Heart Disease Prediction

Mennea Antonio

Sommario

Introduzione

Il progetto **Heart Disease Prediction** è un sistema ibrido che integra tecniche di apprendimento supervisionato con ragionamento basato su regole per la diagnosi delle malattie cardiache. In pratica, il sistema utilizza modelli di machine learning – in particolare Decision Tree e Random Forest – per analizzare dati clinici e prevedere la presenza di malattia. Inoltre, grazie all'integrazione con Prolog, il progetto applica regole logiche per identificare pazienti ad alto rischio e per arricchire le previsioni con un livello interpretativo basato sulla conoscenza medica. Un algoritmo di ricerca, basato su Breadth-First Search (BFS), viene anche impiegato per individuare pazienti con caratteristiche simili, offrendo così un ulteriore strumento di supporto alle decisioni cliniche. In sintesi, il progetto unisce modelli statistici e ragionamento simbolico per fornire una diagnosi assistita, mirata a migliorare l'accuratezza e l'interpretabilità dei risultati.

Supervised Learning

Nel supervised learning (apprendimento supervisionato) si lavora con un insieme di dati etichettati, dove ogni esempio di input è associato a un output corretto. In pratica, il modello impara a mappare gli input ai rispettivi output, così da poter fare predizioni su nuovi dati.

Nel contesto del progetto Heart Disease Prediction, il supervised learning viene utilizzato per addestrare modelli come il Decision Tree e la Random Forest. Questi modelli vengono "insegnati" a riconoscere, dai dati clinici dei pazienti (ad esempio, età, pressione, colesterolo), se un paziente ha o meno una malattia cardiaca. L'obiettivo è che, dopo l'addestramento, il sistema sia in grado di prevedere correttamente lo stato di salute di nuovi pazienti.

Data Preprocessing

Il **Data Preprocessing (Pre-elaborazione dei Dati)** è una fase fondamentale nell'ingegneria della conoscenza perché permette di trasformare dati grezzi e potenzialmente rumorosi in informazioni pulite e coerenti, pronte per essere utilizzate da modelli di apprendimento automatico e sistemi intelligenti.

In pratica, il processo di pre-elaborazione si articola in diverse operazioni:

- **Pulizia dei dati:**
  I dati raccolti in ambito reale spesso contengono errori, valori mancanti o outlier. La pulizia consiste nel gestire queste anomalie – ad esempio, sostituendo o eliminando i valori mancanti – per evitare che influenzino negativamente il modello.
- **Trasformazione e codifica:**
  Molti dataset contengono variabili categoriche (come il sesso, il tipo di dolore toracico, ecc.) che devono essere convertite in formato numerico, operazione spesso realizzata mediante tecniche come il One-Hot Encoding. Questo passaggio è cruciale perché gli algoritmi di machine learning generalmente richiedono input numerici.
- **Normalizzazione e standardizzazione:**
  Le variabili numeriche possono avere scale molto diverse (ad esempio, l'età rispetto alla pressione arteriosa). La normalizzazione o la standardizzazione (ad esempio, trasformando i dati in modo che abbiano media 0 e deviazione standard 1) garantiscono che tutte le feature contribuiscano in maniera equilibrata all'apprendimento del modello, evitando che alcune variabili "pesino" troppo nell'analisi.

- **Suddivisione del dataset:**
  Infine, il dataset viene diviso in set di training e di test. Questo permette di addestrare il modello su una parte dei dati e di valutarne le performance su dati "nuovi", riducendo il rischio di overfitting e garantendo una migliore capacità di generalizzazione.

Collegamento con il progetto:

Nel progetto di **predizione delle malattie cardiache**, questi passaggi di pre-elaborazione vengono applicati in modo pratico per garantire l'affidabilità dei modelli predittivi:

- **Controllo e pulizia dei dati:**
  Il dataset contenente le informazioni sui pazienti viene verificato per la presenza di valori mancanti e per eventuali anomalie, assicurando così che ogni record sia completo e utilizzabile.
- **Codifica delle variabili categoriche:**
  Variabili come sex, cp, fbs, restecg e altre vengono convertite in variabili dummy tramite il One-Hot Encoding, rendendo possibile l'analisi numerica di dati originariamente qualitativi.
- **Standardizzazione delle variabili numeriche:**
  Caratteristiche come age, trestbps e chol vengono standardizzate usando strumenti come il StandardScaler, garantendo che tutte le feature siano sulla stessa scala e che il modello non dia importanza eccessiva a variabili con range più ampi.
- **Divisione in training e test set:**
  Il dataset viene diviso (tipicamente in una proporzione di 80% per l'addestramento e 20% per il test) per consentire una valutazione corretta e imparziale delle performance dei modelli, come l'Albero Decisionale e la Random Forest.

Questa fase di Data Preprocessing è quindi essenziale per assicurare che i modelli abbiano dati di alta qualità su cui operare, migliorando così l'accuratezza e l'affidabilità delle predizioni, soprattutto in contesti delicati come la diagnosi di malattie cardiache.

Cross-Validation

La **Cross-Validation (Validazione Incrociata)** è una tecnica fondamentale per valutare la capacità di generalizzazione di un modello, cioè la sua affidabilità quando si applica a dati nuovi.

Concetti chiave:

- **Divisione del dataset:**
  Il dataset viene suddiviso in *k* parti (dette *fold*). Ad esempio, in una validazione a 10 fold, il dataset viene diviso in 10 segmenti.
- **Iterazioni multiple:**
  In ogni iterazione, il modello viene addestrato utilizzando *k-1* fold e testato sul fold rimanente. Questo processo viene ripetuto *k* volte, cambiando ad ogni volta il fold di test.
- **Valutazione complessiva:**
  Si calcolano le metriche di performance (come accuratezza, precisione, recall, ecc.) per ogni iterazione e si prende la media. Questo fornisce una stima più robusta e affidabile della performance del modello rispetto a una singola divisione training/test.

- **Riduzione dell'overfitting:**
  Utilizzando differenti suddivisioni, la validazione incrociata aiuta a evitare che il modello si adatti troppo specificamente ai dati di un singolo training set, migliorandone la capacità di generalizzazione.

Collegamento con il progetto:

Nel progetto di predizione delle malattie cardiache, la validazione incrociata viene utilizzata per:

- **Ottimizzazione degli iperparametri:**
  Per il **Decision Tree**, il codice esegue una validazione incrociata (ad esempio, a 10 fold) testando differenti profondità dell'albero. Questo permette di scegliere la profondità che porta ai migliori punteggi medi, migliorando così l'efficacia del modello.
- **Confronto tra modelli:**
  Per il **Random Forest**, si valuta l'impatto di differenti valori del numero di stimatori (n_estimators) tramite la validazione incrociata a 5 fold. In questo modo si può individuare il numero ottimale di stimatori che bilancia bene tra bias e varianza.

Utilizzando la cross-validation, il progetto si assicura che i modelli non siano troppo "addestrati" su una specifica suddivisione dei dati, ma abbiano performance stabili e affidabili anche su dati non visti. Questo approccio migliora la robustezza e la credibilità delle predizioni, aspetto cruciale soprattutto in contesti sensibili come la diagnosi di malattie cardiache.

Evaluation Metrics

Le **Metriche di Valutazione** sono strumenti fondamentali per misurare quanto un modello di machine learning sia efficace nel compito di classificazione o regressione. In altre parole, queste metriche quantificano le performance del modello e aiutano a capire se le sue previsioni sono attendibili, sia nel riconoscimento di casi positivi che nel limitare gli errori.

Principali metriche di valutazione:

- **Accuratezza (Accuracy):**
  Indica la percentuale di previsioni corrette sul totale delle previsioni effettuate. È utile per avere una panoramica generale, ma può essere poco informativa in presenza di classi sbilanciate.
- **Precisione (Precision):**
  Misura la proporzione di veri positivi rispetto a tutte le istanze classificate come positive. È particolarmente importante quando il costo dei falsi positivi è elevato.
- **Recall (Sensibilità):**
  Rappresenta la capacità del modello di identificare correttamente tutti i casi positivi presenti nel dataset. È fondamentale quando è critico ridurre i falsi negativi.
- **F1-Score:**
  È la media armonica tra precisione e recall, offrendo un bilanciamento utile quando si desidera considerare entrambi gli aspetti contemporaneamente.
- **ROC-AUC (Area Under the ROC Curve):**
  Valuta la capacità del modello di discriminare tra le classi positive e negative in base a diversi valori di soglia. Un valore più alto indica un migliore potere discriminante.

Collegamento con il progetto:

Nel progetto di **predizione delle malattie cardiache**, le metriche di valutazione assumono un ruolo cruciale per assicurarsi che il modello identifichi correttamente i pazienti a rischio. Ad esempio:

- **Accuratezza:** Fornisce una prima stima di quante volte il modello classifica correttamente i pazienti, offrendo una visione complessiva della sua performance.
- **Precisione e Recall:** Sono particolarmente importanti in ambito sanitario. Una precisione elevata significa che, tra i pazienti segnalati come ad alto rischio, la maggior parte lo è effettivamente, riducendo allarmismi inutili. Un recall elevato garantisce che il modello riesca a individuare il maggior numero possibile di pazienti a rischio, minimizzando il rischio di falsi negativi.
- **F1-Score:** Permette di bilanciare la precisione e il recall, risultando particolarmente utile quando occorre trovare un compromesso tra l'identificazione corretta dei pazienti a rischio e la riduzione degli errori.
- **ROC-AUC:** Aiuta a valutare la capacità del modello di distinguere tra pazienti ad alto rischio e quelli non a rischio, indipendentemente dalla soglia scelta. In questo modo, si può scegliere il threshold che meglio risponde alle esigenze cliniche.

Utilizzando queste metriche, il progetto "Heart Disease Prediction" può essere affinato per garantire un alto livello di affidabilità e sicurezza, fondamentale in contesti dove le decisioni basate sulle predizioni possono influire direttamente sulla salute dei pazienti.

```
Metriche per Decision Tree:
Accuracy: 0.820
Precision: 0.862
Recall: 0.781
F1-score: 0.820
ROC-AUC: 0.822
```

```
Metriche per Random Forest:
Accuracy: 0.869
Precision: 0.875
Recall: 0.875
F1-score: 0.875
ROC-AUC: 0.869
```

Decision Trees

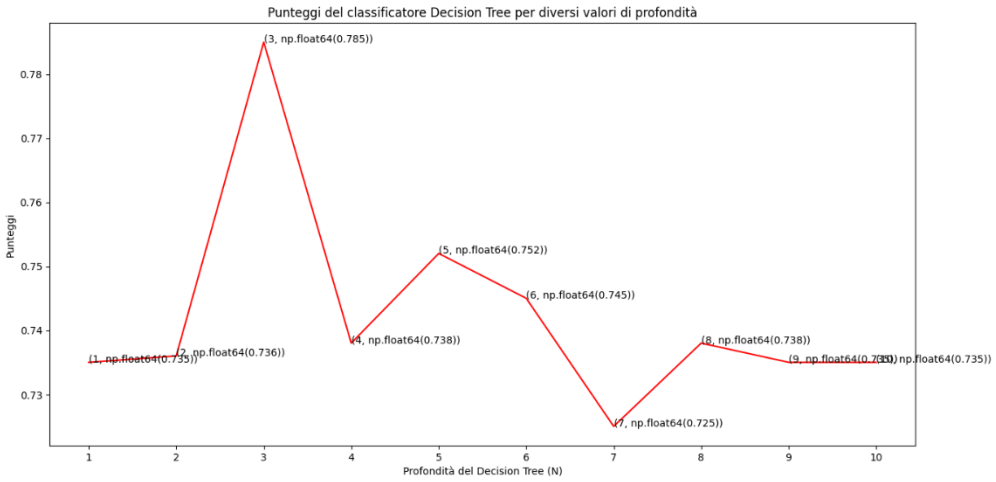
Un **Decision Tree** (Albero Decisionale) è un modello che rappresenta le decisioni attraverso una struttura ad albero. Ogni nodo interno dell'albero corrisponde a una condizione basata su una feature del dataset, mentre ogni foglia rappresenta l'esito (la classe o il valore predetto).

Questo modello viene costruito suddividendo il dataset in modo ricorsivo: a ogni nodo si sceglie la feature che meglio separa i dati. Il processo continua fino a quando non si raggiungono foglie omogenee o non sono soddisfatti determinati criteri di arresto.

Esso possiede:

- **Vantaggi:**
  - **Interpretabile:** La struttura ad albero consente di visualizzare e comprendere facilmente le regole di decisione.
  - **Flessibile:** Può gestire sia variabili numeriche che categoriche.
- **Svantaggi:**
  - **Overfitting:** Gli alberi molto profondi tendono ad adattarsi troppo ai dati di training, compromettendo la capacità di generalizzazione.
  - **Instabilità:** Piccole variazioni nei dati possono portare a alberi strutturalmente molto differenti.

Nel progetto è stato utilizzato per creare un modello interpretabile che identifica i pazienti a rischio. La scelta della profondità (ad esempio, max_depth=3) è stata ottimizzata mediante validazione incrociata, bilanciando la capacità di apprendimento e il rischio di overfitting.



Random Forest

Il **Random Forest** è una tecnica di ensemble che combina più Decision Tree per migliorare la robustezza e l'accuratezza del modello. Invece di basarsi su un singolo albero, la Random Forest costruisce numerosi alberi su campioni differenti del dataset.

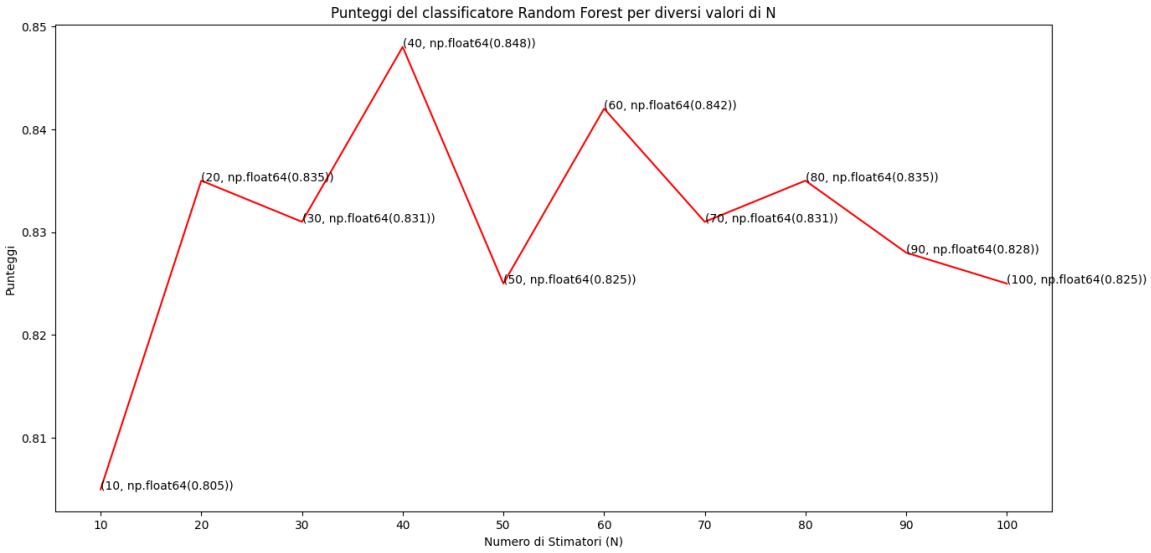
Il suo funzionamento consiste:

- **Campionamento casuale:** Ad ogni albero viene assegnato un sottoinsieme casuale del dataset (con sostituzione, noto come bootstrap).
- **Sottoinsieme di feature:** Per ogni split, viene considerato un sottoinsieme casuale delle feature, contribuendo a ridurre la correlazione tra gli alberi.
- **Aggregazione:** Le predizioni di tutti gli alberi vengono combinate (ad esempio, tramite voto maggioritario nel caso di classificazione) per produrre la previsione finale.

Esso possiede:

- **Vantaggi:**
  - **Riduzione dell'overfitting: L'aggregazione di più alberi stabilizza le predizioni e migliora la generalizzazione.**
  - **Robustezza: È meno sensibile a outlier e rumore nei dati.**
- **Svantaggi:**
  - **Meno interpretabile:** A differenza di un singolo Decision Tree, il modello risultante da una Random Forest è più difficile da interpretare nel dettaglio.
  - **Maggiore complessità computazionale:** Costruire molti alberi può richiedere più risorse, soprattutto su dataset di grandi dimensioni.

Nel progetto è stato usato per migliorare la stabilità e l'accuratezza delle predizioni. Il numero di stimatori (n_estimators) è stato valutato su diversi valori tramite cross-validation, per individuare il modello che offra il miglior compromesso tra bias e varianza.
L'approccio ensemble permette di ottenere un modello più robusto, fondamentale in un contesto clinico dove è cruciale ridurre il rischio di errori nelle diagnosi.



Algoritmi di Ricerca: Breadth-First Search

L'**algoritmo Breadth-First Search (BFS)** è una tecnica di ricerca in ampiezza utilizzata per esplorare strutture dati come grafi o alberi, visitando i nodi livello per livello.

Il suo funzionamento avviene con:

- **Esplorazione a livelli:** BFS inizia da un nodo di partenza e visita tutti i suoi vicini (cioè tutti i nodi al primo livello). Solo dopo aver esplorato completamente questo livello, l'algoritmo passa a quelli successivi (secondo livello, terzo livello, e così via).
- **Utilizzo della coda:** L'algoritmo impiega una struttura dati di tipo FIFO (First In First Out). In pratica:
    - Il nodo iniziale viene inserito nella coda;
    - Ad ogni iterazione, si estrae il nodo in testa, si visitano i suoi nodi adiacenti non ancora esplorati e li si inserisce in coda;
    - Il processo continua finché la coda non è vuota o finché non viene trovata la soluzione desiderata.
- **Garanzia di completezza e ottimalità:** Se esiste una soluzione, BFS la troverà. Inoltre, se tutti i passaggi hanno lo stesso costo, l'algoritmo garantisce di trovare il percorso con il minor numero di mosse (cioè il percorso ottimale in termini di profondità).

Anch'esso possiede:

- **Vantaggi:**
    - **Completezza:** BFS esplora sistematicamente tutti i nodi a un determinato livello prima di passare al successivo, assicurandosi di non saltare possibili soluzioni.
    - **Soluzione ottimale:** In problemi in cui ogni mossa ha lo stesso costo, BFS trova il percorso più breve o con il minor numero di passaggi.

- **Svantaggi:**
    - **Elevato consumo di memoria:** Poiché l'algoritmo deve mantenere in memoria tutti i nodi di ogni livello, può richiedere molta RAM, specialmente in grafi o alberi molto ampi.

Nel progetto esso viene utilizzato per:

- **Ricerca di pazienti simili:** Viene implementato un algoritmo BFS per trovare pazienti con caratteristiche simili, partendo da un paziente iniziale (ad esempio, basato su età e sesso). L'algoritmo esplora i "vicini" nel grafo dei pazienti (definito tramite regole Prolog) per identificare rapidamente altri pazienti che rispondono a determinati criteri, come una pressione arteriosa entro un certo range.
- **Vantaggio nel contesto clinico:** Grazie a BFS, il sistema riesce a individuare in modo sistematico e ordinato i pazienti con profili simili, garantendo che le informazioni rilevanti vengano recuperate in maniera completa e, se possibile, con il percorso più breve in termini di "distanza" (cioè, di similarità) rispetto al paziente iniziale.

```
Pazienti con caratteristiche simili trovati via BFS: [{'Età': 57, 'Sesso': 1}, {'Età': 63, 'Sesso': 1}, '{...,...}']
```

Ragionamento Basato su Regole con Prolog

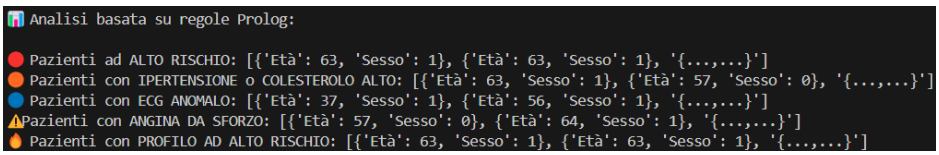
Il **Ragionamento Basato su Regole (Rule-Based Reasoning)** è un approccio dell'intelligenza artificiale che sfrutta una base di conoscenza costituita da **fatti** e **regole logiche** per dedurre nuove informazioni o prendere decisioni. In questo paradigma, la conoscenza viene espressa sotto forma di enunciati "se ... allora ..." che permettono di inferire conclusioni partendo dai dati noti.

I suoi elementi chiave sono:

- **Fatti**: Sono le informazioni di base, come dati o affermazioni, che costituiscono il punto di partenza del ragionamento. Ad esempio, in un contesto medico, un fatto potrebbe essere "Il paziente X ha una pressione alta".
- **Regole**: Le regole esprimono relazioni condizionali, solitamente nella forma "se (condizione) allora (conclusione)". Queste regole consentono al sistema di inferire nuove conoscenze. Ad esempio:
  Se un paziente ha pressione alta e colesterolo elevato, allora è a rischio di malattie cardiache.

Nel contesto del progetto, il ragionamento basato su regole viene integrato per fornire una componente di analisi esperta che si affianca ai modelli statistici:

- **Definizione della Base di Conoscenza**: I file Prolog (ad esempio, heart_rules.pl) contengono una serie di regole che identificano specifici profili di rischio, come "pazienti ad alto rischio", "pazienti con ipertensione o colesterolo alto", e così via. Queste regole sono scritte secondo una logica condizionale e riflettono conoscenze mediche esplicite.
- **Esecuzione delle query**: Dal codice Python, tramite l'interfaccia pyswip, vengono eseguite query per interrogare il motore Prolog. Ad esempio, una query come alto_rischio(Età, Sesso) restituisce i pazienti che, in base alle regole, rientrano nel profilo di alto rischio.
  Questo permette di integrare il ragionamento simbolico (basato su regole) con i metodi statistici, fornendo una visione più completa e interpretabile dei dati.
- **Supporto al processo decisionale**: I risultati del ragionamento basato su regole possono essere visualizzati e utilizzati per indirizzare ulteriori analisi o per fornire spiegazioni interpretabili sulle predizioni, aspetto particolarmente importante in ambito sanitario.



Implementazione

Importazione delle librerie e configurazione iniziale

Il progetto inizia importando le librerie essenziali per la manipolazione dei dati, la visualizzazione, la modellazione e il ragionamento logico:

- **numpy e pandas:** per operazioni numeriche e manipolazione dei DataFrame.
- **matplotlib e seaborn:** per la creazione di grafici e visualizzazioni, utili nella fase esplorativa.
- **pyswip:** per interfacciarsi con Prolog, permettendo l'esecuzione di query basate su regole.
- **collections.deque:** utile per implementare la Breadth-First Search (BFS) in maniera efficiente.
- **scikit-learn:** per le funzioni di valutazione (metriche come accuracy, precision, recall, f1-score, ROC-AUC) e per la suddivisione dei dati in training e test.

Questa fase di setup garantisce che l'ambiente sia pronto per tutte le operazioni successive.

Caricamento ed esplorazione del dataset

Il dataset, contenuto nel file heart.csv, viene letto utilizzando pandas.read_csv(). Le prime operazioni effettuate sono:

- **Verifica della dimensione del dataset:** viene stampata la forma del DataFrame, che indica il numero di righe e colonne.
- **Elenco delle colonne e visualizzazione delle prime righe:** questo aiuta a comprendere quali dati sono disponibili (ad esempio, età, sesso, pressione arteriosa, colesterolo, ecc.).
- **Controllo dei valori mancanti:** viene verificato se ci sono colonne con valori nulli, garantendo l'integrità dei dati.
- **Analisi statistica:** viene eseguita una descrizione statistica (con describe().T) per conoscere le medie, deviazioni standard, valori minimi e massimi.

Queste operazioni iniziali servono a familiarizzare con il dataset e a individuare eventuali problemi da risolvere nella fase di pre-elaborazione.

Visualizzazione dei dati

Per supportare l'analisi esplorativa, il progetto genera diversi grafici:

- **Istogrammi:** Vengono creati istogrammi per ogni variabile del dataset. Questi grafici mostrano la distribuzione dei dati e aiutano a rilevare eventuali outlier o squilibri.
- **Grafico a barre (Countplot) della variabile target:** Un grafico a barre visualizza la distribuzione della variabile target (presenza o assenza di malattia cardiaca), fondamentale per verificare se le classi sono bilanciate.
- **Heatmap della matrice di correlazione:** Viene calcolata la correlazione tra tutte le variabili e visualizzata tramite una heatmap. Questo grafico evidenzia le relazioni tra i diversi attributi, indicando quali variabili potrebbero essere fortemente correlate e, pertanto, potenzialmente ridondanti o particolarmente influenti nel modello.

Pre-elaborazione dei dati

Prima della modellazione, il dataset viene trasformato in modo da essere adatto agli algoritmi di machine learning:

- **Codifica delle variabili categoriche:** Utilizzando pd.get_dummies(), le variabili categoriali (come 'sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal') vengono convertite in variabili dummy. Questo passaggio è cruciale perché molti algoritmi di machine learning richiedono input numerici.
- **Standardizzazione delle variabili numeriche:** Le colonne numeriche (ad esempio, 'age', 'trestbps', 'chol', 'thalach', 'oldpeak') vengono standardizzate con StandardScaler. La standardizzazione garantisce che tutte le variabili abbiano una scala simile (media 0 e deviazione standard 1), migliorando così la convergenza e la performance dei modelli.
- **Suddivisione in variabili indipendenti (X) e target (y):** Dopo la trasformazione, il dataset viene separato in X (tutte le colonne tranne il target) e y (la colonna target).
- **Train-Test Split:** I dati vengono divisi in set di training (80%) e test (20%) usando train_test_split(). Questo permette di addestrare i modelli sui dati di training e di valutarne le prestazioni su dati non visti (test set).

Modellazione con Machine Learning

Il cuore del progetto riguarda la costruzione e la valutazione di modelli di classificazione per prevedere la presenza di malattia cardiaca.

Decision Tree Classifier

- **Validazione Incrociata:** Per determinare la profondità ottimale dell'albero, il modello Decision Tree viene validato con una cross-validation a 10-fold, testando profondità da 1 a 10. I punteggi medi vengono raccolti e visualizzati graficamente per individuare il punto in cui le prestazioni si stabilizzano o iniziano a peggiorare (overfitting).
- **Addestramento:** Sulla base dei risultati ottenuti, viene addestrato un Decision Tree con max_depth=3 sul set di training.
- **Valutazione:** Dopo l'addestramento, il modello viene testato sul set di test e le sue prestazioni vengono misurate utilizzando una funzione dedicata (stampa_metriche), che calcola accuracy, precision, recall, F1-score, ROC-AUC e stampa la matrice di confusione.

Il modello Decision Tree è molto apprezzato per la sua interpretabilità, poiché ogni percorso dell'albero può essere tradotto in una regola decisionale facilmente comprensibile.

Random Forest Classifier

- **Ricerca del Numero Ottimale di Stimatori:** Analogamente al Decision Tree, viene effettuata una validazione incrociata a 5-fold per modelli Random Forest, variando il numero di stimatori (n_estimators) da 10 a 100, in incrementi di 10. Questo consente di identificare il numero di alberi che fornisce le migliori prestazioni.
- **Addestramento:** Il modello Random Forest viene quindi addestrato sul set di training con n_estimators=90.
- **Valutazione:** Le prestazioni del modello sono valutate sul set di test utilizzando le stesse metriche usate per il Decision Tree.

Il Random Forest migliora la robustezza del modello aggregando le predizioni di più alberi, riducendo il rischio di overfitting e fornendo una stima più stabile e generalizzata.

Integrazione con Prolog per l'Inferenza Logica

Una parte innovativa del progetto è l'integrazione con Prolog per sfruttare il ragionamento basato su regole:

- **Consultazione del file Prolog:** Il progetto utilizza il modulo pyswip per caricare il file heart_rules.pl, che contiene regole diagnostiche e la base di conoscenza (heart_kb.pl).
- **Esecuzione di Query:** Vengono eseguite diverse query per identificare pazienti in base a criteri specifici, ad esempio:\n - alto_rischio(Età, Sesso): identifica pazienti ad alto rischio\n - ipertensione_colesterolo(Età, Sesso): individua pazienti con ipertensione o colesterolo elevato\n - ecg_anomalo(Età, Sesso): evidenzia anomalie negli esami ECG\n - angina_sforzo(Età, Sesso) e profilo_alto_rischio(Età, Sesso): per ulteriori categorizzazioni\n\n- **Output delle Query:** I risultati delle query vengono raccolti in liste di dizionari e visualizzati in output. Una funzione di troncamento (tronca_risultati) viene utilizzata per limitare la visualizzazione a un massimo di 3 elementi per categoria, mantenendo l'output conciso e leggibile.

Questa parte del progetto integra il ragionamento simbolico, consentendo di formalizzare conoscenza medica ed eseguire inferenze logiche, che completano l'approccio basato su dati.

Algoritmo di Ricerca: Breadth-First Search (BFS)

Per identificare pazienti con profili clinici simili, il progetto implementa un algoritmo di **Breadth-First Search (BFS)**:

- **Motivazioni:** I concetti di ricerca in spazi di stati, trattati in dettaglio nella letteratura sull'Ingegneria della Conoscenza, forniscono la base teorica per esplorare relazioni nei dati. La BFS è scelta perché garantisce una ricerca sistematica a livello locale, trovando le soluzioni (o pazienti) più vicini in termini di somiglianza.\n\n- **Implementazione:**\n - Si parte da un paziente iniziale (definito da età e sesso).\n - La ricerca si espande in ampiezza fino a raggiungere una profondità massima (default 3 livelli).\n - Viene mantenuto un insieme di pazienti già visitati per evitare duplicazioni e cicli.\n - I pazienti trovati vengono restituiti, e se il numero supera 3, viene aggiunto un segnaposto per indicare che ci sono ulteriori risultati non visualizzati.\n\nQuesto algoritmo permette di individuare gruppi di pazienti simili, utile per approfondire l'analisi e per individuare pattern o anomalie comuni in gruppi di pazienti.

Installazione e Configurazione del Progetto

Per eseguire correttamente il progetto "Heart Disease Prediction" presente nella repository GitHub, segui i passaggi seguenti per installare tutte le dipendenze e configurare l'ambiente di sviluppo.

Requisiti di sistema

Assicurati di avere installato:

- Python 3.7;
- Git;
- pip (gestore di pacchetti Python);
- Link della repository: https://github.com/PrynceMoon/Heart-Disease-Prediction

Clonazione della repository

Apri un terminale o prompt dei comandi ed esegui:

```
git clone https://github.com/PrynceMoon/Heart-Disease-Prediction
cd Heart-Disease-Prediction
```

“git clone” è il comando che permette di clonare la repository all’interno del proprio computer

Installazione delle Dipendenze

Installa tutte le librerie necessarie eseguendo:

```
pip install numpy pandas matplotlib seaborn scikit-learn pyswip
```

Correzione degli errori

- **Errore ModuleNotFoundError**: Verifica di aver attivato l'ambiente virtuale e che tutte le librerie siano installate.
- **Errore con pyswip**: Assicurati di avere SWI-Prolog installato sul sistema (Scarica SWI-Prolog). Dopo l'installazione, aggiungi il percorso di SWI-Prolog alle variabili d'ambiente se necessario.
- **Errore di permessi**: Su Linux/macOS, potresti dover eseguire chmod +x script.sh (se ci sono script di esecuzione) e usare python3 invece di python.

Conclusioni

Il progetto **Heart Disease Prediction** rappresenta un esempio di come sia possibile integrare in modo efficace metodologie di **machine learning**, **ricerca euristica** e **ragionamento basato su regole** per affrontare problemi complessi come la diagnosi delle malattie cardiache. L'approccio ibrido consente non solo di ottenere previsioni accurate, ma anche di interpretare le decisioni grazie alla presenza di regole logiche formalizzate tramite Prolog.

L'integrazione di questi diversi paradigmi permette di sfruttare i punti di forza di ciascuna tecnica: la capacità predittiva dei modelli ML, la sistematicità degli algoritmi di ricerca e la chiarezza interpretativa dei sistemi basati su regole. In tal modo, il progetto si propone come un sistema di supporto alle decisioni cliniche, in grado di offrire una diagnosi assistita robusta e interpretabile.