ogni 12 secondi (5 ogni minuto) il quale non permette, ovviamente, una stima della distribuzione nel breve periodo ma può risultare utile con file di durate maggiori dove non è necessario avere un' elevata risoluzione nei calcoli: ad esempio utilizzando file da 10 minuti si possono avere 50 valori di CPPS i quali permettono di avere una distribuzione sufficientemente accurata.
Quello che si è visto reiterando il procedimento è quanto si riporta in Tabella 3.36.

Tabella 3.36: Tempi per calcolo distribuzione CPPS su 256 frame

Misurazione	Tempo (s)
1	4.394
2	4.441
3	4.388
4	4.488
5	4.379
Media	4.418

Sono state effettuate più misurazioni per ricercare una variazione nel tempo impiegato ma, come si può vedere dalla tabella, il tempo impiegato è quasi costante e non varia se si utilizza un file sempre sopra soglia o uno con degli istanti sotto soglia. Nella tabella sono riportate 5 misurazioni effettuate per estrarre il singolo valore di CPPS su 256 frames. Il file utilizzato per il test è sia del tipo ”sopra soglia” sia del tipo ”sotto soglia”. L’ultima riga della tabella riporta l’indicazione del tempo medio necessario per l’analisi, un tempo che è, comunque, piuttosto inferiore alla durata stessa del file.

CAPITOLO 4

Risultati

Dopo aver analizzato tutti gli algoritmi utilizzando il metodo descritto nel Capitolo 3 e avendo ottenuto i risultati ivi descritti si è potuta avere una panoramica più precisa del comportamento degli stessi al variare delle features utilizzate e dei dataset in esame.
La divisione in dataset *gender-based* è stata fatta dopo un ragionamento che alla base aveva l'assunto che maschi e femmine hanno differenti componenti in frequenza per quanto riguarda il segnale vocale. Si è quindi deciso di andare a sviluppare un analisi parallela al fine di ridurre l'errore introdotto da queste differenze. I risultati ottenuti indicano che il procedimento seguito può portare a buoni risultati soprattutto in fase applicativa in quanto permette di scegliere predittori più correlati e di avere quindi maggiori accuratezze rispetto all'utilizzare un dataset composto indistintamente da donne o uomini. Oltre a questo permette anche di ridurre la complessità computazionale (e quindi il tempo impiegato) per l'estrazione delle features dai file audio perchè, come viene esposto in modo più dettagliato all'interno delle sezioni successive, si può fare a meno dei parametri legati al CPPS senza rinunciare ad una buona accuratezza durante la predizione.

All'interno di questo capitolo si andranno ad esporre tutte le considerazioni effettuate in merito all'accuratezza dei modelli di Machine Learning facendo per ogni dataset, un'analisi approfondita di quelle che sono le principali variazioni e di quelli che sono i comportamenti dei modelli paragonandoli tra di loro. La parte finale del capitolo sarà, poi, dedicata ad un excursus sulla complessità computazionale (*time complexity*) dei modelli scelti e degli algoritmi di estrazione dei parametri essendo il fine ultimo di questa ricerca l'implementazione degli algoritmi in sistemi *embedded* o *standalone* i quali avranno, comunque, differenti prestazioni e capacità. Per fare le osservazioni sulle complessità computazionali degli algoritmi verrà utilizzata la notazione matematica degli $O(..)$ (O grande) la quale permette di definire dei "limiti massimi", degli asintoti verso cui la complessità computazionale tende ma non va oltre; si andrà, perciò, a cercare di stimare il caso peggiore a cui ci si potrà trovare davanti.

4.1 Analisi delle accuratezze

Utilizzando nuovamente la triplice identificazione dei dataset sulla base del sesso dei componenti sono stati trovati i modelli che permettono di raggiungere meglio gli obiettivi che ci si era prefissati in partenza. Di seguito vengono riportati i risultati ottenuti per i tre gruppi di dati analizzati

cercando, per ogni insieme, di mettere in luce i punti di forza e i punti di debolezza di ciascun modello testato avvalendosi di quanto descritto nel Capitolo 3. In particolare, vengono messi a confronto li ”andamenti medi” delle accuratezze ottenute riducendo le feature per correlazione con la forma della voce o per correlazione reciproca. Verranno, altresì, utilizzati alcuni parametri di uso comune per chiarire le differenze tra i dati ottenuti.

4.1.1 Dataset complessivo

Facendo riferimento al grafico contenuto in Figura 3.1 si può notare che l'andamento generale dell'accuratezza è quello di rimanere tendenzialmente costante al decrescere del numero di predittori fino a che si utilizzano 3 parametri per il training. Superata questa soglia, guardando l'andamento della media, si vede che c'è un decadimento della precisione.
La stessa considerazione può essere fatta sul grafico di Figura 3.2 dove si ha, però, un comportamento leggermente differente. L'andamento medio è costante fino a 4 predittori utilizzati. Da 3 in poi si ha un abbassamento marcato dell'accuratezza media che si attesta al 59,39 % con l'utilizzo di una singola feature (risultato ottenuto anche nell'analisi riportata in Figura 3.2.

Tra tutti i modelli utilizzati per l'analisi spicca il comportamento dell'algoritmo Fine KNN il quale restituisce un' accuratezza costante pari al 100%. Un andamento piuttosto singolare se si pensa alla causa di tale risultato. Come già accennato nel Capitolo 3 un' accuratezza del 100% è, solitamente, dovuta ad un problema di overfitting dei dati; ci si può aspettare la scomparsa di questo problema (o l'insorgere di underfitting) utilizzando un numero minimo di features per la predizione (1 o 2) tuttavia, per questo algoritmo, il problema non scompare ma rimane ben presente per qualsiasi numero di features si vada ad utilizzare. Possiamo ragionevolmente supporre che il modello Fine KNN non sia in grado di trovare un pattern generico nei dati utilizzati per il training e che si lasci condizionare eccessivamente dal rumore in essi contenuto creando un modello troppo specifico per il dataset utilizzato [13]; questo è dovuto alla natura dell'algoritmo che, cercando di trovare somiglianze tra dati vicini tra loro, crea un modello di divisione troppo preciso e strettamente legato ai dati in esame. Per questi motivi viene scartato.
Se si utilizzano tutte i parametri a disposizione l'algoritmo che ha la migliore accuratezza è Cubic SVM (98,5 %), superiore del 3,1 % a RUSBoosted Trees (95,4 %). Gli altri modelli presi in esame restituiscono risultati inferiori al 90 % ma, ad esclusione di Quadratic Discriminant il quale fallisce

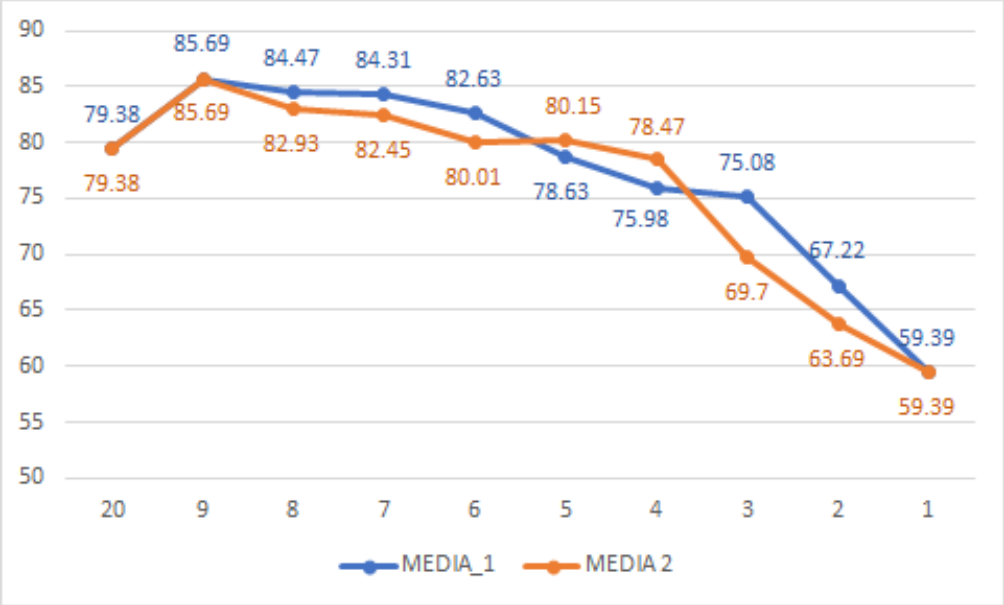
nel training, danno tutti dei risultati che possono ancora essere considerati buoni: il modello a minor esattezza è Linear Discriminant il quale, comunque, produce una precisione del 75,4 %
Selezionando le 9 features che sono state elette col metodo della correlazione con la forma della voce non si registra una variazione nell'accuratezza del modello Cubic SVM il quale si conferma come miglior scelta tra quelli presi in esame. Gli altri algoritmi hanno precisioni inferiori ma non si sono registrate sensibili variazioni rispetto all'utilizzo di tutte le features. Le uniche due eccezioni sono rappresentate da Linear Discriminant e Quadratic Discriminant. Nel primo caso l'accuratezza subisce un calo di più del 10 % mentre nel secondo caso si è passati da un fallimento in fase di training ad una precisione del 90 %.
Dopo questo primo drastico calo l'algoritmo Linear Discriminant continua a mostrare un andamento fortemente negativo e per questo motivo viene escluso dalla valutazione finale così come Quadratic Discriminant che, dopo il picco iniziale al 90 % ad ogni rimozione di una feature perde il 10 % in precisione. La causa delle scarse prestazioni di questi due modelli, basati entrambi sulla Discriminant Analysis, risiede nel fatto che l'algoritmo non è in grado di individuare una varianza appropriata tra le classi e non riesce quindi a costruire un modello adatto ad assegnare i campioni ai gruppi di appartenenza.
Per quanto riguarda l'algoritmo Medium Gaussian SVM, fino a 7 features utilizzate non si registrano sensibili variazioni di accuratezza; superata questa soglia si ha un sensibile calo di precisione dovuto alla natura più grossolana della divisione in classi. Con questo algoritmo viene impostata una dimensione di kernel pari a $\sqrt{P}$, dove P è il numero di predittori utilizzati per il training. La divisione in classi risulta, quindi, meno precisa perché un kernel più grande significa una finestra maggiore all'interno di cui far ricadere i campioni. Superata la soglia di 6 features utilizzate è l'accuratezza del modello Quadratic SVM che comincia a mostrare un andamento fortemente negativo. Anche per questo algoritmo la causa della scarsa accuratezza è da ricercare nel metodo utilizzato per la separazione tra classi.
Tra tutti gli algoritmi considerati nel training si va a scegliere quelli che conservano la più alta accuratezza nonostante il basso numero di features utilizzate. Di particolare rilevanza è l'algoritmo **Fine Gaussian SVM** il quale fino a 2 features utilizzate mantiene un'accuratezza pari o superiore al 90% dimostrandosi come il modello migliore, anche più performante di algoritmi di tipo Ensemble e di algoritmi basati sullo stesso principio dei Support Vector (Cubic SVM). Utiliz-

zando un algoritmo SVM con kernel di tipo Fine Gaussian Matlab imposta automaticamente una dimensione di kernel pari a $\sqrt{P}/4$ [14], permettendo, così, una precisa separazione tra le classi da identificare. Un altro grande vantaggio intrinseco nell'utilizzo di algoritmi di tipo SVM è l'introduzione del Kernel il quale permette di rimappare dati in più dimensioni in due dimensioni così da avere una migliore gestione della divisione. Questo va a discapito della semplicità di interpretazione del modello e dei suoi parametri e della complessità computazionale del modello stesso.
Effettuando un ulteriore step di riduzione del numero di parametri usati per la predizione si arriva ad utilizzarne uno soltanto (Unv_{std}) e, in questo caso, la precisione di Fine Gaussian SVM subisce un calo del 30 % circa attestandosi sul 67,7%, valore condiviso con Fine Tree. Con l'utilizzo di una singola feature l'algoritmo che restituisce la miglior accuratezza è RUSBoosted Trees pari a 83,1 %. Questo risultato ottenuto è legato alla natura di questo algoritmo di tipo Ensemble: il principio su cui si basa è quello del Random Under Sampling (RUS) attraverso cui si estraggono campioni casuali dalle classi utilizzate. Questo tipo di sottocampionamento è molto utile nel caso di dataset sbilanciati perché permette di restituire un buon grado di bilanciamento tuttavia, nel nostro caso, per ogni classe sono presenti lo stesso numero di campioni per cui questo algoritmo non si trova a risolvere alcun problema di classi non bilanciate. Agisce effettuando un campionamento e poi stimando l'accuratezza su più alberi per poi combinarne i risultati ed è qui che risiede la potenzialità di questo modello. Utilizzando degli alberi la complessità computazionale si abbassa e si ha un guadagno anche in semplicità di implementazione dell'algoritmo.

Queste considerazioni sono state fatte analizzando la variazione delle accuratezze al decrescere del numero di features scegliendo quelle da eliminare sulla base della loro correlazione con la forma vocale, tuttavia, si possono fare le stesse considerazioni per la rimozione basata sulla correlazione reciproca tra le features. Come si nota da Figura 3.2 l'andamento quasi-costante di algoritmi come Cubic SVM, RUSBoosted Trees e Fine Gaussian SVM è presente anche in questo caso. Una delle principali differenze, però, è che utilizzando due sole features ($Cpps_{std}$ e Unv_{std}) l'accuratezza degli algoritmi SVM è inferiore a quella ottenuta con lo stesso numero di features nel caso precedente. In generale, si può notare una tendenza ad avere una precisione inferiore per la maggior parte degli algoritmi di cui si è eseguito il training con il metodo dell'eliminazione per correlazione reciproca.

Quello poc'anzi descritto è un risultato che ci si poteva aspettare già nelle fasi preliminari alla reiterazione del training dei vari algoritmi in quanto nella rimozione delle features non si tiene più conto esclusivamente della correlazione con la forma della voce ma principalmente della correlazione reciproca tra i parametri al fine di evitare di avere parametri "inutili" che, cioè, non aggiungono valore alla classificazione. Le features scelte come punto di partenza sono quelle a correlazione massima con la forma vocale per cui si ha comunque un buon grado di accuratezza, quello che veramente influisce sulla precisione è come quest'ultime vengono combinate tra loro.

Figura 4.1: Confronto tra medie degli andamenti - Dataset complessivo. Media 1: media accuratezze con rimozione features per correlazione con forma vocale. Media 2: Media accuratezza con rimozione feature per correlazione reciproca



In Figura 4.1 è riportato un confronto tra le accuratezze medie ottenute nei due test effettuati cambiando la modalità di rimozione dei parametri utilizzati. Si vede che, nella maggioranza delle iterazioni, si hanno risultati migliori andando a considerare la correlazione tra parametri e forma della voce. Solo nel caso di 5 e 4 features utilizzate si ha una maggior accuratezza se si utilizzano quelle a minor correlazione reciproca. Nel caso di correlazione con la forma della voce si utilizzano i seguenti parametri:

- **5 features**: utilizzate $Voiced_{mean}$, $Voiced_{std}$, Unv_{mean}, Unv_{std}, $f0_{mean}$

- **4 features**: utilizzate $Voiced_{mean}$, $Voiced_{std}$, Unv_{std}, $f0_{mean}$

Nel caso, invece, di correlazione reciproca tra features i parametri utilizzati sono:

- **5 features**: utilizzate $Cpps_{std}$, $Cpps_5$, Unv_{std}, $f0_{mean}$, $f0_{std}$
- **4 features**: utilizzate $Cpps_{std}$, $Cpps_5$, Unv_{std}, $f0_{std}$

Per un'ulteriore valutazione delle prestazioni degli algoritmi si introduce il parametro Δ_{Acc} definito da

$$\Delta_{Acc} = \frac{\sum_{i=1}^{n}(Acc_1 - Acc_2)}{n} \tag{4.1}$$

dove n indica il numero delle iterazioni/osservazioni e Acc_1 e Acc_2 indicano, rispettivamente, le accuratezze medie ottenute nei test 1 e 2 (1=correlazione con forma della voce, 2=correlazione reciproca).
Da questo parametro si ottiene la seguente deduzione

$$\Delta_{Acc} = \begin{cases} > 0 & \text{Meglio utilizzare la correlazione con la forma vocale} \\ < 0 & \text{Meglio utilizzare la correlazione reciproca} \end{cases} \tag{4.2}$$

Nel caso del gruppo di dati senza divisione sulla base del sesso utilizzando 4.1 si ottiene $\Delta_{Acc} = 1.092$ e, facendo riferimento a 4.2, si deduce che, in generale, è meglio considerare una riduzione delle features basandosi sulla correlazione tra quest'ultime e la forma della voce.

Basandosi ancora sul grafico di Figura 4.1 si può vedere come il miglior risultato medio si ottenga utilizzando 9 feature per la predizione e, fino all'utilizzo di 3 feature, si conserva una buona accuratezza media potendo, però, ridurre i predittori utilizzati in entrambi i casi di riduzione.

4.1.2 Dataset maschile

Nei grafici di Figura 3.3 e 3.4 sono riportati gli andamenti dell'accuratezza ottenuta eseguendo il training degli algoritmi sul dataset composto da soggetti esclusivamente di sesso maschile.

Guardando all'andamento generale si nota come per la maggioranza degli algoritmi ci sia un netto decadimento delle prestazioni al decrescere del numero di features utilizzate. Lo si può vedere anche dall'andamento della media, la quale subisce una diminuzione di circa 30 punti percentuali dall'inizio alla fine delle reiterazioni. Questo è un risultato che, rispetto al dataset complessivo, è più in linea con quanto è logico supporre accada con la progressiva rimozione delle features utilizzate per la predizione anche alla luce della minor presenza d campioni nell'insieme. L'unica eccezione è rappresentata da Quadratic SVM il quale mantiene un andamento pressoché costante nelle prime iterazioni (fino a 4 features) per poi decrescere ed attestarsi sul 48,6 % utilizzando un solo parametro per la classificazione.

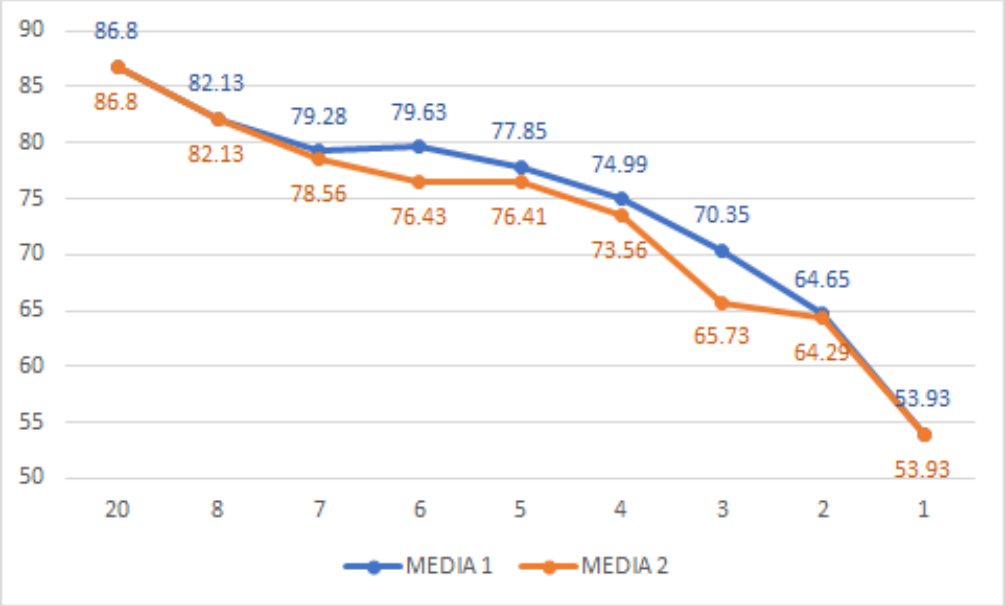
Se si vanno ad utilizzare tutti i parametri disponibili si ha una buona accuratezza generale, escluso il 100% ottenuto con Medium Gaussian SVM indicatore di overfitting. In questo caso i modelli migliori sono 2: Quadratic SVM e Linear Discriminant i quali condividono la stessa precisione del 94,3 %.
Come detto in precedenza riducendo il numero di parametri utilizzati la precisione di Quadratic SVM rimane costante fino a 4 features ($Voiced_{mean}$, $Voiced_{std}$, Unv_{std}, $F0_{mean}$ nel caso di correlazione con la forma della voce. $Cpps_{5prc}$, $Voiced_{mean}$, Unv_{std}, $F0_{mean}$ nel caso di correlazione reciproca tra features) restando nell'ordine del 90 %. Superata tale soglia e arrivando a 3 features (viene tolta Unv_{std} in ambedue i test) l'accuratezza ne risente notevolmente perdendo 10 punti percentuali per ogni feature rimossa chiudendo con un punteggio di 48,6 %. Nonostante questo degradamento prestazionale, Quadratic SVM rimane comunque il modello che fornisce i migliori risultati. Questo è possibile grazie alla scelta del kernel di tipo quadratico che permette di meglio approssimare la divisione in classi dei dati.

Riducendo le features sulla base della loro correlazione reciproca l'algoritmo mostra un andamento comparabile a quello ottenuto riducendo per correlazione con la forma della voce mostrando dei tratti costanti fino a 4 features seguiti da un rapido decadimento delle prestazioni se si utilizza un minor numero di parametri.
In ambedue i casi di rimozione delle features Medium Gaussian SVM restituisce valori di accuratezza lontani dall'essere costanti restituendo, però, valori accettabili fino all'utilizzo di 3 sole features (oltre il 70 % in entrambi i test).

Gli algoritmi di tipo SVM restituiscono una migliore accuratezza generale ma sono anche quelli a maggior costo computazionale; per tale motivo si va a considerare anche il comportamento degli algoritmi Fine Tree e Medium Tree (che in entrambi i casi hanno le stesse accuratezze ad ogni iterazione). Come per gli altri modelli in esame, anche queste due tipologie di albero conservano una buona accuratezza fino ad utilizzare 4 features (superiore al 70 % in entrambi i test fatti) dimostrandosi una valida alternativa ai modelli SVM soprattutto al fine di ridurre la complessità richiesta dal processo di classificazione dei dati.
Tra gli algoritmi di tipo SVM, nonostante la considerazione fatta precedentemente sulla loro maggior accuratezza, sono da notare i risultati inferiori agli altri ottenuti con Linear SVM. Questo algoritmo, basandosi su una divisione di tipo lineare, non è in grado di fornire un' adeguata divisione in classi dei dati non riuscendo a gestire il fatto che ci siano 5 categorie. È meglio ricorrere ad algoritmi che utilizzano Kernel più sofisticati riuscendo quindi ad individuare (e modellare) meglio le divisioni tra classi.

Il modello che fa ottenere i risultati peggiori è Subspace Discriminant, un modello di tipo Ensemble basato sulla Discriminant Analysis. Le cause di tale fallimento sono da ricercare nel modo in cui l'algoritmo lavora: per ogni iterazione sceglie un set di *m* predittori da utilizzare (con m definito a priori), esegue il training di un modello (in questo caso basato su Discriminant Analysis) usando quei predittori, ripete questi due step fino a ad avere *n* modelli addestrati ed infine esegue la classificazione basandosi sui punteggi medi ottenuti nelle fasi precedenti. Dal grafico dell'andamento si nota come il decadimento dell'accuratezza di Subspace Discriminant approssimi quello di Linear Discriminant in entrambi i test effettuati: è facile capire che se per l'algoritmo di tipo ensemble vengono utilizzati meno predittori e come "learner" viene usato Linear Discriminant se riduco il numero di features perché le campiono (subspace) l'accuratezza massima che posso ottenere è molto vicina a quella ottenuta con Linear Discriminant per poche features.

Figura 4.2: Confronto tra medie degli andamenti - Dataset maschile. Media 1: media accuratezze con rimozione features per correlazione con forma vocale. Media 2: Media accuratezza con rimozione feature per correlazione reciproca



Nel grafico di Figura 4.2 si può vedere un confronto tra l'accuratezza media ottenuta rimuovendo le features secondo la loro correlazione con la forma della voce (Media 1) e l'accuratezza ottenuta rimuovendole sulla base della loro correlazione reciproca (Media 2). La differenza media di accuratezza è pari a $\Delta_{Acc_{maschi}} \cong 1.27$; questo risultato, alla luce di quanto scritto in 4.2 indica che è preferibile scegliere i parametri sulla base della loro correlazione con la forma della voce al fine di ottenere una maggior precisione nella predizione. Per avere il miglior risultato medio possibile, come si vede dal grafico in Figura 4.2, conviene utilizzare 8 parametri (cioè tutti quelli maggiormente correlati) oppure 6 se si vuole ridurre al minimo il numero di predittori utilizzati. Per avere una buona accuratezza media si deve fare la selezione dei parametri basandosi sulla loro correlazione con la forma della voce.

4.1.3 Dataset femminile

In questa sezione si fa riferimento ai grafici di Figura 3.5 e 3.6 all'interno di cui sono riportate le variazioni delle accuratezze ottenute con i diversi algoritmi evidenziati come migliori nel Capitolo 3. Come per i gruppi di dati analizzati in precedenza (dataset completo e dataset maschile) si nota la tendenza, per l'accuratezza media, a decrescere col diminuire del numero di parametri usati per la predizione. Tuttavia, rispetto ai casi precedenti la diminuzione è meno marcata e nel tratto compreso tra 9 e 5 parametri si ha un comportamento medio quasi-costante.

In ambedue i test, effettuati cambiando la modalità di selezione dei predittori, spiccano due comportamenti particolari: quello di Fine Gaussian SVM e quello di RUSBoosted Trees. Nel primo caso il modello basato su SVM soffre in modo evidente di overfitting fino all'utilizzo di 3 features restituendo un'accuratezza costante pari al 100 %. Questo risultato è con buona probabilità legato alla precisione del modello, eccessiva per il dataset in esame dove non ci sono sufficienti campioni per un training completamente affidabile. Nel caso di Fine Gaussian SVM viene impostato un Kernel pari a $\sqrt{P}/4$ (dove P è il numero dei predittori) che è la scala minima tra quelle proposte dall' App Classification Learner per i Kernel degli algoritmi SVM.
Per quanto riguarda RUSBoosted Trees mantiene un accuratezza costante pari al 20 % fino a 2 features (3 per l'analisi della correlazione reciproca tra features) per poi avere un picco al 90 % (96,7 % nell'altro caso). Da un lato indica un' incapacità di distinguere le classi se si utilizza un ampio numero di features mentre dall'altro indica una tendenza all'overfitting se si utilizza un basso numero di parametri. Nonostante la buona accuratezza restituita con un basso numero di features non è un risultato da prendere troppo in considerazione perché per il tipo di dataset che si è analizzato questo non è l'algoritmo adatto in quanto è stato ideato per risolvere problemi che nascono con dataset aventi classi sbilanciate. Data la compsizione dei dati utilizzati in questa ricerca non si può affermare con sicurezza che sia un buon modello, per tale motivo viene scartato.

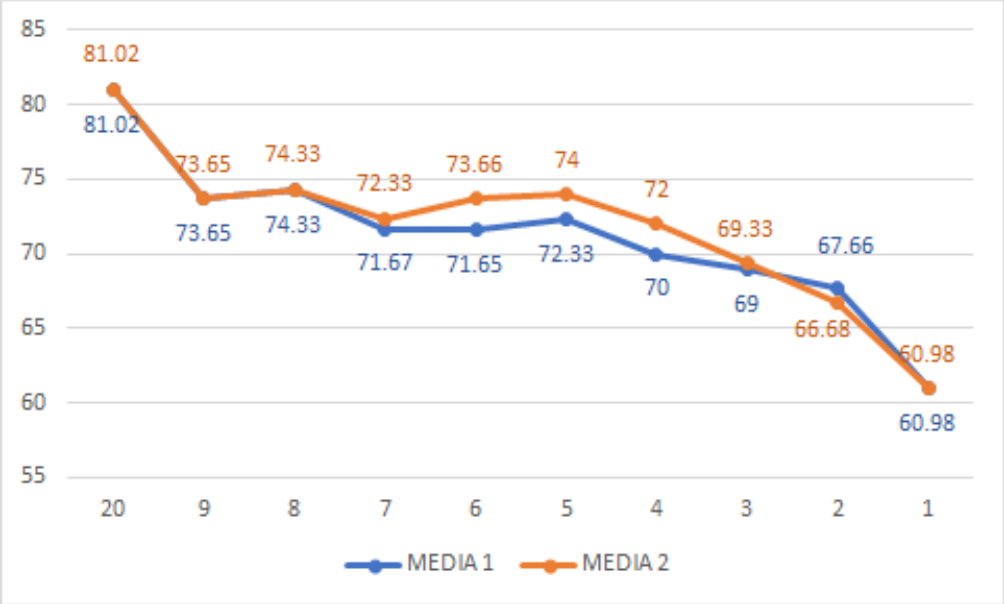
Come già analizzato per il dataset maschile, anche per quello femminile si vede un andamento confrontabile tra i modelli Linear Discriminat e Subspace Discriminant. In ambedue i test effettuati Linear Discriminant restituisce una accuratezza inferiore rispetto alla media generale e, quindi, Subspace Discriminant segue questo trend essendo basato sugli stessi principi. Tra i due modelli

citati si osserva, però, una sostanziale differenza nel training fatto con una sola feature; Linear Discriminant fa registrare una precisione del 73, 3 % mentre l'altro si attesta al 50 %. Se si sceglie di utilizzare un solo predittore la scelta migliore è Linear Discriminant, perché RUSBoosted Trees viene escluso. Resta comunque una scelta poco consigliabile quella di utilizzare un solo predittore.

Nel caso in cui si scelgano di utilizzare tutti i predittori per l'addestramento dei modelli si deve far fronte a una generale tendenza all'overfitting da parte degli algoritmi utilizzati (la precisione media, escludendo RUSBoosted Trees, è del 90 %). Come nei dataset precedenti, questo risultato era attendibile dato il basso numero di campioni presenti nel gruppo e l'alto quantitativo di parametri per la loro classificazione: in questa situazione è facile avere dei modelli che approssimano con eccessiva precisione la struttura dei dati e sono, di conseguenza, poco scalabili su dati simili ma con strutture differenti.

Passando ad analizzare il comportamento che si ottiene con le feature più correlate alla forma vocale sia nel caso di 9 parametri che in quello in cui se ne utilizzano 8 si vede che i modelli basati su SVM restituiscono le migliori accuratezze con risultati molto sopra la media generale. Questo è un buon indicatore anche se è sintomo di un possibile overfitting del modello utilizzato; non si può escludere questa possibilità data la mancanza di un metodo di validazione degli algoritmi utilizzati.

Figura 4.3: Confronto tra medie degli andamenti - Dataset femminile. Media 1: media accuratezze con rimozione features per correlazione con forma vocale. Media 2: Media accuratezza con rimozione feature per correlazione reciproca



In Figura 4.3, come nei casi precedenti, sono riportate le medie degli andamenti. Diversamente dalle due analisi effettuate in precedenza, nel caso del dataset femminile la migliore accuratezza media al variare del numero di parametri utilizzati si ottiene andando a rimuovere le features sulla base della loro correlazione reciproca. In questo caso, infatti, utilizzando 4.1 si ottiene $\Delta_{Acc_{femm}} \cong -0.575$ e servendosi di 4.2 si arriva alla conclusione sopra indicata e cioè che è tendenzialmente meglio scegliere una rimozione basata su correlazione reciproca partendo da una prima selezione basata esclusivamente sulla correlazione tra predittori e forma vocale.

Utilizzando ancora il grafico di Figura 4.3 si nota come il miglior risultato medio, escludendo quello ottenuto servendosi di tutti i parametri, si ottiene con 8 features ma se si vuole ridurre al minimo il numero di predittori usati si può scendere fino a 2 features utilizzate. Per tale valore, scegliendo i parametri sulla base della loro correlazione reciproca, si conserva un'accuratezza media del 70 %.

4.1.4 Considerazioni sulla divisione del dataset

In partenza ci si era prefissati di dividere il dataset in uomini e donne per analizzare se ci fossero degli effettivi vantaggi nel farlo. Dopo aver condotto la ricerca nella sua interezza si possono mettere a confronto le accuratezze ottenute per gli algoritmi comuni ai tre dataset cioè quelli reputati migliori ad ogni analisi; tali algoritmi sono cinque e sono: Medium tree, Quadratic SVM, Medium Gaussian SVM, Linear Discriminant e Fine Tree. Nei grafici riportati in Figura 4.4 per ogni modello è rappresentato l'andamento dell'accuratezza ottenuta analizzando i dati dei 3 dataset.

Si può notare come, a livello generale, la scelta di dividere il dataset principale sulla base del sesso degli attori che hanno eseguito le registrazioni, si è rivelato essere un buon metodo. A parità di numero di feature (non a parità di tipologia, però) dividere il dataset complessivo ed applicare una scelta dei parametri differente per gli uomini e per le donne permette di ottenere un incremento di accuratezza rispetto allo sceglierli senza dividere il gruppo di dati. Gli unici due casi in cui mantenere unito il dataset può essere una buona idea sono gli algoritmi Medium Tree e Fine Tree (che hanno andamenti confrontabili tra loro). Come si vede, sempre da Figura 4.4, per un numero minore di feature utilizzate si ha una migliore accuratezza se si utilizza il dataset nella versione "completa", comportamento che non è uguale per gli altri tre modelli confrontati.

Figura 4.4: Confronto tra le accuratezze restituite dai modelli comuni. Linea blu: dataset complessivo. Linea grigia: dataset femminile. Linea arancio: dataset maschile. Da sinistra: Medium tree, Quadratic SVM, Medium Gaussian SVM, Linear Discriminant, Fine Tree.

4.2 Complessità computazionale

4.2.1 Test condotti su computer

Nel capitolo 3 sono mostrati i metodi utilizzati per la stima della complessità computazionale ed i relativi risultati ottenuti. Nell'analizzare la complessità computazionale generale degli algoritmi si è tenuto conto della proprietà secondo cui, dati $T_1(n)$ e $T_2(n)$ aventi, rispettivamente, complessità computazionale pari a $O(f(n))$ e $O(g(n))$ si ha che

$$T_1(n) + T_2(n) = O(max(f(n), g(n)) \tag{4.3}$$

Sfruttando questa proprietà si è potuti arrivare ad una formulazione della complessità computazionale nel *worst-case scenario* per ognuno degli algoritmi che si è andati ad analizzare.

In Tabella 4.1 sono riportati i risultati finali ottenuti per i diversi algoritmi testati. Per gli algoritmi che estraggono separatamente i parametri viene anche riportata la complessità finale unendo i 4 algoritmi secondo 4.3.

Tabella 4.1: Complessità computazionali finali degli algoritmi testati

Algoritmo	Complessità computazionale	Totale
CPPS.m	$O(n^3)$	$O(n^3)$
f0_rms.m	$O(n)$	$O(n^2)$
VoiceUnvoice.m	$O(n^2)$	
voicedp.m	$O(n)$	
Cpps_opt.m	$O(n^2)$	

La complessità computazionale totale dei 4 algoritmi che estraggono le fetaures in modo separato è inferiore rispetto all'algoritmo che le estrae tutte insieme all'interno dello stesso script. Questo, tuttavia, non è indice di migliori prestazioni dei 4 algoritmi come già visto considerando i tempi di esecuzione degli stessi.

Per la valutazione delle prestazioni di un algoritmo si devono tenere in conto svariati parametri e non esclusivamente la sua complessità computazionale. [15] Quanto un determinato software impiega a svolgere una serie di operazioni è legato non solo al quantitativo di operazioni da effettuare e alla loro richiesta di risorse, ma è anche strettamente correlato alle caratteristiche dell'hardware e del software su cui tali algoritmi vengono eseguiti. Per questo motivo i test sono stati effettuati su più macchine cercando di avere una stima generale della spesa.

In aggiunta ai parametri sopra indicati vengono introdotti degli indicatori più precisi della qualità di un algoritmo i quali permettono di meglio identificare le prestazioni generali sia della macchina sia del codice che viene eseguito su di essa.
I parametri introdotti, con relativa formulazione, sono [15]:

$$Prestazioni = \frac{1}{T_{esecuzione}} \tag{4.4}$$

$$T_{CPU} = \frac{Cicli\ \ di\ \ clock\ \ CPU}{F_{CPU}} \tag{4.5}$$

$$Cicli\ \ di\ \ clock\ \ CPU = Istruzioni\ \ programma \cdot CPI \tag{4.6}$$

Sostituendo 4.6 in 4.5 si ottiene la formula in 4.7

$$Tempo\ \ CPU = \frac{Numero\ \ istruzioni\ \ \cdot\ \ CPI}{F_{CPU}} \tag{4.7}$$

la quale permette di arrivare ad ottenere

$$CPI = \frac{F_{CPU} \cdot\ \ Tempo\ \ CPU}{Numero\ \ istruzioni} \tag{4.8}$$

Si aggiunge ancora un ulteriore parametro denominato MIPS definito come

$$MIPS = \frac{Numero\ \ istruzioni}{Tempo\ \ esecuzione \cdot 10^6} \tag{4.9}$$

Inserendo in 4.8, 4.9 e 4.4 i valori contenuti nelle tabelle della sezione Analisi della complessità computazionale del Capitolo 3 si sono ottenuti i valori presenti nelle tabelle 4.2 e 4.3. In queste tabelle sono riportati i risultati ottenuti considerando come un unico blocco i quattro script che estraggono i parametri in modo separato per avere un confronto migliore con lo script "unico". Per fare le analisi i valori di riferimento sono quelli medi sia per il tempo di esecuzione, per cui

si è utilizzato il valore medio tra le cinque forme vocali testate, sia per la frequenza di clock dei processori, per cui si è andati a considerare il valore medio tra la frequenza massima e la frequenza minima a cui la CPU opera essendo tutti i processori a frequenza variabile.
Per la Macchina 1 la frequenza è di 2.4 GHz (Min: 1.8 GHz - Max: 3.0 GHz). Per la Macchina 2 la frequenza è 2.2 GHz (Min: 1.7 GHz - Max: 2.7 GHz). Infine, per la macchina 3 la frequenza è 3.8 GHz (Min: 3.6 GHz - Max: 4.0 GHz)

Tabella 4.2: Parametri ottenuti per l'algoritmo CPPS.m

Macchina	CPI	MIPS	Tempo di esecuzione	Prestazioni
Macchina 1	1.26	$1.90 \cdot 10^{-}6$	0.115458	$8,66115$
Macchina 2	1.32	$1.67 \cdot 10^{-}6$	0.132001	$7,57570$
Macchina 3	1.35	$2.815 \cdot 10^{-}6$	0.078084	$12,80672$

Tabella 4.3: Parametri ottenuti per gli algoritmi separati

Macchina	CPI	MIPS	Tempo di esecuzione	Prestazioni
Macchina 1	313.38	$7.65 \cdot 10^{-}9$	2.57421	$0,38847$
Macchina 2	293.21	$7.50 \cdot 10^{-}9$	2.627493	$0,38059$
Macchina 3	307.72	$12.35 \cdot 10^{-}9$	1.596468	$0,62638$

I valori ottenuti durante questi test riguardanti la complessità computazionale permettono di confrontare, a parità di macchina, le prestazioni di un algoritmo rispetto ad un altro. Nei grafici delle figure da 4.5 a 4.8 si può vedere un confronto tra i parametri testati. Le barre di colore arancione rappresentano i dati ottenuti per l'insieme dei 4 algoritmi di estrazioni; le barre blu, invece, rappresentano i valori ottenuti per l'algoritmo di estrazione in blocco dei parametri. M1, m2 ed m3 indicano, rispettivamente, macchina 1, macchina 2 e macchina 3 le cui caratteristiche sono contenute nella tabella 3.14 del Capitolo 3.
Dato che gli algoritmi per l'estrazione separata non hanno alcuna ottimizzazione nell'utlizzo delle variabili e delle funzioni è chiaro che quest'ultimi siano meno performanti rispetto all'algoritmo

Figura 4.5: CPI per le tre macchine considerate (espressi in numero di cicli)

Figura 4.6: MIPS per le tre macchine considerate (espresso in milioni di istruzioni al secondo)

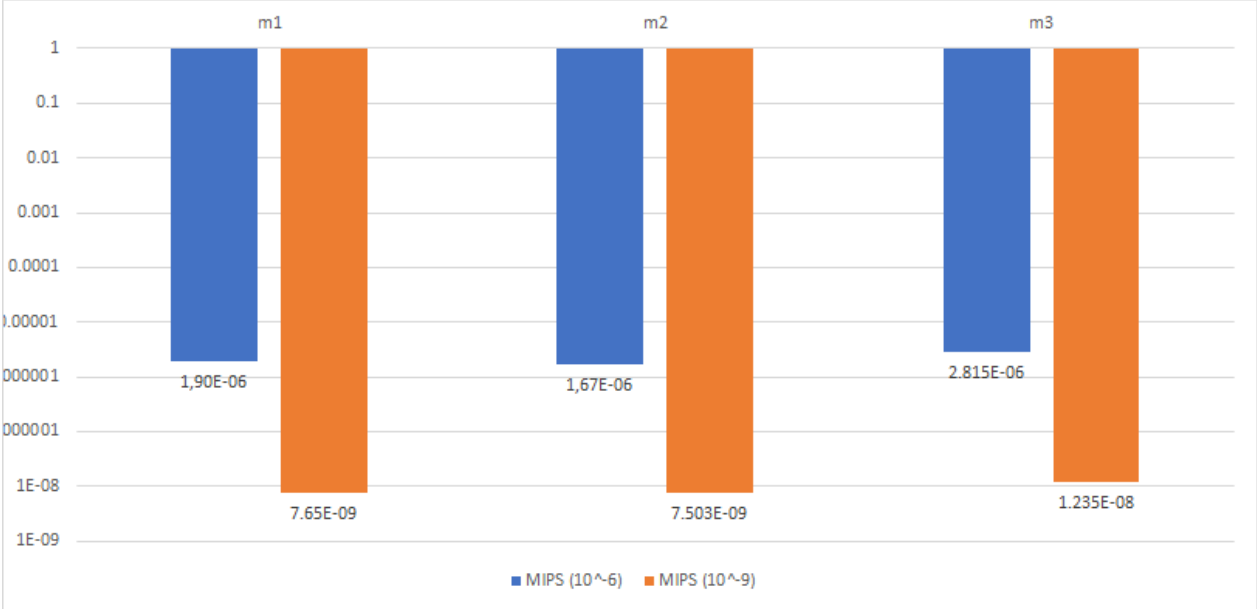


Figura 4.7: Tempi di esecuzione per le tre macchine considerate (espressi in secondi)

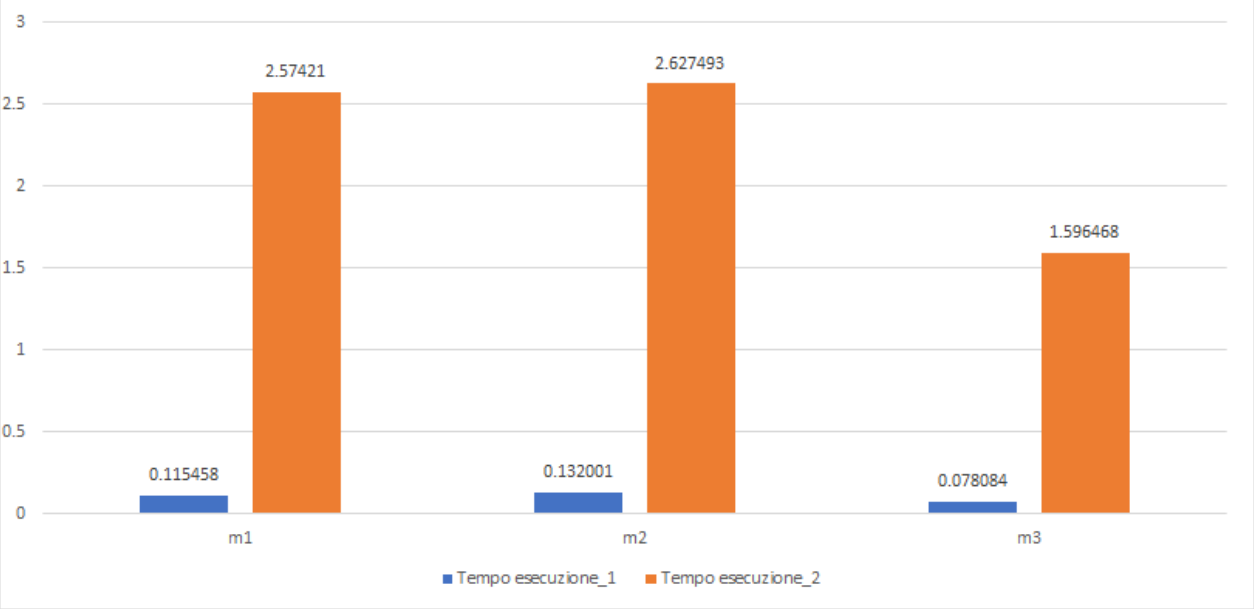
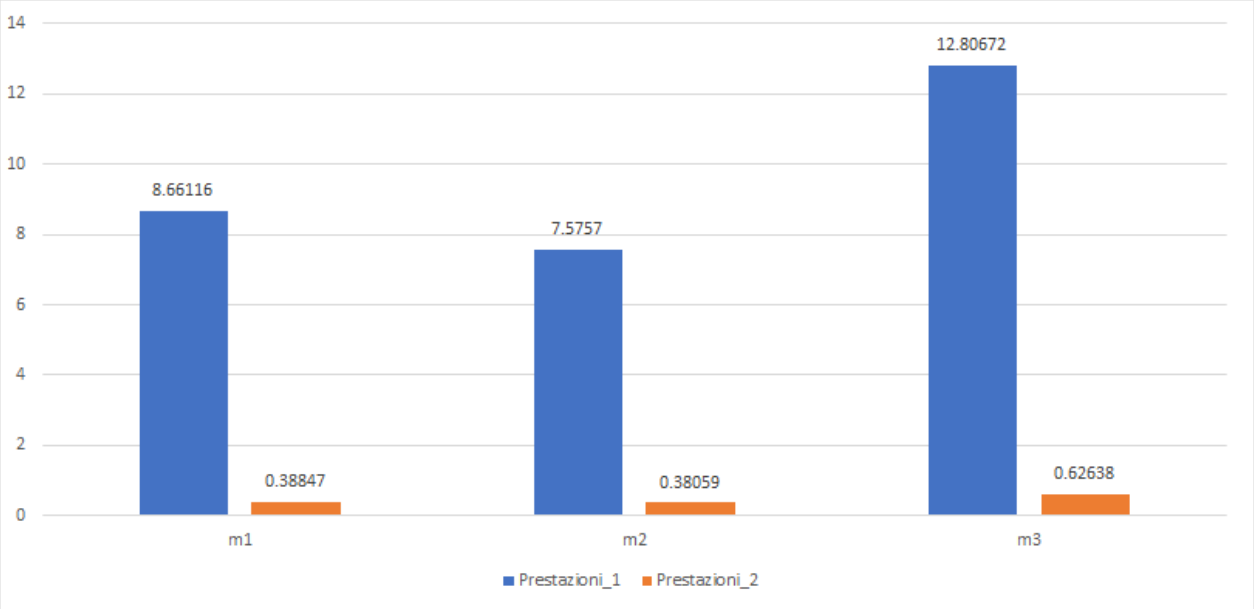


Figura 4.8: Indicatori delle prestazioni per le tre macchine considerate (valori espressi in 1/secondi)



unico di estrazione. Gli script sopra citati, infatti, derivano da precedenti lavori svolti aventi come punto centrale la ricerca ed estrazione dei parametri.

Per tutte le macchine considerate il valore denominato *Tempo di esecuzione* è minore nel caso del primo algoritmo producendo quindi un valore di *Prestazioni* maggiore (per il calcolo si utilizza 4.4). Il risultato ottenuto per il tempo di esecuzione si ripercuote anche sugli indicatori calcolati usando 4.9 e 4.8. In particolare i valori di CPI per CPPS.m (l'algoritmo che estrae in blocco le features) sono inferiori di circa due ordini di grandezza rispetto agli altri script confrontati, mentre quelli relativi alle "*Mega Instructions per second*" (MIPS) sono inferiori di tre ordini di grandezza rispetto a quelli ottenuti per gli altri algoritmi valutati. Considerando quanto detto sopra in merito ai parametri utilizzati è possibile affermare che l'algoritmo denominato CPPS.m sia, a livello prestazionale, di molto superiore all'unione degli altri 4 algoritmi confrontati con esso anche se presenta una complessità computazionale di un ordine di grandezza superiore ($O(N^3)$ contro $O(N^2)$)

Previsione dell'andamento di CPPS.m

Dato che, tra quelli per l'estrazione delle features, l'algoritmo a prestazioni maggiori è CPPS.m è stata ancora svolta un'ulteriore analisi volta ad individuare il comportamento dello script all'aumentare delle dimensioni del file analizzato.

La misurazione è stata effettuata utilizzando quella denominata come *Macchina 1* avvalendosi del software Matlab ™ e delle sue funzioni per il calcolo del tempo di esecuzione del codice. Il test è stato condotto su file di dimensioni crescenti pari a 1 secondo, 2 secondi, 4 secondi, 8 secondi, 16 secondi e 32 secondi con le stesse modalità utilizzate in precedenza: un test per ogni forma vocale e, successivamente, viene calcolata la media dei tempi di esecuzione.

Come nelle analisi precedenti, anche in questo caso i risultati ottenuti sono puramente indicativi in quanto non si può avere un'idea precisa del tempo di esecuzione di un determinato algoritmo in macchine che fanno largo uso di parallelismo e che non dedicano il processore esclusivamente alle operazioni scritte all'interno di esso. I dati riportati in Tabella 4.4 sono da intendersi come riferimento della variazione del tempo di esecuzione e non come un indicatore preciso del tempo necessario sulla macchina utilizzat. Come esposto nel Capitolo 3 bisogna, infatti, considerare le

operazioni non direttamente correlate allo script ma che occupano, comunque, il processore per del tempo.

Tabella 4.4: Tempo (espresso in secondi) necessario per estrazione delle features con file di diversa durata. Pt: piatta, Ton: tonda, Quad: quadra, Tri: triangolare, Spi: spigolosa

File	Pt	Ton	Quad	Tri	Spi	Media
1 secondo	0.124967	0.111714	0.111479	0.117128	0.112	0.115458
2 secondi	0.248564	0.179255	0.170376	0.175696	0.163302	0.187438
4 secondi	0.358568	0.341948	0.396777	0.331967	0.344787	0.354809
8 secondi	0.50923	0.555772	0.507663	0.683239	0.513498	0.55388
16 secondi	0.844894	0.797994	0.826005	0.814349	0.798211	0.816291
32 secondi	1.46081	1.460751	1.470222	1.445704	1.453114	1.45812

Dal grafico di Figura 4.10 si può notare come l'andamento del tempo di esecuzione (linea blu) sia lineare al crescere delle dimensioni del file analizzato, nonostante la prima leggerissima inflessione. Questo tipo di andamento permette di trovare, attraverso apposite funzioni all'interno di Matlab ™, i coefficienti che definiscono la retta che meglio approssima il grafico in figura. Utilizzando, quindi, la funzione *polyfit()* si ottengono i due coefficienti che, sostituiti ad m e q nella forma esplicita della retta $y = mx + q$, permettono di arrivare all'equazione della retta approssimante l'andamento del tempo di esecuzione. In particolare, i coefficienti trovati sono $m = 0.0419$ e $q = 0.1413$ che sostituiti nell'equazione generale permettono di arrivare a

$$y = 0.0419x + 0.1413 \tag{4.10}$$

Utilizzando 4.10 si può tracciare l'approssimazione dell'andamento del tempo di esecuzione necessario allo script CPPS.m per estrarre i parametri dai file in ingresso. In Figura 4.10 è rappresentato l'andamento dell'approssimazione per valori che vanno da 1 secondo a 256 secondi; sull'asse delle ascisse trovano posto i secondi di durata dei file mentre sull'asse delle ordinate è riportato il tempo stimato impiegato per l'estrazione. In Figura 4.9 si vede il confronto tra l'andamento reale e quello approssimato per i valori da 1 secondo fino a 32 secondi.

Figura 4.9: Confronto tra andamento reale (blu) e approssimazione (arancio) dello script CPPS.m

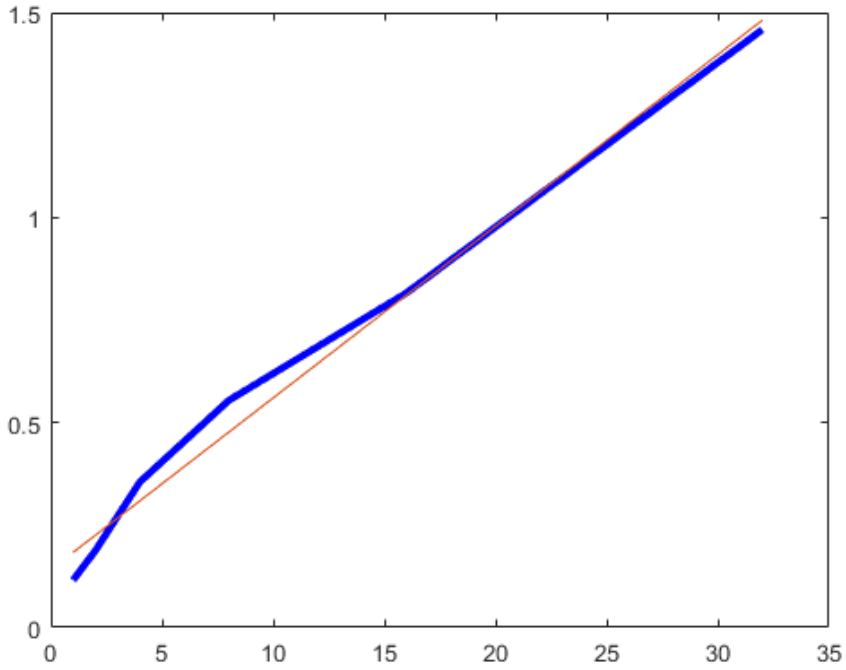
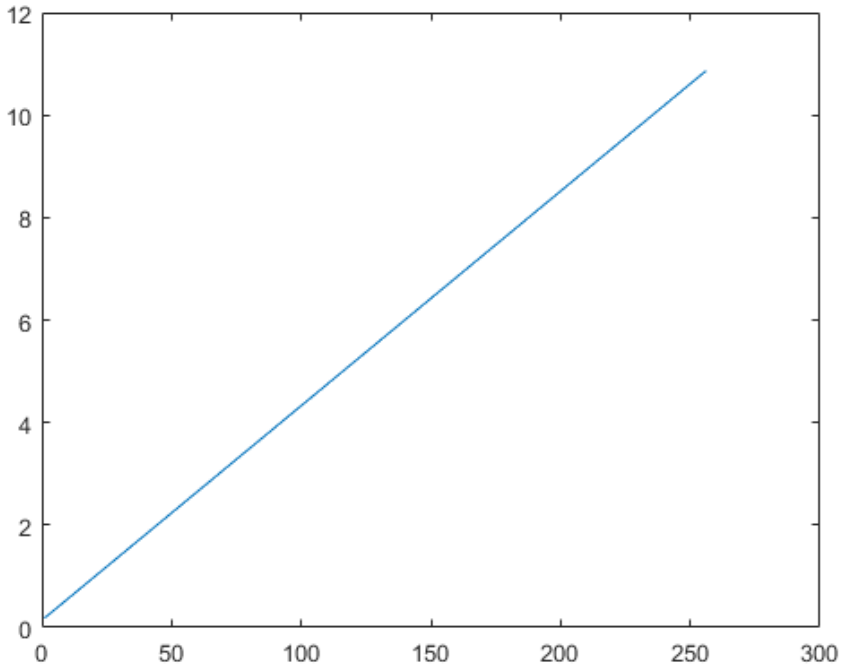


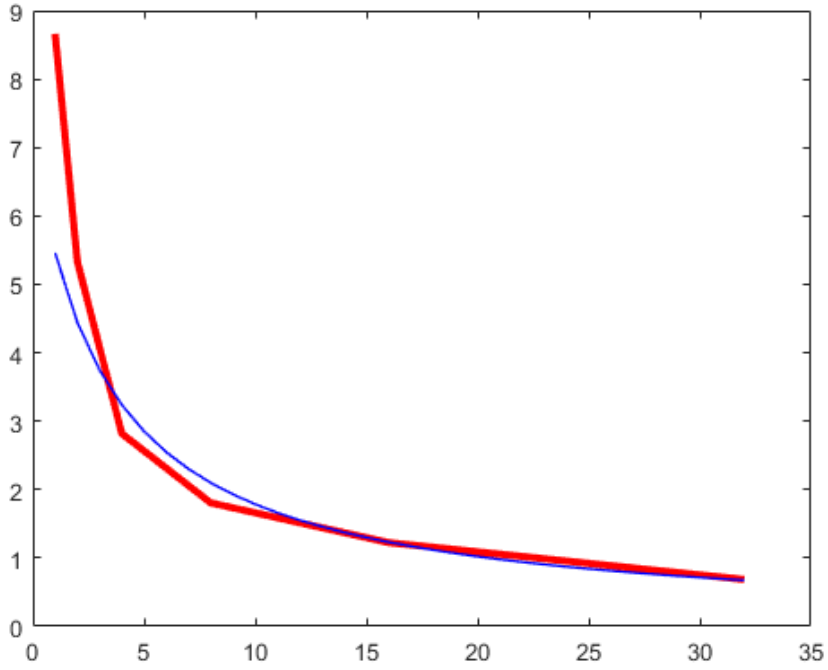
Figura 4.10: Approssimazione del tempo di esecuzione dello script CPPS.m fino a 256 secondi



Guardando al grafico di Figura 4.10 si può vedere il confronto tra l'andamento reale del tempo di esecuzione necessario (disegnato in blu) e quello approssimato tramite apposite funzioni (disegnato in arancio); si nota come nonostante la complessità computazionale sia nella forma $O(N^3)$, il tempo richiesto per l'esecuzione dell'algoritmo con file di dimensioni via via crescenti aumenta linearmente all'aumentare della dimensione dei campioni analizzati.
La causa di questo comportamento è dovuta al fatto che la complessità trovata per via teorica si riferisce all'analisi ed estrazione dei parametri di un singolo frame e quindi se aumenta la dimensione del file il tempo di esecuzione aumenta linearmente per il semplice fatto che la complessità di ogni singolo frame viene moltiplicata per il numero di frame del file.
Utilizzando i risultati ottenuti durante i test sul tempo di esecuzione e quelli derivanti dall'approssimazione dell'andamento è stato possibile, inoltre, tracciare il grafico delle prestazioni dell'algoritmo all'aumentare della dimensione del file analizzato. Nel grafico di Figura 4.11 è disegnato in rosso l'andamento prestazionale trovato utilizzato i file ottenuti per via sperimentale metre la linea blu rappresenta l'andamento ottenuto applicando 4.4 ai dati ottenuti con l'approssimazione.

I due andamenti sono confrontabili con la sola eccezione del primo punto in cui si hanno migliori prestazioni per via sperimentale.
Questa tipologia di andamento era prevedibile in quanto il tempo di esecuzione cresce secondo la forma $y = x$; applicando 4.4 si ottiene un andamento del tipo $y = 1/x$.

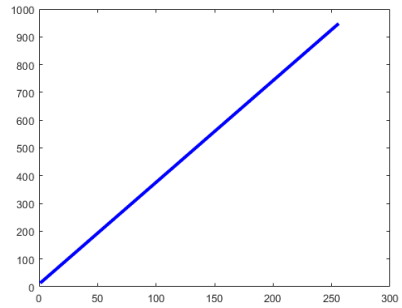
Figura 4.11: Prestazioni dell'algoritmo CPPS.m con relativa approssimazione (linea blu)



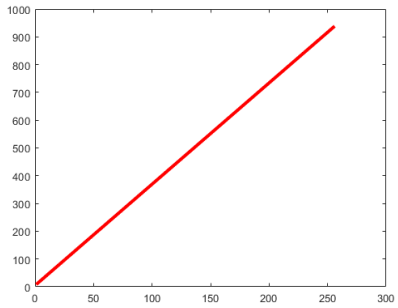
4.2.2 Test condotti su VHM

Per il dispositivo Vocal Holter Med, i cui metodi di analisi sono descritti nella relativa sezione del Capitolo 3, non si è adottato il metodo di misura basato sugli *O grande* in quanto non si aveva la necessità (e la possibilità) di eseguire un confronto con altri dispositivi dello stesso tipo. Il metodo utilizzato è di tipo più sperimentale ed è stato usato per stimare la crescita del tempo di esecuzione di differenti operazioni all'aumentare della dimensione del file utilizzato. L'approssimazione visualizzata sotto forma di linea tratteggiata nei grafici di Figura 3.7 e 3.8 è stata estesa (sempre utilizzando le funzioni polyfit e polyval di Matlab ™) al di fuori dell'intervallo di misurazione ottenendo l'andamento che si vede nelle figure 4.12a e 4.12b.

Figura 4.12: Approsimazioni dei tempi di esecuzione su dispositivo VHM



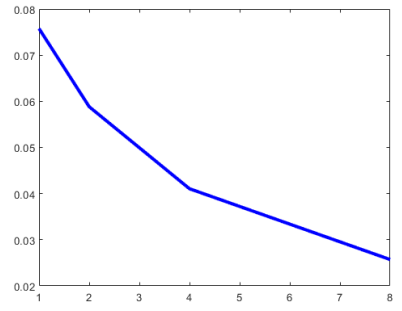
(a) Estrazione CPPS e altri parametri



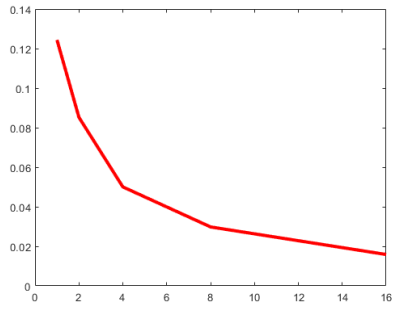
(b) Estrazione solo CPPS

Inoltre, non è stato possibile calcolare tutti i parametri dati dalle equazioni da 4.4 a 4.9 perché non sono disponibili dati sul numero di operazioni effettuate dall'algoritmo. L'unico indicatore calcolabile è quello dato da 4.4, disponibile per tutti i test effettuati. Utilizzando la formula di 4.4 si riesce a tracciare un andamento per le prestazioni degli algoritmi il quale, ovviamente, sarà nella forma $y = 1/x$ restituendo il grafico di Figura 4.14 che è stato calcolato usando i valori graficati in Figura 4.12a ed è, quindi frutto di un'approssimazione. Questo tipo di andamento è comune ad entrambe le tipologie di algoritmo testato come si vede dai due grafici contenuti in Figura 4.13 in quanto la crescita del tempo necessario all'esecuzione è di tipo lineare. Nel grafico di Figura 4.14 viene disegnato l'andamento ottenuto eseguendo il calcolo del parametro di 4.4 sull'approssimazione calcolata con la funzione *polyfit* fino a 256 frame.

Figura 4.13: Prestazioni per l'estrazione con VHM

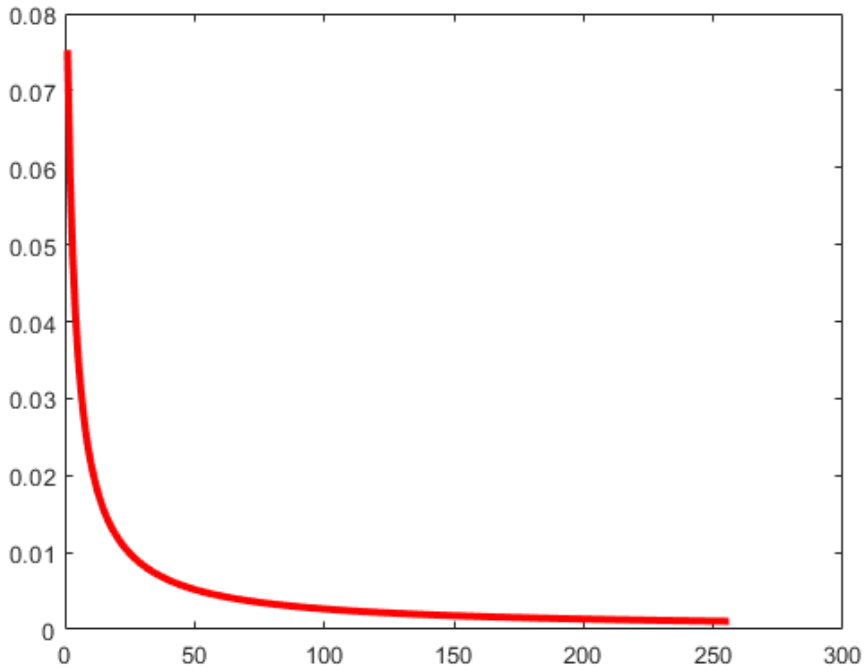


(a) Estrazione CPPS e altri parametri



(b) Estrazione solo CPPS

Figura 4.14: Stima del decadimento delle prestazioni all'aumentare delle dimensioni del file analizzato con VHM



Per quanto riguarda i test effettuati sul tempo necessario all'elaborazione dei parametri necessari si nota come, se si decide di effettuare un'analisi a più alta risoluzione e di avere, quindi, un valore di CPPS per ogni frame del file audio i tempi necessari sono decisamente elevati. Utilizzando l'approssimazione visualizzata nei grafici di Figura 4.12 si nota come per l'analisi di 256 frame presi singolarmente (256 frame sono circa 12 secondi di registrazione) il tempo necessario all'analisi sia di 947 secondi ($\simeq 15.8$ minuti) per l'estrazione di tutti i parametri e 938 secondi ($\simeq 15.6$ minuti) nel caso di estrazione del solo CPPS. Questi tempi di calcolo sono incompatibili con le richieste prestazionali che ci si è prefissati in origine e non sono adatti per un utilizzo immediato del dispositivo. Per un utilizzo a minor ritardo del dispositivo VHM è consigliabile effettuare analisi di più lunga durata imponendo che venga estratto un valore di CPPS ogni 256 frame ottenendone, quindi, 5 ogni minuto di registrazione. Ogni analisi di 256 frames "uniti" richiede all'incirca 4.5 secondi, un valore che è molto al di sotto dei 12 secondi rappresentati dai frame; con questo tempo di elaborazione si può utilizzare il VHM in modo più performante ed avere analisi più rapide a patto, però, di registrare file di dimensioni maggiori. Si è ancora molto lontani da un eventuale

utilizzo del dispositivo in real time in quanto per tale impiego servirebbero dei tempi di calcolo nell'ordine dei millisecondi in grado di garantire l'elaborazione completa della porzione di file in ingresso in un determinato istante con conseguente restituzione dei parametri ad esso relativi.

4.2.3 Algoritmi di predizione

Chiusa l'analisi delle prestazioni e della complessità computazionale degli algoritmi utilizzati per l'estrazione delle features, al fine di scegliere quali siano le features strettamente necessarie per la classificazione e, quindi, quale modello di predizione utilizzare è necessario individuare le richieste computazionali degli algoritmi di predizione. Per trovarle non si è proceduto per via sperimentale ma è stato sufficiente ricercare quali fossero le complessità dei modelli analizzati. Di seguito, nella Tabella 4.5 sono riportate le complessità per la fase di predizione [16], [17]; la fase di training dell'algoritmo non viene presa in considerazione in quanto verrebbe eseguita una volta sola. Gli

Tabella 4.5: Complessità computazionale dei modelli di predizione.

Modello	Complessità computazionale	
Decision Trees	$O(P \cdot d)$	P=numero features d=Profondità dell'albero
SVM	$O(N_{sv} \cdot P)$	P=numero features N_{sv}=Numero di Support Vectors
Boosted Trees	$O((P \cdot d) \cdot n)$	P=numero features d=Profondità dell'albero n= numero di alberi utilizzati
Discriminant Analysis	$O(P)$	P=numero features
KNN	$O(N \cdot d)$	N=numero di candidati con cui confrontare P=numero features

algoritmi elencati quelli che hanno restituito le migliori accuratezze in fase di training e testing. In particolare, la classe di algoritmi che ha dato le migliori accuratezze è SVM il quale però, come detto in precedenza, risulta di difficile implementazione e controllo. Oltre a questo, è difficile prevedere un costo computazionale di tale algoritmo in quanto è strettamente legato al numero

di Support Vectors che vengono utilizzati i quali, a loro volta, dipendono da come è stato eseguito il training e come sono stati impostati i parametri per la classificazione.

CAPITOLO 5

Conclusioni

I risultati fin qui ottenuti permettono di effettuare delle valutazioni sullo stato del processo di classificazione dei file audio registrati e sull'estrazione delle features ad essi relative. Mettendo insieme lo studio effettuato sull'accuratezza restituita dai modelli e sulla complessità computazionale è stato possibile individuare i parametri e gli scenari che permettono di massimizzare l'accuratezza teorica mantenendo bassa la complessità computazionale.

5.1 Risultati finali

Unendo i risultati ottenuti nelle due tipologie di analisi, quella dedicata all'accuratezza e quella dedicata alla complessità computazionale dei differenti algoritmi è stato possibile ultimare la ricerca che è oggetto di questa tesi identificando la miglior combinazione di parametri per cui si massimizza l'accuratezza teorica ottenibile minimizzando il costo computazionale. Per le considerazioni finali sono stati utilizzati i dati rappresentati nei grafici delle Figure da 3.5 a 3.2 per quanto riguarda l'accuratezza e i dati ottenuti per via sperimentale per quanto riguarda le differenze che intercorrono tra i diversi algoritmi e dispositivi testati.

Cominciando dagli algoritmi per l'estrazione delle features si vede con assoluta certezza come la scelta migliore sia utilizzare l'algoritmo in grado di estrarre tutte le features con una singola esecuzione. Lo si nota, in primis, dal tempo di esecuzione sensibilmente inferiore e, poi, dai parametri descritti nelle Figure da 4.5 a 4.8.
Come già detto in precedenza, tale differenza è dovuta principalmente alla natura dei 4 algoritmi separati, i quali si trovano a rieseguire delle operazioni i cui risultati sono già stati calcolati all'interno di altre sequenze di istruzioni. Sicuramente "mettendo in comunicazione" i 4 algoritmi si otterrebbero prestazioni migliori e, magari, confrontabili con quelle ottenute con l'algoritmo CPPS.m (estrazione in blocco dei parametri). Il vantaggio di mantenere una netta separazione tra le differenti metodologie richieste per l'estrazione delle feature risiede nel fatto che è possibile, in questo modo, escludere dal calcolo determinate parti che estraggono parametri non molto utili alla classificazione finale. Ovviamente tale considerazione è possibile farla anche con uno script completo, tuttavia, è meno *user-friendly* nel caso di utilizzo da parte di terzi, i quali si troverebbero a fare reverse engineering su un codice che non conoscono.
Analizzando quanto ottenuto per l'analisi con il VHM si nota che, come già esposto all'interno del

Capitolo 3, il miglior metodo di valutazione dei file audio e di estrazione dei parametri è quello che utilizza finestre più grandi prendendo 256 frame ed estraendo un singolo valore di CPPS. Così facendo si riesce ad abbattere il tempo di esecuzione a discapito, però, della risoluzione della distribuzione. Sono necessari, infatti, file di dimensioni maggiori per ottenere delle distribuzioni complete ed utilizzabili. Se si sceglie di svolgere il calcolo su un singolo frame ci si imbatte in tempi di esecuzione anche molto elevati, come si vede dalla Figura 4.12 del Capitolo 4.

Estratte le feature dai campioni audio vengono utilizzate per l'addestramento dei modelli che serviranno per la classificazione dei dati. Nel capitolo 3 sono illustrati i procedimenti utilizzati per la ricerca effettuata in questa tesi, all'interno del Capitolo 4 sono, invece, illustrati i risultati ottenuti. Servendosi di quanto scritto in queste due parti del presente elaborato è possibile definire per ogni dataset utilizzato quali siano i parametri e i modelli migliori da utilizzare nel caso si voglia ridurre al minimo il numero di feature utilizzate.

Dataset Completo

Dall'analisi della correlazione tra forma della voce e parametri è emerso che ci sono 9 parametri ad elevata correlazione: $CPPS_{std}$, $CPPS_{5prc}$, $voiced_{mean}$, $voiced_{std}$, $unvoiced_{mean}$, $unvoiced_{std}$, $F0_{mean}$, $F0_{std}$, Dt%.
Utilizzandoli per l'addestramento degli algoritmi si vede che l'algoritmo migliore è Cubic SVM (98.5%) seguito da RUSBoosted Tree (92.3%) e Quadratic Discriminant (90.8%). Nonostante l'elevata accuratezza degli algoritmi non è possibile affermare con certezza che rappresentino la scelta migliore in quanto manca un metodo di validazione dei modelli; si potrebbe, facilmente, essere in presenza di overfitting e quindi di un eccessiva aderenza del modello ai dati specifici analizzati. Per tali motivi si deve guardare verso modelli a minor accuratezza quali Quadratic SVM (84.6%), Fine Tree e Medium Tree (entrambi con 80.0%). Tra questi tre proposti, sicuramente, i modelli da tenere veramente in considerazione sono i due basati su strutture ad albero; permettono, infatti, un maggior controllo dell'algoritmo utilizzato, una maggior comprensione per l'utente finale, soprattutto, riescono a conservare una bassa complessità computazionale che è uno degli obiettivi che ci si era prefissati in partenza.

Scegliendo di minimizzare il numero di predittori utilizzati si può arrivare fino ad utilizzarne solo 3

scelti sulla base della loro correlazione con la forma della voce. Così facendo si utilizzano le feature $voiced_{mean}$, $unvoiced_{std}$, $F0_{mean}$. La scelta è ricaduta sulla selezione basata su correlazione con la forma vocale per il fatto che il numero minimo di parametri selezionabili con l'altro metodo i quali permettono di mantenere un buon livello di accuratezza sono 4 ($CPPS_{std}$, $CPPS_{5prc}$,$unvoiced_{std}$, $F0_{std}$) e tra loro figura il CPPS che, come si è visto, richiede numerosi calcoli.
Utilizzando i tre parametri sopra elencati, per i modelli scelti come migliori utilizzando 9 feature si ottengono delle accuratezze ancora relativamente alte: per fine tree e medium tree 73.8% e per Quadratic SVM 67.7%. Nuovamente i modelli basati su albero sono la miglior scelta possibile escludendo quelli che possono dare overfitting e quelli ad elevata complessità computazionale.

La complessità della predizione per il modello completo si attesta quindi a $O(P \cdot d)$ dove P è il numero di features e d è la profondità dell'albero. A differenza dei modelli basati su SVM dove non si ha il pieno controllo del numero di Support Vectors che determinano la complessità computazionale, in questo caso la profondità dell'albero è controllabile e permette di impostare un limite ben preciso limitando, quindi, anche la complessità computazionale.

Dataset Maschile

Dall'analisi della correlazione tra forma della voce e parametri è emerso che ci sono 8 parametri ad elevata correlazione: $CPPS_{mode}$, $CPPS_{5prc}$, $CPPS_{skew}$, $voiced_{mean}$, $voiced_{std}$, $unvoiced_{mean}$, $unvoiced_{std}$, $F0_{mean}$.
Le feature sopra elencate sono state usate per l'addestramento degli algoritmi di predizione. Per questo dataset i risultati migliori sono stati ottenuti col modello Quadratic SVM (94.3%), minor accuratezza è stata registrata utilizzando Medium Gaussian SVM e Linear Discriminant i quali hanno restituito entrambi una precisione del 85.7%. Il modello Medium Tree ha prodotto un'accuratezza pari a 80.0% (risultato condiviso con Fine Tree). Tutti gli altri algoritmi considerati nell'analisi hanno riportato un'accuratezza del 77.1% (Coarse Tree, Linear SVM, Subspace Discriminant).
Il risultato ottenuto con Quadratic SVM non è da considerare pienamente attendibile in quanto un tale livello di accuratezza è sintomatico di overfitting e, per tale motivo non viene scelto tra i modelli da utilizzare. Di maggior rilevanza sono invece i risultati ottenuti con i modelli ad albero perché, come detto già per il caso del dataset completo, permettono un'accuratezza piuttosto buona con una complessità computazionale e di gestione molto inferiore rispetto agli altri mo-

delli valutati. Oltre a quelli ad albero anche il modello Linear Discriminant restituisce una buona accuratezza con una complessità che è ancora inferiore a quella dei modelli ad albero. Per tali motivi la scelta migliore con 8 predittori è rappresentata dall'utilizzare il modello Linear Discriminant. Questo algoritmo potrebbe, tuttavia, presentare delle complicazioni con gruppi di dati più numerosi non riuscendo più ad interpretare correttamente la separazione tra classi.

Anche in questo caso si è deciso di verificare quale fosse il numero minimo di parametri utilizzabili mantenendo comunque una buona accuratezza teorica. Confrontando i dati ottenuti con i due metodi di rimozione delle feature si è visto che il numero minimo di parametri utilizzabili è 3, scelti sulla base della loro correlazione con la forma della voce e non sulla base della correlazione reciproca tra parametri. Usando questo criterio le feature scelte sono $voiced_{mean}$, $voiced_{std}$ e $F0_{mean}$.
Utilizzando i tre parametri sopra elencati per l'addestramento dei modelli indicati come migliori con 8 feature si ottiene un significativo abbassamento di accuratezza per il modello Linear Discriminant il quale ora restituisce una precisione del 65.7%; subisce meno riduzione l'accuratezza di Medium e Fine Tree la quale si attesta al 71.4%. Per un basso numero di parametri la scelta di un algoritmo basato su albero di decisione si rivela vincente da un punto di vista dell'accuratezza e da quello della complessità computazionale.

Utilizzando 8 feature la complessità della predizione varia tra $O(P \cdot d)$ e $O(P)$ (P=numero feature, d= profondità albero) a seconda che si scelga, rispettivamente, un modello basato su albero o Linear Discriminant. Utilizzando 3 feature e scegliendo di usare un modello ad albero per la predizione la complessità è pari a $O(P \cdot d)$.

Dataset Femminile

I parametri che hanno la maggior correlazione con la forma della voce sono $CPPS_{std}$, $voiced_{mean}$, $voiced_{std}$, $unvoiced_{mean}$, $unvoiced_{std}$, $F0_{mean}$, $F0_{std}$, Dt%.
Utilizzandoli per l'addestramento dei modelli è stato possibile arrivare ad accuratezze in media piuttosto elevate. Escludendo, a causa del possibile overfitting, il 100% ottenuto con Fine Gaussian SVM e il 93.3% ottenuto con Cubic SVM gli altri algoritmi hanno fatto registrare precisioni intorno al 80.0%: con Quadratic SVM si è ottenuto il 83.3%, con Medium Tree, Fine Tree e Medium

Gaussian SVM l' 80.0%, con Linear SVM il 73.3% e, infine, con Linear Discriminant e Subspace Discriminant il 63.3%. Tra tutti questi, quelli che rapprensentano le scelte migliori sono i due modelli basati su albero decisionale e quello basato su SVM con Kernel di tipo Medium Gaussian le cui accuratezze restituite senza alcun tipo di validazione sono identiche.
Tuttavia, guardando alla complessità computazionale la scelta migliore da effettuare è quella di utilizzare Fine Tree o Medium Tree i quali hanno possibilità di gestione maggiori rispetto ad un algoritmo di tipo Support Vector. La complessità computazionale di quest'ultima categoria di modelli è $O(N_{sv} \cdot P)$ dove P è il numero di features e N_{sv} è il numero di Support Vectors utilizzati. A primo impatto sembrerebbe che le due complessità siano simili tuttavia, risulta molto complessa la stima del numero di Support Vectors usati per la classificazione in quanto dipende dai parametri usati in fase di addestramento del modello, alcuni anche piuttosto complessi. A differenza di quanto accade per i modelli ad albero, è difficile individuare una dipendenza tra il cambiamento dei criteri utilizzati con il training set e il numero dei Support Vectors. Nello scenario peggiore questa quantità è pari al numero dei campioni utilizzati per l'addestramento, in tal caso si ha una complessità computazionale piuttosto elevata.
Come per i due dataset precedenti si è deciso di ridurre al minimo le feature utilizzate. È possibile ridurre l'utilizzo dei parametri fino a due soli predittori scelti sulla base della loro correlazione con la forma della voce; analizzando i grafici contenuti in Figura 3.5 e 3.6 si vede che i due parametri sono $unvoiced_{std}$ e $F0_{mean}$ con i quali si ottengono accuratezze pari al 70.0% per i modelli Fine e Medium Tree e 73.3%. Come detto in precedenza, anche in questo caso conviene l'utilizzo di un modello basato su albero se si ha la necessità di contenere il costo per la predizioni.
La complessità computazionale per questo modello è rappresentata da uno dei valori tra $O(N_{sv} \cdot P)$ e $O(d \cdot P)$, a seconda del modello che si sceglie per la classificazione.

Unione dei risultati

Come ci si era prefissati in partenza, l'ultimo step della ricerca consiste nell'unione dei risultati provenienti dalle due analisi condotte.
Per quanto riguarda gli algoritmi di estrazione quello che restituisce i risultati migliori in termini di prestazioni e tempo di esecuzione è CPPS.m ovvero quello che permette l'estrazione complessiva di tutte le feature legate al segnale vocale. Nonostante la sua complessità sia $O(N^3)$ restituisce

degli indicatori prestazionali migliori degli altri algoritmi valutati.
Utilizzando tale algoritmo per l'estrazione delle feature e uno dei modelli sopra descritti per la classificazione si ha comunque una complessità computazionale il cui limite massimo è rappresentato da $O(N^3)$. Questo dato è comunque poco significativo se non è supportato da altri indicatori quali il tempo di esecuzione, il CPI, il MIPS o le caratteristiche hardware/software della macchina su cui si esegue l'algoritmo.

Si riporta in Tabella 5.1 un riassunto dei migliori parametri individuati. Per ogni dataset analizzato è riportato, nella colonna *Numero parametri*, il numero massimo di parametri e il numero minimo. Con massimo si intende tutti quelli a più alta correlazione significativa con la forma della voce; con minimo il numero di quelli che permettono di conservare una buona accuratezza nonostante la minimizzazione del numero di parametri usati. Nella colonna *Metodi* sono riportati i metodi di classificazione migliori per il relativo numero di parametri usati; all'interno della colonna *Accuratezze* sono indicate le accuratezze ottenute con tali metodi. Nell'ultima colonna, quella denominata *Complessità*, è contenuta, per ogni metodo di classificazione, la relativa complessità computazionale. Il numero di Support Vector utilizzati dagli algoritmi di tipo SVM è strettamente legato alla fase di addestramento del modello e, quindi, risulta di difficile individuazione. Il valore massimo che può assumere è pari al numero totale dei parametri utilizzati. Per i modelli basati su albero è riportato anche il valore che indica la profondità dell'albero determinato sulla base dell'albero creato usando Classification Learner di Matlab ™ .

Tabella 5.1: Risultati ottenuti - d: profondità albero, p: numero di predittori usati, N_{sv}: numero di Support Vectors usati

Dataset	Numero parametri	Metodi	Accuratezze	Complessità
Completo (Max)	9	Medium Tree	76.9%	$O(d \cdot p); d = 5$
Completo (Min)	3	Medium Tree	73.8%	$O(d \cdot p); d = 6$
Uomini (Max)	8	Medium Tree Linear Discriminant	80% 85.7%	$O(d \cdot p); d = 3$ $O(p)$
Uomini (Min)	3	Medium Tree	71.4%	$O(d \cdot p); d = 3$
Donne (Max)	8	Medium Tree Medium Gaussian SVM	80% 80%	$O(d \cdot p); d = 4$ $O(N_{sv}); N_{sv} \leq p$
Donne (Min)	2	Medium Tree Medium Gaussian SVM	70% 73.3%	$O(d \cdot p); d = 4$ $O(N_{sv}); N_{sv} \leq p$

5.2 Validazione dei modelli

I metodi utilizzati, descritti nel Capitolo 3, sono piuttosto robusti da un punto di vista applicativo e fanno sì che sia possibile avere un'idea precisa del comportamento dei modelli applicati al dataset in esame tuttavia, durante l'analisi, sono state riscontrate delle problematiche intrinseche ai dati utilizzati.

Il problema principale del gruppo di dati utilizzati risiede nella dimensione dei dati stessi. Il dataset su cui è stata eseguita l'intera ricerca che è oggetto di questa tesi era composto da 13 attori ognuno interpretante le cinque forme vocali (piatta, tonda, quadra, triangolare, spigolosa) per un totale di 65 registrazioni divise pressochè equamente tra uomini e donne (7 attori uomini, 6 attrici donne). Per addestrare in modo affidabile i modelli che vengono poi utilizzati per la classificazione sarebbe indicato avere un dataset molto più numeroso così da poterlo dividere in due parti: la prima utilizzata per il training del modello e la seconda utilizzata per la validazione/testing dello stesso. Si potrebbe pensare di effettuare ugualmente la divisione del dataset anche avendo poche osservazioni, tuttavia, questo non permetterebbe di addestrare efficacemente il modello (se si scegliesse di riservare più dati alla validazione) o di effettuare una validazione affidabile (se si

scegliesse di destinare più osservazioni all'addestramento dei modelli).
Nello specifico, all'interno del'applicazione Classification Learner di Matlab ™, sono disponibili due metodi di validazione differenti: K-fold cross validation e Holdout validation. Nel primo caso il dataset viene diviso in K parti; di queste, K-1 sono usate per il training mentre la restante è usata per testare il modello. Il processo è reiterato in modo che tutte le osservazioni siano usate almeno una volta sia per l'addestramento che per il test del modello. L'accuratezza generale è stimata utilizzando tutte le accuratezza ottenute durante la reiterazione della divisione del dataset iniziale.
Nel caso di Holdout Validation, invece, il gruppo di osservazioni, viene diviso in modo statico tra training set e validation set; le osservazioni usate per l'addestramento non vengono mai usate per la validazione, e viceversa.

Il metodo denominato Holdout Validation non è applicabile al dataset utilizzato perché non contiene un alto numero di osservazioni; se si andasse a dividere il dataset in modo statico qualsiasi rapporto di divisione dei dati produrrebbe uno sbilanciamento troppo marcato con conseguenti valori di accuratezza bassi o con problemi di overfitting.
Utilizzando invece il metodo K-fold cross validation non si va più in contro ai problemi individuati per Holdout Validation perché, scegliendo un opportuno valore di K, si riesce ancora a conservare un buon numero di osservazioni per l'addestramento dei modelli. La problematica che potrebbe insorgere utilizzando questo modello è quella dell'overfitting: scegliendo un valore di K troppo elevato verrebbero utilizzati troppi dati per il training e troppo pochi per la validazione (nel caso estremo di Leave One Out Cross Validation si utilizza una sola osservazione per il testing e tutte le rimanenti per il training).

Per i motivi sopra descritti non è stato possibile introdurre, in questa ricerca, un metodo di validazione dei modelli addestrati. L'analisi è stata svolta basandosi sulle accuratezze restituite senza validazione sempre tenendo a mente la possibilità di overfitting dei modelli addestrati.

5.3 Implementazioni future

Gli esiti della ricerca condotta durante il lavoro di questa tesi sono da considerare esclusivamente come punto di partenza e non come punto di arrivo. Basandosi su quanto visto in questi studi,

risulta interessante ampliare la ricerca andando, in primis, a procurarsi un maggior numero di osservazioni per poter introdurre la validazione dei modelli addestrati. Una volta introdotta, sarà possibile fare osservazioni più precise sulle necessità di ogni modello e sulle possibilità da questo offerte.
Nell'introdurre nuove osservazioni bisognerà, però, fare attenzione a non sbilanciare il dataset continuando a mantenere la (quasi) parità tra il numero di componenti maschili e femminili. Avere un dataset bilanciato per quanto riguarda il sesso dei componenti permette di sottocampionarlo sulla base del parametro Gender addestrando in modo differente i modelli per la classificazione degli uomini o delle donne (come è stato fatto in questa tesi). La divisione in uomini e donne ha permesso, infatti, di adattare meglio i modelli alle caratteristiche dell'uno o dell'altro dataset ottenendo un, seppur contenuto, aumento di accuratezza generica rispetto al processare tutte le registrazioni senza distinzioni.
Mantenere un buon bilanciamento dei dati permette anche di slegarsi dall'utilizzo di modelli di classificazione più complessi che mirano a ridurre le differenze tra la numerosità di ciascuna classe (ad esempio i modelli basati su Random UnderSampling o su Support Vectors) potendo quindi avere una minor complessità computazionale sia in fase di addestramento che di classificazione del campione analizzato.

Basandosi su quanto visto durante il lavoro descritto in questa tesi, i parametri estratti dai file audio registrati sono sufficienti per la completa caratterizzazione di tutti gli aspetti del segnale vocale e permettono di individuare degli indicatori robusti per la classificazione dei campioni in esame. Sicuramente, in futuro si rivelerà necessario modificare gli script per l'estrazione delle feature facendo sì che vengano estratte solo quelle di reale interesse diminuendo così il tempo di esecuzione dell'algoritmo stesso e, con buona probabilità, la sua complessità computazionale. Questa riduzione, però, occorre effettuarla dopo aver eseguito nuovamente il training dei modelli come esposto nel Capitolo 3 avvalendosi di un metodo di validazione robusto e, anch'esso, sperimentato. Così facendo si va ad avere una nuova stima di quello che è il comportamento dei modelli se si tende a diminuire i predittori utilizzati; comportamento che potrebbe essere uguale o meno a quello osservato senza la validazione.

Finita la fase di test sulle accuratezze restituite dagli algoritmi si passerà alla fase di implemen-

tazione vera e propria andando a generare un algoritmo che si basa sui modelli addestrati con Classification Learner. Per tale motivo è, altresì, fondamentale scegliere il modello di predizione tenendo bene a mente questa necessità. Se la scelta finale ricade su algoritmi del SVM (Support Vector Machines) si otterrà, con buona probabilità, una miglior accuratezza generica del modello però, di contro si dovrà fronteggiare una crescita notevole nella complessità di creazione, gestione e applicazione del modello.

Fatta l'analisi della variazione di accuratezza, scelto il miglior modello e ottimizzato l'algoritmo per l'estrazione si potrà, finalmente, implementare a livello pratico l'insieme di questi risultati raggiungendo, così, gli obiettivi citati nel Capitolo 1. Sarà necessario creare differenti versioni degli algoritmi utilizzati in quanto la destinazione finale varierà tra un'implementazione su macchine classiche (PC) e una su dispositivi single-purpose dedicati in modo specifico a questa tipologia di analisi. Dopo l'implementazione sui dispositivi si aprirà la fase di testing e ottimizzazione del prodotto, al termine della quale si potrà dire terminata la ricerca…forse.

5.4 Considerazioni finali

Grazie a questa ricerca si è riusciti a mettere in luce un aspetto fondamentale della comunicazione che deriva in buona parte dal primo dei 4 assiomi sulla comunicazione di Watzlawick: è impossibile non comunicare. È stato possibile vedere come il variare di un parametro oggettivo e misurabile con metodi scientifici sia indicatore più o meno preciso della variazione di uno stato d'animo in chi parla e trasmette un messaggio facendo largo uso, a sua insaputa, della comunicazione paraverbale. La possibilità di rendere con parametri oggettivi qualcosa che molte volte sfugge al controllo fornisce nuove possibilità di utilizzo per chi lavora con le emozioni o per chi su di esse fa ricerca dando l'opportunità di supportare in modo rigoroso tutto lo studio.
Il segnale vocale è solo uno degli indicatori utilizzabili per valutare l'efficacia della comunicazione paraverbale; studi futuri potrebbero muoversi nella direzione del riconoscimento di espressioni e mimiche facciali attraverso procedure di Machine Learning, Computer Vision e Image Understanding, strumenti altamente performanti e in fase di sviluppo attraverso cui si riesce ad interpretare quello che una persona vuole (o non vuole) comunicare grazie ai segnali inviati dal volto e dal corpo nella sua interezza. Unire i due paradigmi di analisi, quello vocale e quello basato sull'interpretazione dei segnali visuali, crea uno scenario decisamente interessante che permette di arrivare a comprendere meglio il complesso mondo delle emozioni e della loro interpretazione.

Sono convinto che, grazie a questi studi e ai loro sviluppi, le macchine possano, veramente, passare dall'essere al servizio dell'uomo all'essere al suo fianco addirittura interpretando le emozioni che proviamo fornendo supporto emotivo e psicologico a chi lo ritiene necessario. Ad esempio, per i terapeuti può essere uno strumento da fornire ai pazienti per lavorare in autonomia, al di fuori delle sedute, sul monitoraggio dello stato d'animo durante la giornata, dando quindi la possibilità di riuscire ad associare eventi quotidiani a cambi nello stato emotivo del paziente.

Rimanendo più legati all'origine di questi studi, il cui obiettivo era la creazione di uno strumento da fornire agli attori per la valutazione della loro prestazione attoriale, sicuramente lo sviluppo della ricerca sarà in grado di mettere nelle mani di Actor Coach e Vocal Coach un qualcosa che sia di supporto reale all'allenamento della performance da preparare e che sia in grado di indirizzare l'attore e correggerlo migliorando la sua recitazione.

Se si riesce ad implementare, all'interno di un dispositivo simile per dimensioni e portabilità al VHM, un algoritmo che in tempo reale sia in grado di giudicare la forma vocale che un attore sta riproducendo durante le riprese di un film, un regista avrebbe un ulteriore strumento per giudicare la performance ed indirizzare il suo film.

In chiusura, occorre ricordare che per quanto la ricerca tecnologica possa fare passi avanzati e fornire strumenti precisi ed affidabili non esiste, quantomeno in questo campo, una macchina, una tecnologia o un algoritmo in grado di sostituire la valutazione umana fatta di sentimenti, emozioni, empatia e ragionamenti complessi che, sovente, trascendono dalla comprensione dell'essere umano.

CAPITOLO 6

Appendici

6.1 Appendice A - La comunicazione

Il termine comunicazione deriva dal latino *communicatio -onis.* Come definito dal dizionario Treccani tale termine può essere inteso in tre sensi differenti. Un senso più ampio e generico, un senso proprio e un senso più astratto. In senso generico la comunicazione è l'azione, il fatto di comunicare, cioè trasmettere ad altro o ad altri. Ad esempio: comunicazione del movimento alle parti di un meccanismo, comunicazione dei privilegi. In senso proprio si intende comunicazione come il rendere partecipe qualcuno di un contenuto o di uno stato d'animo. In senso più astratto, infine, si intende una relazione tra persone in grado di creare tra di esse un legame di partecipazione reciproca e comprensione.
Il tipo di comunicazione su cui ci si concentra per questi studi è quello di tipo prettamente "umano" (*proprio*, come definito da Treccani) ovvero quello utilizzato per far arrivare ad un'altra persona uno stato d'animo o una determinata emozione. Questo tipo di comunicazione può anche essere definita interpersonale in quanto coinvolge due o più individui i quali si influenzano reciprocamente attraverso un meccanismo noto come feedback ovvero la continua acquisizione di stimoli esterni volti a modellare meglio il messaggio trasmesso così da aumentarne l'efficacia.
Attingendo da quanto scritto da Gregory Bateson in Steps to an Ecology of Mind (1972) Paul Watzlawick, psicologo ed esponente di spicco della scuola di Palo Alto, elaborò 5 assiomi fondamentali validi per qualsiasi tipo di comunicazione[1]. Queste regole sono contenute nel libro Pragmatica della Comunicazione Umana e possiamo sintetizzarli come segue:

1. È impossibile non comunicare
2. In ogni comunicazione ci sono due livelli: contenuto e relazione (comunicazione e metacomunicazione)
3. Il flusso della comunicazione si articola secondo il proprio modo di vedere (la propria punteggiatura)
4. La comunicazione avviene attraverso canali verbali e non verbali (digitali e analogici)
5. La comunicazione può avvenire in modo simmetrico o asimettrico (peer-to-peer o client-server)

Di particolare interesse in questo studio è il quarto assioma. Secondo tale postulato, infatti, durante una comunicazione si utilizzano quelle che Watzlawick definisce modalità analogiche e digitali[1]. I due termini non sono sovrapponibili al concetto odierno di digitale e analogico nel campo, ad esempio, della trasmissione dei segnali in quanto, peraltro, sono stati utilizzati per la prima volta in un periodo in cui di tecnologie digitali ancora non si parlava così diffusamente.
Con il termine digitale si intende, infatti, l'insieme delle parole utilizzare, cioè gli elementi definiti della comunicazione. Con il termine analogico si fa riferimento al concetto di analogia e cioè qualcosa che evoca qualcos'altro per associazione diretta (basti pensare al noto gesto del "dito medio" il quale ha un suo preciso significato che non necessita di alcuna parola).
L'insieme di modalità analogiche e digitali coerenti tra loro fa sì che la comunicazione sia efficace e correttamente comprensibile dal ricevente: si pensi a quanto possa essere strano vedere una mamma che dice "ti voglio bene" al proprio figlio mentre con la mano fa il gesto di dargli uno schiaffo, non si avrebbe un'interpretazione univoca della comunicazione in atto. Si può quindi dire che un piano sia supportivo nei confronti dell'altro; supportivo e necessario in quanto se manca un piano l'altro perde la quasi totalità del suo significato. Le considerazioni derivanti dal quarto assioma di Watzlawick portano direttamente ad individuare 3 piani (o livelli) della comunicazione. Il piano verbale, il piano non verbale e, infine, il piano paraverbale, quello su cui ci concentra in questo scritto.
Il livello verbale è costituito da tutte le parole (scritte o dette) utilizzate per comunicare. Si riferisce ad esempio al linguaggio utilizzato, la sintassi, la struttura grammaticale e delle frasi.
Il livello non verbale riguarda tutto quello che si trasmette durante la comunicazione attraverso la propria postura, il proprio sguardo, i movimenti o la posizione che si occupa nello spazio. Si può vederla come la dimensione più fisica della comunicazione interpersonale.
Il livello paraverbale è formato da tutti quei modi in cui si esprime una comunicazione. Se la comunicazione è scritta esso può essere costituito dalla punteggiatura. Se la trasmissione è orale, come nel caso in esame, con comunicazione paraverbale si intende l'insieme di tono, timbro, volume, pause, silenzi e ritmo utilizzati dal mittente per arrivare al destinatario.
È proprio su questi indicatori paraverbali che la presente tesi si concentra andando a identificare dei metodi che possano sfruttare il piano paraverbale per la classificazione dell'intera comunicazione ad un livello, probabilmente, più profondo.

6.2 Appendice B - Precedenti studi

Lo studio effettuato in questa tesi si basa sul lavoro dell'equipe guidata dal Professor Carullo le cui ricerche si possono considerare in buona parte simili a quelle presenti in questo scritto per quanto riguarda la fase di classificazione dei segnali vocali registrati. I metodi utilizzati sono stati, infatti, ripresi in questa ricerca adattandoli ai dati utilizzati e integrandoli, dove necessario.

Lo studio dell'equipe del Prof. Carullo è cominciato con l'obiettivo di individuare ed estrarre parametri che potessero essere utili per differenti applicazioni che spaziano dall'acustica architettonica alla rilevazione di patologie dell'apparato fonatorio. Per lo studio di queste casistiche è stato creato il Vocal Holter Med (VHM), un dispositivo studiato appositamente per la registrazione della voce e per l'estrazione dei parametri dai file registrati. Tale dispositivo si occupa, appunto, di stabilire, attraverso determinati parametri della voce, le condizioni di salute di chi parla registrando il segnale vocale per mezzo di due microfoni: uno cosiddetto in aria, integrato nel dispositivo ed un altro a contatto indossabile da chi è sottoposto al test. Vengono utilizzate queste due tipologie di microfono in quanto permettono, attraverso l'elaborazione delle due registrazioni, di ottenere un file piuttosto pulito da un punto di vista acustico e, quindi, maggiormente utilizzabile ai fini della ricerca in oggetto.
Il VHM è stato utilizzato per studiare patologie dell'apparato fonatorio e per fare test riguardanti lo sforzo vocale in soggetti particolarmente esposti come insegnanti o cantanti. Durante queste ricerche si è cercato di definire una strada, per l'analisi de segnale vocale, che fosse una solida base per la ricerca dei parametri e la loro comprensione ai fini di utilizzarli nella classificazione.

Successivamente alle patologie dell'apparato fonatorio, il focus della ricerca si è spostato sui soggetti affetti dal Morbo di Parkinson. È stato richiesto supporto ai centri specializzati in tale malattia grazie ai quali sono stati raccolti campioni vocali da soggetti affetti dal Parkinson e, contemporaneamente, da soggetti sani per poterli confrontare.

Così facendo, è stato possibile determinare quali siano i valori dei parametri utilizzati che permettono di distinguere un soggetto sano da uno malato e qual è l'affidabilità di tale distinzione. Si è determinata l'affidabilità della classificazione in quanto risulta, ovviamente, importante non confondere soggetti sani con soggetti malati.

Gli studi sopra indicati sono stati impostati in modo non molto diverso dal metodo utilizzato nella presente tesi. La prima fase di lavoro si è concentrata sull'acquisizione del segnale vocale di un buon numero di soggetti malati. Per le registrazioni sono state utilizzate procedure che prevedevano l'utilizzo di fonemi e brani precedentemente preparati ed individuati come adatti allo scopo così da avere un riferimento condiviso tra tutti i soggetti partecipanti e poter effettuare un' analisi che fosse il più possibile coerente.

Successivamente, sono stati individuati dei parametri nel dominio del tempo, nel dominio della frequenza e in quello della *quefrenza* attraverso l'uso di appositi algoritmi. Questi indicatori, come accade per la ricerca fatta in questa tesi, sono stati utilizzati per la classificazione dei soggetti e per discernere tra soggetti sani e soggetti malati cercando di ottenere la più alta percentuale di accuratezza possibile. Il procedimento per la ricerca dei migliori parametri di classificazione è confrontabile con quello utilizzato in questa tesi in quanto si è basato, anch'esso, sulla correlazione tra parametri e forma della voce.

6.3 Appendice C - Modelli di classificazione

Per la suddivisione in classi dei dati ottenuti dalle registrazioni degli attori, sono stati utilizzati i classici metodi di classificazione i quali, attraverso procedure di addestramento, sono in grado di individuare dei *pattern* all'interno dei dati inseriti come input e, successivamente, utilizzare questi schemi come criterio per la classificazione di dati "sconosciuti", cioè nuovi dati mai visti prima dall'algoritmo inseriti come input e di cui viene richiesta la classificazione.

Utilizzando l'applicativo Classification Learner di Matlab™ si hanno a disposizione diverse classi di algoritmi per la classificazione dei dati. Vengono divisi in 6 macrocategorie identificabili come:

- Decision trees
- Discriminant Analysis
- Logistic Regression
- Support Vector Machines
- Nearest Neighbor Classifiers
- Ensemble Classifiers

Per ognuna delle classi si presenta una breve discussione teorica delle principali caratteristiche.

6.3.1 Decision Trees

Un albero di decisione è usato, di solito, per visualizzare scelte binarie avendo una struttura che, come dice il nome, è simile a quella di un albero con le sue ramificazioni[18]. Tra le classi di algoritmi per il machine learning quella degli alberi è una delle più semplici da implementare data la bassa complessità del metodo usato per la classificazione dei dati e la facilità di gestione dell'algoritmo generato.
Ad ogni nodo dell'albero è associata una scelta che può condurre ad una delle ramificazioni del nodo stesso. Tali ramificazioni conducono ad altre *foglie* sottostanti dove il processo di scelta è reiterato fino ad arrivare al punto in cui non c'è più alcuna scelta possibile: quel nodo finale è la decisione presa dall'algoritmo [18].

Sulla base della quantità di nodi si effettuano distinzioni tra[14]

- **Coarse Tree**: Pochi nodi. Numero massimo di split:4.
- **Medium Tree**: Maggior numero di nodi. Numero massimo di split:20.
- **Fine Tree**: Alto numero di nodi. Numero massimo di split: 100.

La quantità di nodi presente in un albero ne influenza, inoltre, la precisione in fase di classificazione. Un albero di tipo *coarse* è molto meno preciso rispetto ad un albero di tipo *fine* in senso generale; considerazioni di questo tipo vanno ulteriormente adattate alla tipologia di dati e al numero di parametri utilizzati.

6.3.2 Discriminant Analysis

Grazie a questa classe di algoritmi si riesce a superare la limitazione data dalla regressione logistica, la quale funziona in modo ottimale fino a quando si hanno due sole classi in cui dividere i dati in ingresso[19].
Gli algoritmi di tipo Discriminant Analysis si basano sulla stima della probabilità che un singolo input appartenga ad una determinata classe. Grazie a questo calcolo della probabilità si riescono a tracciare dei limiti tra le classi. Quando viene trovata la classe con la maggior probabilità di appartenenza il dato in ingresso viene classificato come appartenente a tale classe.[19].
Possiamo individuare due sotto-categorie di modelli:[20]

- **Linear Discriminant**: Suppone che tutte le classi abbiano la stessa matrice di covarianza. La funzione di divisione delle classi è lineare. Richiede meno risorse ma è anche meno precisa come divisione.
- **Quadratic Discriminant**: Suppone che ogni classe abbia la propria matrice di covarianza. La funzione di divisione delle classi è quadratica. Richiede più risorse a favore di una maggior efficienza di classificazione.

6.3.3 Logistic Regression

La regressione logistica è un modello di classificazione usato per assegnare osservazioni ad un set discreto di classi. [21] La particolarità di questo metodo è quella di restituire un valore di

probabilità di appartenenza ad una classe per ogni osservazione del dataset. Per eseguire il calcolo della probabilità si utilizza la Sigmoid Function, una funzione che mappa ogni valore reale su un altro numero compreso tra 0 e 1 usando una formula del tipo[21]

$$f(x) = \frac{1}{1 + e^{-x}} \tag{6.1}$$

In questo modo si generano dei *confini* per le due classi di destinazione. A seconda del punto in cui giace l'osservazione si decreta l'appartenenza ad una o all'altra classe. Questo metodo ha un'implementazione estremamente semplice ed un costo computazionale contenuto; ha, però, lo svantaggio di essere adatto solo per classificazioni binarie ed è per questo motivo che non viene preso in considerazione per il lavoro di cui si tratta in questo scritto.

6.3.4 Support Vector Machines

Gli algoritmi di tipo Support Vector Machine (SVM) si pongono come obiettivo il trovare un piano in N dimensioni (Iperpiano) che permetta di separare i dati in classi. [22] Analogamente a quanto avviene per la regressione logistica la posizione di un punto rispetto al piano ne determina l'appartenenza ad una classe.

Per la generazione dei piani di separazione l'algoritmo utilizza i *support vectors*, ovvero i punti che sono vicini a questi *iperpiani* e che ne influenzano la posizione e l'orientamento [22] e, com'è facile intuire, modificando il numero di questi punti si va a cambiare anche la posizione dei piani. Una delle particolarità di questo metodo, la quale è anche un punto di forza, è l'utilizzo dei *kernel*. Attraverso la loro implementazione è possibile mappare dati separabili ma non lineari in spazi di dimensione maggiore dove è più facile separarli.[23]. Utilizzando questo mapping su un nuovo spazio si riesce ad eliminare la dipendenza dai support vectors nel decidere l'appartenenza di un campione ad una determinata classe, potendo fare il calcolo esclusivamente utilizzando la funzione di Kernel, la quale è molto più semplice da un punto di vista computazionale.

Questa semplicità nel calcolo si paga, però, con una più difficile implementazione dell'algoritmo stesso in quanto si hanno più parametri da gestire e non tutti sono intuitivi nella modifica. Ad esempio, il numero di Support Vector utilizzati da un algoritmo è di difficile individuazione se non dopo un'ampia (e dispendiosa) procedura di reverse engineering sull'algoritmo finale; questo fa sì che anche la complessità computazionale dell'algoritmo sia difficile da stimare.

Sulla base del kernel utilizzato (e della sua complessità) distinguiamo in: [14]

- **Linear SVM**: Kernel con funzione di tipo linear
- **Quadratic SVM**: kernel con funzione di tipo quadratico
- **Cubic SVM**: kernel con funzione di tipo cubico
- **Fine Gaussian SVM**: fa una distinzione piuttosto precisa tra le classi. La dimensione del kernel è impostata a $\sqrt{P}/4$ dove P è il numero dei parametri usati per la predizione.
- **Medium Gaussian SVM**: La dimensione del Kernel è impostata a $\sqrt{P}$
- **Coarse Gaussian SVM**: La dimensione del Kernel è impostata a $\sqrt{P} * 4$. La divisione in classi è effettuata in modo grossolano a vantaggio del tempo di esecuzione.

6.3.5 Nearest Neighbor Classifiers

Questo tipo di modelli partono dall'assunzione che punti tra loro vicini abbiano le stesse caratteristiche.[24] Per ogni campione nel dataset da classificare l'algoritmo calcola la distanza dal punto di riferimento e aggiunge il campione ad una lista ordinata in base alla distanza ascendente. Fatto questo prende i primi K (definito all'inizio) campioni di questa lista e ne restituisce la classificazione. Abbiamo a disposizione sei classi di algoritmi a seconda di come viene calcolata la distanza o il numero di vicini:[14]

- **Fine KNN**: numero dei vicini impostato a 1.
- **Medium KNN**: numero dei vicini impostato a 10.
- **Coarse KNN**: numero dei vicini impostato a 100.
- **Cosine KNN**: distanza misurata col metodo del coseno (angolo tra i due vettori)
- **Cubic KNN**: utilizza la distanza cubica tra due vettori.
- **Weighted KNN**: si può specificare una pesatura per il calcolo della distanza

6.3.6 Ensemble Classifiers

I modelli di questo tipo sono costruiti unendo più classificatori "base" ottenendo un modello che è più performante dei suoi componenti. [25] Si distinguono quattro tipi di classificatori ensemble:[25]

- **Stacking**: un singolo dataset di training è passato a più modelli i quali effettuano il processo di apprendimento.Il dataset è poi diviso usando *k-fold validation* e, sfruttando questa divisione viene generato un nuovo modello(quello finale). Ogni modello effettua una predizione che viene poi usata per il training del modello finale.
- **Blending**: il procedimento è simile a quello dello stacking con l'unica differenza ch eil dataset iniziale non è diviso con k-fold ma è diviso staticamente in set per training e validazione.
- **Bagging**: vengono scelti randomicamente dei campioni dal dataset iniziale; per ogni campione viene fatto il training di un modello che lavora in parallelo ed è indipendente dagli altri. La predizione finale è determinata combinando le predizioni di tutti i modelli.
- **Boosting**: è una tecnica di autoapprendimento basata sull'assegnazione di pesi ai vari elementi del set. Il training inizia con pesi tutti uguali ma ad ogni iterazione viene corretto il loro valore sulla base dei risultati del training. Il modello finale deriva dai modelli addestrati in precedenza presi in base ai loro pesi. A differenza del Bagging qua i modelli vengono eseguiti in sequenza.

CAPITOLO 7

Bibliografia e sitografia

[1] P. Watzlawick, J. B. Bavelas, and D. D. Jackson, *Pragmatics of human communication: A study of interactional patterns, pathologies and paradoxes.* WW Norton & Company, 2011.

[2] A. Carullo, A. Anibaldi, A. Astolfi, A. Atzori, V. Cennamo, and G. Zito, *A new paradigm of effective communication based on Voice Shapes.* Politecnico di Torino, Interago Academy, 2019.

[3] T. Wang, Y.-c. Lee, and Q. Ma, "Within and across-language comparison of vocal emotions in mandarin and english," *Applied Sciences*, vol. 8, p. 2629, 12 2018.

[4] Y. Xu, "Speech melody as articulatorily implemented communicative functions," *Speech Communication*, vol. 46, no. 3, pp. 220 – 251, 2005. Quantitative Prosody Modelling for Natural Speech Description and Generation.

[5] C. E. Williams and K. N. Stevens, "Emotions and speech: Some acoustical correlates," *The Journal of the Acoustical Society of America*, vol. 52, no. 4B, pp. 1238–1250, 1972.

[6] S. Warhurst, C. Madill, P. McCabe, S. Ternström, E. Yiu, and R. Heard, "Perceptual and acoustic analyses of good voice quality in male radio performers," *Journal of voice*, vol. 31, no. 2, pp. 259–e1, 2017.

[7] J. Hillenbrand, R. A. Cleveland, and R. L. Erickson, "Acoustic correlates of breathy vocal quality," *Journal of Speech, Language, and Hearing Research*, vol. 37, no. 4, pp. 769–778, 1994.

[8] Y. Maryn, N. Roy, M. De Bodt, P. Van Cauwenberge, and P. Corthals, "Acoustic measurement of overall voice quality: A meta-analysis," *The Journal of the Acoustical Society of America*, vol. 126, no. 5, pp. 2619–2634, 2009.

[9] B. P. Bogert, "The quefrency alanysis of time series for echoes," *Time series analysis*, pp. 209–243, 1963.

[10] U. D. of Phonetics and Linguistics, "Introduction to computer programming with matlab," 2019. [Online; accessed on 17 October 2019].

[11] S. Angra and S. Ahuja, "Machine learning and its applications: A review," in *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*, pp. 57–60, March 2017.

[12] A. Pant, "Sworkflow of a machine learning project," 2019. [Online; accessed on 18 October 2019].

[13] E. D. Science, "Overfitting in machine learning: what it is and how to prevent it." [Online; accessed on 29 October 2019].

[14] Mathworks, "Classifier options," 2019. [Online; accessed on 14 October 2019].

[15] L. Palopoli, "Valutazione delle prestazioni," 2012. [Online; accessed 12 November 2019].

[16] M. Claesen, F. De Smet, J. A. Suykens, and B. De Moor, "Fast prediction with svm models containing rbf kernels," *arXiv preprint arXiv:1403.0736*, 2014.

[17] D. Cai, X. He, and J. Han, "Training linear discriminant analysis in linear time," in *2008 IEEE 24th International Conference on Data Engineering*, pp. 209–217, April 2008.

[18] P. Gupta, "Decision trees in machine learning," 2017. [Online; accessed 14-October-2019].

[19] J. Brownlee, "Linear discriminant analysis for machine learning," 2016. [Online; accessed on 14 October 2019].

[20] "Linear vs. quadratic discriminant analysis – comparison of algorithms," 2018. [Online; accessed 14 October 2019].

[21] A. Pant, "Introduction to logistic regression," 2016. [Online; accessed on 14 October 2019].

[22] R. Gandhi, "Support vector machine — introduction to machine learning algorithms," 2018. [Online; accessed on 15 October 2019].

[23] Saptashwa, "Support vector machine: Kernel trick; mercer's theorem," 2019. [Online; accessed on 15 October 2019].

[24] O. Harrison, "Machine learning basics with the k-nearest neighbors algorithm," 2018. [Online; accessed on 15 October 2019].

[25] T. Acharya, "Advanced ensemble classifiers," 2019. [Online; accessed on 15 October 2019].

Ringraziamenti

Arrivato al termine della stesura di questa tesi e, soprattutto, dei cinque anni di università ritengo sia doveroso fare dei ringraziamenti.
Desidero innanzitutto ringraziare i ”due Alessio”, Alessio Carullo e Alessio Atzori. Il primo, relatore di questa tesi, per l’avermi spronato, anche indirettamente, nell’andare sempre un po’ oltre la prima soluzione disponibile trasmettendomi il suo spirito da ricercatore; il secondo, compagno di scrivania durante questo lavoro di tesi, per il grande aiuto e supporto dato nelle fasi operative e per i consigli su come procedere con la ricerca.
Il vero ringraziamento va fatto però a chi ha supportato senza sosta questo percorso di studi: i miei genitori. Gli Eandi non sono quel tipo di famiglia in cui ci si dimostra affetto in modo palese, con gesti evidenti; siamo più abituati a muoverci ”nell’ombra” e da lì supportarci a vicenda. Per questo motivo voglio ringraziare i miei genitori, per l’essere un’ombra, un qualcosa che sa precedere ma che sa anche seguire nella consapevolezza che, anche quando non la vedi, l’ombra è lì, accanto a te, alcune volte più vicina e definita, altre più lontana e sfocata ma è presente ogni giorno ed ogni ora anche quando la luce è poca. Li ringrazio, inoltre, per avermi insegnato alcuni valori come il rispetto, l’onestà e l’ascolto; sono quei valori che reputo più importanti nella vita di un uomo e che fanno, davvero, la differenza nella società.
Un altro ringraziamento, ultimo ma non meno importante, vorrei farlo a chi in questi anni è rimasto accanto a me, ma anche a chi non l’ha fatto perché tutti, in un modo o nell’altro, hanno lasciato un segno nel mio percorso contribuendo a creare la persona che sono adesso. Vorrei in particolare ringraziare quella persona che tempo fa mi disse la frase che è diventata un po’ la guida di quanto ho fatto fino ad ora. Grazie per avermi insegnato a combattere, perché *se vuoi, puoi.*