DEMENTIA PREDICTION-DOCUMENTAZIONE

Progetto Ingegneria Della Conoscenza

Autore: *Eraclea Vincenzo Pio, matricola 775561,*
v.eraclea@studenti.uniba.it

Professore del corso: *Fanizzi Nicola*

Anno di corso: *2024/2025*

Link GitHub: *https://github.com/Vincenzoera/ProgICON*

INDICE

INTRODUZIONE

‘Dementia Prediction’, è un progetto software che ha come obbiettivo la predizione di pazienti affetti da demenza senile. Questo sistema si basa sulla conoscenza per la classificazione del deterioramento cognitivo e della demenza senile, utilizzando modelli di machine learning supervisionati e non supervisionati.

LISTA ARGOMENTI

- **Apprendimento supervisionato** (DecisionTree, RandomForest, Knn, Mlp(ReteNeurale), LogisticRegression, Grid search, k-fold)
- **Apprendimento non supervisionato** (Clustering con K-Means)

LIBRERIE UTILIZZATE

- **Matplotlib** (Crea grafici e visualizzazioni personalizzabili per dati 2D,3D)
- **Numpy** (Fornisce array multidimensionali e operazioni su di essi, utile per calcoli numerici ad alte prestazioni)
- **Scikit-Learn** (Implementa algoritmi di machine learning e strumenti per analisi di dati come classificazione e clustering)
- **Pandas**(Gestisce e analizza dati. Gestisce manipolazioni avanzate di dati come con Dataframe e Series)
- **Plotly** (Utile per la creazione di Dashboard interattive tra cui heatmap, grafici a barre e istogrammi)
- **Seaborn** (Utile per analisi statistiche e esplorative è un integrazione avanzata con Pandas)

- Progetto sviluppato in Python 3.11

RACCOMANDAZIONI PER UN USO COMPLETO

! * LE DIPENDENZE IMPORTATE DA 'KERAS' COME 'TENSORFLOW', ALL'INTERNO DEL CODICE SUPPORTANO LE VERSIONI DI PYTHON FINO ALLA 3.11, IN CASO DI UNA VERSIONE SUPERIORE EFFETTUARE UN DOWNGRADE DELLA VERSIONE PER IL FUNZIONAMENTO CORRETTO. *!

Per il corretto funzionamento del sw, installare le versioni che ho rilasciato nel file requirements.txt, grazie.

INFORMAZIONI SUL DATASET

Il dataset utilizzato per l'addestramento del sistema è tratto dal sito www.kaggle.com ed è visualizzabile tramite il seguente link https://www.kaggle.com/code/obrienmitch94/alzheimer-s-analysis/report

Il dataset è basato su due file '**oasis_cross-sectional.csv**' e '**oasis_longitudinal.csv**'.

Dunque, si basa su una raccolta longitudinale di 150 soggetti di età compresa tra i 60 e 96 anni. Ciascun soggetto del database è stato sottoposto a risonanza magnetica pesata' in T1 in due o più visite, a distanza di almeno un anno, per un totale di 373 scansioni di imaging.

Ci sono 9 variabili/features binarie e 5 categoriali

Descrizione delle feature principali:

- **ID**: Identificativo del paziente
- **Group**: Stato della demenza
- **Visit**: Numero della visita effettuata
- **M/F**: Sesso del paziente
- **Hand**: Mano dominante
- **Age**: Età del soggetto
- **Educ**: Anni di istruzione ricevuti
- **SES**: Stato socio-economico
- **MMSE**: Risultato del Mini-Mental State Examination

- **CDR**: Classificazione della demenza clinica
- **eTIV**: Volume intracranico totale stimato
- **nWBV**: Volume cerebrale normalizzato
- **ASF**: Fattore di scala dell'atlante

DESCRIZIONE DELLE 5 FEATURES DI CATEGORIA (dettagliata)

Mini-Mental State Examination (MMSE)

Il **MMSE**, è un questionario di 30 domande ampiamente utilizzato in ambito clinico e di ricerca per valutare il deterioramento cognitivo e per il monitoraggio della demenza nel tempo. Può inoltre aiutare a misurare la risposta di un paziente a un trattamento.

Il punteggio massimo ottenibile è 30, e i risultati vengono interpretati come segue:

- **≥ 24**: Funzione cognitiva normale
- **19-23**: Deterioramento lieve
- **10-18**: Deterioramento moderato
- **≤ 9**: Deterioramento grave

È importante considerare che fattori come il livello di istruzione e l'età possono influenzare il punteggio.

Clinical Dementia Rating (CDR)

La scala **CDR** è un metodo di valutazione a cinque livelli che misura la compromissione cognitiva e funzionale in sei ambiti chiave:

1. Memoria
2. Orientamento
3. Giudizio e risoluzione dei problemi

4. Attività nella comunità
5. Vita domestica
6. Cura personale

L’assegnazione del punteggio avviene attraverso un'intervista semi-strutturata con il paziente e un familiare. I valori possibili sono:

- **0** → Nessun segno di demenza
- **0,5** → Segni iniziali di demenza
- **1** → Deterioramento lieve
- **2** → Deterioramento moderato
- **3** → Deterioramento grave

Volume intracranico totale stimato (eTIV)

L'**eTIV** rappresenta il volume stimato della cavità cranica e viene calcolato sulla base della materia cerebrale sopratentoriale. Poiché rimane costante nel tempo, è un parametro utile per studiare le variazioni strutturali cerebrali nei disturbi neurodegenerativi, tra cui Alzheimer e declino cognitivo.

Volume cerebrale normalizzato (nWBV)

nWBV indica la percentuale di tessuto cerebrale (materia grigia e bianca) presente rispetto al volume totale, calcolata tramite segmentazione automatica delle immagini.

Atlas Scaling Factor (ASF)

L'**ASF** è un coefficiente utilizzato per scalare le immagini cerebrali in relazione a un atlante di riferimento, permettendo il confronto tra soggetti diversi.

DESCRIZIONE DEL PROGETTO

Elaborazione dei dati

La variabile **Group (stato della demenza)** presenta tre possibili valori: *Nondemented*, *Demented* e *Converted*. Quest'ultimo si riferisce ai pazienti che, pur essendo inizialmente privi di sintomi di demenza senile, hanno successivamente sviluppato la patologia. Per semplificare l'analisi, questi soggetti vengono riclassificati come *Demented*.

Di conseguenza, i dati vengono suddivisi in due categorie:

- **0** → *Nondemented*
- **1** → *Demented*

Lo stesso processo di codifica binaria è stato applicato ad altre variabili:

- **M/F** (sesso) → *Maschio (0)*, *Femmina (1)*
- **Hand** (mano dominante) → *Destra (0)*, *Sinistra (1)*

Funzionamento e Analisi del Dataset

Visualizzazione immagini

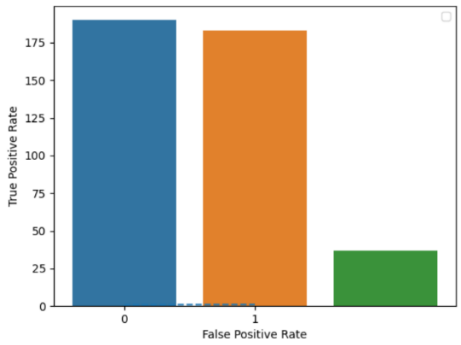


Fig.1

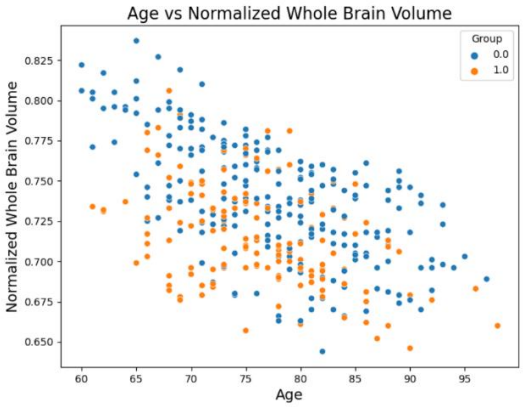


Fig.2

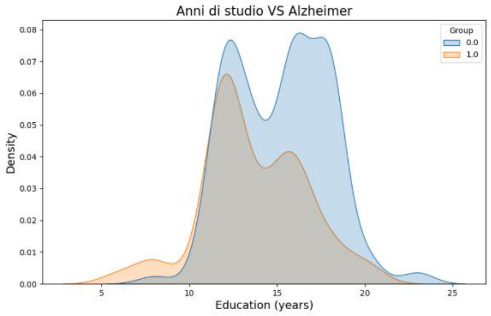


Fig.3

Come possiamo notare nelle figure in alto, una volta che ho determinato il dataset da usare è stato realizzato un istogramma (Fig.1), che contiene i casi distribuiti rispetto al dataset. Il risultato è che ci sono in maniera equivalente sia i pazienti affetti da demenza senile(arancio) sia i pazienti non affetti da demenza senile(blu), mentre gli ultimi(verde) sono i pazienti che sono stati convertiti come soggetti affetti da demenza.

Dato che la distribuzione dei dati è equa in tutto il dataset ho ricercato fattori adatti che riescano a caratterizzare un soggetto affetto da demenza senile, tramite l'utilizzo di grafici più specifici (Fig.2 & Fig.3). Raffigurando

così features più adatte da cui possiamo trarre un'affinità nel rilevamento della demenza senile, come l'eta , gli anni di studio o il suo lavoro.

Otteniamo infine un risultato che non rappresenta caratteristiche importanti per la demenza senile. Quindi ho generato un altro grafico, precisamente un Heatmap (Fig.4) per visualizzare i sintomi più comuni tra loro.

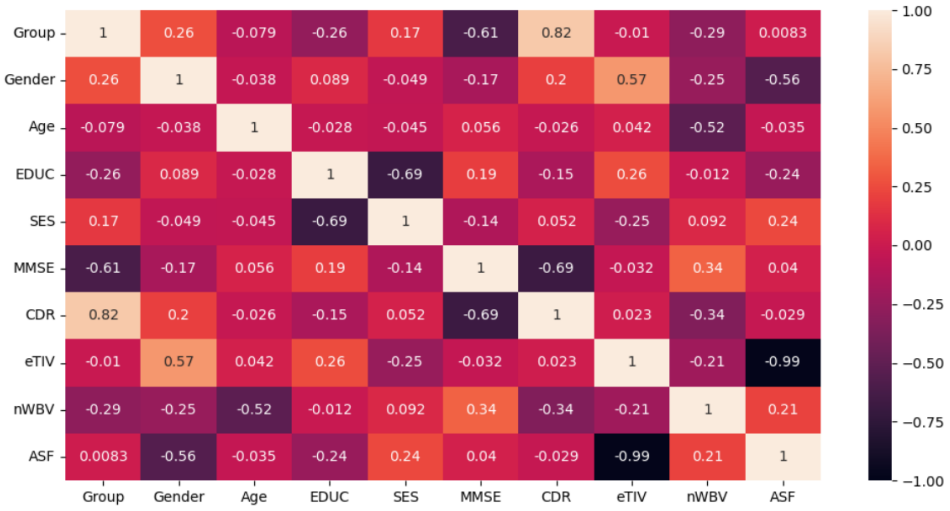


Fig.4

Il risultato dell'heatmap è ottenuto dai valori, dei features, indicati con '1', ovvero quelli altamente correlati tra loro che hanno corrispondenza veritiera e una classificazione migliore. L'obbiettivo però è quello di sviluppare un sistema di predizione e classificazione di soggetti affetti da demenza senile, mediante l'addestramento e l'uso di alcuni classificatori presenti nel machine learning tramite anche la ricerca dei migliori iper-parametri. Il fine è quello di poter far usare questo prototipo di software ad un medico, in grado così di sviluppare più prognosi.

Apprendimento Supervisionato

L'addestramento mediante gli algoritmi di classificazione, dell'apprendimento supervisionato, non comprende tutte le features descritte in precedenza, scelta effettuata per evitare valori nulli o obsoleti.

Le features scelte sono sesso,età,educazione,SES,MMSE,eTIV,nWBV,ASF.

Il Dataset comprende due fasi il training (75%) e il test (25) o meglio è stato diviso in 4 insiemi due per la fase di training e due per la fase di testing.

Gli algoritmi riportati per questo apprendimento sono:

Decision Tree, K-Nearest Neighbors, Logistic Regression, Random Forest, Rete Neurale(mlp)

Per la ricerca degli iperparametri invece si è fatto affidamento alla Grid Search che va a definire per ogni algoritmo una lista di quelli che sono gli attributi. La Grid Search testa l'algoritmo con ogni combinazione degli attributi specificati su di esso, effettuando una k-fold validation con un valore k = 5 (come profondità massima) e calcolando lo score di accuratezza relativo all'iterazione corrente.

Infine, dopo che le combinazioni sono state testate, per ogni algortimo, viene restituita la combinazione degli attributi con l'accuracy score maggiore.

Metriche e scelte di valutazione dei classificatori

Matrice di confusione e 0 – 1 Loss.

Classification Report per la visualizzazione di Precision, Recall, F1-score, Accuracy, Macro Average e Weighted Average di ogni classificatore.

Cross Validation Score per calcolare lo score della cross validation

Learning Curve per valutare l'accuratezza dei classificatori.

DECISION TREE

L'algoritmo **Decision Tree** ha un ruolo specifico, ogni nodo interno dell'albero indica una condizione e i valori derivati dagli esempi di input costituiscono dei 'sottoalberi'. Possiamo osservare come il valore di una 'feature obbiettivo ' venga classificato sulla base di regole di decisione, basate sui dati input che si inseriscono. Le foglie invece hanno il valore della 'feature obbiettivo '.

Per l'implementazione sul progetto ho fatto affidamento alla **Grid Search**, operazione effettuata precedentemente.

Ho stabilito quale fosse il miglior criterio tra **Entropy** e **Gini** (Metriche utilizzate negli alberi decisionali per misurare l'impurità di un nodo e decidere il miglior attributo su cui effettuare una suddivisione).

Il criterio scelto è stato **Gini** che permette di minimizzare la probabilità di classificazioni errate.

$$Gini = 1 - \sum_{i=1}^{c} p_i^2$$

La **Grid Search** ha suggerito una profondità massima di 14 nodi per l'albero, valori più alti non sono stati presi per evitare il problema dell'**overfitting** nella fase di test.

RISULTATI:

	WITHOUT FINE OPTIMIZATION	WITH FINE-OPTIMIZATION
TRAIN SCORE	0,79	1
TEST SCORE	0,73	0,86
DEVIAZIONE STANDARD	0,07	0,08
VARIANZA	0,004	0,007
0-1 LOSS	0,26	0,13
ACCURACY	**0,74**	**0,86**

Classification Report con ottimizzazione parametri:

	PRECISION	RECALL	F1 SCORE	SUPPORT
NONDEMENTED	0.86	0,86	0.86	35
DEMENTED	0.88	0.88	0.88	40
ACCURACY			0.77	94
MACRO AVG	0.77	0.77	0.77	94
WEIGHTED AVG	0.77	0.77	0.77	94

KNN

L'algoritmo **K-Nearest-Neighbors** (KNN), si basa sulla memorizzazione del Dataset che comprende la feature obbiettivo, senza apprendere un modello. Per la classificazione è semplice, si confronta il nuovo esempio con un insieme formato da k vicini che decreteranno la classe di appartenenza del nuovo esempio. Il decreto finale avviene come calcolo della media o a seguito dell'interpolazione dei k-vicini.

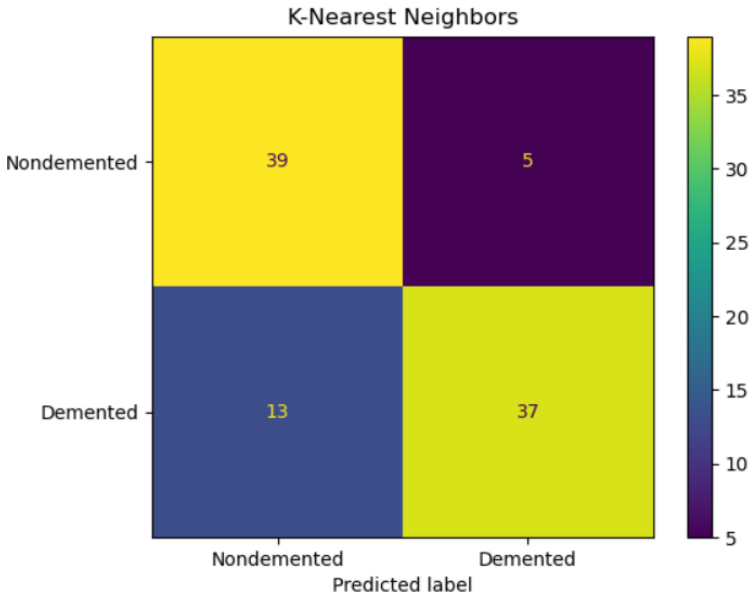
RISULTATI:

	WITHOUT FINE OPTIMIZATION	WITH FINE-OPTIMIZATION
TRAIN SCORE	0.71	1
TEST SCORE	0.78	0.80
DEVIAZIONE STANDARD	0.30	0.04
VARIANZA	0.0009	0.001
0-1 LOSS	0.21	0.19
ACCURACY	**0.79**	**0.81**

Classification Report con ottimizzazione parametri:

	PRECISION	RECALL	F1 SCORE	SUPPORT
NONDEMENTED	0.75	0.89	0.81	44
DEMENTED	0.88	0.84	0.80	50
ACCURACY			0.81	94
MACRO AVG	0.82	0.81	0.81	94
WEIGHTED AVG	0.82	0.81	0.81	94

Matrice di confusione per una maggior accuratezza:



RANDOM FOREST

L'algoritmo **Random Forest** è basato sull'addestramento di un numero N di alberi decisionali, dove ognuno effettua una classificazione per ogni esempio. Quando tutti gli alberi (della foresta) hanno classificato l'esempio, si effettua una conta e la classe che ha prodotto una stima maggiore viene presa come predizione 'della foresta'. Viene effettuato per tutti gli esempi.

Possiamo visualizzare ottimi risultati ottenuti tramite la Grid Search, infatti ripetendola anche per questo algoritmo abbiamo ottenuto che il numero più adatto(di estimatori) per la foresta è pari a 90. Per ogni albero che compone la foresta il miglior criterio utilizzato è '**Entropy'**, a differenza dell'algoritmo Decision Tree dove il miglior criterio era 'Gini'.

FORMULA:

$$Entropy = -\sum_{i=1}^{c} p_i \log_2 p_i$$

I parametri indicati come 'migliori' dalla Grid Search , sono: {'bootstrap': False, 'criterion': 'entropy' ', 'max_depth': 9, 'max_features': 'sqrt', 'n_estimators': 90}

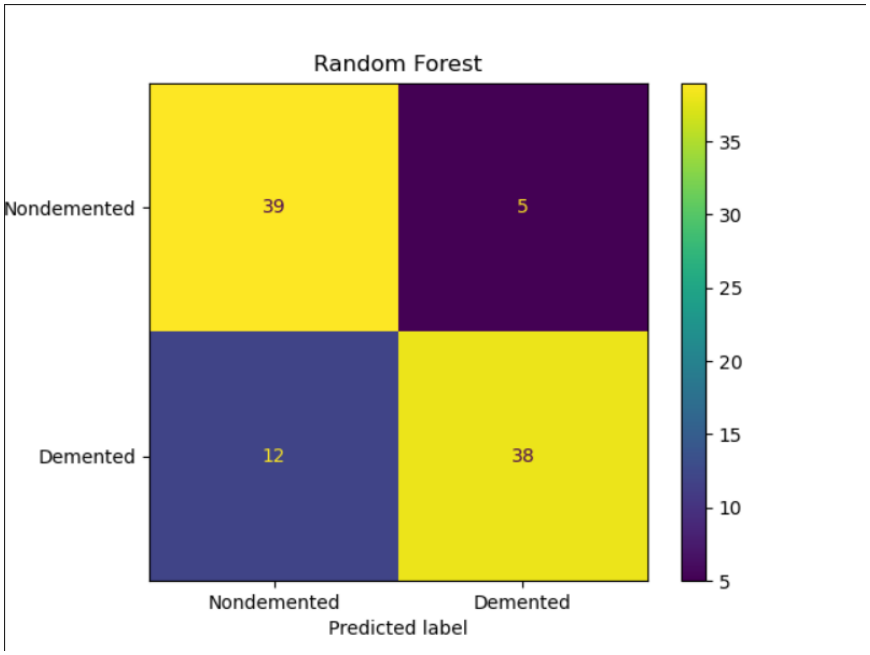
RISULTATI:

	WITHOUT FINE OPTIMIZATION	WITH FINE-OPTIMIZATION
TRAIN SCORE	0.87	1
TEST SCORE	0.84	0.86
DEVIAZIONE STANDARD	0.10	0.08
VARIANZA	0.01	0.007
0-1 LOSS	0.18	0.13
ACCURACY	0.81	0.86

Classification Report con ottimizzazione parametri:

	PRECISION	RECALL	F1 SCORE	SUPPORT
NONDEMENTED	0.80	0.93	0.86	44
DEMENTED	0.93	0.80	0.86	50
ACCURACY			0.86	94
MACRO AVG	0.87	0.87	0.86	94
WEIGHTED AVG	0.87	0.86	0.86	94

Matrice di confusione per una maggior accuratezza:



LOGISTIC REGRESSION

L'algoritmo di classificazione, di regressione logistica, è basato sui pesi di una funzione lineare, appiattita dalle sigmoidee, minimizzando un tipo di errore su E. Posso evidenziare il fatto che parte come un modello di regressione per poi venire appiattito con la log loss.

Funzionamento specifico dell'algoritmo (TEORIA)

- **Inizia come un modello lineare**, combinando i pesi con le feature.
- **Viene trasformato dalla sigmoide**, che lo appiattisce e lo rende interpretabile come probabilità. (tra 0 e 1)
- **Minimizza la Log Loss**, che misura quanto le previsioni si discostano dalle classi vere.
- **Usa la discesa del gradiente** per aggiornare i pesi e migliorare la classificazione.

Nel progetto sw, i parametri migliori sono anche qui indicati con la Grid Searchd, e sono: {'bootstrap': False, 'criterion': 'entropy', 'max_depth': 9, 'max_features': 'sqrt', 'n_estimators': 90}

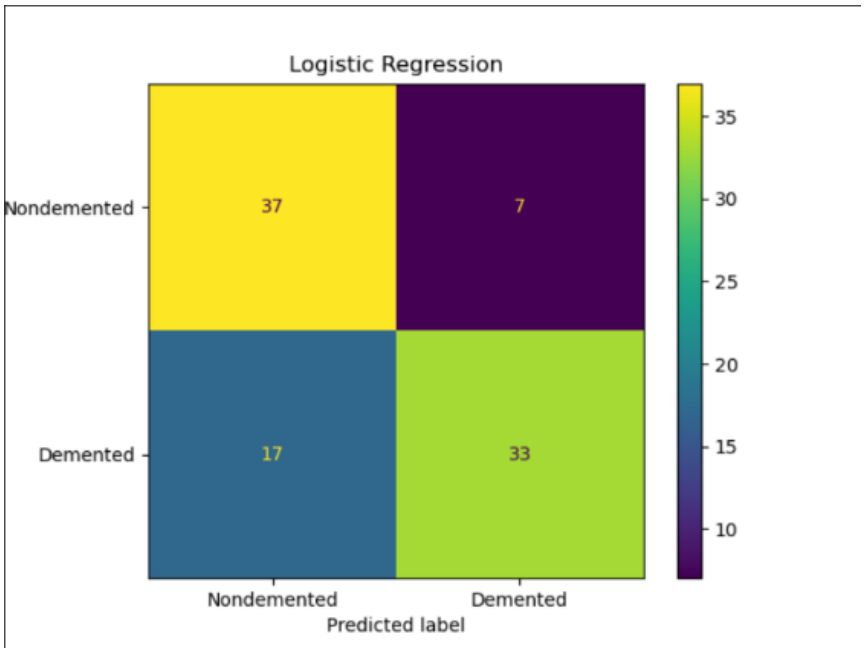
RISULTATI:

	WITHOUT FINE OPTIMIZATION	WITH FINE-OPTIMIZATION
TRAIN SCORE	0.82	0.82
TEST SCORE	0.81	0.74
DEVIAZIONE STANDARD	0.10	0.09
VARIANZA	0.01	0.009
0-1 LOSS	0.18	0.25
ACCURACY	0.81	0.75

Classification Report con ottimizzazione parametri:

	PRECISION	RECALL	F1 SCORE	SUPPORT
NONDEMENTED	0.69	0.84	0.76	44
DEMENTED	0.82	0.66	0.73	50
ACCURACY			0.74	94
MACRO AVG	0.76	0.75	0.74	94
WEIGHTED AVG	0.76	0.76	0.74	94

Matrice di confusione per una maggior accuratezza:



RETE NEURALE (MLP)

Una **Rete Neurale** è un modello di apprendimento supervisionato, ispirato al funzionamento dei neuroni biologici. Per il nostro problema, abbiamo utilizzato una **Rete Neurale Feed-Forward**, caratterizzata da una struttura gerarchica in cui le trasformazioni lineari sono intervallate da funzioni di attivazione non lineari (**MULTI-LAYER-PERCEPTRON**).

Il modello riceve in input un insieme di feature, che vengono elaborate attraverso **strati nascosti** (detti anche feature non osservate). Questi strati

applicano funzioni di attivazione per apprendere rappresentazioni complesse dei dati, restituendo infine uno o più valori di output corrispondenti alle feature obiettivo. Gli strati o layer presenti sono divisi come:

-un **layer** che contiene tutte le feature che rappresentano i sintomi (layer d'input.

-un **layer** lineare di cui il modello matematico è rappresentato dalla seguente formula:

$$y = \sum_i w_i * x_i$$

Dove y è la classe da predire, x è il numero di feature in input e w sono i pesi da assegnare ad ogni feature di input.

-Una funzione **f** (di attivazione).

L'utilizzo di una rete neurale è fondamentale nel caso si disponga di molti dati(adatta per questo modello), dato però l'elevato costo computazionale mi sono limitato ad effettuare test per la ricerca della miglior funzione di attivazione per la nostra rete.

***Si fa uso della Back-propagation avere una classificazione accurata.**

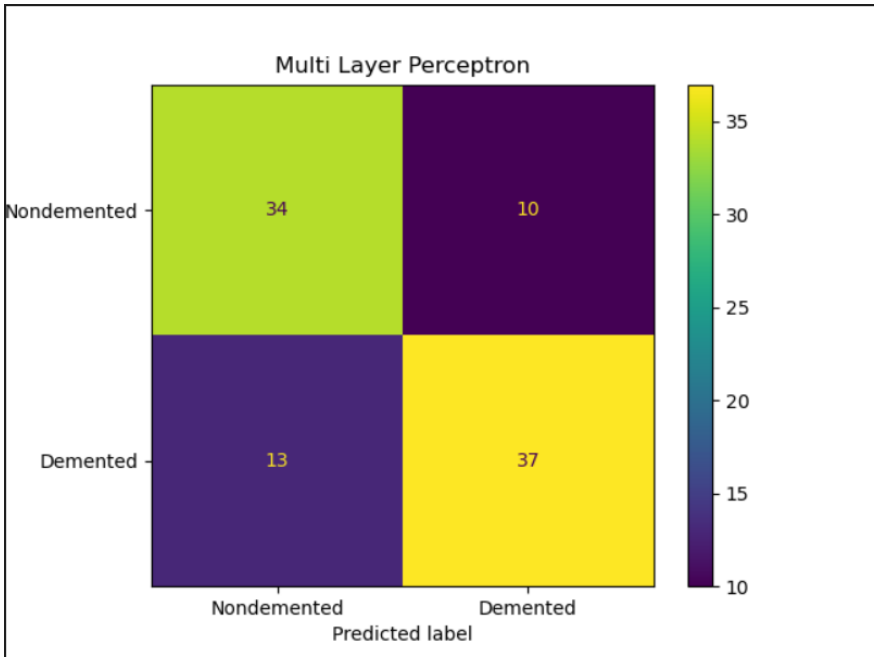
RISULTATI:

	WITHOUT FINE OPTIMIZATION	WITH FINE-OPTIMIZATION
TRAIN SCORE	0.87	0.89
TEST SCORE	0.76	0.77
DEVIAZIONE STANDARD	0.10	0.015
VARIANZA	0.01	0.0002
0-1 LOSS	0.18	0.22
ACCURACY	0.81	0.77

Classification Report con ottimizzazione parametri:

	PRECISION	RECALL	F1 SCORE	SUPPORT
NONDEMENTED	0.73	0.82	0.77	44
DEMENTED	0.82	0.74	0.78	50
ACCURACY			0.74	94
MACRO AVG	0.78	0.78	0.78	94
WEIGHTED AVG	0.78	0.78	0.78	94

Matrice di confusione per una maggior accuratezza:



Apprendimento Non Supervisionato

Si parla di apprendimento non supervisionato, quando le feature obbiettivo non sono presenti negli esempi di training e i dati non sono etichettati.

K-MEANS

L'algoritmo **K-Means** sfrutta il 'clustering', quindi separa i dati in cluster distinti che non sono stati etichettati nei dati. I punti dati possono appartenere ad un solo cluster. Ad ogni cluster viene assegnata un'etichetta al punto dati.

Nella prima fase, scalando il dataset ho effettuato un processo di standardizzazione dei dati così da avere solo dati che variano nel range di -1 e 1. (Fig.5)

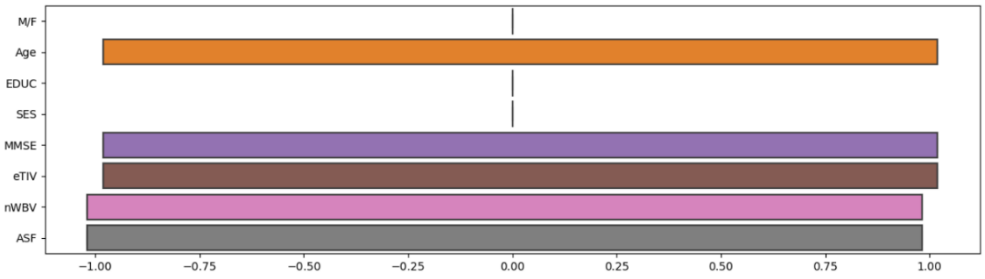
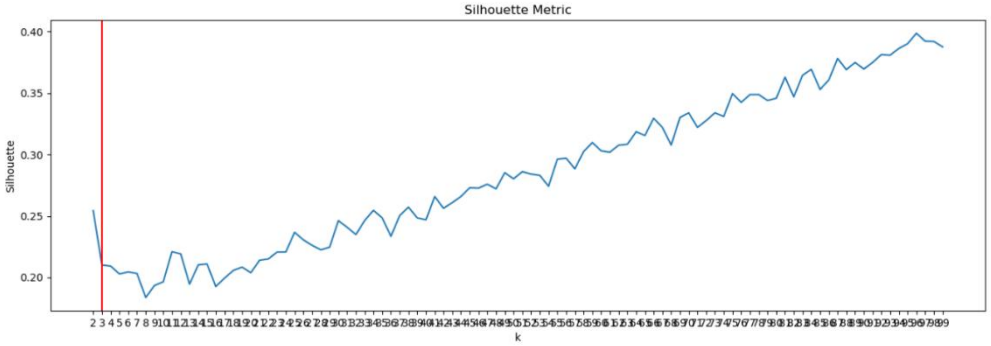


FIGURA 5

Dopo aver standardizzato i dati si sceglie un numero di cluster che si vuole identificare (es k = 3) e si scelgono 3 valori che rappresentano i clusters iniziali. Infine, si misura la distanza tra il primo punto nel dataset e i tre cluster iniziali, a chi ha distanza minore viene assegnato quel dataset ad un cluster, questa operazione si ripete per ogni punto. In uno spazio a due dimensioni, per ogni cluster si trova il centroide (si fa la media della distanza dei punti assegnati al singolo cluster j).

Per trovare il giusto valore **K** (usato da questo algoritmo, K-Means) ho utilizzato una metrica chiama **silhouette Index**, che indica quando un cluster è ‘puro’.

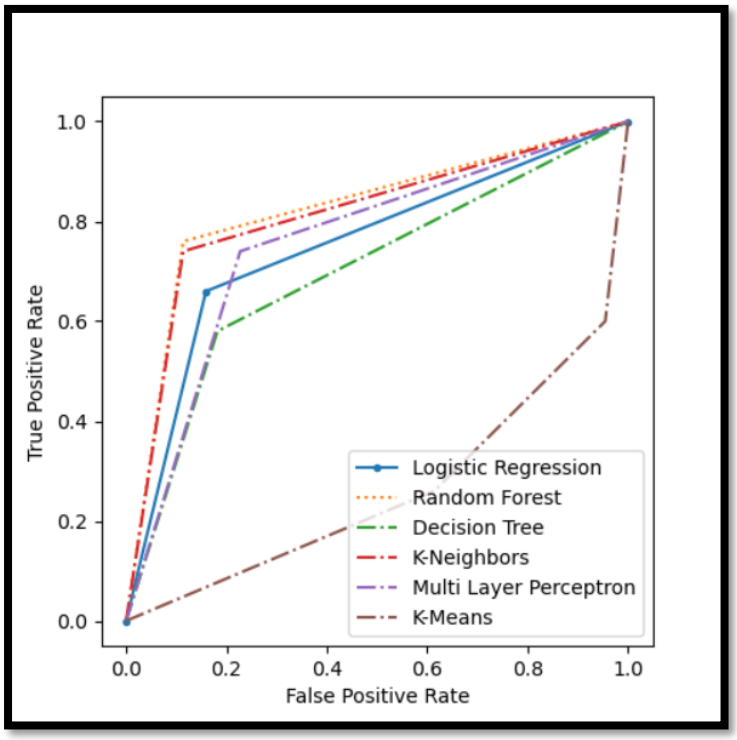
VALORE K CON METODO SILHOUETTE INDEX: K=3



Classification Report con ottimizzazione del K:

	PRECISION	RECALL	F1 SCORE	SUPPORT
NONDEMENTED	0.50	0.30	0.37	44
DEMENTED	0.44	0.46	0.45	50
ACCURACY			0.38	94
MACRO AVG	0.31	0.25	0.27	94
WEIGHTED AVG	0.47	0.38	0.41	94

CONCLUSIONI APPRENDIMENTI



Come possiamo osservare dal grafico finale, che compara tutti gli algoritmi di classificazioni utilizzati, il **Random Forest** e il **KNN** hanno un apprendimento migliore e rapido rispetto agli altri. Apprendono fin dall'inizio restituendo un minore numero di falsi positivi. Anche gli algoritmi di **Logistic Regression** e la **rete neurale (MLP)** seguono una traiettoria buona anche se tendono a sbagliare la fase iniziale però infine restituiscono anche loro un minor numero di falsi positivi. L'algoritmo **Decision Tree** invece ha un apprendimento molto lento rispetto quest'ultimi.

Per l'algoritmo **K-Means** invece, appartenente alla classe di algoritmi di apprendimento non supervisionato, possiamo notare come ha un apprendimento molto lento e non preciso sbagliando la fase iniziale e iniziandosi ad affinare solo alla fine dei nostri test effettuati.

UTILIZZO DEL PROGETTO CON CLI

L'utilizzo del software prevede un menù, con il quale è possibile tramite CLI creare modelli di predizione, attraverso i due tipi di apprendimento visualizzati in precedenza per predire se un paziente è affetto da demenza senile o meno, ricercando inoltre i parametri migliori attraverso la Grid Search. Inoltre, inserendo in input i valori del paziente tramite la funzione 'run test' (per semplicità in questo punto ho gestito un array statico che prende i dati in input) è possibile predire se un paziente è affetto da demenza senile. Il risultato finale dell'esito ricade dalla maggioranza dei classificatori che esprimeranno ciascuno un giudizio pari o a 0 oppure a 1.

Dove con 1 viene espresso il risultato come 'paziente affetto da demenza', quindi demented e con 0 'paziente non affetto da demenza', quindi non demented.

Le scelte le possiamo effettuare tramite il menu visualizzato. In basso ci sono alcuni esempi di scelte.

Menù

```
Alzheimer prediction

1 -- Learning Supervised
2 -- Learning Unsupervised
3 -- Optimize parameters
4 -- Run test
5 -- Exit
Enter your choice:
```

Esempio scelta = 1: utilizzo apprendimento supervisionato con Random Forest (Migliore)

```
-- RANDOM FOREST CLASSIFIER
Train score:  1.0
Test score:  0.8191489361702128
Cross Validated Score:  0.8461538461538461
Standard Deviation:  0.0908838379258696
Variance:  0.008259871996135733
0-1 Loss:  0.18085106382978722
Accuracy: 0.8231818181818182
              precision    recall  f1-score   support

           0       0.76      0.89      0.82        44
           1       0.88      0.76      0.82        50

    accuracy                           0.82        94
   macro avg       0.82      0.82      0.82        94
weighted avg       0.83      0.82      0.82        94
```

Esempio scelta = 3: OPTIMIZE PARAMS attraverso la Grid search e il K per il K-Means

```
BEST PARAMS:
- DTC: {'criterion': 'entropy', 'max_depth': 14} - 1.9423 seconds
- KNN: {'algorithm': 'kd_tree', 'n_neighbors': 5, 'p': 2, 'weights': 'distance'} - 0.3865 seconds
- RFC: {'bootstrap': True, 'criterion': 'gini', 'max_depth': 8, 'max_features': 'sqrt', 'n_estimators': 100} - 10.6772 seconds
- MLP: {'activation': 'relu'} - 0.4813 seconds
- LR: {'C': 5, 'penalty': 'l2'} - 0.1228 seconds
- kMeans: 0.1886 - 4.3946 seconds
-- Fine optimization end in 18.0047 seconds
```

Esempio scelta = 4: RUNT TEST attraverso un array statico

Parametri dell'array

```
# 'M/F', 'Age', 'EDUC', 'SES', 'MMSE', 'eTIV', 'nWBV', 'ASF'
xx = [[0, 87, 14, 2, 27, 1987, 0.696, 0.883]]  # real: nondemented 1
xy = [[1, 80, 12, 2.0, 22.0, 1698, 0.701, 1.034]]  # real: demented 4
```

OUTPUT GENERATO CASO 1 'DEMENTED'

```
-- Loaded trainend models in 0.0068 seconds

TEST:  [1, 80, 12, 2.0, 22.0, 1698, 0.701, 1.034]

   DCT  KN  RaF  MLP  LR  KMS
0    1   1    1    1   1    0
The most predicted disease is demented

-- Predicted in 0.0344 seconds
```

OUTPUT GENERATO CASO 2 'NON DEMENTED'

```
-- Loaded trainend models in 0.0102 seconds

TEST:  [0, 87, 14, 2, 27, 1987, 0.696, 0.883]

   DCT  KN  RaF  MLP  LR  KMS
0    0   0    0    1   0    1
The most predicted disease is nondemented

-- Predicted in 0.0152 seconds
```

SVILUPPI FUTURI

Come sviluppi futuri ho pensato alla gestione di un array dinamico e un’implementazione anche di una base di conoscenza attraverso le due logiche, per renderlo migliore dal punto di vista della conoscenza.