UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO
DIPARTIMENTO DI INFORMATICA

Strumento diagnostico per attacchi di cuore

Ingegneria della conoscenza

AA 2024-25

Gruppo di lavoro:

Giuseppe Damato, 778583, g.damato40@studenti.uniba.it

URL Repository : https://github.com/giusedam/icon

Sommario

Introduzione

I problemi cardiaci rappresentano una sfida significativa sia a livello globale che in ambito medico. Le malattie cardiovascolari sono la principale causa di morte nel mondo, responsabili del 32% dei decessi nel 2020, secondo l'Organizzazione Mondiale della Sanità (OMS). Queste patologie possono colpire individui di tutte le età, sebbene siano più comuni tra gli anziani. Diversi fattori di rischio contribuiscono allo sviluppo delle malattie cardiache, tra cui l'età avanzata, il fumo, l'obesità, l'ipertensione, il diabete e la sedentarietà. La gestione di questi fattori è essenziale per la prevenzione. Negli ultimi anni, la medicina ha fatto notevoli progressi nella diagnosi delle patologie cardiache, grazie a tecniche avanzate di imaging e a test del sangue per l'identificazione di biomarcatori specifici. Tuttavia, la prevenzione resta fondamentale e si attua attraverso campagne di sensibilizzazione, programmi di educazione alimentare e la promozione di uno stile di vita sano.
Infine, la tecnologia sta rivoluzionando la gestione delle malattie cardiache, grazie allo sviluppo della telemedicina e all'impiego dell'intelligenza artificiale per la diagnosi e il monitoraggio delle condizioni cardiache.

Elenco argomenti di interesse

- **Apprendimento supervisionato per la diagnosi dell'attacco cardiaco**

Ottimizzazione degli iper parametri per migliorare l'accuratezza del modello di diagnosi

- **Knowledge Base**

È stata sviluppata una base di conoscenza per analizzare le informazioni contenute nel dataset e determinare, in base ai parametri di un paziente, la probabilità che possa subire un attacco cardiaco, attraverso l'utilizzo di query specifiche.

Analisi dei dati

Sommario

Prima di avviare il lavoro sul sistema di diagnosi di un attacco cardiaco, è stato fondamentale eseguire alcune analisi preliminari sul dataset denominato "heart", situato nella directory "dataset".

È importante conoscere alcuni dettagli sul dataset prima di procedere con l'analisi.
Esso è composto dalle seguenti variabili:

- **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, misurato mediante un sensore BMI.
- **fbs**: Zucchero nel sangue a digiuno (> 120 mg/dl) (1 = vero, 0 = falso).
- **restecg**: Risultati dell'elettrocardiogramma a riposo
- (0 = normale, 1 = anomalie dell'onda ST-T, 2 = ipertrofia del ventricolo sinistro).
- **thalachh**: Frequenza cardiaca massima raggiunta.
- **exng**: Angina pectoris indotta dall'esercizio fisico (1 = sì, 0 = no).
- **oldpeak**: Depressione del segmento ST rispetto alla linea di base.
- **slp**: Pendenza del segmento ST durante l'esercizio.
- **caa**: Numero di grandi vasi visibili in fluoroscopia.
- **thall**: Risultato del test al tallio (valori possibili: 0, 1, 2, 3).
- **output**: Variabile target, che indica la presenza o assenza di malattia cardiaca.

Strumenti utilizzati

Tramite l'utilizzo dell'EDA (*Exploratory Data Analysis*, ovvero Analisi Esplorativa dei Dati) in Python, è stata rilevata la presenza di righe duplicate, che sono state successivamente eliminate.

Di seguito, una breve analisi delle variabili:

- Sex (Sesso):
  La variabile *sesso* sembra avere una forte correlazione con la presenza di attacchi cardiaci. Le donne presentano una minore probabilità di esserne colpite, mentre negli uomini il rischio è significativamente maggiore.

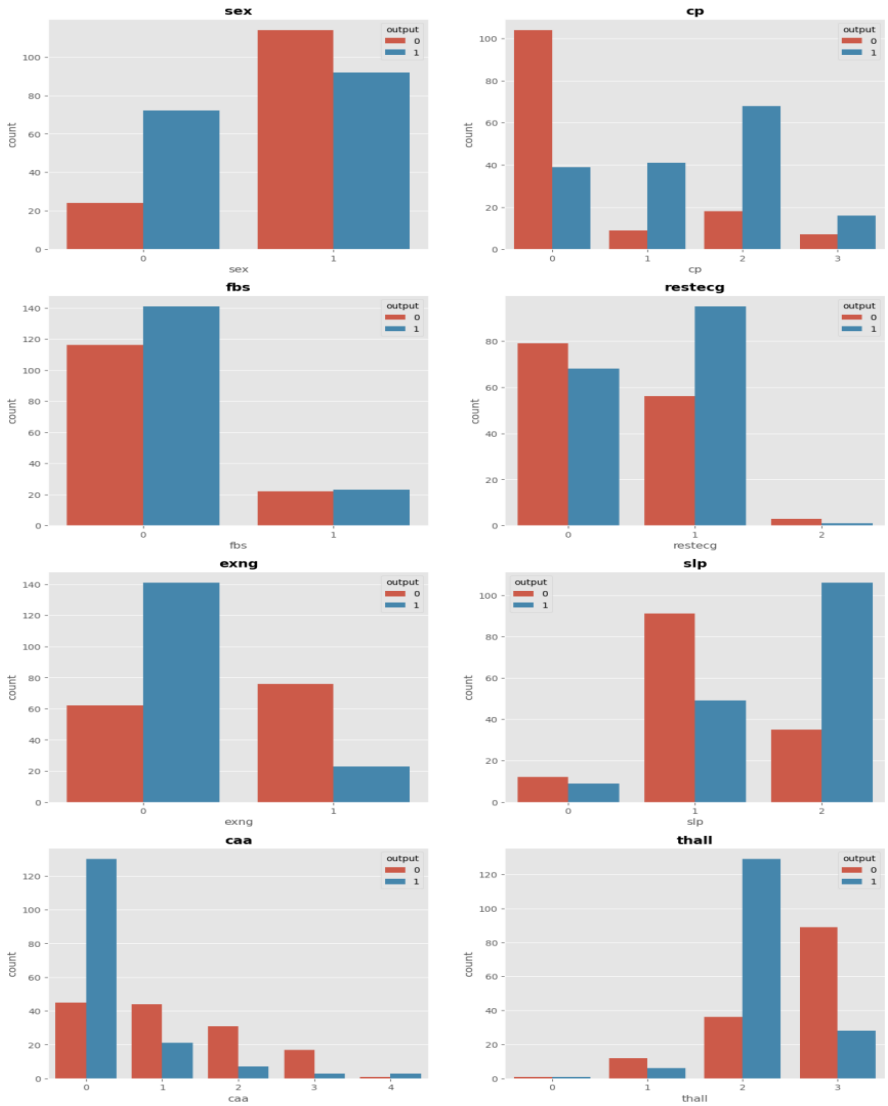
- CP (Tipo di Dolore Toracico):
  La maggior parte delle persone con *angina tipica* non ha subito attacchi cardiaci. Al contrario, nelle altre categorie di dolore toracico, il rischio di attacco cardiaco risulta più elevato.

- Exng (Angina Pectoris Causata dall'Esercizio):
  Se l'angina pectoris è indotta dall'esercizio fisico, la maggior parte delle persone nel dataset non ha subito attacchi cardiaci. Viceversa, tra coloro la cui angina non è correlata all'attività fisica, il numero di attacchi cardiaci è significativamente più alto.

- Slp (Pendenza del Segmento ST):
  Con un valore di *slp* pari a 0, la distribuzione tra persone con e senza attacco cardiaco appare bilanciata. Un valore di *slp* pari a 1 è associato a una maggiore presenza di persone senza attacco cardiaco, mentre un valore di 2 è più comune tra coloro che hanno subito un attacco.

- Caa (Numero di Grandi Vasi):
  L'aumento del numero di grandi vasi (ad eccezione del 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 sviluppare problemi cardiaci rispetto a quelle senza grandi vasi o con un valore pari a 4.

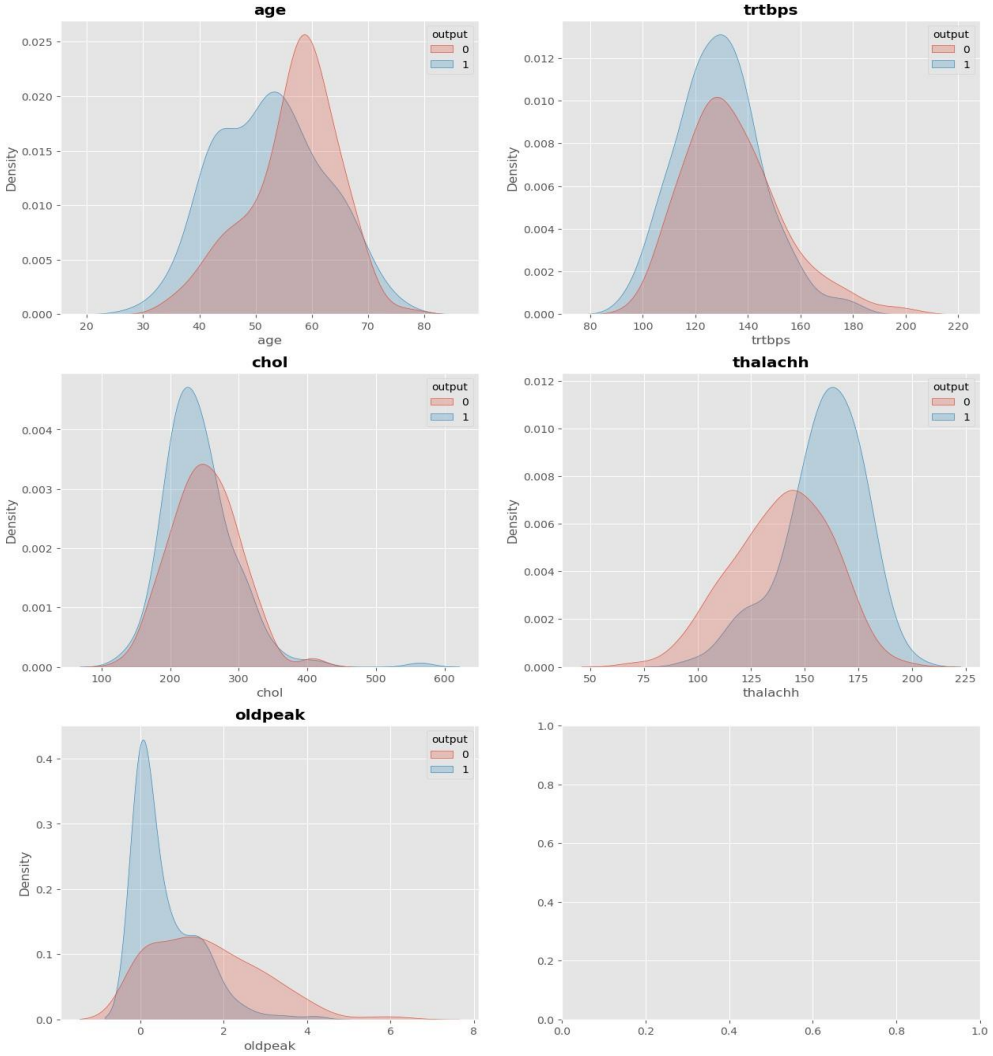
- Thall (Risultato del Test al Tallio):
  La maggior parte dei valori di questa variabile rientra nei gruppi 2 e 3, che permettono una chiara suddivisione della variabile target. Il gruppo con risultato 2 presenta una maggiore incidenza di attacchi cardiaci, mentre il gruppo con risultato 3 include prevalentemente individui sani.



- Age (Età):
  Gli individui tra i 30 e i 60 anni risultano più inclini a soffrire di attacchi cardiaci. D'altra parte, i dati suggeriscono che, oltre i 50 anni, il rischio tenda a diminuire.

- Thalachh (Frequenza Cardiaca Massima):
  Questa variabile mostra una chiara separazione rispetto alla variabile target. Una frequenza cardiaca inferiore a 130 è associata a un basso rischio di attacco cardiaco, mentre valori superiori a 130 corrispondono a un aumento significativo del rischio.

- Oldpeak (Depressione del Segmento ST):
  Anche questa variabile risulta utile nella distinzione tra individui con e senza attacchi cardiaci. Valori bassi di *oldpeak* sono più comuni tra coloro che hanno subito un attacco cardiaco, mentre all'aumentare del valore di *oldpeak*, la probabilità di attacco cardiaco diminuisce e aumenta quella di non averne subito uno.



In sintesi, le caratteristiche più rilevanti risultano essere: la frequenza cardiaca massima (*thalachh*), il picco precedente (*oldpeak*), il numero di grandi vasi (*caa*), il risultato del test Thallium (*thall*), il sesso (*sex*), il tipo di dolore toracico (*cp*), l'angina pectoris causata dall'esercizio (*exng*) e la pendenza (*slp*).
Questi saranno i parametri utilizzati nel mio sistema di diagnosi.

Knowledge Base

Sommario e Strumenti utilizzati

Una base di conoscenza in logica del primo ordine è costituita da un insieme di proposizioni, note come assiomi, che vengono accettate come vere senza necessità di dimostrazione. Questo tipo di struttura viene impiegato per rappresentare informazioni relative a un determinato dominio all'interno di un sistema o di un programma informatico.
Il processo di costruzione di una base di conoscenza si articola in diverse fasi:

- **Definizione del Dominio:**
  Il primo passo consiste nello stabilire l'ambito di riferimento che si intende modellare. Questo può riguardare il mondo reale, un universo astratto (ad esempio, numeri o insiemi) oppure un contesto immaginario.

- **Identificazione delle Proposizioni Atomiche:**
  Si procede quindi all'individuazione delle proposizioni atomiche, che rappresentano le unità fondamentali con cui sarà costruita la base di conoscenza.

- **Assiomatizzazione del Dominio:**
  In questa fase vengono definite le proposizioni che si considerano vere per descrivere il dominio scelto. Questi assiomi fungono da fondamento della base di conoscenza.

- **Interrogazione del Sistema**:
  Una volta strutturata la base di conoscenza, è possibile formulare domande o query per ottenere inferenze logiche. Il sistema analizzerà se determinate proposizioni derivano logicamente dagli assiomi definiti, valutandone la validità in tutti i modelli della base di conoscenza.

Va sottolineato che il sistema non possiede una comprensione diretta del significato dei simboli utilizzati nella base di conoscenza. Esso si limita ad applicare regole logiche sugli assiomi forniti per dedurre nuove informazioni. Spetta quindi al progettista interpretare i risultati ottenuti e verificarne la coerenza con la conoscenza del dominio.
Per implementare questa base di conoscenza in logica del primo ordine, è stato adottato il linguaggio di programmazione **Prolog**, supportato dalla libreria **pyswip**.

Decisioni di Progetto

Di seguito sono riportate alcune regole implementate in **Prolog**, attraverso le quali l'utente può interagire con il sistema formulando query. Queste regole consentono di effettuare deduzioni basate sulla base di conoscenza definita, permettendo di ottenere informazioni in modo logico e strutturato.

```
%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 in **Prolog** è stata progettata per valutare la probabilità che un individuo possa subire un attacco cardiaco, basandosi su diversi parametri forniti come input. La logica alla base della regola prevede una serie di condizioni che, se verificate, suggeriscono un potenziale rischio di attacco cardiaco. In caso contrario, se tali condizioni non sono soddisfatte, la variabile **PuoAvereAttacco** assume il valore "no", indicando che l'individuo non è a rischio.

I criteri presi in esame includono il **tipo di dolore toracico**, la **presenza di angina da sforzo**, la **pendenza dell'ECG**, il **numero di grandi vasi**, il **risultato del test al thallium**, l'**età**, la **frequenza cardiaca massima** e il **valore del picco precedente**. Se nessuna delle condizioni che indicano rischio è soddisfatta, il sistema conclude che l'individuo **non ha probabilità di subire un attacco cardiaco**. In caso contrario, viene segnalata una possibile predisposizione al problema.

Tutti i valori e le soglie utilizzate nella regola derivano direttamente dall'**analisi esplorativa dei dati** condotta sul 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 in **Prolog** è stata progettata per determinare la possibilità che una persona possa subire un attacco cardiaco, calcolando una somma di valori associati a diverse condizioni. Ogni singola condizione viene esaminata individualmente e, se la condizione risulta vera, il suo valore viene impostato a **0**, altrimenti a **1**. Dopo aver valutato tutte le condizioni, i valori ottenuti vengono sommati.

Se la somma totale delle condizioni supera il valore di **4**, la variabile **PuoAvereAttacco** viene settata su "si", indicando che l'individuo potrebbe essere a rischio di attacco cardiaco. In caso contrario, se la somma è pari o inferiore a **4**, la variabile viene impostata su "no", segnalando che la persona non è a rischio.

In sintesi, questa regola esamina un insieme di condizioni basate su determinati parametri di input, calcolando la loro somma. Se la somma supera una soglia predefinita (4 in questo caso), si considera che l'individuo potrebbe avere un attacco cardiaco. La soglia è stata determinata attraverso prove e test, tenendo conto dei dati del dataset utilizzato.

Implementazione delle Regole Prolog per la Diagnosi di Attacco Cardiaco

Di seguito vengono spiegate le regole implementate in Prolog per determinare se una persona potrebbe avere un attacco cardiaco, calcolare l'età media delle persone con attacco cardiaco, determinare il tipo di dolore toracico più comune e altre funzionalità.

Regola 1:

Determinare se una persona può avere un attacco cardiaco

Questa regola verifica una serie di condizioni per determinare se una persona potrebbe avere un attacco cardiaco. Le condizioni includono il tipo di dolore toracico, se l'angina è causata dall'esercizio, la pendenza dell'ECG, il numero di vasi, i risultati del test al thallium, l'età, la frequenza cardiaca massima e il picco precedente.

Se nessuna delle condizioni è soddisfatta, la persona potrebbe non avere un attacco cardiaco.

Esempio di Query:

?- puo_avere_attacco_cardiaco(55, 1, 1, 1, 2, 3, 140, 1.5, PuoAvereAttacco).

Risultato:

PuoAvereAttacco = no.

```
% Regola 1: per determinare se una persona può avere un attacco cardiaco.
puo_avere_attacco_cardiaco(Eta, TipoDolore, AnginaEsercizio, Pendenza, NumeroVasi, RisultatoThallium, FrequenzaCardiaca, PiccoPrecedente, 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
        (Eta > 50; Eta < 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.
```

Regola 2:

Determinare se una persona può avere un attacco cardiaco (con somma di valori)

In questa regola, ogni condizione viene valutata separatamente. Se la condizione è vera, il suo valore è 0; se non è vera, il valore è 1. Poi, questi valori vengono sommati. Se la somma è maggiore di 4, la persona potrebbe avere un attacco cardiaco, altrimenti no.

Esempio di Query:

?- puo_avere_attacco_cardiaco_prob(55, 1, 0, 1, 3, 3, 120, 1.5, PuoAvereAttacco).

Risultato:

PuoAvereAttacco = no.

```
% Regola 2: per determinare se una persona può avere un attacco
% cardiaco, usando una somma di valori.
puo_avere_attacco_cardiaco_prob(Eta, 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
        ((Eta > 50; Eta < 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 +
    ValoreCondizione6 + ValoreCondizione7 + ValoreCondizione8,
    % Se la somma è maggiore di 4, allora può avere un attacco cardiaco
    (SommaCondizioni > 4 -> PuoAvereAttacco = si; PuoAvereAttacco = no).
```

Regola 3:

Calcolare l'età media delle persone con attacco cardiaco

Questa regola calcola l'età media delle persone con attacco cardiaco. Prima estrae le età delle persone dal database, poi calcola la somma delle età e divide per il numero di persone per ottenere l'età media.

Esempio di Query:

?- eta_media_persone_attacco_cardiaco(MediaEtà).

Risultato:

MediaEtà = 55.6

```
% 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
```

Regola 4: Determinare il tipo di dolore toracico più comune

Questa regola trova il tipo di dolore toracico più comune tra i dati. I tipi di dolore toracico vanno da 0 a 3. La regola analizza quale tipo è presente più frequentemente.

Esempio di Query:

?- tipo_dolore_piu_comune(TipoDolorePiuComune).

Risultato:

TipoDolorePiuComune = 2.

```
% 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
```

Regola 5:
Contare quanti individui hanno ciascun tipo di dolore toracico

Questa regola conta quante persone hanno ciascun tipo di dolore toracico, utilizzando una lista di dati.

Esempio di Query:

?- conta_tipi_dolore([0, 1, 2, 2, 3, 3, 2], TipiDoloreConteggi).

Risultato:

TipiDoloreConteggi = [(0,1), (1,1), (2,3), (3,2)].

```
% Regola 5: per contare quanti individui hanno ciascun tipo di dolore toracico
conta_tipi_dolore([], _, []).
conta_tipi_dolore([Tipo|Resto], TipiDolore, [(Tipo, Conteggio)|RestoConteggi]) :-
    conta_occorrenze_tipo(Tipo, TipiDolore, Conteggio),
    conta_tipi_dolore(Resto, TipiDolore, RestoConteggi).
```

Regola 6:
Contare quante volte un tipo di dolore appare nella lista

Questa regola calcola quante volte un determinato tipo di dolore appare in una lista di dati.

Esempio di Query:

?- conta_occorrenze_tipo(2, [0, 1, 2, 2, 3, 3, 2], Conteggio).

Risultato:

Conteggio = 3.

```
% Regola 6: per contare quante volte un tipo di dolore appare nella lista
conta_occorrenze_tipo(_, [], 0).
conta_occorrenze_tipo(Tipo, [Tipo|Resto], Conteggio) :-
    conta_occorrenze_tipo(Tipo, Resto, ConteggioResto),
    Conteggio is ConteggioResto + 1.
conta_occorrenze_tipo(Tipo, [_|Resto], Conteggio) :-
    conta_occorrenze_tipo(Tipo, Resto, Conteggio).
```

Regola 7:
Trovare il tipo di dolore toracico più comune

Questa regola identifica il tipo di dolore toracico che appare con maggiore frequenza tra i dati. La regola esamina tutte le occorrenze e determina quale tipo è più comune.

Esempio di Query:

?- trova_tipo_piu_comune([(0,1), (1,1), (2,3), (3,2)], TipoDolorePiuComune).

Risultato:

TipoDolorePiuComune = 2.

```
% Regola 7: per trovare il tipo di dolore più comune
trova_tipo_piu_comune([(Tipo, Conteggio)], Tipo).
trova_tipo_piu_comune([(Tipo1, Conteggio1), (Tipo2, Conteggio2)|Resto], TipoPiuComune) :-
    (Conteggio1 >= Conteggio2 -> trova_tipo_piu_comune([(Tipo1, Conteggio1)|Resto], TipoPiuComune)
    ; trova_tipo_piu_comune([(Tipo2, Conteggio2)|Resto], TipoPiuComune)).
```

Apprendimento supervisionato

Sommario

Questo caso di studio si concentra sulla diagnosi della possibilità che un paziente abbia un problema cardiaco, con particolare attenzione alla previsione della variabile "output", che indica la presenza di una condizione cardiaca, utilizzando altre caratteristiche del paziente. A causa della limitata quantità di dati a disposizione, sono stati scartati modelli ad alta complessità, come la regressione lineare e le reti neurali. In seguito, sono stati selezionati e valutati i seguenti modelli:

- **K-Nearest Neighbors (KNN)**
  utilizzando la libreria Scikit-learn, un algoritmo non parametrico che classifica un campione in base alla maggioranza dei suoi vicini più prossimi.

- **Random Forest**
  anch'esso implementato in Scikit-learn, un ensemble di alberi decisionali che è noto per la sua robustezza e capacità di gestire dati complessi.

- **Decision Tree**
  un altro modello basato su alberi decisionali implementato in Scikit-learn, utile per le sue capacità di interpretabilità e di gestione di variabili sia numeriche che categoriche.

Decisioni di Progetto

Inizialmente, i vari modelli sono stati testati senza ottimizzazione di parametri specifici, al fine di ottenere una valutazione preliminare delle loro performance. I risultati iniziali ottenuti sono stati i seguenti:
In seguito, è stato generato un *classification_report* per ottenere una valutazione più dettagliata delle prestazioni dei modelli, includendo metriche come *precision* e *recall*.

- **Precisione**
  Questa metrica è utile per ridurre il numero di falsi positivi, che si verificano quando il modello classifica erroneamente un caso come positivo, pur essendo negativo. Un'alta precisione indica che, quando il modello predice un caso come positivo, è molto probabile che lo sia effettivamente, minimizzando i falsi positivi.
- **Recall**
  Questa metrica mira a ridurre il numero di falsi negativi, ossia i casi in cui il modello predice erroneamente un risultato negativo, pur essendo positivo. Un *recall* elevato indica che il modello riesce a individuare la maggior parte dei casi positivi, riducendo al minimo i falsi negativi.

Queste misure sono fondamentali per valutare le performance di un modello, soprattutto in scenari di classificazione in cui una delle classi è notevolmente meno frequente dell'altra. In un contesto medico, come la diagnosi di attacchi cardiaci, è cruciale ridurre sia i falsi positivi (che potrebbero generare ansia nei pazienti) sia i falsi negativi (che potrebbero avere conseguenze gravi se non rilevati).
L'ottimizzazione di queste metriche può risultare complessa, poiché solitamente esiste un compromesso tra precisione e recall. La scelta dell'ottimizzazione dipenderà dalle priorità specifiche del problema, come l'importanza di evitare falsi positivi rispetto ai falsi negativi.
Dai risultati iniziali, il modello *Random Forest* ha mostrato le migliori prestazioni in termini di *F2 Score* e *Precisione*. I modelli *Decision Tree* e *K-Nearest Neighbors (KNN)* hanno registrato prestazioni leggermente inferiori.
Successivamente, si è proceduto a ottimizzare i parametri dei modelli *Random Forest* e *Decision Tree*, sfruttando specifici iperparametri per migliorare le loro prestazioni. Il modello *KNN*, invece, non è stato addestrato con iperparametri ottimizzati. Vediamo nel dettaglio le modifiche apportate.

```
PS C:\Users\giuse\Desktop\prog.r\Icon24-25\classifier> python decision_tree.py
Cross-validation accuracy: [0.81818182 0.83333333 0.74074074 0.7962963  0.81481481]

Mean accuracy: 0.8006734006734006
Std deviation: 0.03220024300607472
Test accuracy: 0.8064516129032258

Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.93      0.81        14
           1       0.92      0.71      0.80        17

    accuracy                           0.81        31
   macro avg       0.82      0.82      0.81        31
weighted avg       0.83      0.81      0.81        31
```

```
C:\Users\giuse\Desktop\prog.r\Icon24-25\classifier>python random_forest.py
Cross-validation accuracy: [0.81818182 0.85185185 0.83333333 0.77777778 0.87037037]

Mean accuracy: 0.8303030303030303
Std deviation: 0.03157452176296343
Test accuracy: 0.8709677419354839

Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.86      0.86        14
           1       0.88      0.88      0.88        17

    accuracy                           0.87        31
   macro avg       0.87      0.87      0.87        31
weighted avg       0.87      0.87      0.87        31
```

```
C:\Users\giuse\Desktop\prog.r\Icon24-25\classifier>py knn.py
[0.65454545 0.66666667 0.61111111 0.51851852 0.7037037 ]

mean = 0.630909090909091
std = 0.06349793434165 95
0.6451612903225806
0.6774193548387096
              precision    recall  f1-score   support

           0       0.61      0.79      0.69        14
           1       0.77      0.59      0.67        17

    accuracy                           0.68        31
   macro avg       0.69      0.69      0.68        31
weighted avg       0.70      0.68      0.68        31
```

In seguito, è stato creato un *classification_report* per ottenere una valutazione più dettagliata delle prestazioni dei modelli, includendo metriche come la **precision** e il **recall**.

- ## Precision (Precisione)

  Questa metrica è utile per ridurre il numero di falsi positivi, che si verificano quando il modello classifica erroneamente un caso come positivo, pur essendo negativo. Un'alta precisione indica che, quando il modello predice un caso come positivo, è molto probabile che lo sia effettivamente, minimizzando i falsi positivi.

- ## Recall

  Questa metrica mira a ridurre il numero di falsi negativi, ossia i casi in cui il modello predice erroneamente un risultato negativo, pur essendo positivo. Un recall elevato indica che il modello riesce a individuare la maggior parte dei casi positivi, riducendo al minimo i falsi negativi.

Queste misure sono fondamentali per valutare le performance di un modello, soprattutto in scenari di classificazione in cui una delle classi è notevolmente meno frequente dell'altra. In un contesto medico, come la diagnosi di attacchi cardiaci, è cruciale ridurre sia i falsi positivi (che potrebbero generare ansia nei pazienti) sia i falsi negativi (che potrebbero avere conseguenze gravi se non rilevati).

L'ottimizzazione di queste metriche può risultare complicata, poiché solitamente c'è un compromesso tra precisione e recall. L'ottimizzazione dipenderà dalle priorità specifiche del problema, come l'importanza di evitare falsi positivi rispetto a falsi negativi.

Dai risultati iniziali, il modello **Random Forest** ha mostrato le migliori prestazioni in termini di F2 Score e Precisione. I modelli **Decision Tree** e **K-Nearest Neighbors (KNN)** hanno registrato prestazioni leggermente inferiori.

Successivamente, si è proceduto a ottimizzare i parametri dei modelli **KNN**, **Decision Tree** e **Random Forest**, poiché questi avevano maggiori margini di miglioramento. Vediamo nel dettaglio le modifiche apportate.

Decision Tree

Esistono diversi iperparametri nell'algoritmo dell'albero decisionale che possono essere configurati per ottimizzare le prestazioni del modello. Di seguito vengono descritti alcuni dei principali:

- **max_depth**
  La profondità massima dell'albero. Questo parametro determina il numero massimo di divisioni (livelli) che l'albero può fare. Una profondità elevata può causare sovradattamento **(**over fitting**)**, ossia il modello si adatta troppo ai dati di addestramento, perdendo la capacità di generalizzare su nuovi dati. Al contrario, una profondità bassa può portare a **sotto** adattamento **(**under fitting**)**, in cui il modello non riesce a catturare la complessità dei dati.

- **min_samples_leaf**
  Il numero minimo di campioni richiesti in un nodo foglia (ovvero alla fine di un ramo dell'albero). Se, dopo una divisione, il numero di campioni in uno dei nuovi rami è inferiore a questo valore, la divisione non viene effettuata. Un valore maggiore aiuta a evitare l'overfitting, mentre un valore troppo basso potrebbe permettere al modello di imparare troppo dai dati specifici, aumentando il rischio di overfitting.

- **Splitter**
  La strategia utilizzata per selezionare l'attributo migliore per la divisione. Le due opzioni principali sono:
    - **"best"**: Viene scelto il miglior attributo disponibile per fare la divisione.
    - **"random"**: Viene scelto un attributo casuale tra quelli disponibili per la divisione.

Ora analizziamo ciascuno di questi iperparametri separatamente e successivamente combineremo i valori ottimali per trovare la configurazione che massimizza l'accuratezza del modello.
Per quanto riguarda max_depth e min_samples_leaf, i risultati ottenuti sono i seguenti:

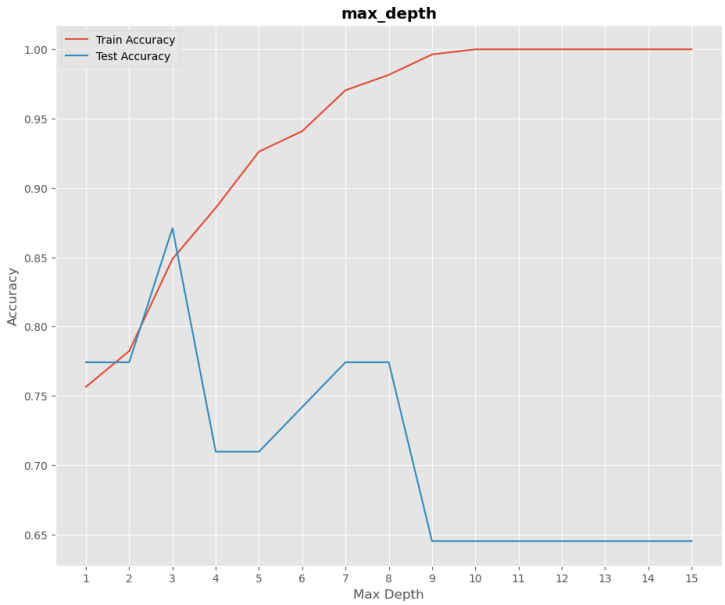


Figura 1

Come mostrato in **Figura 1**, il parametro ottimale in questo caso è il valore della profondità dell'albero, che risulta essere **3**. Con questa configurazione, l'accuratezza sui dati di test supera addirittura quella sui dati di addestramento, il che suggerisce che il modello, con questo parametro, possiede un'ottima capacità di generalizzazione.

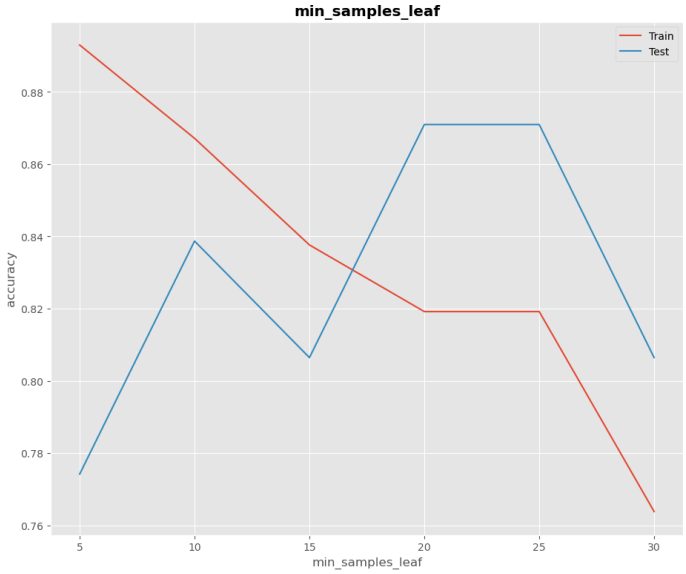


Figura 2

Come evidenziato dal grafico in Figura 2, con l'aumento del numero di elementi nel nodo foglia, l'accuratezza del modello tende a diminuire. Tuttavia, osservando i dati sui test, si nota che i valori migliori si verificano quando il numero di elementi nel nodo foglia è pari a 20 e 25. Questo suggerisce che il modello raggiunge la migliore accuratezza e una buona capacità di apprendimento a questi specifici valori.

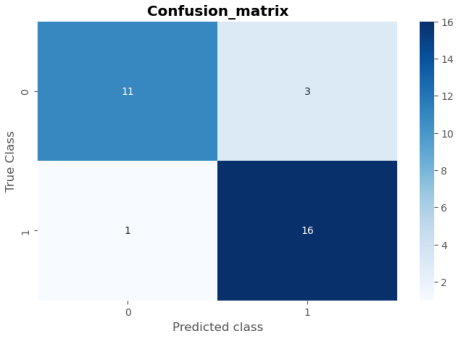
Con un numero elevato di valori nel nodo foglia, il modello perde la capacità di generalizzare e finisce per "memorizzare" il set di dati di addestramento, senza adattarsi correttamente. Ad esempio, con un valore di **5**, il modello si riaddestra facilmente a causa del basso numero di valori nel nodo foglia, ma in questo caso memorizza i dati senza generalizzarli, risultando inefficace su nuovi dati. Pertanto, scegliere il numero giusto di valori nel nodo foglia è fondamentale per garantire che il modello possa generalizzare correttamente 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 aver ottimizzato i modelli, abbiamo esaminato separatamente l'accuratezza, che è aumentata dal 71**%** all'87%, un miglioramento che rappresenta un risultato molto positivo.

Random Forest

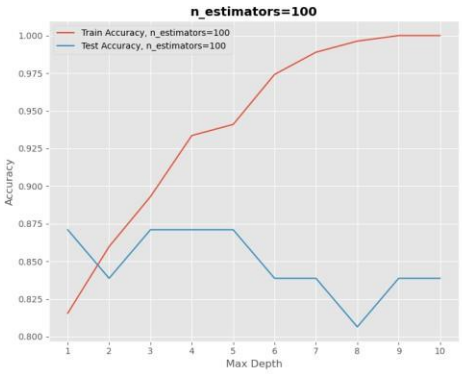
In un modello di **Random Forest**, ci sono diversi **iperparametri** che possono essere personalizzati per ottimizzare le prestazioni. Di seguito sono descritti alcuni dei principali:

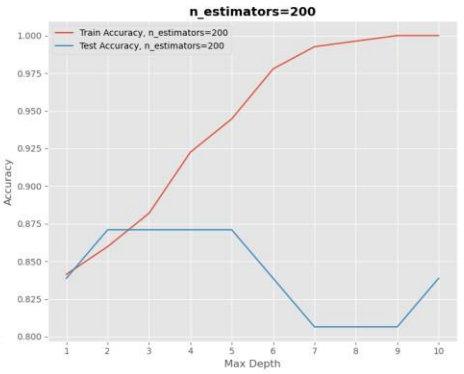
- **n_estimators**
  Il numero di alberi presenti nella foresta. Questo parametro determina quanti alberi saranno creati nell'insieme. Un numero maggiore di alberi può migliorare la performance del modello, ma potrebbe anche aumentare il tempo di addestramento. Il valore predefinito è **100**.

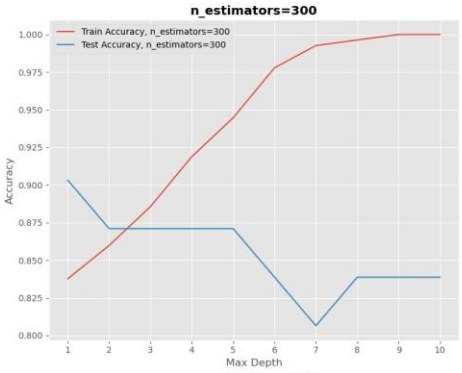
- **max_depth**
  La profondità massima di ciascun albero. Limitare la profondità degli alberi può aiutare a prevenire l'**overfitting**. Se impostato su **None**, gli alberi cresceranno fino a quando tutte le foglie saranno pure o fino a raggiungere il numero minimo di campioni per la divisione.

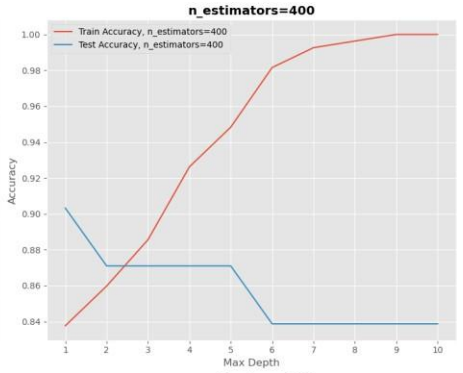
- **min_samples_leaf**
  Il numero minimo di campioni richiesti per formare una foglia dell'albero. Se il numero di campioni in un nodo è inferiore al valore specificato, la divisione non verrà effettuata. Aumentare questo valore può ridurre il rumore nei dati e migliorare la capacità di generalizzazione del modello. Il valore predefinito è **1**.

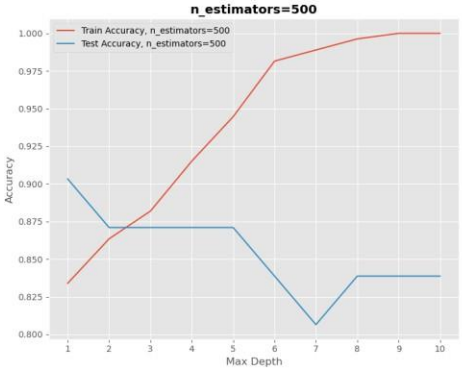
- **max_features**
  Il numero di caratteristiche (features) prese in considerazione per ciascuna divisione dell'albero. Limitare il numero di caratteristiche disponibili per la separazione può aiutare a combattere l'overfitting. Per impostazione predefinita, il valore è **"auto"**, che corrisponde alla radice quadrata del numero di caratteristiche.

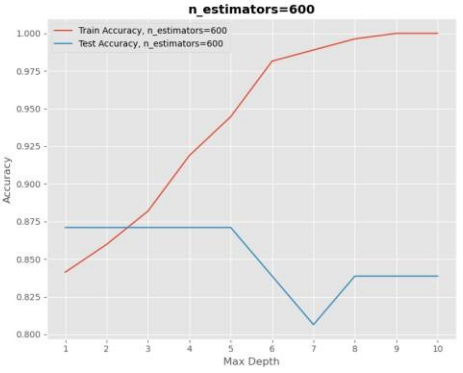

n_estimators=100
Train Accuracy, n_estimators=100
Test Accuracy, n_estimators=100
Accuracy
Max Depth

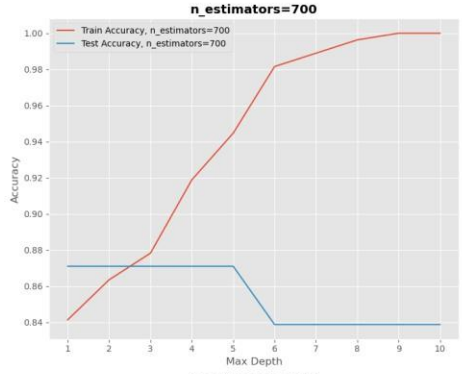

n_estimators=200
Train Accuracy, n_estimators=200
Test Accuracy, n_estimators=200
Accuracy
Max Depth

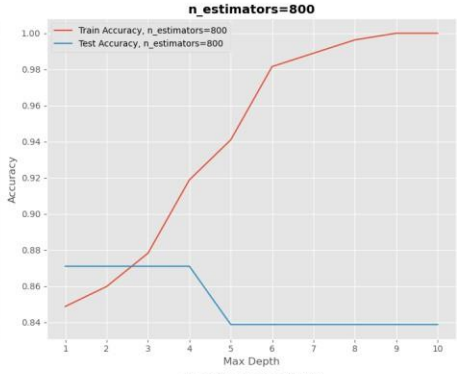

n_estimators=300
Train Accuracy, n_estimators=300
Test Accuracy, n_estimators=300
Accuracy
Max Depth

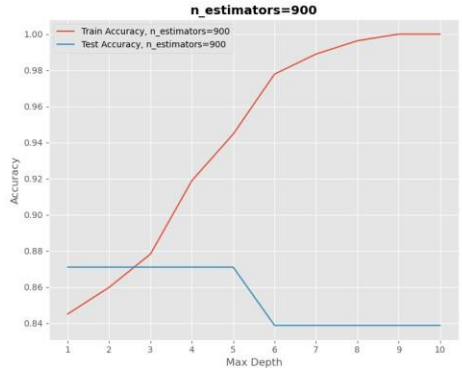

n_estimators=400
Train Accuracy, n_estimators=400
Test Accuracy, n_estimators=400
Accuracy
Max Depth

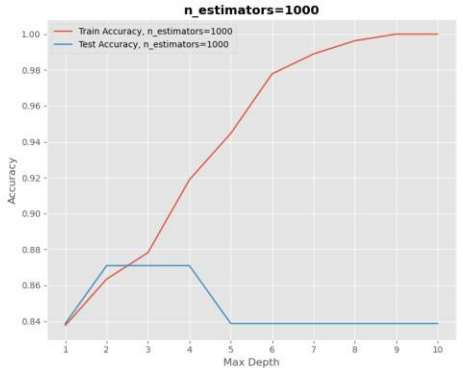

n_estimators=500
Train Accuracy, n_estimators=500
Test Accuracy, n_estimators=500
Accuracy
Max Depth


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

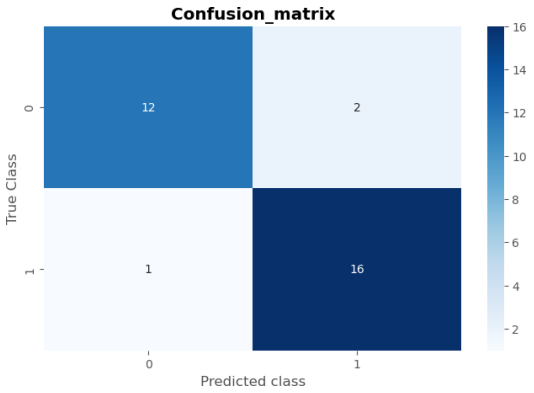
Il modello ha mostrato un'alta accuratezza con parametri come 300, 400 e 500 alberi, e una profondità dell'albero pari a **1**. Dopo aver addestrato il modello e testato su un campione di test utilizzando questi parametri, si ottengono i seguenti risultati:

```
[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%, migliorando il risultato dal 81% al 90%. Ora, il modello Random Forest risulta essere il migliore, superando l'albero decisionale, che ha ottenuto un'accuratezza dell'87% anche dopo l'ottimizzazione degli iperparametri.

Anche altre metriche, come precisione e recall, sono migliorate nel modello Random Forest. La precisione è diventata più accurata, riducendo il numero di falsi positivi (un aspetto cruciale in un contesto medico). Il **recall** è anch'esso migliorato, diminuendo i falsi negativi (ovvero, riducendo il rischio che il modello non rilevi un paziente malato quando lo è davvero), un aspetto fondamentale quando si tratta di diagnosi mediche, dove la vita delle persone è in gioco.

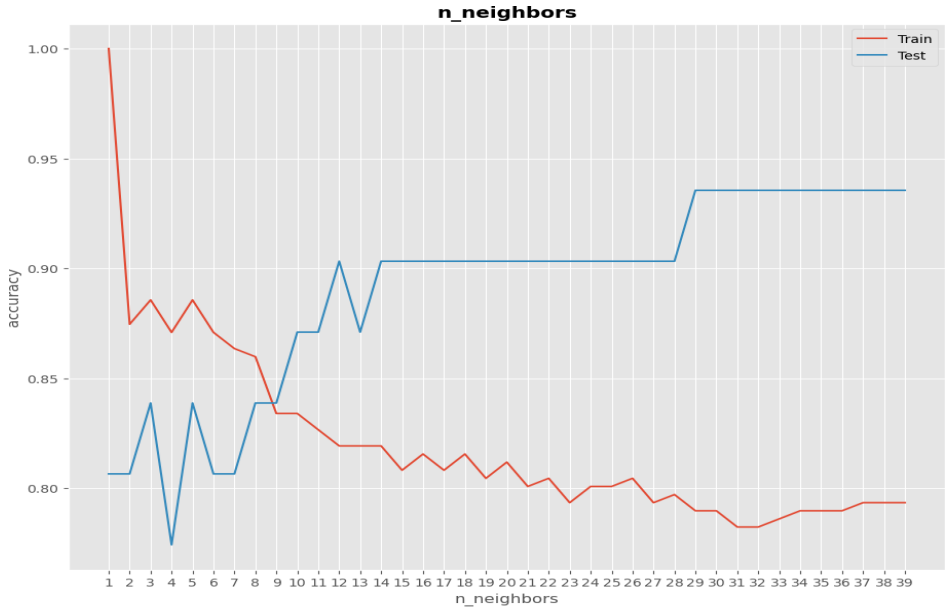
La conclusione di questa analisi è che l'accuratezza delle previsioni è aumentata di quasi il **10%**, rappresentando un miglioramento significativo. Grazie alla regolazione dei parametri, in questa fase, il modello Random Forest risulta essere il migliore.

KNN

Per ottimizzare il modello K-Nearest Neighbors (KNN), che si basa sulla distanza tra i punti dati e le classi dei vicini più prossimi, è fondamentale standardizzare i dati per garantire che tutte le variabili abbiano lo stesso peso. Dopo aver standardizzato i dati, si è osservato un miglioramento significativo dell'accuratezza del modello nel campione di test, passando dal 64% all'83%, un risultato decisamente positivo. I passi successivi per ottimizzare ulteriormente il modello prevedono la configurazione degli iperparametri, come il numero di vicini e la scelta della metrica di distanza da utilizzare.

Per l'algoritmo k-Nearest Neighbors (kNN), i principali iperparametri da configurare includono: n_neighbors

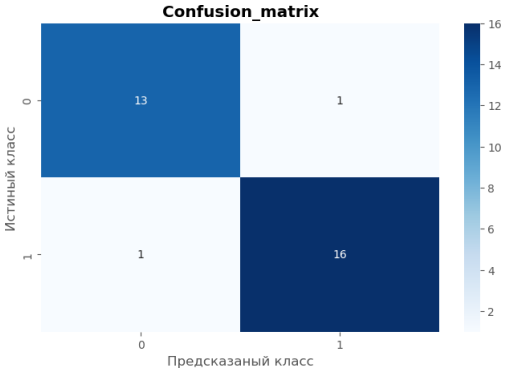
> Il numero di vicini più prossimi da considerare durante il processo di classificazione o regressione. Questo è uno degli iperparametri più rilevanti per il funzionamento del kNN.



Come mostrato nel grafico, l'accuratezza del modello aumenta con l'aumento del numero di vicini, raggiungendo il suo valore massimo quando n_neighbors è pari a 29. Questo valore indica che il modello ha una buona capacità di generalizzazione. Ad esempio, con un valore di 1, il modello si baserebbe solo sul primo vicino per effettuare la classificazione. Al contrario, utilizzando un numero elevato di vicini, i risultati vengono confrontati tra loro e viene selezionato il valore più comune tra i vicini. Pertanto, addestrare il modello con 29 vicini implica considerare una varietà di punti dati vicini prima di prendere una decisione di classificazione. Questo approccio, che si basa su un "voto di maggioranza", tende a produrre previsioni più accurate in molte situazioni.

Di seguito sono riportati 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



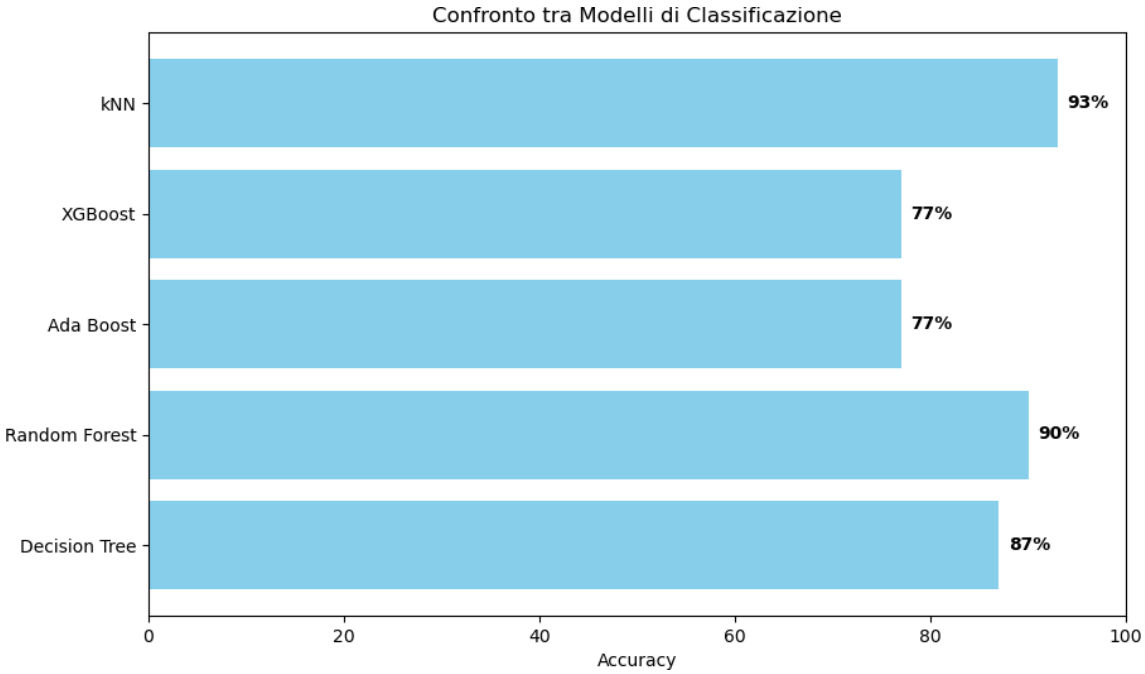
Il risultato ottenuto indica che il modello k-Nearest Neighbors (kNN) è diventato il migliore tra tutti quelli testati. L'accuratezza del modello ha raggiunto il 93%, con un miglioramento significativo rispetto al valore iniziale del 64%.

Le metriche di valutazione, come precisione e richiamo, sono migliorate notevolmente. In particolare, il modello commette solo circa il 6% - 7% di falsi positivi (cioè quando il modello etichetta erroneamente una persona sana come malata) e un altro 6% - 7% di falsi negativi (cioè quando il modello non riesce a riconoscere una persona malata, classificandola come sana).

Questi risultati sono particolarmente rilevanti nel contesto di un dataset medico, dove minimizzare i falsi positivi e negativi è cruciale. Pertanto, l'accuratezza finale del 94% rappresenta un risultato molto positivo, confermando l'efficacia del modello kNN nel classificare correttamente i casi.

Conclusioni

In generale, il modello k-Nearest Neighbors (kNN) emerge come il migliore tra quelli valutati, raggiungendo un'accuratezza del 93%. Inoltre, il modello gestisce efficacemente i falsi positivi e falsi negativi, due metriche particolarmente rilevanti in un contesto medico, dove la precisione nelle diagnosi è fondamentale.



Test del Modello su Dati Esterni al Dataset

In questo capitolo sono stati testati i modelli KNN, Random Forest e Decision Tree utilizzando dati generati realistici. I risultati sono i seguenti:

Esempio 1:

- Età: 63, Sesso: Maschio, Dolore toracico: 3, Pressione sanguigna: 145, Colesterolo: 233, Glicemia a digiuno: 1, Elettrocardiogramma: 0, Frequenza cardiaca massima: 150, Esercizio anginoso: No, Depressione ST: 2.3, Pendenza ST: 0, Vasi colorati: 0, Thalassemia: 1, Output (malattia cardiaca): 1.

Esempio 2:

- Età: 41, Sesso: Femmina, Dolore toracico: 1, Pressione sanguigna: 130, Colesterolo: 204, Glicemia a digiuno: 0, Elettrocardiogramma: 0, Frequenza cardiaca massima: 172, Esercizio anginoso: No, Depressione ST: 1.4, Pendenza ST: 2, Vasi colorati: 0, Thalassemia: 2, Output (malattia cardiaca): 1.

Risultati:

- **Random Forest** e **Decision Tree**
  hanno predetto correttamente che entrambe le persone sono a rischio di malattia cardiaca (output = 1).

```
Esempio 1: Il modello predice che la persona puo avere un attacco di cuore.
Esempio 2: Il modello predice che la persona puo avere un attacco di cuore.
```

```
Esempio 1: Il modello predice che la persona puo avere un attacco di cuore.
Esempio 2: Il modello predice che la persona puo avere un attacco di cuore.
```

- Il modello **KNN** ha erroneamente predetto che nessuna delle due persone fosse a rischio (output = 0).

```
Esempio 1: Il modello predice che la persona non puo avere un attacco di cuore.
Esempio 2: Il modello predice che la persona non puo avere un attacco di cuore.
```

Suggerimenti per migliorare KNN:

1. **Ottimizzazione del parametro k**: Scegliere un valore ottimale di k attraverso la validazione incrociata.
2. **Scaling avanzato**: Utilizzare tecniche di scaling come la normalizzazione Min-Max per migliorare il calcolo delle distanze.
3. **Distanze ponderate**: Impiegare distanze ponderate, dove i vicini più prossimi hanno un peso maggiore.
4. **Selezione delle feature**: Rimuovere le variabili non rilevanti per migliorare la performance.
5. **Adattamento delle metriche di distanza**: Testare metriche di distanza alternative per affinare la capacità del modello di identificare pattern complessi.

Questi miglioramenti potrebbero affinare le predizioni di KNN, rendendolo più preciso.

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