UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO
DIPARTIMENTO DI INFORMATICA

Sistema di diagnosi di un attacco di cuore

Ingegneria della conoscenza

AA 2022-23

Gruppo di lavoro

Flaviana Corallo, 736356, f.corallo8@studenti.uniba.it

URL Repository: https://github.com/CoralloFlaviana/Icon22-23

Sommarioo

Introduzione 3
Elenco argomenti di interesse 3
Analisi dei dati 4
Sommario 4
Strumenti utilizzati 4
Knowledge Base 7
Sommario e Strumenti utilizzati 7
Decisioni di Progetto 7
Apprendimento supervisionato 10
Sommario 10
Decisioni di Progetto 10
Decision Tree 11
Random Forest 13
KNN 15
Conclusioni 17
Apprendimento non supervisionato 18
Sommario 18
Decisioni di Progetto 18
Valutazione 18
Riferimenti Bibliografici 19

Introduzione

I problemi cardiaci nel mondo e in medicina rappresentano una sfida significativa. Le malattie cardiovascolari sono la principale causa di morte a livello globale, contribuendo al 32% delle morti nel 2020 secondo l'OMS. Queste condizioni colpiscono individui di tutte le età, sebbene siano più comuni tra gli anziani.

Diversi fattori di rischio contribuiscono allo sviluppo dei problemi cardiaci, tra cui l'età avanzata, il fumo, l'obesità, l'ipertensione, il diabete e la mancanza di attività fisica. La gestione di questi fattori è cruciale per la prevenzione.

La medicina ha compiuto notevoli progressi nella diagnosi dei problemi cardiaci grazie a moderne tecniche di imaging cardiaco e test del sangue per biomarcatori specifici.

La prevenzione rimane cruciale attraverso campagne di sensibilizzazione, programmi di educazione alimentare e promozione di uno stile di vita sano.

La tecnologia sta rivoluzionando la gestione dei problemi cardiaci attraverso la telemedicina e l'uso dell'intelligenza artificiale per la diagnosi e la gestione delle condizioni cardiache.

Elenco argomenti di interesse

- Apprendimento supervisionato per la diagnosi dell’attacco di cuore
    - Apprendimento supervisionato con iperparametri per la diagnosi dell’attacco di cuore
- Knowledge Base è stata creata una base di conoscenza per condurre analisi delle informazioni contenute nel dataset, al fine di dedurre in base ai parametri di un paziente se esso potrebbe avere un attacco cardiaco, utilizzando opportune query
- Clustering una tecnica di apprendimento non supervisionato utilizzato per raggruppare i pazienti in diverse categorie

Analisi dei dati

Sommario

Prima di iniziare a lavorare sul sistema di diagnosi di un attacco di cuore è stato cruciale eseguire alcune analisi sui dati contenute nel dataset rinominato da me “heart” [6].

Importante conoscere alcuni dettagli sul dataset prima di proseguire l’analisi, il dataset è così composto:

- age: Età del paziente.
- sex: Genere del paziente (1 = maschio, 0 = femmina).
- cp: Tipo di dolore toracico (0 = angina tipica, 1 = angina atipica, 2 = dolore non correlato all'angina, 3 = asintomatico).
- TRTBPS: Pressione sanguigna a riposo (in mmHg).
- CHOL: Livello di colesterolo in mg/dl, determinato utilizzando un sensore BMI.
- fbs: Zucchero nel sangue a digiuno (> 120 mg/dl) (1 = vero, 0 = falso).
- restecg: Risultati dell'elettrocardiografia a riposo (0 = normale, 1 = normalità dell'onda ST-T, 2 = ipertrofia del ventricolo sinistro).
- thalachh: Frequenza cardiaca massima raggiunta.
- exng: Angina pectoris causata dall'esercizio (1 = sì, 0 = no).
- old peak: Picco precedente.
- slp: Pendenza.
- caa: Numero di grandi vasi.
- thall: Risultato del test Thallium ~ (0.3).
- output: Variabile target.

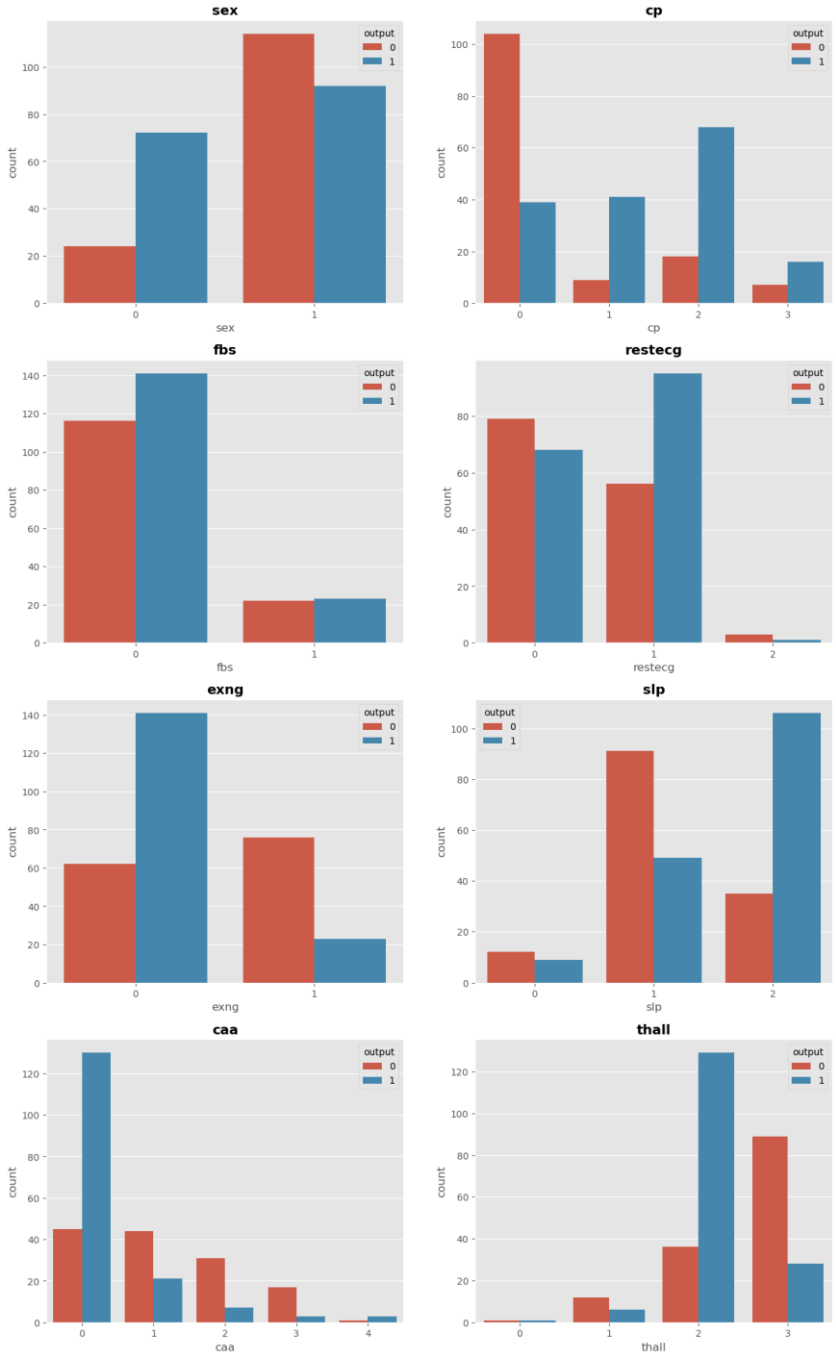
Strumenti utilizzati

Tramite l’utilizzo di "EDA", in Python è l'abbreviazione di "Exploratory Data Analysis" (Analisi Esplorativa dei Dati), è stato rilevata la presenza di righe duplicate e sono state eliminate.

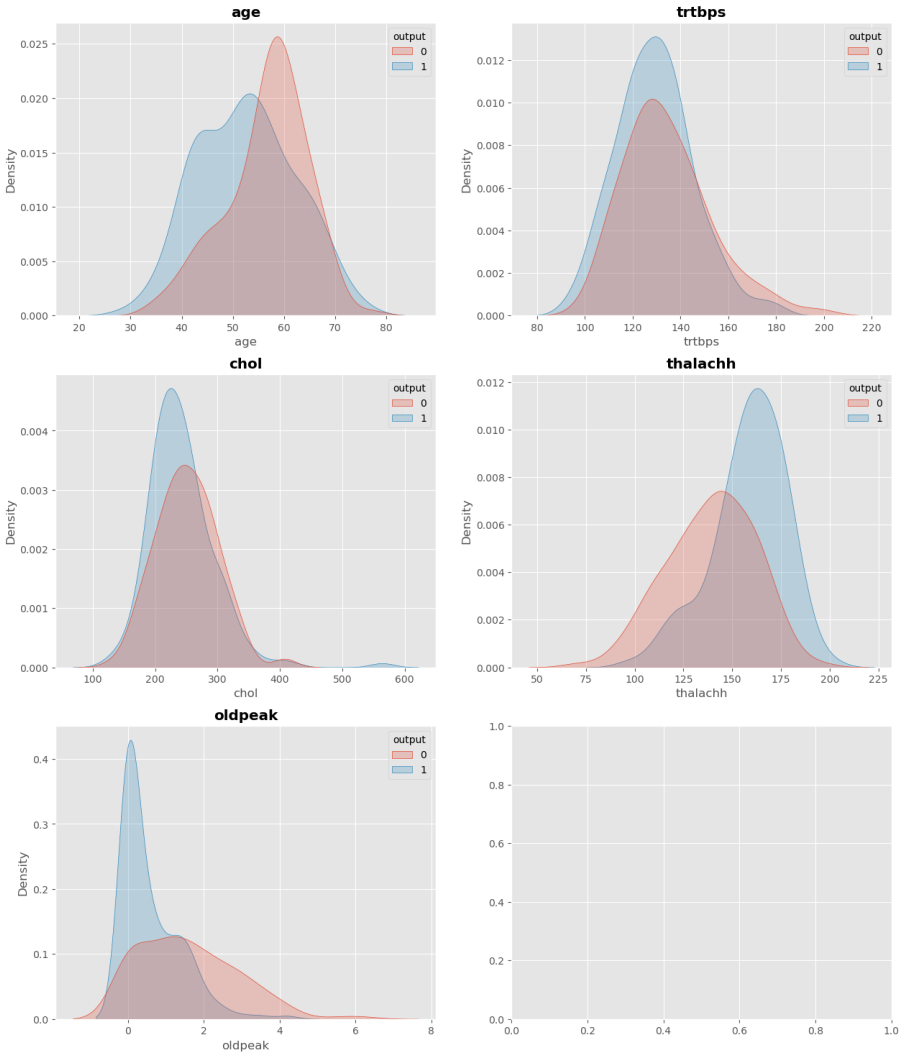
In seguito, una breve analisi dei dati.

- Sex (Sesso): La variabile "sesso" sembra avere una forte relazione con la presenza di attacchi cardiaci. Le donne hanno meno probabilità di avere attacchi cardiaci, mentre gli uomini hanno una probabilità significativamente maggiore.
- CP (Tipo di Dolore Toracico): La variabile "tipo di dolore toracico" mostra che la maggior parte delle persone con "angina tipica" non ha attacchi cardiaci. Al contrario, in altre categorie di dolore toracico (diverse da quella tipica), la presenza di attacchi cardiaci è più comune.
- Exng (Angina Pectoris Causata dall'Esercizio): Se l'angina pectoris è causata da sforzo fisico, la maggior parte delle persone nel dataset non ha attacchi cardiaci. Al contrario, se l'angina pectoris non è associata all'esercizio fisico, un numero significativo di persone ha avuto attacchi cardiaci.
- Slp (Pendenza): La variabile "pendenza" mostra che con uno slp uguale a 0, la classe delle malattie cardiache è approssimativamente bilanciata. Uno slp pari a 1 è associato a una predominanza di persone senza attacchi cardiaci, mentre uno slp pari a 2 è associato a una predominanza di persone con attacchi cardiaci.

- Caa (Numero di Grandi Vasi): L'aumento del numero di grandi vasi (escluso il valore 4) sembra essere correlato a una minore incidenza di attacchi cardiaci. Le persone con 1, 2 o 3 grandi vasi hanno meno probabilità di avere attacchi cardiaci rispetto a quelle senza grandi vasi o con 4 grandi vasi.
- Thall (Risultato del Test Thallium): La variabile "thall" mostra che la maggior parte dei valori si trova nei gruppi 2 e 3, e questi gruppi suddividono bene la variabile target. Nel gruppo con risultato 2, prevalgono le persone con attacchi cardiaci, mentre nel gruppo con risultato 3, prevalgono le persone senza attacchi cardiaci.



- Età (Age): Le persone comprese tra i 30 e i 60 anni sembrano essere più inclini a soffrire di attacchi cardiaci. D'altra parte, le persone più anziane, di età superiore ai 50 anni, sembrano avere meno probabilità di avere attacchi cardiaci.
- Thalachh (Frequenza Cardiaca Massima): Questa variabile mostra una buona separazione rispetto alla variabile target. L'aumento della frequenza cardiaca è associato a un aumento del rischio di attacchi cardiaci. Al di sotto del valore di 130, la maggior parte delle persone non soffre di attacchi cardiaci, ma oltre questo valore, il rischio di attacco cardiaco aumenta notevolmente.
- Oldpeak (Picco Precedente): Anche questa variabile sembra essere una buona separazione rispetto alla variabile target. La maggior parte dei valori si concentra sui minimi, ma la distribuzione ha una coda verso destra. Le persone con un valore basso di "oldpeak" (più vicino al minimo) tendono a soffrire principalmente di attacchi cardiaci. Tuttavia, con un aumento dei valori di "oldpeak", la probabilità di avere un attacco cardiaco diminuisce e la probabilità di non averne uno aumenta, specialmente a valori più elevati.



In breve, risultano rilevanti le seguenti caratteristiche: la frequenza cardiaca massima (thalachh) e il picco precedente (oldpeak), Caa (Numero di Grandi Vasi), Thall (Risultato del Test Thallium), Sex(sesso), CP (Tipo di Dolore Toracico), Exng (Angina Pectoris Causata dall'Esercizio), Slp (Pendenza).

Queste saranno le caratteristiche che useremo nel nostro sistema di diagnosi.

Knowledge Base

Sommario e Strumenti utilizzati

Una base di conoscenza in logica del primo ordine è un insieme di proposizioni, dette assiomi, che sono considerate vere senza bisogno di dimostrazione. Questa base di conoscenza viene utilizzata per rappresentare la conoscenza di un particolare dominio all'interno di un sistema o una macchina.

Il processo di creazione di una base di conoscenza coinvolge i seguenti passaggi:

- **Definizione del Dominio**: Iniziamo decidendo il dominio che desideriamo rappresentare. Questo può essere il mondo reale, un mondo immaginario o un mondo astratto, come numeri o insiemi.
- **Definizione delle Proposizioni Atomiche**: Successivamente, identifichiamo le proposizioni atomiche che serviranno a rappresentare il nostro dominio. Queste proposizioni sono gli elementi di base con cui costruiremo la nostra base di conoscenza.
- **Assiomatizzazione del Dominio**: Qui definiamo le proposizioni che saranno considerate vere nell'interpretazione del dominio. Queste proposizioni costituiranno gli assiomi della base di conoscenza e rappresenteranno le verità fondamentali del nostro dominio.
- **Interrogazione del Sistema**: Una volta definita la base di conoscenza, possiamo porre domande o query al sistema. Il sistema determinerà se specifiche proposizioni sono conseguenze logiche della base di conoscenza, ossia se sono vere in tutti i modelli della base di conoscenza.

Il sistema stesso, a differenza del progettista, non comprende il significato dei simboli utilizzati nella base di conoscenza. Il sistema si basa sugli assiomi definiti e può decidere se una particolare proposizione è una conseguenza logica della base di conoscenza. Spetta al progettista, in base all'interpretazione del dominio, valutare se il risultato ottenuto dal sistema è valido o coerente con la conoscenza del dominio.

Per implementare una base di conoscenza in logica del primo ordine, è stata utilizzata il linguaggio di programmazione Prolog con la libreria pyswip. Le caratteristiche selezionate sono state rappresentate come fatti nella base di conoscenza, con la definizione dei loro domini.

Decisioni di Progetto

Di seguito alcune regole implementate in Prolog grazie le quali l'utente può porre query al sistema.

```
%Regola 1: per determinare se una persona può avere un attacco cardiaco.
puo_avere_attacco_cardiaco(Età, TipoDolore, AnginaEsercizio, Pendenza, NumeroVasi, RisultatoThallium, FrequenzaCardiaca, Picco
Precedente, PuoAvereAttacco) :-

    % Condizioni per non avere un attacco cardiaco
    not((
        TipoDolore == 0, % Angina tipica
        AnginaEsercizio == 1, % Angina pectoris causata dall'esercizio
        (Pendenza == 0; Pendenza == 1), % Pendenza pari a 0 o 1
        NumeroVasi =< 3, % Numero di grandi vasi minore o uguale a 3
        RisultatoThallium == 3, % Risultato del test al thallium uguale a 3
        (Età > 50; Età < 30), % Età maggiore di 50 o minore di 30
        FrequenzaCardiaca < 130, % Frequenza cardiaca massima minore di 130
        PiccoPrecedente > 2.0 % Picco precedente alto
    )),

    % Se non soddisfa le condizioni, allora può avere un attacco cardiaco
    PuoAvereAttacco = no.
```

Questa regola Prolog è stata creata per determinare se una persona può avere un attacco cardiaco in base a diversi attributi forniti come input. La regola considera una serie di condizioni che, se soddisfatte, indicano che la persona potrebbe avere un attacco cardiaco. In caso contrario, se le

condizioni non sono soddisfatte, la variabile PuoAvereAttacco viene impostata su "no", indicando che la persona non può avere un attacco cardiaco.

Le condizioni che vengono verificate includono il tipo di dolore toracico, se l'angina è causata dall'esercizio fisico, la pendenza dell'ECG, il numero di grandi vasi, il risultato del test al thallium, l'età, la frequenza cardiaca massima e il picco precedente. Se una persona soddisfa tutte queste condizioni (cioè nessuna delle condizioni negate è vera), allora PuoAvereAttacco sarà impostato su "no attacco cardiaco", altrimenti potrebbe avere un attacco cardiaco.

Tutti i valori considerati sono stati dedotti dall'analisi dei dati del dataset.

```
% Regola 2: per determinare se una persona può avere un attacco
% cardiaco, usando una somma. di valori.
puo_avere_attacco_cardiaco_prob(Età, TipoDolore, AnginaEsercizio, Pendenza, NumeroVasi, RisultatoThallium, FrequenzaCardiaca,
PiccoPrecedente, PuoAvereAttacco) :-

    % Condizioni per non avere un attacco cardiaco
    (
        (TipoDolore == 0 -> ValoreCondizione1 = 0; ValoreCondizione1 = 1), % Angina tipica
        (AnginaEsercizio == 1 -> ValoreCondizione2 = 0; ValoreCondizione2 = 1), % Angina pectoris causata dall'esercizio
        ((Pendenza == 0; Pendenza == 1) -> ValoreCondizione3 = 0; ValoreCondizione3 = 1), % Pendenza pari a 0 o 1
        (NumeroVasi =< 3 -> ValoreCondizione4 = 0; ValoreCondizione4 = 1), % Numero di grandi vasi minore o uguale a 3
        (RisultatoThallium == 3 -> ValoreCondizione5 = 0; ValoreCondizione5 = 1), % Risultato del test al thallium uguale a 3
        ((Età > 50; Età < 30) -> ValoreCondizione6 = 0; ValoreCondizione6 = 1), % Età maggiore di 50 o minore di 30
        (FrequenzaCardiaca < 130 -> ValoreCondizione7 = 0; ValoreCondizione7 = 1), % Frequenza cardiaca massima minore di 130
        (PiccoPrecedente > 2.0 -> ValoreCondizione8 = 0; ValoreCondizione8 = 1) % Picco precedente alto
    ),

    % Somma dei valori delle condizioni
    SommaCondizioni is ValoreCondizione1 + ValoreCondizione2 + ValoreCondizione3 + ValoreCondizione4 + ValoreCondizione5 + Val
oreCondizione6 + ValoreCondizione7 + ValoreCondizione8,

    % Se la somma è maggiore di 4, allora può avere un attacco cardiaco
    (SommaCondizioni > 4 -> PuoAvereAttacco = si; PuoAvereAttacco = no).
```

Questa regola Prolog è stata creata per determinare se una persona può avere un attacco cardiaco utilizzando una somma di valori. Ogni condizione viene valutata separatamente e se una condizione è vera, il suo valore viene impostato a 0, altrimenti a 1. Successivamente, i valori di queste condizioni vengono sommati. Se la somma delle condizioni è maggiore di 4, la variabile PuoAvereAttacco viene impostata su "si", indicando che la persona potrebbe avere un attacco cardiaco. In caso contrario, se la somma è 4 o inferiore, la variabile PuoAvereAttacco viene impostata su "no", indicando che la persona non potrebbe avere un attacco cardiaco.

In sintesi, questa regola valuta una serie di condizioni specifiche basate su attributi dati in input e calcola una somma di queste condizioni. Se la somma supera una soglia specifica (4 in questo caso), si conclude che la persona potrebbe avere un attacco cardiaco. La soglia è stata decisa dopo determinati test di prova tenendo conto il dataset di partenza.

```
?- puo_avere_attacco_cardiaco(33,1,2,2,2,3,130,2,PuoAvereAttacco ).
PuoAvereAttacco = no.

?- puo_avere_attacco_cardiaco_prob(33,1,2,2,2,3,130,2,PuoAvereAttacco ).
PuoAvereAttacco = si.
```

esempio di query della regola 1 e 2

```
% Regola 3: per calcolare l'età media delle persone con attacco cardiaco
eta_media_persone_attacco_cardiaco(MediaEtà) :-
    findall(Age, (age(Age), Age > 0), ListeEtà), % Estrai le età delle persone con attacco cardiaco
    length(ListeEtà, NumeroPersone), % Conta il numero di persone con attacco cardiaco
    sum_list(ListeEtà, SommaEtà), % Somma le età
    MediaEtà is SommaEtà / NumeroPersone. % Calcola l'età media
```

Questa regola Prolog è stata creata per calcolare l'età media delle persone che hanno avuto un attacco cardiaco all'interno del dataset. La regola fa uso della funzione findall per estrarre tutte le età delle persone con attacco cardiaco e le memorizza in una lista chiamata ListeEtà. Successivamente, la

lunghezza di questa lista viene calcolata con length per determinare il numero di persone con attacco cardiaco.

La somma di tutte le età presenti nella lista viene calcolata utilizzando sum_list, e infine, l'età media viene ottenuta dividendo la somma per il numero di persone con attacco cardiaco. Il risultato finale viene memorizzato nella variabile MediaEtà, che rappresenta l'età media delle persone con attacco cardiaco.

```
% Regola 4: per determinare il tipo di dolore toracico più comune
tipo_dolore_piu_comune(TipoDolorePiuComune) :-
    findall(Tipo, (cp(Tipo), Tipo >= 0, Tipo =< 3), TipiDolore), % Estrai i tipi di dolore toracico
    list_to_set(TipiDolore, TipiUnici), % Rimuovi duplicati
    conta_tipi_dolore(TipiUnici, TipiDolore, TipoDoloreConteggi), % Conta quanti individui hanno ciascun tipo di dolore
    trova_tipo_piu_comune(TipoDoloreConteggi, TipoDolorePiuComune). % Trova il tipo di dolore più comune
```

Questa regola Prolog è stata creata per determinare il tipo di dolore toracico più comune all'interno di un dataset. La regola sfrutta la funzione findall per estrarre tutti i tipi di dolore toracico presenti nei dati. Successivamente, viene utilizzata la funzione list_to_set per rimuovere eventuali duplicati dalla lista dei tipi di dolore.

Una volta ottenuta la lista dei tipi di dolore unici, la regola chiama la funzione conta_tipi_dolore per contare quante volte ciascun tipo di dolore compare nel dataset. Infine, utilizzando la funzione trova_tipo_piu_comune, viene identificato il tipo di dolore toracico più comune tra quelli presenti.

Apprendimento supervisionato

Sommario

Questo caso di studio si occupa di diagnosticare se un paziente possa avere un problema cardiaco, in particolare utilizzando la variabile "*output*" come target da predire basandosi sulle altre features relative al paziente.

È stato necessario decidere quali modelli utilizzare, considerando la quantità limitata di dati nel dataset, non sono stati considerati modelli con complessità elevata come la regressione lineare e le reti neurali.

Sono stati infine scelti e valutati i seguenti modelli: XGBoost, utilizzando la libreria xgboost [1]; K-nearest-neighbors (KNN), utilizzando la libreria Scikit learn [2]; Random Forest, utilizzando la libreria Scikit learn [3], Ada Boost CLassifier, utilizzando la libreria Scikit learn [4]; Decision Tree, utilizzando la libreria Scikit learn [5].

Decisioni di Progetto

Per prima cosa i vari modelli sono stati testati senza usare particolari parametri, per capire quali fossero i risultati iniziali che sono stati i seguenti:

KNN Classification :

	precision	recall	f1-score	support
0	0.60	0.64	0.62	14
1	0.69	0.65	0.67	17
accuracy			0.65	31
macro avg	0.64	0.64	0.64	31
weighted avg	0.65	0.65	0.65	31

F2 Score: 0.6547619047619049

Decision Tree Classification :

	precision	recall	f1-score	support
0	0.65	0.79	0.71	14
1	0.79	0.65	0.71	17
accuracy			0.71	31
macro avg	0.72	0.72	0.71	31
weighted avg	0.72	0.71	0.71	31

F2 Score: 0.6707317073170732

Random Forest Classification :

	precision	recall	f1-score	support
0	0.75	0.86	0.80	14
1	0.87	0.76	0.81	17
accuracy			0.81	31
macro avg	0.81	0.81	0.81	31
weighted avg	0.81	0.81	0.81	31

F2 Score: 0.783132530120482

Ada Boost Classifier Classification :

	precision	recall	f1-score	support
0	0.68	0.93	0.79	14
1	0.92	0.65	0.76	17
accuracy			0.77	31
macro avg	0.80	0.79	0.77	31
weighted avg	0.81	0.77	0.77	31

F2 Score: 0.6875000000000001

XGBoost Classifier Classification :

	precision	recall	f1-score	support
0	0.68	0.93	0.79	14
1	0.92	0.65	0.76	17
accuracy			0.77	31
macro avg	0.80	0.79	0.77	31
weighted avg	0.81	0.77	0.77	31

F2 Score: 0.6875000000000001

È stato anche creato un classification_report e ottenuto ulteriori valutazioni delle prestazioni del modello, come precision e recall.

Precision (Precisione): Questa metrica aiuta a minimizzare i risultati dei falsi positivi. I falsi positivi si verificano quando il modello predice un valore come classe positiva, anche se il valore in realtà non appartiene a una classe positiva. Una precision elevata significa che quando il modello classifica un caso come positivo, è molto probabile che sia effettivamente positivo, riducendo al minimo i falsi positivi.

Recall: Questa metrica aiuta a minimizzare i risultati dei falsi negativi. I falsi negativi si verificano quando il modello predice la classe come negativa, anche se il caso è in realtà positivo. Un recall elevato significa che il modello è bravo a individuare la maggior parte dei casi positivi senza lasciare indietro troppi falsi negativi.

Queste misure sono essenziali nella valutazione delle prestazioni del modello, soprattutto in problemi di classificazione in cui una classe è significativamente più rara dell'altra. Ad esempio, in un problema medico come la diagnosi di attacchi cardiaci, è fondamentale ridurre al minimo sia i falsi positivi (che potrebbero causare ansia inutile ai pazienti) sia i falsi negativi (che potrebbero essere fatali se non diagnosticati).

Ottimizzare queste metriche può essere una sfida poiché spesso c'è un compromesso tra precisione e recall. L'ottimizzazione dipenderà dalle esigenze specifiche del tuo problema e da quanto peso si desidera dare ai falsi positivi rispetto ai falsi negativi.

Da questa valutazione, sembra che il modello "Random Forest" abbia le prestazioni migliori in termini di F2 Score e Precision, seguito da "Ada Boost Classifier" e "XGBoost Classifier". "Decision Tree" e "K-nearest-neighbors (KNN)" hanno prestazioni leggermente inferiori.

Dopo questa prima analisi si è proseguito andando cercare quelli che sono i parametri ottimali per i modelli KNN, Decision Tree e Random Forest per avere maggiori prestazioni, vediamo nel dettaglio.

In particolare sono stati scelti questi 3 modelli sopracitati in quanto hanno margine di miglioramento migliore.

Decision Tree

Ci sono diversi iperparametri nell'albero decisionale che possono essere configurati per ottimizzare il modello. Ecco alcuni di essi:

- max_depth: La massima profondità dell'albero. Questo parametro determina quante volte l'albero verrà suddiviso in livelli (divisioni). Una profondità elevata può portare a un sovradattamento (overfitting), mentre una profondità bassa può portare a un sottoadattamento (underfitting).
- min_samples_leaf: Il numero minimo di campioni che dovrebbero essere presenti nel nodo foglia (dopo la separazione). Se dopo la divisione il numero di campioni in uno dei nuovi rami diventa inferiore a questo valore, la divisione non viene effettuata.
- splitter: Una strategia per la selezione di un attributo per la divisione. Può essere "best" (il miglior attributo) o "random" (attributo casuale).

Ora consideriamo ciascuno di questi iperparametri separatamente e successivamente combiniamo i valori migliori per trovare la migliore combinazione con l'accuratezza più alta.

Per il max_depth e il min_samples_leaf risulta:

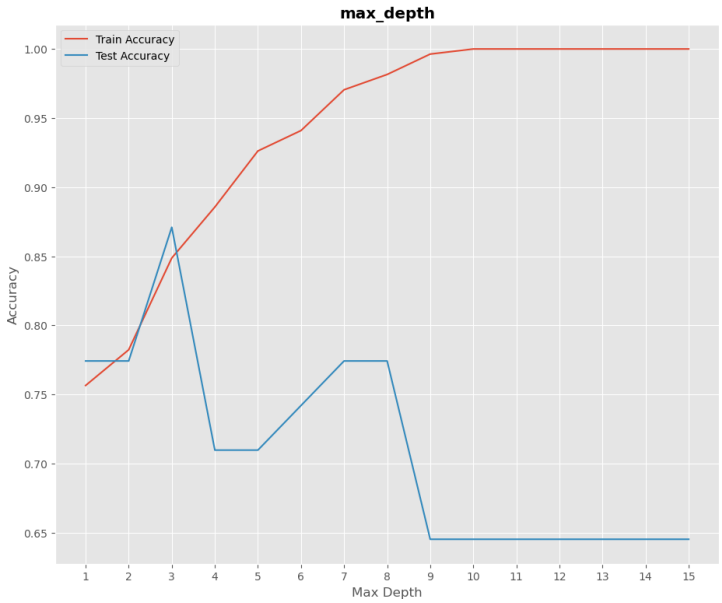


Figura 1

Come si vede in figura 1, il miglior parametro in questo caso è il valore della profondità dell'albero = 3. Con questo valore, l'accuratezza sui dati di test supera addirittura l'accuratezza sui dati di addestramento, il che indica che il modello con questo parametro ha un'ottima capacità di generalizzazione.

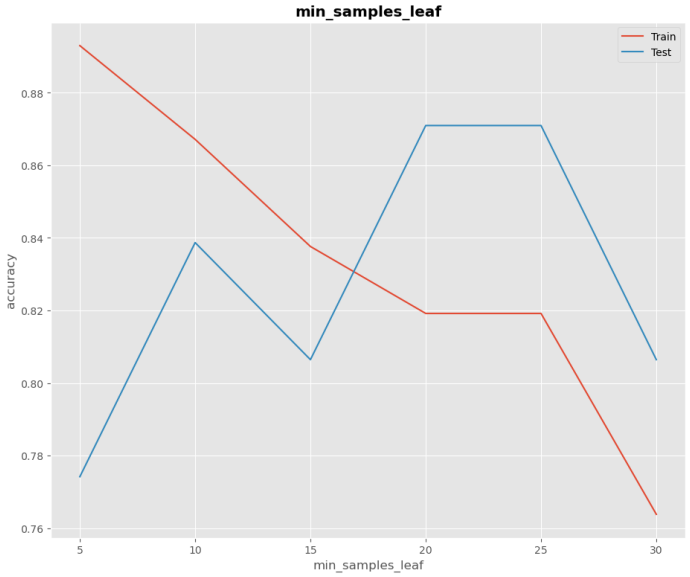


Figura 2

Come si può vedere da questo grafico (figura 2), con l'aumento del numero di elementi nel nodo foglia, l'accuratezza del modello diminuisce. Tuttavia, al contrario, con l'aumento del numero di valori nel nodo foglia sui dati di test, i valori migliori si verificano a 20 e 25. Ciò significa che è a tali valori che il modello mostra la migliore accuratezza e ha una buona capacità di apprendimento.

Con un grande numero di valori nel nodo foglia, il modello non è in grado di riaddestrarsi e memorizzare il set di dati di addestramento. Ad esempio, con un valore di 5, il modello si riaddestra perché ci sono pochi valori nel nodo foglia e il modello memorizza semplicemente i valori, ma non li generalizza in alcun modo. Pertanto, la scelta dei giusti valori per il numero di valori nel nodo foglia è cruciale per garantire che il modello sia in grado di generalizzare correttamente i dati di addestramento e ottenere buone prestazioni sui dati di test.

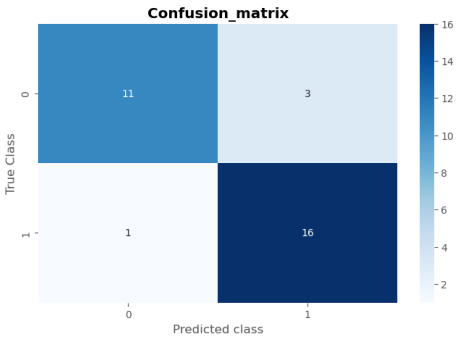
I risultati ottenuti scegliendo:

```
cross val accuracy: [0.65454545 0.74074074 0.74074074 0.72222222 0.85185185]

mean accuracy =  0.7420202020202021
std accuracy =  0.06341932954064633
0.8709677419354839
              precision    recall  f1-score   support

           0       0.92      0.79      0.85        14
           1       0.84      0.94      0.89        17

    accuracy                           0.87        31
   macro avg       0.88      0.86      0.87        31
weighted avg       0.88      0.87      0.87        31
```

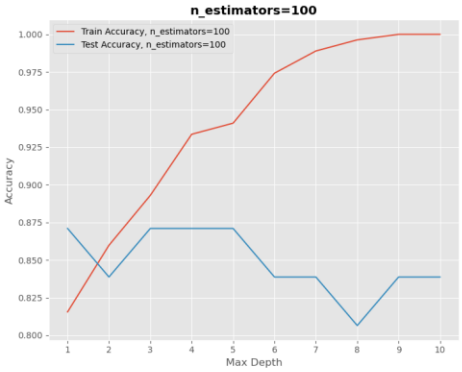


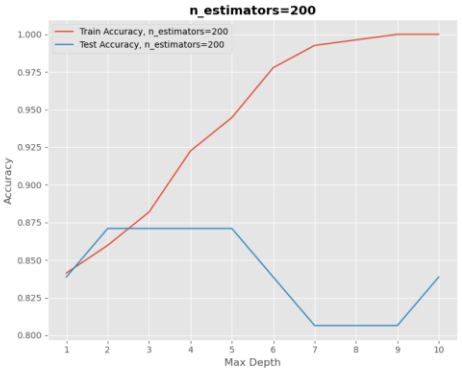
Dopo l'ottimizzazione dei modelli, abbiamo esaminato l'accuratezza separatamente, che è aumentata dal 71% all'87%, il che rappresenta un risultato molto positivo.

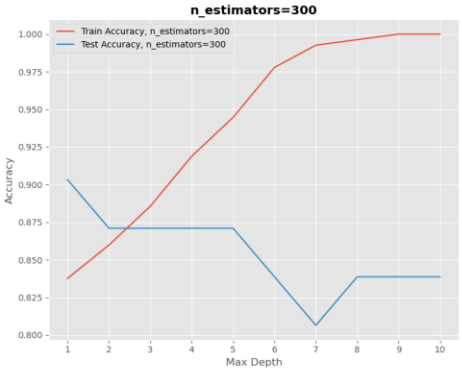
Random Forest

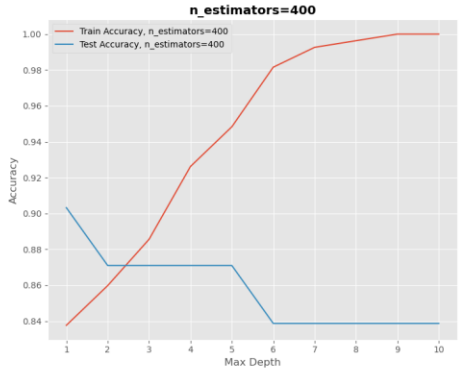
In un random forest, ci sono diversi iperparametri che possono essere personalizzati per ottenere una migliore performance del modello. Ecco alcuni dei principali iperparametri di un random forest:

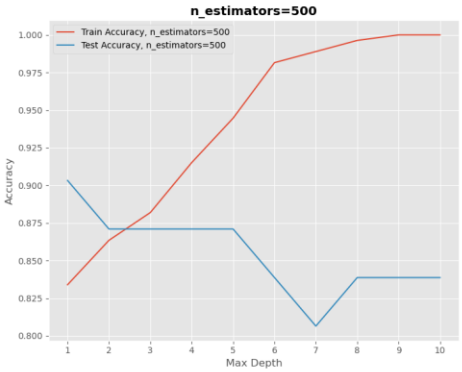
- n_estimators: Il numero di alberi in un random forest. Questo parametro specifica quanti alberi verranno creati nell'insieme. Più alberi possono migliorare le prestazioni del modello, ma possono anche portare a un aumento del tempo di addestramento. Il valore predefinito è 100.
- max_depth: La massima profondità di ciascun albero. Limitare la profondità degli alberi può aiutare a combattere l'overfitting. Se il valore è None, gli alberi cresceranno fino a quando tutte le foglie saranno pure o fino a quando verrà raggiunto il numero minimo di campioni per la separazione.
- min_samples_leaf: Il numero minimo di campioni richiesti per formare una foglia dell'albero. Se ci sono meno campioni nel nodo rispetto al valore specificato, si interrompe la creazione della foglia. Aumentare questo valore può aiutare a eliminare il rumore e aumentare la capacità di generalizzazione del modello. Il valore predefinito è 1.
- max_features: Il numero di features considerate in ciascuna divisione dell'albero. Specificare un valore specifico può limitare il numero di features che possono essere utilizzate nella separazione e può contribuire a combattere l'overfitting. Per default, il valore è "auto", il che significa sqrt(n_features), dove n_features è il numero di features.

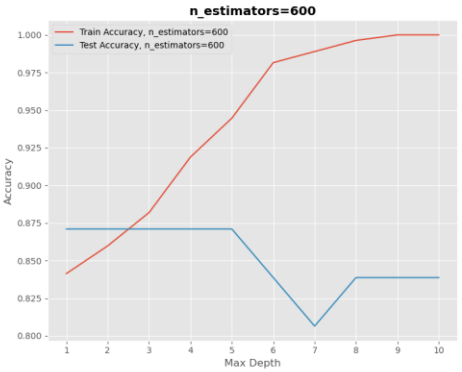

n_estimators=100
Train Accuracy, n_estimators=100
Test Accuracy, n_estimators=100
Accuracy
Max Depth

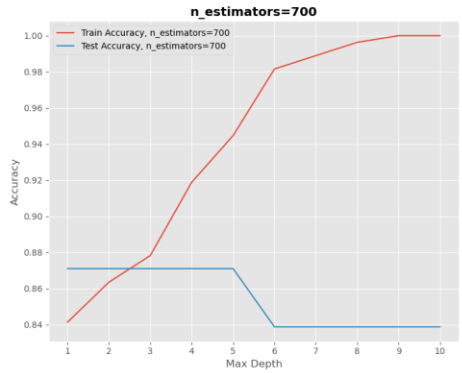

n_estimators=200
Train Accuracy, n_estimators=200
Test Accuracy, n_estimators=200
Accuracy
Max Depth

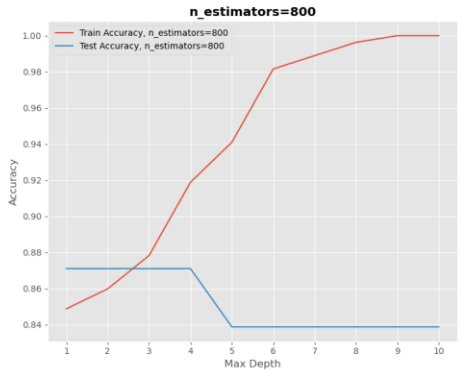

n_estimators=300
Train Accuracy, n_estimators=300
Test Accuracy, n_estimators=300
Accuracy
Max Depth

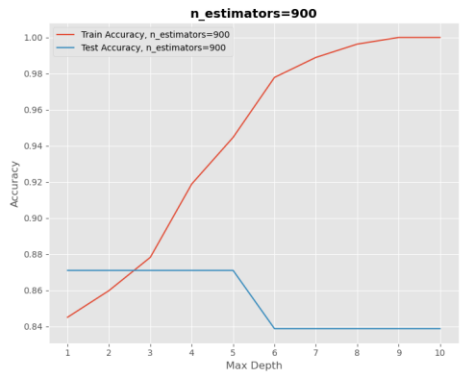

n_estimators=400
Train Accuracy, n_estimators=400
Test Accuracy, n_estimators=400
Accuracy
Max Depth

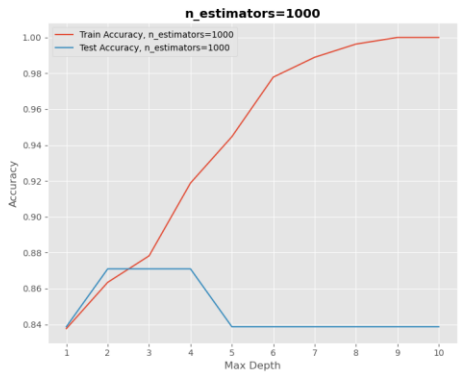

n_estimators=500
Train Accuracy, n_estimators=500
Test Accuracy, n_estimators=500
Accuracy
Max Depth


n_estimators=600
Train Accuracy, n_estimators=600
Test Accuracy, n_estimators=600
Accuracy
Max Depth


n_estimators=700
Train Accuracy, n_estimators=700
Test Accuracy, n_estimators=700
Accuracy
Max Depth


n_estimators=800
Train Accuracy, n_estimators=800
Test Accuracy, n_estimators=800
Accuracy
Max Depth


n_estimators=900
Train Accuracy, n_estimators=900
Test Accuracy, n_estimators=900
Accuracy
Max Depth


n_estimators=1000
Train Accuracy, n_estimators=1000
Test Accuracy, n_estimators=1000
Accuracy
Max Depth

Il modello mostra un'alta accuratezza su parametri come il numero di alberi 300, 400, 500 e la profondità dell'albero 1.

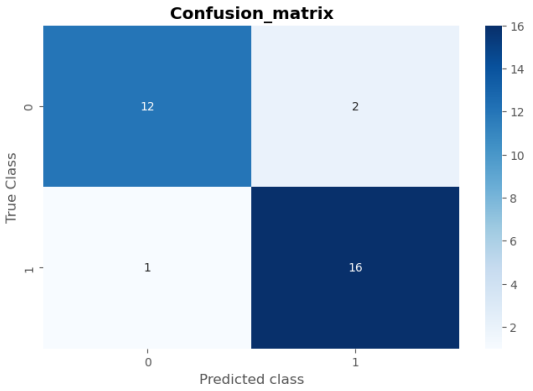
Addestrando il modello e testandolo su un campione di test con questi parametri, si ottiene:

```
[0.8        0.87037037 0.85185185 0.7962963  0.85185185]

mean accuracy 0.834074074074074
std accuracy 0.030125480603944814
0.9032258064516129
              precision    recall  f1-score   support

           0       0.92      0.86      0.89        14
           1       0.89      0.94      0.91        17

    accuracy                           0.90        31
   macro avg       0.91      0.90      0.90        31
weighted avg       0.90      0.90      0.90        31
```



Alla fine, abbiamo ottenuto un modello con un'accuratezza del 90%. Abbiamo migliorato l'accuratezza dal 81% al 90%. Ora il modello Random Forest è il migliore, poiché è migliore dell'albero decisionale (e l'albero decisionale ha un'accuratezza dell'87% con iperparametri configurati).

Altri parametri, come la precisione e il richiamo, sono migliorati anche nel Random Forest.

La precisione è diventata meno errata e meno incline a fornire stime di falsi positivi (il che svolge un ruolo molto importante per il set di dati medici). Il valore del richiamo è diventato meno incline a fornire risultati falsi negativi (ovvero, se una persona è malata, il modello dice che non è malata - ci sono meno di questi risultati), il che è importante anche per il set di dati medici poiché tutto in medicina riguarda la vita delle persone. Pertanto, è importante avere un modello di buona qualità e con un'alta accuratezza.

La conclusione che si può trarre da questa analisi del modello è che l'accuratezza delle previsioni è aumentata quasi del 10%, il che è un buon risultato. Abbiamo regolato i parametri del modello e, in questa fase, il modello Random Forest è il migliore.

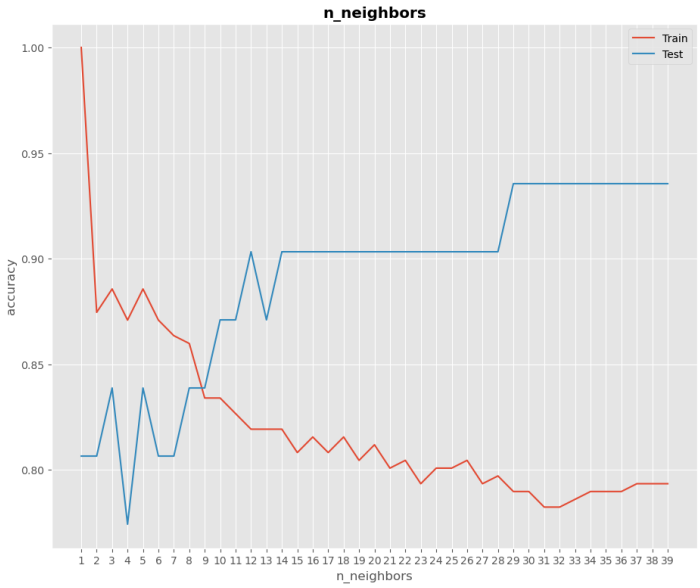
KNN

Per ottimizzare il modello K-Nearest Neighbors (KNN), che utilizza la distanza tra i punti dati e le classi dei vicini più vicini, è cruciale standardizzare i dati affinché tutte le variabili abbiano lo stesso peso.

Standardizzando i dati è possibile notare come sia cambiata l'accuratezza del modello nel campione di test senza la selezione degli iperparametri. Pertanto, concludiamo che in questa fase il modello è migliorato dal 64% all'83%, il che è un indicatore molto buono. I passi successivi consistono nella configurazione dei parametri del modello, come ad esempio il numero di vicini e la scelta di una formula per il calcolo della distanza.

Per l'algoritmo k-Nearest Neighbors (kNN), ci sono i seguenti iperparametri che possono essere configurati:

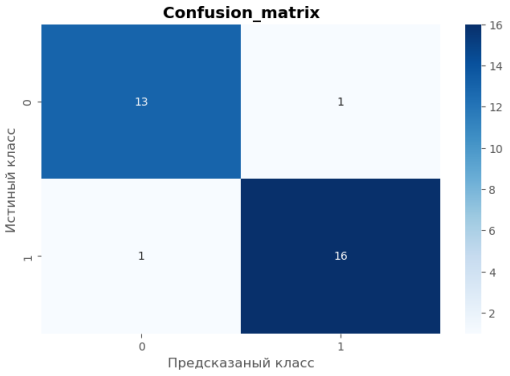
- n_neighbors: Il numero di vicini più vicini che saranno presi in considerazione durante la classificazione o la regressione. Questo è uno dei più importanti iperparametri del kNN.



Come si può vedere dal grafico, con l'aumento del numero di vicini, l'accuratezza del modello aumenta e raggiunge il massimo quando il valore dei vicini è 29. Il numero 29 indica che il modello ha una buona capacità di generalizzazione. Ad esempio, con un valore di 1, il modello prenderebbe semplicemente il primo vicino e lo classificherebbe allo stesso modo. Con un valore elevato di vicini, i risultati vengono confrontati e viene selezionato il valore più popolare tra i vicini, quindi addestrare il modello con 29 vicini significa considerare una varietà di punti di dati vicini per prendere una decisione di classificazione. Questo approccio di "voto della maggioranza" può portare a previsioni più accurate in molti casi.

Ed ecco infine i risultati ottenuti:

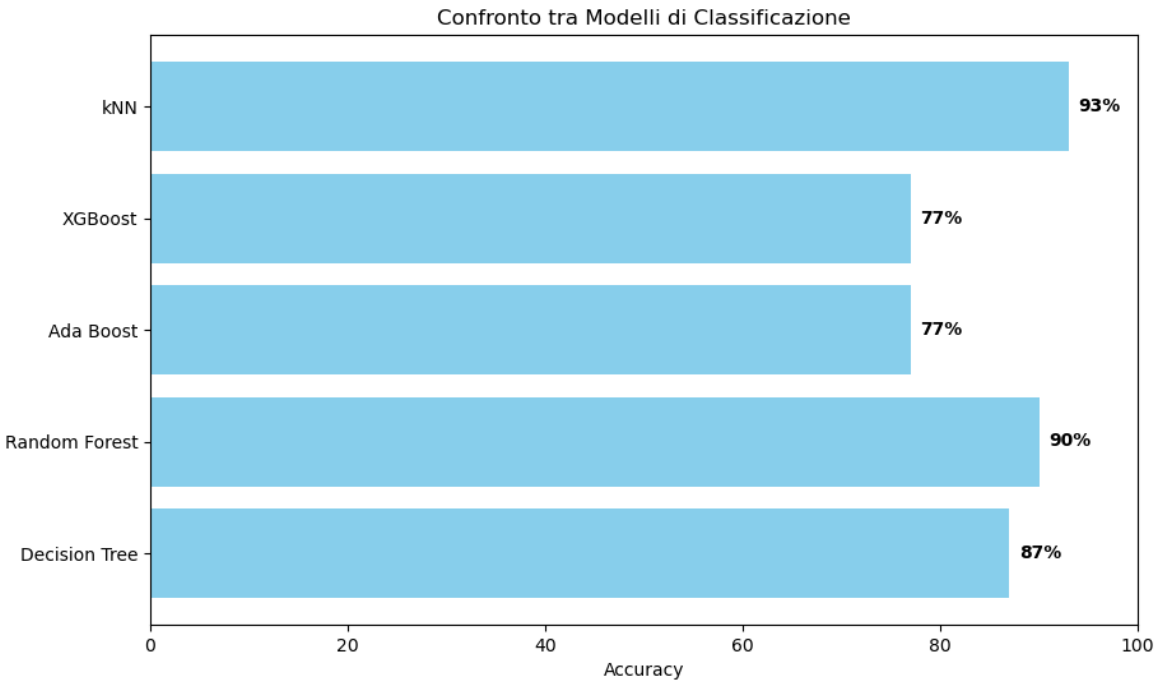
	precision	recall	f1-score	support
0	0.93	0.93	0.93	14
1	0.94	0.94	0.94	17
accuracy			0.94	31
macro avg	0.93	0.93	0.93	31
weighted avg	0.94	0.94	0.94	31



Il risultato è che il modello k-Nearest Neighbors (kNN) è diventato il miglior modello tra tutti quelli che abbiamo addestrato. La sua accuratezza raggiunge il 93%, e le metriche come la precisione e il richiamo sono migliorate. In altre parole, il modello commette solo il 6% - 7% di falsi positivi (cioè quando una persona non è realmente malata, il modello dice che è malata) e anche il 6% - 7% di falsi negativi (cioè quando una persona è effettivamente malata, il modello dice che è sana). Questi sono indicatori importanti nel contesto di un dataset medico, e possiamo anche affermare che l'accuratezza raggiunta del 94% è un buon indicatore, considerando che l'accuratezza iniziale di questo modello era del 64%.

Conclusioni

In generale, il modello k-Nearest Neighbors (kNN) sembra essere il migliore tra quelli che hai valutato, con un'accuratezza del 93% e una buona gestione di falsi positivi e falsi negativi, che sono particolarmente importanti in un contesto medico.



Apprendimento non supervisionato

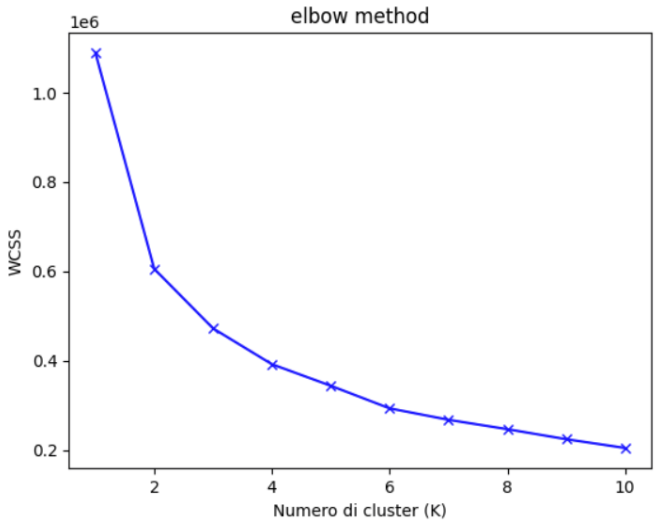
Sommario

Nel contesto del nostro caso di studio, abbiamo adottato un approccio di clustering rigido, noto come k-means tramite la libreria scikit-learn.

K-Means è uno degli algoritmi di clustering più comuni e ben noti nell'apprendimento non supervisionato. L'obiettivo dell'algoritmo K-Means è suddividere un insieme di dati in un numero predefinito di cluster (K) in modo che ciascun punto dati appartenga al cluster il cui centroide (punto medio) è più vicino. L'algoritmo procede iterativamente assegnando punti dati ai cluster più vicini e quindi aggiornando i centroidi dei cluster. L'obiettivo è minimizzare la somma dei quadrati delle distanze tra i punti dati e i centroidi dei rispettivi cluster (chiamata "inerzia").

Decisioni di Progetto

Per determinare il numero ottimale di cluster è stato utilizzato il metodo del gomito (elbow method), dove sono rappresentati i valori delle somme dei quadrati intracluster (WCSS) e il valore k che rappresenta il numero dei cluster.



Per determinare il numero ottimale di cluster dobbiamo selezionare il valore di k al “gomito”, cioè il punto dopo il quale la distorsione/inerzia inizia a diminuire in modo lineare. Pertanto, per i dati forniti, concludiamo che il numero ottimale di cluster per i dati è 4.

Il valore di k da me scelto è 2.

Valutazione

L'inerzia misura la coesione dei cluster, mentre il punteggio silhouette misura la separazione dei cluster e l'assegnazione dei punti dati. Entrambi sono indicatori della qualità del clustering, con un'inerzia più bassa e un punteggio silhouette più alto che suggeriscono un buon clustering.

L’indice di silhouette varia da −1 a +1, dove un valore elevato indica che l'oggetto è ben abbinato al proprio cluster e scarsamente abbinato ai cluster vicini, nel nostro caso ha valore di circa 0.39. Possiamo dedurre che il nostro clustering ha avuto risultati mediocri, in quanto il valore non è elevato.

Dimostrando così che il clustering non è stato utile nella nostro caso di studio.

Riferimenti Bibliografici

[1]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

[2] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[3] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[4] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

[5] https://scikit-learn.org/stable/modules/tree.html

[6] https://www.kaggle.com/datasets/pritsheta/heart-attack