Università degli Studi di Padova
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
Laurea Magistrale in Ingegneria Informatica

TESI DI LAUREA MAGISTRALE

Modelli per il riconoscimento automatico delle curve di pre-enfasi e post-enfasi applicate ai documenti sonori

Laureando:
Fabio Casamento
Matricola: 1104412

Relatore:
Prof. Sergio Canazza

Correlatore:
Ing. Niccolò Pretto

Anno Accademico 2016-2017

Alla mia famiglia

Ringraziamenti

Un ringraziamento speciale va alla mia famiglia, per avermi dato la possibilità di raggiungere questo traguardo e per avermi supportato.
A Ylenia, per essere stata sempre presente, soprattutto nei momenti complicati.
Ai miei amici, per tutti i momenti trascorsi a riderci o suonarci su.
A Niccolò, Valentina e ai Prof. Canazza, Rodà e Di Nunzio, per la disponibilità, la pazienza, il supporto e gli utili consigli.
A tutti coloro che sono stati presenti, in un modo o nell'altro, e mi hanno accompagnato lungo questo percorso straordinario.

Sommario

Le equalizzazioni di pre-enfasi e post-enfasi vengono applicate ai documenti sonori durante il processo di registrazione e riproduzione su supporti magnetici essenzialmente per rendere lo spettro del segnale *equally likely to overload* e per migliorare il rapporto *Signal-to-Noise*. A partire dagli anni Cinquanta, per promuovere l'industria musicale sono stati istituiti e promossi diversi standard relativi al processo di registrazione magnetica. Tuttavia, la tendenza dei produttori musicali, sia attuali che dei decenni passati, di non allegare ai supporti informazioni tecniche sul processo di registrazione, causa problemi relativi alla scelta della corretta equalizzazione da adottare in fase di riproduzione dei contenuti.
In questo elaborato viene descritto un esperimento che dimostra come sia possibile applicare tecniche di *machine learning* per decidere, in modo automatico, se la curva di post-enfasi applicata a documenti sonori registrati su nastro magnetico sia quella giusta o meno. Il *dataset* su cui sono stati testati i classificatori è costituito da campioni che si distinguono per velocità di riproduzione (7.5 o 15 ips), contenuto (traccia silenziosa o rumore bianco), pre-equalizzazione applicata (standard CCIR o standard NAB) e post-equalizzazione applicata (standard CCIR o standard NAB). Alcuni tra i modelli predittivi testati sono in grado di separare i campioni equalizzati in modo corretto (curve di pre-enfasi e post-enfasi dello stesso standard) da quelli equalizzati in modo errato (curva di pre-enfasi NAB e curva di post-enfasi CCIR, o viceversa) con errore di cross-validazione pari a zero e *accuracy* nella predizione di nuovi dati del 100%.

Indice

Capitolo 1

Introduzione

La nascita e l'evoluzione di tecnologie per la registrazione di documenti sonori su supporti magnetici hanno interessato gran parte del Novecento. Il primo dispositivo in grado di registrare e riprodurre segnali sonori risale alla fine dell'Ottocento ed è stato brevettato da Valdemar Poulsen. Questo evento ha posto le basi di un percorso di ricerca e sviluppo che è iniziato a intensificarsi alle fine degli anni Venti e ha raggiunto il suo picco negli anni Ottanta, periodo in cui la registrazione digitale su supporti ottici ha iniziato ad assumere importanza maggiore. Tuttavia, fino alla fine degli anni Novanta, la registrazione magnetica è stata la tecnologia più utilizzata al mondo per registrare suoni, immagini e altre tipologie di informazione.
Nel processo di registrazione sonora su supporti magnetici, quasi fin dalle prime sperimentazioni si è capito quanto fosse importante utilizzare tecniche di pre-enfasi e post-enfasi, che consistono essenzialmente nell'applicare al segnale di input una curva di pre-equalizzazione in fase di registrazione e una curva di post-equalizzazione uguale e opposta in fase di riproduzione. Applicando correttamente queste tecniche, si può ottenere una distribuzione più equa dell'energia spettrale del segnale audio, migliorando il rapporto segnale-rumore e rendendo un'esperienza d'ascolto di qualità superiore.
A partire dagli anni Cinquanta, sono stati introdotti diversi standard di pre-enfasi e post-enfasi per la registrazione magnetica, in modo tale da favorire la fruizione dei contenuti su dispositivi provenienti da produttori differenti e promuovere, in questo modo, l'industria musicale. Tuttavia, l'introduzione e la diffusione di diversi standard, in America e in Europa, e la tendenza degli addetti ai lavori di non allegare ai supporti informazioni tecniche riguardo al processo di registrazione, hanno reso problematica la scelta della corretta curva di post-equalizzazione. Infatti, molti dispositivi di riproduzione permettono di scegliere tra due o più curve di post-enfasi, ma non conoscendo quale sia lo standard utilizzato in pre-enfasi, la scelta diviene del tutto arbitraria. Purtroppo, in diversi contesti - specie in ambito musicologico - applicare la giusta post-equalizzazione non è una scelta trascurabile, in quanto serve a garantire un ascolto dei contenuti qualitativamente coerente e ottimale.
Per questi motivi sono stati effettuati degli esperimenti per capire se fosse possibile utilizzare tecniche di *machine learning* per comprendere, in modo automatico, se l'equalizzazione applicata a un documento sonoro in fase di riproduzione sia quella corretta. Verrà mostrato come, in determinati contesti, sia possibile rico-

noscere automaticamente le equalizzazioni applicate in post-enfasi e postefasi con un'*accuracy* del 100%.

In questo capitolo verranno discussi:

- alcuni tra gli eventi principali che hanno segnato la storia della tecnologia di registrazione magnetica (Sezione 1.1);
- modelli fisici e tecnologie alla base della registrazione magnetica (Sezione 1.2);
- modello di *Analog Tape Recorder* (ATR) utilizzato durante gli esperimenti, *Studer A810* (Sezione 1.3);
- tecniche de pre-enfasi e post-enfasi e standard più diffusi (Sezione 1.4);
- problema relativo alla scelta della corretta curva di post-enfasi (Sezione 1.5).

1.1 Storia della registrazione su nastro magnetico

L'ipotesi più accreditata sulla nascita della registrazione magnetica è quella secondo cui spetta al danese Valdemar Poulsen il merito di aver realizzato, nel 1892, il primo dispositivo in grado di registrare e riprodurre segnali sonori (in questo caso si trattava di conversazioni telefoniche), il *telegrafono*. Tuttavia, nonostante il dispositivo fu premiato all'Esposizione Universale di Parigi del 1900 e, negli anni immediatamente successivi, fu notevolmente migliorato, la registrazione magnetica venne quasi dimenticata per più di vent'anni. Infatti, il primo nastro magnetico simile agli attuali fu brevettato da J. A. O'Neill nel 1927, e Fritz Pfleumer realizzò il primo registratore a nastro l'anno successivo.
A partire dalla seconda metà degli anni Trenta è iniziato un intenso processo di evoluzione della tecnologia per la registrazione magnetica, che ha raggiunto il suo apice a cavallo tra anni Settanta e anni Ottanta, periodo in cui le tecniche di registrazione digitali su supporti ottici cominciavano ad assumere rilievo sempre maggiore.
A seguire alcune fra le tappe principali della storia della registrazione magnetica [1].

1888 Nella rivista *The Electrical World* viene pubblicato "Some Possible Forms of the Phonograph". Si tratta di un articolo in cui l'ingegnere americano Oberlin Smith suggerisce che un filo o nastro di materiale magnetico impregnato di polvere magnetica potrebbe registrare e riprodurre il suono elettromagneticamente.

1898 L'ingegnere Danese Valdemar Poulsen dimostra con successo che la registrazione magnetica sia possibile.

1900 Poulsen esibisce il *Telegraphone* (Figura 1.1) all'Esposizione di Parigi e si aggiudica il Grand Prix.

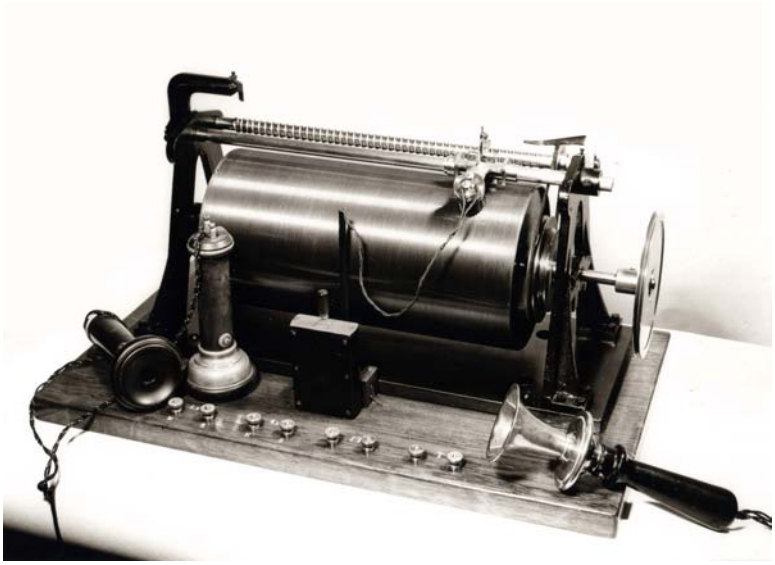

Figura 1.1: Telegraphone di Poulsen, 1900.

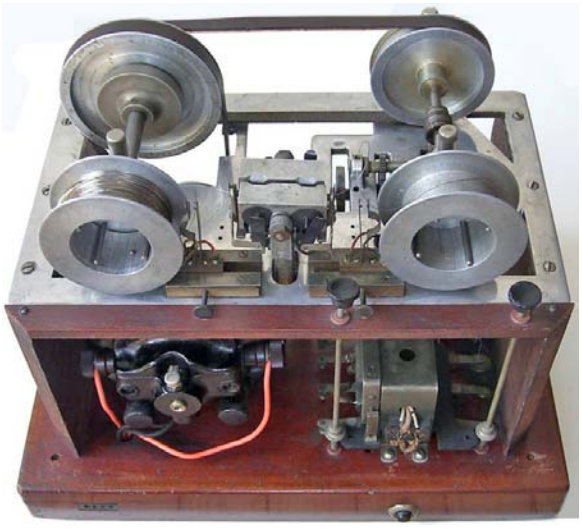

Figura 1.2: Telegraphone della Emerson Record, 1915.

1903 La compagnia American Telegraphone, capitalizzata a $5.000.000, comincia a produrre registratori a filo, che rappresentano la prima tecnologia di registrazione magnetica ad essere commercializzata.

1906 Supporti di registrazione composti da strati sottili sono stati placcati su un substrato.

1907 Viene brevettato il *DC bias* da Pederson e Poulsen.

1913 Un gruppo di ricercatori del U.S. Naval Research Laboratory descrivono una forma di *AC bias* utilizzata per aumentare la sensibilità e ridurre il rumore in un "Radio Telegraph System".

1927 In un brevetto americano di J. A. O'Neill viene suggerito un nastro di registrazione di materiale magnetico in polvere.

Figura 1.3: Blattnerphone di Blattner, 1930.

1928 In un brevetto tedesco di Fritz Pfleumer viene suggerito un registratore su nastro di materiale magnetico in polvere, poi venduto con il nome *Magnetophon K1* e riconosciuto come il primo registratore a nastro commercializzato della storia.

1930 In Inghilterra, Ludwing Blattner introduce il *Blattnerphone* (Figura 1.3), un registratore a nastro d'acciaio. La compagnia di Blattner fu acquistata dalla British Marconi, che utilizzò il *Blattnerphone* come punto di partenza per produrre una macchina più sofisticata, poi utilizzata dalla BBC, il *Marconi-Stille*.

1932 Il giorno di Natale Viene trasmesso in *broadcast* un discorso di Re George V registrato sulla macchina a nastro d'acciaio della BBC.
L'Echophone Company di Karl Bauer inventa il *Dailygraph*, un registratore a filo per dettatura e registrazione telefonica.

1933 Viene inventato il *Textophone*, una versione migliorata del Dailygraph.

1935 La German General Electric Co. (A.E.G.) annuncia il *Magnetophon K1* (Figura 1.4), che viene esposto all'Esposizione Radio di Berlino del 1935. La principale caratteristica del magnetofono era l'uso di nastro di plastica largo 6.5 mm, rivestito all'inizio con polvere di ferro carbonile, successivamente con magnetite. Inizialmente la macchina era stata progettata per dettatura e aveva una qualità del suono mediocre.
Nei Laboratori Bell viene inventato *Mirror for Voice*, un registratore a nastro che utilizzava un breve loop di nastro in metallo.

1936 Al quartier generale della I.G. Farben (Ludwigshafen, Germania) viene registrata la London Philharmonic Orchestra.

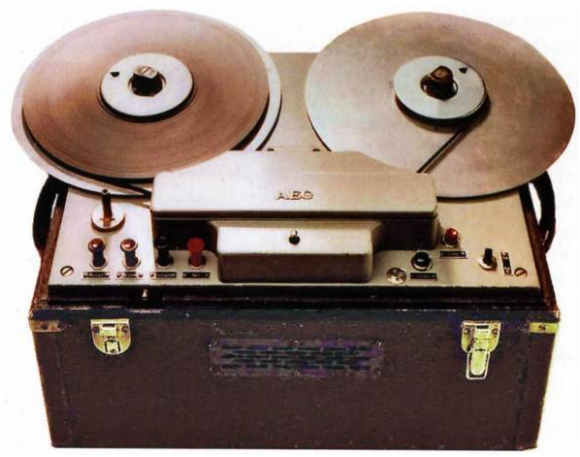

Figura 1.4: AEG Magnetophon K1, 1935.

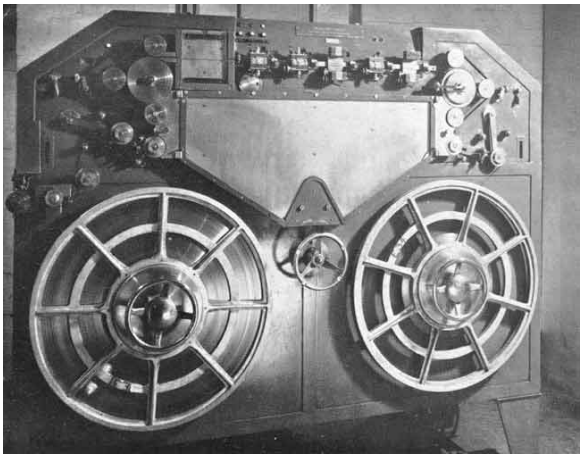

Figura 1.5: BBC Marconi-Stille, 1938.

1937 Ai Laboratori Bell si riesce a raggiungere la velocità di registrazione di 16 pollici al secondo ed effettuare registrazioni del suono ad alta qualità grazie all'introduzione della *vicalloy*, un nuovo materiale per nastri, e al miglioramento delle testine di registrazione.

1938 Viene descritto un registratore a nastro d'acciaio utilizzato per le trasmissioni BBC (miglioramento del *Marconi-Stille*, Figura 1.5).
Un nuovo registratore a filo introduce delle innovazioni che porteranno al registratore *Model 50* della ARF (Armour Research Foundation).
In Giappone viene pubblicato un articolo su esperimenti di cancellazione e polarizzazione con AC.

1939 La Brush Development, in Ohio, commercializza il *Soundmirror*, la prima macchina per registrazioni su nastro progettata e commercializzata negli Stati Uniti.
In Germania, le stazioni radio iniziano ad utilizzare i registratori a nastro.

1940 Registratori a filo sono utilizzati dall'Armour Research Foundation e dall'Armour Institute of Technology per fare ricerca nell'ambito delle rilevazioni marine, delle classi di linguaggio e delle registrazioni musicali. Viene sviluppato il registratore a filo *Model 50* (Figura 1.6).

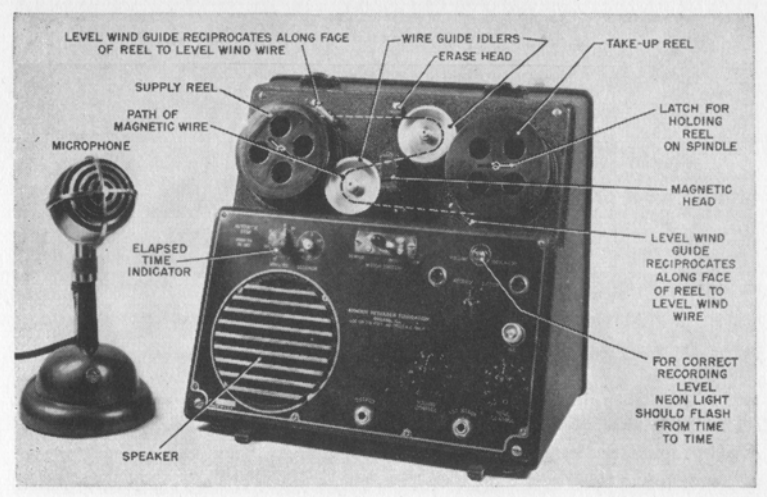

Figura 1.6: Model 50 della Armour Research Foudation, 1940.

La Western Electric Co. sviluppa il *Mirrophone*, registratore a nastro in grado di registrare per un minuto.

1941 Nei registratori a filo *Model 50* viene utilizzato il *bias* ad alta frequenza.

1942 Vengono introdotte nuove versioni del *Model 50*.
L'Armour Research Foundation continua la ricerca e lo sviluppo finalizzati al miglioramento dei dispositivi di registrazione magnetica.

1943 L'acciaio inossidabile per registrazioni a filo e a nastro rappresenta il principale progresso nei mezzi metallici, in quanto magneticamente superiore ai precedenti, non si arrugginisce e non si corrode.

1944 L'Armour Research Foundation introduce i registratori a filo tascabili alimentati a batteria.

1945 Vengono messi in vendita i primi registratori a filo per uso domestico.

1946 L'Armour Research Foundation sviluppa l'ossido di nastro magnetico, che viene universalmente utilizzato nei nastri magnetici fino al giorno d'oggi.
Brush Development introduce *Model BK401 SoundMirror* (Figura 1.7), che utilizzava, come supporto, nastro di carta da 1/4 di pollice rivestita con ossido a bassa coercitività.

Membri dello staff di Armour Reserach Foundation fondano a Chicago la Magnecord Corporation, un'azienda per la produzione di registratori professionali.
La Magnecord Co. introduce il *Magnecord Model SD-1*, un registratore master a filo con risposta dai 35 ai 1500 Hz e flutter sotto lo 0.1% per*broadcast audio recording*.

1947 La Minnesota Mining and Manufacturing produce e commercializza il primo nastro di ossido magnetico ad alta coercitività.
I registratori a filo per utilizzo domestico vengono prodotti in massa e tra questi spicca il *Webster-Chicago Model 80* (Figura 1.8), che diviene il più popolare per basso costo, portabilità e relativa affidabilità.

Figura 1.7: BK401 SoundMirror, della Brush Development, 1946.

Figura 1.8: Webster-Chicago Model 80, 1947.

Figura 1.9: Magnecord Model PT6, 1948.

La Armour Research Foundation dimostra i concetti di registrazione a nastro stereo, a tre canali, a due canali e binaurale.

1948 La Armour Foundation commercializza *Eicore*, un registratore a nastro a due direzioni economico. Simile e commercializzato lo stesso anno è *Crestwood*, prodotto in Canada.
Viene introdotto il *Magnecord Model PT6-A* (Figura 1.9), che diventa il più popolare registratore professionale per convenienza e qualità del suono.

1948-49 Vengono commercializzati numerosi registratori per uso domestico.
Viene dimostrata la stampa a contatto dei nastri magnetici e diventa effettiva la stampa del nastro *master* sui nastri copia.

1950-59 Vengono sviluppati i primi prototipi e i primi prodotti commerciali per la registrazione video su nastro magnetico.
Sony introduce il primo registratore a nastro stereofonico per uso domestico, il *TC-551*.
Ampex Corp. introduce registratore videotape *Quadruplex* (Figura 1.10).

1960-69 Ampex rende disponibili i primi registratori dati a banda larga a 2 MHz, 120 in/sec.
Viene introdotta la cassetta compatta Philips.
Durante una mostra NAB, Ampex mostra un registratore per usi commerciali che utilizza dischi flessibili.
Sony commercializza registratori video in bianco e nero a nastro per uso casalingo.
Alla fine degli anni Sessanta, la produzione di nastro magnetico supera quella di ogni altro materiale.

1970-79 Eastman Kodak Company mette in commercio videocamere, proiettori e pellicole per scopi amatoriali basati su supporto magnetico.
Sony introduce il sistema a video cassette *U-Matic* (Figura 1.11).

Vengono commercializzati i *floppy disk* con capacità di lettura e scrittura.
A metà degli anni '70 vengono introdotti i registratori video per uso domestico *Betamax* (Figura 1.12), prodotto da Sony Corporation, e *V-cord*

Figura 1.10: Ampex Quadruplex videotape recorder, 1956.

Figura 1.11: Sony U-Matic SP tape recorder, 1972.

Figura 1.12: Sony Betamax tape recorder, 1975.

Mark-H VTR, prodotto da Toshiba e Sanyo. Ben presto i video registratori vengono prodotti in massa.
Alla fine degli anni Settanta viene commercializzato il *Video Home System* (VHS), sviluppato dalla compagnia Giapponese JVC.

1980-89 Nel mercato americano, Hitachi, JVC e altri produttori introducono videocamere con supporto videocassette da 8 mm e altri dispositivi per la registrazione video magnetica.

1.2 Fondamenti di registrazione magnetica

Fino a circa la metà degli anni Novanta, la registrazione magnetica rappresentava la tecnologia più utilizzata al mondo per registrare suoni, immagini e altre tipologie di informazione. Nel corso della seconda metà del Novecento, diverse aziende di tutto il mondo hanno sviluppato e lanciato sul mercato prodotti per registrare e riprodurre varie tipologie di segnali, utilizzando nastro magnetico come supporto. Sono state sviluppate e rese disponibili migliaia di soluzioni tecnologiche pensate per soddisfare sia esigenze professionali di entità come radio, televisioni, case discografiche, che esigenze di privati (tipicamente amatori e appassionati di registrazione audio/video) o legate al mondo della ricerca e della sperimentazione.
In questa sezione verranno descritti brevemente i modelli fisici e tecnologici alla base della registrazione magnetica.

1.2.1 Campo magnetico e induzione elettromagnetica

Il magnetismo rappresenta uno dei fenomeni fisici più osservati dall'uomo, nonché tra i più affascinanti. La proprietà di attirare la limatura di ferro, mostrata da alcuni minerali di ferro, era già nota nel VII secolo a.C., ma i primi tentativi di caratterizzare il fenomeno e metterlo in relazione con l'elettrostatica furono fatti dal fisico britannico W. Gilbert nel XVI [2].
Le azioni magnetiche, che si osservano, per esempio, quando si avvicinano due calamite, sono il risultato dell'interazione tra cariche in moto. Adottando la rappresentazione tramite un campo, diciamo che l'azione magnetica è dovuta al fatto

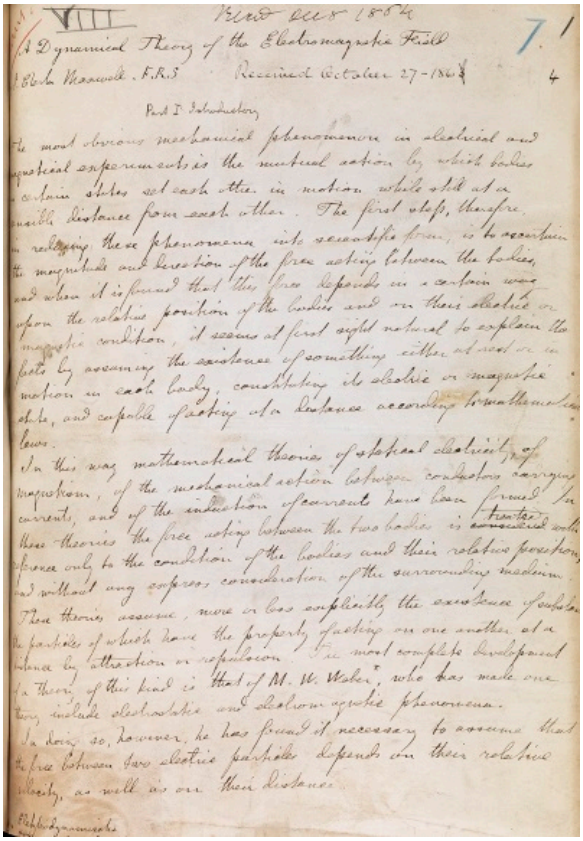

Figura 1.13: Pagina del manoscritto "A Dynamical Theory of the Electromagnetic Field", di James Clerk Maxwell, 1865.

che un sistema di cariche in moto genera un campo magnetico B e che l'altro sistema di cariche in moto risente di una forza in quanto immerso in B.
Gli studi effettuati nel XIX secolo circa i fenomeni legati all'elettromagnetismo convergono nelle Equazioni di Maxwell (comparse per la prima volta nel testo *A Dynamical Theory of the Electromagnetic Field* pubblicato da James Clerk Maxwell nel 1865, Figura 1.13), che, insieme alla forza di Lorentz, costituiscono le leggi fondamentali che governano l'interazione elettromagnetica [3].

Per quanto riguarda la registrazione magnetica, il fenomeno fisico più interessante è rappresentato dalla legge di Faraday ξ dell'induzione elettromagnetica, secondo la quale, ogni qual volta il flusso del campo magnetico $\Phi(B)$, concatenato con un circuito, varia nel tempo, nel circuito si genera una forza elettromotrice indotta (definita come l'integrale di un campo elettrico lungo una linea chiusa), data dall'opposto della derivata del flusso rispetto al tempo [2]:

$$\xi = -\frac{d\Phi(B)}{dt}.$$

Questo modello mostra come la variazione di un campo magnetico nel tempo generi un campo elettrico non conservativo, che in alcuni dispositivi può dar luogo a una forza elettromotrice e ad una corrente in un circuito chiuso. Il fenomeno dell'induzione elettromagnetica è il principio su cui si basano molte tecnologie, da motori e generatori elettrici a testine fonografiche, microfoni dinamici e pick-up per chitarra.

1.2.2 Materiali magnetici

Sia μ la permeabilità magnetica di un materiale generico. Vale la seguente formula:

$$B = \mu H,$$

dove B rappresenta una misura dell'induzione magnetica, mentre H del campo magnetico applicato al materiale. Come si può intuire dall'equazione, la permeabilità magnetica rappresenta una misura dell'attitudine di un materiale a magnetizzarsi in presenza di campo magnetico.
A seconda del valore della permeabilità, si possono distinguere tre categorie di sostanze magnetiche.

- **Paramagnetiche**: sostanze con permeabilità appena superiore a uno; a livello atomico, sono costituite da dipoli magnetici che tendono ad allinearsi leggermente con il campo magnetico a cui sono sottoposti.

- **Diamagnetiche**: sostanze con permeabilità leggermente inferiore a 1; a livello atomico, sono costituite da dipoli magnetici che tendono debolmente ad allinearsi in verso opposto rispetto a quello del campo magnetico al quale vengono sottoposti. Appartengono a questa categoria acqua, la maggior parte delle sostanze organiche e alcuni metalli (oro, argento, rame, piombo, mercurio).

- **Ferromagnetiche**: sostanze con permeabilità molto maggiore di uno; a livello atomico, sono costituite da dipoli magnetici che subiscono molto l'influenza del campo magnetico a cui vengono sottoposti; entro certe temperature, queste sostanze sono in grado di mantenere nel tempo la magnetizzazione, fungendo quindi da magneti. Appartengono a questa categoria ferro, nichel, cobalto e altri metalli e leghe metalliche.

- **Amagnetiche**: sostanze prive di proprietà ferromagnetiche.

Ciclo di isteresi Supponiamo di disporre di un solenoide, all'interno del quale può passare corrente elettrica, che avvolge un nucleo di ferrite smagnetizzato (punto O di Figura 1.14, $B = 0$ e $H = 0$). Supponiamo, inoltre, di poter modificare l'intensità di corrente che passa attraverso il solenoide, e, quindi, di poter modificare il campo magnetico H generato mediante induzione elettromagnetica. Infine, sia B la misura del campo magnetico del nucleo. Si può osservare che aumentando l'intensità corrente, e quindi l'intensità di H, B aumenta, tracciando, come si osserva in Figura 1.14, la curva detta di prima magnetizzazione, fino a raggiungere un valore di magnetizzazione massimo Bs (magnetizzazione di saturazione). A questo punto, diminuendo l'intensità di corrente, si osserva che la curva di magnetizzazione del nucleo non percorre lo stesso percorso. In particolare, quando la corrente si annulla, nel nucleo rimane un campo magnetico residuo Br, detto anche rimanenza (*retentivity*). Questo significa che il nucleo ha acquistato una magnetizzazione permanente.
È possibile eliminare la magnetizzazione residua applicando una corrente negativa fino a raggiungere il punto in cui $B = 0$ e $H = Hc$ (forza coercitiva o *coercitivity*), e, diminuendo ulteriormente H, magnetizzare il nucleo negativamente fino al suo

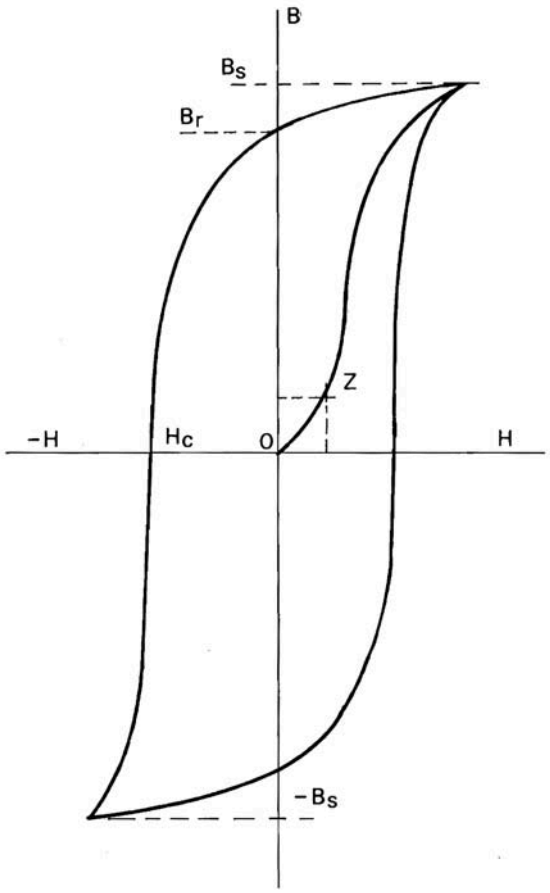

Figura 1.14: Curva di isteresi elettromagnetica.

valore di saturazione $-Bs$.
Si può nuovamente far crescere l'intesità del campo magnetico indotto e ripetere il ciclo: quello che si ottiene è la curva di Figura 1.14, detta *curva di isteresi magnetica*. Da notare che la curva di per sé non passa dall'origine: infatti, per riportare il nucleo di ferrite alle condizioni iniziali ($B = 0$, $H = 0$) occorre sottoporlo ad una temperatura, detta di *Curie*, che dipende dal tipo di materiale.
La curva di isteresi contiene tutte le informazioni circa lo stato di magnetizzazione di un materiale.

1.2.3 Componenti base di un registratore magnetico

Nonostante la grande varietà di registratori a nastro introdotti negli anni, si può affermare che ciascuno di essi sia costituito essenzialmente da due componenti:

- una parte meccanica, che si occupa di far scorrere il nastro, cercando di mantenere costante la velocità di scorrimento;
- una parte elettromagnetica, costituita da blocchi di testine, chiamate anche *tape heads*, che possono registrare, riprodurre o cancellare il segnale memorizzato sul supporto.

Nei paragrafi successivi verranno discussi le componenti e il funzionamento della parte elettromagnetica.

	Mumetall	Vacodur	Recovac 100	Vitrovac
Permeabilità iniziale	50.000	8.000	40.000	100.000
Permeabilità max	120.000	40.000	100.000	400.000
Saturazione (Gauss)	8.000	9.000	5.000	5.500
Coercitiva (A/m)	0,012	0,04	0,015	0,004
Durezza Vickers (HV)	105	250	220	1.000

Figura 1.15: Caratteristiche delle principali leghe usate per le teste magnetiche.

Nucleo delle testine Le testine di cancellazione, registrazione e riproduzione, essenzialmente sono costituite da un nucleo avvolto da una bobina elettrica. Il nucleo tipicamente è in ferro, per le sue buone proprietà elettro-acustiche, o in ferrite, che rispetto al materiale precedente ha una maggiore resistenza all'usura. Tuttavia negli anni sono state sperimentate molte tipologie di materiale, da ceramica a leghe sintetiche, sia per migliorare le proprietà magnetiche dell'intero sistema registrazione e riproduzione, sia per limitare le problematiche legate alla corrosione.
In generale, affinché i materiali destinati a formare il nucleo magnetico si prestino bene per la registrazione, è importante che abbiano basse retentività e forza coercitiva. Purtroppo a questa condizione, detta di *addolcimento magnetico*, tipicamente corrisponde il cosiddetto *addolcimento meccanico*, che si traduce in rapido deterioramento della testina man mano che viene sfregata con il nastro. Per questo motivo, negli anni le industrie del settore hanno effettuato numerosi sforzi per creare leghe che sempre più si adattassero agli scopi. Una delle leghe che ha avuto più successo è la *recovac 100*, prodotta dalla Vacuumschmelze GMBH, che si caratterizza per essere molto resistente dal punto di vista meccanico e per avere le stesse caratteristiche magnetiche del *mumetal*, un materiale già da prima molto utilizzato come nucleo per le testine. Un altro materiale sviluppato dalla Vacuumschmelze GMBH che ha delle proprietà molto interessanti è il *vitrovac*, le cui caratteristiche sono elencate nella tabella di Figura 1.15 tratta da [4], insieme a quelle di altri materiali.

Supporti per la registrazione magnetica Per quanto concerne il supporto per la registrazione, la qualità della sua risposta di lunghezza d'onda lunga (bassa-frequenza) è proporzionale alla *retentivity* (Br nella funzione di isteresi), mentre la sua risposta di lunghezza d'onda breve (alta-frequenza) migliora al crescere della *coercitivity* (Hc nella funzione di isteresi). Quindi, in generale si vorrebbe che i materiali utilizzati come supporto per la registrazione magnetica siano caratterizzati da alte *retentivity* e *coercitivity*.
Negli anni sono state introdotte diverse tipologie di supporto per la memorizzazione, da fili e dischi a cinture, nastri, ecc.. Ad oggi il mezzo più utilizzato è rappresentato da un nastro generalmente composto da uno strato sottile di ossido magnetico polverizzato, legato chimicamente a uno strato in cloruro di polivinile (PVC) e coperto da un rivestimento protettivo.
La tabella in Figura 1.16, tratta da [1], mostra le proprietà magnetiche di alcuni supporti di registrazione introdotti negli anni man mano che l'arte della registrazione progrediva.

Molto importanti sono anche le caratteristiche meccaniche del supporto. Per esempio, un buon nastro dovrebbe avere una buona stabilità dimensionale e alta resistenza a trazione, corrosione, umidità e temperatura. Inoltre, dovrebbe

RECORDING MATERIAL	YEAR INTRODUCED (APPROX)	COERCIVITY	RETENTIVITY
Carbon steel, cold worked, wire or tape	1900	10	10,800
Tungsten steel tape	1936	27	12,200
Vicalloy tape	1937	260	7,800
High carbon steel, heat treated	1940	30	10,100
420 Stainless steel	1943	60	7,000
18-8 Stainless steel	1942	270	2,250
Plated bronze wire, alloy of 80% Co, 20% Ni	1944	220	10,400
NiCo plated discs	1946	250	10,000
Carbonyl iron coated tape	1935	5	100
Low coercivity gamma Fe_2O_3 impregnated tape	1942	90	112
Low coercivity gamma Fe_2O_3 coated tape	1942	110	300
Low coercivity magnetite Fe_3O_4 coated tape	1946	120	375
Hyflux (metal particle) coated tape	1946	500	2,500
High coercivity magnetite Fe_3O_4 coated tape	1946	340	800
High coercivity gamma Fe_2O_3 coated tape	1946	250	1,200
Metal particle coated tape (Fe, Co, Ni)	1962	830	2,800
Cobalt-doped gamma Fe_2O_3 tape I	1965	290	960
Chromium dioxide coated tape	1966	550	1,400
Cobalt doped gamma Fe_2O_3 coated tape II	1978	600	1,100
Cobalt adsorbed iron oxide coated tape	1978	580	1,550
Metal particle high coercivity tape	1979	1100	3,100
Berthollide iron oxide coated tape	1979	550	1,400

Figura 1.16: Proprietà magnetiche di alcuni supporti di registrazione.

essere compatibile con le dimensione delle testine, avere una superficie liscia, minimizzare il rumore ed essere prodotto a costi ragionevoli.

1.2.4 Processo di registrazione magnetica

Il principio di funzionamento del nastro magnetico può essere schematizzato come quello di un trasformatore elettrico in cui l'avvolgimento primario è rappresentato dalla *record head*, il secondario è rappresentato dalla *repro head* e la comunicazione tra le due componenti avviene indirettamente mediante il nastro magnetico, il quale memorizza il segnale elettromagnetico indotto.
Essenzialmente, durante il processo di registrazione, il segnale elettrico in ingresso viene utilizzato per polarizzare le particelle magnetiche di cui è costituito il nastro, sfruttando il fenomeno dell'induzione elettromagnetica. Il nastro magnetico, grazie alle sue proprietà, è in grado di mantenere la polarizzazione nel tempo. In fase di riproduzione, lo scorrimento di un nastro magnetico polarizzato genera un campo magnetico variabile che, all'interno della testina di riproduzione, induce un campo elettrico, che, a sua volta, rappresenta il segnale di lettura.

Corrente di Bias Il sistema di registrazione e riproduzione magnetica porta con sé delle non linearità dovute principalmente alla natura delle testine e del nastro. Queste non linearità influenzano principalmente la risposta in frequenza, causando distorsione armonica e penalizzando l'intera dinamica del sistema.
Sia $B(H)$ la funzione di trasferimento di un materiale ferromagnetico, che è possibile ricavare dalla curva di isteresi magnetica del materiale stesso (B rappresenta la misura del livello di magnetizzazione del materiale al variare del campo magnetico indotto H); è possibile dividere la curva $B(H)$ in tre zone:

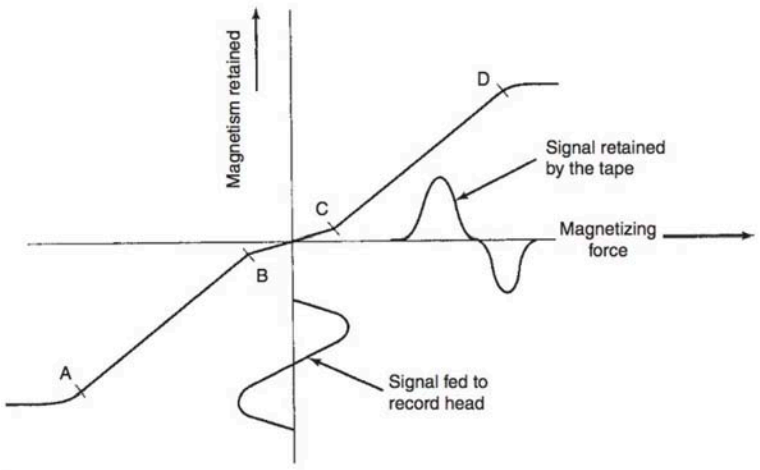

Figura 1.17: La figura mostra come la funzione di trasferimento dei materiali ferromagnetici utilizzati nel processo di registrazione tenda a distorcere il segnale in input.

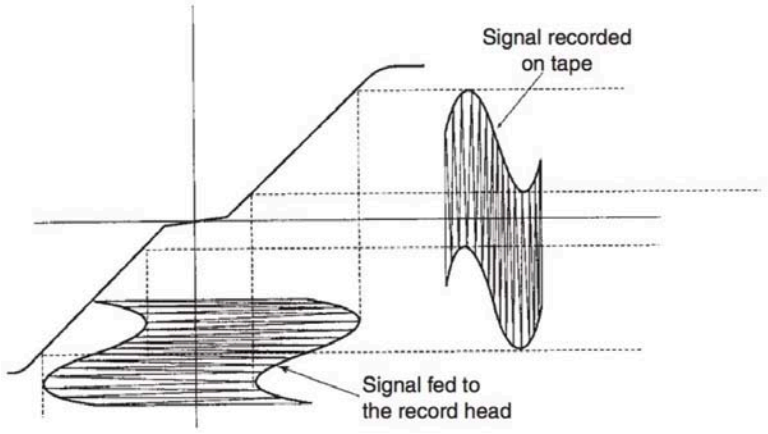

Figura 1.18: La figura mostra l'effetto dello spostamento del segnale in zona lineare grazie all'introduzione della corrente di Bias.

- parte iniziale, non lineare;
- parte centrale, lineare;
- zona di saturazione.

Purtroppo, il processo di magnetizzazione del supporto tenderebbe a operare in zona di saturazione. Per cui, l'isteresi magnetica intrinseca ai materiali ferromagnetici genererebbe una forte distorsione del segnale (si veda esempio in Figura 1.17).

Per risolvere questo problema viene introdotta la corrente di Bias o di Pre-Magnetizzazione, che ha il compito di spostare il segnale in zona lineare (Figura 1.18). Questa corrente si caratterizza per andamento sinusoidale, ampiezza media molto più grande rispetto a quella del segnale da registrare e frequenza che normalmente varia tra i 75 kHz e i 150 kHz, quindi, molto più elevata della banda del segnale audio di registrazione. Il contributo della corrente di Bias viene ignorato dalla testina di lettura, per via delle limitazioni fisiche della stessa, che si comporta, quindi, come un filtro passa-basso.

Figura 1.19: Registratore analogico a nastro, modello *Studer Revox A77*, 1967.

Figura 1.20: Registratore analogico a nastro, modello Studer J37s.

1.3 Studer A810

La Studer è un'azienda fondata in Svizzera nel 1948, specializzata in progettazione e costruzione di strumenti audio avanzati per studi di registrazione e broadcaster. A partire dagli anni Cinquanta, la Studer è divenuta il punto di riferimento europeo per la registrazione magnetica professionale. Uno dei prodotti di maggior successo dell'azienda fu il Revox A77 (Figura 1.19), introdotto alla fine degli anni Sessanta, che ad oggi è considerato una pietra miliare nella storia della registrazione a nastro.

La Studer si è specializzata anche nella produzione di registratori multitraccia, come il modello J37s (Figura 1.20), rilasciato nel 1964 e utilizzato tre anni dopo per la registrazione dell'album dei Beatles *Sgt. Pepper's Lonely Hearts Club Band*. Hanno avuto molto successo anche i modelli della serie A, come lo Studer A80 (Figura 1.21), utilizzato dai Queen ai *Mountain Studios*, e lo Studer A820 (Figura 1.22).

Quello che ad oggi è uno dei modelli più popolari per applicazioni di *broadcasting* e televisive, negli studi di registrazione, negli archivi e negli istituti scienti-

Figura 1.21: Registratore analogico a nastro, modello Studer A80.

Figura 1.22: Registratore analogico a nastro, modello Studer A820.

fici, nonché il modello utilizzato negli esperimenti descritti nei capitoli successivi, è lo Studer A810 (Figura 1.23). Si tratta di un registratore magnetico professionale introdotto nel mercato negli anni Ottanta, che si è distinto per buone performance, costruzione robusta, flessibilità e un fattore di forma relativamente compatto. In seguito sono elencate alcune tra le caratteristiche più rilevanti dello Studer A810 [5].

- Applicazioni mono, 2-canali o stereo, con o senza pannello VU-meter.
- Opera sia verticale che in orizzontale.
- Modalità *Low Speed* e *High Speed*: nella prima, 2 di tre velocità del nastro disponibili (3.75, 7.5 e 15 pollici al secondo o ips) sono selezionabili dal pannello frontale. Impostazioni standard: 7.5/15 ips. Nella seconda modalità, 4 velocità (3.75, 7.5, 15 e 30 ips) selezionabili dal pannello frontale con selettore rotante.
- Input e output bilanciati e *floating*, disponibili con o senza trasformatori input/output.
- Possibilità di scegliere equalizzazione tra NAB e CCIR (a 7.5 e 15 ips).
- *Tape bias switch* per due tipi di nastro con dati di calibrazione diversi.
- Pulsanti di selezione dell'output: INP (input), REP (riproduci) e SYNC.
- Pannello VU-meter con *switch* SAFE/READY, registrazione e riproduzione, controlli del livello, pulsanti per bypassarre il controlli di livello. Possibilità di scegliere tra monitoraggio VU (*Volume Unit*) o PPM (*Peak Programme Meter*).
- Selezione voltaggio di linea: 100, 120, 140, 200, 220, 240 VAC $\pm$10%, a 50 o 60Hz.
- Terminali per connettere *fader start*, *controller* remoto parallelo e controllo di velocità.
- Monitor *speaker* incorporato nella cover del trasporto del nastro.
- Timer elettronico del nastro con *display real-time*.

1.4 Tecniche di pre-enfasi e post-enfasi

Durante il processo di registrazione su supporto magnetico, il segnale di input tende a perdere molta energia alle alte frequenze. Questo fenomeno è dovuto prevalentemente alla natura della *record head* e per contrastarlo bisognerebbe aumentare notevolmente la *driving current*. Per questo motivo, nel sistema vengono inseriti dei filtri costituiti da serie di capacitori, in modo tale da equalizzare lo spettro del segnale, enfatizzando un preciso *range* di frequenze, che dipende dai valori di capacità e di resistenza dei filtri.

Figura 1.23: Registratore analogico a nastro, modello Studer A810.

Tuttavia, nella registrazione del parlato e della musica, il segnale di input potrebbe non avere uno spettro piatto, e quindi alcune frequenze potrebbero venire sovraccaricate, mentre altre potrebbero subire delle perdite, degradando la qualità sonora complessiva. Quando lo spettro è noto, è logico enfatizzare le frequenze più deboli, prima di registrare, in modo tale da rendere piatto lo spettro finale e rendere tutte le frequenze *equally likely to overload* (ELO). Questa equalizzazione viene considerata una sorta di pre-enfasi e dovrebbe essere bilanciata, durante la riproduzione, con un'equalizzazione uguale e opposta o di post-enfasi. Di fatto è dimostrato che, se utilizzate in modo appropriato, le tecniche di pre-enfasi e post-enfasi migliorano il rapporto segnale-rumore complessivo del sistema, rendendo l'esperienza d'ascolto finale di qualità superiore [1].
Tra i primi ad utilizzare queste tecniche, nel 1929 vi furono Silvian, Dunn e White dei laboratori Bell, i quali pubblicarono un paper su esperimenti effettuati durante la registrazione di un'esecuzione orchestrale. Nel loro esperimento, utilizzarono la curva della risposta in frequenza complessiva media del segnale audio (Figura 1.24) per ricavare le curve di equalizzazione da utilizzare in pre-enfasi e post-enfasi (Figura 1.25), soprattutto per migliorare la risposta alle alte frequenze.

In generale, per ottenere miglioramenti dello spettro del segnale alle basse frequenze e ridurre il ronzio di fondo, dovuto principalmente alla corrente di alimentazione, in pre-enfasi si applica un *boost* nell'intorno dei 30 Hz, che poi viene controbilanciato in post-enfasi.

1.4.1 Standard di pre- e post-equalizzazione

Storicamente sono stati introdotti diversi standard di equalizzazione di pre-enfasi e post-enfasi, soprattutto per rendere dischi in vinile e nastri magnetici

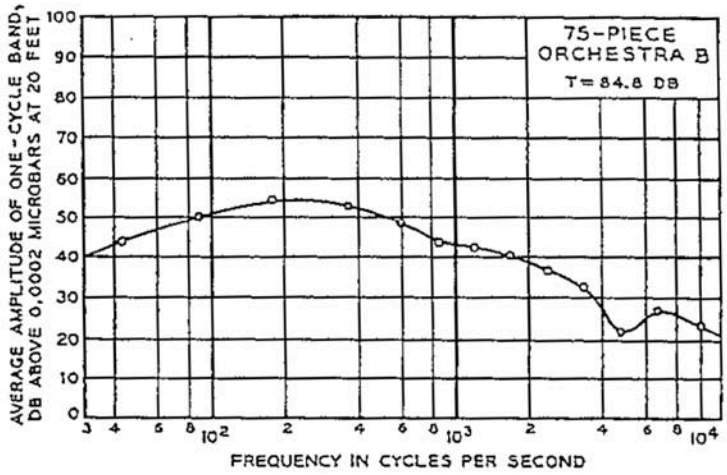

Figura 1.24: Risposta in frequenza complessiva media del segnale audio registrato durante l'esperimento di Sivian, Dunn, White, 1929.

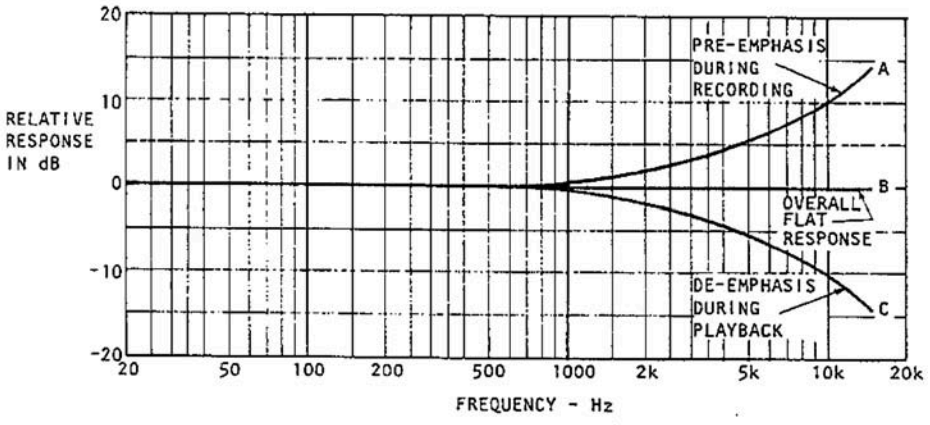

Figura 1.25: Curve di pre-enfasi e post-enfasi ricavate durante l'esperimento di Sivian, Dunn, White, 1929.

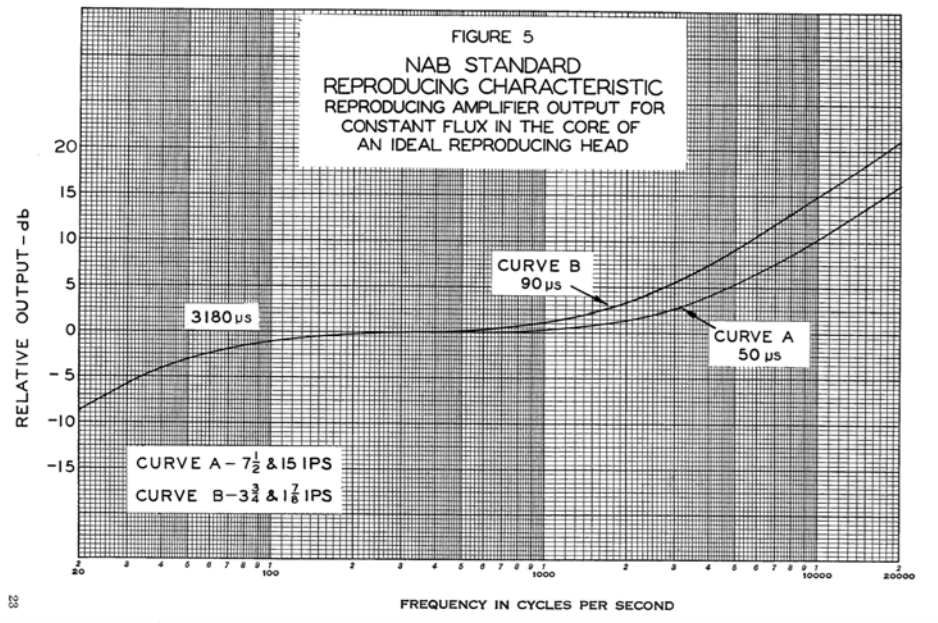

Figura 1.26: Curva di post-equalizzazione NAB.

riproducibili su dispositivi provenienti da produttori differenti e promuovere, in questo modo, l'industria musicale.
In America, per quanto riguarda la registrazione su nastro magnetico, lo standard più diffuso si è rivelato quello stabilito dalla National Association of Broadcasters (NAB), istituita nel 1941. Quest'associazione è nata con lo scopo di promuovere delle buone pratiche che gli ingegneri che lavorano nell'industria del suono dovrebbero utilizzare nelle varie fasi della registrazione. Ne hanno preso parte diverse organizzazioni internazionali, per poter assicurare il massimo grado di coordinazione tra le industrie del settore, permettere intercambiabilità e allo stesso tempo sfruttare gli ultimi avanzamenti tecnologici. Lo standard, oltre a elencare delle buone pratiche e stabilire le specifiche delle componenti che fanno parte del sistema di registrazione su nastro magnetico, stabilisce la caratteristica dell'equalizzazione da applicare in pre-enfasi ed in post-enfasi (Figura 1.26).

L'Europa, a sua volta, per motivi del tutto analoghi, lavorò ai propri standard. Nel 1960 il *Comitè Consultatif International pour la Radio* (*CCIR*) sviluppò un altro standard, più tardi adottato dall'IEC. La curva di equalizzazione imposta da questo standard filtra le alte frequenze, senza però andare a modificare le basse. In Figura 1.27 viene mostrata la curva CCIR/IEC di registrazione su nastro magnetico.

Un altro importante standard americano, che in passato riguardava soprattutto l'industria della registrazione su dischi in vinile, è lo standard *RIAA* (*Recording Industry Association of America*), introdotto negli anni Cinquanta. Come mai prima, in quegli anni ogni azienda del settore prevedeva l'utilizzo delle proprie equalizzazioni. Di conseguenza, dispositivi di riproduzione provenienti da produttori diversi e utilizzati a velocità di riproduzione diverse, applicavano, durante il *playback*, post-enfasi diverse. Questo fatto, il più delle volte, si traduceva in qualità dell'esperienza di ascolto inferiore a quella che effettivamente si otterrebbe utilizzando la post-equalizzazione opportuna.
La curva di equalizzazione RIAA, che rappresenta uno standard per le registra-

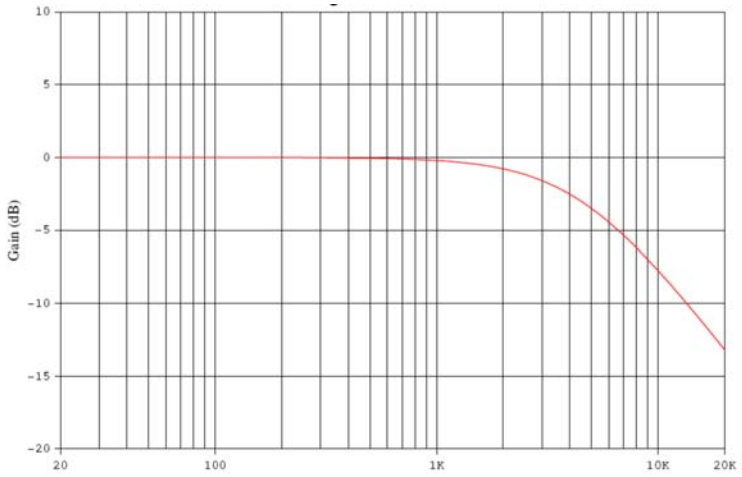

Figura 1.27: Curva di pre-equalizzazione CCIR.

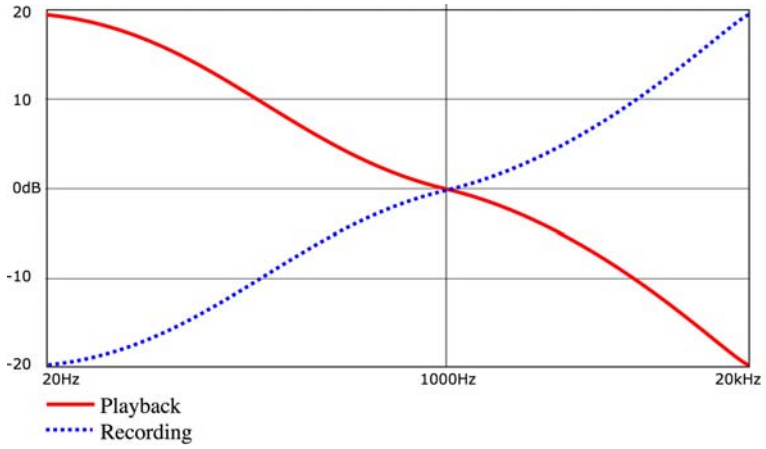

Figura 1.28: Curva RIAA di pre- e post-equalizzazione.

zioni su vinile, prevede un'attenuazione sulle basse frequenze e un potenziamento sulle alte in fase di registrazione e un'equalizzazione uguale e opposta in fase di riproduzione. Anche in questo caso, l'obiettivo è ottenere una risposta in frequenza finale piatta e un rapporto segnale-rumore più elevato.
Nel 1976 l'*IEC* (*International Electrotechnical Commission*) ha proposto una modifica alla curva di equalizzazione RIAA che consiste nell'aggiunta di un polo a circa 20 Hz alla funzione di trasferimento, in modo tale da ridurre l'uscita subsonica dell'amplificatore *phono*. Le curve RIAA di pre-enfasi e post-enfasi sono illustrate in Figura 1.28.

Gli standard di pre-enfasi e post-enfasi sono definiti anche al variare della velocità di registrazione e di riproduzione. Infatti, per velocità diverse, gli standard in genere prevedono alcune variazioni nelle rispettive curve.

1.5 Scelta della corretta curva di post-enfasi

È facile intuire come la storica introduzione di più standard per le equalizzazioni da applicare in fase di registrazione e riproduzione in sistemi analogici, possa causare dei problemi relativi alla scelta della corretta curva di post-enfasi. A peggiorare le cose, infatti, vi è la tendenza dei produttori musicali, sia attuali

che dei decenni passati, di non allegare ai supporti informazioni tecniche riguardanti il processo di registrazione.
Molti dispositivi di riproduzione di documenti sonori analogici registrati su supporti magnetici permettono di scegliere di applicare una tra due o più standard di post-equalizzazione. Ad oggi, che si tratti di privati appassionati dell'audio registrato in analogico, o di archivi musicali, molti enti sparsi nel mondo dispongono di documenti sonori dei quali non si conosce la pre-equalizzazione utilizzata durante il processo di registrazione.
Anche se in alcuni casi risulta poco rilevante la scelta di applicare, in riproduzione, una curva di post-enfasi piuttosto che un'altra, spesso è molto importante scegliere quella corretta. Sebbene applicare la giusta post-enfasi di per sé garantisca un ascolto qualitativamente ottimale e coerente dei contenuti sonori, talvolta si rivela fondamentale, soprattutto in contesti legati alla ricerca musicologica.
Un primo esperimento legato alla percezione umana delle curve di equalizzazione applicate ai documenti sonori analogici viene descritto in [6]. L'esperimento, effettuato presso il Centro di Sonologia Computazionale di Padova, si basa sull'utilizzo di forme di comparazione uditiva e dimostra la differente percezione di campioni registrati e riprodotti con curve di equalizzazione corretta ed errata, caratterizzando statisticamente i risultati ottenuti. Tuttavia, l'utilizzo dell'orecchio come unico strumento per riconoscere se l'equalizzazione applicata sia quella giusta si è dimostrato non sufficiente per decidere con certezza.
Queste sono le argomentazioni che motivano l'esperimento descritto nei capitoli successivi. Ciò che, essenzialmente, si è cercato di capire è se sia sia possibile applicare tecniche e algoritmi di apprendimento automatico (*machine learning*) a file ottenuti digitalizzando registrazioni sonore su nastro magnetico, al fine di riconoscere automaticamente quali siano le equalizzazioni di pre-enfasi e post-enfasi applicate.
Ci si è limitati allo studio della possibilità di distinguere tra le equalizzazioni di pre-enfasi e post-enfasi stabilite dagli standard CCIR e NAB, che sono i più utilizzati nella registrazione su nastro magnetico rispettivamente in Europa e negli Stati Uniti. A tal fine sono stati effettuati diversi esperimenti ed è risultato che, in determinate condizioni, effettivamente è possibile riconoscere automaticamente l'equalizzazione applicata, che sia quella corretta (pre-enfasi e post-enfasi entrambe NAB o entrambe CCIR) o quella errata (pre-enfasi NAB e post-enfasi CCIR oppure pre-enfasi CCIR e post-enfasi NAB), raggiungendo, con alcune tipologie di modelli predittivi, un'*accuracy* nella predizione di nuovi dati del 100%.

La parte rimanente di questo elaborato è strutturata nel seguente modo:

- Capitolo 2: descrizione delle tecniche di *machine-learning* utilizzate;
- Capitolo 3: descrizione dell'impostazione dell'esperimento, dalla creazione del dataset alla struttura del codice per l'apprendimento automatico;
- Capitolo 4: elenco e discussione dei risultati ottenuti;
- Capitolo 5: conclusioni e possibili sviluppi futuri;
- Capitolo 6: bibliografia di riferimento;
- Capitolo 7: elenco delle figure.

Capitolo 2

Machine learning: tecniche utilizzate

Con il termine *machine learning*, o *apprendimento automaticato* si indica la scienza che si occupa del problema di "costruire programmi per computer che migliorino automaticamente con l'esperienza" [7]. Un'altra definizione, probabilmente la prima, indica il *machine learning* come la scienza nata con lo scopo di "dare ai computer l'abilità di apprendere senza essere programmati esplicitamente" [8]. Negli ultimi anni, grazie allo sviluppo di tecniche di *machine learning* sempre più efficaci, è stato possibile sviluppare auto *self-driving*, rendere più pratico il riconoscimento vocale, migliorare la *web search*. Altre applicazione legate all'apprendimento automatico interessano la finanza computazionale (valutazione della solvibilità delle controparti, *trading* algoritmico), la biologia computazionale (rilevazione di tumori, identificazione farmacologica e sequenziazione del DNA), la produzione di energia (previsione dei prezzi e della curva di carico), il *processing* automatico del linguaggio naturale, il riconoscimento di immagini e suoni, il *marketing* per il suggerimento automatico, l'analisi di *Big Data* [9].
Essenzialmente, l'obiettivo degli algoritmi di *machine learning* è quello di analizzare dati al fine di ottenere informazioni utili, senza utilizzare modelli matematici noti a priori.
È possibile individuare due principali tipologie di tecniche di *machine learning*:

- tecniche di **apprendimento supervisionato**, che si pongono l'obiettivo di adattare modelli matematici alla classificazione di dati di cui sono note le classi di appartenenza;

- tecniche di **apprendimento non supervisionato**, che cercano di individuare *pattern* e strutture ricorrenti in insiemi di dati di cui non sono note le classi di appartenenza.

In altre parole, le tecniche di apprendimento supervisionato servono a costruire modelli predittivi partendo dalla conoscenza di dati di *input*, chiamati anche *occorrenze*, e dei rispettivi *output*, o classi di appartenenza. Tali modelli, una volta creati, vengono utilizzati per classificare dati di cui non si conoscono le classi di appartenenza.
Le tecniche di apprendimento supervisionato più famose sono [9]:

- *Decision Tree*;

- *Support vector machine* (SVM);

- *Naive Bayes*;

- *K-Nearest Neighbor*;

- Analisi discriminante;

- Reti neurali.

Le tecniche di apprendimento non supervisionato permettono, invece, di individuare nei dati *pattern* nascosti o strutture intrinseche. Le più famose sono:

- *K-Means Clustering*;

- *Hierarchical Clustering*;

- Misture di Gaussiane (tramite algoritmo *Expectation Maximization*);

- Catene di Markov nascoste;

- Mappe auto-organizzanti (SOM);

- Reti neurali;

- Reti neurali supervisionate con apprendimento competitivo LVQ.

In questo capitolo verranno introdotte le tecniche di machine learning utilizzate nell'esperimento descritto nel capitolo successivo, in particolare:

- per apprendimento supervisionato (Sezione 2.1), *Decision Tree* (Sezione 2.1.1), *K-Nearest Neighbors* (Sezione 2.1.3) e *Support Vector Machine* (Sezione 2.1.2);

- per apprendimento non supervisionato (Sezione 2.3), *K-Means Clustering* (Sezione 2.3.1) e *Hiearchical Clustering* (Sezione 2.3.2) .

Per maggiori approfondimenti si rimanda ai testi [7], [10] e [11].

2.1 Apprendimento supervisionato

L'obiettivo degli algoritmi di *apprendimento supervisionato* è quello di costruire modelli predittivi in grado di stabilire, autonomamente, per ogni *input* il relativo *output*, partendo da coppie note di valori di *input* e di *output*. In letteratura, gli *input* sono spesso chiamati *predictor*, ma anche variabili indipendenti o *feature*. Gli *output* sono detti *response* o *variabili dipendenti* e possono essere misure sia *quantitative* (numeri), sia *qualitative* (colori, forme, ecc).
Il *task* di predizione può assumere diversi nomi a seconda della tipologia di *output*; infatti, se gli *output* sono di tipo quantitativo allora si parla di *regression*, altrimenti, per *output* di tipo qualitativo si parla di *classification*. Le due tipologie di *task* hanno molto in comune e, in particolare, i modelli su cui si basano entrambe possono essere visti come funzioni di approssimazione.
Le variabili di tipo qualitativo sono tipicamente rappresentate in forma di codici

numerici; se le categorie, o classi, sono due (per esempio "successo" e "fallimento"), per codificarle di solito si utilizzano singoli numeri, come 0 e 1, oppure -1 e 1. Tali codici sono spesso chiamati *target*.
Il *task* di apprendimento può essere descritto nel seguente modo: dato un vettore di *input* X a N componenti, il modello dovrà essere in grado di effettuare una buona predizione, che chiameremo Y e avrà N componenti, dove la componente y_i di Y rappresenta la *response* relativa al *predictor* x_i di X.
Per costruire un modello predittivo occorrono:

- un *training set* S, cioè un insieme di N coppie di valori (x_i, y_i), $S = \{(x_1, y_1), ...(x_N, y_N)\}$, tali per cui y_i rappresenta l'*output* relativo a x_i, per $i = 1, ..., N$;

- un algoritmo di apprendimento supervisionato, il quale costruirà le regole di predizione partendo dai valori di S, durante una fase chiamata *addestramento*, o *training*, del modello.

Nei tre paragrafi successivi, verranno descritti gli approcci di apprendimento supervisionato utilizzato nell'esperimento di Capitolo 3.

2.1.1 Decision Tree

I metodi di partizione basati su *tree*, o *albero*, dividono lo spazio delle *feature* in un insieme di rettangoli e poi associano a ciascuno di essi un semplice modello (per esempio una disequazione) [10]. Tali metodi sono concettualmente semplici e molto potenti.
Per fare un esempio, consideriamo un problema di regressione con risposta continua Y e *input* X_1 e X_2. Il pannello in alto a sinistra di Figura 2.1 mostra una partizione dello spazio delle attraverso delle linee parallele agli assi con le coordinate. In ogni partizione possiamo modellare Y con una costante diversa.
Invece, il pannello in alto a destra di Figura 2.1 è stato ottenuto dividendo inizialmente lo spazio in due regioni, modellando la risposta con la media di Y in ogni regione e scegliendo variabile e punto di *split* in modo da ottenere il *fit* migliore. Successivamente, una o entrambe le regioni sono state divise in due ulteriori regioni e il processo è andato avanti finché non si è attivata una regola di stop. In particolare, prima si è diviso in $X_1 = t_1$, poi la regione $X_1 \leq t_1$ è stata divisa in $X_2 = t_2$ e la regione $X_1 > t_1$ è a $X_1 = t_3$. Infine, la regione $X_1 > t_3$ è stata divisa a $X_2 = t_4$, e il risultato è una partizione nelle cinque regioni R_1, R_2, R_3, R_4, R_5 mostrate in figura.
Lo stesso modello può essere rappresentato sotto forma di albero binario di decisione, o *decision tree*, come quello raffigurato nel pannello in basso a sinistra di Figura 2.1, in cui la radice rappresenta l'intero *dataset*, le osservazioni che soddisfano la condizione di ogni giunzione sono assegnate al *branch* sinistro, le altre a destra, e i nodi terminali corrispondono alle regioni R_1, R_2, R_3, R_4, R_5.
Infine, il pannello in basso a destra di Figura 2.1 è un *plot* della superficie di regressione ottenuta da questo modello.

Uno dei vantaggi nell'utilizzare alberi binari ricorsivi è la loro facile interpretabilità, in quanto un solo albero è in grado di descrivere la partizione dell'intero spazio delle *feature*.

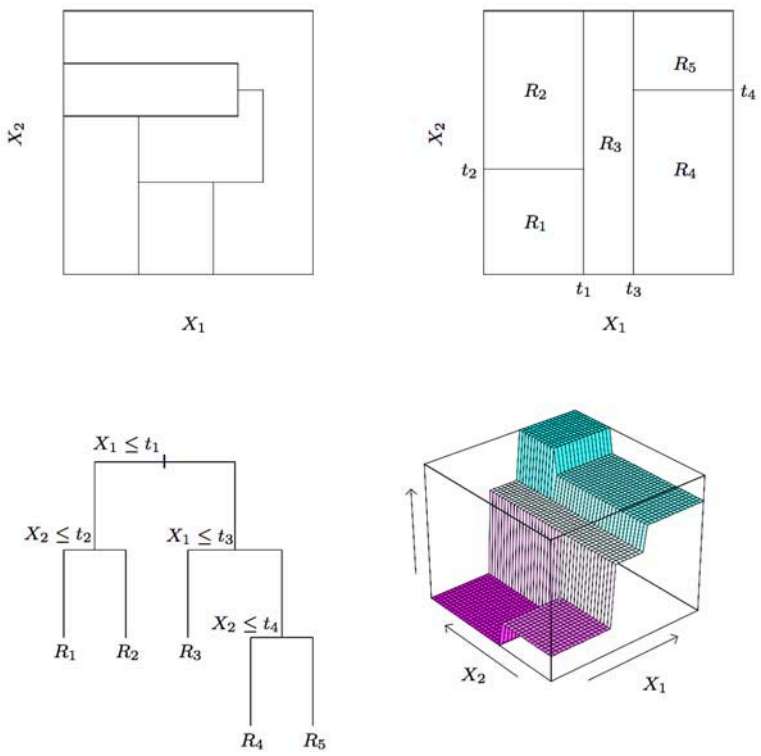

Figura 2.1: Esempio di partizione del *dataset* per la classificazione tramite *decision tree*. Il pannello in alto a destra mostra una partizione dello spazio delle *feature* a due dimensioni attraverso *splitting* binario ricorsivo, applicato ad alcuni dati. Il pannello in alto a sinistra mostra una partizione generale che non può essere ottenuta attraverso splitting binario ricorsivo. Il pannello in basso a sinistra mostra l'albero corrispondente alla partizione in alto a destra e, infine, nel pannello in basso a destra appare un *plot* della superficie di predizione.

Il problema di determinare la struttura ottimale (inclusa la scelta della variabile di *input* per ogni *split* e la corrispondente soglia) per minimizzare l'errore di somma dei quadrati residui è computazionalmente impraticabile per l'enorme numero di possibili soluzioni [11]. Ciò che si fa di solito è adottare un approccio di tipo *greedy*, cominciando con un singolo nodo radice, corrispondente all'intero spazio di *input*, e poi accrescendo l'albero aggiungendo un nodo alla volta. Ad ogni passaggio, nello spazio di *input* ci sarà un certo numero di regioni candidate per lo *split* e, per ognuna di esse, occorrerà scegliere quali delle variabili di *input* dividere e il rispettivo valore di soglia. Per effettuare una scelta congiunta ottimale di *split*, variabili di *input* e soglia, si può effettuare una ricerca esaustiva notando che la scelta ottimale di variabili predittive è data dalla media locale dei dati. Questo si ripete per tutte le possibili scelte di variabili da dividere e alla fine viene mantenuta quella che offre l'errore di somma dei quadrati residui più piccolo.
Come anticipato, per fermare l'algoritmo bisogna introdurre un'apposita regola di stop. Per esempio si potrebbe decidere di fermare la costruzione del *decision tree* nel momento in cui l'errore residuo scende al di sotto di una certa soglia. Tuttavia, poiché spesso nessuno degli *split* disponibili produce una significativa riduzione dell'errore, è pratica comune far crescere molto l'albero, utilizzando un criterio basato sul numero di *data-point* associati a nodi foglia, e poi potare l'albero risultante.
Nei problemi di regressione, la potatura, o *pruning*, di solito viene effettuata cercando di bilanciare l'errore residuo rispetto alla complessità del modello. In

particolare, sia T_0 l'albero di partenza da potare e $T \subset T_0$ un sottoalbero di T_0 che può essere ottenuto potando i nodi di T_0. Supponiamo che i nodi foglia siano indicizzati da τ, con $\tau = 1, ..., \left|T\right|$, dove il nodo foglia τ rappresenta la regione R_τ dello spazio di *input*, il quale ha N_τ *data-point*, e sia $\left|T\right|$ il numero totale di nodi foglia. La predizione ottimale per la regione R_τ è data da

$$y_\tau = \frac{1}{N_\tau} \sum_{x_n \in R_\tau} t_n \tag{2.1}$$

e il corrispondente contributo di somma dei quadrati residui è

$$Q_\tau(T) = \sum_{x_n \in R_\tau} \left\{t_n - y_\tau\right\}^2. \tag{2.2}$$

Il criterio di potatura è dato da

$$C(T) = \sum_{\tau=1}^{\left|T\right|} Q_\tau(T) + \lambda \left|T\right| \tag{2.3}$$

Il parametro λ determina il *trade-off* tra l'errore di somma dei quadrati residui complessivo e complessità del modello, misurata come il numero $\left|T\right|$ di nodi foglie, e il suo valore è scelto mediante *cross-validation*.
Per problemi di classificazione, il processo di crescita e potatura dell'albero è simile, eccetto che per l'errore di somma dei quadrati che viene rimpiazzato da una misura di *performance* più appropriata. Se definiamo $p_{\tau k}$ la proporzione dei *data-point* nella regione R_τ assegnata alla classe k, dove $k = 1, ...K$, allora due scelte comunemente usate sono la *cross-entropia*

$$Q_\tau(T) = \sum_{k=1}^{K} p_{\tau k} \ln p_{\tau k} \tag{2.4}$$

e l'*indice di Gini*

$$Q_\tau(T) = \sum_{k=1}^{K} p_{\tau k}(1 - p_{\tau k}). \tag{2.5}$$

Queste misure sono degli indici differenziabili e quindi adatti a metodi di ottimizzazione basati su gradiente.

In Figura 2.2 viene illustrato un esempio di *decision tree* tratto dalla documentazione di MATLAB.

2.1.2 Support Vector Machine

Il classificatore *Support Vector Machine*, o *SVM*, può essere utilizzato quando i dati da classificare hanno esattamente due classi [9]. L'obiettivo di SVM è di trovare l'iperpiano che, nello spazio delle *feature*, separa i *data-point* delle due classi nel miglior modo possibile, ovvero garantendo il margine più largo tra le due classi. I vettori di supporto, o *support vector*, sono i *data-point* più vicini all'iperpiano di separazione. In Figura 2.3 viene illustrata questa definizione;

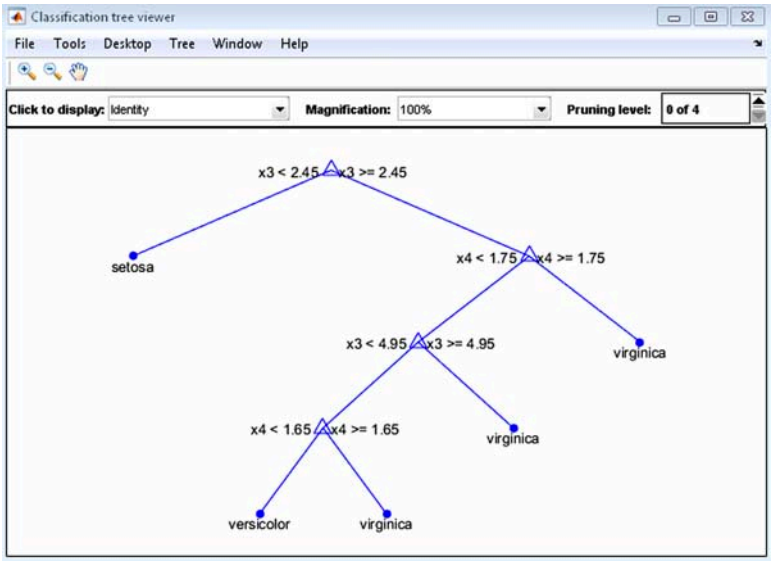

Figura 2.2: Esempio di *decision tree* tratto dalla documentazione di MATLAB.

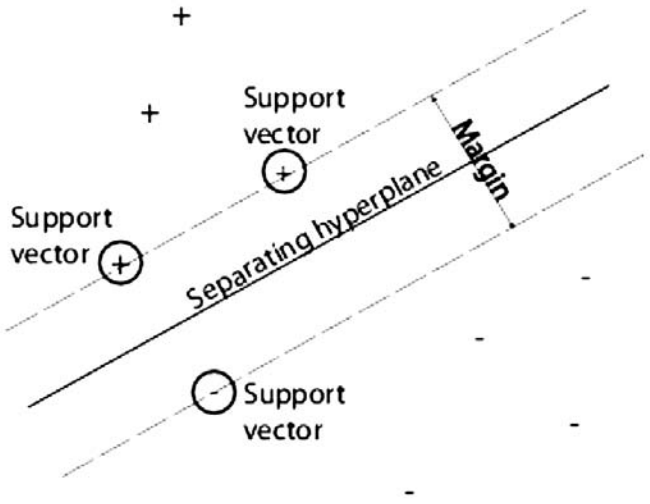

Figura 2.3: Esempio che illustra la definizione di *Support Vector*, in cui + indica i *data-point* di tipo 1 e − quello di tipo −1.

in particolare, supponendo che ciascuna osservazione possa avere *label* 1 o −1, vengono mostrati i vettori di supporto della prima classe, con il simbolo +, quelli della seconda classe, con il simbolo −, l'iperpiano di separazione e il margine.

Se x' è un vettore di osservazioni, l'iperpiano cercato è descritto dall'equazione

$$f(x) = x'\beta + b = 0, \tag{2.6}$$

dove d è la dimensione di x', $\beta \in R^d$ e b è un numero reale. L'obiettivo di SVM è quello di trovare il miglior iperpiano e, quindi, di trovare β e b che minimizzino $\|\beta\|$, in modo tale che, per tutti i data point (x_i, y_i), dove x_i rappresenta un'osservazione e y_i la rispettiva classe, valga

$$y_i f(x_i) \geq 1. \tag{2.7}$$

I vettori di supporto sono gli x_i sul confine, cioè quelli per cui vale $y_i f(x_i) = 1$. Tipicamente, il problema di minimizzare $\|\beta\|$ viene formulato in un modo equiva-

lente, i cui rispettivi costi computazionali di risoluzione sono quadratici rispetto alla dimensione dei dati.
La soluzione ottima $(\hat{\beta}, \hat{b})$ permette la classificazione di un vettore z come segue:

$$class(z) = sign(z'\hat{\beta} + \hat{b}) = sign(\hat{f}(z)), \tag{2.8}$$

dove $\hat{f}(z)$ è il punteggio di classificazione e rappresenta la distanza tra z e il confine di decisione.
Dato un insieme di dati, nel rispettivo spazio potrebbe non esistere un iperpiano in grado di separarli. In questo caso SVM può ricorrere all'utilizzo di *soft margin*, cioè iperpiani che separano la maggior parte dei dati, ma non necessariamente tutti.
Per problemi di classificazione binaria che non dispongono di un semplice iperpiano, né tanto meno di un criterio vantaggioso di separazione, esiste un diverso approccio matematico, basato sull'utilizzo di *kernel*. L'algoritmo che ne risulta è formalmente simile alla formulazione originale di SVM, solo che la funzione *kernel* di cui si serve, che è non lineare, permette di calcolare l'iperpiano a massimo margine in uno spazio di *feature* di dimensione più elevata rispetto all'originale. La mappatura nello spazio delle *feature* comporta problemi legati all'introduzione di non linearità, mentre l'elevata dimensione dello spazio delle *feature* implica l'aumento dell'*errore di generalizzazione* (si veda 2.2).
Esiste anche un approccio SVM multiclasse, che permette di classificare dati con più di due classi. L'idea di base è quella di ridurre il singolo problema multiclasse in più problemi di classificazione binaria.
Per ulteriori dettagli su SVM si consiglia di leggere [10].

In Figura 2.4 viene mostrato un esempio di applicazione di SVM tratto dalla documentazione ufficiale di MATLAB. In questo esempio, i dati sono raffigurati mediante dei pallini colorati in uno spazio a due dimensioni e i vettori di supporto sono riconoscibili dai contorni.

2.1.3 K-Nearest Neighbors

Nearest neighbors è uno dei metodi di apprendimento automatico più semplici, sia concettualmente che dal punto di vista implementativo. Utilizza le osservazioni del *training set* T più vicine a x nello spazio degli *input* per formare $\hat{Y}$ [10]. Nello specifico, il *fit k-nearest neighbor* è definito come segue:

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} yi, \tag{2.9}$$

dove $N_k(x)$ è l'insieme dei k punti più vicini a x_i nel *training set*. Per misurare la distanza tra i punti, tipicamente si utilizza la distanza Euclidea.
Essenzialmente, l'obiettivo del classificatore è di interrogare le k osservazioni x_i più vicine a x nello spazio di *input*, e mediare le loro *response* per decidere la classe di x.
In Figura 2.5 viene illustrato il risultato della classificazione di *data-point* a due dimensioni, utilizzando un modello *15-nearest-neighbor*. In quest'esempio, $\hat{Y}$ è la porzione di punti arancioni nel *neighborhood* (insieme dei vicini), e così viene

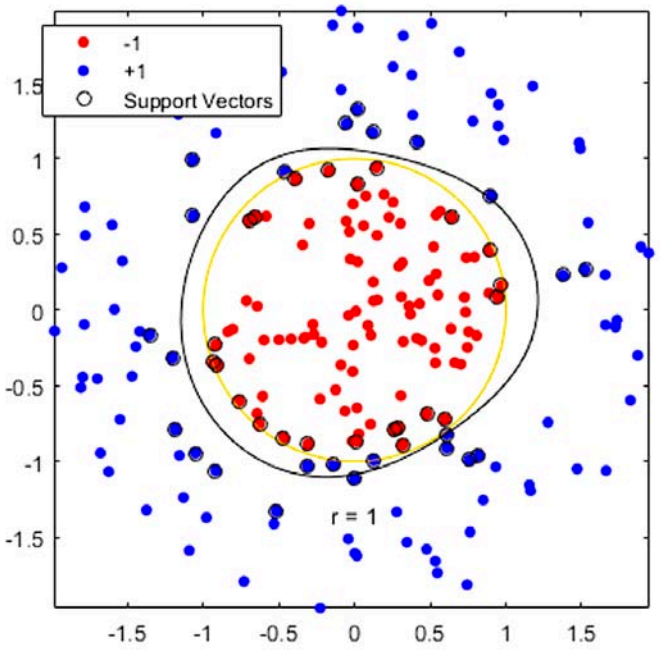

Figura 2.4: Esempio di applicazione di SVM tratto dalla documentazione di MATLAB.

assegnata la classe arancione se $\hat{Y} > 0.5$ conta il maggior numero di voti nelle regioni vicine.

In Figura 2.6, invece, viene illustrato un esempio di classificazione *1-nearest-neighbor*, in cui $\hat{Y}(x)$ corrisponde al valore y_l del punto del *training set* x_l più vicino a x. In questo caso, il calcolo delle regioni di classificazione ha costo molto basso e corrisponde alla *tassellazione di Voronoi* del *training set*. A ogni punto x_i, semplicemente viene associata la classe di appartenenza del punto di *training* più vicino. Il risultato è una separazione delle regioni più irregolare rispetto al caso precedente.

È facile intuire quanto la scelta del parametro k sia decisiva per le prestazioni complessive del classificatore. In generale, occorre scegliere k in modo tale da minimizzare l'errore di classificazione, che tende a decrescere all'aumentare di k, a costo, però, di una maggiore inconsistenza. La scelta di k varia a seconda della tipologia di dati da classificare e può essere effettuata utilizzando tecniche specifiche, come, per esempio, *cross-validation*.

Vediamo più in dettaglio come funziona un classificatore *k-nearest-neighbor*. Dato un query point x_0, ciò che viene cercato sono i k traininig-point $x_r, r = 0, ..., k$ meno distanti da x_0, il quale viene poi classificato mediando il voto dei k punti trovati.
La metrica di distanza tipicamente utilizzata è quella Euclidea, che è definita come:

$$d_{(i)} = \left\| x_{(i)} - x_0 \right\|. \tag{2.10}$$

Nonostante la sua semplicità, *k-nearest-neighbor* ha avuto molto successo in molti problemi di classificazione, dal problema del riconoscimento di cifre scritte a mano e di immagini satellitari, a quello dell'individuazione di *pattern* EKG.

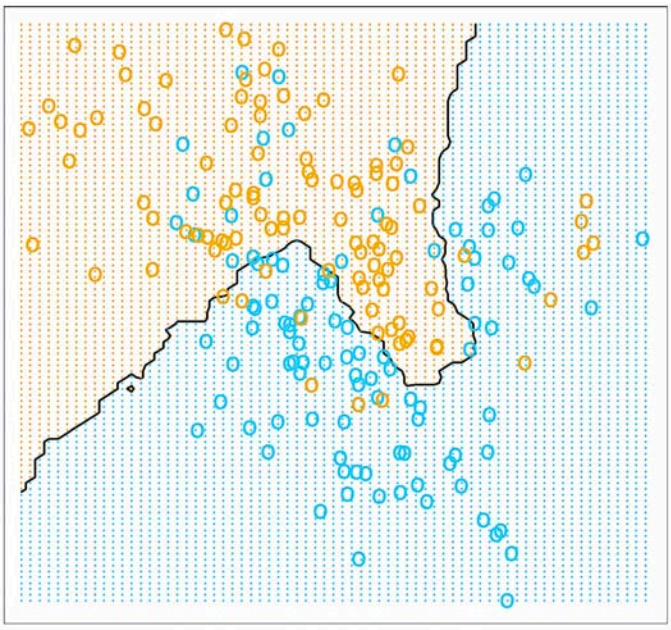

Figura 2.5: In questo esempio i dati da classificare hanno due dimensioni, possono appartenere alla classe blu o a quella arancione. Lo spazio dei *data-point* viene adattato con classificatore *15-nearest-neighbor* e la classe predetta viene quindi scelta mediando il voto dei 15 neighbor più vicini.

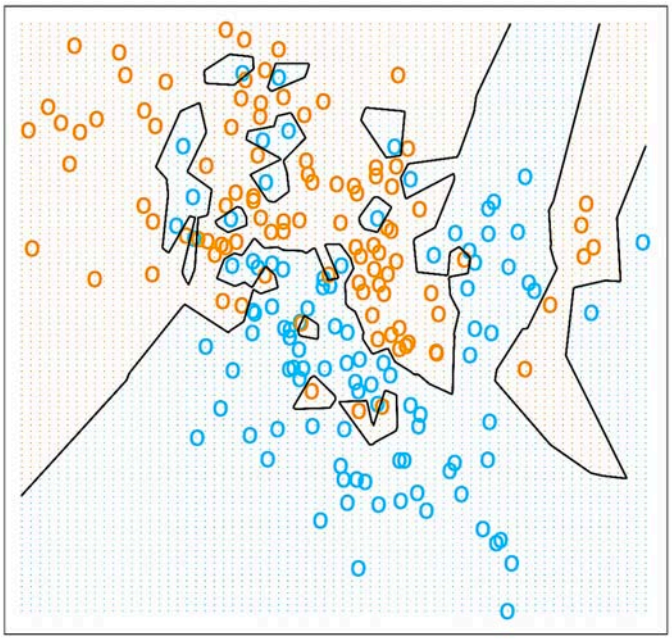

Figura 2.6: In questo esempio i dati da classificare hanno due dimensioni e possono appartenere alla classe blu o a quella arancione. Lo spazio dei *data-point* viene adattato con un classificatore *1-nearest-neighbor* e la classe predetta viene quindi scelta uguale a quella del punto più vicino a quello da classificare.

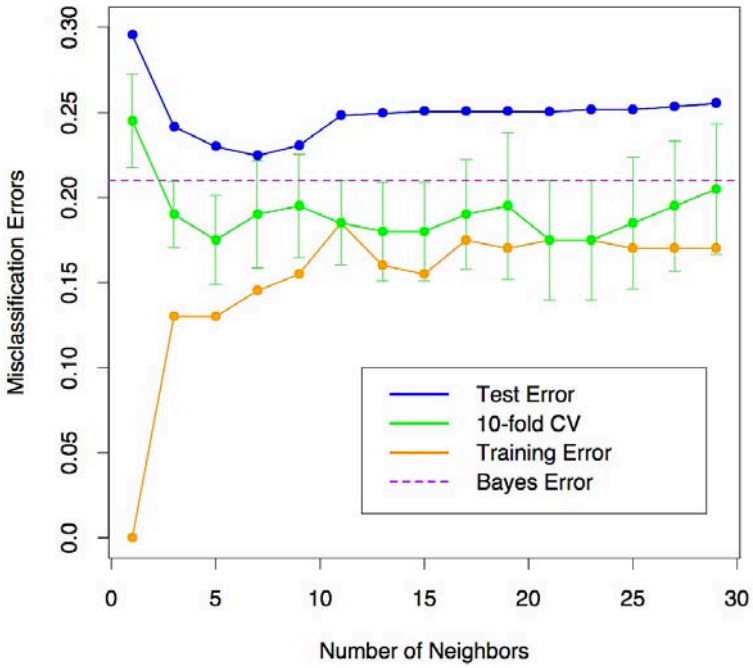

Figura 2.7: *Performance* di un classificatore *k-nearest neighbor* applicato ad un generico problema di classificazione a due classi, al variare della scelta del numero di *neighbor*.

In Figura 2.7 vengono mostrati gli errori di *training*, di *test*, di *10-cross-validation* e di *Bayes* calcolati al variare del numero di neighbor k, per un generico problema di classifazione a due classi. Per l'errore di *cross-validation*, dato che rappresenta la media calcolata su 10 numeri, viene stimata anche la deviazione standard.

2.2 Valutazione e selezione del modello

In generale, un problema di classificazione può essere affrontato utilizzando diversi modelli predittivi, e la scelta del modello da utilizzare, ovviamente, ricade sul più performante. Quindi, un passo fondamentale nell'utilizzo dei modelli predittivi, riguarda la valutazione delle loro *performance*, intesa come la capacità di classificare correttamente un *dataset* indipendente.
Supponiamo di avere una variabile target Y, relativa a vettore di *input* X, e un modello predittivo $\hat{f}(X)$ addestrato mediante il *training set* T. La misura dell'errore tra Y e $\hat{f}(X)$, che indichiamo con $L(Y, \hat{f}(X))$, è chiamata funzione di perdita, o *loss function*, e tipicamente viene calcolata mediante errore quadratico o errore assoluto:

$$L(Y, \hat{f}(X)) = \begin{cases} (Y - \hat{f}(X))^2, & \text{errore quadratico} \\ \left|Y - \hat{f}(X)\right|, & \text{errore assoluto} \end{cases} \tag{2.11}$$

Il *test error* Err_T, o errore di generalizzazione, è l'errore di predizione su un *dataset* indipendente:

$$Err_T = E[L(Y, \hat{f}(X)|T)], \tag{2.12}$$

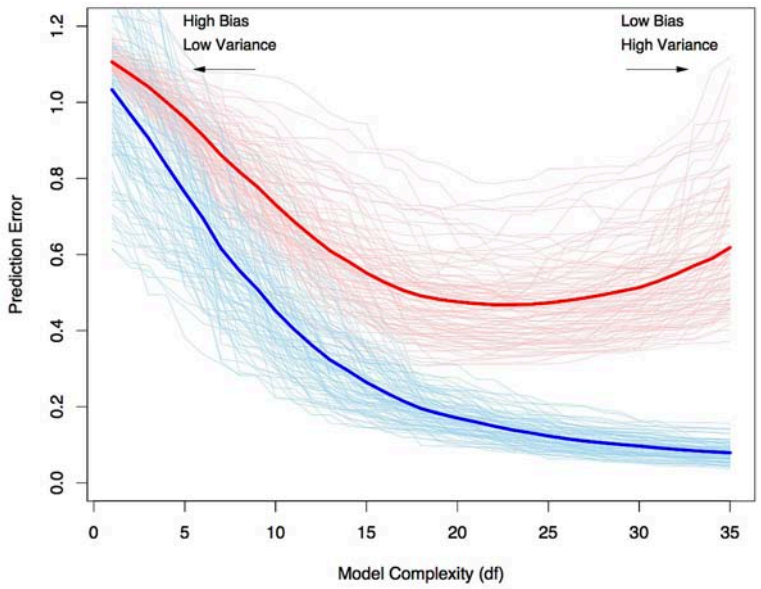

Figura 2.8: Errore di predizione medio (linea blu) e errore di *training* medio (linea rossa) al variare della complessità di un modello. La stima degli errori è stata effettuata mediando l'errore relativo, in funzione della complessità del modello, calcolato su 100 *training set* di 50 campioni ciascuno.

dove sia X che Y sono ricavati in modo *random* dalla loro distribuzione congiunta. In questo caso, il *training* set T è fissato e il *test error* fa riferimento all'errore per questo specifico *training set*.
Una misura correlata è l'*errore di predizione atteso*

$$Err = E[L(Y, \hat{f}(X)] = E[Err_T]. \tag{2.13}$$

In Figura 2.8 vengono mostrati gli errori di predizione Err_T (curve leggere rosse), per 100 *training set* di 50 campioni ciascuno, e la loro media (curva forte rossa) al variare della complessità del modello.

Un'altra metrica importante è l'*errore di training*, o *training error*, $\overline{err}$, che misura la perdita media sul training set:

$$\overline{err} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i)). \tag{2.14}$$

È interessante notare come l'errore di *training*, rappresentato in Figura 2.8 con una curva blu, tenda a zero al crescere della complessità del modello. Un modello con errore di *training* pari a zero probabilmente presenta un *overfit* sul *training set*, e tipicamente non funziona molto bene se utilizzato per classificare *dataset* indipendenti.
Tipicamente, un modello dispone di un parametro α di *tuning* che va a incidere sulla sua complessità (in generale, i parametri possono essere più di uno). Per questo motivo, durante la fase di valutazione delle *performance* del modello, si cerca il valore di α che minimizzi l'errore di generalizzazione medio.
Tipicamente si opera in due fasi: durante la prima, chiamata *model selection*, si stimano le *performance* di diversi modelli in modo da scegliere il migliore; scelto il modello, si entra nella fase chiamata *model assessment*, durante la quale si stima l'errore di di generalizzazione su nuovi dati.

Figura 2.9: Proporzione tipiche di suddivisione di un *dataset* in *training set* (50%), *validation set* (25%) e *test set* (25%).

Se si dispone di un *dataset* molto grande, tipicamente conviene dividerlo in tre parti:

- *training set*, utilizzato per addestrare il modello;
- *validation set*, utilizzato per stimare l'errore di predizione per la selezione del modello;
- *test set*, utilizzato per stimare l'errore di generalizzazione del modello scelto, una volta giunti al termine dell'analisi dei dati.

La suddivisione del *dataset* in *training*, *validation* e *test set* conviene effettuarla in modo *random*. Le proporzioni di suddivisione tipiche sono 50% per *training set*, 25% per *validation set* e 25% per *test set* (Figura 2.9).

2.2.1 K-Fold Cross-Validation

La *cross-validation* probabilmente rappresenta una delle tecniche più popolari per stimare l'errore di predizione di un modello. Questa tecnica stima direttamente il valore attesto dell'errore di generalizzazione che si misura quando un modello $\hat{f}(X)$ viene utilizzato per classificare un test set indipendente dalle distribuzioni congiunte X e Y.
La tecnica della *K-Fold Cross-Validation* viene comunemente usata quando il *dataset* a disposizione non è abbastanza grande da poter essere diviso in *training set*, *validation set* e *test set*. L'algoritmo consiste nel dividere il *dataset* in K parti, o *fold*, di ugual dimensione e, a ogni iterazione, una delle K parti viene utilizzata per validare il modello, le restanti per allenarlo. Quindi, si procede con l'addestramento, si calcola l'errore di predizione e si ripete il ciclo $K-1$ volte, cambiando di volta in volta il fold utilizzato come *validation set*. Alla fine si calcola la media dei K errori di predizione stimati per ottenere il *Cross-Validation Error*.
Più precisamente, sia $\kappa : \{1, ..., N\} \mapsto \{1, ..., K\}$ una funzione che indica la partizione in cui viene allocata l'osservazione i. Sia $\hat{f}^{-k}(x)$ la funzione calcolata con la k-esima parte dei dati rimossi. La *cross-validation* stima l'errore di predizione

$$CV(\hat{f}) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i)). \tag{2.15}$$

I valori tipici di K sono 5 e 10, mentre il caso in cui $K = N$ è noto come *cross-validation leave-one-out*.
Nell'esempio di Figura 2.10, si può osservare l'errore di predizione e la curva di *10-fold cross-validation* stimata da un unico *training set*, al variare della dimensione p dei suoi sottoinsiemi (e quindi al variare di K), per un problema di classificazione a due classi affrontato con un modello lineare.

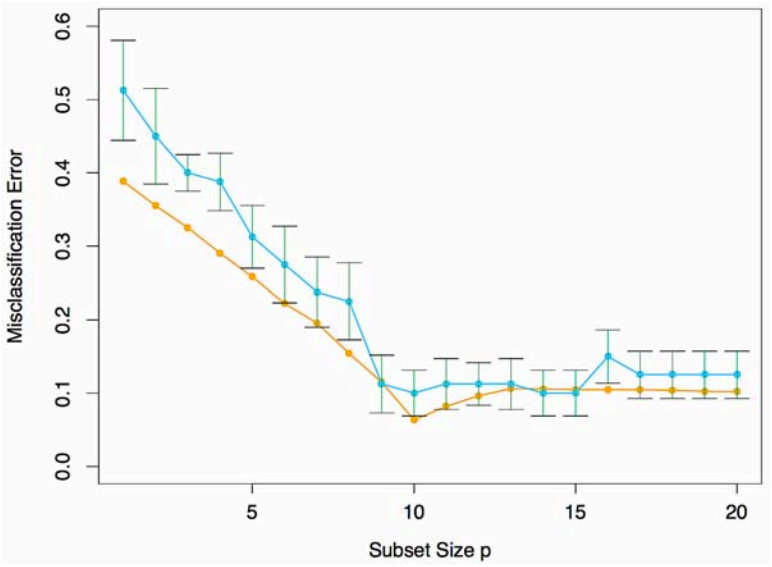

Figura 2.10: Errore di predizione (arancione) e curva di *10-fold cross-validation* (blu) stimata da un unico *training set*, al variare della dimensione p dei suoi sottoinsiemi.

2.2.2 Curve di prestazione e matrice di confusione

Come è facile intuire, esistono vari modi per misurare la qualità di un modello predittivo. Dato un insieme di osservazioni di cui sono note le rispettive classi di appartenenza, per avere un'idea quantitativa della bontà di un classificatore nel predirle, si può ricorrere al calcolo della cosiddetta *matrice di confusione*. Si tratta di una rappresentazione statistica dell'accuratezza nella classificazione; infatti, la cella (i, j) della matrice indica il numero di osservazioni di classe i che il modello classifica come appartenenti alla classe j, con $i, j = 1, ..., C$ e C numero di classi possibili.
Nel caso in cui si tratti di classificazione binaria, chiamando *positive* e *negative* le due possibili classi di appartenenza, si indicano:

- *true positive*, o *TP*, le osservazioni di classe positiva correttamente classificate;
- *true negative*, o *TN*, le osservazioni di classe negativa correttamente classificate;
- *false positive*, o *FP*, le osservazioni di classe negativa erroneamente classificati come positive;
- *false negative*, o *FN*, le osservazioni di classe positiva erroneamente classificati come negative.

È possibile valutare la bontà di un classificatore binario rispetto a un insieme di dati, di cui sono note le classi, utilizzando alcuni indici che derivano dall'*Information Retrieval*, tra cui:

- *sensitivity*, *recall*, *hit rate* o *true positive rate*, definito come $TPR = \frac{TP}{TP+FN}$;
- *specificity*, o *true negative rate*, $TNR = \frac{TN}{FP+TN}$;

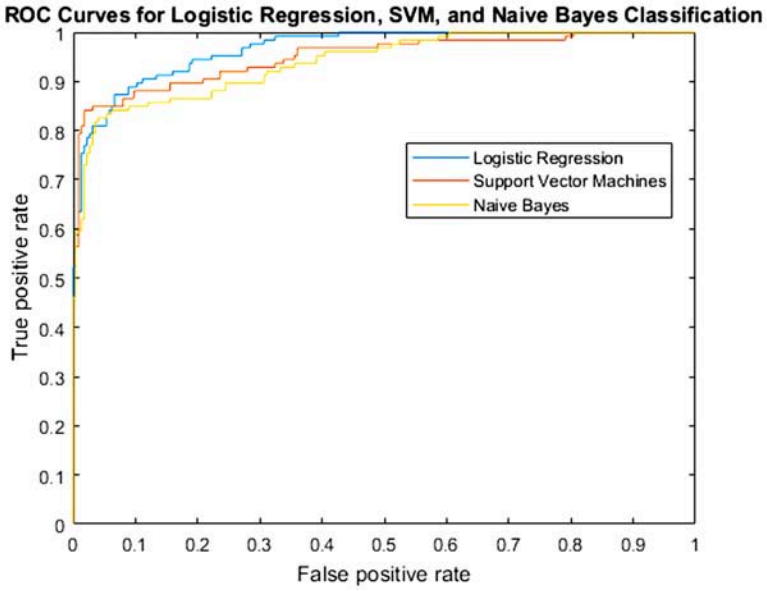

Figura 2.11: Esempio di curva ROC tratto dalla documentazione di MATLAB.

- *precision*, o *positive predictive value*, $PPV = \frac{TP}{TP+FP}$;
- *negative predictive value*, $NPV = \frac{TN}{TN+FN}$;
- *fall-out*, o *false positive rate*, $FPR = \frac{FP}{FP+TN} = 1 - TNR$;
- *false negative rate*, $FNR = \frac{FN}{FN+TP} = 1 - TPR$;
- *accuracy*, $ACC = \frac{TP+TN}{TP+TN+FP+FN}$;
- *F1 score*, $F_1 = \frac{2TP}{2TP+FP+FN}$.

Un altro strumento utile per valutare le prestazioni di un modello per la classificazione binaria è la *Receiver Operating Characteristic* (ROC), che mostra la relazione tra *true positive rate* (asse delle ordinate) e *false positive rate* (asse delle ascisse).
Dalla ROC è possibile ricavare l'*Area Under the Curve* (AUC), cioè la misura dell'area al di sotto della curva, che fornisce una sintesi estrema delle *performance* complessive del classificatore. Il valore dell'AUC può variare tra 0 e 1, e i due casi limite rappresentano, rispettivamente, prestazioni del classificatore minime e massime.

In Figura 2.11 viene illustrato un esempio, tratto dalla documentazione di MATLAB, in cui viene confrontata la curva ROC di più classificatori testati sullo stesso *dataset*.

2.3 Apprendimento non supervisionato: cluster analysis

Dato un insieme di N osservazioni $(x_1, x_2, ...x_N)$ di un p-vector X (vettore a p componenti) con densità di probabilità congiunta $P_r(X)$, l'obiettivo dell'ap-

prendimento non supervisionato è quello di dedurre le proprietà di questa densità senza l'aiuto di un supervisore che, per ogni osservazione, fornisca la risposta corretta o il grado di errore [10]. Rispetto all'apprendimento supervisionato, talvolta si lavora con vettori X più grandi e le proprietà di interesse sono spesso più complicate da individuare.
Per problemi di piccola dimensione (ad esempio $p \leq 3$) si usano diversi metodi non-parametrici efficaci per stimare direttamente $P_r(X)$ stessa per tutti i valori di X, e rappresentarli graficamente. Se, invece, la dimensione di X è molto grande, questi metodi falliscono e bisogna accontentarsi di modelli in grado di calcolare stime piuttosto approssimate, come misture Gaussiane o semplici statistiche descrittive che caratterizzano $P_r(X)$.
Generalmente, queste statistiche descrittive cercano di caratterizzare i valori di X o sottoinsiemi di tali valori per cui $P_r(X)$ è relativamente grande. Per esempio, la *cluster analysis* cerca di trovare le regioni convesse, nello spazio del vettore X, che contengano modi di $P_r(X)$, cioè classi di modalità caratterizzate da massima frequenza. In questo modo si può dire se $P_r(X)$ può essere rappresentata come combinazione di densità più semplici, che rappresentano tipi o classi di osservazioni. *Mixture modeling* ha un obiettivo simile alla *cluster analysis*, mentre *association rules* cerca di costruire semplici descrizioni, dette *regole*, che descrivano regioni di alta densità, nel caso si lavori con grandi moli di dati a valori binari.
Nell'apprendimento supervisionato, il successo della classificazione dei dati può essere misurato in diversi modi, per esempio attraverso *cross-validation*. Nell'apprendimento non supervisionato, invece, non esiste una misura diretta del successo e, quindi, è difficile stimare la qualità degli *output* dei rispettivi algoritmi. In generale si ricorre a euristiche e a interpretazioni che dipendono dal contesto e dalla tipologia di dati con cui si opera.

Nell'esperimento descritto nei capitoli successivi, per verificare se i dati in *input* celassero dei raggruppamenti naturali in base a qualche *feature*, si è scelto di ricorrere alla *cluster analysis*. In generale, lo scopo della *cluster anaylisis* è quello di raggruppare - o segmentare - collezioni di oggetti in sottoinsiemi, chiamati *cluster*, tali per cui oggetti nello stesso *cluster* siano più strettamente correlati tra loro rispetto ad oggetti assegnati a *cluster* diversi. Talvolta si può aggiungere l'obiettivo di raggruppare i *cluster* creati in ulteriori *cluster*, formando, in questo modo, delle gerarchie naturali di *cluster*.
La *cluster analysis* può essere utilizzata anche per effettuare statistiche descrittive che verifichino se i dati siano divisibili in sottogruppi distinti, ciascuno dei quali con proprietà diverse.
Nella *cluster analysis* è fondamentale la definizione di misura del grado di somiglianza tra gli oggetti; infatti, la formazione di una partizione di oggetti piuttosto che di un'altra, può dipendere molto dal tipo di misura utilizzata.
Figura 2.12 mostra un esempio di applicazione dell'algoritmo di *clustering K-means* per il raggruppamento in tre *cluster* di dati simulati.

Si possono individuare tre categorie principali di algoritmi per la *cluster analysis*:

- *algoritmi combinatoriali* lavorano direttamente con i dati osservati senza

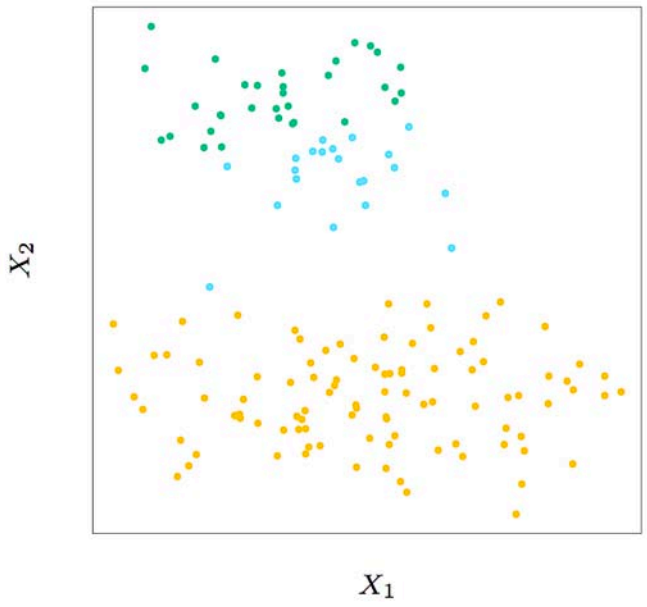

Figura 2.12: Dati raggruppati in tre classi (rappresentate dai colori arancione, blu e verde) mediante l'algoritmo di *clustering K-means*.

riferimenti diretti a modelli di probabilità sottostanti;

- *mixture modeling* suppongono che i dati siano campioni i.i.d. (indipendenti e identicamente distribuiti) descritti da una funzione di densità di probabilità, che è caratterizzata da un modello parametrizzato considerato come una combinazione di funzioni di densità più semplici, ciascuna delle quali descrive un *cluster*;

- *mode seeking* assume una prospettiva non-parametrica, cercando di stimare direttamente modi distinti della funzione di densità di probabilità.

Un'altra possibile classificazione degli algoritmi per la *cluster analysis* è elencata in seguito [12].

- *Connectivity-based clustering* (o *hierarchical clustering*) *algorithms*: l'idea è quella di connettere - per formare un *cluster* - gli oggetti del *dataset* in base alla loro distanza; infatti, si assume che tanto più vicini sono gli oggetti, tanto più sono correlati tra loro. Ciascun *cluster* può essere descritto dalla massima distanza necessaria per connettere le parti che lo compongono. Per rappresentare il risultato del *clustering* si ricorre ai *dendrogrammi*, che mostrano graficamente il modo in cui sono stati raggruppati i dati. In particolare, l'asse delle y di un dendrogramma rappresenta la distanza (o *similarity/dissimilarity*) tra i vari *cluster*, mentre lungo l'asse x vengono disposti gli oggetti che ne fanno parte. Gli algoritmi appartenenti a questa famiglia differiscono principalmente per la scelta del metodo per calcolare la distanza e del criterio di *linkage* degli oggetti.

- *Centroid-based clustering algorithms*: in questo caso i *cluster* vengono rappresentati da un vettore centrale, o centroide, che non è necessariamente un membro del *dataset*. Il problema *NP-hard* che si pongono di affrontare gli algoritmi appartenenti a questa categoria è il seguente: fissato il numero k

di *cluster* che si vuol individuare, trovare i k centroidi e assegnare ciascun oggetto al *cluster* il cui centroide dista meno, in modo tale da minimizzare le distanze tra i *cluster*. Nonostante l'elevata complessità del problema, esistono dei metodi di approssimazione molto conosciuti, come per esempio l'algoritmo di *Lloyd*, meglio noto come *k-means*.

- *Distribution-based clustering algorithms*: i *cluster* vengono identificati da modelli di distribuzione e ciascun oggetto viene inserito nel *cluster* tale per cui è minima la probabilità che la distribuzione dell'oggetto sia uguale a quella del *cluster*. L'aspetto positivo di quest'approccio è che ricorda il modo in cui vengono creati insiemi artificiali di dati, cioè campionando oggetti casuali da una distribuzione. Tuttavia, questa categoria di algoritmi tendono a soffrire di problemi legati all'*overfitting*, che possono essere affrontati agendo sulla complessità (intesa come numero di vincoli) dei modelli utilizzati. Appartiene a questa categoria il modello *Gaussian mixture*: in questo caso, il *dataset* viene modellato con un numero fisso di distribuzioni Gaussiane inizializzate con parametri *random*, i quali vengono poi ottimizzati iterativamente per trovare la migliore approssimazione del *dataset*.

- *Density-based clustering algorithms*: con riferimento allo spazio dei dati che compongono il *dataset*, i *cluster* sono definiti come le aree in cui la densità di dati è maggiore rispetto a una soglia. Uno dei metodi più popolari di questa categoria è DBSCAN [13], che, come nel *clustering connectivity-based*, connette i punti entro una certa soglia di distanza, però solo se soddisfano un criterio di densità, definito come numero minimo di altri punti presenti entro un certo raggio.

Come già accennato, nell'effettuare l'esperimento descritto nel Capitolo 3 di questo elaborato si è scelto utilizzare *K-Means Clustering* e *Hierarchical Clustering*, i cui funzionamenti di base vengono approfonditi nei due paragrafi successivi.

2.3.1 K-Means Clustering

L'algoritmo *K-means* è uno dei metodi di *clustering* più popolari e si presta bene nei casi in cui tutte le variabili che rappresentano i dati sono di tipo quantitativo. Come metrica per misurare la *dissimilarity* fra coppie di oggetti diversi viene tipicamente utilizzata la distanza Euclidea quadratica, definita come

$$d(x_i, x_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = \left\| x_i - x_{i'} \right\|^2 . \tag{2.16}$$

L'obiettivo dell'algoritmo è assegnare le N osservazioni (elementi del *dataset*) ai K *cluster* in modo tale che, in ogni *cluster*, la *dissimilarity* tra la media delle osservazioni e il centroide del *cluster* sia minimizzata. Un algoritmo iterativo per risolvere il problema di ottimizzazione

$$C^* = \min_{C} \sum_{k=1}^{K} N_k \sum_{C(i)=k} \left\| x_i - \bar{x}_k \right\|^2 \tag{2.17}$$

può essere ottenuto notando che, per ogni insieme S di osservazioni,

$$\bar{x}_S = \arg\min_m \sum_{i \in S} \left\| x_i - m \right\|^2 . \tag{2.18}$$

Quindi si può ottenere C^* risolvendo il problema di ottimizzazione

$$\min_{C, \left\{m_k\right\}_1^K} \sum_{k=1}^{K} N_k \sum_{C(i)=k} \left\| x_i - m_k \right\|^2 , \tag{2.19}$$

per esempio alternando le procedure date dall'Algoritmo 1. In pratica, ciascuna delle fasi 1 e 2 riduce il valore del criterio (2.19) in modo tale da assicurare la convergenza, anche se il risultato può rappresentare un minimo locale sub-ottimo. In generale, si dovrebbe avviare l'algoritmo più volte scegliendo diversi valori di k e scegliere la soluzione che porta al valore più piccolo della funzione obiettivo. In Figura 2.13 viene mostrato un esempio di applicazione dell'algoritmo *k-means clustering* su dati simulati. Alla prima iterazione vengono individuati i centroidi dei *cluster* e si prosegue con il partizionamento dei dati; alle iterazioni successive si cerca di migliorare il partizionamento alternando i seguenti due passaggi:

- per ogni punto, viene identificato il centro *cluster* più vicino (mediante distanza Euclidea);
- ogni centro *cluster* viene rimpiazzato dalle coordinate della media di tutti i punti che si trovano al suo interno.

Quest'algoritmo, a differenza di altre tipologie di *clustering*, adotta una procedura *top-down*, in quanto parte da un singolo *cluster*, che viene iterativamente diviso in più *cluster*, i quali, a loro volta, vengono migliorati in modo iterativo. Al contrario, negli approcci *bottom-up*, si parte col considerare ogni elemento del *dataset* come un *cluster* a sé stante e, ad ogni iterazione, si procede con l'accorpare due *cluster* in un unico *cluster*. *Hiearchical clustering* è un algoritmo che può essere eseguito scegliendo di adottare quest'ultima strategia.

Algorithm 1 K-means Clustering.

1. Per un dato *cluster* di assegnazione C, la varianza totale del *cluster* (2.19) è minimizzata rispetto a $\left\{m_1, ..., m_K\right\}$ producendo i *means* dei *cluster* attualmente assegnati (2.18).
2. Dato un insieme corrente di medie $\left\{m_1, ..., m_K\right\}$, (2.19) è miminizzata assegnando a ogni osservazione la media del *cluster* (corrente) più vicino, cioè
$$C(i) = \arg\min_{1 \le k \le K} \left\| x_i - m_k \right\|^2 . \tag{2.20}$$
3. I passaggi 1 e 2 sono iterati finché gli assegnamenti non cambiano.

2.3.2 Hierarchical Clustering

L'algoritmo *hierarchical clustering*, o *clustering gerarchico*, a differenza di *K-means*, non richiede di specificare a priori il numero di *cluster* che si vuole ottenere. In questo caso, però, assume ruolo cruciale la metrica scelta per misurare

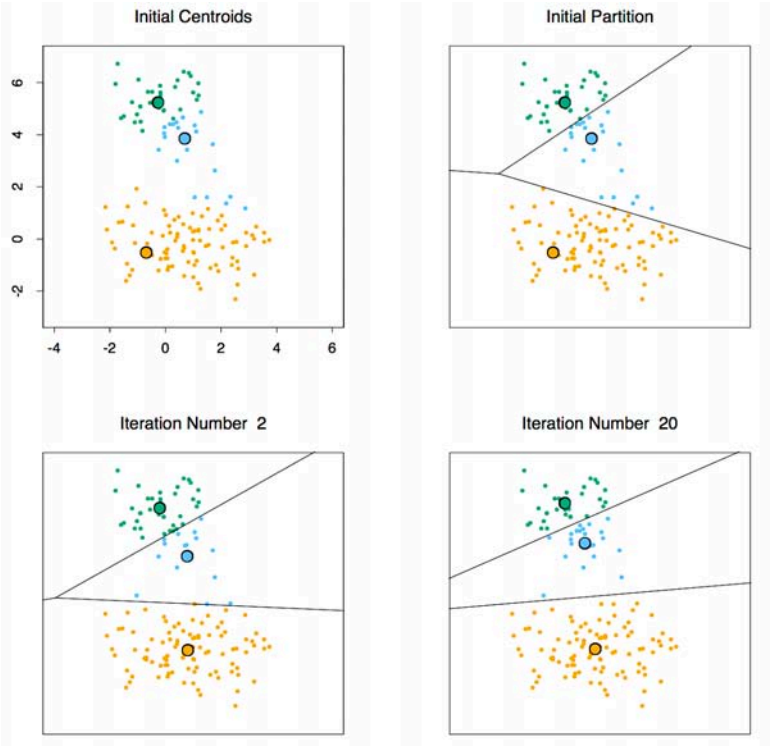

Figura 2.13: Esempio di applicazione dell'algoritmo *k-means clustering* sui dati rappresentati in Figura 2.12.

la *dissimilarity* tra gruppi disgiunti di osservazioni. L'algoritmo produce in uscita una rappresentazione gerarchica in cui i *cluster* di ogni livello possono essere creati unendo i *cluster* al livello inferiore. I nodi esterni di questa struttura ad albero rappresentano le singole osservazioni.
Si può effettuare *clustering* gerarchico seguendo due strategie:

- *agglomerativa* (*bottom-up*), che parte considerando ogni osservazione come un *cluster* a sé stante e, ad ogni iterazione, raggruppa coppie di *cluster* in un unico *cluster*, in modo da formare gruppi a minima *dissimilarity*;

- *divisiva* (*top-down*), che parte considerando l'insieme di tutte le osservazioni come un unico *cluster* e divide ricorsivamente ogni livello di *cluster* in più *cluster*; a ciacun livello, la divisione viene effettuata in modo da creare due nuovi gruppi con massima *dissimilarity*.

Se N è il numero di osservazioni, con entrambi gli approcci alla fine si produce una struttura gerarchica ad $N - 1$ livelli, che può essere visualizzata sotto forma di albero binario in cui:

- le foglie rappresentano le singole osservazioni;

- i nodi intermedi rappresentano i raggruppamenti;

- la radice rappresenta l'intero *dataset*.

L'albero binario può essere raffigurato in modo tale che l'altezza di ogni nodo sia proporzionale al valore di dissimilarità tra i suoi due nodi figli, mentre i nodi terminali si trovano ad altezza zero. Il grafico che ne risulta, comunemente

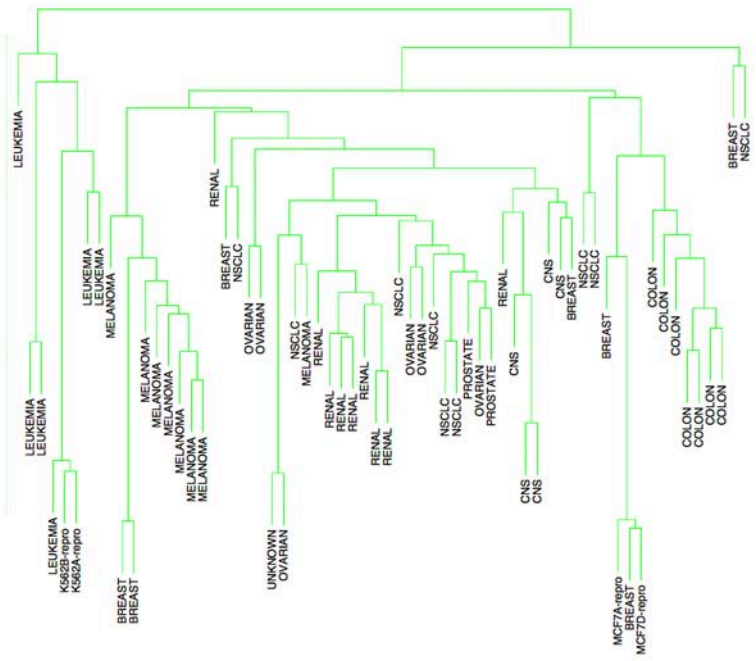

Figura 2.14: Esempio di dendrogramma costruito mediante *clustering* gerarchico agglomerativo.

chiamato *dendrogramma*, fornisce una descrizione completa e facilmente interpretabile del risultato del *clustering* gerarchico e se ne può osservare un esempio in Figura 2.14.

Un dendrogramma può essere visto come una sintesi della struttura dei dati. Tuttavia questa interpretazione presenta dei limiti, perché diversi metodi di *linkage* dei dati possono generare dendrogrammi molto diversi. Inoltre, i metodi gerarchici impongono una struttura gerarchica anche se tale struttura non esiste realmente nei dati. Per questi motivi ha senso misurare la qualità di un dendrogramma e a tal scopo si utilizza il *coefficiente di correlazione cofenetica*. Questo coefficiente non è altro che la misura della correlazione tra le $N(N-1)/2$ coppie di dissimilarità delle osservazioni $d_{ii'}$ in *input* all'algoritmo e le loro corrispondenti dissimilarità cofenetiche $C_{ii'}$ derivanti dal dendrogramma. La dissimilarità cofenetica $C_{ii'}$ tra due osservazioni $(i, i^{'})$ è la dissimilarità alla quale le osservazioni i e $i^{'}$ sono per la prima volta uniti insieme nello stesso *cluster*. Il coefficiente di correlazione cofenetica, in sintesi, misura quanto una struttura gerarchica in forma di dendrogramma rappresenta effettivamente i dati.
In Figura 2.15 viene mostrato un esempio di dendrogramma plottato mediante la funzione *dendrogram* del *Statistics and Machine Learning Toolbox* di MATLAB.

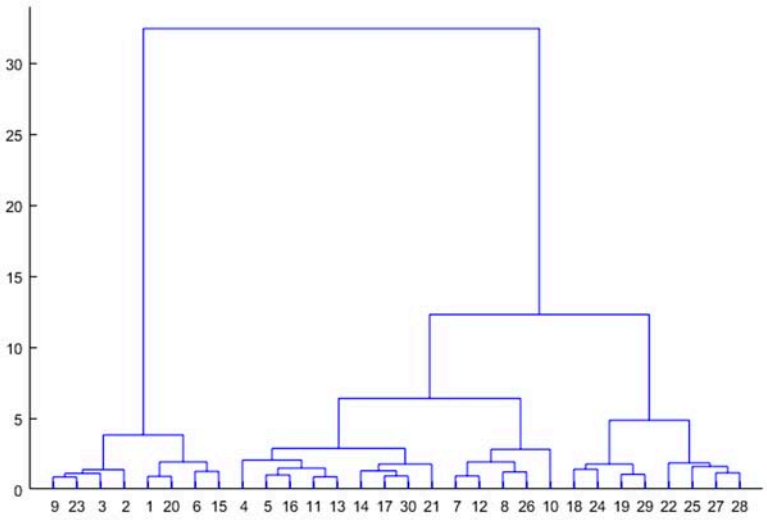

Figura 2.15: Esempio di dendrogramma costruito mediante la funzione *dendrogram* del *Statistics and Machine Learning Toolbox* di MATLAB.

Capitolo 3

Descrizione dell'esperimento

In questo capitolo verrà descritto il *setup* dell'esperimento effettuato per valutare la capacità del calcolatore di riconoscere le equalizzazioni di pre-enfasi e post-enfasi applicate a documenti sonori di origine analogica, utilizzando tecniche di *machine learning*, o apprendimento automatico.

In particolare, si discuteranno:

- il *workflow* e gli strumenti utilizzati per la costruzione del *dataset* (Sezione 3.1);
- le strutture dati utilizzate per memorizzare i campioni audio digitali (Sezione 3.2);
- le *feature* estratte dai dati che verranno utilizzate come *input* per gli algoritmi di *cluster analysis* e di *fitting* dei modelli predittivi (Sezione 3.3);
- la struttura del codice per la *cluster analysis* (Sezione 3.4);
- la struttura del codice per la classificazione (Sezione 3.5).

3.1 Creazione del dataset

Il *dataset* per l'esperimento è stato creato al Centro di Sonologia Computazionale (CSC) dell'Università degli Studi di Padova, in collaborazione con la Dott.ssa Valentina Burini e sotto la supervisione dell'Ing. Niccolò Pretto e del Prof. Sergio Canazza.

Il procedimento seguito per la generazione dei campioni audio si può riassumere nei seguenti passaggi:

1 generazione delle tracce audio;

2 registrazione delle tracce audio su nastro magnetico vergine;

3 riproduzione e digitalizzazione dei documenti sonori registrati al passo precedente.

Figura 3.1: Modello di nastro magnetico utilizzato per l'esperimento.

3.1.1 Generazione delle tracce

Le tracce utilizzate per l'esperimento sono due e ciascuna dura 5 minuti. La loro sintesi è stata effettuata in ambiente *Pro Tools*[1], con frequenza di campionamento 96 kHz e risoluzione 24 bit. La prima traccia contiene silenzio ed è delimitata da due *bip* sonori di circa mezzo secondo, introdotti al fine di facilitare la sincronizzazione nelle fasi di registrazione e di riproduzione. La seconda traccia contiene rumore bianco.

3.1.2 Registrazione su nastro vergine

La registrazione su nastro magnetico delle tracce audio silenziosa e rumorosa generate in *Pro Tools* è stata effettuata interfacciando la *workstation* all'*Analogic Tape Recoder* (ATR) *Studer A810* tramite scheda audio *Prism Sound Lyra2*.
Prima di effettuare la registrazione, l'ATR è stato sottoposto a taratura e le sue componenti che vanno a contatto con il nastro (*capstain*, *pinchroller*, testine, guida del nastro, tendinastro e perni) sono state ripulite con *cotton fioc* imbevuti di alcol isopropilico.
Per la registrazione delle tracce sono stati utilizzati due nastri magnetici vergini *SM911* da 1/4 di pollice (Figura 3.1), ciascuno con durata complessiva di poco più di 60 minuti, se riprodotto a 7.5 ips.

Al fine di ottenere un *dataset* più eterogeneo possibile, su ciascun nastro sono state registrate più tracce audio; ogni registrazione è stata resa diversa rispetto a tutte le altre per una o più delle seguenti proprietà:

- tipo di traccia registrata (silenziosa o rumorosa);
- velocità di registrazione (7.5 ips o 15 ips);
- equalizzazione di pre-enfasi adottata (CCIR o NAB).

La registrazione è stata effettuata in modo tale che ciascuno dei due nastri venisse registrato in una prima parte a 7.5 ips e in una seconda a 15 ips, facendo in modo che entrambe avessero la stessa durata (la seconda parte necessita del doppio del

[1] http://www.avid.com/pro-tools

nastro rispetto alla prima). Ciascuna delle due parti, è stata a sua volta registrata per metà con pre-enfasi CCIR e per metà con pre-enfasi NAB. Infine, in ciascuna di queste ultime due parti è stata registrata per metà una traccia silenziosa, per metà una traccia con rumore bianco.
Di conseguenza, alla fine della fase di registrazione il contenuto di ciascuno dei due nastri è risultato il seguente:

- 5 minuti registrati a 7.5 ips, con pre-equalizzazione CCIR e traccia silenziosa;
- 5 minuti registrati a 7.5 ips, con pre-equalizzazione CCIR e traccia rumorosa;
- 5 minuti registrati a 7.5 ips, con pre-equalizzazione NAB e traccia silenziosa;
- 5 minuti registrati a 7.5 ips, con pre-equalizzazione NAB e traccia rumorosa;
- 5 minuti registrati a 15 ips, con pre-equalizzazione CCIR e traccia silenziosa;
- 5 minuti registrati a 15 ips, con pre-equalizzazione CCIR e traccia rumorosa;
- 5 minuti registrati a 15 ips, con pre-equalizzazione NAB e traccia silenziosa;
- 5 minuti registrati a 15 ips, con pre-equalizzazione NAB e traccia rumorosa;

3.1.3 Riproduzione e digitalizzazione

L'ultima fase del processo di creazione del dataset è consistita nella digitalizzazione di quanto registrato nella fase precedente, alternando, in lettura, le post-equalizzazioni CCIR e NAB.
Per la riproduzione e la digitalizzazione sono stati utilizzati, rispettivamente, lo stesso ATR, la stessa scheda audio e la stessa *workstation* utilizzati in fase di registrazione e i campioni sono stati digitalizzati a 96 kHz, con risoluzione 24 bit. La lettura del nastro è avvenuta cambiando, per ogni metà di ciascun frammento di 5 minuti, la post-equalizzazione. In questo modo sono stati creati 16 tipologie di campioni audio digitali, ciascuno della durata di 2.5 minuti, con le seguenti caratteristiche:

campione 1 velocità 7.5 ips, pre-eq. CCIR, post-eq. CCIR, traccia silenziosa;

campione 2 velocità 7.5 ips, pre-eq. CCIR, post-eq. NAB, traccia silenziosa;

campione 3 velocità 7.5 ips, pre-eq. NAB, post-eq. CCIR, traccia silenziosa;

campione 4 velocità 7.5 ips, pre-eq. NAB, post-eq. NAB, traccia silenziosa;

campione 5 velocità 7.5 ips, pre-eq. CCIR, post-eq. CCIR, traccia rumorosa;

campione 6 velocità 7.5 ips, pre-eq. CCIR, post-eq. NAB, traccia rumorosa;

campione 7 velocità 7.5 ips, pre-eq. NAB, post-eq. CCIR, traccia rumorosa;

campione 8 velocità 7.5 ips, pre-eq. NAB, post-eq. NAB, traccia rumorosa;

campione 9 velocità 15 ips, pre-eq. CCIR, post-eq. CCIR, traccia silenziosa;

campione 10 velocità 15 ips, pre-eq. CCIR, post-eq. NAB, traccia silenziosa;

campione 11 velocità 15 ips, pre-eq. NAB, post-eq. CCIR, traccia silenziosa;

campione 12 velocità 15 ips, pre-eq. NAB, post-eq. NAB, traccia silenziosa;

campione 13 velocità 15 ips, pre-eq. CCIR, post-eq. CCIR, traccia rumorosa;

campione 14 velocità 15 ips, pre-eq. CCIR, post-eq. NAB, traccia rumorosa;

campione 15 velocità 15 ips, pre-eq. NAB, post-eq. CCIR, traccia rumorosa;

campione 16 velocità 15 ips, pre-eq. NAB, post-eq. NAB, traccia rumorosa.

3.2 Divisione dei campioni

I file digitali creati nel modo descritto nel paragrafo precedente, rappresentano l'*input* della *routine* MATLAB per l'addestramento, la validazione e il test di alcuni classificatori per il riconoscimento dell'equalizzazione adottata.
Una volta importati in MATLAB sotto forma di vettori, i campioni audio sono stati divisi in frammenti da 1 secondo, o osservazioni, poi organizzati in matrici nxm, dove n rappresenta il numero complessivo di osservazioni ed m il numero di componenti di ciascuna osservazione. Trattandosi di campioni audio digitalizzati a 96 kHz, le riga di ciascuna matrice contiene 96 mila componenti.

Nei paragrafi successivi vengono descritti le feature e gli algoritmi utilizzati per *cluster analysis* e classificazione. Le rispettive *performance* sono state valutate, separatamente, per i seguenti *dataset*:

- campioni silenziosi registrati e letti a 7.5 ips;

- campioni rumorosi registrati e letti a 7.5 ips;

- campioni silenziosi registrati e letti a 15 ips;

- campioni rumorosi registrati e letti a 15 ips;

Ciascuno dei dataset precedentemente elencati è composto da 1200 campioni audio di durata 1 secondo, di cui:

- 300 pre-equalizzati CCIR e post-equalizzati CCIR;

- 300 pre-equalizzati CCIR e post-equalizzati NAB;

- 300 pre-equalizzati NAB e post-equalizzati CCIR;

- 300 pre-equalizzati NAB e post-equalizzati NAB.

3.3 Tipologie di campioni utilizzate

Inizialmente si è pensato di utilizzare come *feature* per la classificazione l'intero spettro del segnale audio. La *routine* utilizzata per il calcolo delle *feature* si serve di alcune funzioni di *MIRtoolbox* (*Music Information Retrieval toolbox*), un *free software* che contiene molti estrattori di *feature* musicali e descrittori statistici, molto apprezzato nel mondo accademico. In Figura 3.2 è illustrata un'anteprima delle funzioni disponibili in *MIRtoolbox I.2* [14].

Per ciascun frammento audio la *routine* calcola lo spettro normalizzato, utilizzando la funzione *mirspectrum*, e tutti gli spettri vengono sistemati in apposite matrici nxl, dove n è il numero totale di campioni ed l il numero di componenti dello spettro ($l = \frac{F_c}{2} = \frac{96000}{2} = 48000$).

Un'altra tipologia di *feature* utilizzata per testare i classificatori è costituita dai *mel-frequency cepstral coefficient*, o coefficienti cepstrali, che offrono una descrizione molto compatta della forma spettrale del suono. Il calcolo dei coefficienti cepstrali avviene in modo molto simile rispetto al calcolo del *cepstrum*, definito come l'*Inverse Fourier Transform* (IFT) applicata al logaritmo del modulo dello spettro del segnale (schema mostrato in Figura 3.3). In pratica, per ottenere i coefficienti cepstrali di un segnale, viene dapprima calcolato il logaritmo del modulo della trasformata di *Fourier* del segnale stesso, che viene poi mappato in scala di *Mel* e dato in input all'algoritmo per il calcolo del *Discrete Cosine Transform* o DCT (Figura 3.4).

Questo tipo di *feature* si caratterizza per essere molto compatto, in quanto riesce a concentrare la maggior parte dell'informazione relativa allo spettro segnale in poche componenti (di *default* solo 13).
Per calcolare i coefficienti cepstrali di ciascun campione, ci si è serviti dapprima della funzione *mirspectrum*, per il calcolo del logaritmo del modulo dello spettro in scala di *Mel*, e poi della funzione *mirmfcc*, che di *default* restituisce un vettore di 13 componenti. Come nel caso delle *feature* rappresentate dall'intero spettro, i coeffienti cepstrali dei campioni sono stati organizzati in matrici nxc, dove n è il numero di campioni, mentre c rappresenta il numero di coefficienti cepstrali di ciascun campione ($c = 13$).

I risultati illustrati nel capitolo successivo mostreranno che addestrare i classificatori con la seconda tipologia di *feature* sia la scelta vincente ai fini del riconoscimento delle equalizzazioni applicate ai campioni audio. Infatti, l'utilizzo dei coefficienti cepstrali permette di:

- caratterizzare i campioni audio in modo molto più preciso e compatto rispetto all'utilizzo delle *feature* rappresentate dall'intero spettro;

- rendere la computazione e, in particolare, l'addestramento dei classificatori, molto veloce (ovvia conseguenza del fatto che l'intero spettro abbia, nel nostro caso, 48000 componenti, contro le 13 del vettore con i coefficienti cepstrali).

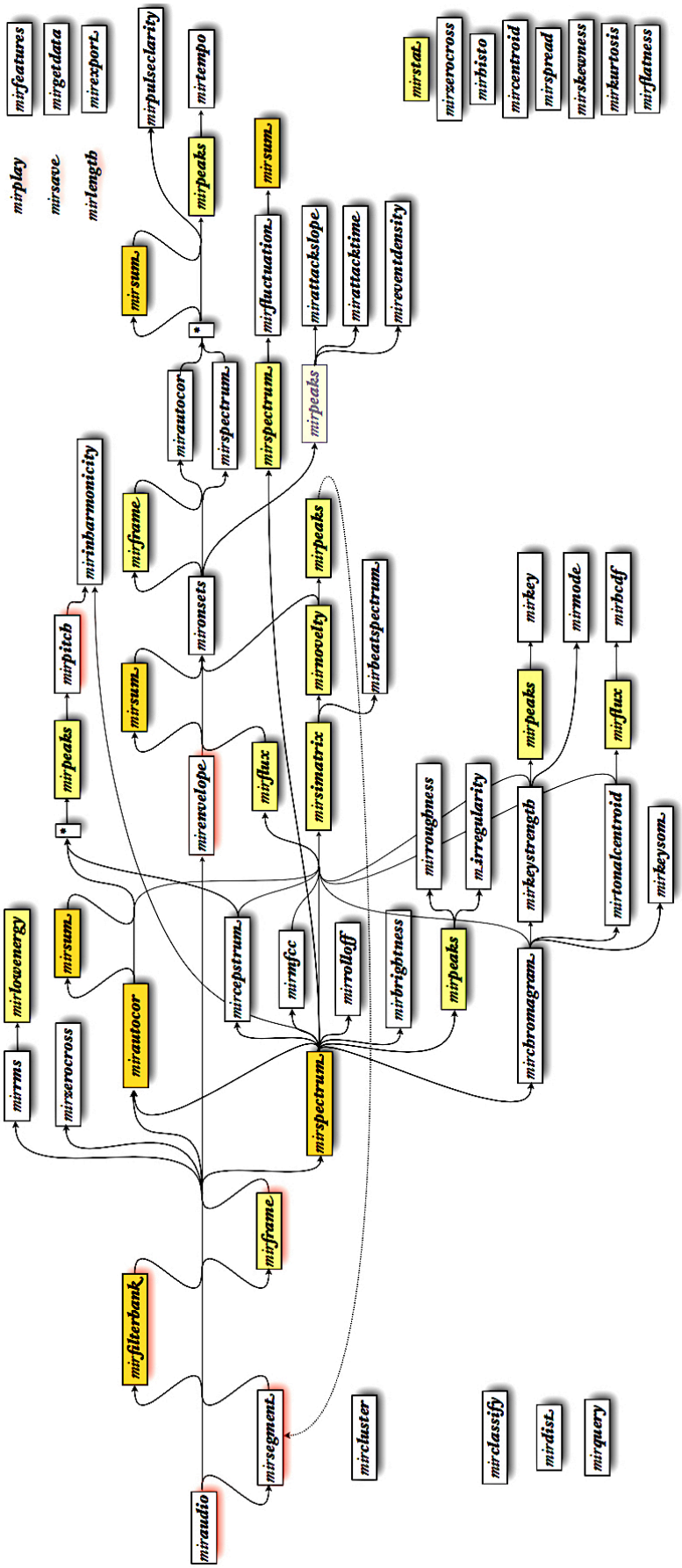

Figura 3.2: Anteprima delle funzioni disponibili in MIRtoolbox I.2.

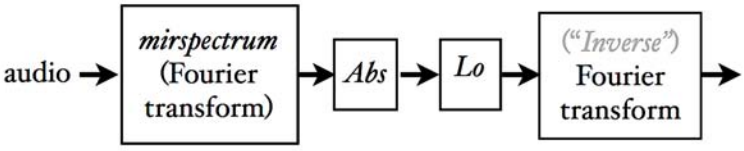

Figura 3.3: Schema che descrive il calcolo del *cepstrum* di un segnale. I blocchi rappresentano rispettivamente: *mirspectrum*, la funzione di *MIRtoolbox* per il calcolo dello spettro di un segnale; *Abs*, la funzione modulo; *Lo*, la funzione logaritmo; *Inverse Fourier Transform*, la funzione per il calcolo della IFT.

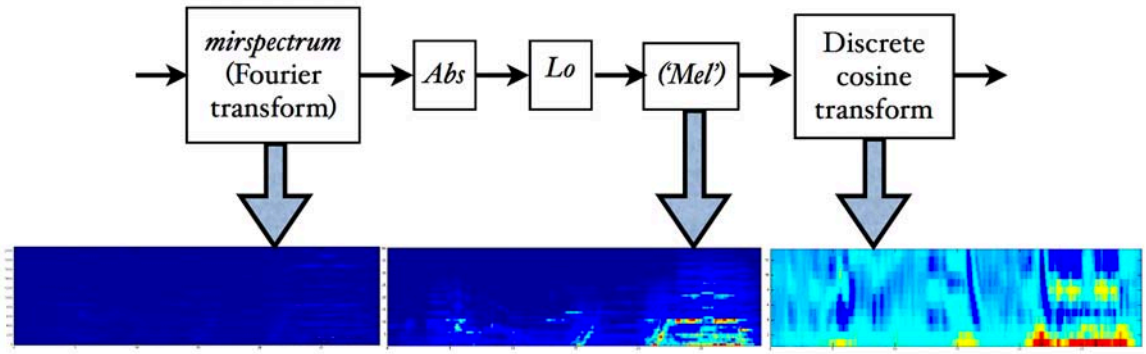

Figura 3.4: Schema che descrive il calcolo dei coefficienti cepstrali di un segnale. I blocchi rappresentano rispettivamente: *mirspectrum*, la funzione di *MIRtoolbox* per il calcolo dello spettro di un segnale; *Abs*, la funzione modulo; *Lo*, la funzione logaritmo; *'Mel'*, la mappatura in scala di *Mel*; *Discrete cosine transform*, la funzione per il calcolo dei coefficienti della trasformata discreta del coseno.

3.4 Cluster analysis con MATLAB

Prima di procedere con l'addestramento dei classificatori si è ritenuto utile cercare di capire, mediante *cluster analisys*, se di per sé i dati contenessero al loro interno *pattern* nascosti o partizionamenti "naturali". L'obiettivo iniziale era quello di trovare una partizione naturale dei dati nello spazio delle *feature*, basata sulla tipologia di equalizzazione applicata.
Alla fine dei test relativi alla *cluster analysis*, sono emerse due principali tendenze:

- divisione dei campioni silenziosi in due cluster, uno contenente quelli post-equalizzati CCIR e l'altro quelli post-equalizzati NAB;

- divisione dei campioni rumorosi in tre cluster, il primo contenente quelli equalizzati in modo corretto (pre-enfasi e post-enfasi entrambe NAB o CCIR), il secondo quelli post-equalizzati con pre-enfasi NAB e post-enfasi CCIR, il terzo quelli post-equalizzati con pre-enfasi CCIR e post-enfasi NAB.

Le tendenze si sono rilevate lievemente più "marcate" per i campioni registrati e riprodotti a 15 ips. Per ulteriori dettagli sui risultati della *cluster analysis*, si rimanda al Capitolo 4.

Nei prossimi due paragrafi verrà descritta l'impostazione del codice MATLAB per la *cluster analysis*.

3.4.1 Hierarchical Clustering

I passi essenziali dell'algoritmo utilizzato per effettuare *hierarchical clustering* sono elencati in seguito.

1 Importazione delle matrici contenenti, in ogni riga, le *feature* di un campione audio. Tali matrici sono state calcolate in precedenza e salvate in un apposito file con estensione *.mat*.

2 Scelta di una metrica di distanza, tra i *preset* della funzione *pdlist* del *Statistics and Machine Learning toolbox* di MATLAB. Le scelte possibili sono [9]:

- *euclidean*, distanza Euclidea;
- *squaredeuclidean*, distanza Euclidea Quadrata;
- *seuclidean*, distanza Euclidea Standardizzata;
- *cityblock*, metrica *City block*;
- *minkowski*, distanza di *Minkowski*, con esponente pari a 2;
- *chebychev*, distanza di *Chebychev*;
- *cosine*, uno meno il coseno dell'angolo compreso tra i punti campione;
- *correlation*, uno meno la correlazione dei punti campione;
- *spearman*, uno meno la correlazione per ranghi di *Spearman* delle osservazioni;
- *hamming*, distanza di *Hamming*;
- *jaccard*, uno meno il coefficiente di *Jaccard*.

3 Scelta di un metodo di *linkage*, tra i *preset* della funzione *linkage* del *Statistics and Machine Learning toolbox* di MATLAB, per calcolare la distanza tra i *cluster*. Le scelte possibili sono [9]:

- *average*, distanza media non pesata (UPGMA);
- *centroid*, distanza tra i centroidi (UPGMC), appropriato solo per distanza Euclidea.
- *complete*, distanza maggiore;
- *median*, distanza pesata del centro di massa (WPGMC), appropriato solo per distanza Euclidea;
- *single*, distanza minore;
- *ward*, varianza minima, appropriato solo per distanza Euclidea;
- *weighted*, distanza media pesata (WPGMA), appropriato solo per distanza Euclidea.

4 Calcolo della distanza per ogni coppia di oggetti del dataset di input utilizzando la funzione *pdlist*, secondo la metrica scelta al passo 2.

5 Calcola della distanza tra i *cluster* utilizzando la funzione *linkage*, secondo la metrica di scelta al passo 3.

6 Creazione del *cluster* utilizzando la funzione *cluster* del *Statistics and Machine Learning toolbox* di MATLAB, fissando a 4 il numero massimo di *cluster*.

7 Calcolo del coefficiente di correlazione cofenetica del *cluster* costruito al passo 6, utilizzando la funzione *cophenet* del *Statistics and Machine Learning toolbox* di MATLAB, per valutare la qualità del *cluster*.

8 Dato che la classe di appartenenza di ciascun campione è nota, calcolo di quanti campioni per tipologia di equalizzazione sono stati inseriti in ciascun cluster.

9 Ripetizione dei passi dal 2 all'8, per testare tutte le possibili combinazioni di metrica di distanza e algoritmo di *linkage*, tra quelli di *preset* nelle apposite funzioni MATLAB.

10 Costruzione di una tabella contenente i risultati ottenuti al passo 9, per ogni scelta dei parametri dell'algoritmo.

11 Salvataggio della tabella costruita al passo 10 in un *file* con estensione **.cvs*, poi importata in un documento *Excel* per maggiore leggibilità.

La scelta di fissare a 4 il numero massimo di *cluster* non è obbligatoria, ma motivata dalla natura dei dati. Infatti, l'algoritmo per la creazione dei *cluster* è stato preventivamente testato sia senza fissare un valore massimo, sia utilizzando valori diversi da 4. Ciò che è emerso è che, nei casi in cui i dati venissero divisi in più *cluster*, il numero di *cluster* molto popolati oscillava sempre tra 2 e 4, da cui la scelta di fissare il massimo a 4.

In Appendice A viene mostrato il codice della funzione MATLAB *cep_hierarchical_clustering*, implementata per testare l'algoritmo di *hierarchical clustering* al variare dei parametri.

3.4.2 K-means clustering

La routine MATLAB utilizzata per effettuare *k-means clustering* si articola nei passi elencati in seguito.

1 Importazione delle matrici contenenti, in ogni riga, le *feature* di un campione audio. Tali matrici sono state calcolate in precedenza e salvate in un apposito file con estensione *.mat*.

2 Scelta di una metrica di distanza, tra i *preset* della funzione *kmeans* del *Statistics and Machine Learning toolbox* di MATLAB. Le scelte possibili sono [9]:

- *sqedeuclidean*, distanza Euclidea Quadrata;
- *cityblock*, metrica *City block*;

- *cosine*, uno meno il coseno dell'angolo compreso tra i punti campione;
- *correlation*, uno meno la correlazione dei punti campione;
- *hamming*, distanza di *Hamming*;

3 Partizionamento delle osservazioni in *kcluster*, utilizzando la funzione *kmeans* del *Statistics and Machine Learning toolbox* di MATLAB.

4 Calcolo della *silhouette* media del partizionamento ottenuto al passo 3. In generale, tanto più alto è questo valore, tanto migliore è la scelta di k.

5 Dato che la classe di appartenenza di ciascun campione è nota, calcolo di quanti campioni per tipologia di equalizzazione sono stati inseriti in ciascun cluster.

6 Ripetizione dei passi dal 3 al 5, cambiando ogni volta il valore di k. Procedura testata per $k = 2, 3, 4$.

7 Ripetizione del passo 6, cambiano ogni volta la metrica di distanza, secondo i *preset* della funzione *kmeans*.

8 Costruzione di una tabella contenente i risultati ottenuti nei passi precedenti, per ogni scelta dei parametri dell'algoritmo.

9 Salvataggio della tabella costruita al passo 8 in un file con estensione **.cvs*, poi importata in un documento *Excel* per maggiore leggibilità.

Le considerazioni in merito alla scelta dei valori di k sono del tutto analoghe a quelle esposte nel caso di *hierarchical clustering*.

In Appendice A viene mostrato il codice della funzione MATLAB *cep_k_means_clustering*, implementata per testare l'algoritmo di *k-means clustering* al variare dei parametri.

3.5 Apprendimento supervisionato con MATLAB

La classificazione è una tipologia di apprendimento supervisionato in cui un algoritmo "impara" a classificare nuovi dati, basandosi su dati di cui conosce già la classe di appartenenza, detti dati di *training*.
Dopo essersi accertati, tramite *cluster analysis*, che è possibile dividere i campioni audio in base alle loro equalizzazioni di pre-enfasi e post-enfasi, si è proceduto a testare alcuni algoritmi per la classificazione.
In particolare, si è scelto di valutare la classificazione dando in *input* agli algoritmi ciascuno dei quattro *dataset* creati precedentemente (campioni silenziosi registrati a 7.5 ips, campioni rumorosi registrati a 7.5 ips, campioni silenziosi registrati a 15 ips e campioni rumorosi registrati a 15 ips) e valutandone separatamente le prestazioni. Inoltre, per ciascuno dei *dataset* sono state utilizzate quattro tipologie di etichettature dei campioni, in seguito elencate.

- *Tipologia 1*: etichetta 0 ai campioni equalizzati correttamente, etichetta 1 ai campioni equalizzati in modo errato;

- *Tipologia 2*: etichetta 0 ai campioni equalizzati correttamente, etichetta 1 ai campioni pre-equalizzati CCIR e post-equalizzati NAB, etichetta 2 ai campioni pre-equalizzati NAB e post-equalizzati CCIR;

- *Tipologia 3*: etichetta 0 ai campioni post-equalizzati CCIR, etichetta 1 ai campioni post-equalizzati NAB;

- *Tipologia 4*: etichetta 0 ai campioni pre-equalizzati e post-equalizzati CCIR, etichetta 1 ai campioni pre-equalizzati CCIR e post-equalizzati NAB, etichetta 2 ai campioni pre-equalizzati NAB e post-equalizzati CCIR, etichetta 3 ai campioni pre-equalizzati e post-equalizzati NAB.

I dati con le rispettive etichettature sono necessari per :

- fornire agli algoritmi le informazioni necessarie per allenare i modelli per la classificazione di nuovi dati;

- valutare le *performance* di ciascun classificatore.

Il *dataset* che verrà utilizzato per la costruzione e la validatione dei modelli è stato diviso in due parti:

- per il 75%, **training set**;

- per il restante 25%, **test set**.

Il *training set* contiene i dati che verranno utilizzati per "allenare" ciascun modello e validarlo mediante *10-fold cross-validation*. Invece, il *test set* contiene i dati che verranno utilizzati per valutare le *perfomance* della classificazione su dati diversi da quelli utilizzati per addestrare il modello.

Si vedrà che alcuni modelli si caratterizzano per errore di cross-validazione pari a 0, accompagnato da un'*accuracy* nella predizione di nuovi dati del 100%. Per ulteriori dettagli circa i risultati dell'esperimento, si rimanda al Capitolo 4.

Nei tre paragrafi successivi verrà descritta la struttura del codice utilizzato per valutare le prestazioni dei modelli scelti.

3.5.1 Decision Tree

I passi fondamentali della routine MATLAB per la valutazione delle prestazioni del classificatore *Decision Tree*, o albero di decisione, sono elencati in seguito.

1 Scelta del numero massimo di ramificazioni dell'albero.

2 Costruzione dell'albero di decisione utilizzando la funzione *fitctree* del *Statistics and Machine Learning toolbox* di MATLAB, che prende in input l'insieme delle osservazioni, o *training set*, e le rispettive etichette, *output* o *label*.

3 Partizionamento del modello costruito al passo 2 in 10 sotto-modelli, utilizzando la funzione *crossval* del *Statistics and Machine Learning toolbox* di MATLAB. Ciascuno dei sotto-modelli viene addestrato con 9 di 10 parti disgiunte di ugual dimensione del *training set* e validato con la decima, cambiando ogni volta partizione (tecnica *10-fold cross-validation*).

4 Stima dell'errore medio di cross-validazione utilizzando la funzione *kfoldLoss* del *Statistics and Machine Learning toolbox* di MATLAB. Tale funzione calcola l'errore di classificazione del *training set* per ciascuno dei sotto-modelli costruiti al passo 3, e dopo ne restituisce la media.

5 Scelta del sotto-modello migliore, cioè quello con errore di predizione più basso.

6 Predizione della classe di appartenenza delle osservazioni del *training set* utilizzando il modello scelto al passo 5, servendosi della funzione *predict* del *Statistics and Machine Learning toolbox* di MATLAB.

7 Calcolo della matrice di confusione e valutazione delle *performance* del modello nella classificazione dei dati di *training*.

8 Predizione della classe di appartenenza delle osservazioni del *test set* utilizzando il modello scelto al passo 5, servendosi della funzione *predict* del *Statistics and Machine Learning toolbox* di MATLAB.

9 Calcolo della matrice di confusione e valutazione delle *performance* del modello nella classificazione dei dati di *test*.

10 Ripetizione dei passi da 2 a 9, cambiando ogni volta la scelta del numero massimo di ramificazioni consentite nella costruzione dell'albero di decisione. I valori testati sono 4, 20, 100.

11 Costruzione di una tabella contenente i risultati ottenuti nei passi precedenti, per ogni scelta dei parametri dell'algoritmo.

12 Salvataggio della tabella costruita al passo 11 in un file con estensione **.cvs*, poi importata in un documento *Excel* per maggiore leggibilità.

In Appendice A viene mostrato il codice della funzione MATLAB *decision_tree_perform*, implementata per valutare le prestazioni di *decision tree*.

3.5.2 Nearest Neighbors

I passi fondamentali della routine MATLAB per la valutazione delle prestazioni del classificatore *K-Nearest Neighbors*, o KNN, sono elencati in seguito.

1 Scelta del numero $NumNeighbors$ di *nearest neighbors* e della metrica di distanza.

2 Costruzione del classificatore KNN utilizzando la funzione *fitcknn* del *Statistics and Machine Learning toolbox* di MATLAB, che prende in input l'insieme delle osservazioni, o *training set*, e le rispettive etichette, *output* o *label*.

3 Partizionamento del modello costruito al passo 2 in 10 sotto-modelli, utilizzando la funzione *crossval* del *Statistics and Machine Learning toolbox* di MATLAB. Ciascuno dei sotto-modelli viene addestrato con 9 di 10 parti disgiunte di ugual dimensione del *training set* e validato con la decima, cambiando ogni volta partizione (tecnica *10-fold cross-validation*).

4 Stima dell'errore medio di cross-validazione utilizzando la funzione *kfoldLoss* del *Statistics and Machine Learning toolbox* di MATLAB. Tale funzione calcola l'errore di classificazione sul *training set* per ciascuno dei sotto-modelli costruiti al passo 3, e dopo ne restituisce la media.

5 Scelta del sotto-modello migliore, cioè quello con errore di predizione più basso.

6 Predizione della classe di appartenenza delle osservazioni del *training set* utilizzando il modello scelto al passo 5, servendosi della funzione *predict* del *Statistics and Machine Learning toolbox* di MATLAB.

7 Calcolo della matrice di confusione e valutazione delle *performance* del modello nella classificazione dei dati di *training*.

8 Predizione della classe di appartenenza delle osservazioni del *test set* utilizzando il modello scelto al passo 5, servendosi della funzione *predict* del *Statistics and Machine Learning toolbox* di MATLAB.

9 Calcolo della matrice di confusione e valutazione delle *performance* del modello nella classificazione dei dati di *test*.

10 Ripetizione dei passi da 2 a 9, cambiando ogni volta $NumNeighbors$ e/o tipologia di distanza. Le coppie ($NumNeighbors$, tipo di distanza) testate sono le seguenti:

- (1, Euclidea);
- (5, Euclidea);
- (10, Euclidea);
- (1, coseno);
- (1, distanza inversa).

11 Costruzione di una tabella contenente i risultati ottenuti nei passi precedenti, per ogni scelta dei parametri dell'algoritmo.

12 Salvataggio della tabella costruita al passo 11 in un file con estensione **.cvs*, poi importata in un documento *Excel* per maggiore leggibilità.

In Appendice A viene mostrato il codice della funzione MATLAB *KNN_perform*, implementata per valutare le prestazioni di *k-nearest neighbors*.

3.5.3 Support Vector Machine

I passi fondamentali della routine MATLAB per la valutazione delle prestazioni del classificatore *Support Vector Machine*, o SVM, sono elencati in seguito.

1 Scelta del tipo di funzione *kernel* da utilizzare per la costruzione del modello *SVM*.

2 Costruzione del classificatore SVM utilizzando la funzione *fitcecoc* del *Statistics and Machine Learning toolbox* di MATLAB per il *fitting* di modelli multi-classe. La funzione prende in input la funzione *kernel* scelta al passo 1, l'insieme delle osservazioni, o *training set*, e le rispettive etichette, *output* o *label*.

3 Partizionamento del modello costruito al passo 2 in 10 sotto-modelli, utilizzando la funzione *crossval* del *Statistics and Machine Learning toolbox* di MATLAB. Ciascuno dei sotto-modelli viene addestrato con 9 di 10 parti disgiunte di ugual dimensione del *training set* e validato con la decima, cambiando ogni volta partizione (tecnica *10-fold cross-validation*).

4 Stima dell'errore medio di cross-validazione utilizzando la funzione *kfoldLoss* del *Statistics and Machine Learning toolbox* di MATLAB. Tale funzione calcola l'errore di classificazione sul *training set* per ciascuno dei sotto-modelli costruiti al passo 3, e dopo ne restituisce la media.

5 Scelta del sotto-modello migliore, cioè quello con errore di predizione più basso.

6 Predizione della classe di appartenenza delle osservazioni del *training set* utilizzando il modello scelto al passo 5, servendosi della funzione *predict* del *Statistics and Machine Learning toolbox* di MATLAB.

7 Calcolo della matrice di confusione e valutazione delle *performance* del modello nella classificazione dei dati di *training*.

8 Predizione della classe di appartenenza delle osservazioni del *test set* utilizzando il modello scelto al passo 5, servendosi della funzione *predict* del *Statistics and Machine Learning toolbox* di MATLAB.

9 Calcolo della matrice di confusione e valutazione delle *performance* del modello nella classificazione dei dati di *test*.

10 Ripetizione dei passi da 2 a 9, cambiando ogni volta il tipo di funzione *kernel*. Sono state testate le funzioni *kernel* lineare, polinomiale di ordine 2, polinomiale di ordine 3 e gaussiana.

11 Costruzione di una tabella contenente i risultati ottenuti nei passi precedenti, per ogni scelta dei parametri dell'algoritmo.

12 Salvataggio della tabella costruita al passo 11 in un file con estensione **.cvs*, poi importata in un documento *Excel* per maggiore leggibilità.

In Appendice A viene mostrato il codice della funzione MATLAB *SVM_perform*, implementata per valutare le prestazioni di *support vector machine*.

Capitolo 4

Risultati sperimentali

In questo capitolo verranno illustrati i risultati dell'esperimento descritto nel capitolo precedente. In particolare, in Sezione 4.1 verranno descritti i risultati della *cluster analysis*, utilizzando come *feature* sia le componenti dell'intero spettro, sia i coefficienti cepstrali dei campioni audio, mentre in Sezione 4.2 si illustreranno i risultati dei *test* relativi alla classificazione, utilizzando come *feature* solo i coefficienti cepstrali.
I *test* sono stati effettuati in modo separato per ciascuno dei quattro *dataset* descritti nel capitolo precedente.

Per semplicità, nei prossimi paragrafi ciascun *dataset* verrà chiamato nei modi indicati nell'elenco successivo.

Dataset A : campioni silenziosi registrati e riprodotti a 7.5ips;

Dataset B : campioni con rumore bianco registrati e riprodotti a 7.5ips;

Dataset C : campioni silenziosi registrati e riprodotti a 15ips;

Dataset D : campioni con rumore bianco registrati e riprodotti a 15ips.

4.1 Clustering analysis

L'obiettivo della prima parte dell'esperimento è di capire, tramite *cluster analysys*, se sia possibile separare i campioni in base alle equalizzazioni applicate. Questa fase è tornata molto utile per comprendere quali tipologie di *feature* fossero più adatte allo scopo.
È stata effettuata la *cluster analysys* utilizzando come *feature* dapprima l'intero spettro, in un secondo momento i coefficienti cepstrali dei campioni audio. Alla fine si è scelto di continuare la sperimentazione utilizzando quest'ultima tipologia di *feature*, in quanto in grado di caratterizzare i campioni audio in modo più compatto e preciso. Infatti, entrambe le tipologie di *feature* si sono rilevate sufficienti per separare i campioni in base alle equalizzazioni, ma mentre lo spettro contiene 48000 componenti (perché i campioni sono stati digitalizzati a 96 kHz), il vettore dei coefficienti cepstrali ne contiene solo 13.

4.1.1 *Clustering analysis* con spettro

Dalla *cluster analysis*, utilizzando come *feature* le componenti dello spettro, sono emerse due principali tendenze, elencate in seguito.

1 Capacità di separare i campioni dei *dataset* A e C in due *cluster*: in uno vengono inseriti i campioni con post-equalizzazione CCIR, nell'altro quelli con post-equalizzazione NAB. Questo risultato si è manifestato in modo più evidente con *k-means clustering*. Infatti, mentre l'algoritmo di *hierarchical clustering* separa i campioni solo con alcune configurazioni dei parametri, *k-means clustering* li separa con quasi tutte le configurazioni testate.

2 Capacità di separare i campioni dei *dataset* B e D in tre *cluster*: nel primo vengono inseriti i campioni con pre-equalizzazione e post-equalizzazione dello stesso tipo (NAB o CCIR), nel secondo quelli con pre-equalizzazione NAB e post-equalizzazione CCIR, nel terzo quelli con pre-equalizzazione CCIR e post-equalizzazione NAB. L'unico algoritmo che ha portato a questo risultato è stato *k-means clustering*.

In questa prima parte dell'esperimento, l'algoritmo più performante, in generale, si è rilevato *k-means clustering*. Per i dettagli numerici consultare Appendice B.

4.1.2 *Clustering analysis* con coefficienti cepstrali

Dalla *cluster analysis*, utilizzando come *feature* i coefficienti cepstrali, sono emerse tendenze del tutto analoghe a quelle descritte nel paragrafo precedente. La differenza principale rispetto al caso precedente consiste nell'abbattimento dei tempi di calcolo, che sono scesi dall'ordine del minuto all'ordine del millisecondo. Per questa ragione, si è scelto di proseguire l'indagine utilizzando solo i coefficienti cepstrali.
Per completezza, sono in seguito elencate le due principali tendenze riscontrate.

1 Capacità di separare i campioni dei *dataset* A e C in due *cluster*: in uno vengono inseriti i campioni con post-equalizzazione CCIR, nell'altro quelli con post-equalizzazione NAB. Come nel caso precedente, questa tendenza si è manifestata in modo più evidente con *k-means clustering*.

2 Capacità di separare i campioni dei *dataset* B e D in tre *cluster*: nel primo sono stati inseriti i campioni con pre-equalizzazione e post-equalizzazione dello stesso tipo (NAB o CCIR), nel secondo quelli con pre-equalizzazione NAB e post-equalizzazione CCIR, nel terzo quelli con pre-equalizzazione CCIR e post-equalizzazione NAB. A differenza del caso precedente, l'unico algoritmo che ha portato a questo risultato per entrambi i *dataset* è stato *hierarchical clustering*.

Per i dettagli numerici consultare Appendice C.

Dataset	Prediction classes	Average training time (ms)	Average cross-validation error	Average accuracy on training set	Average test time (ms)	Average accuracy on test set
7.5 ips, silence	correct Eq. / wrong Eq.	132,33	0,2853	0,8702	0,9025	0,6367
7.5 ips, silence	correct Eq. / CCIR_NAB Eq. / NAB_CCIR Eq.	1173,48	0,2609	0,8902	222,15	0,6611
7.5 ips, silence	NAB post-Eq. / CCIR post-Eq.	193,79	0,0169	0,9970	37,35	0,9869
7.5 ips, silence	CCIR_CCIR Eq. / CCIR_NAB Eq./ NAB_CCIR Eq. / NAB_NAB Eq.	20,67	0,2628	0,8953	4	0,6575
7.5 ips, white noise	correct Eq. / wrong Eq.	170,82	0,1286	0,9183	47,51	0,9203
7.5 ips, white noise	correct Eq. / CCIR_NAB Eq. / NAB_CCIR Eq.	293,53	0,0949	0,9626	78,91	0,9619
7.5 ips, white noise	NAB post-Eq. / CCIR post-Eq.	126,73	0,2230	0,9146	39,30	0,8872
7.5 ips, white noise	CCIR_CCIR Eq. / CCIR_NAB Eq./ NAB_CCIR Eq. / NAB_NAB Eq.	93,96	0,2213	0,9299	35,81	0,8964
15 ips, silence	correct Eq. / wrong Eq.	48,32	0,4244	0,8031	46,56	0,5447
15 ips, silence	correct Eq. / CCIR_NAB Eq. / NAB_CCIR Eq.	85,35	0,4255	0,7823	167,63	0,5561
15 ips, silence	NAB post-Eq. / CCIR post-Eq.	173,87	0,0083	0,9978	35,46	0,9939
15 ips, silence	CCIR_CCIR Eq. / CCIR_NAB Eq./ NAB_CCIR Eq. / NAB_NAB Eq.	10,54	0,42093	0,78185	4,97	0,55028
15 ips, white noise	correct Eq. / wrong Eq.	182	0,0988	0,9373	49,42	0,9333
15 ips, white noise	correct Eq. / CCIR_NAB Eq. / NAB_CCIR Eq.	258,37	0,0647	0,9794	35,15	0,9750
15 ips, white noise	NAB post-Eq. / CCIR post-Eq.	121,09	0,2697	0,8225	35,58	0,7575
15 ips, white noise	CCIR_CCIR Eq. / CCIR_NAB Eq./ NAB_CCIR Eq. / NAB_NAB Eq.	89,48	0,2882	0,8598	6,6092	0,7928

Figura 4.1: Prestazioni medie dei classificatori.

4.2 Classificazione

Come anticipato nel capitolo precedente, per ciascuno dei quattro *dataset* sono state valutate le prestazioni di tre classificatori, *decision tree*, *support vector machine* e *k-nearest neighbors*, al variare di alcuni parametri. Come metriche per la misura delle prestazioni sono stati utilizzati:

- tempo di *training*;
- errore di cross-validazione;
- *accuracy* nella predizione del *training set* con il modello scelto mediante *cross-validation*;
- *recall, specificity, precision* e AUC (rispetto a ciascuna delle classi, nel caso di classificazione multiclasse), nella predizione del *training set* con il modello scelto mediante *cross-validation*;
- tempo di classificazione di nuovi dati (*test set*);
- *accuracy* nella predizione del *test set* con il modello scelto mediante *cross-validation*;
- *recall, specificity, precision* e AUC (rispetto a ciascuna delle classi, nel caso di classificazione multiclasse), nella predizione del *test set* con il modello scelto mediante *cross-validation*.

La macchina utilizzata per i *test* è un *MacBook Pro* con processore *Intel Core i7* da 2 GHz e 2 GB di RAM.
La tabella di Figura 4.1 descrive le prestazioni medie dei classificatori testati, per tutte le combinazioni di *dataset* e classi di predizione. Alcuni dati della tabella sono illustrati sotto forma di istogramma nelle Figure 4.2, 4.3, 4.4 e 4.5.

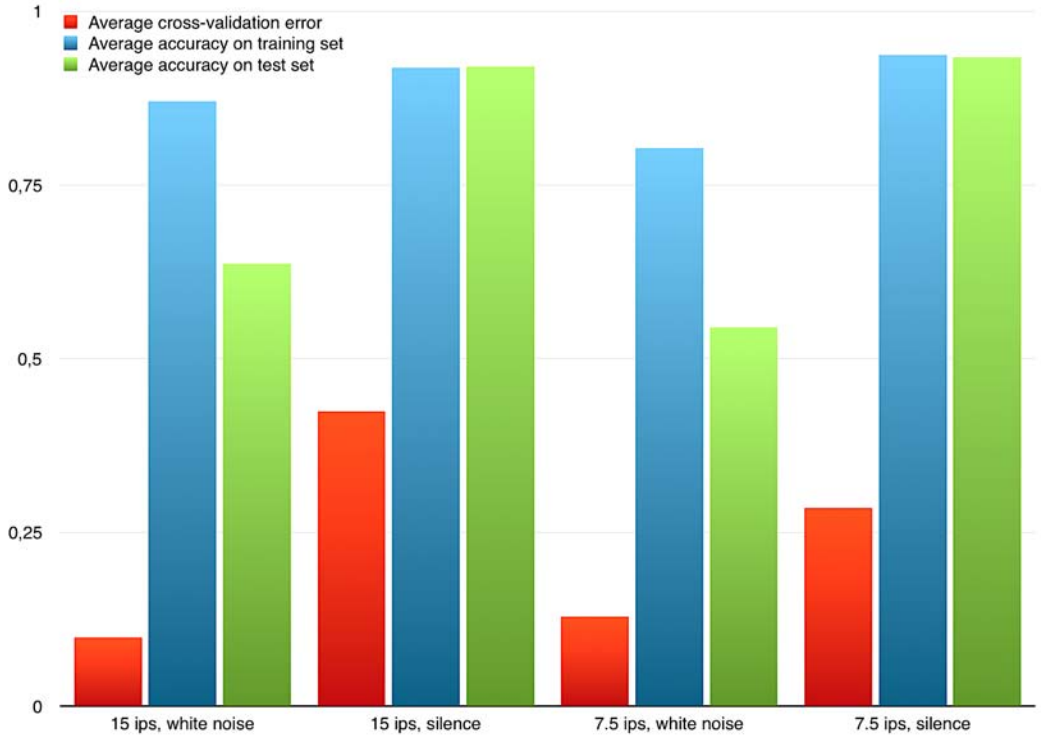

Figura 4.2: Prestazioni dei classificatori sulle classi di predizione *eq. corretta / eq. errata*.

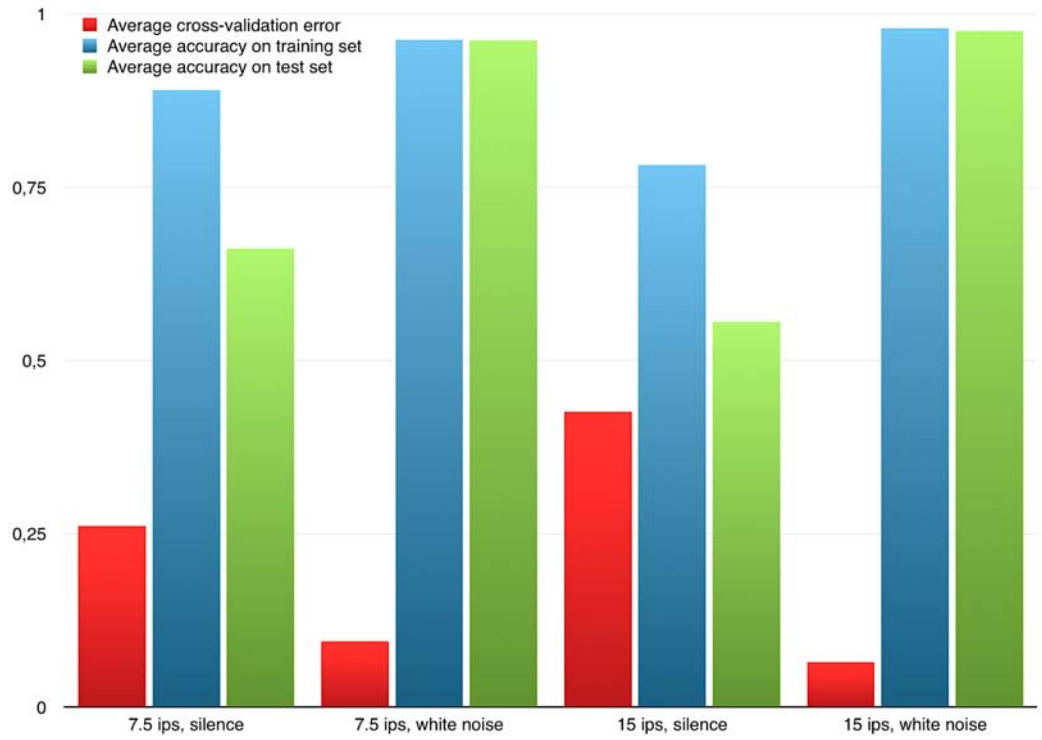

Figura 4.3: Prestazioni dei classificatori sulle classi di predizione *eq. corretta / pre-eq. CCIR, post-eq. NAB / pre-eq. NAB, post-eq. CCIR*.

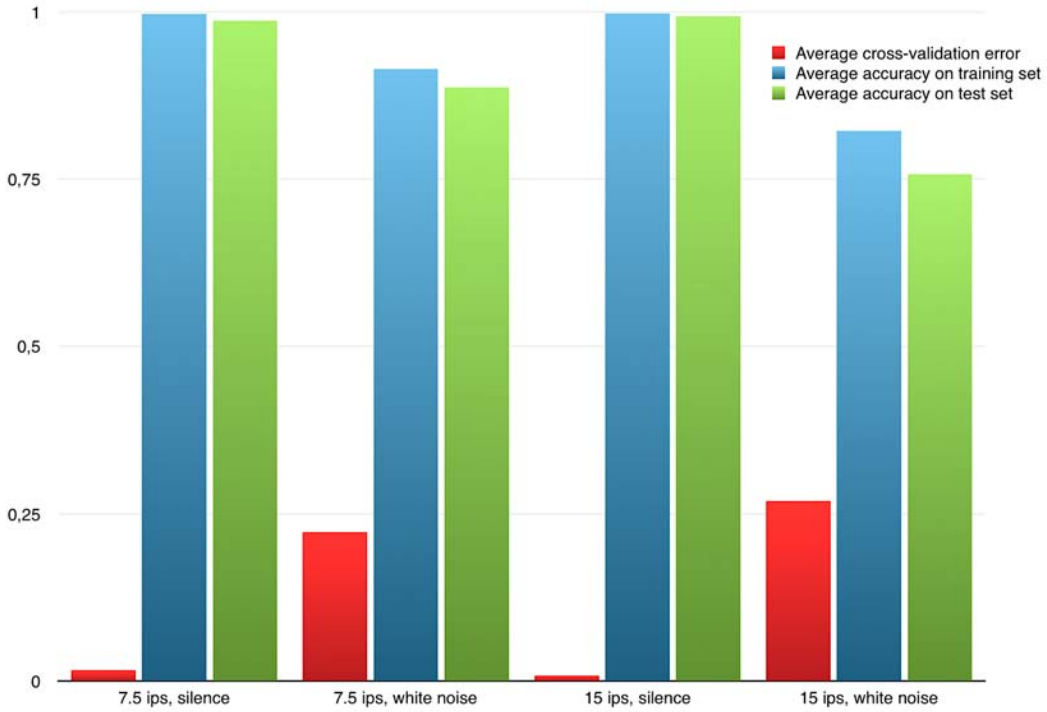

Figura 4.4: Prestazioni dei classificatori sulle classi di predizione *post-eq. CCIR / post-eq. NAB*.

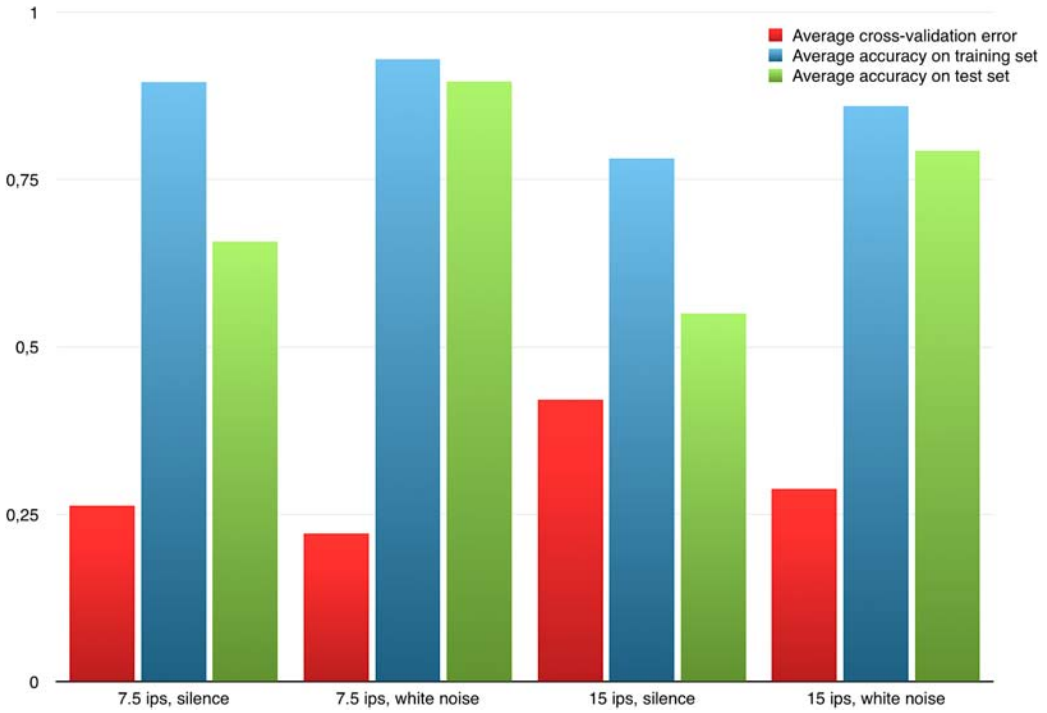

Figura 4.5: Prestazioni dei classificatori sulle classi di predizione *pre-eq. CCIR, post-eq. CCIR / pre-eq. CCIR, post-eq. NAB / pre-eq. NAB, post-eq. CCIR* / pre-eq. NAB, post-eq. NAB.

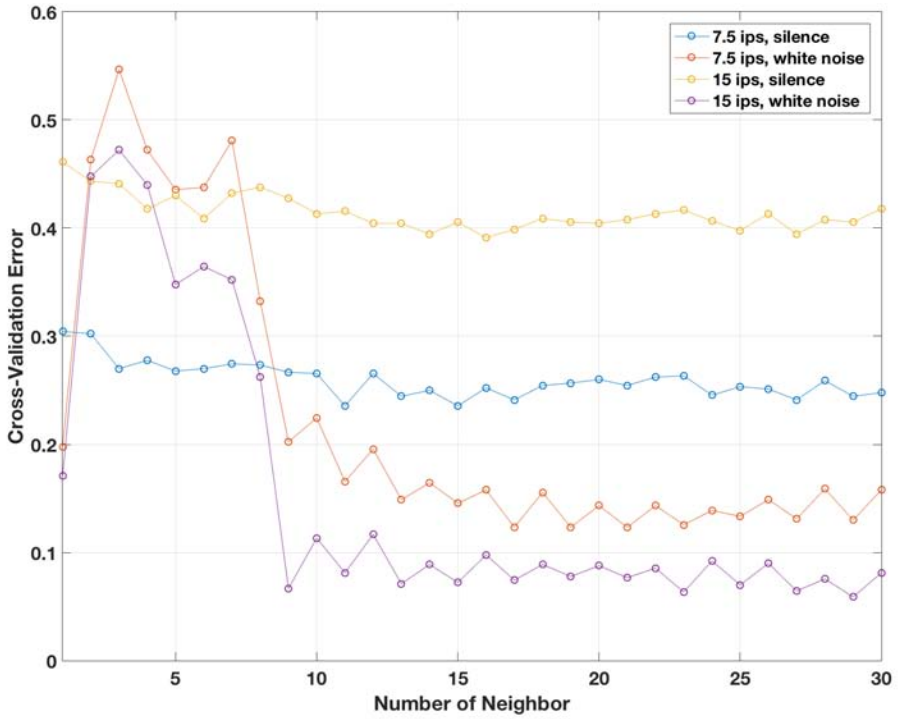

Figura 4.6: Prestazioni di *KNN* al variare di k.

In Figura 4.6 viene mostrato l'errore di *cross-validation* del classificatore KNN al variare di K, per ciascuno dei quattro *dataset*.

In Figura 4.7, invece, viene mostrato l'errore di *cross-validation* del classificatore *decision tree* al variare del numero massimo di ramificazioni concesse durante il *training*, per ciascuno dei quattro *dataset*.

Per maggiori dettagli sulle *performance* dei classificatori consultare Appendice D.

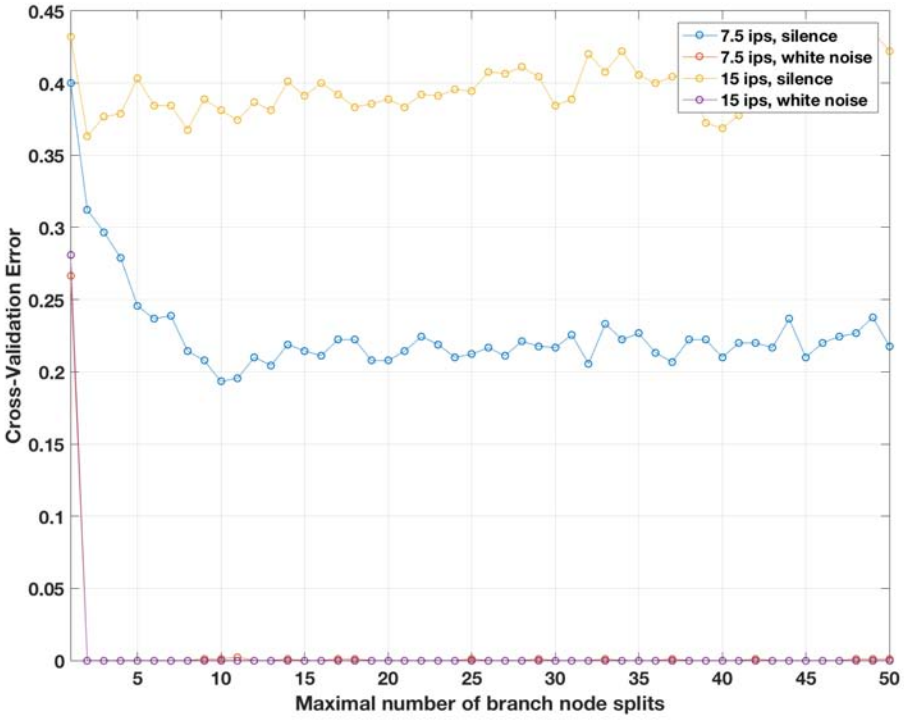

Figura 4.7: Prestazioni di *Decision Tree* al variare del numero massimo di ramificazioni concesse durante il *training*.

Capitolo 5

Conclusioni e sviluppi futuri

È stato mostrato come sia possibile scoprire, utilizzando tecniche di apprendimento automatico, se le equalizzazioni di pre-enfasi e post-enfasi utilizzate, rispettivamente, nella registrazione e nella riproduzione di documenti sonori analogici, siano applicate in modo corretto o errato. In particolare, si è visto come, mediante *cluster analysis*, sia possibile individuare dei raggruppamenti naturali all'interno di *dataset* contententi campioni audio registrati a 7.5 o 15 ips, con curve di pre-enfasi e post-enfasi secondo gli standard CCIR e NAB.
Dalla valutazione delle *performance* di algoritmi di classificazione applicati su diverse tipologie di campioni audio, rispetto a determinate classi di predizione, è emersa la capacità, di alcuni modelli, di ottenere un errore di cross-validazione inferiore al 10% e un'*accuracy* nella predizione del *test set* pari al 100%.
Inoltre, l'indagine ha mostrato che l'utilizzo dei coefficienti cepstrali dei campioni come *feature* per l'apprendimento automatico sia una buona scelta, perché sono molto precisi e compatti nel descrivere la natura dei dati e comportano oneri computazionali molto bassi.
Inoltre, è emerso che, in linea di massima, i campioni più adatti a capire se l'equalizzazione applicata sia quella giusta, sono quelli con rumore bianco, mentre quelli silenziosi si prestano meglio per il riconoscimento della curva di post-enfasi applicata.

Possibili sviluppi futuri sono elencati in seguito.

- Per ogni tipo di predizione, *tuning* degli iperparametri del modello che si è dimostrato più performante.

- Individuazione e selezione delle *feature* migliori tra i 13 coefficienti cepstrali, cercando il *trade-off* ottimale tra numero di *feature* utilizzate per la classificazione e *accuracy* nella predizione di nuovi dati.

- Valutazione delle *performance* dei modelli al diminuire della durata dei campioni con cui vengono addestrati, cercando il *trade-off* ottimale tra durata e *accuracy* nella predizione di nuovi dati.

- Valutazione delle *performance* dei modelli su nuove tipologie di *dataset* simulati. In particolare, si potrebbero effettuare nuovi *test* con dati registrati su diversi modelli di nastro magnetico e utilizzando *Analogic Tape*

Recorder e standard di pre-enfasi e post-enfasi diversi da quelli utilizzati nell'esperimento descritto in questo elaborato.

- Valutazione delle *performance* dei modelli al variare dell'ampiezza dei disturbi del segnale audio, come quelli di orgine elettrica (per esempio, il ronzio in bassa frequenza dovuto alla corrente alternata di alimentazione, detto comunemente *hum*).

- Valutazione delle *performance* dei modelli su *dataset* reali.

I risultati di questa e di future indagini potrebbero essere utilizzati per sviluppare *software* applicativi che, ricevendo in *input* documenti sonori analogici, siano in grado di stabilire autonomamente:

- se l'equalizzazione di post-enfasi applicata sia quella giusta o meno;

- quali siano le curve di pre-enfasi e/o di post-enfasi applicate;

- se il dispositivo utilizzato per la registrazione e/o riproduzione contenga errori di taratura.

Bibliografia

[1] M. Camras. *Magnetic Recording Books*. VNB, 1988.

[2] C. Voci P. Mazzoldi, M. Nigro. *Fisica Volume II*. EdiSES, 2007.

[3] John D. Jackson. *Classical Electrodynamics*. Wiley, 1999.

[4] C. Tagliabue. *La Registrazione Magnetica del Suono*. Rostro, 1975.

[5] John D. Jackson. Studer a810, operating and service instructions. `https://www.manualslib.com/manual/997470/Studer-A810.html`.

[6] V. Burini F. Altieri, S. Canazza. Rilevamenti sperimentali per la conservazione attiva dei documenti sonori su nastro magnetico: Individuazione delle curve di equalizzazione. *Atti del XXI CIM, Cagliari*, 2016.

[7] Tom M. Mitchell. *Machine Learning*. McGraw-Hill International Editions, Computer Science Series, 1997.

[8] Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 1959.

[9] MathWorks. Matlab documentation. `https://it.mathworks.com/help/matlab/index.html`, 2017.

[10] J. Friedman T.Hastie, R. Tibshirani. *The Elements of Statistical Learning*. Springer, 2008.

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

[12] Vladimir Estivill-Castro. Why so many clustering algorithms — a position paper. *ACM SIGKDD Explorations Newsletter*, 4(1):65–75, 2002.

[13] Hans-Peter; Sander Jörg; Xu Xiaowei Ester, Martin; Kriegel. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press*, page 226–231, 1996.

[14] Olivier Lartillot. Mir toolbox 1.6.1, user's manual, 2014.

[15] V. Burini. *Filologia digitale per la conservazione degli archivi sonori*. Tesi di laurea, SAE Institute, Milan, 2016.

[16] Louis D. Fielder. Pre- and postemphasis techniques as applied to audio recording systems. *J. Audio Eng. Soc*, 33(9):649–658, 1985.

[17] Engineering Department National Association of Broadcasters. *Nab Magnetic Tape Recording And Reproducing Standards*. Engineering Department National Association of Broadcasters, 1965.

Elenco delle figure

Appendice A

Codice MATLAB

Nelle pagine successive viene illustrato il codice delle funzioni MATLAB di supporto per la *cluster analysis* e per la valutazione delle prestazioni dei classificatori.

In particolare, verranno mostrate le funzioni:

- *cep_hiearchical_clustering*, che prende in *input* il nome del *dataset* da analizzare e il numero massimo di *cluster* consentiti, e restituisce una tabella con i risultati dell'algoritmo *hierarchical clustering* al variare dei parametri, applicato al *dataset* specificato;

- *cep_k_means_clustering*, che prende in *input* il nome del *dataset* da analizzare, e restituisce una tabella con i risultati dell'algoritmo *k-means clustering* al variare dei parametri, applicato al *dataset* specificato;

- $decision\_tree\_perform$, che prende in *input* le osservazioni X del *training set*, con i rispettivi *output* (X_output), le osservazioni T del *test set*, con i rispettivi *output* (T_output) e il numero massimo di ramificazioni consentite, e restituisce due *structure array* con le misure delle *performance* di validazione e di *test* del classificatore *decision tree*, applicato ai dati in *input*;

- $KNN\_perform$, che prende in *input* le osservazioni X del *training set*, con i rispettivi *output* (X_output), le osservazioni T del *test set*, con i rispettivi *output* (T_output), il numero di *neighbors* e la metrica di *distanza*, e restituisce due *structure array* con le misure delle *performance* di validazione e di *test* del classificatore *KNN*, applicato ai dati in *input*;

- $SVM\_perform$, che prende in *input* le osservazioni X del *training set*, con i rispettivi *output* (X_output), le osservazioni T del *test set*, con i rispettivi *output* (T_output) e il tipo di funzione *kernel*, e restituisce due *structure array* con le misure delle *performance* di validazione e di *test* del classificatore *SVM*, applicato ai dati in *input*.

```
function tab = cep_hiearchical_clustering(filename,max_clust)

% Hierarchical clustering with cepstral coefficients

% Load data
filename2 = strcat('../../data/','cepstral_',filename,'.mat');
load(filename2);

% Combine samples in a single matrix
X = [cep_ccir_ccir; cep_ccir_nab; cep_nab_ccir; cep_nab_nab];

% Output data
MAX_CLUSTER = [];
DIST_METRIC = [];
LINK_METHOD = [];
CCIR_CCIR_clust1 = [];
CCIR_CCIR_clust2 = [];
CCIR_CCIR_clust3 = [];
CCIR_CCIR_clust4 = [];
CCIR_NAB_clust1 = [];
CCIR_NAB_clust2 = [];
CCIR_NAB_clust3 = [];
CCIR_NAB_clust4 = [];
NAB_CCIR_clust1 = [];
NAB_CCIR_clust2 = [];
NAB_CCIR_clust3 = [];
NAB_CCIR_clust4 = [];
NAB_NAB_clust1 = [];
NAB_NAB_clust2 = [];
NAB_NAB_clust3 = [];
NAB_NAB_clust4 = [];
COPHENET= [];

distances = {euclidean};
linkages = {average','centroid','complete','median','single','ward','weighted};
link_size = length(linkages);

for i = 1 : link_size

    dist = distances{1};
    link = linkages{i};

    % Calculate the distance between every pair of objects in a data set
    Y = pdist(X, dist);

    % Determine how objects in the data set should be grouped into clusters
    Z = linkage(Y,link);

    % Create clusters
    T = cluster(Z,'maxclust',max_clust);

    % Verify dissimilarity
    c = cophenet(Z,Y);

    % Count the number of samples inserted in each cluster
    V = f_verify_cluster(T);

    % Update output data
    MAX_CLUSTER = [MAX_CLUSTER; max_clust];
    DIST_METRIC = [DIST_METRIC; distances(1)];
    LINK_METHOD = [LINK_METHOD; linkages(i)];
    CCIR_CCIR_clust1 = [CCIR_CCIR_clust1; V(1,1)];
    CCIR_CCIR_clust2 = [CCIR_CCIR_clust2; V(1,2)];
    CCIR_CCIR_clust3 = [CCIR_CCIR_clust3; V(1,3)];
    CCIR_CCIR_clust4 = [CCIR_CCIR_clust4; V(1,4)];
    CCIR_NAB_clust1 = [CCIR_NAB_clust1; V(2,1)];
    CCIR_NAB_clust2 = [CCIR_NAB_clust2; V(2,2)];
    CCIR_NAB_clust3 = [CCIR_NAB_clust3; V(2,3)];
    CCIR_NAB_clust4 = [CCIR_NAB_clust4; V(2,4)];
    NAB_CCIR_clust1 = [NAB_CCIR_clust1; V(3,1)];
    NAB_CCIR_clust2 = [NAB_CCIR_clust2; V(3,2)];
    NAB_CCIR_clust3 = [NAB_CCIR_clust3; V(3,3)];
    NAB_CCIR_clust4 = [NAB_CCIR_clust4; V(3,4)];
    NAB_NAB_clust1 = [NAB_NAB_clust1; V(4,1)];
    NAB_NAB_clust2 = [NAB_NAB_clust2; V(4,2)];
    NAB_NAB_clust3 = [NAB_NAB_clust3; V(4,3)];
    NAB_NAB_clust4 = [NAB_NAB_clust4; V(4,4)];
    COPHENET = [COPHENET; c];
end

distances = {squaredeuclidean','seuclidean','cityblock'...
    'minkowski','chebychev','cosine','correlation',...
    'spearman','hamming','jaccard'};

linkages = {average','complete','single','weighted};

dist_size = length(distances);
```

```
link_size = length(linkages);

for j = 1 : dist_size
    for i = 1 : link_size

        dist = distances{j};
        link = linkages{i};

        % Calculate the distance between every pair of objects in a data set
        Y = pdist(X, dist);

        % Determine how objects in the data set should be grouped into clusters
        Z = linkage(Y,link);

        % Create clusters
        T = cluster(Z,'maxclust',max_clust);

        % Count the number of samples inserted in each cluster
        V = f_verify_cluster(T);

        % Update output data
        MAX_CLUSTER = [MAX_CLUSTER; max_clust];
        DIST_METRIC = [DIST_METRIC; distances(j)]
        LINK_METHOD = [LINK_METHOD; linkages(i)];
        CCIR_CCIR_clust1 = [CCIR_CCIR_clust1; V(1,1)];
        CCIR_CCIR_clust2 = [CCIR_CCIR_clust2; V(1,2)];
        CCIR_CCIR_clust3 = [CCIR_CCIR_clust3; V(1,3)];
        CCIR_CCIR_clust4 = [CCIR_CCIR_clust4; V(1,4)];
        CCIR_NAB_clust1 = [CCIR_NAB_clust1; V(2,1)];
        CCIR_NAB_clust2 = [CCIR_NAB_clust2; V(2,2)];
        CCIR_NAB_clust3 = [CCIR_NAB_clust3; V(2,3)];
        CCIR_NAB_clust4 = [CCIR_NAB_clust4; V(2,4)];
        NAB_CCIR_clust1 = [NAB_CCIR_clust1; V(3,1)];
        NAB_CCIR_clust2 = [NAB_CCIR_clust2; V(3,2)];
        NAB_CCIR_clust3 = [NAB_CCIR_clust3; V(3,3)];
        NAB_CCIR_clust4 = [NAB_CCIR_clust4; V(3,4)];
        NAB_NAB_clust1 = [NAB_NAB_clust1; V(4,1)];
        NAB_NAB_clust2 = [NAB_NAB_clust2; V(4,2)];
        NAB_NAB_clust3 = [NAB_NAB_clust3; V(4,3)];
        NAB_NAB_clust4 = [NAB_NAB_clust4; V(4,4)];
        COPHENET = [COPHENET; c];
    end
end

% Insert output data into a table
tab = table(MAX_CLUSTER, DIST_METRIC, LINK_METHOD, CCIR_CCIR_clust1,
    CCIR_NAB_clust1,NAB_CCIR_clust1,..
    NAB_NAB_clust1,CCIR_CCIR_clust2,CCIR_NAB_clust2,NAB_CCIR_clust2,
    NAB_NAB_clust2,CCIR_CCIR_clust3,CCIR_NAB_clust3,NAB_CCIR_clust3,
    NAB_NAB_clust3,CCIR_CCIR_clust4,CCIR_NAB_clust4,NAB_CCIR_clust4,
    NAB_NAB_clust4,COPHENET);

% Save table in .cvs file
table_name = strcat('../results/Cep_Hierarchical_Clustering_',filename,'.csv');
writetable(tab, table_name,'Delimiter',',','QuoteStrings',true);

end
```

```
function tab = cep_k_means_clustering(filename)

% K-means clustering with cepstral coefficients

% Load data
filename2 = strcat('../../data/','cepstral_',filename,'.mat');
load(filename2);

% Combine samples in a single matrix
X = [cep_ccir_ccir; cep_ccir_nab; cep_nab_ccir; cep_nab_nab];

% Output data
CLUSTERS = [];
DIST_METRIC = [];
AVERAGE_SIL = [];
CCIR_CCIR_clust1 = [];
CCIR_CCIR_clust2 = [];
CCIR_CCIR_clust3 = [];
CCIR_CCIR_clust4 = [];
CCIR_NAB_clust1 = [];
CCIR_NAB_clust2 = [];
CCIR_NAB_clust3 = [];
CCIR_NAB_clust4 = [];
NAB_CCIR_clust1 = [];
NAB_CCIR_clust2 = [];
NAB_CCIR_clust3 = [];
NAB_CCIR_clust4 = [];
NAB_NAB_clust1 = [];
NAB_NAB_clust2 = [];
NAB_NAB_clust3 = [];
NAB_NAB_clust4 = [];

distances = {'sqeuclidean','cityblock','cosine','correlation'};
dist_size = length(distances);

for j = 1 : dist_size
    for i = 2 : 4
        dist = distances{j};

        % Create clusters and determine separation
        Y = kmeans(X,i,'Distance',dist);
        [silh,~] = silhouette(X,Y,dist);
        clust = mean(silh);

        % Count the number of samples inserted in each cluster
        V = f_verify_cluster(Y);

        % Update output data
        DIST_METRIC = [DIST_METRIC; distances(j)];
        CLUSTERS = [CLUSTERS; i];
        AVERAGE_SIL = [AVERAGE_SIL; clust];
        CCIR_CCIR_clust1 = [CCIR_CCIR_clust1; V(1,1)];
        CCIR_CCIR_clust2 = [CCIR_CCIR_clust2; V(1,2)];
        CCIR_CCIR_clust3 = [CCIR_CCIR_clust3; V(1,3)];
        CCIR_CCIR_clust4 = [CCIR_CCIR_clust4; V(1,4)];
        CCIR_NAB_clust1 = [CCIR_NAB_clust1; V(2,1)];
        CCIR_NAB_clust2 = [CCIR_NAB_clust2; V(2,2)];
        CCIR_NAB_clust3 = [CCIR_NAB_clust3; V(2,3)];
        CCIR_NAB_clust4 = [CCIR_NAB_clust4; V(2,4)];
        NAB_CCIR_clust1 = [NAB_CCIR_clust1; V(3,1)];
        NAB_CCIR_clust2 = [NAB_CCIR_clust2; V(3,2)];
        NAB_CCIR_clust3 = [NAB_CCIR_clust3; V(3,3)];
        NAB_CCIR_clust4 = [NAB_CCIR_clust4; V(3,4)];
        NAB_NAB_clust1 = [NAB_NAB_clust1; V(4,1)];
        NAB_NAB_clust2 = [NAB_NAB_clust2; V(4,2)];
        NAB_NAB_clust3 = [NAB_NAB_clust3; V(4,3)];
        NAB_NAB_clust4 = [NAB_NAB_clust4; V(4,4)];
    end
end

% Insert output data into a table
tab = table(CLUSTERS,DIST_METRIC,AVERAGE_SIL,CCIR_CCIR_clust1,CCIR_NAB_clust1,NAB_CCIR_clust1,
    NAB_NAB_clust1,CCIR_CCIR_clust2,CCIR_NAB_clust2,NAB_CCIR_clust2,
    NAB_NAB_clust2,CCIR_CCIR_clust3,CCIR_NAB_clust3,NAB_CCIR_clust3,
    NAB_NAB_clust3,CCIR_CCIR_clust4,CCIR_NAB_clust4,NAB_CCIR_clust4,
    NAB_NAB_clust4);

% Save table in .cvs file
table_name = strcat('../results/Cep_K-Means_Clustering',filename,'.csv');
writetable(tab, table_name,'Delimiter',',','QuoteStrings',true);

end
```

```
function [train_performance,test_performance]...
    = decision_tree_perform(X,X_output,T,T_output,MaxNumSplits)

% Train Decision Tree and evaluate performance
tic;
Mdl = fitctree(X,X_output,'MaxNumSplits',MaxNumSplits);
TRAINING_T = toc;

% Creates a partitioned model, http://it.mathworks.com/help/stats/classificationtree.crossval.html
cvmodel = crossval(Mdl,'KFold',10);

% Estimate Cross-Validated Classification Error
Er = kfoldLoss(cvmodel);

% Choose the best model
L = kfoldLoss(cvmodel,'mode','individual');
[~, ind] = min(L);
bestMdl = cvmodel.Trained{ind};

% Calculate confusion matrix and evaluate performance
[label1,score1,~] = predict(bestMdl,X);
conf_mat = confusionmat(X_output,label1);
train_performance = conf_mat_to_perform(conf_mat,X_output,score1);
train_performance.CV_CLASSIFIC_ERR = Er;
train_performance.TRAINING_TIME = TRAINING_T;

% Estimate classifier performance on new dataset
tic;
[label2,score2,~] = predict(bestMdl,T);
TEST_T = toc;
conf_mat2 = confusionmat(T_output,label2);
test_performance = conf_mat_to_perform(conf_mat2,T_output,score2);
test_performance.TEST_TIME = TEST_T;

end
```

```
function [train_performance,test_performance]...
    = KNN_perform(X,X_output,T,T_output,NumNeighbors, distance)

% Train KNN and evaluate performance
Mdl = [];
if (strcmp(distance,'cosine'))
    tic;
    Mdl = fitcknn(X,X_output,'NumNeighbors',NumNeighbors,...
    'NSMethod','exhaustive','Distance','cosine',...
    'Standardize',1);
    TRAINING_T = toc;
elseif (strcmp(distance,'weighting'))
    tic;
    Mdl = fitcknn(X,X_output,'NumNeighbors',NumNeighbors,...
    'DistanceWeight','inverse','Standardize',1);
    TRAINING_T = toc;
else
   tic;
   Mdl = fitcknn(X,X_output,'NumNeighbors',NumNeighbors,'Standardize',1);
   TRAINING_T = toc;
end

% Creates a partitioned model, https://it.mathworks.com/help/stats/classificationknn.crossval.html
cvmodel = crossval(Mdl,'KFold',10);

% Estimate Cross-Validated Classification Error
Er = kfoldLoss(cvmodel);

% Choose the best model
L = kfoldLoss(cvmodel,'mode','individual');
[~, ind] = min(L);
bestMdl = cvmodel.Trained{ind};

% Calculate confusion matrix and evaluate performance
[label1,score1,~] = predict(bestMdl,X);
conf_mat = confusionmat(X_output,label1);
train_performance = conf_mat_to_perform(conf_mat,X_output,score1);
train_performance.CV_CLASSIFIC_ERR = Er;
train_performance.TRAINING_TIME = TRAINING_T;

% Estimate classifier performance on new dataset
tic;
[label2,score2,~] = predict(bestMdl,T);
TEST_T = toc;
conf_mat2 = confusionmat(T_output,label2);
test_performance = conf_mat_to_perform(conf_mat2,T_output,score2);
test_performance.TEST_TIME = TEST_T;

end
```

```
function [train_performance,test_performance...
    = SVM_perform(X,X_output,T,T_output,KernelFunction)

% Train SVM and evaluate performance
templ = [];
if (strcmp(KernelFunction,'quadratic'))
    templ = templateSVM('Standardize',1,'KernelFunction','polynomial',...
        'PolynomialOrder',2);
elseif (strcmp(KernelFunction,'cubic'))
    templ = templateSVM('Standardize',1,'KernelFunction','polynomial',...
        'PolynomialOrder',3);
elseif (strcmp(KernelFunction,'gaussian'))
    templ = templateSVM('Standardize',1,'KernelFunction','gaussian');
elseif (strcmp(KernelFunction,'linear'))
    templ = templateSVM('Standardize',1,'KernelFunction','linear');
else
    error('Error. \nKernelFunction must be linear, quadratic, cubic or gaussian'
end

tic;
Mdl = fitcecoc(X,X_output,'Learners',templ,'FitPosterior',1);
TRAINING_T = toc;

% Creates a partitioned model, https://it.mathworks.com/help/stats/classificationsvm.crossval.html
cvmodel = crossval(Mdl,'KFold',10);

% Estimate Cross-Validated Classification Error
Er = kfoldLoss(cvmodel);

% Choose the best model
L = kfoldLoss(cvmodel,'mode','individual');
[~, ind] = min(L);
bestMdl = cvmodel.Trained{ind};

% Calculate confusion matrix and evaluate performance
[label1,~,~, Posterior1] = predict(bestMdl,X);
conf_mat = confusionmat(X_output,label1);
train_performance = conf_mat_to_perform(conf_mat,X_output,Posterior1);
train_performance.CV_CLASSIFIC_ERR = Er;
train_performance.TRAINING_TIME = TRAINING_T;

% Estimate classifier performance on new dataset
tic;
[label2,~,~, Posterior2] = predict(bestMdl,T);
TEST_T = toc;
conf_mat2 = confusionmat(T_output,label2);
test_performance = conf_mat_to_perform(conf_mat2,T_output,Posterior2);
test_performance.TEST_TIME = TEST_T;

end
```

Appendice B

Cluster analysis con spettro intero - risultati

Le tabelle illustrate nelle pagine successive mostrano i risultati della *cluster analysis* effettuata su ciascuno dei quattro *dataset*, al variare dei parametri degli algoritmi. Come *feature* sono state utilizzate le 48000 componenti dello spettro dei campioni audio.
La chiave di lettura della matrice *result* è inserita alla fine di ogni tabella. I risultati più significativi sono stati evidenziati.

Hierarchical clustering - 7.5 ips - silenzio

Range FFT [Hz:Hz]	Distance	Linkage	Max cluster	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	euclidean	average	4	0 298 2 0 0 298 2 0 1 297 0 2 0 298 2 0
[1:48001]	euclidean	centroid	4	0 0 2 298 0 0 2 298 1 1 0 298 0 0 2 298
[1:48001]	euclidean	complete	4	5 293 2 0 2 296 2 0 2 296 0 2 3 295 2 0
[1:48001]	euclidean	median	4	0 298 2 0 0 298 2 0 1 297 0 2 0 298 2 0
[1:48001]	euclidean	single	4	298 2 0 0 298 2 0 0 298 0 1 1 298 2 0 0
[1:48001]	euclidean	ward	4	5 293 2 0 2 0 2 296 2 296 2 0 3 0 2 295
[1:48001]	euclidean	weighted	4	5 293 2 0 2 296 2 0 2 296 0 2 3 295 2 0
[1:48001]	squaredeuclidean	average	4	5 293 2 0 2 296 2 0 2 296 0 2 3 295 2 0
[1:48001]	squaredeuclidean	complete	4	5 293 2 0 2 296 2 0 2 296 0 2 3 295 2 0
[1:48001]	squaredeuclidean	single	4	298 2 0 0 298 2 0 0 298 0 1 1 298 2 0 0
[1:48001]	squaredeuclidean	weighted	4	5 293 2 0 2 296 2 0 2 296 0 2 3 295 2 0
[1:48001]	seuclidean	average	4	0 299 1 0 0 300 0 0 1 298 0 1 0 300 0 0
[1:48001]	seuclidean	complete	4	5 294 1 0 0 300 0 0 2 296 0 2 0 300 0 0
[1:48001]	seuclidean	single	4	0 299 1 0 0 300 0 0 1 298 0 1 0 300 0 0

Range FFT [Hz:Hz]	Distance	Linkage	Max cluster	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	seuclidean	weighted	4	0 0 1 299 0 0 0 300 1 1 0 298 0 0 0 300
[1:48001]	cityblock	average	4	0 0 1 299 0 0 0 300 1 1 0 298 0 0 0 300
[1:48001]	cityblock	complete	4	1 298 1 0 300 0 0 0 0 298 0 2 300 0 0 0
[1:48001]	cityblock	single	4	0 299 1 0 0 300 0 0 1 298 0 1 0 300 0 0
[1:48001]	cityblock	weighted	4	0 0 1 299 0 0 0 300 1 1 0 298 0 0 0 300
[1:48001]	minkowski	average	4	0 298 2 0 0 298 2 0 1 297 0 2 0 298 2 0
[1:48001]	minkowski	complete	4	5 293 2 0 2 296 2 0 2 296 0 2 3 295 2 0
[1:48001]	minkowski	single	4	298 2 0 0 298 2 0 0 298 0 1 1 298 2 0 0
[1:48001]	minkowski	weighted	4	5 293 2 0 2 296 2 0 2 296 0 2 3 295 2 0
[1:48001]	chebychev	average	4	0 300 0 0 2 298 0 0 0 297 1 2 3 297 0 0
[1:48001]	chebychev	complete	4	0 6 294 0 0 2 298 0 1 3 294 2 0 5 295 0
[1:48001]	chebychev	single	4	0 0 0 300 0 0 0 300 1 1 1 297 0 0 0 300
[1:48001]	chebychev	weighted	4	0 300 0 0 2 298 0 0 0 297 1 2 3 297 0 0
[1:48001]	cosine	average	4	0 293 5 2 0 296 2 2 1 295 2 2 0 295 3 2

Range FFT [Hz:Hz]	Distance	Linkage	Max cluster	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	cosine	complete	4	0 293 5 2 0 296 2 2 1 295 2 2 0 295 3 2
[1:48001]	cosine	single	4	0 297 1 2 1 297 0 2 0 298 0 2 0 298 0 2
[1:48001]	cosine	weighted	4	5 293 0 2 2 296 0 2 2 290 6 2 3 295 0 2
[1:48001]	correlation	average	4	0 293 5 2 0 293 5 2 2 294 2 2 0 295 3 2
[1:48001]	correlation	complete	4	0 293 5 2 0 292 6 2 2 294 2 2 0 291 7 2
[1:48001]	correlation	single	4	1 296 1 2 0 298 0 2 0 298 0 2 0 298 0 2
[1:48001]	correlation	weighted	4	6 292 0 2 5 293 0 2 2 294 2 2 5 293 0 2
[1:48001]	spearman	average	4	1 297 1 1 0 300 0 0 0 300 0 0 0 300 0 0
[1:48001]	spearman	complete	4	17 182 5 96 0 300 0 0 3 252 0 45 0 300 0 0
[1:48001]	spearman	single	4	1 297 1 1 0 300 0 0 0 300 0 0 0 300 0 0
[1:48001]	spearman	weighted	4	11 282 5 2 0 300 0 0 0 300 0 0 0 300 0 0
[1:48001]	hamming	average	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	hamming	complete	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	hamming	single	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1

Range FFT [Hz:Hz]	Distance	Linkage	Max cluster	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	hamming	weighted	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	jaccard	average	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	jaccard	complete	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	jaccard	single	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	jaccard	weighted	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1

* Interpretazione matrice **result**: la cella (**i**,**j**) contiene il numero di campioni con equalizzazione **i** (*i = 1 —> cci_ccir; i = 2 —> cci_nab; i = 3 —> nab_ccir; i = 4 —> nab_nab*) inseriti nel cluster **j**. Quindi le righe rappresentano le tipologie di equalizzazione, le colonne rappresentano i cluster. Per esempio, se la riga 3 contiene i valori [2, 292, 6, 0], vuol dire che dei 300 campioni equalizzati nab_ccir, 2 sono stati inseriti nel cluster 1, 292 nel cluster 2, 6 nel cluster 3 e 0 nel cluster 4.

Hierarchical clustering - 7.5 ips - rumore

Range FFT [Hz:Hz]	Distance	Linkage	Max cluster	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	euclidean	average	4	0 300 0 0 0 300 0 0 2 294 2 2 0 300 0 0
[1:48001]	euclidean	centroid	4	0 300 0 0 0 300 0 0 1 297 1 1 0 300 0 0
[1:48001]	euclidean	complete	4	19 11 2 268 2 2 0 296 156 114 30 0 11 6 1 282
[1:48001]	euclidean	complete	2	268 32 0 0 296 4 0 0 0 300 0 0 282 18 0 0
[1:48001]	euclidean	median	4	0 300 0 0 0 300 0 0 1 297 1 1 0 300 0 0
[1:48001]	euclidean	single	4	0 300 0 0 0 300 0 0 1 297 1 1 0 300 0 0
[1:48001]	euclidean	ward	4	40 228 32 0 40 228 32 0 37 197 28 38 40 228 32 0
[1:48001]	euclidean	weighted	4	0 300 0 0 0 300 0 0 2 291 4 3 0 300 0 0
[1:48001]	squaredeuclidean	average	4	0 300 0 0 0 300 0 0 2 294 2 2 0 300 0 0
[1:48001]	squaredeuclidean	complete	4	19 11 2 268 2 2 0 296 156 114 30 0 11 6 1 282
[1:48001]	squaredeuclidean	complete	2	268 32 0 0 296 4 0 0 0 300 0 0 282 18 0 0
[1:48001]	squaredeuclidean	single	4	0 300 0 0 0 300 0 0 1 297 1 1 0 300 0 0
[1:48001]	squaredeuclidean	weighted	4	0 299 1 0 0 300 0 0 6 282 8 4 0 300 0 0
[1:48001]	seuclidean	average	4	0 0 0 300 0 0 0 300 1 2 25 272 0 0 0 300

Range FFT [Hz:Hz]	Distance	Linkage	Max cluster	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	seuclidean	complete	4	0 0 285 15 0 0 300 0 3 33 0 264 0 0 280 20
[1:48001]	seuclidean	complete	3	285 15 0 0 300 0 0 0 0 264 36 0 280 20 0 0
[1:48001]	seuclidean	single	4	0 300 0 0 0 300 0 0 1 297 1 1 0 300 0 0
[1:48001]	seuclidean	weighted	4	0 0 0 300 0 0 0 300 35 19 3 243 0 0 0 300
[1:48001]	cityblock	average	4	0 300 0 0 0 300 0 0 2 296 1 1 0 300 0 0
[1:48001]	cityblock	complete	4	5 9 24 262 0 0 0 300 32 94 174 0 4 5 18 273
[1:48001]	cityblock	complete	3	24 14 262 0 0 0 300 0 174 126 0 0 18 9 273 0
[1:48001]	cityblock	complete	2	262 38 0 0 300 0 0 0 0 300 0 0 273 27 0 0
[1:48001]	cityblock	single	4	0 300 0 0 0 300 0 0 1 297 1 1 0 300 0 0
[1:48001]	cityblock	weighted	4	0 300 0 0 0 300 0 0 9 288 1 2 0 300 0 0
[1:48001]	minkowski	average	4	0 300 0 0 0 300 0 0 2 294 2 2 0 300 0 0
[1:48001]	minkowski	complete	4	19 11 2 268 2 2 0 296 156 114 30 0 11 6 1 282
[1:48001]	minkowski	complete	3	2 30 268 0 0 4 296 0 30 270 0 0 1 17 282 0
[1:48001]	minkowski	complete	2	268 32 0 0 296 4 0 0 0 300 0 0 282 18 0 0

Range FFT [Hz:Hz]	Distance	Linkage	Max cluster	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	minkowski	single	4	0 300 0 0 0 300 0 0 1 297 1 1 0 300 0 0
[1:48001]	minkowski	weighted	4	0 300 0 0 0 300 0 0 2 291 4 3 0 300 0 0
[1:48001]	chebychev	average	4	0 300 0 0 0 300 0 0 2 291 4 3 0 300 0 0
[1:48001]	chebychev	complete	4	300 0 0 0 300 0 0 0 241 18 37 4 300 0 0 0
[1:48001]	chebychev	complete	3	0 300 0 0 0 300 0 0 37 259 4 0 0 300 0 0
[1:48001]	chebychev	single	4	0 300 0 0 0 300 0 0 1 297 1 1 0 300 0 0
[1:48001]	chebychev	weighted	4	0 300 0 0 0 300 0 0 56 236 2 6 0 300 0 0
[1:48001]	cosine	average	4	14 254 24 8 14 254 24 8 14 254 24 8 14 254 24 8
[1:48001]	cosine	complete	4	18 104 54 124 18 104 54 124 18 103 55 124 18 104 54 124
[1:48001]	cosine	single	4	2 294 2 2 2 294 2 2 2 294 2 2 2 294 2 2
[1:48001]	cosine	weighted	4	24 100 174 2 24 100 174 2 24 100 174 2 24 100 174 2
[1:48001]	correlation	average	4	14 32 246 8 14 32 246 8 14 32 246 8 14 32 246 8
[1:48001]	correlation	complete	4	36 86 86 92 36 86 86 92 36 86 86 92 36 86 86 92
[1:48001]	correlation	single	4	2 294 2 2 2 294 2 2 2 294 2 2 2 294 2 2

Range FFT [Hz:Hz]	Distance	Linkage	Max cluster	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	correlation	weighted	4	86 166 40 8 86 166 40 8 86 166 40 8 86 166 40 8
[1:48001]	spearman	average	4	0 300 0 0 0 300 0 0 1 297 1 1 0 300 0 0
[1:48001]	spearman	complete	4	1 40 4 255 1 53 4 242 1 33 4 262 1 42 4 253
[1:48001]	spearman	single	4	0 300 0 0 0 300 0 0 1 297 1 1 0 300 0 0
[1:48001]	spearman	weighted	4	4 296 0 0 4 296 0 0 4 293 1 2 4 296 0 0
[1:48001]	hamming	average	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	hamming	complete	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	hamming	single	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	hamming	weighted	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	jaccard	average	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	jaccard	complete	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	jaccard	single	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	jaccard	weighted	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1

* Interpretazione matrice **result**: la cella (**i**,**j**) contiene il numero di campioni con equalizzazione **i** (*i = 1 —> cci_ccir; i = 2 —> cci_nab; i = 3 —> nab_ccir; i = 4 —> nab_nab*) inseriti nel cluster **j**. Quindi le righe rappresentano le tipologie di equalizzazione, le colonne rappresentano i cluster. Per esempio, se la riga 3 contiene i valori [2, 292, 6, 0], vuol dire che dei 300 campioni equalizzati nab_ccir, 2 sono stati inseriti nel cluster 1, 292 nel cluster 2, 6 nel cluster 3 e 0 nel cluster 4.

Hierarchical clustering - 15 ips - silenzio

Range FFT [Hz:Hz]	Distance	Linkage	Max cluster	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	euclidean	average	4	296 3 1 0 294 4 2 0 295 3 0 2 295 2 3 0
[1:48001]	euclidean	centroid	4	4 295 1 0 4 294 2 0 3 295 0 2 2 295 3 0
[1:48001]	euclidean	complete	4	1 298 1 0 2 296 2 0 2 296 0 2 2 295 3 0
[1:48001]	euclidean	median	4	295 4 1 0 294 4 2 0 295 3 0 2 295 2 3 0
[1:48001]	euclidean	single	4	1 0 299 0 0 2 298 0 0 0 298 2 3 0 297 0
[1:48001]	euclidean	ward	4	5 294 0 1 6 0 292 2 6 292 0 2 6 0 291 3
[1:48001]	euclidean	weighted	4	291 8 1 0 292 6 2 0 292 6 0 2 291 6 3 0
[1:48001]	squaredeuclidean	average	4	296 3 1 0 294 4 2 0 295 3 0 2 295 2 3 0
[1:48001]	squaredeuclidean	complete	4	1 298 1 0 2 296 2 0 2 296 0 2 2 295 3 0
[1:48001]	squaredeuclidean	single	4	1 0 299 0 0 2 298 0 0 0 298 2 3 0 297 0
[1:48001]	squaredeuclidean	weighted	4	291 8 1 0 292 6 2 0 292 6 0 2 291 6 3 0
[1:48001]	seuclidean	average	4	0 0 0 300 0 1 1 298 0 0 0 300 1 2 0 297
[1:48001]	seuclidean	complete	4	0 0 300 0 5 293 0 2 0 0 300 0 3 294 0 3
[1:48001]	seuclidean	single	4	0 300 0 0 1 298 0 1 0 300 0 0 0 299 1 0

Range FFT [Hz:Hz]	Distance	Linkage	Max cluster	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	seuclidean	weighted	4	0 300 0 0 1 1 296 2 0 300 0 0 0 3 296 1
[1:48001]	cityblock	average	4	0 1 0 299 1 0 1 298 0 2 0 298 0 3 0 297
[1:48001]	cityblock	complete	4	0 0 1 299 0 2 0 298 1 0 2 297 0 0 3 297
[1:48001]	cityblock	single	4	0 300 0 0 0 298 1 1 0 300 0 0 1 299 0 0
[1:48001]	cityblock	weighted	4	0 1 0 299 1 1 0 298 0 2 1 297 0 3 0 297
[1:48001]	minkowski	average	4	296 3 1 0 294 4 2 0 295 3 0 2 295 2 3 0
[1:48001]	minkowski	complete	4	1 298 1 0 2 296 2 0 2 296 0 2 2 295 3 0
[1:48001]	minkowski	single	4	1 0 299 0 0 2 298 0 0 0 298 2 3 0 297 0
[1:48001]	minkowski	weighted	4	291 8 1 0 292 6 2 0 292 6 0 2 291 6 3 0
[1:48001]	chebychev	average	4	1 0 299 0 2 2 296 0 2 1 295 2 2 0 298 0
[1:48001]	chebychev	complete	4	153 146 1 0 0 296 4 0 209 86 3 2 0 298 2 0
[1:48001]	chebychev	complete	3	1 299 0 0 4 296 0 0 3 295 2 0 2 298 0 0
[1:48001]	chebychev	single	4	1 299 0 0 0 299 1 0 1 297 0 2 0 300 0 0
[1:48001]	chebychev	weighted	4	299 0 1 0 296 2 2 0 295 1 2 2 298 0 2 0

Range FFT [Hz:Hz]	Distance	Linkage	Max cluster	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	cosine	average	4	0 1 8 291 0 2 6 292 1 2 6 291 0 3 6 291
[1:48001]	cosine	complete	4	0 292 7 1 292 0 6 2 0 292 6 2 291 0 6 3
[1:48001]	cosine	single	4	0 299 1 0 0 298 2 0 1 297 2 0 0 296 3 1
[1:48001]	cosine	weighted	4	0 291 8 1 0 292 6 2 1 291 6 2 0 291 6 3
[1:48001]	correlation	average	4	7 292 0 1 8 290 0 2 6 291 0 3 8 288 1 3
[1:48001]	correlation	complete	4	0 294 1 5 0 292 2 6 0 293 3 4 1 290 3 6
[1:48001]	correlation	single	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	correlation	weighted	4	5 294 0 1 8 290 0 2 6 291 0 3 7 289 1 3
[1:48001]	spearman	average	4	0 300 0 0 0 299 0 1 0 300 0 0 1 298 1 0
[1:48001]	spearman	complete	4	0 300 0 0 1 122 58 119 0 300 0 0 4 105 63 128
[1:48001]	spearman	complete	2	300 0 0 0 123 177 0 0 300 0 0 0 109 191 0 0
[1:48001]	spearman	single	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	spearman	weighted	4	0 0 300 0 0 0 299 1 0 0 300 0 2 2 295 1
[1:48001]	hamming	average	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1

Range FFT [Hz:Hz]	Distance	Linkage	Max cluster	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	hamming	complete	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	hamming	single	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	hamming	weighted	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	jaccard	average	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	jaccard	complete	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	jaccard	single	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	jaccard	weighted	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1

* Interpretazione matrice **result**: la cella (**i**,**j**) contiene il numero di campioni con equalizzazione **i** (*i = 1 —> cci_ccir; i = 2 —> cci_nab; i = 3 —> nab_ccir; i = 4 —> nab_nab*) inseriti nel cluster **j**. Quindi le righe rappresentano le tipologie di equalizzazione, le colonne rappresentano i cluster. Per esempio, se la riga 3 contiene i valori [2, 292, 6, 0], vuol dire che dei 300 campioni equalizzati nab_ccir, 2 sono stati inseriti nel cluster 1, 292 nel cluster 2, 6 nel cluster 3 e 0 nel cluster 4.

Hierarchical clustering - 15 ips - rumore

Range FFT [Hz:Hz]	Distance	Linkage	Max cluster	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	euclidean	average	4	0 300 0 0 3 295 1 1 0 300 0 0 0 300 0 0
[1:48001]	euclidean	centroid	4	0 300 0 0 1 297 1 1 0 300 0 0 0 300 0 0
[1:48001]	euclidean	complete	4	287 4 6 3 66 112 76 46 300 0 0 0 283 9 6 2
[1:48001]	euclidean	median	4	0 300 0 0 1 297 1 1 0 300 0 0 0 300 0 0
[1:48001]	euclidean	single	4	0 300 0 0 1 297 1 1 0 300 0 0 0 300 0 0
[1:48001]	euclidean	ward	4	24 44 32 200 24 44 32 200 20 44 32 204 22 44 32 202
[1:48001]	euclidean	weighted	4	0 300 0 0 34 252 6 8 0 300 0 0 0 300 0 0
[1:48001]	squaredeuclidean	average	4	0 300 0 0 2 294 3 1 0 300 0 0 0 300 0 0
[1:48001]	squaredeuclidean	complete	4	287 4 6 3 66 112 76 46 300 0 0 0 283 9 6 2
[1:48001]	squaredeuclidean	single	4	0 300 0 0 1 297 1 1 0 300 0 0 0 300 0 0
[1:48001]	squaredeuclidean	weighted	4	0 300 0 0 24 256 14 6 0 300 0 0 0 300 0 0
[1:48001]	seuclidean	average	4	0 0 0 300 1 2 1 296 0 0 0 300 0 0 0 300
[1:48001]	seuclidean	complete	4	20 0 0 280 271 22 7 0 0 0 0 300 10 0 0 290
[1:48001]	seuclidean	complete	2	280 20 0 0 0 300 0 0 300 0 0 0 290 10 0 0

Range FFT [Hz:Hz]	Distance	Linkage	Max cluster	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	seuclidean	single	4	0 300 0 0 1 297 1 1 0 300 0 0 0 300 0 0
[1:48001]	seuclidean	weighted	4	300 0 0 0 294 3 1 2 300 0 0 0 300 0 0 0
[1:48001]	cityblock	average	4	0 300 0 0 3 294 2 1 0 300 0 0 0 300 0 0
[1:48001]	cityblock	complete	4	3 291 3 3 34 180 26 60 0 300 0 0 3 290 4 3
[1:48001]	cityblock	single	4	0 300 0 0 1 297 1 1 0 300 0 0 0 300 0 0
[1:48001]	cityblock	weighted	4	0 300 0 0 20 268 8 4 0 300 0 0 0 300 0 0
[1:48001]	minkowski	average	4	0 300 0 0 3 295 1 1 0 300 0 0 0 300 0 0
[1:48001]	minkowski	complete	4	287 4 6 3 66 112 76 46 300 0 0 0 283 9 6 2
[1:48001]	minkowski	single	4	0 300 0 0 1 297 1 1 0 300 0 0 0 300 0 0
[1:48001]	minkowski	weighted	4	0 300 0 0 34 252 6 8 0 300 0 0 0 300 0 0
[1:48001]	chebychev	average	4	300 0 0 0 300 0 0 0 272 20 2 6 300 0 0 0
[1:48001]	chebychev	complete	4	0 0 300 0 0 0 300 0 24 40 224 12 0 0 300 0
[1:48001]	chebychev	single	4	0 300 0 0 1 297 1 1 0 300 0 0 0 300 0 0
[1:48001]	chebychev	weighted	4	0 300 0 0 0 300 0 0 2 290 6 2 0 300 0 0

Range FFT [Hz:Hz]	Distance	Linkage	Max cluster	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	cosine	average	4	72 120 86 22 72 120 86 22 72 120 86 22 72 120 86 22
[1:48001]	cosine	complete	4	118 122 40 20 118 122 40 20 118 122 40 20 118 122 40 20
[1:48001]	cosine	single	4	2 294 2 2 2 294 2 2 2 294 2 2 2 294 2 2
[1:48001]	cosine	weighted	4	6 20 266 8 6 20 266 8 6 20 266 8 6 20 266 8
[1:48001]	correlation	average	4	14 26 244 16 14 26 244 16 14 26 244 16 14 26 244 16
[1:48001]	correlation	complete	4	50 82 116 52 50 82 116 52 50 82 116 52 50 82 116 52
[1:48001]	correlation	single	4	2 294 2 2 2 294 2 2 2 294 2 2 2 294 2 2
[1:48001]	correlation	weighted	4	36 122 134 8 36 122 134 8 36 122 134 8 36 122 134 8
[1:48001]	spearman	average	4	4 288 4 4 4 288 4 4 4 288 4 4 4 288 4 4
[1:48001]	spearman	complete	4	4 266 26 4 4 266 26 4 4 266 26 4 4 266 26 4
[1:48001]	spearman	single	4	2 294 2 2 2 294 2 2 2 294 2 2 2 294 2 2
[1:48001]	spearman	weighted	4	8 282 4 6 8 282 4 6 8 282 4 6 8 282 4 6
[1:48001]	hamming	average	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	hamming	complete	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1

Range FFT [Hz:Hz]	Distance	Linkage	Max cluster	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	hamming	single	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	hamming	weighted	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	jaccard	average	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	jaccard	complete	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	jaccard	single	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1
[1:48001]	jaccard	weighted	4	0 300 0 0 0 300 0 0 0 300 0 0 1 297 1 1

* Interpretazione matrice **result**: la cella (**i**,**j**) contiene il numero di campioni con equalizzazione **i** (*i = 1 —> cci_ccir; i = 2 —> cci_nab; i = 3 —> nab_ccir; i = 4 —> nab_nab*) inseriti nel cluster **j**. Quindi le righe rappresentano le tipologie di equalizzazione, le colonne rappresentano i cluster. Per esempio, se la riga 3 contiene i valori [2, 292, 6, 0], vuol dire che dei 300 campioni equalizzati nab_ccir, 2 sono stati inseriti nel cluster 1, 292 nel cluster 2, 6 nel cluster 3 e 0 nel cluster 4.

k-means clustering - 7.5 ips - silenzio

Range FFT [Hz:Hz]	# of cluster	Distance	Average silhouette values [0,1]**	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	4	sqeuclidean	0.2069	2 298 0 0 2 0 298 0 0 298 0 2 2 0 298 0
[1:48001]	3	sqeuclidean	0.2033	299 1 0 0 0 2 298 0 298 2 0 0 0 2 298 0
[1:48001]	2	sqeuclidean	0.9069	299 1 0 0 298 2 0 0 298 2 0 0 298 2 0 0
[1:48001]	4	cityblock	0.1061	0 0 0 300 0 300 0 0 1 0 1 298 0 300 0 0
[1:48001]	3	cityblock	0.1041	296 0 4 0 0 300 0 0 289 0 11 0 0 300 0 0
[1:48001]	2	cityblock	0.0414	1 299 0 0 0 300 0 0 0 300 0 0 0 300 0 0
[1:48001]	4	cosine	0.0339	177 115 1 7 0 1 292 7 190 107 0 3 0 2 291 7
[1:48001]	3	cosine	0.0595	7 0 293 0 8 291 1 0 3 0 297 0 7 291 2 0
[1:48001]	2	cosine	0.4582	293 7 0 0 292 8 0 0 296 4 0 0 293 7 0 0
[1:48001]	4	correlation	-0.0099	7 291 2 0 297 0 2 1 2 296 2 0 298 0 2 0
[1:48001]	3	correlation	0.2963	8 290 2 0 10 288 2 0 6 292 2 0 10 288 2 0
[1:48001]	2	correlation	0.0481	6 294 0 0 298 2 0 0 2 298 0 0 298 2 0 0

* Interpretazione matrice **result**: la cella (**i**,**j**) contiene il numero di campioni con equalizzazione **i** (*i = 1 —> cci_ccir; i = 2 —> cci_nab; i = 3 —> nab_ccir; i = 4 —> nab_nab*) inseriti nel cluster **j**. Quindi le righe rappresentano le tipologie di equalizzazione, le colonne rappresentano i cluster. Per esempio, se la riga 3 contiene i valori [2, 292, 6, 0], vuol dire che dei 300 campioni equalizzati nab_ccir, 2 sono stati inseriti nel cluster 1, 292 nel cluster 2, 6 nel cluster 3 e 0 nel cluster 4.

** *Average silhouette value (ASV)* rappresenta una misura quantitativa utile per confrontare i risultati ottenuti utilizzando la stessa tipologia di distanza ma diverso numero di cluster. Idealmente, a parità di tipo di distanza calcolata, maggiore è l'ASV, maggiore è la probabilità che il numero di cluster scelto sia quello giusto.

k-means clustering - 7.5 ips - rumore

Range FFT [Hz:Hz]	# of cluster	Distance	Average silhouette values [0,1]**	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	4	sqeuclidean	0.1690	1 1 0 298 0 0 0 300 2 3 295 0 1 1 0 298
[1:48001]	3	sqeuclidean	0.32	0 0 300 0 300 0 0 0 0 300 0 0 0 0 300 0
[1:48001]	2	sqeuclidean	0.2593	0 300 0 0 0 300 0 0 300 0 0 0 0 300 0 0
[1:48001]	4	cityblock	0.1224	0 298 1 1 0 300 0 0 293 0 4 3 0 299 1 0
[1:48001]	3	cityblock	0.0972	299 0 1 0 300 0 0 0 0 298 2 0 299 0 1 0
[1:48001]	2	cityblock	0.1688	300 0 0 0 300 0 0 0 0 300 0 0 300 0 0 0
[1:48001]	4	cosine	0.012	0 187 15 98 285 0 15 0 0 194 2 104 0 191 15 94
[1:48001]	3	cosine	0.0279	300 0 0 0 0 0 300 0 0 300 0 0 300 0 0 0
[1:48001]	2	cosine	0.0465	300 0 0 0 0 300 0 0 300 0 0 0 300 0 0 0
[1:48001]	4	correlation	0.0164	0 0 288 12 0 289 0 11 293 0 0 7 0 0 288 12
[1:48001]	3	correlation	0.0484	4 0 296 0 0 0 300 0 64 236 0 0 5 0 295 0
[1:48001]	2	correlation	0.0597	300 0 0 0 300 0 0 0 0 300 0 0 300 0 0 0

* Interpretazione matrice **result**: la cella (**i**,**j**) contiene il numero di campioni con equalizzazione **i** (*i = 1 —> cci_ccir; i = 2 —> cci_nab; i = 3 —> nab_ccir; i = 4 —> nab_nab*) inseriti nel cluster **j**. Quindi le righe rappresentano le tipologie di equalizzazione, le colonne rappresentano i cluster. Per esempio, se la riga 3 contiene i valori [2, 292, 6, 0], vuol dire che dei 300 campioni equalizzati nab_ccir, 2 sono stati inseriti nel cluster 1, 292 nel cluster 2, 6 nel cluster 3 e 0 nel cluster 4.

** *Average silhouette value (ASV)* rappresenta una misura quantitativa utile per confrontare i risultati ottenuti utilizzando la stessa tipologia di distanza ma diverso numero di cluster. Idealmente, a parità di tipo di distanza calcolata, maggiore è l'ASV, maggiore è la probabilità che il numero di cluster scelto sia quello giusto.

k-means clustering - 15 ips - silenzio

Range FFT [Hz:Hz]	# of cluster	Distance	Average silhouette values [0,1]**	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab] *
[1:48001]	4	sqeuclidean	-0.0168	0 0 109 191 297 1 2 0 1 0 176 123 297 0 3 0
[1:48001]	3	sqeuclidean	0.1943	295 0 5 0 2 292 6 0 294 0 6 0 3 291 6 0
[1:48001]	2	sqeuclidean	0.9097	299 1 0 0 298 2 0 0 298 2 0 0 297 3 0 0
[1:48001]	4	cityblock	0.0519	300 0 0 0 0 1 298 1 300 0 0 0 0 0 300 0
[1:48001]	3	cityblock	0.0327	300 0 0 0 299 0 1 0 300 0 0 0 299 1 0 0
[1:48001]	2	cityblock	0.0555	300 0 0 0 0 300 0 0 300 0 0 0 0 300 0 0
[1:48001]	4	cosine	0.1535	9 0 291 0 8 86 0 206 9 0 291 0 9 132 0 159
[1:48001]	3	cosine	0.2019	9 0 291 0 8 292 0 0 9 0 291 0 9 291 0 0
[1:48001]	2	cosine	0.1979	300 0 0 0 2 298 0 0 300 0 0 0 3 297 0 0
[1:48001]	4	correlation	0.2140	291 0 1 8 289 0 2 9 291 0 3 6 289 1 3 7
[1:48001]	3	correlation	0.0722	1 0 299 0 2 1 297 0 3 0 297 0 3 0 297 0
[1:48001]	2	correlation	0.0573	0 300 0 0 0 300 0 0 0 300 0 0 1 299 0 0

* Interpretazione matrice **result**: la cella (**i**,**j**) contiene il numero di campioni con equalizzazione **i** (*i = 1 —> cci_ccir; i = 2 —> cci_nab; i = 3 —> nab_ccir; i = 4 —> nab_nab*) inseriti nel cluster **j**. Quindi le righe rappresentano le tipologie di equalizzazione, le colonne rappresentano i cluster. Per esempio, se la riga 3 contiene i valori [2, 292, 6, 0], vuol dire che dei 300 campioni equalizzati nab_ccir, 2 sono stati inseriti nel cluster 1, 292 nel cluster 2, 6 nel cluster 3 e 0 nel cluster 4.

** *Average silhouette value (ASV)* rappresenta una misura quantitativa utile per confrontare i risultati ottenuti utilizzando la stessa tipologia di distanza ma diverso numero di cluster. Idealmente, a parità di tipo di distanza calcolata, maggiore è l'ASV, maggiore è la probabilità che il numero di cluster scelto sia quello giusto.

k-means clustering - 15 ips - rumore

Range FFT [Hz:Hz]	# of cluster	Distance	Average silhouette values [0,1]**	Result [ccir_ccir; ccir_nab; nab_ccir; nab_nab]*
[1:48001]	4	sqeuclidean	-5.4667e-04	293 7 0 0 0 17 283 0 0 0 0 300 292 8 0 0
[1:48001]	3	sqeuclidean	0.0827	0 4 296 0 296 4 0 0 0 3 297 0 0 4 296 0
[1:48001]	2	sqeuclidean	0.0606	2 298 0 0 3 297 0 0 1 299 0 0 2 298 0 0
[1:48001]	4	cityblock	-0.0029	0 300 0 0 205 0 0 95 0 0 300 0 0 300 0 0
[1:48001]	3	cityblock	0.0405	0 296 4 0 297 0 3 0 0 298 2 0 0 297 3 0
[1:48001]	2	cityblock	0.1336	300 0 0 0 0 300 0 0 300 0 0 0 300 0 0 0
[1:48001]	4	cosine	0.0171	0 116 110 74 0 118 104 78 300 0 0 0 0 116 109 75
[1:48001]	3	cosine	00191	300 0 0 0 0 300 0 0 0 0 300 0 300 0 0 0
[1:48001]	2	cosine	0.0416	0 300 0 0 0 300 0 0 300 0 0 0 0 300 0 0
[1:48001]	4	correlation	0.0203	23 16 261 0 24 16 260 0 3 11 0 286 23 16 261 0
[1:48001]	3	correlation	0.0203	0 0 300 0 300 0 0 0 0 300 0 0 0 0 300 0
[1:48001]	2	correlation	0.0312	0 300 0 0 300 0 0 0 0 300 0 0 0 300 0 0

* Interpretazione matrice **result**: la cella (**i**,**j**) contiene il numero di campioni con equalizzazione **i** (*i = 1 —> cci_ccir; i = 2 —> cci_nab; i = 3 —> nab_ccir; i = 4 —> nab_nab*) inseriti nel cluster **j**. Quindi le righe rappresentano le tipologie di equalizzazione, le colonne rappresentano i cluster. Per esempio, se la riga 3 contiene i valori [2, 292, 6, 0], vuol dire che dei 300 campioni equalizzati nab_ccir, 2 sono stati inseriti nel cluster 1, 292 nel cluster 2, 6 nel cluster 3 e 0 nel cluster 4.

** *Average silhouette value (ASV)* rappresenta una misura quantitativa utile per confrontare i risultati ottenuti utilizzando la stessa tipologia di distanza ma diverso numero di cluster. Idealmente, a parità di tipo di distanza calcolata, maggiore è l'ASV, maggiore è la probabilità che il numero di cluster scelto sia quello giusto.

Appendice C

Cluster analysis con coefficienti cepstrali - risultati

Le tabelle illustrate nelle pagine successive mostrano i risultati della *cluster analysis* effettuata su ciascun *dataset*, al variare dei parametri degli algoritmi. Come *feature* sono stati utilizzati i 13 coefficienti cepstrali di ciascun campione audio.
Le colonne con nome del tipo $preEq\_postEq\_clust_i$ rappresentano il numero di campioni con pre-equalizzazione *preEq* (NAB o CCIR) e post-equalizzazione *postEq* (NAB o CCIR) inseriti nel *cluster* i ($i = 1, 2, 3, 4$), al variare dei parametri degli algoritmi (righe).
Per facilitare la lettura delle tabelle, le colonne relative allo stesso *cluster* sono state colorate allo stesso modo e le righe con risultati significativi sono state evidenziate.

HIERARCHICAL CLUSTERING con COEFFICIENTI CEPSTRALI. DATASET 7.5 IPS - SILENZIO																				
MAX_CLUST	DIST_METRIC	LINK_METHOD	CCIR_CCIR_clust1	CCIR_NAB_clust1	NAB_CCIR_clust1	NAB_NAB_clust1	CCIR_CCIR_clust2	CCIR_NAB_clust2	NAB_CCIR_clust2	NAB_NAB_clust2	CCIR_CCIR_clust3	CCIR_NAB_clust3	NAB_CCIR_clust3	NAB_NAB_clust3	CCIR_CCIR_clust4	CCIR_NAB_clust4	NAB_CCIR_clust4	NAB_NAB_clust4	COPHENET
4	euclidean	average	4	2	5	4	296	298	292	293	0	0	1	0	0	0	2	3	0,9167
4	euclidean	centroid	4	1	5	4	296	299	292	293	0	0	1	0	0	0	2	3	0,9156
4	euclidean	complete	4	2	5	4	296	298	292	293	0	0	1	0	0	0	2	3	0,8145
4	euclidean	median	7	2	6	4	293	298	291	293	0	0	2	3	0	0	1	0	0,8037
4	euclidean	single	0	0	0	1	300	300	297	297	0	0	2	2	0	0	1	0	0,7088
4	euclidean	ward	0	0	2	2	7	7	6	11	2	292	1	283	291	1	291	4	0,5054
4	euclidean	weighted	7	6	7	10	293	294	290	287	0	0	2	3	0	0	1	0	0,8023
4	squaredeuclidean	average	4	2	5	4	296	298	292	293	0	0	1	0	0	0	2	3	0,8023
4	squaredeuclidean	complete	4	2	5	4	296	298	292	293	0	0	1	0	0	0	2	3	0,8023
4	squaredeuclidean	single	0	0	0	1	300	300	297	297	0	0	2	2	0	0	1	0	0,8023
4	squaredeuclidean	weighted	291	292	288	287	9	8	9	10	0	0	1	0	0	0	2	3	0,8023
4	seuclidean	average	0	0	2	2	2	1	5	5	298	299	292	293	0	0	1	0	0,8023
4	seuclidean	complete	0	0	2	2	3	2	5	5	0	0	1	0	297	298	292	293	0,8023
4	seuclidean	single	0	0	0	1	300	300	297	297	0	0	2	2	0	0	1	0	0,8023
4	seuclidean	weighted	0	0	2	2	3	2	5	5	0	0	1	0	297	298	292	293	0,8023
4	cityblock	average	2	2	5	4	0	0	2	3	298	298	292	293	0	0	1	0	0,8023
4	cityblock	complete	293	294	291	287	7	6	6	10	0	0	1	0	0	0	2	3	0,8023
4	cityblock	single	0	0	0	1	300	300	297	297	0	0	2	2	0	0	1	0	0,8023
4	cityblock	weighted	293	293	291	287	7	7	6	10	0	0	1	0	0	0	2	3	0,8023
4	minkowski	average	4	2	5	4	296	298	292	293	0	0	1	0	0	0	2	3	0,8023
4	minkowski	complete	4	2	5	4	296	298	292	293	0	0	1	0	0	0	2	3	0,8023
4	minkowski	single	0	0	0	1	300	300	297	297	0	0	2	2	0	0	1	0	0,8023
4	minkowski	weighted	7	6	7	10	293	294	290	287	0	0	2	3	0	0	1	0	0,8023
4	chebychev	average	0	2	0	5	300	298	297	293	0	0	1	0	0	0	2	2	0,8023
4	chebychev	complete	0	0	1	0	292	0	291	2	8	300	6	295	0	0	2	3	0,8023
4	chebychev	single	0	0	0	2	0	0	2	0	300	300	297	298	0	0	1	0	0,8023
4	chebychev	weighted	2	299	0	292	298	0	297	1	0	0	1	0	0	1	2	7	0,8023
4	cosine	average	0	0	5	0	296	300	287	293	4	0	7	7	0	0	1	0	0,8023
4	cosine	complete	4	6	2	5	292	292	285	287	0	0	6	0	4	2	7	8	0,8023
4	cosine	single	0	0	1	0	300	300	297	300	0	0	1	0	0	0	1	0	0,8023
4	cosine	weighted	4	6	2	6	292	294	289	287	4	0	7	7	0	0	2	0	0,8023
4	correlation	average	0	0	1	0	291	293	288	287	5	7	4	6	4	0	7	7	0,8023
4	correlation	complete	3	6	2	6	5	0	7	7	0	0	2	0	292	294	289	287	0,8023
4	correlation	single	0	0	1	0	300	300	296	300	0	0	2	0	0	0	1	0	0,8023
4	correlation	weighted	156	53	176	84	137	245	114	206	0	0	2	0	7	2	8	10	0,8023
4	spearman	average	19	17	10	16	2	3	0	0	279	278	290	283	0	2	0	1	0,8023
4	spearman	complete	3	24	2	3	140	109	125	117	81	82	114	112	76	85	59	68	0,8023
4	spearman	single	0	1	0	0	300	298	299	300	0	0	1	0	0	1	0	0	0,8023
4	spearman	weighted	128	147	122	132	172	145	177	167	0	6	0	0	0	2	1	1	0,8023
4	hamming	average	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,8023
4	hamming	complete	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,8023
4	hamming	single	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,8023
4	hamming	weighted	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,8023
4	jaccard	average	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,8023
4	jaccard	complete	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,8023
4	jaccard	single	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,8023
4	jaccard	weighted	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,8023

HIEARCHICAL CLUSTERING con COEFFICIENTI CEPSTRALI. DATASET 7.5 IPS - RUMORE BIANCO																			
MAX_CLUSTER	DIST_METRIC	LINK_METHOD	CCIR_CCIR_clust1	CCIR_NAB_clust1	NAB_CCIR_clust1	NAB_NAB_clust1	CCIR_CCIR_clust2	CCIR_NAB_clust2	NAB_CCIR_clust2	NAB_NAB_clust2	CCIR_CCIR_clust3	CCIR_NAB_clust3	NAB_CCIR_clust3	NAB_NAB_clust3	CCIR_CCIR_clust4	CCIR_NAB_clust4	NAB_CCIR_clust4	NAB_NAB_clust4	COPHENET
4	euclidean	average	0	2	0	0	0	298	0	0	300	0	0	300	0	0	300	0	0,7689
4	euclidean	centroid	0	0	2	0	0	0	298	0	0	300	0	0	300	0	0	300	0,7688
4	euclidean	complete	132	0	0	110	168	0	0	190	0	300	0	0	0	0	300	0	0,7584
4	euclidean	median	0	0	282	0	0	0	18	0	300	0	0	300	0	300	0	0	0,7602
4	euclidean	single	0	0	2	0	0	0	298	0	300	0	0	300	0	300	0	0	0,7219
4	euclidean	ward	166	0	0	168	134	0	0	132	0	300	0	0	0	0	300	0	0,7650
4	euclidean	weighted	0	0	2	0	0	0	298	0	300	0	0	300	0	300	0	0	0,7627
4	squaredeuclidean	average	0	0	2	0	0	0	298	0	300	0	0	300	0	300	0	0	0,7627
4	squaredeuclidean	complete	132	0	0	110	168	0	0	190	0	300	0	0	0	0	300	0	0,7627
4	squaredeuclidean	single	0	0	2	0	0	0	298	0	300	0	0	300	0	300	0	0	0,7627
4	squaredeuclidean	weighted	12	0	0	12	288	0	0	288	0	0	300	0	0	300	0	0	0,7627
4	seuclidean	average	6	6	6	6	268	268	268	268	22	22	22	22	4	4	4	4	0,7627
4	seuclidean	complete	54	16	110	54	92	92	92	92	52	56	48	52	102	136	50	102	0,7627
4	seuclidean	single	2	2	2	2	294	294	294	294	2	2	2	2	2	2	2	2	0,7627
4	seuclidean	weighted	224	224	224	224	28	28	28	28	2	2	2	2	46	46	46	46	0,7627
4	cityblock	average	4	4	2	4	296	36	16	296	0	260	0	0	0	0	282	0	0,7627
4	cityblock	complete	0	0	288	0	206	0	12	206	78	284	0	78	16	16	0	16	0,7627
4	cityblock	single	0	0	2	0	298	298	294	298	0	0	2	0	2	2	2	2	0,7627
4	cityblock	weighted	22	276	0	22	16	22	0	16	2	2	2	2	260	0	298	260	0,7627
4	minkowski	average	0	2	0	0	0	298	0	0	300	0	0	300	0	0	300	0	0,7627
4	minkowski	complete	132	0	0	110	168	0	0	190	0	300	0	0	0	0	300	0	0,7627
4	minkowski	single	0	0	2	0	0	0	298	0	300	0	0	300	0	300	0	0	0,7627
4	minkowski	weighted	0	0	2	0	0	0	298	0	300	0	0	300	0	300	0	0	0,7627
4	chebychev	average	4	0	0	0	296	0	0	300	0	300	0	0	0	0	300	0	0,7627
4	chebychev	complete	70	0	0	20	230	0	0	280	0	300	0	0	0	0	300	0	0,7627
4	chebychev	single	0	2	0	0	0	298	0	0	300	0	0	300	0	0	300	0	0,7627
4	chebychev	weighted	8	0	0	8	292	0	0	292	0	300	0	0	0	0	300	0	0,7627
4	cosine	average	0	0	34	0	300	300	256	300	0	0	6	0	0	0	4	0	0,7627
4	cosine	complete	0	0	62	0	0	0	66	0	0	0	72	0	300	300	100	300	0,7627
4	cosine	single	0	0	2	0	300	300	294	300	0	0	2	0	0	0	2	0	0,7627
4	cosine	weighted	0	0	4	0	300	300	72	300	0	0	122	0	0	0	102	0	0,7627
4	correlation	average	0	0	4	0	300	300	288	300	0	0	6	0	0	0	2	0	0,7627
4	correlation	complete	0	0	34	0	0	0	30	0	0	0	34	0	300	300	202	300	0,7627
4	correlation	single	0	0	2	0	300	300	294	300	0	0	2	0	0	0	2	0	0,7627
4	correlation	weighted	0	0	28	0	0	0	56	0	300	300	162	300	0	0	54	0	0,7627
4	spearman	average	82	82	82	82	108	108	108	108	86	84	84	86	24	26	26	24	0,7627
4	spearman	complete	39	22	56	32	143	156	126	148	22	20	30	22	96	102	88	98	0,7627
4	spearman	single	2	2	2	2	292	292	290	292	6	6	6	6	0	0	2	0	0,7627
4	spearman	weighted	26	26	26	26	88	88	92	88	26	26	26	26	160	160	156	160	0,7627
4	hamming	average	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,7627
4	hamming	complete	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,7627
4	hamming	single	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,7627
4	hamming	weighted	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,7627
4	jaccard	average	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,7627
4	jaccard	complete	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,7627
4	jaccard	single	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,7627
4	jaccard	weighted	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,7627

HIEARCHICAL CLUSTERING con COEFFICIENTI CEPSTRALI. DATASET 15 IPS - SILENZIO																				
MAX_CLUSTER	DIST_METRIC	LINK_METHOD	CCIR_CCIR_clust1	CCIR_NAB_clust1	NAB_CCIR_clust1	NAB_NAB_clust1	CCIR_CCIR_clust2	CCIR_NAB_clust2	NAB_CCIR_clust2	NAB_NAB_clust2	CCIR_CCIR_clust3	CCIR_NAB_clust3	NAB_CCIR_clust3	NAB_NAB_clust3	CCIR_CCIR_clust4	CCIR_NAB_clust4	NAB_CCIR_clust4	NAB_NAB_clust4	COPHENET
4	euclidean	average	1	0	0	0	295	290	293	292	1	0	2	2	3	10	5	6	0,9066
4	euclidean	centroid	1	0	0	0	3	10	5	6	1	0	2	2	295	290	293	292	0,9045
4	euclidean	complete	2	10	2	6	12	0	9	0	284	290	284	292	2	0	5	2	0,8867
4	euclidean	median	1	0	0	0	295	300	293	297	1	0	2	2	3	0	5	1	0,8640
4	euclidean	single	0	0	0	1	298	300	298	297	1	0	2	2	1	0	0	0	0,8865
4	euclidean	ward	1	0	2	2	12	10	13	6	287	0	285	1	0	290	0	291	0,5481
4	euclidean	weighted	10	0	8	0	286	290	285	292	1	0	2	2	3	10	5	6	0,8656
4	squaredeuclidean	average	1	0	0	0	295	290	293	292	1	0	2	2	3	10	5	6	0,8656
4	squaredeuclidean	complete	2	10	2	6	12	0	9	0	284	290	284	292	2	0	5	2	0,8656
4	squaredeuclidean	single	0	0	0	1	298	300	298	297	1	0	2	2	1	0	0	0	0,8656
4	squaredeuclidean	weighted	1	0	0	0	294	290	291	292	1	0	2	2	4	10	7	6	0,8656
4	seuclidean	average	1	0	0	0	294	290	291	292	1	0	2	2	4	10	7	6	0,8656
4	seuclidean	complete	1	0	2	2	1	6	4	3	283	286	280	286	15	8	14	9	0,8656
4	seuclidean	single	0	0	1	0	298	300	297	298	1	0	0	0	1	0	2	2	0,8656
4	seuclidean	weighted	1	0	0	0	294	290	291	292	1	0	2	2	4	10	7	6	0,8656
4	cityblock	average	1	0	0	0	295	290	291	292	1	0	2	2	3	10	7	6	0,8656
4	cityblock	complete	284	286	283	284	12	7	10	11	1	0	2	2	3	7	5	3	0,8656
4	cityblock	single	0	0	1	0	298	300	297	298	1	0	2	2	1	0	0	0	0,8656
4	cityblock	weighted	1	0	0	0	295	290	291	292	1	0	2	2	3	10	7	6	0,8656
4	minkowski	average	1	0	0	0	295	290	293	292	1	0	2	2	3	10	5	6	0,8656
4	minkowski	complete	2	10	2	6	12	0	9	0	284	290	284	292	2	0	5	2	0,8656
4	minkowski	single	0	0	0	1	298	300	298	297	1	0	2	2	1	0	0	0	0,8656
4	minkowski	weighted	10	0	8	0	286	290	285	292	1	0	2	2	3	10	5	6	0,8656
4	chebychev	average	295	290	294	292	3	10	3	6	1	0	0	0	1	0	3	2	0,8656
4	chebychev	complete	1	0	0	1	4	10	6	6	294	290	291	291	1	0	3	2	0,8656
4	chebychev	single	0	0	0	1	298	300	298	297	1	0	2	2	1	0	0	0	0,8656
4	chebychev	weighted	4	10	6	6	295	0	291	1	0	290	0	291	1	0	3	2	0,8656
4	cosine	average	16	4	12	8	280	285	281	283	0	1	0	1	4	10	7	8	0,8656
4	cosine	complete	3	96	1	99	281	190	281	184	7	3	8	9	9	11	10	8	0,8656
4	cosine	single	0	0	0	1	300	298	300	299	0	1	0	0	0	1	0	0	0,8656
4	cosine	weighted	9	3	8	2	4	11	7	8	287	282	285	283	0	4	0	7	0,8656
4	correlation	average	14	4	11	10	282	285	282	281	0	1	0	1	4	10	7	8	0,8656
4	correlation	complete	0	11	0	21	281	5	285	33	7	270	3	236	12	14	12	10	0,8656
4	correlation	single	0	0	0	1	300	299	300	298	0	0	0	1	0	1	0	0	0,8656
4	correlation	weighted	0	47	0	32	283	236	279	249	0	3	0	3	17	14	21	16	0,8656
4	spearman	average	0	1	0	3	281	285	273	282	0	0	3	0	19	14	24	15	0,8656
4	spearman	complete	24	1	61	0	97	108	83	100	151	174	127	182	28	17	29	18	0,8656
4	spearman	single	0	0	1	0	300	299	298	300	0	1	0	0	0	0	1	0	0,8656
4	spearman	weighted	185	236	137	224	63	59	55	60	3	3	5	10	49	2	103	6	0,8656
4	hamming	average	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,8656
4	hamming	complete	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,8656
4	hamming	single	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,8656
4	hamming	weighted	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,8656
4	jaccard	average	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,8656
4	jaccard	complete	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,8656
4	jaccard	single	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,8656
4	jaccard	weighted	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,8656

HIEARCHICAL CLUSTERING con COEFFICIENTI CEPSTRALI. DATASET 15 IPS - RUMORE BIANCO																				
MAX_CLUSTER	DIST_METRIC	LINK_METHOD	CCIR_CCIR_clust1	CCIR_NAB_clust1	NAB_CCIR_clust1	NAB_NAB_clust1	CCIR_CCIR_clust2	CCIR_NAB_clust2	NAB_CCIR_clust2	NAB_NAB_clust2	CCIR_CCIR_clust3	CCIR_NAB_clust3	NAB_CCIR_clust3	NAB_NAB_clust3	CCIR_CCIR_clust4	CCIR_NAB_clust4	NAB_CCIR_clust4	NAB_NAB_clust4	COPHENET
4	euclidean	average	6	0	0	6	294	0	0	294	0	0	300	0	0	300	0	0	0,7930
4	euclidean	centroid	0	0	2	0	0	0	298	0	300	0	0	300	0	300	0	0	0,7929
4	euclidean	complete	128	0	0	128	172	0	0	172	0	0	300	0	0	300	0	0	0,7782
4	euclidean	median	0	0	2	0	0	0	298	0	300	0	0	300	0	300	0	0	0,7776
4	euclidean	single	0	0	2	0	0	0	298	0	300	0	0	300	0	300	0	0	0,7496
4	euclidean	ward	136	0	0	136	164	0	0	164	0	0	300	0	0	300	0	0	0,7894
4	euclidean	weighted	6	0	0	6	294	0	0	294	0	0	300	0	0	300	0	0	0,7781
4	squaredeuclidean	average	2	0	0	2	298	0	0	298	0	0	300	0	0	300	0	0	0,7781
4	squaredeuclidean	complete	128	0	0	128	172	0	0	172	0	0	300	0	0	300	0	0	0,7781
4	squaredeuclidean	single	0	0	2	0	0	0	298	0	300	0	0	300	0	300	0	0	0,7781
4	squaredeuclidean	weighted	2	0	0	2	298	0	0	298	0	0	300	0	0	300	0	0	0,7781
4	seuclidean	average	8	8	8	8	278	278	278	278	12	12	12	12	2	2	2	2	0,7781
4	seuclidean	complete	74	0	106	74	148	26	152	148	12	224	2	12	66	50	40	66	0,7781
4	seuclidean	single	2	2	2	2	294	294	294	294	2	2	2	2	2	2	2	2	0,7781
4	seuclidean	weighted	146	138	162	146	140	148	124	140	12	12	12	12	2	2	2	2	0,7781
4	cityblock	average	2	2	2	2	298	2	4	298	0	0	294	0	0	296	0	0	0,7781
4	cityblock	complete	288	0	0	288	10	0	10	10	0	0	290	0	2	300	0	2	0,7781
4	cityblock	single	0	2	0	0	298	298	298	298	2	0	0	2	0	0	2	0	0,7781
4	cityblock	weighted	4	0	4	4	4	2	296	4	0	296	0	0	292	2	0	292	0,7781
4	minkowski	average	6	0	0	6	294	0	0	294	0	0	300	0	0	300	0	0	0,7781
4	minkowski	complete	128	0	0	128	172	0	0	172	0	0	300	0	0	300	0	0	0,7781
4	minkowski	single	0	0	2	0	0	0	298	0	300	0	0	300	0	300	0	0	0,7781
4	minkowski	weighted	6	0	0	6	294	0	0	294	0	0	300	0	0	300	0	0	0,7781
4	chebychev	average	2	0	0	2	298	0	0	298	0	0	300	0	0	300	0	0	0,7781
4	chebychev	complete	46	0	0	46	254	0	0	254	0	0	300	0	0	300	0	0	0,7781
4	chebychev	single	0	0	2	0	0	0	298	0	300	0	0	300	0	300	0	0	0,7781
4	chebychev	weighted	0	2	0	0	0	298	0	0	300	0	0	300	0	0	300	0	0,7781
4	cosine	average	0	4	0	0	0	290	0	0	0	6	0	0	300	0	300	300	0,7781
4	cosine	complete	0	26	0	0	0	126	0	0	0	148	0	0	300	0	300	300	0,7781
4	cosine	single	0	2	0	0	0	296	0	0	0	2	0	0	300	0	300	300	0,7781
4	cosine	weighted	0	12	0	0	0	282	0	0	0	6	0	0	300	0	300	300	0,7781
4	correlation	average	0	8	0	0	0	286	0	0	0	6	0	0	300	0	300	300	0,7781
4	correlation	complete	0	44	0	0	0	88	0	0	0	168	0	0	300	0	300	300	0,7781
4	correlation	single	0	2	0	0	0	296	0	0	0	2	0	0	300	0	300	300	0,7781
4	correlation	weighted	0	98	0	0	0	198	0	0	0	4	0	0	300	0	300	300	0,7781
4	spearman	average	0	104	0	0	0	92	0	0	0	102	0	0	300	2	300	300	0,7781
4	spearman	complete	72	2	70	72	228	0	230	228	0	128	0	0	0	170	0	0	0,7781
4	spearman	single	0	2	0	0	300	294	300	300	0	2	0	0	0	2	0	0	0,7781
4	spearman	weighted	0	102	0	0	0	176	0	0	0	20	0	0	300	2	300	300	0,7781
4	hamming	average	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,7781
4	hamming	complete	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,7781
4	hamming	single	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,7781
4	hamming	weighted	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,7781
4	jaccard	average	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,7781
4	jaccard	complete	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,7781
4	jaccard	single	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,7781
4	jaccard	weighted	0	0	0	1	300	300	300	297	0	0	0	1	0	0	0	1	0,7781

K-MEANS CLUSTERING con COEFFICIENTI CEPSTRALI. DATASET 7.5 IPS - SILENZIO

CLUSTERS	DIST_METRIC	AVERAGE_SIL	CCIR_CCIR_clust1	CCIR_NAB_clust1	NAB_CCIR_clust1	NAB_NAB_clust1	CCIR_CCIR_clust2	CCIR_NAB_clust2	NAB_CCIR_clust2	NAB_NAB_clust2	CCIR_CCIR_clust3	CCIR_NAB_clust3	NAB_CCIR_clust3	NAB_NAB_clust3	CCIR_CCIR_clust4	CCIR_NAB_clust4	NAB_CCIR_clust4	NAB_NAB_clust4
2	sqeuclidean	0,5102	8	300	8	299	292	0	292	1	0	0	0	0	0	0	0	0
3	sqeuclidean	0,2478	2	300	6	295	208	0	50	5	90	0	244	0	0	0	0	0
4	sqeuclidean	0,5952	3	298	0	291	0	0	1	0	4	2	7	8	293	0	292	1
2	cityblock	0,2481	293	0	292	1	7	300	8	299	0	0	0	0	0	0	0	0
3	cityblock	0,2202	186	0	270	0	0	295	1	285	114	5	29	15	0	0	0	0
4	cityblock	0,1627	291	0	291	0	4	2	7	9	2	85	0	232	3	213	2	59
2	cosine	0,8182	293	298	292	290	7	2	8	10	0	0	0	0	0	0	0	0
3	cosine	0,3049	7	2	8	10	151	3	274	51	142	295	18	239	0	0	0	0
4	cosine	0,1800	7	2	8	10	79	254	10	154	83	0	214	18	131	44	68	118
2	correlation	0,2193	121	256	45	174	179	44	255	126	0	0	0	0	0	0	0	0
3	correlation	0,2982	7	2	8	10	177	289	61	235	116	9	231	55	0	0	0	0
4	correlation	0,1882	110	222	39	157	7	2	8	10	71	5	181	33	112	71	72	100

K-MEANS CLUSTERING con COEFFICIENTI CEPSTRALI. DATASET 7.5 IPS - RUMORE BIANCO

CLUSTERS	DIST_METRIC	AVERAGE_SIL	CCIR_CCIR_clust1	CCIR_NAB_clust1	NAB_CCIR_clust1	NAB_NAB_clust1	CCIR_CCIR_clust2	CCIR_NAB_clust2	NAB_CCIR_clust2	NAB_NAB_clust2	CCIR_CCIR_clust3	CCIR_NAB_clust3	NAB_CCIR_clust3	NAB_NAB_clust3	CCIR_CCIR_clust4	CCIR_NAB_clust4	NAB_CCIR_clust4	NAB_NAB_clust4
2	sqeuclidean	0,5936	275	0	300	299	25	300	0	1	0	0	0	0	0	0	0	0
3	sqeuclidean	0,6589	299	0	0	300	1	300	0	0	0	0	300	0	0	0	0	0
4	sqeuclidean	0,5199	0	0	169	0	1	300	0	0	0	0	131	0	299	0	0	300
2	cityblock	0,2951	3	300	0	0	297	0	300	300	0	0	0	0	0	0	0	0
3	cityblock	0,3019	1	300	0	0	299	0	0	300	0	0	300	0	0	0	0	0
4	cityblock	0,1954	0	0	299	0	117	0	1	170	0	300	0	0	183	0	0	130
2	cosine	0,6243	300	300	59	300	0	0	241	0	0	0	0	0	0	0	0	0
3	cosine	0,6236	300	300	32	300	0	0	156	0	0	0	112	0	0	0	0	0
4	cosine	0,6349	0	0	70	0	300	300	29	300	0	0	111	0	0	0	90	0
2	correlation	0,6386	0	0	206	0	300	300	94	300	0	0	0	0	0	0	0	0
3	correlation	0,6452	300	300	34	300	0	0	93	0	0	0	173	0	0	0	0	0
4	correlation	0,2922	0	0	116	0	222	291	42	217	0	0	94	0	78	9	48	83

K-MEANS CLUSTERING con COEFFICIENTI CEPSTRALI. DATASET 15 IPS - SILENZIO

CLUSTERS	DIST_METRIC	AVERAGE_SIL	CCIR_CCIR_clust1	CCIR_NAB_clust1	NAB_CCIR_clust1	NAB_NAB_clust1	CCIR_CCIR_clust2	CCIR_NAB_clust2	NAB_CCIR_clust2	NAB_NAB_clust2	CCIR_CCIR_clust3	CCIR_NAB_clust3	NAB_CCIR_clust3	NAB_NAB_clust3	CCIR_CCIR_clust4	CCIR_NAB_clust4	CCIR_CCIR_clust5	NAB_NAB_clust4
2	sqeuclidean	0,4994	300	10	300	10	0	290	0	290	0	0	0	0	0	0	0	0
3	sqeuclidean	0,6846	0	290	0	291	5	10	9	8	295	0	291	1	0	0	0	0
4	sqeuclidean	0,4166	0	174	0	137	5	10	9	8	295	0	291	1	0	116	232	154
2	cityblock	0,2968	0	290	0	292	300	10	300	8	0	0	0	0	0	0	0	0
3	cityblock	0,3411	0	290	0	291	294	0	291	1	6	10	9	8	0	0	0	0
4	cityblock	0,2292	200	0	155	1	4	10	9	8	96	0	136	2	0	290	580	289
2	cosine	0,3094	293	22	297	51	7	278	3	249	0	0	0	0	0	0	0	0
3	cosine	0,3817	2	278	2	244	12	14	11	10	286	8	287	46	0	0	0	0
4	cosine	0,3497	2	150	1	132	2	132	1	129	12	13	11	10	284	5	-274	29
2	correlation	0,3111	287	28	296	68	13	272	4	232	0	0	0	0	0	0	0	0
3	correlation	0,3857	5	273	1	227	282	13	287	62	13	14	12	11	0	0	0	0
4	correlation	0,3642	7	124	1	129	276	11	286	51	4	152	2	110	13	13	13	10

K-MEANS CLUSTERING con COEFFICIENTI CEPSTRALI. DATASET 15 IPS - RUMORE BIANCO

CLUSTERS	DIST_METRIC	AVERAGE_SIL	CCIR_CCIR_clust1	CCIR_NAB_clust1	NAB_CCIR_clust1	NAB_NAB_clust1	CCIR_CCIR_clust2	CCIR_NAB_clust2	NAB_CCIR_clust2	NAB_NAB_clust2	CCIR_CCIR_clust3	CCIR_NAB_clust3	NAB_CCIR_clust3	NAB_NAB_clust3	CCIR_CCIR_clust4	CCIR_NAB_clust4	NAB_CCIR_clust4	NAB_NAB_clust4
2	sqeuclidean	0,6432	300	0	300	300	0	300	0	0	0	0	0	0	0	0	0	0
3	sqeuclidean	0,4623	300	0	300	300	0	139	0	0	0	161	0	0	0	0	0	0
4	sqeuclidean	0,4583	160	0	0	162	0	300	0	0	140	0	0	138	0	0	300	0
2	cityblock	0,3386	0	300	0	0	300	0	300	300	0	0	0	0	0	0	0	0
3	cityblock	0,2310	300	300	0	300	0	0	141	0	0	0	159	0	0	0	0	0
4	cityblock	0,2807	0	0	147	0	300	0	0	300	0	0	153	0	0	300	0	0
2	cosine	0,8019	300	1	300	300	0	299	0	0	0	0	0	0	0	0	0	0
3	cosine	0,2897	143	1	103	146	157	0	197	154	0	299	0	0	0	0	0	0
4	cosine	0,2574	103	0	110	102	0	299	0	0	90	0	122	90	107	1	68	108
2	correlation	0,8121	0	299	0	0	300	1	300	300	0	0	0	0	0	0	0	0
3	correlation	0,3096	0	299	0	0	158	0	186	152	142	1	114	148	0	0	0	0
4	correlation	0,2050	156	0	185	152	0	122	0	0	0	177	0	0	144	1	115	148

Appendice D

Prestazioni dei classificatori

Le tabelle illustrate nelle pagine successive mostrano le *performance* nella classificazione dei dati di ciascuno dei quattro *dataset*, al variare delle classi di predizione e del tipo di classificatore.

TRAINING PERFORMANCE - DATASET 7.5 ips, SILENZIO - CLASSI correct eq./wrong eq.							
CLASSIFIER	TRAINING_TIME (ms)	CV_ERR	ACCUR	RECALL_1	SPECIF_1	PRECIS_1	AUC_1
simpleTree	725,83	0,2678	0,7711	0,8667	0,6756	0,7276	0,7744
mediumTree	83,13	0,2167	0,8533	0,9178	0,7889	0,8130	0,8856
complexTree	36,61	0,2589	0,9433	0,9289	0,9578	0,9565	0,9749
linearSVM	2,02	0,5111	0,5711	0,5622	0,5800	0,5724	0,5711
quadrativSVM	50,45	0,2167	0,8656	0,9133	0,8178	0,8337	0,9249
cubicSVM	6,27	0,2678	0,9800	0,9889	0,9711	0,9716	0,9832
gaussianSVM	473,97	0,2589	0,9811	0,9733	0,9889	0,9887	0,9987
fineKNN	127,82	0,2833	0,9778	0,9778	0,9778	0,9778	0,9778
mediumKNN	15,62	0,2689	0,8044	0,9289	0,6800	0,7438	0,8997
coarseKNN	15,54	0,2733	0,7444	0,9200	0,5689	0,6809	0,8605
cosineKNN	22,88	0,3000	0,9756	0,9733	0,9778	0,9777	0,9756
weigthedKNN	27,86	0,3000	0,9744	0,9711	0,9778	0,9776	0,9744
	132,33	0,2853	0,8702				

ACCURACY, RECALL, SPECIFICITY, PRECISION: sono calcolati a partire dalla matrice di confusione ottenuta classificando il training set (75% del dataset iniziale) con il miglior modello, scelto mediante k-fold Validation
AUC (Area Under Curve): area sotto la curva ROC, che ha lungo l'asse x il False Positive Rate (1 - Specificity), e lungo l'asse y il True Positive Rate (Sensitivity o Recall),
TRAINING TIME: espresso in millisecondi, tempo per addestrare il classificatore
TEST_SET_ACCURACY, TEST_SET_RECALL, TEST_SET_SPECIFICITY, TEST_SET_PRECISION, TEST_SET_AUC: sono calcolati a partire dalla matrice di confusione ottenuta classificando il test set (25% del dataset iniziale)
TEST TIME: in millisecondi, tempo che impiega il modello addestrato a classificare i campioni del test set

CLASSIFICATION PERFORMANCE - DATASET 7.5 ips, SILENZIO - CLASSI correct eq./wrong eq.						
CLASSIFIER	TEST_TIME (ms)	TEST_ACCUR	TEST_RECALL_1	TEST_SPECIF_1	TEST_PRECIS_1	TEST_AUC_1
simpleTree	0,8511	0,6333	0,7333	0,5333	0,6111	0,6422
mediumTree	0,9271	0,6400	0,7267	0,5533	0,6193	0,6722
complexTree	0,9293	0,6733	0,6800	0,6667	0,6711	0,7253
linearSVM	138,887,446	0,5267	0,5067	0,5467	0,5278	0,5114
quadrativSVM	146,141,318	0,7133	0,7867	0,6400	0,6860	0,7704
cubicSVM	133,238,421	0,6067	0,6667	0,5467	0,5952	0,6304
gaussianSVM	141,309,287	0,6267	0,5667	0,6867	0,6439	0,6855
fineKNN	2,707,926	0,6333	0,6467	0,6200	0,6299	0,6333
mediumKNN	3,609,784	0,6600	0,8133	0,5067	0,6224	0,7300
coarseKNN	6,618,786	0,6700	0,8667	0,4733	0,6220	0,7443
cosineKNN	2,913,179	0,6100	0,5867	0,6333	0,6154	0,6100
weigthedKNN	2,926,419	0,6467	0,6800	0,6133	0,6375	0,6467
	0,9025	0,6367				

TRAINING PERFORMANCE - DATASET 7.5 ips, SILENZIO - CLASSI correct eq. / eq. CCIR_NAB / eq. NAB_CCIR																
CLASSIFIER	TRAINING_TIME (ms)	CV_ERROR	ACCUR	RECALL_1	SPECIF_1	PRECIS_1	AUC_1	RECALL_2	SPECIF_2	PRECIS_2	AUC_2	RECALL_3	SPECIF_3	PRECIS_3	AUC_3
simpleTree	24,96	0,2600	0,7789	0,7111	0,8578	0,8333	0,8174	0,8978	0,8652	0,6894	0,9201	0,7956	0,9348	0,8027	0,9180
mediumTree	333,60	0,2200	0,8611	0,8556	0,8689	0,8671	0,8945	0,8844	0,9437	0,8397	0,9451	0,8489	0,9585	0,8721	0,9677
complexTree	19,60	0,2311	0,9311	0,9467	0,9178	0,9201	0,9761	0,9600	0,9748	0,9270	0,9899	0,8711	0,9881	0,9608	0,9836
linearSVM	1492,17	0,1856	0,8289	0,8400	0,8178	0,8217	0,9126	0,8133	0,9556	0,8592	0,9731	0,8222	0,9378	0,8150	0,9685
quadrativSVM	10022,61	0,2133	0,9133	0,9422	0,8844	0,8908	0,9548	0,8889	0,9748	0,9217	0,9862	0,8800	0,9867	0,9565	0,9879
cubicSVM	1205,66	0,3067	0,8944	0,9867	0,8022	0,8330	0,9862	0,9644	0,9956	0,9864	0,9945	0,6400	0,9956	0,9796	0,9959
gaussianSVM	933,33	0,3233	0,9733	0,9956	0,9511	0,9532	0,9984	0,9378	1,0000	1,0000	0,9995	0,9644	0,9970	0,9909	0,9992
fineKNN	18,17	0,2933	0,9856	0,9778	0,9933	0,9932	0,9856	0,9911	0,9956	0,9867	0,9933	0,9956	0,9896	0,9697	0,9926
mediumKNN	10,15	0,2456	0,8044	0,8733	0,7378	0,7691	0,8974	0,6622	0,9778	0,9085	0,9650	0,8089	0,9363	0,8089	0,9662
coarseKNN	9,49	0,2644	0,7511	0,9511	0,5511	0,6794	0,8608	0,5556	0,9867	0,9328	0,9674	0,5467	0,9807	0,9044	0,9569
cosineKNN	1,73	0,2933	0,9800	0,9733	0,9867	0,9865	0,9800	0,9822	0,9926	0,9779	0,9874	0,9911	0,9896	0,9696	0,9904
weigthedKNN	10,27	0,2944	0,9800	0,9867	0,9733	0,9737	0,9800	0,9733	0,9956	0,9865	0,9844	0,9733	0,9956	0,9865	0,9844
	1173,48	0,2609	0,8902												

CLASSIFICATION PERFORMANCE - DATASET 7.5 ips, SILENZIO - CLASSI correct eq. / eq. CCIR_NAB / eq. NAB_CCIR															
CLASSIFIER	TEST_TIME (ms)	TEST_ACCUR	TEST_RECALL_1	TEST_SPECIF_1	TEST_PRECIS_1	TEST_AUC_1	TEST_RECALL_2	TEST_SPECIF_2	TEST_PRECIS_2	TEST_AUC_2	TEST_RECALL_3	TEST_SPECIF_3	TEST_PRECIS_3	TEST_AUC_3
simpleTree	0,72	0,6600	0,6000	0,7267	0,6870	0,6654	0,7467	0,8667	0,6512	0,8796	0,6933	0,8622	0,6265	0,8722
mediumTree	0,92	0,7133	0,7333	0,6933	0,7051	0,7429	0,6933	0,9333	0,7761	0,8966	0,6933	0,8889	0,6753	0,9040
complexTree	0,65	0,7033	0,7333	0,6733	0,6918	0,7139	0,5867	0,9289	0,7333	0,7959	0,7600	0,8933	0,7037	0,8665
linearSVM	179,38	0,7333	0,7400	0,7267	0,7303	0,8104	0,6400	0,9511	0,8136	0,9490	0,8133	0,8756	0,6854	0,9381
quadrativSVM	613,25	0,6933	0,7533	0,6400	0,6766	0,7485	0,5600	0,9378	0,7500	0,9240	0,7067	0,8933	0,6883	0,9170
cubicSVM	995,28	0,6800	0,8333	0,5267	0,6378	0,7078	0,5333	0,9689	0,8511	0,8881	0,5200	0,9200	0,6842	0,8646
gaussianSVM	856,64	0,5800	0,7800	0,3800	0,5571	0,6512	0,2267	0,9644	0,6800	0,8984	0,5333	0,8889	0,6154	0,8642
fineKNN	3,03	0,6267	0,6667	0,5867	0,6173	0,6267	0,4667	0,9244	0,6731	0,6956	0,7067	0,8533	0,6163	0,7800
mediumKNN	3,41	0,6467	0,7067	0,5867	0,6310	0,7062	0,4133	0,9644	0,7949	0,9269	0,7600	0,8400	0,6129	0,9024
coarseKNN	7,07	0,6733	0,8933	0,4533	0,6204	0,7575	0,3200	0,9822	0,8571	0,9329	0,5867	0,9467	0,7857	0,9304
cosineKNN	2,72	0,6000	0,5867	0,6133	0,6027	0,6000	0,5467	0,9022	0,6508	0,7244	0,6800	0,8222	0,5604	0,7511
weigthedKNN	2,71	0,6233	0,6667	0,5800	0,6135	0,6233	0,4667	0,9200	0,6604	0,6933	0,6933	0,8578	0,6190	0,7756
	222,15	0,6611												

TRAINING PERFORMANCE - DATASET 7.5 ips, SILENZIO - CLASSI post eq. NAB / post eq. CCIR							
CLASSIFIER	TRAINING_TIME (ms)	CV_ERROR	ACCUR	RECALL_1	SPECIF_1	PRECIS_1	AUC_1
simpleTree	11,41	0,0111	0,9967	0,9933	1,0000	1,0000	0,9988
mediumTree	11,99	0,0133	0,9967	0,9933	1,0000	1,0000	0,9988
complexTree	11,42	0,0133	0,9967	0,9933	1,0000	1,0000	0,9988
linearSVM	499,94	0,0022	1,0000	1,0000	1,0000	1,0000	1,0000
quadrativSVM	988,27	0,0056	1,0000	1,0000	1,0000	1,0000	1,0000
cubicSVM	252,06	0,0044	1,0000	1,0000	1,0000	1,0000	1,0000
gaussianSVM	498,50	0,0278	0,9989	0,9978	1,0000	1,0000	1,0000
fineKNN	10,33	0,0322	0,9989	0,9978	1,0000	1,0000	0,9989
mediumKNN	9,39	0,0211	0,9844	0,9911	0,9778	0,9781	0,9993
coarseKNN	9,37	0,0078	0,9933	0,9867	1,0000	1,0000	0,9999
cosineKNN	11,41	0,0278	1,0000	1,0000	1,0000	1,0000	1,0000
weigthedKNN	11,41	0,0356	0,9989	0,9978	1,0000	1,0000	0,9989
	193,79	0,0169	0,9970				

CLASSIFICATION PERFORMANCE - DATASET 7.5 ips, SILENZIO - CLASSI post eq. NAB / post eq. CCIR						
CLASSIFIER	TEST_TIME (ms)	TEST_ACCUR	TEST_RECALL_1	TEST_SPECIF_1	TEST_PRECIS_1	TEST_AUC_1
simpleTree	0,49	0,9900	0,9933	0,9867	0,9868	0,9932
mediumTree	0,49	0,9900	0,9933	0,9867	0,9868	0,9932
complexTree	0,49	0,9900	0,9933	0,9867	0,9868	0,9932
linearSVM	138,69	0,9967	1,0000	0,9933	0,9934	1,0000
quadrativSVM	14,26	0,9867	0,9933	0,9800	0,9803	0,9999
cubicSVM	136,72	1,0000	1,0000	1,0000	1,0000	1,0000
gaussianSVM	140,89	0,9700	1,0000	0,9400	0,9434	0,9992
fineKNN	2,43	0,9767	0,9933	0,9600	0,9613	0,9767
mediumKNN	3,36	0,9867	0,9933	0,9800	0,9803	0,9996
coarseKNN	5,52	1,0000	1,0000	1,0000	1,0000	1,0000
cosineKNN	2,52	0,9833	1,0000	0,9667	0,9677	0,9833
weigthedKNN	2,37	0,9733	0,9867	0,9600	0,9610	0,9733
	37,35	0,9869				

TRAINING PERFORMANCE - DATASET 7.5 ips, SILENZIO - CLASSI eq. CCIR_CCIR / eq. CCIR_NAB / eq. NAB_CCIR / eq. NAB_NAB																			
CLASSIFIER	TRAINING_TIME (ms)	CV_ERROR	ACCUR	RECALL_1	SPECIF_1	PRECIS_1	AUC_1	RECALL_2	SPECIF_2	PRECIS_2	AUC_2	RECALL_3	SPECIF_3	PRECIS_3	AUC_3	RECALL_4	SPECIF_4	PRECIS_4	AUC_4
simpleTree	17,45	0,2700	0,7633	0,7956	0,9319	0,7956	0,9321	0,6933	0,9274	0,7610	0,9112	0,7822	0,9363	0,8037	0,9259	0,7822	0,8889	0,7012	0,9050
mediumTree	17,34	0,2044	0,8522	0,8533	0,9556	0,8649	0,9635	0,8800	0,9363	0,8216	0,9565	0,8578	0,9496	0,8502	0,9610	0,8178	0,9615	0,9190	0,9527
complexTree	19,33	0,2600	0,9389	0,9467	0,9748	0,9261	0,9892	0,9422	0,9822	0,9464	0,9890	0,9156	0,9837	0,9493	0,9848	0,9511	0,9778	0,8996	0,9882
linearSVM	1,84	0,1922	0,8333	0,7956	0,9467	0,8326	0,9692	0,8267	0,9570	0,8651	0,9743	0,8400	0,9319	0,8043	0,9689	0,8711	0,9422	0,8043	0,9730
quadrativSVM	6,69	0,2189	0,9311	0,9067	0,9822	0,9444	0,9880	0,9378	0,9778	0,9336	0,9890	0,9467	0,9689	0,9103	0,9889	0,9333	0,9793	0,9509	0,9884
cubicSVM	7,70	0,2656	0,9844	0,9956	0,9911	0,9739	0,9990	0,9822	0,9956	0,9866	0,9957	0,9689	0,9985	0,9954	0,9981	0,9911	0,9941	0,9604	0,9950
gaussianSVM	137,23	0,2500	0,9789	0,9689	0,9956	0,9864	0,9990	0,9822	0,9896	0,9693	0,9994	0,9867	0,9956	0,9867	0,9995	0,9778	0,9911	0,9823	0,9996
fineKNN	10,62	0,3167	0,9722	0,9556	0,9956	0,9862	0,9756	0,9778	0,9911	0,9735	0,9844	0,9867	0,9852	0,9569	0,9859	0,9689	0,9911	0,9911	0,9800
mediumKNN	0,91	0,2844	0,7867	0,8622	0,8993	0,7405	0,9551	0,8000	0,9363	0,8072	0,9672	0,7067	0,9526	0,8325	0,9662	0,7778	0,9274	0,7098	0,9405
coarseKNN	8,76	0,2556	0,7544	0,8178	0,9096	0,7510	0,9151	0,7156	0,9141	0,7352	0,9633	0,7244	0,9511	0,8316	0,9574	0,7600	0,8978	0,6792	0,8889
cosineKNN	10,53	0,3222	0,9722	0,9689	0,9911	0,9732	0,9800	0,9600	0,9956	0,9863	0,9778	0,9733	0,9911	0,9733	0,9822	0,9867	0,9852	0,9440	0,9859
weigthedKNN	9,63	0,3133	0,9756	0,9733	0,9970	0,9910	0,9852	0,9733	0,9881	0,9648	0,9807	0,9956	0,9911	0,9739	0,9933	0,9600	0,9911	0,9000	0,9756
	20,67	0,2628	0,8953																

CLASSIFICATION PERFORMANCE - DATASET 7.5 ips, SILENZIO - CLASSI eq. CCIR_CCIR / eq. CCIR_NAB / eq. NAB_CCIR / eq. NAB_NAB																			
CLASSIFIER	TEST_TIME (ms)	TEST_ACCUR	TEST_RECALL_1	TEST_SPECIF_1	TEST_PRECIS_1	TEST_AUC_1	TEST_RECALL_2	TEST_SPECIF_2	TEST_PRECIS_2	TEST_AUC_2	TEST_RECALL_3	TEST_SPECIF_3	TEST_PRECIS_3	TEST_AUC_3	TEST_RECALL_4	TEST_SPECIF_4	TEST_PRECIS_4	TEST_AUC_4
simpleTree	0,56	0,6200	0,6000	0,8933	0,6522	0,8837	0,4933	0,9022	0,6271	0,8678	0,6933	0,8667	0,6341	0,8854	0,6933	0,8311	0,5778	0,8606
mediumTree	0,55	0,6467	0,5200	0,8889	0,6094	0,8512	0,6267	0,9289	0,7460	0,8991	0,6800	0,8267	0,5667	0,8617	0,7600	0,8844	0,6145	0,8911
complexTree	0,64	0,6400	0,4800	0,8889	0,5902	0,8008	0,5467	0,9556	0,8039	0,8137	0,6933	0,8267	0,5714	0,8409	0,8400	0,8489	0,5361	0,8986
linearSVM	11,43	0,7400	0,5867	0,9600	0,8302	0,9333	0,6400	0,9511	0,8136	0,9482	0,8933	0,8622	0,6837	0,9393	0,8400	0,8800	0,7444	0,9418
quadrativSVM	4,95	0,6933	0,6000	0,9333	0,7500	0,9273	0,5333	0,9511	0,7843	0,9306	0,8000	0,8667	0,6667	0,9291	0,8400	0,8400	0,6061	0,9191
cubicSVM	4,90	0,6667	0,5733	0,8844	0,6232	0,8689	0,5467	0,9644	0,8367	0,9380	0,6800	0,8578	0,6145	0,8980	0,8667	0,8489	0,5152	0,9147
gaussianSVM	7,53	0,6500	0,4400	0,9511	0,7500	0,8693	0,5200	0,9511	0,7800	0,9089	0,8667	0,7822	0,5702	0,8892	0,7733	0,8489	0,7065	0,8971
fineKNN	2,58	0,6200	0,5733	0,8978	0,6515	0,7356	0,4800	0,9244	0,6792	0,7022	0,7200	0,8444	0,6067	0,7822	0,7067	0,8267	0,5870	0,7667
mediumKNN	3,32	0,6767	0,6667	0,8667	0,6250	0,8753	0,6133	0,9422	0,7797	0,9364	0,6133	0,8933	0,6571	0,9045	0,8133	0,8667	0,5055	0,8950
coarseKNN	5,83	0,7100	0,7467	0,9289	0,7778	0,9055	0,5600	0,9156	0,6885	0,9323	0,7733	0,9156	0,7532	0,9308	0,7600	0,8533	0,6444	0,8625
cosineKNN	2,67	0,6200	0,5200	0,8933	0,6190	0,7067	0,6133	0,9067	0,6866	0,7600	0,6800	0,8222	0,5604	0,7511	0,6667	0,8711	0,6456	0,7689
weigthedKNN	3,01	0,6067	0,5733	0,8933	0,6418	0,7333	0,4533	0,9200	0,6538	0,6867	0,7067	0,8444	0,6023	0,7756	0,6933	0,8178	0,5699	0,7556
	4,00	0,6575																

TRAINING PERFORMANCE - DATASET 7.5 ips, RUMORE - CLASSI correct eq. / wrong eq.							
CLASSIFIER	TRAINING_TIME (ms)	CV_ERROR	ACCUR	RECALL_1	SPECIF_1	PRECIS_1	AUC_1
simpleTree	11,24	0,0000	1,0000	1,0000	1,0000	1,0000	1,0000
mediumTree	11,91	0,0011	1,0000	1,0000	1,0000	1,0000	1,0000
complexTree	12,05	0,0000	1,0000	1,0000	1,0000	1,0000	1,0000
linearSVM	755,21	0,4133	0,5056	1,0000	0,4556	0,5051	0,4984
quadrativSVM	624,41	0,0000	1,0000	1,0000	1,0000	1,0000	1,0000
cubicSVM	1,35	0,0033	1,0000	1,0000	1,0000	1,0000	1,0000
gaussianSVM	580,72	0,0200	1,0000	1,0000	1,0000	1,0000	1,0000
fineKNN	13,22	0,1922	0,9867	0,9956	0,9778	0,9782	0,9867
mediumKNN	9,31	0,2256	0,8222	0,9822	0,6622	0,7441	0,9352
coarseKNN	9,05	0,2889	0,7333	1,0000	0,4667	0,6522	0,9858
cosineKNN	11,29	0,1944	0,9867	1,0000	0,9733	0,9740	0,9867
weigthedKNN	10,03	0,2044	0,9856	1,0000	0,9711	0,9719	0,9856
	170,82	0,1286	0,9183				

ACCURACY, RECALL, SPECIFICITY, PRECISION: sono calcolati a partire dalla matrice di confusione ottenuta classificando il training set (75% del dataset iniziale) con il miglior modello, scelto mediante k-fold Validation
AUC (Area Under Curve): area sotto la curva ROC, che ha lungo l'asse x il False Positive Rate (1 - Specificity), e lungo l'asse y il True Positive Rate (Sensitivity o Recall),
TRAINING TIME: espresso in millisecondi, tempo per addestrare il classificatore
TEST_SET_ACCURACY, TEST_SET_RECALL, TEST_SET_SPECIFICITY, TEST_SET_PRECISION, TEST_SET_AUC: sono calcolati a partire dalla matrice di confusione ottenuta classificando il test set (25% del dataset iniziale)
TEST TIME: in millisecondi, tempo che impiega il modello addestrato a classificare i campioni del test set

CLASSIFICATION PERFORMANCE - DATASET 7.5 ips, RUMORE - CLASSI correct eq. / wrong eq.						
CLASSIFIER	TEST_TIME (ms)	TEST_ACCUR	TEST_RECALL_1	TEST_SPECIF_1	TEST_PRECIS_1	TEST_AUC_1
simpleTree	0,56	1,0000	1,0000	1,0000	1,0000	1,0000
mediumTree	0,43	1,0000	1,0000	1,0000	1,0000	1,0000
complexTree	0,40	1,0000	1,0000	1,0000	1,0000	1,0000
linearSVM	138,88	0,5367	0,5800	0,4933	0,5337	0,4983
quadrativSVM	138,04	1,0000	1,0000	1,0000	1,0000	1,0000
cubicSVM	131,93	1,0000	1,0000	1,0000	1,0000	1,0000
gaussianSVM	143,85	1,0000	1,0000	1,0000	1,0000	1,0000
fineKNN	2,37	0,9600	0,9867	0,9333	0,9367	0,9600
mediumKNN	3,25	0,8667	1,0000	0,7333	0,7895	0,9847
coarseKNN	5,55	0,7500	1,0000	0,5000	0,6667	0,9822
cosineKNN	2,44	0,9667	1,0000	0,9333	0,9375	0,9667
weigthedKNN	2,41	0,9633	1,0000	0,9267	0,9317	0,9633
	47,51	0,9203				

TRAINING PERFORMANCE - DATASET 7.5 ips, RUMORE - CLASSI correct eq. / eq. CCIR_NAB / eq. NAB_CCIR															
CLASSIFIER	TRAINING_TIME (ms)	CV_ERROR	ACCUR	RECALL_1	SPECIF_1	PRECIS_1	AUC_1	RECALL_2	SPECIF_2	PRECIS_2	AUC_2	RECALL_3	SPECIF_3	PRECIS_3	AUC_3
simpleTree	13,50	0,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
mediumTree	11,99	0,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
complexTree	11,19	0,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
linearSVM	757,61	0,0033	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
quadrativSVM	1,13	0,0011	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
cubicSVM	752,59	0,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
gaussianSVM	976,20	0,0711	0,9978	1,0000	1,0000	1,0000	1,0000	0,9956	0,9985	0,9956	1,0000	0,9956	0,9985	0,9956	1,0000
fineKNN	9,69	0,2056	0,9867	1,0000	0,9733	0,9740	0,9867	0,9822	1,0000	1,0000	0,9911	0,9644	1,0000	1,0000	0,9822
mediumKNN	8,84	0,1867	0,8600	0,9956	0,7244	0,7832	0,9385	0,6578	1,0000	1,0000	0,9989	0,7911	0,9970	0,9889	0,9983
coarseKNN	8,64	0,2911	0,7356	1,0000	0,4711	0,6541	0,9818	0,4844	1,0000	1,0000	0,9989	0,4578	1,0000	1,0000	0,9994
cosineKNN	11,94	0,1822	0,9856	0,9911	0,9800	0,9802	0,9856	0,9778	0,9941	0,9821	0,9859	0,9822	1,0000	1,0000	0,9911
weigthedKNN	959,00	0,1978	0,9856	0,9956	0,9756	0,9760	0,9856	0,9733	0,9970	0,9910	0,9852	0,9778	1,0000	1,0000	0,9889
	293,53	0,0949	0,9626												

CLASSIFICATION PERFORMANCE - DATASET 7.5 ips, RUMORE - CLASSI correct eq. / eq. CCIR_NAB / eq. NAB_CCIR															
CLASSIFIER	TEST_TIME (ms)	TEST_ACCUR	TEST_RECALL_1	TEST_SPECIF_1	TEST_PRECIS_1	TEST_AUC_1	TEST_RECALL_2	TEST_SPECIF_2	TEST_PRECIS_2	TEST_AUC_2	TEST_RECALL_3	TEST_SPECIF_3	TEST_PRECIS_3	TEST_AUC_3
simpleTree	0,52	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
mediumTree	0,41	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
complexTree	0,44	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
linearSVM	20,55	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
quadrativSVM	19,05	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
cubicSVM	17,57	0,9967	1,0000	0,9933	0,9934	1,0000	0,9867	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
gaussianSVM	18,04	0,9967	1,0000	1,0000	1,0000	1,0000	0,9867	1,0000	1,0000	0,9999	1,0000	0,9956	0,9868	0,9999
fineKNN	2,45	0,9633	1,0000	0,9267	0,9317	0,9633	0,9600	1,0000	1,0000	0,9800	0,8933	1,0000	1,0000	0,9467
mediumKNN	3,34	0,9167	1,0000	0,8333	0,8571	0,9839	0,7600	1,0000	1,0000	0,9991	0,9067	1,0000	1,0000	0,9994
coarseKNN	584,00	0,7433	1,0000	0,4867	0,6608	0,9809	0,4667	1,0000	1,0000	0,9991	0,5067	1,0000	1,0000	0,9998
cosineKNN	278,00	0,9667	0,9733	0,9600	0,9605	0,9667	0,9600	0,9911	0,9730	0,9756	0,9600	0,9911	0,9730	0,9756
weigthedKNN	2,53	0,9600	0,9867	0,9333	0,9367	0,9600	0,9333	1,0000	1,0000	0,9667	0,9333	0,9911	0,9722	0,9622
	78,91	0,9619												

TRAINING PERFORMANCE - DATASET 7.5 ips, RUMORE - CLASSI post eq. NAB / post eq. CCIR							
CLASSIFIER	TRAINING_TIME (ms)	CV_ERROR	ACCUR	RECALL_1	SPECIF_1	PRECIS_1	AUC_1
simpleTree	1,26	0,0878	0,9144	0,9533	0,8756	0,8845	0,9572
mediumTree	13,16	0,1222	0,9356	0,9578	0,9133	0,9170	0,9808
complexTree	15,80	0,1122	0,9700	0,9778	0,9622	0,9628	0,9915
linearSVM	1,20	0,2233	0,7933	0,7800	0,8067	0,8014	0,8997
quadrativSVM	14,65	0,1867	0,8967	0,9000	0,8933	0,8940	0,9431
cubicSVM	974,30	0,0978	0,9967	1,0000	0,9933	0,9934	1,0000
gaussianSVM	455,49	0,4833	0,9544	0,9400	0,9689	0,9680	0,9885
fineKNN	10,96	0,2967	0,9811	0,9800	0,9822	0,9822	0,9811
mediumKNN	8,83	0,2822	0,8056	0,9222	0,6889	0,7477	0,9019
coarseKNN	11,05	0,2456	0,7667	0,7689	0,7644	0,7655	0,8833
cosineKNN	1,29	0,2633	0,9811	0,9844	0,9778	0,9779	0,9811
weigthedKNN	12,80	0,2744	0,9800	0,9800	0,9800	0,9800	0,9800
	126,73	0,2230	0,9146				

CLASSIFICATION PERFORMANCE - DATASET 7.5 ips, RUMORE - CLASSI post eq. NAB / post eq. CCIR						
CLASSIFIER	TEST_TIME (ms)	TEST_ACCUR	TEST_RECALL_1	TEST_SPECIF_1	TEST_PRECIS_1	TEST_AUC_1
simpleTree	0,41	0,9100	0,9067	0,9133	0,9128	0,9550
mediumTree	0,50	0,9000	0,8867	0,9133	0,9110	0,9516
complexTree	0,48	0,9100	0,8667	0,9533	0,9489	0,9234
linearSVM	138,84	0,7833	0,7867	0,7800	0,7815	0,8938
quadrativSVM	14,78	0,8233	0,8000	0,8467	0,8392	0,8989
cubicSVM	149,85	0,9533	0,9333	0,9733	0,9722	0,9913
gaussianSVM	149,27	0,9467	0,9067	0,9867	0,9855	0,9772
fineKNN	2,36	0,9600	0,9667	0,9533	0,9539	0,9600
mediumKNN	3,30	0,8000	0,9067	0,6933	0,7473	0,9027
coarseKNN	6,02	0,7867	0,8067	0,7667	0,7756	0,8881
cosineKNN	2,84	0,9400	0,9533	0,9267	0,9286	0,9400
weigthedKNN	2,89	0,9333	0,9400	0,9267	0,9276	0,9333
	39,30	0,8872				

TRAINING PERFORMANCE - DATASET 7.5 ips, RUMORE - CLASSI eq. CCIR_CCIR / eq. CCIR_NAB / eq. NAB_CCIR / eq. NAB_NAB

CLASSIFIER	TRAINING_TIME (ms)	CV_ERROR	ACCUR	RECALL_1	SPECIF_1	PRECIS_1	AUC_1	RECALL_2	SPECIF_2	PRECIS_2	AUC_2	RECALL_3	SPECIF_3	PRECIS_3	AUC_3	RECALL_4	SPECIF_4	PRECIS_4	AUC_4
simpleTree	20,66	0,0878	0,9144	0,9067	0,9170	0,7846	0,9430	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,7511	0,9689	118,4211	0,9430
mediumTree	16,72	0,1178	0,9433	0,9111	0,9541	0,8686	0,9799	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,8622	0,9704	105,1402	0,9769
complexTree	19,32	0,1300	0,9744	0,9511	0,9822	0,9469	0,9879	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,9467	0,9837	100,4464	0,9879
linearSVM	1,72	0,1011	0,9144	0,8311	0,9422	0,8274	0,9650	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,8267	0,9437	100,4464	0,9652
quadrativSVM	4,61	0,1111	0,9600	0,9333	0,9689	0,9091	0,9871	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,9067	0,9778	102,7397	0,9876
cubicSVM	1,19	0,0611	0,9978	0,9956	0,9985	0,9956	0,9985	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,9956	0,9985	1,0000	0,9970
gaussianSVM	1,55	0,3111	0,9756	0,9600	0,9867	0,9600	0,9985	1,0000	0,9941	0,9825	1,0000	0,9822	1,0000	1,0000	1,0000	0,9600	0,9867	0,9822	0,9985
fineKNN	99,00	0,3778	0,9711	0,9733	0,9778	0,9359	0,9756	0,9689	1,0000	1,0000	0,9844	0,9822	1,0000	1,0000	0,9911	0,9600	0,9837	0,9736	0,9719
mediumKNN	10,24	0,3700	0,7700	0,8044	0,7911	0,5621	0,7922	0,9244	0,9867	0,9585	0,9966	0,9244	0,9956	0,9858	0,9979	0,4267	0,9200	138,6667	0,7982
coarseKNN	8,61	0,2289	0,8022	0,6889	0,8622	0,6250	0,8530	0,9556	0,9926	0,9773	0,9989	0,9467	0,9956	0,9861	0,9993	0,6178	0,8859	0,9861	0,8563
cosineKNN	10,95	0,3833	0,9678	0,9689	0,9763	0,9316	0,9726	0,9778	0,9970	0,9910	0,9874	0,9733	1,0000	1,0000	0,9867	0,9511	0,9837	0,9733	0,9674
weigthedKNN	933,00	0,3756	0,9678	0,9689	0,9689	0,9121	0,9689	0,9778	1,0000	1,0000	0,9889	0,9644	1,0000	1,0000	0,9822	0,9600	0,9881	0,9688	0,9741
	93,96	0,2213	0,9299																

CLASSIFICATION PERFORMANCE - DATASET 7.5 ips, RUMORE - CLASSI eq. CCIR_CCIR / eq. CCIR_NAB / eq. NAB_CCIR / eq. NAB_NAB

CLASSIFIER	TEST_TIME (ms)	TEST_ACCUR	TEST_RECALL_1	TEST_SPECIF_1	TEST_PRECIS_1	TEST_AUC_1	TEST_RECALL_2	TEST_SPECIF_2	TEST_PRECIS_2	TEST_AUC_2	TEST_RECALL_3	TEST_SPECIF_3	TEST_PRECIS_3	TEST_AUC_3	TEST_RECALL_4	TEST_SPECIF_4	TEST_PRECIS_4	TEST_AUC_4
simpleTree	0,53	0,9100	0,8133	0,9422	0,8243	0,9400	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,8267	0,9378	0,9868	0,9400
mediumTree	0,49	0,8967	0,7067	0,9600	0,8548	0,9460	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,8800	0,9022	0,8523	0,9371
complexTree	0,78	0,8867	0,6933	0,9511	0,8254	0,8612	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,8533	0,8978	0,8621	0,9012
linearSVM	23,12	0,9000	0,7600	0,9467	0,8261	0,9623	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,8400	0,9200	0,9259	0,9609
quadrativSVM	20,01	0,8900	0,6800	0,9600	0,8500	0,9636	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,8800	0,8933	0,8333	0,9647
cubicSVM	24,50	0,9733	0,9067	0,9956	0,9855	0,9946	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,9867	0,9689	0,9259	0,9905
gaussianSVM	18,66	0,9300	0,8800	0,9556	0,8684	0,9839	0,9867	0,9911	0,9737	0,9999	0,9733	1,0000	1,0000	1,0000	0,8800	0,9600	0,9733	0,9781
fineKNN	2,77	0,9367	0,9467	0,9467	0,8554	0,9467	0,9467	1,0000	1,0000	0,9733	0,9467	1,0000	1,0000	0,9733	0,9067	0,9689	0,9467	0,9378
mediumKNN	327,00	0,7667	0,8000	0,7867	0,5556	0,8403	0,9067	0,9911	0,9714	0,9911	0,9467	0,9956	0,9861	0,9989	0,4133	0,9156	1,0000	0,8417
coarseKNN	6,85	0,8267	0,7200	0,8844	0,6750	0,8558	0,9600	0,9956	0,9863	0,9993	0,9467	0,9956	0,9861	0,9994	0,6800	0,8933	0,9467	0,8481
cosineKNN	2,62	0,9167	0,9067	0,9600	0,8831	0,9333	0,9600	0,9911	0,9730	0,9756	0,9067	1,0000	1,0000	0,9533	0,8933	0,9378	0,8395	0,9156
weigthedKNN	2,44	0,9233	0,9333	0,9422	0,8434	0,9378	0,9600	1,0000	1,0000	0,9800	0,9067	1,0000	1,0000	0,9533	0,8933	0,9556	0,8831	0,9244
	35,81	0,8964																

TRAINING PERFORMANCE - DATASET 15 ips, SILENZIO - CLASSI correct eq. / wrong eq.							
CLASSIFIER	TRAINING_TIME (ms)	CV_ERROR	ACCUR	RECALL_1	SPECIF_1	PRECIS_1	AUC_1
simpleTree	13,77	0,3789	0,6489	0,7489	0,5489	0,6241	0,6627
mediumTree	19,07	0,3889	0,7278	0,6889	0,7667	0,7470	0,7788
complexTree	25,18	0,4233	0,8822	0,8822	0,8822	0,8822	0,9454
linearSVM	1,40	0,4878	0,5722	0,5711	0,5733	0,5724	0,5779
quadrativSVM	7,41	0,3744	0,7133	0,7289	0,6978	0,7069	0,7751
cubicSVM	41,17	0,4556	0,9522	0,9444	0,9600	0,9594	0,9436
gaussianSVM	432,91	0,4089	0,9622	0,9667	0,9578	0,9581	0,9959
fineKNN	1,07	0,4489	0,9622	0,9689	0,9556	0,9561	0,9622
mediumKNN	8,55	0,4156	0,6600	0,7422	0,5778	0,6374	0,7240
coarseKNN	8,52	0,4100	0,6267	0,6422	0,6111	0,6228	0,6840
cosineKNN	11,62	0,4622	0,9656	0,9689	0,9622	0,9625	0,9656
weigthedKNN	9,18	0,4389	0,9633	0,9689	0,9578	0,9582	0,9633
	48,32	0,4244	0,8031				

ACCURACY, RECALL, SPECIFICITY, PRECISION: sono calcolati a partire dalla matrice di confusione ottenuta classificando il training set (75% del dataset iniziale) con il miglior modello, scelto mediante k-fold Validation
AUC (Area Under Curve): area sotto la curva ROC, che ha lungo l'asse x il False Positive Rate (1 - Specificity), e lungo l'asse y il True Positive Rate (Sensitivity o Recall),
TRAINING TIME: espresso in millisecondi, tempo per addestrare il classificatore
TEST_SET_ACCURACY, TEST_SET_RECALL, TEST_SET_SPECIFICITY, TEST_SET_PRECISION, TEST_SET_AUC: sono calcolati a partire dalla matrice di confusione ottenuta classificando il test set (25% del dataset iniziale)
TEST TIME: in millisecondi, tempo che impiega il modello addestrato a classificare i campioni del test set

CLASSIFICATION PERFORMANCE - DATASET 15 ips, SILENZIO - CLASSI correct eq. / wrong eq.						
CLASSIFIER	TEST_TIME (ms)	TEST_ACCUR	TEST_RECALL_1	TEST_SPECIF_1	TEST_PRECIS_1	TEST_AUC_1
simpleTree	0,50	0,5267	0,6533	0,4000	0,5213	0,5289
mediumTree	0,52	0,5833	0,5133	0,6533	0,5969	0,6047
complexTree	0,71	0,5867	0,5133	0,6600	0,6016	0,5935
linearSVM	132,66	0,5033	0,4267	0,5800	0,5039	0,5076
quadrativSVM	137,66	0,5900	0,5733	0,6067	0,5931	0,6180
cubicSVM	137,67	0,5400	0,5533	0,5267	0,5390	0,5543
gaussianSVM	132,44	0,5633	0,5133	0,6133	0,5704	0,5869
fineKNN	2,48	0,5267	0,5200	0,5333	0,5270	0,5267
mediumKNN	3,42	0,5467	0,6000	0,4933	0,5422	0,5708
coarseKNN	5,71	0,5433	0,5067	0,5800	0,5468	0,5599
cosineKNN	2,44	0,5100	0,4800	0,5400	0,5106	0,5100
weigthedKNN	2,51	0,5167	0,4867	0,5467	0,5177	0,5167
	46,56	0,5447				

TRAINING PERFORMANCE - DATASET 15 ips, SILENZIO - CLASSI correct eq. / eq. CCIR_NAB / eq. NAB_CCIR															
CLASSIFIER	TRAINING_TIME (ms)	CV_ERROR	ACCUR	RECALL_1	SPECIF_1	PRECIS_1	AUC_1	RECALL_2	SPECIF_2	PRECIS_2	AUC_2	RECALL_3	SPECIF_3	PRECIS_3	AUC_3
simpleTree	16,52	0,3656	0,6611	0,5867	0,7467	0,6984	0,6794	0,6178	0,8889	0,6495	0,8739	0,8533	0,8281	0,6234	0,8889
mediumTree	17,37	0,3722	0,7478	0,7578	0,7422	0,7462	0,8058	0,7511	0,8978	0,7101	0,9110	0,7244	0,9378	0,7951	0,9427
complexTree	23,84	0,4567	0,8900	0,8978	0,8822	0,8840	0,9433	0,8622	0,9733	0,9151	0,9777	0,9022	0,9585	0,8788	0,9756
linearSVM	1,80	0,3656	0,6789	0,7178	0,6400	0,6660	0,7288	0,6444	0,8978	0,6776	0,9043	0,6356	0,9141	0,7114	0,9176
quadrativSVM	23,27	0,4267	0,7456	0,8778	0,6133	0,6942	0,7838	0,6533	0,9467	0,8033	0,9399	0,5733	0,9719	0,8716	0,9416
cubicSVM	4,62	0,5011	0,5456	0,9956	0,0956	0,5240	0,6639	0,1911	0,9970	0,9556	0,9859	0,0000	1,0000	1,0000	0,8467
gaussianSVM	892,39	0,4311	0,9644	0,9844	0,9444	0,9466	0,9968	0,9600	0,9941	0,9818	0,9990	0,9289	0,9956	0,9858	0,9988
fineKNN	8,95	0,4533	0,9611	0,9644	0,9578	0,9581	0,9611	0,9644	0,9852	0,9559	0,9748	0,9511	0,9911	0,9727	0,9711
mediumKNN	8,39	0,4344	0,6389	0,6222	0,6578	0,6452	0,7103	0,8000	0,8119	0,5863	0,8972	0,5111	0,9348	0,7233	0,9091
coarseKNN	8,33	0,3978	0,6244	0,7000	0,5489	0,6081	0,6831	0,5822	0,8607	0,5822	0,8847	0,5156	0,9393	0,7389	0,9142
cosineKNN	9,58	0,4656	0,9644	0,9556	0,9733	0,9729	0,9644	0,9689	0,9837	0,9520	0,9763	0,9778	0,9867	0,9607	0,9822
weigthedKNN	9,18	0,4356	0,9656	0,9600	0,9711	0,9708	0,9656	0,9822	0,9896	0,9693	0,9859	0,9600	0,9837	0,9515	0,9719
	85,35	0,4255	0,7823												

CLASSIFICATION PERFORMANCE - DATASET 15 ips, SILENZIO - CLASSI correct eq. / eq. CCIR_NAB / eq. NAB_CCIR															
CLASSIFIER	TEST_TIME (ms)	TEST_ACCUR	TEST_RECALL_1	TEST_SPECIF_1	TEST_PRECIS_1	TEST_AUC_1	TEST_RECALL_2	TEST_SPECIF_2	TEST_PRECIS_2	TEST_AUC_2	TEST_RECALL_3	TEST_SPECIF_3	TEST_PRECIS_3	TEST_AUC_3
simpleTree	0,45	0,5433	0,4533	0,6400	0,5574	0,5489	0,4800	0,8533	0,5217	0,8356	0,7867	0,7778	0,5413	0,8529
mediumTree	0,50	0,5967	0,5733	0,6200	0,6014	0,6202	0,7200	0,8622	0,6353	0,8967	0,5200	0,8533	0,5417	0,8549
complexTree	1,03	0,5933	0,5800	0,6067	0,5959	0,6138	0,6267	0,8667	0,6104	0,8142	0,5867	0,8533	0,5714	0,8079
linearSVM	466,52	0,6233	0,6067	0,6400	0,6276	0,6652	0,7467	0,8489	0,6222	0,8992	0,5333	0,8889	0,6154	0,8855
quadrativSVM	544,92	0,5433	0,6000	0,4867	0,5389	0,5692	0,6400	0,8578	0,6000	0,8932	0,3333	0,8756	0,4717	0,8412
cubicSVM	440,63	0,5067	0,9267	0,0867	0,5036	0,4888	0,1733	0,9511	0,5417	0,8392	0,0000	1,0000	1,0000	0,8183
gaussianSVM	539,04	0,5700	0,7667	0,3733	0,5502	0,5935	0,4133	0,9289	0,6596	0,8617	0,3333	0,9156	0,5682	0,8605
fineKNN	2,57	0,5333	0,5267	0,5400	0,5338	0,5333	0,5867	0,8089	0,5057	0,6978	0,4933	0,8756	0,5692	0,6844
mediumKNN	3,50	0,5500	0,4800	0,6200	0,5581	0,5632	0,7600	0,7822	0,5377	0,8472	0,4800	0,8711	0,5538	0,8601
coarseKNN	6,62	0,5967	0,6200	0,5733	0,5924	0,5844	0,6667	0,8578	0,6098	0,8854	0,4800	0,8889	0,5902	0,8668
cosineKNN	3,08	0,4867	0,4467	0,5267	0,4855	0,4867	0,5200	0,7867	0,4483	0,6533	0,5333	0,8444	0,5333	0,6889
weigthedKNN	2,72	0,5300	0,5267	0,5333	0,5302	0,5300	0,5467	0,8222	0,5062	0,6844	0,5200	0,8622	0,5571	0,6911
	167,63	0,5561												

TRAINING PERFORMANCE - DATASET 15 ips, SILENZIO - CLASSI post eq. NAB / post eq. CCIR							
CLASSIFIER	TRAINING_TIME (ms)	CV_ERROR	ACCUR	RECALL_1	SPECIF_1	PRECIS_1	AUC_1
simpleTree	10,25	0,0056	0,9989	1,0000	0,9978	0,9978	1,0000
mediumTree	9,75	0,0067	0,9989	1,0000	0,9978	0,9978	1,0000
complexTree	958,00	0,0056	0,9989	1,0000	0,9978	0,9978	1,0000
linearSVM	274,56	0,0022	1,0000	1,0000	1,0000	1,0000	1,0000
quadrativSVM	64,88	0,0067	1,0000	1,0000	1,0000	1,0000	1,0000
cubicSVM	226,09	0,0022	1,0000	1,0000	1,0000	1,0000	1,0000
gaussianSVM	492,83	0,0178	1,0000	1,0000	1,0000	1,0000	1,0000
fineKNN	10,84	0,0100	1,0000	1,0000	1,0000	1,0000	1,0000
mediumKNN	8,82	0,0111	0,9900	0,9867	0,9933	0,9933	0,9997
coarseKNN	9,18	0,0144	0,9867	1,0000	0,9733	0,9740	1,0000
cosineKNN	11,46	0,0078	1,0000	1,0000	1,0000	1,0000	1,0000
weigthedKNN	9,76	0,0100	1,0000	1,0000	1,0000	1,0000	1,0000
	173,87	0,0083	0,9978				

CLASSIFICATION PERFORMANCE - DATASET 15 ips, SILENZIO - CLASSI post eq. NAB / post eq. CCIR						
CLASSIFIER	TEST_TIME (ms)	TEST_ACCUR	TEST_RECALL_1	TEST_SPECIF_1	TEST_PRECIS_1	TEST_AUC_1
simpleTree	0,56	0,9867	0,9867	0,9867	0,9867	0,9899
mediumTree	0,44	0,9867	0,9867	0,9867	0,9867	0,9899
complexTree	0,53	0,9867	0,9867	0,9867	0,9867	0,9899
linearSVM	131,50	1,0000	1,0000	1,0000	1,0000	1,0000
quadrativSVM	130,09	1,0000	1,0000	1,0000	1,0000	1,0000
cubicSVM	131,10	0,9967	1,0000	0,9933	0,9934	0,9934
gaussianSVM	13,48	0,9967	1,0000	0,9933	0,9934	1,0000
fineKNN	2,42	1,0000	1,0000	1,0000	1,0000	1,0000
mediumKNN	3,30	0,9933	1,0000	0,9867	0,9868	0,9999
coarseKNN	7,20	0,9833	1,0000	0,9667	0,9677	1,0000
cosineKNN	2,52	0,9967	0,9933	1,0000	1,0000	0,9967
weigthedKNN	2,41	1,0000	1,0000	1,0000	1,0000	1,0000
	35,46	0,9939				

TRAINING PERFORMANCE - DATASET 15 ips, SILENZIO - CLASSI eq. CCIR_CCIR / eq. CCIR_NAB / eq. NAB_CCIR / eq. NAB_NAB																			
CLASSIFIER	TRAINING_TIME (ms)	CV_ERROR	ACCUR	RECALL_1	SPECIF_1	PRECIS_1	AUC_1	RECALL_2	SPECIF_2	PRECIS_2	AUC_2	RECALL_3	SPECIF_3	PRECIS_3	AUC_3	RECALL_4	SPECIF_4	PRECIS_4	AUC_4
simpleTree	20,37	0,37556	0,66111	0,68000	0,89778	0,68919	0,89295	0,61778	0,88889	0,64953	0,87317	0,70222	0,88000	0,66109	0,88794	0,64444	0,88148	0,70222	0,87540
mediumTree	1,82	0,39667	0,72444	0,56444	0,95259	0,79874	0,92302	0,72444	0,91852	0,74771	0,92350	0,86222	0,85037	0,65763	0,92149	0,74667	0,91111	0,85088	0,92444
complexTree	21,96	0,41444	0,88111	0,92444	0,94815	0,85597	0,98127	0,92000	0,94370	0,84490	0,96825	0,85778	0,97333	0,91469	0,98361	0,82222	0,97630	0,96020	0,96574
linearSVM	1,67	0,34778	0,67222	0,68889	0,90370	0,70455	0,92232	0,64889	0,88000	0,64317	0,90447	0,71111	0,89630	0,69565	0,92182	0,64000	0,88296	0,71749	0,90453
quadrativSVM	22,78	0,40889	0,77889	0,80444	0,92296	0,77682	0,94113	0,76889	0,92444	0,77232	0,94166	0,76889	0,93481	0,79724	0,93664	0,77333	0,92296	0,7655	0,94517
cubicSVM	7,78	0,50667	0,55778	0,14222	0,99852	0,96970	0,84018	0,99556	0,69926	0,52459	0,83666	0,99556	0,71407	0,53717	0,86133	0,09778	0,99852	0,9739	0,81523
gaussianSVM	1,39	0,40444	0,97222	0,96444	0,99407	0,98190	0,99920	0,97333	0,98963	0,96903	0,99904	0,98222	0,98815	0,96507	0,99901	0,96889	0,99111	0,9866	0,99896
fineKNN	9,13	0,46111	0,96000	0,96444	0,98667	0,96018	0,97556	0,95556	0,98667	0,95982	0,97111	0,96000	0,98815	0,96429	0,97407	0,96000	0,98519	0,9558	0,97259
mediumKNN	9,11	0,43556	0,63222	0,77778	0,84889	0,63177	0,91488	0,80000	0,80000	0,57143	0,89639	0,54222	0,93185	0,72619	0,91622	0,40889	0,92889	0,8714	0,89411
coarseKNN	9,48	0,40444	0,62000	0,74222	0,87111	0,65748	0,90022	0,76889	0,79407	0,55449	0,88816	0,61778	0,89481	0,66190	0,91073	0,35111	0,93333	0,1121	0,85123
cosineKNN	11,34	0,45111	0,96111	0,96889	0,98963	0,96889	0,97926	0,95111	0,98667	0,95964	0,96889	0,96889	0,98815	0,96460	0,97852	0,95556	0,98370	0,9646	0,96963
weigthedKNN	9,62	0,44444	0,96111	0,96889	0,98222	0,94783	0,97556	0,97778	0,98519	0,95652	0,98148	0,94667	0,98815	0,96380	0,96741	0,95111	0,99259	0,97260	0,97185
	10,54	0,42093	0,78185																

CLASSIFICATION PERFORMANCE - DATASET 15 ips, SILENZIO - CLASSI eq. CCIR_CCIR / eq. CCIR_NAB / eq. NAB_CCIR / eq. NAB_NAB																			
CLASSIFIER	TEST_TIME (ms)	TEST_ACCUR	TEST_RECALL_1	TEST_SPECIF_1	TEST_PRECIS_1	TEST_AUC_1	TEST_RECALL_2	TEST_SPECIF_2	TEST_PRECIS_2	TEST_AUC_2	TEST_RECALL_3	TEST_SPECIF_3	TEST_PRECIS_3	TEST_AUC_3	TEST_RECALL_4	TEST_SPECIF_4	TEST_PRECIS_4	TEST_AUC_4
simpleTree	0,43	0,52667	0,56000	0,84444	0,54545	0,83470	0,48000	0,85333	0,52174	0,83659	0,53333	0,84889	0,54054	0,84563	0,53333	0,82222	0,50000	0,82590
mediumTree	0,79	0,60333	0,38667	0,93778	0,67442	0,85627	0,66667	0,85333	0,60241	0,88133	0,81333	0,79556	0,57009	0,87194	0,54667	0,88444	0,91045	0,85864
complexTree	0,73	0,57000	0,64000	0,85333	0,59259	0,83473	0,64000	0,81333	0,53333	0,78409	0,57333	0,88889	0,63235	0,84794	0,42667	0,87111	0,70492	0,78874
linearSVM	15,59	0,62333	0,53333	0,90222	0,64516	0,87532	0,74667	0,83556	0,60215	0,89529	0,70667	0,84444	0,60227	0,87716	0,50667	0,91556	0,92982	0,89446
quadrativSVM	5,10	0,57333	0,50667	0,84444	0,52055	0,83828	0,72000	0,84444	0,60674	0,90305	0,53333	0,83556	0,51948	0,86086	0,53333	0,90667	0,6557	0,89286
cubicSVM	2,01	0,51333	0,13333	0,97333	0,62500	0,83087	0,93333	0,68889	0,50000	0,79413	0,93333	0,71111	0,51852	0,86228	0,05333	0,97778	0,7778	0,76942
gaussianSVM	15,46	0,56000	0,42667	0,91556	0,62745	0,86406	0,50667	0,85778	0,54286	0,85641	0,73333	0,81333	0,56701	0,85742	0,57333	0,82667	0,6707	0,86086
fineKNN	2,86	0,51667	0,57333	0,82667	0,52439	0,70000	0,54667	0,82222	0,50617	0,68444	0,48000	0,85778	0,52941	0,66889	0,46667	0,84889	0,5217	0,65778
mediumKNN	4,32	0,54000	0,62667	0,83556	0,55952	0,86833	0,72000	0,77333	0,51429	0,84859	0,50667	0,87111	0,56716	0,86489	0,30667	0,90667	0,8636	0,83876
coarseKNN	6,76	0,54333	0,58667	0,85778	0,57895	0,85167	0,70667	0,77778	0,51456	0,88201	0,57333	0,84000	0,54430	0,86607	0,30667	0,91556	0,1024	0,82761
cosineKNN	3,13	0,50667	0,54667	0,85778	0,56164	0,70222	0,49333	0,80000	0,45122	0,64667	0,57333	0,85333	0,56579	0,71333	0,41333	0,83111	0,6232	0,62222
weigthedKNN	2,43	0,52667	0,57333	0,83111	0,53086	0,70222	0,61333	0,80889	0,51685	0,71111	0,49333	0,85778	0,53623	0,67556	0,42667	0,87111	0,60656	0,64889
	4,97	0,55028																

TRAINING PERFORMANCE - DATASET 15 ips, RUMORE - CLASSI correct eq./wrong eq.							
CLASSIFIER	TRAINING_TIME (ms)	CV_ERROR	ACCUR	RECALL_1	SPECIF_1	PRECIS_1	AUC_1
simpleTree	10,82	0,0000	1,0000	1,0000	1,0000	1,0000	1,0000
mediumTree	11,22	0,0000	1,0000	1,0000	1,0000	1,0000	1,0000
complexTree	11,70	0,0000	1,0000	1,0000	1,0000	1,0000	1,0000
linearSVM	902,02	0,3889	0,5033	0,6000	0,4067	0,5028	0,4991
quadrativSVM	604,14	0,0000	1,0000	1,0000	1,0000	1,0000	1,0000
cubicSVM	1,12	0,0033	1,0000	1,0000	1,0000	1,0000	1,0000
gaussianSVM	593,33	0,0233	1,0000	1,0000	1,0000	1,0000	1,0000
fineKNN	9,32	0,1611	0,9889	1,0000	0,9778	0,9783	0,9889
mediumKNN	9,67	0,1122	0,9400	0,9978	0,8822	0,8944	0,9929
coarseKNN	8,22	0,1689	0,8378	1,0000	0,6756	0,7550	0,9996
cosineKNN	11,12	0,1533	0,9889	0,9956	0,9822	0,9825	0,9889
weigthedKNN	11,26	0,1744	0,9889	1,0000	0,9778	0,9783	0,9889
	182,00	0,0988	0,9373				

ACCURACY, RECALL, SPECIFICITY, PRECISION: sono calcolati a partire dalla matrice di confusione ottenuta classificando il training set (75% del dataset iniziale) con il miglior modello, scelto mediante k-fold Validation
AUC (Area Under Curve): area sotto la curva ROC, che ha lungo l'asse x il False Positive Rate (1 - Specificity), e lungo l'asse y il True Positive Rate (Sensitivity o Recall),
TRAINING TIME: espresso in millisecondi, tempo per addestrare il classificatore
TEST_SET_ACCURACY, TEST_SET_RECALL, TEST_SET_SPECIFICITY, TEST_SET_PRECISION, TEST_SET_AUC: sono calcolati a partire dalla matrice di confusione ottenuta classificando il test set (25% del dataset iniziale)
TEST TIME: in millisecondi, tempo che impiega il modello addestrato a classificare i campioni del test set

CLASSIFICATION PERFORMANCE - DATASET 15 ips, RUMORE - CLASSI correct eq./wrong eq.						
CLASSIFIER	TEST_TIME (ms)	TEST_ACCUR	TEST_RECALL_1	TEST_SPECIF_1	TEST_PRECIS_1	TEST_AUC_1
simpleTree	0,41	1,0000	1,0000	1,0000	1,0000	1,0000
mediumTree	0,40	1,0000	1,0000	1,0000	1,0000	1,0000
complexTree	0,39	1,0000	1,0000	1,0000	1,0000	1,0000
linearSVM	151,63	0,5033	0,5933	0,4133	0,5028	0,4984
quadrativSVM	141,94	1,0000	1,0000	1,0000	1,0000	1,0000
cubicSVM	136,40	1,0000	1,0000	1,0000	1,0000	1,0000
gaussianSVM	145,15	1,0000	1,0000	1,0000	1,0000	1,0000
fineKNN	2,80	0,9733	0,9933	0,9533	0,9551	0,9733
mediumKNN	3,23	0,9467	1,0000	0,8933	0,9036	0,9991
coarseKNN	5,43	0,8500	1,0000	0,7000	0,7692	0,9995
cosineKNN	2,75	0,9600	0,9867	0,9333	0,9367	0,9600
weigthedKNN	2,55	0,9667	1,0000	0,9333	0,9375	0,9667
	49,42	0,9333				

TRAINING PERFORMANCE - DATASET 15 ips, RUMORE - CLASSI correct eq. / eq. CCIR_NAB / eq. NAB_CCIR															
CLASSIFIER	TRAINING_TIME (ms)	CV_ERROR	ACCUR	RECALL_1	SPECIF_1	PRECIS_1	AUC_1	RECALL_2	SPECIF_2	PRECIS_2	AUC_2	RECALL_3	SPECIF_3	PRECIS_3	AUC_3
simpleTree	13,10	0,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
mediumTree	13,23	0,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
complexTree	13,99	0,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
linearSVM	629,10	0,0011	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
quadrativSVM	868,47	0,0022	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
cubicSVM	582,16	0,0022	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
gaussianSVM	935,91	0,0544	0,9967	1,0000	1,0000	1,0000	1,0000	0,9867	1,0000	1,0000	1,0000	1,0000	0,9956	0,9868	1,0000
fineKNN	9,21	0,1700	0,9867	1,0000	0,9733	0,9740	0,9867	0,9600	1,0000	1,0000	0,9800	0,9867	1,0000	1,0000	0,9933
mediumKNN	8,38	0,0689	0,9578	1,0000	0,9156	0,9221	0,9869	0,9689	1,0000	1,0000	0,9999	0,8622	1,0000	1,0000	1,0000
coarseKNN	8,43	0,1678	0,8311	1,0000	0,6622	0,7475	0,9994	0,7022	1,0000	1,0000	1,0000	0,6222	1,0000	1,0000	1,0000
cosineKNN	9,51	0,1444	0,9900	1,0000	0,9800	0,9804	0,9900	0,9778	1,0000	1,0000	0,9889	0,9822	1,0000	1,0000	0,9911
weigthedKNN	8,97	0,1656	0,9900	0,9956	0,9844	0,9846	0,9900	0,9822	0,9985	0,9955	0,9904	0,9867	0,9985	0,9955	0,9926
	258,37	0,0647	0,9794												

CLASSIFICATION PERFORMANCE - DATASET 15 ips, RUMORE - CLASSI correct eq. / eq. CCIR_NAB / eq. NAB_CCIR															
CLASSIFIER	TEST_TIME (ms)	TEST_ACCUR	TEST_RECALL_1	TEST_SPECIF_1	TEST_PRECIS_1	TEST_AUC_1	TEST_RECALL_2	TEST_SPECIF_2	TEST_PRECIS_2	TEST_AUC_2	TEST_RECALL_3	TEST_SPECIF_3	TEST_PRECIS_3	TEST_AUC_3
simpleTree	0,51	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
mediumTree	0,50	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
complexTree	0,55	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
linearSVM	19,15	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
quadrativSVM	18,84	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
cubicSVM	19,06	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
gaussianSVM	16,92	0,9900	1,0000	1,0000	1,0000	1,0000	0,9600	1,0000	1,0000	1,0000	1,0000	0,9867	0,9615	1,0000
fineKNN	2,46	0,9700	1,0000	0,9400	0,9434	0,9700	0,9467	1,0000	1,0000	0,9733	0,9333	1,0000	1,0000	0,9667
mediumKNN	333,00	0,9567	1,0000	0,9133	0,9202	0,9966	1,0000	1,0000	1,0000	1,0000	0,8267	1,0000	1,0000	1,0000
coarseKNN	5,66	0,8433	1,0000	0,6867	0,7614	0,9992	0,7867	1,0000	1,0000	1,0000	0,5867	1,0000	1,0000	1,0000
cosineKNN	2,69	0,9767	1,0000	0,9533	0,9554	0,9767	0,9733	1,0000	1,0000	0,9867	0,9333	1,0000	1,0000	0,9667
weigthedKNN	2,42	0,9633	0,9867	0,9400	0,9427	0,9633	0,9467	0,9911	0,9726	0,9689	0,9333	1,0000	1,0000	0,9667
	35,15	0,9750												

TRAINING PERFORMANCE - DATASET 15 ips, RUMORE - CLASSI post eq. NAB / post eq. CCIR							
CLASSIFIER	TRAINING_TIME (ms)	CV_ERROR	ACCUR	RECALL_1	SPECIF_1	PRECIS_1	AUC_1
simpleTree	12,03	0,2211	0,8022	0,6844	0,9200	0,8953	0,9021
mediumTree	14,20	0,2578	0,8511	0,7822	0,9200	0,9072	0,9322
complexTree	19,13	0,2611	0,9322	0,9289	0,9356	0,9351	0,9697
linearSVM	57,44	0,2111	0,8056	0,7978	0,8133	0,8104	0,9155
quadrativSVM	7,75	0,2200	0,8833	0,8800	0,8867	0,8859	0,9534
cubicSVM	1,03	0,1622	0,9900	0,9933	0,9867	0,9868	0,9943
gaussianSVM	411,81	0,5500	0,0744	0,0622	0,0867	0,0638	0,0526
fineKNN	9,07	0,2856	0,9789	0,9756	0,9822	0,9821	0,9789
mediumKNN	8,58	0,3078	0,7867	0,9200	0,6533	0,7263	0,9025
coarseKNN	8,42	0,2122	0,8056	0,8000	0,8111	0,8090	0,9071
cosineKNN	9,62	0,2744	0,9767	0,9778	0,9756	0,9756	0,9767
weigthedKNN	894,00	0,2733	0,9833	0,9822	0,9844	0,9844	0,9833
	121,09	0,2697	0,8225				

CLASSIFICATION PERFORMANCE - DATASET 15 ips, RUMORE - CLASSI post eq. NAB / post eq. CCIR						
CLASSIFIER	TEST_TIME (ms)	TEST_ACCUR	TEST_RECALL_1	TEST_SPECIF_1	TEST_PRECIS_1	TEST_AUC_1
simpleTree	0,54	0,7633	0,6133	0,9133	0,8762	0,8817
mediumTree	0,43	0,7767	0,6600	0,8933	0,8609	0,8844
complexTree	0,53	0,7867	0,7267	0,8467	0,8258	0,8447
linearSVM	132,02	0,7767	0,7067	0,8467	0,8217	0,9052
quadrativSVM	130,48	0,8200	0,7133	0,9267	0,9068	0,9238
cubicSVM	132,48	0,8867	0,8200	0,9533	0,9462	0,9560
gaussianSVM	12,74	0,1333	0,1000	0,1667	0,1071	0,0625
fineKNN	2,34	0,8567	0,7533	0,9600	0,9496	0,8567
mediumKNN	3,24	0,7833	0,9000	0,6667	0,7297	0,8927
coarseKNN	6,69	0,7800	0,7200	0,8400	0,8182	0,9009
cosineKNN	3,17	0,8533	0,7600	0,9467	0,9344	0,8533
weigthedKNN	2,32	0,8733	0,7733	0,9733	0,9667	0,8733
	35,58	0,7575				

TRAINING PERFORMANCE - DATASET 15 ips, RUMORE - CLASSI eq. CCIR_CCIR / eq. CCIR_NAB / eq. NAB_CCIR / eq. NAB_NAB																			
CLASSIFIER	TRAINING_TIME (ms)	CV_ERROR	ACCUR	RECALL_1	SPECIF_1	PRECIS_1	AUC_1	RECALL_2	SPECIF_2	PRECIS_2	AUC_2	RECALL_3	SPECIF_3	PRECIS_3	AUC_3	RECALL_4	SPECIF_4	PRECIS_4	AUC_4
simpleTree	13,33	0,2078	0,8089	0,6222	0,8711	0,6167	0,8775	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,6133	0,8741	0,9000	0,8775
mediumTree	17,18	0,2600	0,8633	0,7022	0,9170	0,7383	0,9279	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,7511	0,9007	0,9534	0,9265
complexTree	2,21	0,2656	0,9444	0,8889	0,9630	0,8889	0,9760	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,8889	0,9630	1,0000	0,9671
linearSVM	1,50	0,2111	0,8067	0,6178	0,8696	0,6123	0,8882	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,6089	0,8726	0,9000	0,8883
quadrativSVM	8,02	0,2378	0,8900	0,7556	0,9348	0,7944	0,9403	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,8044	0,9185	0,9534	0,9403
cubicSVM	1,64	0,2389	0,9833	0,9556	0,9926	0,9773	0,9875	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,9778	0,9852	0,9783	0,9875
gaussianSVM	139,06	0,1022	0,5900	0,2178	0,7156	0,2033	0,7168	0,9956	1,0000	1,0000	1,0000	1,0000	0,9985	0,9956	1,0000	0,1467	0,7393	0,9500	0,7147
fineKNN	8,93	0,4211	0,9656	0,9556	0,9748	0,9267	0,9652	0,9911	0,9970	0,9911	0,9941	0,9689	0,9985	0,9954	0,9837	0,9467	0,9837	0,9732	0,9652
mediumKNN	854,00	0,3567	0,7600	0,3778	0,8948	0,5449	0,8122	0,9822	1,0000	1,0000	0,9999	0,9778	1,0000	1,0000	1,0000	0,7022	0,7852	0,7261	0,7941
coarseKNN	8,25	0,2778	0,7800	0,5200	0,8681	0,5680	0,8468	1,0000	1,0000	1,0000	1,0000	1,0000	0,9970	0,9912	1,0000	0,6000	0,8415	0,9298	0,8406
cosineKNN	9,92	0,4344	0,9644	0,9467	0,9748	0,9261	0,9607	0,9867	1,0000	1,0000	0,9933	0,9733	1,0000	1,0000	0,9867	0,9511	0,9778	0,9563	0,9644
weigthedKNN	9,68	0,4456	0,9611	0,9422	0,9778	0,9339	0,9600	0,9689	1,0000	1,0000	0,9844	0,9733	1,0000	1,0000	0,9867	0,9600	0,9704	0,9280	0,9652
	89,48	0,2882	0,8598																

CLASSIFICATION PERFORMANCE - DATASET 15 ips, RUMORE - CLASSI eq. CCIR_CCIR / eq. CCIR_NAB / eq. NAB_CCIR / eq. NAB_NAB																			
CLASSIFIER	TEST_TIME (ms)	TEST_ACCUR	TEST_RECALL_1	TEST_SPECIF_1	TEST_PRECIS_1	TEST_AUC_1	TEST_RECALL_2	TEST_SPECIF_2	TEST_PRECIS_2	TEST_AUC_2	TEST_RECALL_3	TEST_SPECIF_3	TEST_PRECIS_3	TEST_AUC_3	TEST_RECALL_4	TEST_SPECIF_4	TEST_PRECIS_4	TEST_AUC_4
simpleTree	0,4495	0,7633	0,3733	0,8933	0,5385	0,8435	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,6800	0,7911	0,7653	0,8435
mediumTree	0,5552	0,7933	0,4267	0,9156	0,6275	0,8738	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,7467	0,8089	0,7576	0,8738
complexTree	0,5525	0,8100	0,5467	0,8978	0,6406	0,8011	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,6933	0,8489	0,8721	0,8011
linearSVM	14,3031	0,7867	0,4267	0,9067	0,6038	0,8734	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,7200	0,8089	0,7732	0,8734
quadrativSVM	14,7851	0,8133	0,3467	0,9689	0,7879	0,8917	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,9067	0,7822	0,6410	0,8917
cubicSVM	13,6566	0,8367	0,3733	0,9911	0,9333	0,9285	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	0,9733	0,7911	0,6250	0,9283
gaussianSVM	17,9157	0,7167	0,7333	0,7156	0,4622	0,7626	0,9867	1,0000	1,0000	1,0000	1,0000	0,9956	0,9868	1,0000	0,1467	0,9111	0,2410	0,7257
fineKNN	2,5505	0,8200	0,5200	0,9378	0,7358	0,7289	0,9733	0,9867	0,9605	0,9800	0,9333	0,9956	0,9859	0,9644	0,8533	0,8400	0,7000	0,8467
mediumKNN	3,7007	0,7600	0,3200	0,9067	0,5333	0,8215	1,0000	1,0000	1,0000	1,0000	0,9600	1,0000	1,0000	1,0000	0,7600	0,7733	0,6667	0,8271
coarseKNN	5,8337	0,7533	0,4133	0,8711	0,5167	0,8439	1,0000	0,9956	0,9868	1,0000	1,0000	1,0000	1,0000	1,0000	0,6000	0,8044	0,8427	0,8336
cosineKNN	2,5238	0,8267	0,5200	0,9422	0,7500	0,7311	0,9600	1,0000	1,0000	0,9800	0,9200	1,0000	1,0000	0,9600	0,9067	0,8267	0,6449	0,8667
weigthedKNN	2,4842	0,8333	0,5733	0,9422	0,7679	0,7578	0,9333	1,0000	1,0000	0,9667	0,9200	1,0000	1,0000	0,9600	0,9067	0,8356	0,6571	0,8711
	6,6092	0,7928																