UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea triennale in Ingegneria Elettronica

TECNICHE DI MACHINE LEARNING APPLICATE ALLA CLASSIFICAZIONE DI SOGGETTI SANI E SOGGETTI AFFETTI DA ALZHEIMER TRAMITE L'ELABORAZIONE DI SEGNALI ELETTROENCEFALOGRAFICI (EEG).

Relatore: Chiar.mo

Prof. *Claudio Turchetti*

Correlatore: Chiar.ma

Prof.ssa *Laura Falaschetti*

Tesi di Laurea di:

Valeria Vetrano

A.A 2019/2020

INDICE

INTRODUZIONE

Nei tempi odierni l'uso dell'intelligenza artificiale ricopre un ruolo di grande importanza grazie alla molteplici funzionalità che essa possiede e la maturità tecnologica ha fatto si che uscisse dall'alveo dell ricerca per entrare, di fatto, nella vita quotidiana.

L'intellgenza artificiale è definibile come misura di efficacia del comportamento umano, attraverso la generazione di modelli attui a garantire delle scelte che possano ridurre al minimo l'intervento dell'uomo, considerata quasi come fosse un "cervello ausiliario".

Gli ambiti che essa ricopre spaziano tra i più svariati settori, come ad esempio quello del marketing, della sanità, del cybercrime oppure della sicurezza pubblica, questi appena citati rappresentano una piccola parte fra tutte le applicazioni oggi note.

In particolare, l'intelligenta artificiale è, sempre più spesso, applicata in ambito biomedicale, per i cosidetti sistemi *CAD (Computer-aided diagnosis)*[1], permettendo di effettuare una diagnosi precoce sulla base dei dati. Questi inizialmente captati con strumentazione apposita sotto forma di segnali e successivamente convertiti e trasformati in veri e propri database, sono fondamentali affinchè si possano "addestrare" modelli di intelligenza artificiale, attraverso le tecniche di machine learning[2], per l'identificazione di eventuali malattie.

Il lavoro illustrato nel seguito espone l'analisi e la conversione in un grande dataset dei segnali elettroencefalografici (EEG), captati attraversi gli elettrodi posti sul capo dei vari pazienti.

L'obiettivo di questa tesi è di analizzare alcune delle tecniche di machine learning, relative alla task di classificazione, per poter individuare l'approccio migliore nella

[1] La diagnosi assistita da computer *(CAD)* è l'uso di un output generato dal computer come strumento di supporto per un medico per fare una diagnosi. È diverso dalla diagnosi computerizzata automatizzata, in cui la diagnosi finale si basa solo su un algoritmo informatico.

[2] Apprendimento automatico, approfondito nella successiva trattazione.

classificazione dei soggetti sani e malati di Alzheimer, tramite l'elaborazione dei segnali elettroencefalografici.

Struttura tesi:

- Descrizione delle modalità di impiego della machine learning, in particolare sulla classificazione e sulle tecniche di feature extraction.
- Analisi dei segnali trattati (EEG) con conseguente creazione dei datasets di grandi dimensioni, data_matrix e data_matrix_NEW.
- Valutazione delle percentuali di accuracy confrontate con i vari algoritmi, attraverso l'ambiente di calcolo MATLAB.
- Osservazioni riguardo ai datasets trattati, rispetto ai risultati ottenuti.

CAPITOLO 1

MACHINE LEARNING

Il significato di *machine learning*, che tranducendolo in italiano significa apprendimento automatico e rappresenta una branca dell'intelligenza artificiale, può essere sintetizzato citando una celebre frase del prof. Tom Mitchell, della Carnegy Mellon university:

"Si dice che un programma impara da una certa esperienza E rispetto a una classe di compiti T ottenendo una performance P, se la sua perfomance nel realizzare i compiti T, misurata dalla performance P, migliora con l'esperienza E."[3]

Attraverso questa affermazione ciò che si vuole evidenziare è di come un computer possa essere "addestrato" a migliorare lo svolgimento di un compito rispetto ad un'esperienza precedente.
Quel che è interessante è chiederci come sia possibile costruire dei sistemi informatici che in maniera autonoma possano migliorare con l'esperienza e quali possono essere le leggi che governano tutti i processi di apprendimento.
Secondo Mitchell, come scritto nel *The Discipline of Machine Learning,* una risposta a tale quesito può inglobare diversi rami come, ad esempio, estrarre i dati dalle cartelle cliniche per verificare come i futuri pazienti risponderanno ai trattamenti oppure progettare robot odierni rendendoli autonomi e molto altro [1] .
L'apprendimento automatico (*machine learning*) sussiste grazie ad un vasto numero di dati (f*eature*), diversificati in base ai settori scelti dall'utente, che vengono forniti alla macchina affinché possa migliorare ed essere un vero è proprio ausilio alla mente umana, grazie ad algoritmi rigorosamente definiti.
Sostanzialmente, gli algoritmi di Machine Learning usano metodi matematico-computazionali per apprendere informazioni direttamente dai dati, senza modelli

[3] Tom M. Mitchell, *The Discipline of Machine Learning*, Machine Learning Department School of Computer Science Carnegie Mellon University Pittsburgh, PA 15213, July 2006.

matematici ed equazioni predeterminate, migliorando le loro prestazioni in modo "adattivo", progressivamente ad un aumento degli "esempi" da cui apprendere.
Volendo effettuare un'osservazione da una prospettiva informatica anziché scrivere il codice di programmazione attraverso il quale, passo dopo passo, si "dice" alla macchina cosa fare, al programma vengono forniti "solo" dei set di dati che vengono elaborati attraverso questi algoritmi sviluppando una propria logica per svolgere la funzione, l'attività, il compito richiesti (per esempio imparare a riconoscere una figura).

1.1 Categorie di Machine Learning

Il *Machine Learning*, ovvero apprendimento automatico, è caratterizzato da tre categorie[4],definite come paradigmi, quali:

- Unsupervised learning (apprendimento non supervisionato).
- Supervised learning (apprendimento supervisionato).
- Reinforcement learning (apprendimento per rinforzo).

Ognuno di questi viene utilizzato per determinare la risoluzione di problemi ben precisi, a seconda del fatto che si diano al computer esempi completi da utilizzare come l'indicazione per eseguire il compito richiesto, oppure che si lasci lavorare il software senza alcun contributo.
Tutto ciò è reso possibile mediante alcune tecniche quali: la classificazione, il clustering[5], la regressione, la riduzione della dimensionalità dei dati e la stima della densità di probabilità.

[4] Le macrocategorie del machine learning più rilevanti riguardano l'apprendimento supervisionato e quello non supervisionato, ma vi sono altri sottoinsemi che definiscono una suddivisione ancor più specifica dell'apprendimento automatico, in relazione ai tipi di problemi da risolvere.
[5] Il clustering è anche detto raggruppamento o analisi di gruppi, cui membri hanno caratteristiche simili ma non tutte.

Nell*'supervised learning*[6] (apprendimento supervisionato), che è anche quello frequentemente utilizzato, il computer riceve come input sia un set di dati (training test), su cui verrà "addestrato", sia una serie di output che identificano i risultati attesi su un set di istanze (test set), con lo scopo di generare un modello che crei una relazione tra gli input e output e che usufruisca di questa connessione anche per compiti di volta in volta differenti.
La funzione che viene trattata per la risoluzione di tali problemi è chiamata *loss function* e mette in risalto il divario tra i valori attesi e quelli predetti dal modello in fase di "training".

Come scrive Adam Geitgey nel suo articolo *"Machine Learning is Fun!":*

«*Nell'apprendimento supervisionato il lavoro di risoluzione viene lasciato al computer. Una volta compresa la funzione matematica che ha portato a risolvere uno specifico insieme di problemi, sarà possibile riutilizzare la funzione per rispondere a qualsiasi altro problema similare* [2]»

L'apprendimento supervisionato, in merito all'output atteso dal problema, può essere identificato attraverso due sottoclassi:

1. La classificazione: Questa tecnica ha come scopo quello di predire un risultato inerente a classi, quindi dati, che verranno fornite in futuro alla macchina, attraverso delle *label*[7], grazie al precedente lavoro di apprendimento, mediante un training set fornito come input.
2. La regressione: Nella regressione si dispone di un numero di variabili predittive (descrittive) e una variabile target continua (il risultato). In questo tipo di problema si cerca di trovare una relazione tra queste variabili al fine di prevedere un risultato. Dunque viene esaminata la relazione di due o più variabili, entrambe indipendenti.

[6] Questo paradigma è quello di interesse rispetto alla trattazione della tesi, nello specifico nella sezione successiva verrà analizzato il problema della classificazione e quindi le procedure dello scenario in questione.
[7] Le label, cioè le etichette, sono i valori discreti non ordinati, che appartengono ad un insieme di una classe.

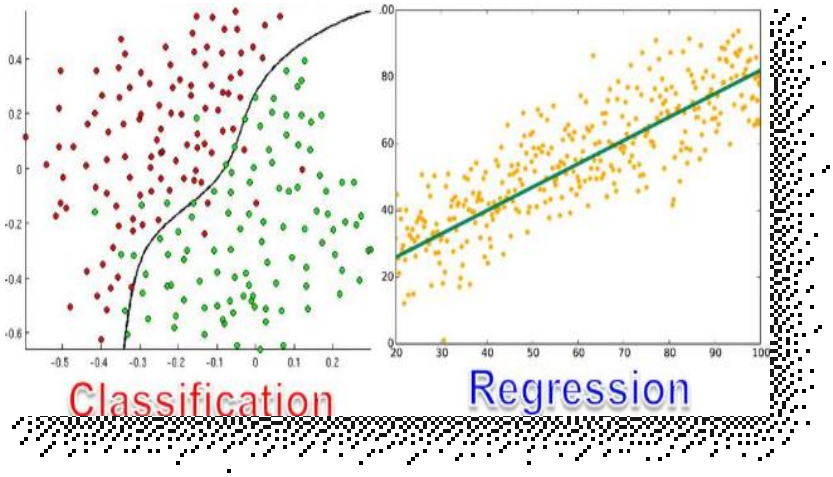


Figura 1 Esempi di algortmi di classificazione e regressione.[8]

Nell'*unsupervised learning* vengono forniti dei dati come input e non vi è alcun risultato atteso come output, cioè lo scopo è conoscere la struttura intrinseca dei datasets, che non vengono identificati con etichette in modo esplicito.

L'apprendimento non supervisionato può rappresentare un obiettivo a sé stante (ad esempio per la scoperta di pattern nascosti nei dati) o essere rivolto all'estrapolazione di caratteristiche salienti dei dati (feature) utili per l'esecuzione di un'altra *task*[9] di machine learning.

Anche in tal caso, come per il supervised learning, è lecito illustrare la suddivisione dei problemi di apprendimento non supervisionato in:

1. Clustering (raggruppamento): È di grande utilità nell'analisi dei big data, in quanto determina il raggruppamento intrinseco tra i dati non etichettati forniti. Questo algoritmo non supervisionato, deve fare alcune ipotesi che costituiscono la somiglianza dei punti e ogni assunzione rende cluster diversi e ugualmente validi, come in figura 2[10].

[8] https://www.deeplearningitalia.com/a-general-introduction-to-learning-methods-2/.
[9] Task: compito da eseguire a seconda del problema di classificazione scelto.
[10] http://www.andreaminini.com/ai/machine-learning/clustering.

x1	x2	x3
2,4	1,25	1
3,1	1,3	2
3,3	1,31	2
2,2	1,24	1
2,3	1,25	1
3,2	1,32	2
2,2	1,27	1
3,1	1,33	2

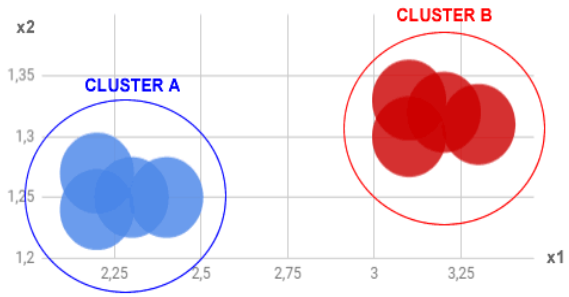


Figura 2 Esempio di clustering.

2. Riduzione della dimensione dei dati: Questo algoritmo offre una soluzione al problema della *curse of dimensionality*, quando vi è l'esigenza di ridurre la dispersione dei pattern[11] di un big dataset, senza perdere contenuto informativo, al fine di rendere il processo più efficace dal punto di vista della complessità temporale e complessità spaziale.

Infine per quanto concerne il *reinforcement learning* (apprendimento per riforzo) il pensiero da cui origina tale tecnica è molto intuitivo. Il computer interagisce con un ambiente dinamico che deve raggiungere un obiettivo specifico (ad esempio, guidare un'auto o affrontare un avversario in una partita). Quando il computer esplora l'ambito del problema, riceve un feedback in termini di ricompense (feedback positivo) o punizioni (feedback negativo) per guidarlo alla soluzione migliore, attraverso tentativi ed errori.

Rispetto al paradigma interessato alla supervisione, vi è una netta differenza, evidenziata dall'assenza di una chiave di risposta intrinseca nel set di dati (training set), in quanto è la funzione di rinforzo che fornisce la decisione adeguata a eseguire uno specifico compito.

[11] Quando si parla di pattern nell'apprendimento automatico si vuole indicare l'uso di potenti algoritmi per etichettare la regolarità dei dati forniti.

Quindi lo scopo dell'apprendimento per rinforzo è la massimizzazione della funzione di rinforzo.

Le tecniche di machine learning illustrate precedentemente sono schematizzate nell'immagine di figura 3[12].

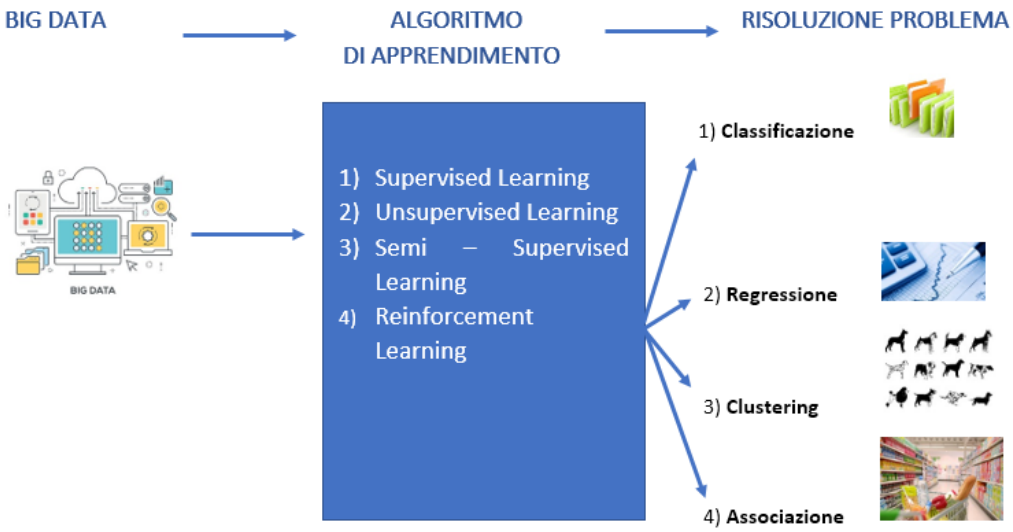


Figura 3 Paradigmi del machine learning.

[12] https://lorenzogovoni.com/machine-learning-e-funzionamento/.

1.2 Classificazione

Nel paragrafo precedente si è discusso di come il machine learning sia catalogato in paradigmi che si differenziano a seconda degli output richiesti rispetto al problema. In particolare si vuole approfondire la discussione rispetto alle variabili di risposta caratterizzanti il *supervised learning,* se è di tipo categorico si parla di *classificazione*[13].

La classificazione è una procedura di previsione sulla classe di specifici data points. Le classi sono talvolta denominate come tag o labels. La modellazione predittiva di classificazione è il compito di approssimare una funzione di mappatura (f) da variabili di input (X) a variabili di output discrete (y) [3].

Ad esempio, il rilevamento dello spam nei provider di servizi di posta elettronica può essere identificato come un problema di classificazione. Questa è una classificazione binaria poiché ci sono solo 2 classi come spam e non spam. Un classificatore utilizza alcuni dati di addestramento per comprendere in che modo le variabili di input fornite si riferiscono alla classe. In questo caso, come dati di addestramento devono essere utilizzati messaggi di posta elettronica noti come spam e non spam. Quando il classificatore viene addestrato in modo accurato, può essere utilizzato per rilevare un'e-mail sconosciuta [3].

Attraverso i cosiddetti parametri stimatori quali:

- *True positive (TP)*
- *True negative (TN)*
- *False positive (FP)*
- *False negative (FN)*

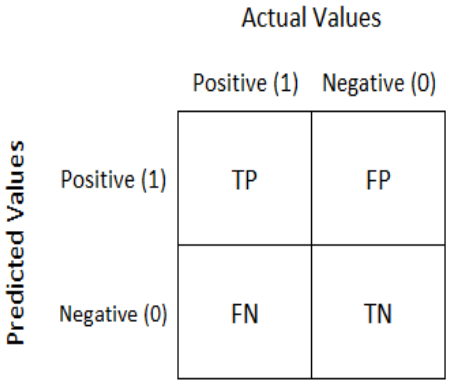
è possibile ricavare la misura di *accuracy* (accuratezza), definita come il grado di "bontà" rispetto al risultato richiesto, più la percentuale è elevata maggiore sarà la performance del classificatore.

[13] Nello specifico la classificazione è intesa come classificazione binaria, in quanto in relazione all'ambito clinico in questione, le classi potranno essere solo di due tipologie: pazienti sani e pazienti malati.

È espressa come il rapporto tra i dati classificati correttamente e quelli totali, in formule:

$$Accuracy = \frac{T\,P + TN}{TP + TN + FP + FN}$$

Inoltre i parametri scritti precedentemente, possono essere estratti dalla matrice di confusione (*confusion matrix)*, detta anche matrice di errata classificazione o dispersione, restituisce una rappresentazione dell'accuratezza di classificazione statistica. È una tabella con 4 diverse combinazioni di valori previsti e effettivi.



Ci si può rendere conto di come l'accuratezza sia un fondamentale alleato per il successo di un buon classificatore, ma bisogna considerare anche la rapidità con cui un elaboratore processa il risultato (*training time*), che sicuramente va oltre la capacità umana.

Proprio per questo il *machine learning* sta occupando uno spazio via via maggiore nell'ambito medico, finalizzato a diagnosi precoci.

L'attività descritta viene svolta grazie ad algoritmi specifici, detti algoritmi statistici, che vengono scelti a seconda del *dataset* in esame, quindi alcuni saranno più vantaggiosi di altri, ovviamente il vantaggio è valutato confrontando le percentuali di *accuracy*, la rapidità di esecuzione e la complessità del modello.

1.3 Model Evaluation

La capacità di generalizzare il modello su nuovi dati futuri è un aspetto altrettanto fondamentale, bisogna avere la certezza che il modello effettui delle previsioni su campioni che "non ha mai visto prima".
Verranno illustrate delle *tecniche di valutazione*, che consentono di esaminare le capacità di generalizzare un modello di machine learning.
I metodi per valutare le prestazioni di un modello sono divisi in due categorie, vale a dire: Holdout-validation e Cross-validation. Entrambi i metodi utilizzano un test set per valutare le prestazioni del modello, è consigliabile far riferimento a dati che non sono stati impiegati per creare il modello. Altrimenti quest'ultimo potrebbe ricordare l'intero set di allenamento e quindi predirà sempre l'etichetta corretta per qualsiasi punto del set di addestramento. Questo è noto come *overfitting.*

1.3.1 Holdout-validation

L'obiettivo dell'Holdout-validation è di testare dei dati differenti da quelli del modello di addestramento, fornendo una stima imparziale delle prestazioni di apprendimento.
Il dataset è suddiviso in maniera casuale in tre sottoinsiemi:

- Training set: sottoinsieme del set di dati su cui viene addestrato il modello, generando poi il modello predittivo.
- Validation set: sottoinsieme del dataset su cui si effettuano delle valutazioni, in termini di prestazioni, del modello realizzato, durante il training.
- Test set: sottoinsieme del dataset utilizzato per esaminare, in termini probabilistici, le prestazioni future di un modello. Nel momento in cui un modello di adatta meno al set di test, rispetto al training set, probabilmente è dovuto all'overfitting [4].

Nell' Holdout, solitamente, una suddivisione comune prevede l'uso dell'80% dei dati per l'addestramento e il restante 20% dei dati per il test.

Questa tecnica è spesso adottata per via della sua velocità, semplicità e versatilità, tuttavia richiede un numero di osservazioni N maggiore del numero di campioni M, quindi sicuramente ha delle limitazioni.

1.3.2 Cross-validation

Nella Cross-validation il set di dati di osservazione originale è partizionato in un set di addestramento ed un set indipendente su cui viene effettuata la valutazione del modello.
La tecnica più comune è k-fold Cross-validation, in cui il set di dati originale è partizionato in k sottocampioni di uguale dimensione, chiamati *pieghe*. k è un numero intero fissato dall'utente, generalmente si preferisce un valore di k pari a 5 o 10. Questo procedimento viene ripetuto k volte, in modo tale che ogni volta uno dei k sottoinsiemi venga utilizzato come insieme di test e gli altri k-1 sottoinsiemi vengano messi insieme per formare un insieme di addestramento. Per ricavare l'adeguatezza del modello si stima l'errore medio rispetto le k prove [4].
La Cross-validation, richiede un tempo di esecuzione elevato poiché stima più modelli e l'algoritmo di addestramento è rieseguito da zero k volte, ma favorisce la riduzione della distorsione del modello e ne aumenta l'efficacia, per questo è ideale quando si lavora con *small data.*

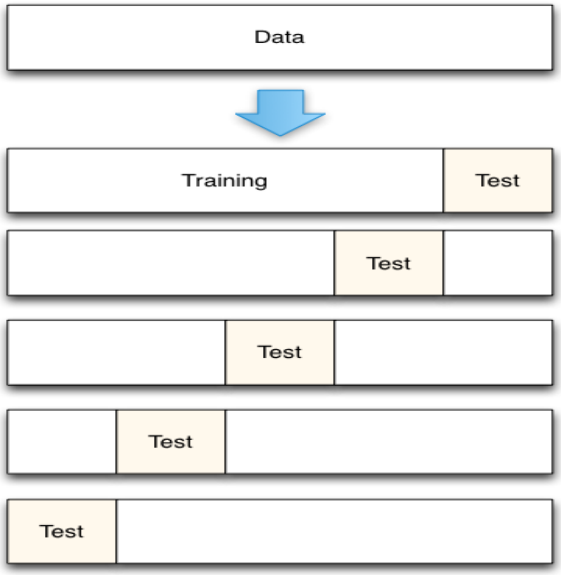


Figura 4 Cross-validation k-fold.[14]

1.4 Classification Algorithms

In questa sezione verranno descritti alcuni degli algoritmi di classificazione utilizzati in fase di sperimentazione, d'altronde la loro conoscenza è considerevole poiché si presume che nel prossimo futuro, oltre il 25% degli attuali lavori sarà soppiantato da algoritmi di machine learning[15].

- SVM

  Support Vector Machine è un algoritmo di apprendimento automatico supervisionato per problemi di classificazione o regressione, in cui il set di dati insegna a questo algoritmo le classi, in modo tale da classificare qualsiasi nuovo dato.

  Dividendo i dati di apprendimento in classi il risultato, l'SVM fa in modo che la distanza tra queste ultime sia massimizzata, se viene identificata la linea (*hyperplane*) che massimizza la distanza tra le classi, aumenta la probabilità di generalizzare bene i dati invisibili.

[14] https://lorenzogovoni.com/5-tecniche-di-cross-validation/.
[15] https://www.dezyre.com/article/top-10-machine-learning-algorithms/202.

Vantaggi:

i. Migliori prestazioni di accuratezza sui dati di *training.*
ii. Efficienza per classificazione di dati futuri.
iii. Non vi è sovradattamento di dati.

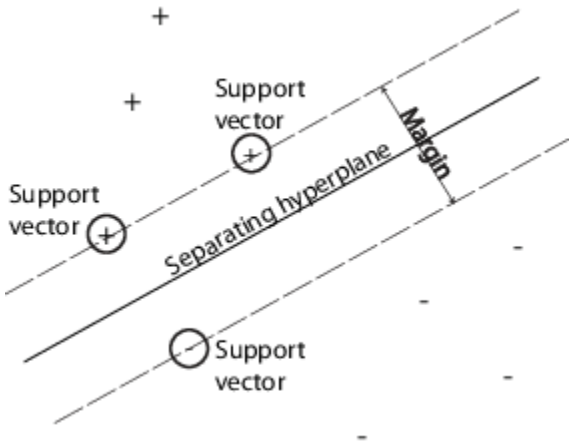


Figura 5 SVN Algorithm.[16].

- K-Nearest Neighbor (KNN)

  Conoscendo i *data points*[17], separati in diverse classi, il suo obiettivo è di predire una nuova istanza. Il suo *operatio* si basa sull'affinità delle caratteristiche: più un'istanza è vicina a un data point, più il KNN li considererà simili.

[16] https://www.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html
[17] Un data point è un'unità discreta di informazioni in senso generale, ogni singolo fatto è un punto dati. In un contesto statistico o analitico, un punto dati è solitamente derivato da una misurazione o ricerca e può essere rappresentato numericamente e / o graficamente.

La somiglianza viene ricavata attraverso la distanza euclidea e vi è la definizione di un parametro k, che identifica il numero dei *data points* più vicini. La classe che ottiene il maggior numero di queste distanze è scelta come previsione.

Per quanto riguarda i vantaggi, si può dire che a seconda del parametro k il potere predittivo sarà più o meno elevato. In quanto con un k piccolo stiamo limitando la regione di una determinata previsione e costringendo il nostro classificatore ad essere "più cieco" rispetto alla distribuzione generale, al contrario con un k più grande vi è una diminuzione della varianza causata dall'impatto casuale, ma ciò potrebbe inficiare sui piccoli dettagli, che verrebbero ignorati anche se di rilevante importanza. Quindi vi è una sorta di *trade-off* nella scelta di tale parametro.

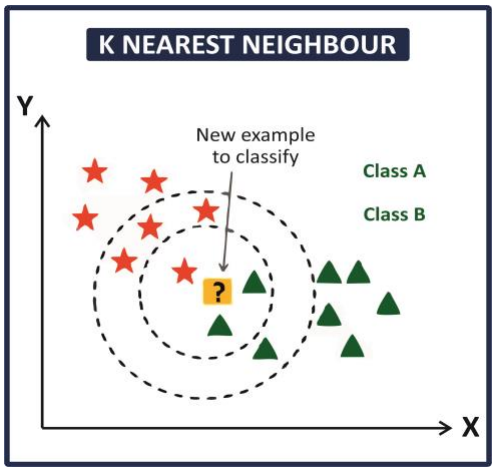


Figura 6 Esempio di algoritmo KNN[18].

- Decision Tree

  Un albero decisionale è una rappresentazione grafica che utilizza la metodologia di ramificazione per esemplificare tutti i possibili

[18] http://test.basel.in/product/knn-naive-bayes-classifier-using-excel/.

risultati di una decisione, in base a determinate condizioni. In un albero decisionale, il nodo interno rappresenta un test sull'attributo, ogni ramo dell'albero rappresenta il risultato del test e il nodo foglia rappresenta una particolare etichetta di classe, ovvero la decisione presa dopo aver calcolato tutti gli attributi. Le regole di classificazione sono rappresentate attraverso il percorso (detto *path*) dalla radice (detta *root*) al nodo foglia.

Vantaggi:

i. Non richiedono ipotesi sulla linearità.
ii. Gli alberi decisionali eseguono implicitamente la selezione delle caratteristiche.
iii. Minimizzazione del tempo nella preparazione dei dati.
iv. I valori "anomali" non influiscono sugli alberi decisionali, poiché la suddivisione dei dati avviene in base ad alcuni campioni all'interno dell'intervallo di suddivisione e non su valori assoluti esatti.

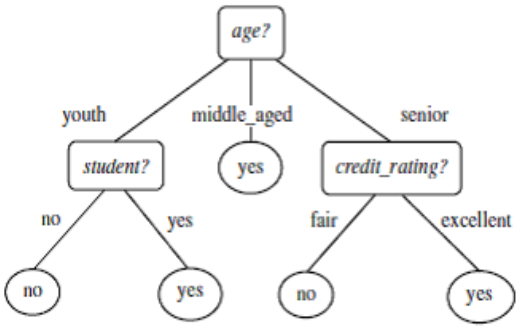


Figura 7 Esempio di Decision Tree[19].

- Linear Regression

  L'algoritmo di regressione lineare determina il legame tra due variabili e come la modifica di una variabile influisce sull'altra.

[19] https://www.kdnuggets.com/2016/10/decision-trees-concise-technical-overview.html.

L'algoritmo mostra l'impatto sulla variabile dipendente attraverso la variazione della variabile indipendente. Le variabili indipendenti sono indicate come variabili esplicative, poiché spiegano i fattori che influenzano la variabile dipendente quest'ultima viene spesso definita fattore di interesse o *predittore.*

Vantaggi:

i. È uno degli algoritmi che può essere interpretato con maggior facilità.
ii. Tecnica di apprendimento che funziona velocemente.

- Linear Discriminant

  Linear Discriminant Analysis (LDA) è uno schema ben noto per l'estrazione di elementi e la riduzione delle dimensioni, è basato sulla ricerca di combinazioni lineari di *feature,* che caratterizza o scinde due o più classi di oggetti o eventi. È stato ampiamente utilizzato in molte applicazioni che coinvolgono dati ad alta dimensione. Una limitazione intrinseca dell'LDA classica è il cosiddetto problema di singolarità, cioè fallisce quando tutte le matrici di dispersione sono singolari [5].

  Per ovviare a tale problematica si ricorre, soprattutto quando si lavora con matrici di grandi dimensioni (*big data*), a delle tecniche mirate alla riduzione delle dimensioni, non compromettendo il contenuto informativo. Le tecniche di cui si discute prendono il nome di PCA (*principal component analysis*) e Robust PCA, queste ultime non solo permettono di ridurre i costi di calcolo, per un determinato problema di classificazione, ma prevengono l'*overfitting* (eccesso di adattamento) rendendo minimo l'errore nella stima dei parametri.

CAPITOLO 2

2 FEATURE EXTRACTION

2.1 PCA (principal component analysis)

L'idea su cui si fonda questa tecnica prevede di preservare le informazioni statistiche, riducendo le dimensionalità di un *big data,* tutelandone la "variabilità". Affinché sia preservata la variabilità vengono ricercate nuove variabili, funzioni lineari del set di dati originale, che massimizzano la varianza rendendole incorrelate tra loro. La ricerca delle componenti principali si traduce in un problema di autovalori e autovettori, quindi sull'analisi della matrice di covarianza [6] .
Considerando una matrice *X* di dimensioni *n x m*, la cui j-esima colonna è il vettore xj delle osservazioni sulla j-esima variabile, si andrà a ricercare una combinazione lineare delle colonne della matrice X con la massima varianza. La prima componente principale sarà caratterizzata dalla seguente espressione, in formule:

$$Y = a_{11}X_1 + a_{12}X_2 + \ldots\ldots + a_{1m}X_m \tag{1}$$

o ancora

$$Y_1 = \overrightarrow{a_1} \cdot \vec{X} \tag{2}$$

Massimizzando la varianza della (2) si andranno a ricercare i coefficienti, i pesi possono essere scelti arbitrariamente larghi.
Si ha quindi:

$$Var(Y_1) = \vec{a}_1^T \sum_x \overrightarrow{a_1} \tag{3}$$

Dove $\sum_x$ è la matrice di covarianza di X.

Il procedimento sarà iterativo, al fine di trovare m variabili, con pesi a_i denominati *loadings*[20].

Dunque per determinare le componenti principali di un campione casuale multivariato, è necessario calcolare gli autovalori e gli autovettori della matrice di covarianza associata al campione:

$$\sum\vec{x} = \lambda\vec{x}$$, con λ autovalore e $\vec{x}$ autovettore.

Questo iter contribuirà alla formazione della nuova matrice ridimensionata, cui per ogni autovettore è associato un autovalore corrispondente alla varianza della componente principale ad esso associato.

Quantitativamente la scelta dei componenti, sufficienti a riprodurre con buona approssimazione i dati di partenza può essere fatto graficando gli autovalori, cui raffigurazione è chiamata *screen plot.* Lo *screen plot* presenta sull'asse delle ascisse il numero d'ordine degli autovalori (k) e in ordinata i relativi autovalori (λ). Osservando il grafico, la preferenza sul numero di componenti andrà a coincidere con il punto di "gomito" della spezzata (Fig. 8), gli autovalori con maggior pendenza saranno quelli maggiormente caratterizzanti nel processo.

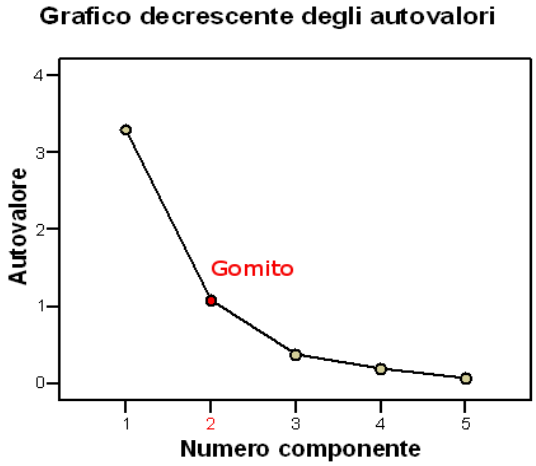


Figura 8 Esempio di screen plot (PCA).

[20] I loadings sono gli autovettori associati agli autovalori, ossia $\vec{a}_1$ è tale che $\sum_x \overrightarrow{a_1} = \lambda_1\vec{a}_1$,..etc.

2.2 Robust PCA

Oltre alla problematica relativa alla dimensionalità della matrice di dati, causa numero elevato di attributi, non è da sottovalutare una seconda problematica, spesso ricorrente nei big data, la presenza di valori ridondanti (*outliers*), cioè valori pressoché uguali che rivestono una grande porzione nella totalità dei dati, vengono considerati "pericolosi" in quanto potrebbero inficiare il risultato delle sperimentazioni, risultando falsate. Gli *outliers* potrebbero essere causati da eventuali errori nella fase di set-up delle misure.

La PCA è facilmente realizzabile attraverso la *decomposizione del valore singolare* (SVD) sulla matrice di dati, se questi ultimi possono essere definiti "puliti". Quando vi sono dati corrotti da pochi outliers, lo stesso problema diventa più complicato, in quanto l'SVD risulta sensibile ai valori erratici. L'incognita principale dell'apprendimento PCA o subspazio per i dati anomali corrotti si chiama Robust PCA o "*apprendimento subspaziale*" robusto.

Poiché il termine "outlier" non ha un preciso significato matematico, il problema della robust PCA non era, fino a poco tempo fa, ben definito, ciononostante, esistevano molte tecniche classiche per risolverlo. Negli ultimi anni, ci sono stati molteplici tentativi di qualificare questo termine. Il più popolare tra questi è l'idea di trattare un caso anomalo come una corruzione additiva, questa è una definizione valida perché modella il fatto che i valori erratici si verificano raramente e permette loro di avere qualsiasi grandezza. In particolare, la loro grandezza può essere molto più grande di quella dei punti di dati reali. Partendo da tale definizione la robust PCA è stata esaminata come il problema di scomporre una data matrice di dati, M, nella somma di una matrice di basso rango, L, la cui colonna subspaziale dà le componenti principali, e una matrice sparsa (matrice di outliers), S, spesso indicata come la formulazione sparse+low-rank (S+LR) [7]. Quanto scritto viene eseguito risolvendo il seguente problema di ottimizzazione chiamato *Principal component pursuit (PCP)* [8], in cui:

$$\|L\|_* + \lambda\|S\|_1$$ [21]

$$L + S = M$$

Viene recuperato L_0, matrice di basso rango e S_0, matrice "sparsa"[22].

Algoritmicamente, il problema di cui sopra può essere risolto con algoritmi efficienti e scalabili, ad un costo non molto più alto del classico PCA[23]. Quindi, affinchè sia possibile ricavare la matrice di basso rango, vi sono degli algoritmi (sia per casi pienamente osservati sia per casi osservati parzialmente) di Robust PCA basati sull'algoritmo di discesa del gradiente che offrono una garanzia teorica per il recupero di tale matrice di sottospazio, con tasso di convergenza migliore e miglior accuratezza; oppure algoritmi non convessi come Fast RPCA (algoritmo di osservazione completa) e Fast RPCA with partial observations (algoritmo di osservazione parziale).

[21] ||.||_* è la norma nucleare, mentre ||.||_1 è la norma.
[22] E. J. Cand`es, X. Li, Y. Ma e J. Wright, «ROBUST PRINCIPAL COMPONENT ANALYSIS?,» December 17, 2009.
[23] Infatti l'algoritmo utilizzato nell'implementazione della Robust PCA nella tesi in questione, è quello presente nell'articolo citato nella suddetta sezione, di più facile comprensione rispetto ad altri che sono analiticamente più complessi.

CAPITOLO 3

3 DATASET

3.1 Segnali Elettroencefalografici (EEG)

Un segnale EEG è una misura delle correnti che fluiscono durante l'eccitazione sinaptica dei dendriti[24] di molti neuroni piramidali nella corteccia cerebrale. Quando le cellule cerebrali (neuroni) sono attive, le correnti sinaptiche sono prodotte all'interno dei dendriti. Questa corrente genera un campo magnetico, misurabile da un Elettromiogramma (EMG) e un campo elettrico secondario al di sopra del cuoio capelluto, misurabile da un sistema EEG [9].

Dunque il segnale di interesse, per l'elettroencefalografia, è un biopotenziale, ovvero potenziale d'azione, presente su tutte le membrane cellulari eccitabili, che viene acquisito e analizzato mediante una rappresentazione utile, il tracciato.

A *riposo* il potenziale d'azione in un neurone è a circa -70mV; quando giunge uno stimolo questo valore aumenta progressivamente, fino ad una soglia massima, detta soglia di eccitazione, che una volta superato genera il potenziale d'azione.

Il potenziale d'azione stimola il rilascio di neurotrasmettitori, particolari sostanze che veicolano l'informazione, che vengono intercettati da neuroni presenti sui dendriti dello spazio postsinaptico e registrati mediante l'elettroencefalogramma.

Il segnale EEG viene captato attraverso 16-24 paia di elettrodi, questi ultimi sono dischetti di Ag-AgCl[25], cui nella parte inferiore viene posto un gel conduttore, che favorisce la diminuzione d'impedenza e, quindi, migliora la trasmissione del segnale.

[24] I dendriti costituiscono le fibre minori che si ramificano, partendo dal neurone, trasportando il segnale nervoso.

[25] Questi elettrodi possiedono la capacità di far restare inalterato il potenziale, ad una specifica temperatura, così da poter essere utilizzati in soluzioni differenti.

Gli elettrodi acquisiscono il segnale a coppia, in ognuna è presente un elettrodo attivo e uno di riferimento posizionato, nel caso del segnale EEG, dietro l'orecchio.

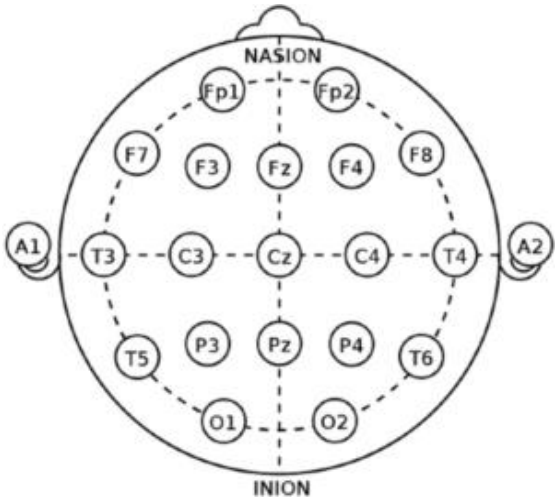


Figura 9 Posizionamento elettrodi secondo la configurazione standard 10-20[26].

Come si può osservare dalla figura 9, le lettere che identificano gli elettrodi si differenziano a seconda della zona dello scalpo, diviso in emisfero destro e sinistro, dove andranno collocati.

La frequenza del suddetto segnale è centrata nell'intervallo che va dai 0.5-100 Hz, suddiviso a sua volta in cinque sotto bande, dove ogni banda corrisponde ad una specifica attività cerebrale. Generalmente le basse frequenze (ampiezze maggiori) sono associate ad uno stato di "rilassamento", mentre le frequenze più alte caratterizzano uno stato di "attività", infatti le onde determinanti una specifica attività possono distinguersi in:

- onde *alfa*: identificano uno stato di rilassamento.
- onde *beta* e *gamma*: identificano uno stato in cui la mente è attiva e concentrata.
- onde *delta*: identificano uno stato patologico come il coma.

[26] https://biomedicalcue.it/elettroencefalografia-cose-funziona/10469/.

- onde *teta*: identificano uno stato di sonno profondo.[27]

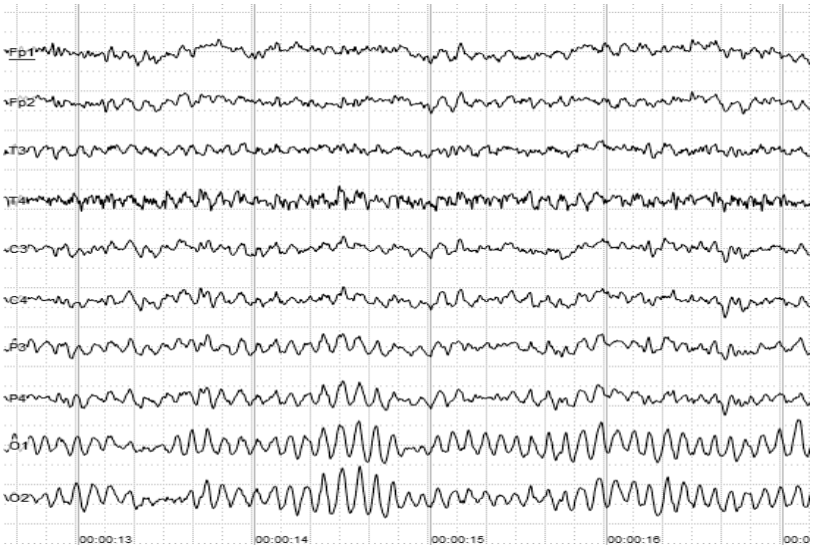


Figura 10 Esempio di tracciato EEG.

3.2 Dati EEG: Alzheimer's Subjects - Healthy Subjects

I segnali elettroencefalografici (EEG) trattati, in questo lavoro, riguardano in parte pazienti affetti dalla malattia degenerativa nota come morbo di Alzheimer e in parte pazienti sani.

I segnali sono stati acquisti mediante fonte anonima, sotto forma di dati "grezzi" nei seguenti formati: EDF (european data format), DAT, GNT.

I dati sono stati quindi suddivisi in due cartelle denominate in relazione alle due classi specifiche, cui:

- Soggetti affetti da mordo di Alzheimer → AD (*Alzheimer disease*).
- Soggetti sani → HS (*healthy subjects*).

[27] Fonte figura 10: https://team.inria.fr/potioc/old-research-topics/eeg-signal-processing/.

I soggetti di cui sono stati forniti i dati sono in totale 13, 7 corrispondenti alla classe AD e i restanti 6 corrispondenti alla classe HS, inoltre per ognuno di essi vi sono dalle 21 alle 23 tracce, acquisite mediante gli elettrodi. Tra i vari formati, con cui sono stati presentati i dati inizialmente, si è scelto il formato EDF con conseguente conversione in formato MAT. Questo procedimento è stato eseguito su ogni soggetto, per entrambe le classi, ottenendo 13 matrici di dimensioni differenti.

Le fasi di conversione dati, creazione *data_matrix* (matrice di dati) con applicazione di tecniche di *feature extraction* e classificazione sono state svolte per mezzo dell'ambiente di calcolo scientifico Matlab (*Matrix Laboratory*).

Attraverso la generazione delle matrici sono stati plottati i tracciati per ogni soggetto, ma al fine di una visualizzazione più accurata dei segnali è stato necessario impiegare un software, specifico per i file con l'estensione scelta, chiamato *edfbrowser*[28], un visualizzatore e toolbox gratuito, open source, multipiattaforma, universale progettato per, ma non solo, file di archiviazione di timeseries come EEG, EMG, ECG, BioImpedance ed altri file format.

Il software ha quindi consentito la raffigurazione, in un'unica finestra, delle 21-23 tracce di ogni paziente, della durata media di circa 24 minuti[29], in cui la scala temporale è stata compressa a tutto l'intervallo di registrazione e l'ampiezza adattata alla dimensione della finestra, le relative rappresentazioni sono mostrate in Fig 11-12.

[28] https://www.teuniz.net/edfbrowser/.
[29] Una registrazione di base standard deve comprendere almeno 20 minuti di tracciato ben eseguito, le registrazioni più prolungate vengono effettuate quando si vogliono acquisire informazioni aggiuntive circa un paziente.

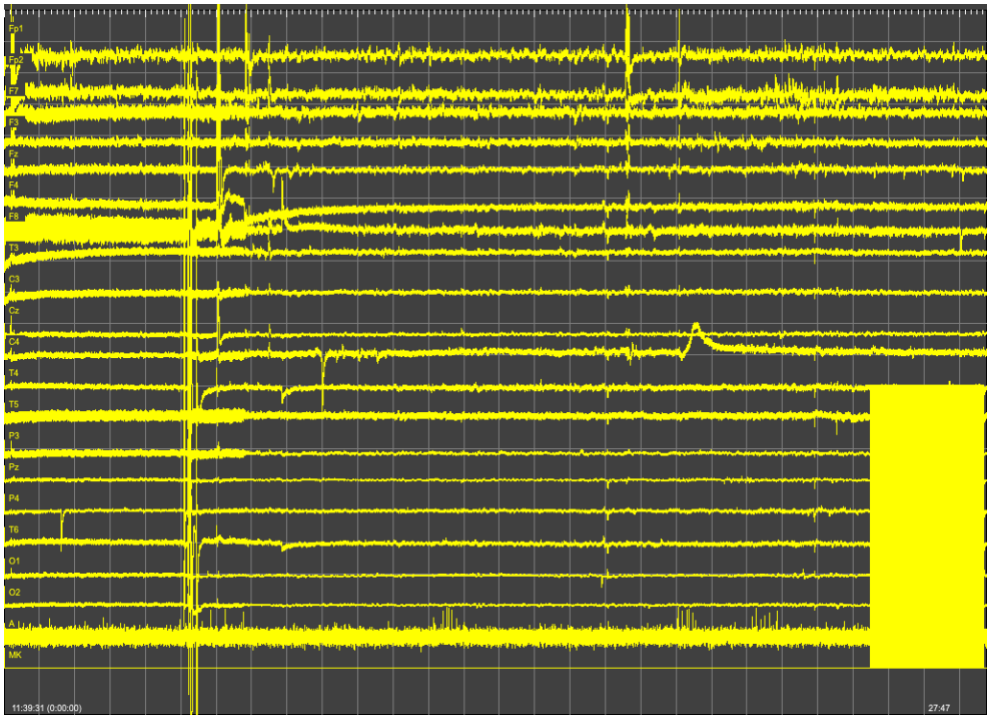


Figura 11 Rappresentazione tracciato EEG riguardante un paziente AD.

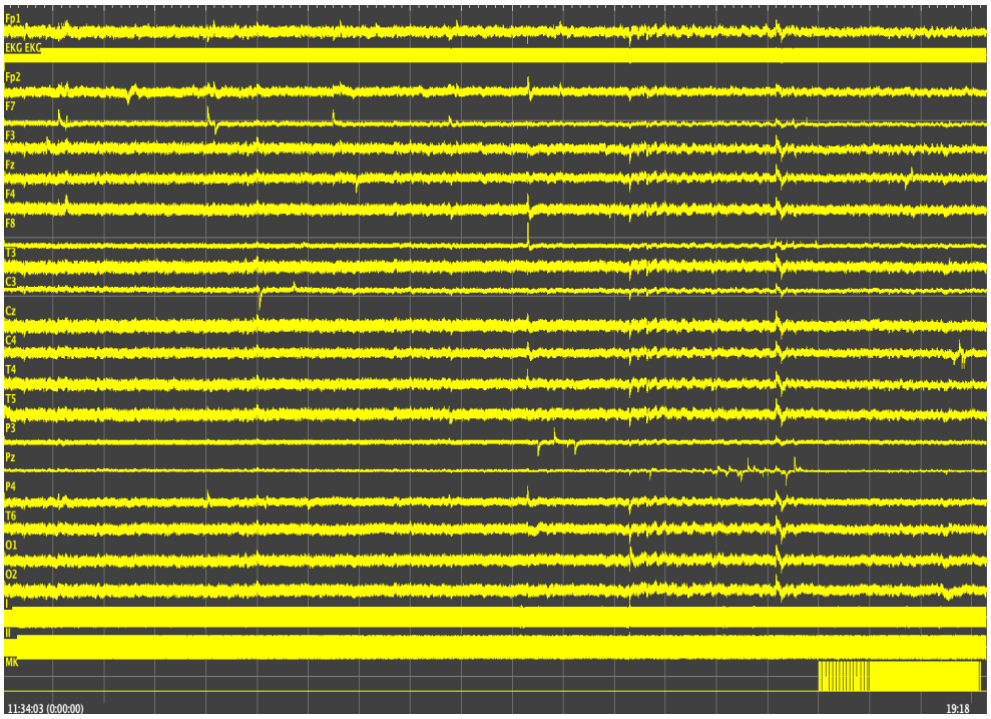


Figura 12 Rappresentazione tracciato EEG riguardante un paziente HS.

La porzione di codice corrispondente la conversione dei file è la seguente [10]:

```
%CONVERSIONE FILES AD
path = 'EEGdata/AD/';


for i = 1:7
    input_filename = strcat('s', int2str(i), '.edf');
    [hdr,record] = edfread(strcat(path, input_filename));
    output_filename = strcat('s', int2str(i), '.mat');
    save(strcat(path, output_filename), 'record');
    %plot(record(i,:));
end


%CONVERSIONE FILES HS
path = 'EEGdata/HS/';


for i = 1:6
    input_filename = strcat('s', int2str(i), '.edf');
    [hdr,record] = edfread(strcat(path, input_filename));
    output_filename = strcat('s', int2str(i), '.mat');
    save(strcat(path, output_filename), 'record');
    %plot(record(i,:));
end
```

3.3 Data Matrix – Data Matrix New

Le matrici ottenute relative ai 13 pazienti (AD-HS) presentavano dimensioni differenti, a causa del numero di colonne altamente variabile ed elevato, non permettendo la concatenazione fra le stesse. Pertanto è stato necessario determinare un valore minimo comune a tutte le matrici in modo da troncarle al medesimo valore, così da ottenere un numero equo di colonne e proseguire con la creazione della matrice di dati, chiamata *data_matrix*, le cui dimensioni sono le seguenti:

- Data Matrix dimension = 283x148098

Nella fase di concatenazione, sono state aggiunte due colonne che indentificano la classe di appartenenza del soggetto (AD o HS) e il numero del soggetto stesso. Questo passaggio è stato fondamentale, poichè scegliendo nella fase di train un modello di classificazione, il task è di identificare delle label, nello specifico sono state definite:

- 1→ Label coincidente con la classe AD.
- 2→ Label coincidente con la classe HS.

Di seguito il codice relativo alla fase di composizione della matrice.

```
path = 'EEGdata/HS/';

H = load(strcat(path, 's1.mat'));
I = load(strcat(path, 's2.mat'));
L = load(strcat(path, 's3.mat'));
M = load(strcat(path, 's4.mat'));
N = load(strcat(path, 's5.mat'));
O = load(strcat(path, 's6.mat'));

path= 'EEGdata/AD/';

A = load (strcat(path, 's1.mat'));
B = load (strcat(path, 's2.mat'));
C = load (strcat(path, 's3.mat'));
D = load (strcat(path, 's4.mat'));
E = load (strcat(path, 's5.mat'));
F = load (strcat(path, 's6.mat'));
G = load (strcat(path, 's7.mat'));
```

```
%min_col_value

min_col_value = min([size(A.record,2) , size(B.record,2) ,
size(C.record,2) , ...
     size(D.record,2), size(E.record,2), size(F.record,2),
size(G.record,2), ...
     size(H.record,2), size(I.record,2) , size(L.record,2),
size(M.record,2), ...
    size(N.record,2), size(O.record,2)]);



     ad_tag = 1;
     hs_tag = 2;

     A_new = A
     B_new = B
     C_new = C
     D_new = D
     E_new = E
     F_new = F
     G_new = G
     H_new = H
     I_new = I
     L_new = L
     M_new = M
     N_new = N
     O_new = O



%add tag to matrices

A_new.record  = A.record (:, 1:min_col_value)
col_tag  =  ones(size(A.record,1), 1)*ad_tag;
col_subj =  ones(size(A.record,1), 1)*1;
A_new.record  = [A_new.record, col_tag, col_subj]

B_new.record  = B.record (:,1:min_col_value)
col_tag  =  ones(size(B.record,1), 1)*ad_tag;
col_subj =  ones(size(B.record,1), 1)*2
B_new.record  = [B_new.record ,col_tag ,col_subj]

C_new.record  = C.record (:,1:min_col_value)
col_tag  =  ones(size(C.record,1), 1)*ad_tag;
col_subj =  ones(size(C.record,1), 1)*3;
C_new.record  = [C_new.record ,col_tag ,col_subj]


D_new.record  = D.record (:,1:min_col_value);
col_tag  =  ones(size(D.record,1), 1)*ad_tag;
col_subj =  ones(size(D.record,1), 1)*4;
D_new.record  = [D_new.record ,col_tag ,col_subj];


E_new.record  = E.record (:,1:min_col_value)
col_tag  =  ones(size(E.record,1), 1)*ad_tag;
col_subj =  ones(size(E.record,1), 1)*5;
E_new.record  = [E_new.record ,col_tag ,col_subj]

F_new.record  = F.record (:,1:min_col_value)
col_tag  =  ones(size(F.record,1), 1)*ad_tag;
col_subj =  ones(size(F.record,1), 1)*6;
```

```
F_new.record  = [F_new.record ,col_tag ,col_subj]

G_new.record  = G.record (:,1:min_col_value)
col_tag  =  ones(size(G.record,1), 1)*ad_tag;
col_subj =  ones(size(G.record,1), 1)*7;
G_new.record  = [G_new.record ,col_tag ,col_subj]

H_new.record  = H.record (:,1:min_col_value)
col_tag  =  ones(size(H.record,1), 1)*hs_tag;
col_subj =  ones(size(H.record,1), 1)*8;
H_new.record  = [H_new.record ,col_tag ,col_subj]

I_new.record  = I.record (:,1:min_col_value)
col_tag  =  ones(size(I.record,1), 1)*hs_tag;
col_subj =  ones(size(I.record,1), 1)*9;
I_new.record  = [I_new.record ,col_tag ,col_subj]

L_new.record  = L.record (:,1:min_col_value)
col_tag  =  ones(size(L.record,1), 1)*hs_tag;
col_subj =  ones(size(L.record,1), 1)*10;
L_new.record  = [L_new.record ,col_tag ,col_subj]


M_new.record  = M.record (:,1:min_col_value)
col_tag  =  ones(size(M.record,1), 1)*hs_tag;
col_subj =  ones(size(M.record,1), 1)*11;
M_new.record  = [M_new.record ,col_tag ,col_subj]

N_new.record  = N.record (:,1:min_col_value)
col_tag  =  ones(size(N.record,1), 1)*hs_tag;
col_subj =  ones(size(N.record,1), 1)*12;
N_new.record  = [N_new.record ,col_tag ,col_subj]

O_new.record  = O.record (:,1:min_col_value)
col_tag  =  ones(size(O.record,1), 1)*hs_tag;
col_subj =  ones(size(O.record,1), 1)*13;
O_new.record  = [O_new.record ,col_tag ,col_subj]


%concatenate

data_matrix = [A_new.record; B_new.record; C_new.record;...
               D_new.record; E_new.record; F_new.record;...
               G_new.record; H_new.record; I_new.record;...
               L_new.record; M_new.record; N_new.record;...
               O_new.record]
```

La dimensionalità della matrice ha contribuito alla scelta riguardante l'uso di tecniche di *feature extraction*, necessarie per ovviare l'elevato numero di campioni rispetto al numero di osservazioni, che altrimenti avrebbe reso difficile la classificazione.

Infatti inizialmente la classificazione è stata svolta tramite server, causa di un tempo di training eccessivo e non supportato dal solo elaboratore.

Durante la fase di sperimentazione, che sarà illustrata in seguito, sono sorti dei dubbi a causa dei risultati ottenuti, successivamente alla manipolazione della data_

matrix mediante la tecnica di *windowing*[30], i valori di accuratezza ottenuti tramite l'ausilio dei vari algoritmi predittivi corrispondevano tutti al 100%, per tale ragione sono stati analizzati attentamente tutti i segnali corrispondenti ai 13 pazienti.
Da questa analisi è emersa la presenza di valori ridondanti (*outliers*) nella parte di signal head e signal tail, che variano per ogni soggetto, andando ad inficiare sulla classificazione.
Le matrici relative ai vari pazienti sono state modificate, eliminando il range di valori ridondanti e successivamente sono state concatenate con la stessa procedura illustrata precedentemente, ma con l'aggiunta di alcune righe di codice:

```
% Eliminazione degli N campioni uguali(alternati), in testa e in coda,
per
    % ogni paziente (AD e HS)

     A_new.record  = A.record (:,542:183110)
     B_new.record  = B.record (:,1199:147800)
     C_new.record  = C.record (:,953:190023)
     D_new.record  = D.record (:,820:195058);
     E_new.record  = E.record (:,36633:196830)
     F_new.record  = F.record (:,113:164026)
     G_new.record  = G.record (:,1577:178432)
     H_new.record  = H.record (:,342:148106)
     I_new.record  = I.record (:,801:155341)
     L_new.record  = L.record (:,1115:201674)
     M_new.record  = M.record (:,3451:171504)
     N_new.record  = N.record (:,4170:240764)
     O_new.record  = O.record (:,717:198257)
```

Le dimensioni della matrice a cui sono stati rimossi i valori ridondanti sono i seguenti:

- Data Matrix New dimension = 283x146604.

[30] Tecnica di finestramento applicata per ottenere la riduzione dei campioni, senza perdere contenuto informativo. In questo lavoro non si soffermerà l'attenzione su tale tecnica, in quanto, come verrà discusso, verranno utilizzate altre tecniche con vantaggi maggiori.

CAPITOLO 4

4 RISULTATI SPERIMENTALI

4.1 Classificazione su Server

Le dimensioni elevate delle matrici di dati non consentivano la classificazione, attraverso il classification learner, dell'ambiente di calcolo scientifico Matlab, perciò è stato doveroso far uso del server. I risultati ottenuti saranno illustrati nelle Tab.1 e Tab.2.

Gli algoritmi utilizzati, nelle fasi di sperimentazioni, sono i seguenti:

- TREE (fine).
- SVM (gaussian, linear, quadratic).
- KNN (fine).
- NAIVE (kernel).

Tab.1 -CLASSIFICAZIONE ORIGINAL DATA MATRIX- Split: Holdout train=80% test=20%

TREE(fine)	SVM(gaussian)	SVM(quadratic)	SVM(linear)	KNN(fine)
96,4%	53,57%	89,2%	92,8%	92,8%

Come si può osservare dalla Tab.1 i risultati della classificazione sulla matrice originale sono già soddisfacenti per quanto riguarda gli algoritmi: TREE, SVM (quadratic), SVM (linear) e KNN.
Questi risultati però potrebbero essere "falsati" da due motivi:

i) Ridotto numero di osservazioni (283) in rapporto al numero di campioni/predittori (148096).

ii) Ogni soggetto è caratterizzato da un numero significativo di tracce (derivate dall'acquisizione del segnale EEG tramine N elettrodi) quindi,

nonostante la randomizzazione applicata in fase di splitting, nella matrice di test ci saranno sempre delle componenti appartenenti a pazienti su cui è stato trainato(addestrato) il modello.

Per questo motivo, si procede ad una validazione più "veritiera" dei risultati applicando lo splitting per soggetto. In particolare i 4 pazienti scelti per il test corrispondono ai primi due appartenenti alla classe AD (1,2) e gli ultimi due appartenenti alla classe HS (12,13), mentre i 9 scelti per il train corrispondono ai restanti soggetti di entrambe le classi.

Tab.2 -CLASSIFICAZIONE ORIGINAL DATA MATRIX -Split: train= 9 pazienti test=4 pazienti

TREE(fine)	SVM(gaussian)	SVM(quadratic)	SVM(linear)	KNN(fine)
34,1%	52,3%	51,1%	36,4%	52,3%

Quest'ultimo risultato, ha di conseguenza portato alla manipolazione della data_ matrix iniziale, in quanto, come già anticipato, osservando attentamente i campioni relativi ad ogni paziente, sono stati individuati dei valori ridondanti nella parte di signal head e signal tail, che differiscono per ogni soggetto; si presume che ciò sia dovuto alla fase iniziale e finale di setup della misura, attraverso gli elettrodi.
Per ovviare al problema riscontrato, sono state adottate due differenti tecniche:

1. Eliminazione dei valori ridondanti, analizzando il range di campioni differenti delle matrici ricavate per ogni paziente, con conseguente concatenazione per la formazione di una nuova data_matrix, le cui dimensioni sono 283x146604 (ultime due colonne relative alle label delle classi e dei soggetti).

2. Utilizzo della ROBUST PCA applicata alla data_matrix originale, che permette di automatizzare la procedura di rimozione degli outliers.

Prima di applicare la PCA e la Robust PCA, si è proseguito nell'analizzare sperimentalmente la data_matrix_New (matrice senza valori ridondanti), sempre tramite server.

Tab.3 -CLASSIFICAZIONE DATA MATRIX SENZA OUTLIERS -Split: Holdout train=80% test=20%

TREE(fine)	SVM(gaussian)	SVM(quadratic)	SVM(linear)	KNN(fine)
82,1%	53,57%	91,0%	92,8%	92,8%

Si evince che attraverso l'eliminazione dei valori ridondanti (outliers) le percentuali di accuracy sono di poco ridotte, rispetto ai risultati della data_matrix precedente, in quanto la totale diversità dei dati comporta una "maggior difficoltà" da parte del classificatore nella ricerca di trend comuni.

Tab.4 -CLASSIFICAZIONE DATA MATRIX SENZA OUTLIERS -Split: train= 9 pazienti test=4 pazienti

TREE(fine)	SVM(gaussian)	SVM(quadratic)	SVM(linear)	KNN(fine)
45,4%	52,3%	62,5%	55,7%	43,2%

È chiaro che l'eccessivo numero di attributi rispetto alle osservazioni e qualità pessima di dati (presenza di outliers) può causare il fenomeno di *overfitting*.
Nel machine learning l'overfitting è il rischio di sovradattamento durante il processo di apprendimento induttivo, in quanto la presenza di attributi irrilevanti porta in secondo piano o nasconde del tutto gli attributi rilevanti. Per tale ragione si è pensato di utilizzare come tecnica statistica, descritta nel capitolo1, per riconoscere gli attributi irrilevanti, la Cross-validation, suddividendo il training test in k parti di uguale dimensione, nello specifico si è scelto k pari a 10.

Tab.5 -CLASSIFICAZIONE DATA MATRIX SENZA OUTLIERS (CROSS-VALIDATION k-folds=10)

TREE(fine)	SVM(gaussian)	SVM(quadratic)	SVM(linear)	KNN(fine)
89,2%	96,4%	82,1%	85,7%	96,4%

Tab.6 -CLASSIFICAZIONE ORIGINAL DATA MATRIX (CROSS VALIDATION k-folds=10)

TREE(fine)	SVM(gaussian)	SVM(quadratic)	SVM(linear)	KNN(fine)
92,8%	53,7%	78,5%	85,7%	96,4%

Le percentuali di accuracy ottenute con il metodo Cross si discostano relativamente da quelle ottenute con l'Holdout.

Ai fini di una possibile applicazione clinica, la classificazione verrà effettuata tramite uno split per paziente, oltre alla divisione su base percentuale. Di conseguenza da questo punto in poi il lavoro si concentrerà sullo studio e applicazione delle tecniche di feature extraction volte ad incrementare l'accuratezza della classificazione delle data_matrix ottenute, suddividendo train set e test set in modo che contengano pazienti di classi differenti.

4.2 Classificazione con PCA e Robust PCA

Successivamente alle prime sperimentazioni eseguite, ai fini di una ottimizzazione dei risultati, si è ricorso all'utilizzo di tecniche di feature extraction che potessero ridurre la dimensione dei due *big data*, non compromettendo il contenuto informativo.

Le tecniche in questione sono:

- PCA (principal component analysis).
- Robust PCA.

Su cui si è discusso nel capitolo 2.

Gli scopi primari di entrambe sussistono nella riduzione di un numero più o meno elevato di variabili (rappresentanti altrettante caratteristiche del fenomeno analizzato) in alcune variabile latenti, che considerate singolarmente presentano comunque informazioni (*feature reduction*), in più la Robust PCA permette di ovviare al problema dei valori ridondanti.
I dati aventi varianza maggiore costituiscono la componente principale, ciò significa che riportano al meglio il contenuto informativo.
Per trovare la componente principale, la PCA, calcola gli autovettori e gli autovalori della matrice. L'autovettore è la direzione degli assi cartesiani su cui verranno posti i dati originali (orizzontale, verticale, etc.…); gli autovalori sono il valore numerico che descrive quanta varianza c'è lungo la direzione determinata quindi, la componente principale può essere definita come l'autovettore che ha autovalore maggiore fra tutti. Una volta trovati gli autovettori che descrivono al meglio i dati, è possibile riportarli in questi nuovi assi, cambiando dunque punto di vista verso una prospettiva migliore, ovvero maggiormente informativa, così da ridurre le dimensioni della matrice di dati.
Per determinare il range di estrazione delle features, sulla data_matrix, data_matrix_New e data_matrix a cui è applicata la Robust PCA, è stato graficato l'andamento degli autovalori, come si evince dalla figura 13, nonostante l'eccessivo numero di attributi, solo una minima parte è fondamentale per risolvere il problema

della classificazione, infatti il numero di autovalori rientra in un range con un valore minimo intorno a 50 e un massimo di 100.

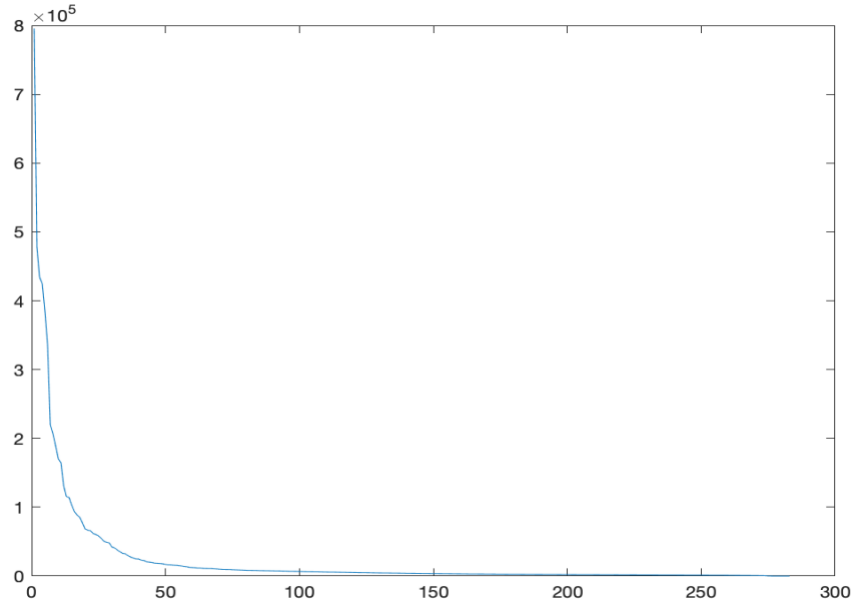


Figura 13 Grafico che rappresenta l'andamento degli autovalori, relativo alle matrci di dati.

La riduzione della matrice è stata eseguita mediante le successive righe di codice:

```
%Applicazione PCA--> data_matrix e data_matrix_NEW
input_mat_filename ='data_matrix_NEW.mat';
input_mat = importdata(input_mat_filename);
input_mat_noLabel = input_mat(:, 1:end-2); % escludo soggetti e classi



% KLT
[U, lambda, V] = svd(input_mat_noLabel, 'econ');

% Ricostruzione segnale
figure; plot(diag(lambda));% range = 50:100
num_feat =50; %valore modificato di volta in volta
M_new = U' * input_mat_noLabel(:, 1:num_feat);
M_new = [M_new, input_mat(:, end-1:end)];



% save matrix
save(strcat('M_', int2str(num_feat), 'features.mat'), 'M_new');
```

In questo modo è stato possibile ridurre il numero di campioni rispetto al numero di osservazioni.

Dopo aver generato le matrici, in base al numero di features, modificato di volta in volta, si è proseguito con la classificazione, per divisione percentuale (metodo Holdout), delle due data_matrix.

4.2.1 Holdout method

Tab.7 -CLASSIFICAZIONE: ORIGINAL DATA MATRIX CON PCA -Split :Holdout train=80% test=20%

PCA (n°features)	TREE (fine)	SVM (gaussian)	SVM (quadratic)	KNN (fine)	NAIVE (kernel)
50	82,1%	89,3%	92,9%	78,6%	91,1%
60	87,5%	91,1%	92,9%	83,9%	89,3%
70	87,5%	91,1%	91,1%	78,6%	89,3%
80	87,5%	91,1%	91,1%	78,6%	89,3%
90	87,5%	89,3%	91,1%	78,6%	89,3%
100	87,5%	89,3%	91,1%	78,6%	89,3%

Tab.8 -CLASSIFICAZIONE: DATA MATRIX SENZA OUTLIERS CON PCA - Split: Holdout train=80% test=20%

PCA (n°features)	TREE (fine)	SVM (gaussian)	SVM (quadratic)	KNN (fine)	NAIVE (kernel)
50	89,3%	83,9%	82,1%	83,9%	91,1%
60	89,3%	85,7%	83,9%	85,7%	91,1%
70	87,5%	85,7%	82,1%	85,7%	91,1%
80	87,5%	85,7%	83,9%	82,1%	91,1%
90	87,5%	83,9%	82,1%	83,9%	91,1%
100	87,5%	83,9%	83,9%	82,1%	91,1%

Dalle Tab.7 e Tab.8 si deduce di come l'accuratezza sia migliorata grazie alla tecnica di *feature extraction* PCA, rispetto ai risultati della classificazione su server. Nonostante l'uso della PCA per la data_matrix original, gli outliers potrebbero comunque essere la causa di risultati non veritieri, in quanto valori erratici che dovrebbero essere esclusi dal set di dati, infatti i valori più alti sono stati ricavati utilizzando la data_matrix_NEW.
La procedura di rimozione degli outliers è una procedura manuale e quindi onerosa, pertanto si è pensato di automatizzare il meccanismo di eliminazione usufruendo del metodo innovativo, illustrato nel capitolo 2, la Robust PCA, permettendo inoltre di ricavare dei risultati che siano inequivocabili.
Il codice corrispondente è il medesimo del precedente, con l'aggiunta di alcune righe corrispondenti alla Robust PCA:

```
%Robust PCA
lambda =1/sqrt(max(size(input_mat_noLabel)));
tic
[L,S] = RobustPCA(input_mat_noLabel, lambda, 10*lambda, 1e-5);
toc
```

La ricostruzione del segnale viene fatta prendendo in considerazione la matrice L, che sta per *low rank*, le cui dimesioni sono:

- L (Data_matrix_Robust) dimension → 283x148096[31]

Cui è applicata la *singular value decomposition* (SVD), affinchè si potesse procedere all'estrazione delle features più significative, il grafico degli autovalori corrispondenti è lo stesso di quello rappresentato in figura 13, quindi anche in tal caso il range del numero di componenti è 50-100.

[31] Dalla matrice L, state rimosse inizialmente le due colonne identificanti le label relative al numero del soggetto e la classe di appartenenza (sano-malato), che sono state aggiunte nella fase di ricostruzione, così come anche nelle altre matrici di dati.

Tab.9 -CLASSIFICAZIONE: ORIGINAL DATA MATRIX CON Robust PCA - Split: Holdout train=80% test=20%

Robust PCA (n°features)	TREE (fine)	SVM (gaussian)	SVM (quadratic)	KNN (fine)	NAIVE (kernel)
50	89,3%	89,3%	92,9%	80,4%	89,3%
60	82,1%	92,9%	94,6%	82,1%	91,1%
70	82,1%	92,9%	94,6%	82,1%	91,1%
80	82,1%	92,9%	94,6%	82,1%	89,3%
90	82,1%	92,9%	94,6%	82,1%	89,3%
100	82,1%	91,1%	94,6%	82,1%	89,3%

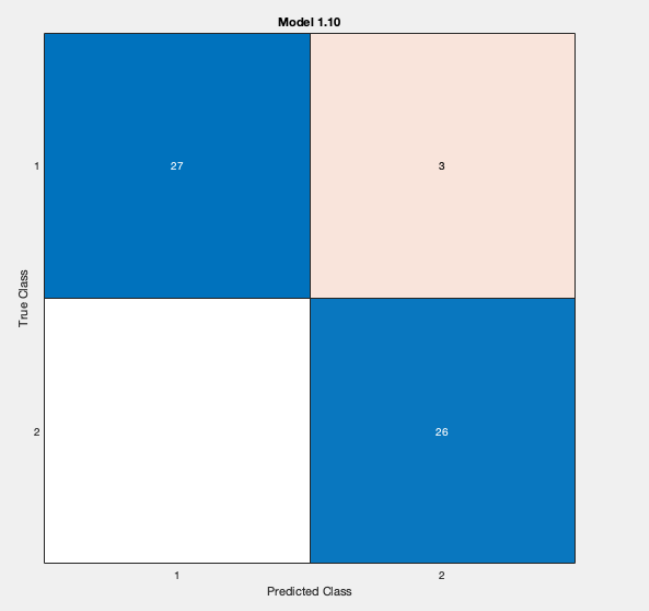
È doveroso commentare i risultati presenti nelle tabelle 7,8 e 9 aggiungendo delle considerazioni in merito alla *confusion matrix* e *training time*, che saranno di ausilio nella scelta del modello migliore.

In particolare osservando la Tab.7 i valori più alti di accuracy appartengono all'algoritmo SVM(quadratic) con circa il 92,9%, nella Tab.9 le percentuali di accuracy maggiori vengono fornite dall'algoritmo NAIVE(kernel) con circa il 91,1% ed infine valutando la Tab.9 i parametri considerevoli sono dati dall'algoritmo SVM(quadratic) con 94,6%.

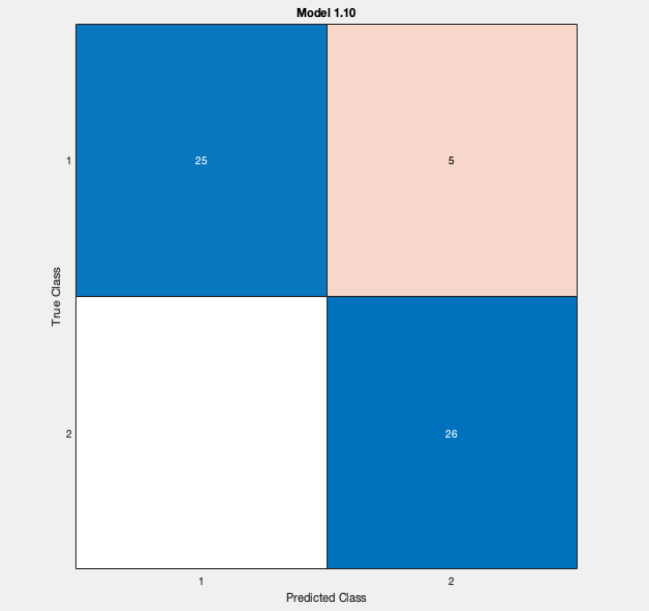
Quindi in prima analisi, non si potrebbe definire con certezza quale potrebbe essere il modello che meglio generalizza i dati.

Sono state studiate alcune delle *confusion matrix*, relative alle tre matrici di dati, nello specifico saranno rappresentate quelle inerenti ad un numero di *features* pari a 90.

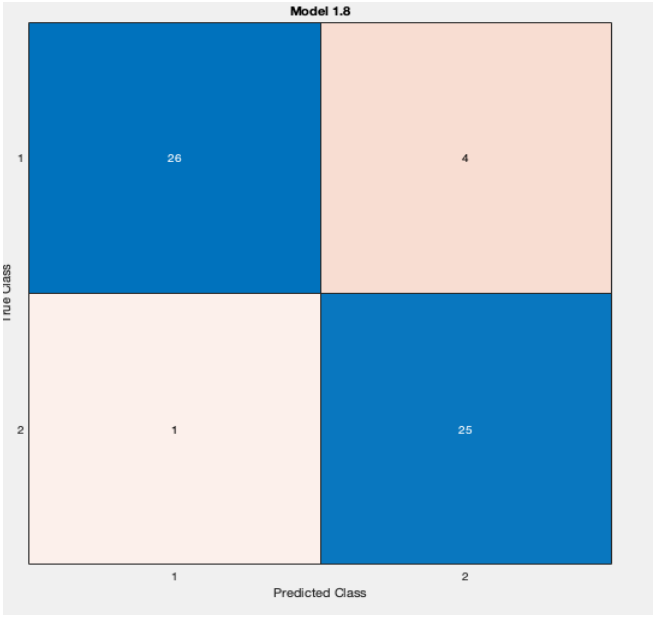
- SVM (quadratic)→ Data_matrix con Robust PCA



- SM (quadratic)→ Data_matrix



- NAIVE (kernel)→ Data_matrix_NEW



Dalle *confusion matrix* emerge di come, a parità di attributi, il risultato migliore sia relativo alla sperimentazione a cui viene applicata la Robust PCA alla matrice di dati originale, infatti dei pazienti affetti da morbo di Alzheimer solo 3 di essi vengono classificati come sani, al contrario questi attributi classificano correttamente i pazienti sani, non presentando alcun "*false negative*". I risultati peggiori, in questi termini, si riscontrano analizzando la *confusion matrix* relativa alla data_matrix original, in cui dei soggetti affetti da Alzheimer 5 ne vengono classificati come sani, in ambito clinico può tradursi in un ingente problema per la garanzia di diagnosi corretta.

L'ulteriore considerazione necessaria riguarda il *training time*, di seguito una tabella illustrante i tempi di lavoro degli algoritmi con valori di accuracy più significativi:

90° features	*Algorithm*	*Accuracy*	*Training Time*
Data_matrix	SVM(quadratic)	91,1%	0,28145 sec
Data_matrix_NEW	NAIVE(kernel)	91,1%	3,1902 sec
Data_matrix (Robust)	SVM(quadratic)	94,6%	0,25339 sec

4.2.2 Split for subjects

Come già discusso nei precedenti paragrafi, affinché le percentuali di accuracy potessero aumentare, ma principalmente al fine di una generalizzazione del modello per analisi future su nuovi dati, in tal caso per uno scenario clinico, la classificazione si è basata su una suddivisione per pazienti di train e test:

- Train → 9 pazienti.
- Test → 4 pazienti.

Allo stesso tempo si è usufruito della tecnica PCA e Robust PCA, modificando il precedente codice con l'aggiunta delle seguenti righe:

```
%split per subj
%13 soggetti tot:1,2,12,13 per test e restanti 9 per il train.
% 1:46 (1,2); 242:end (12,13) per il test.
% 47:241 (3,4,5,6,7,8,...11) per il train.

test_data = [M_new(1:46,:); M_new(242:end,:)];
train_data = M_new(47:241,:);


% save matrix
%save(strcat('Msubj_', int2str(num_feat), 'features.mat'), 'M_new');
save(strcat('M_trainsubjorig_', int2str(num_feat), 'features.mat'),
'train_data');
save(strcat('M_testsubjorig_', int2str(num_feat), 'features.mat'),
'test_data');
```

Successivamente si è proseguito con la classificazione attraverso il modello di train, e una volta esportato il modello la validazione è stata eseguita sul modello di test, nello specifico:

```
%predizione
x_test= M_new(:, 1:end-2);
y_test= M_new(:, end-1);
y_predict = predict(trainedModel.ClassificationNaiveBayes,x_test)
y_predict = round(y_predict)

%calcolo accuratezza
accuracy = (sum(y_test==y_predict) / numel(y_test))*100;

%matrice di confusione
cm = confusionchart(y_test,y_predict);
```

Usando le caratteristiche (i dati utilizzati per prevedere le etichette), vengono previste le etichette (i dati che vogliamo prevedere).

x_train → include tutte le variabili indipendenti, che saranno usate per addestrare il modello, in tal caso i dati relativi ai 9 pazienti sono stati impiegati per addestrare/adattare il modello.

x_test →Questa è la rimanente porzione delle variabili indipendenti dai dati che non è utilizzata nella fase di addestramento, ma per fare previsioni e testare l'accuratezza del modello.

y_predict → Rappresenta la variabile dipendente che deve essere prevista da questo modello, include le etichette di categoria rispetto variabili indipendenti, specificando la variabile dipendente durante la formazione/adattamento del modello.

y_test → Questi dati hanno delle etichette di categoria (le classi relative ai pazienti), queste etichette saranno utilizzate per testare l'accuratezza tra le categorie effettive e quelle previste.

Tab.10 -CLASSIFICAZIONE: ORIGINAL DATA MATRIX CON PCA - Split: train= 9 pazienti test=4 pazienti

PCA (n°features)	TREE (fine)	SVM (gaussian)	SVM (quadratic)	KNN (fine)	NAIVE (kernel)
50	100%	100%	98,8%	97,7%	98,8%
60	98,8%	100%	97,7%	97,7%	98,8%
70	97,7%	100%	96,6%	97,7%	98,9%
80	97,7%	100%	96,6%	97,7%	98,9%
90	97,7%	100%	96,6%	97,7%	98,9%
100	97,7%	100%	96,6%	97,7%	98,9%

Tab.11 -CLASSIFICAZIONE: DATA MATRIX SENZA OUTLIERS CON PCA - Split: train= 9 pazienti test=4 pazienti

PCA (n°features)	TREE (fine)	SVM (gaussian)	SVM (quadratic)	KNN (fine)	NAIVE (kernel)
50	96,6%	100%	100%	100%	98,8%
60	94,3%	100%	100%	97,7%	98,8%
70	94,3%	100%	98,8%	100%	98,8%
80	94,3%	100%	100%	98,8%	100%
90	94,3%	100%	98,8%	98,8%	100%
100	94,3%	100%	98,8%	98,8%	100%

Tab.12 -CLASSIFICAZIONE: ORIGINAL DATA MATRIX CON Robust PCA - Split: train= 9 pazienti test=4 pazienti

Robust PCA (n°features)	TREE (fine)	SVM (gaussian)	SVM (quadratic)	KNN (fine)	NAIVE (kernel)
50	98,9%	100%	100%	98,9%	98,9%
60	98,9%	100%	100%	98,9%	98,9%
70	98,9%	100%	97,6%	98,9%	98,9%
80	98,9%	100%	97,6%	98,9%	98,9%
90	98,9%	100%	97,6%	98,9%	98,9%
100	98,9%	100%	98,9%	98,9%	98,9%

Nel complesso le percentuali di accuracy delle tabelle 10,11 e 12, confrontate con le sperimentazioni precedenti, presentano un netto incremento, basti osservare le percentuali massime in relazione all'algoritmo SVM(gaussian).

Questo miglioramento generale, può essere esplicato in quanto combinando i soggetti, sani e malati, tra set di formazione e set di convalida (test_set), può aumentare artificialmente l'accuratezza della previsione.

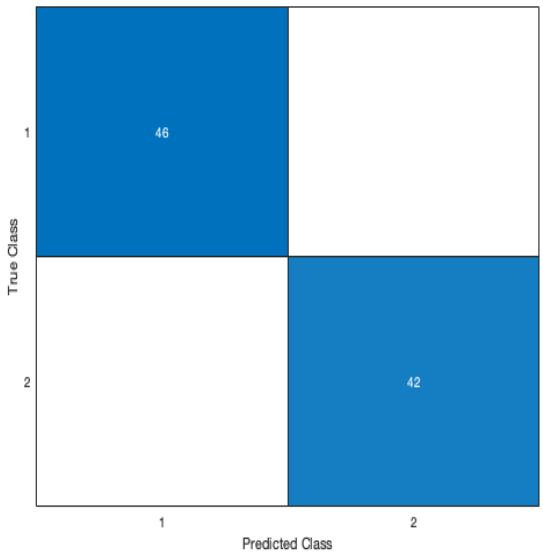
Si potrebbe pensare che i risultati della Tab.11 siano fuorvianti, in quanto presentano dei valori di accuracy, che oltre ad essere elevati (100%), sono più frequenti di quelli presenti nella Tab.10 e Tab.12, quindi si potrebbe affermare che probabilmente una scelta ottimale ricadrebbe sulla matrice di dati, cui sono stati rimossi gli outliers manualmente.

Chiaramente, come ulteriore epilogo di quanto scritto, l'eliminazione di valori ridondati, se fatta manualmente, non permette di generalizzare la procedura, ma al contrario la rende specifica di un determinato set di dati, rendendo la fase di *preprocessing*[32] lunga.

[32] La preelaborazione dei dati è quella fase in cui i dati vengono trasformati, o codificati, per portarli a uno stato tale che ora la macchina può facilmente analizzarli. In altre parole, le caratteristiche dei dati possono ora essere facilmente interpretate dall'algoritmo.

Spostando l'analisi sulla *confusion matrix* e *training time*:

- SVM(gaussian)→ Data_matrix con Robust PCA (90° features)[33]



Il modello generato non presenta errori, classifica correttamente sia i pazienti affetti da Alzheimer sia i pazienti sani.

90° features	*Algorithm*	*Accuracy train*	*Accuracy test*	*Training time*
Data_matrix	SVM(gaussian)	91,8%	100%	0,1283 sec
Data_matrix_NEW	SVM(gaussian)	98,5%	100%	0,1427 sec
Data_matrix(Robust)	SVM(gaussina)	91,8%	100%	0,12576 sec

[33] Le confusion matrix corrispondenti le altre due matrici sono identiche alla data matrix a cui è applicata la Robust PCA, quindi sono state omesse, alla luce delle stesse percentuali di accuracy, stesso algoritmo e stessi attributi.

CONCLUSIONI

L'obiettivo del lavoro discusso era quello di usufruire di alcune fra le tecniche di machine learning, relative alla task di classificazione, per poter individuare l'approccio migliore nella classificazione dei soggetti sani e soggetti affetti da morbo di Alzheimer, tramite l'elaborazione dei segnali elettroencefalografici.

Dunque in seguito ad una fase di preprocessing dei dati, inizialmente presentati in forma "grezza", alla generazione della matrice originale e della matrice cui sono stati rimossi i valori ridondanti, si è proseguito attraverso le varie sperimentazioni di classificazione, mediante poi l'utilizzo delle tecniche di *feature reduction* (PCA e Robust PCA).

Ciò che si è voluto evidenziare, conseguentemente alle varie sperimentazioni, si riflette in una valutazione circa l'affidabilità delle percentuali di accuracy, in relazione agli algoritmi proposti, ma per eventuali previsioni in ambito clinico non è stato l'unico fattore da considerare. Valutazioni sulla confusion matrix e training time sono state necessarie per determinare quale modello potesse essere più efficace per una corretta classificazione, ottenere un modello di analisi rapido ma non accurato, non ne determina l'efficacia, come allo stesso tempo avere accuratezza massima ma training time eccessivi.

Rispetto al metodo Holdout, lo suddivisione per soggetti, di cui alcuni facenti parte del train_data e alcuni del test_data, ha generato dei risultati ottimi sia in termini di accuratezza e velocità di analisi, ma soprattutto analizzando la *confusion matrix*, si può affermare che il modello generato dall'impiego dell'algoritmo di SVM(gaussian), classifica correttamente sia i soggetti sani che i soggetti malati.

Questo fa notare di come la scelta di un giusto metodo di validazione sia da non sottovalutare, perché in vista di una diagnosi precoce è imperativo ridurre al minimo possibile, l'errore di valutazione.

La perplessità circa i valori di accuracy massimi più frequenti della Tab.11 rispetto alla Tab.12, "cadono" nel momento in cui ci si sofferma a pensare che operare con un metodo innovativo come la Robust PCA permette di automatizzare il meccanismo di rimozione di valori ridondanti, che possono rappresentare esempi di istanze di dati rilevanti per il problema, determinando interpretazioni fuorvianti

dei dati raccolti. Ciò permette di trascurare i risultati ottenuti mostrati nelle Tab.10 e Tab.11, convenendo che un modello che meglio generalizza i dati è originato dalla classificazione con splitting per soggetto della data_matrix modificata con tecnica Robust PCA.

La capacità diagnostica dell'apprendimento automatico, può generare grandi vantaggi, per il trattamento di una malattia degenerativa come l'Alzheimer, aiutando i medici nella scelta della terapia più adatta, prima che la malattia abbia fatto il suo corso.

Ciò è reso possibile dalla rapidità con cui gli elaboratori aumentano la loro potenza di calcolo, quindi impiego di algoritmi più complessi eseguiti su quantità di dati innumerevoli ad elevata dimensionalità e solo mediante un giusto iter di convalida la trasformazione della medicina verso il machine learning può riscuotere effettivamente successo.

BIBLIOGRAFIA

[1] T. M. Mitchell, The Discipline of Machine Learning, Pittsburgh, PA 15213, USA: School of Computer Science Carnegie Mellon University, July 2006.

[2] A. Geitgey, «Machine learning is fun,» 5 may 2014.

[3] S. Asiri, «Machine Learning Classifiers,» 11 June 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623.

[4] S. Mutuvi, «Introduction to Machine Learning Model Evaluation,» [Online]. Available: https://heartbeat.fritz.ai/introduction-to-machine-learning-model-evaluation-fa859e1b2d7f.

[5] J. Ye, R. Janardan e L. Qi , «Two-Dimensional Linear Discriminant Analysis,» Vancouver, British Columbia, Canada, January 2004.

[6] I. T. Jolliffe e J. Cadima, «Principal component analysis: a review and recent developments,» *Phil. Trans. R. Soc. A.37420150202,* 19 January 2016 .

[7] T. Zhang e Y. Yang, «Robust PCA by Manifold Optimization,» 2018.

[8] E. J. Cand`es, X. Li, Y. Ma e J. Wright, «ROBUST PRINCIPAL COMPONENT ANALYSIS?,» December 17, 2009.

[9] S. Sanei e J. Chambers, in *EEG signal processing*, Cardiff University, UK, 2007.

[10] B. Kemp, A. Värri, A. C. Rosa , K. Nielsen e J. Gade , «A simple format for exchange of digitized polygraphic recordings. Electroencephalogr,» pp. 82(5):391-3. doi: 10.1016/0013-4694(92)90009-7. PMID: 1374708., 1992 May.

[11] O. Colpani, «Machine learning: the ability to predict applied to research and clinical practice».

[12] X. Yi, D. Park, Y. Chen e C. Caramanis, «Fast Algorithms for Robust PCA via Gradient Descent,» 19 settembre 2016.

SITOGRAFIA

1. https://lorenzogovoni.com/machine-learning-e-funzionamento/
2. https://www.thedifferentgroup.com/2020/03/20/tom-mitchell/#1-1
3. https://vitolavecchia.altervista.org/differenza-tra-apprendimento-supervisionato-non-supervisionato-e-con-rinforzo/
4. https://www.ai4business.it/intelligenza-artificiale/machine-learning/machine-learning-cosa-e-applicazioni/.
5. https://medium.com/@jacopokahl/i-tre-principali-tipi-di-machine-learning-77ca20d0dbdd.
6. https://www.deeplearningitalia.com/a-general-introduction-to-learning-methods-2/
7. https://www.geeksforgeeks.org/clustering-in-machine-learning/
8. http://www.andreaminini.com/ai/machine-learning/riduzione-dimensionalita-dati
9. https://www.mygreatlearning.com/blog/pattern-recognition-machine-learning/
10. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62
11. https://www.dezyre.com/article/top-10-machine-learning-algorithms/202
12. https://www.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html
13. https://lorenzogovoni.com/knn/
14. http://test.basel.in/product/knn-naive-bayes-classifier-using-excel/
15. https://www.webtutordimatematica.it/materie/statistica-e-probabilita/analisi-multivariata/analisi-delle-componenti-principali/pca-basata-sulla-matrice-di-correlazione
16. https://biomedicalcue.it/elettroencefalografia-cose-funziona/10469/
17. https://www.biorxiv.org/content/10.1101/059774v1.full