Predizione presenza malattia cardiache

Gruppo di lavoro

- Angelo Prete, 774281, a.prete20@studenti.uniba.it

Link GitHub: https://github.com/AngeloPrete/HeartDiseasePrediction.git

AA 2024-25

Introduzione

Secondo l'Istituto Superiore di Sanità, le malattie cardiache rappresentano la prima causa di morte nel mondo occidentale, e nel nostro paese. Una percentuale significativa degli italiani ha almeno tre fattori di rischio, nonostante sia possibile evitare l'80% dei decessi dovuti a queste patologie con la prevenzione.

Questo progetto mira a costruire un sistema di classificazione binario adatto a predire, in base a delle feature di input, se una determinata persona possa essere affetta da una malattia cardiaca. Il sistema, sviluppato in Python, utilizza un dataset sulle malattie cardiache, preso da Kaggle.

Le feature presenti nel dataset sono:

Nome	Descrizione
Age	Età del paziente.
Sex	Sesso del paziente (M = maschio, F = femmina).
ChestPainType	Tipologia di dolore toracico (ASY = asintomatico, ATA = angina atipica, NAP = no angina pain, TA = angina tipica).
RestingBP	Pressione sanguigna a riposo (misurata in mmHg).
Cholesterol	Livello di colesterolo totale nel sangue (misurato in mg/dl).
FastingBS	Glicemia a digiuno superiore a 120 mg/dl (1 = vero, 0 = falso).
RestingECG	Risultato dell'elettrocardiogramma a riposo (Normal = normale, ST = anomalia dell'onda ST-T, LVH = ipertrofia ventricolare sinistra).
MaxHR	Frequenza cardiaca massima raggiunta durante lo sforzo.
ExerciseAngina	Presenza di angina indotta da esercizio fisico (N = no, Y = sì).
Oldpeak	Depressione del tratto ST indotta dall'esercizio rispetto al riposo (misurata in mm).
ST_Slope	Pendenza del segmento ST durante l'esercizio (DOWN = in discesa, FLAT = piatto, UP = in pendenza).
HeartDisease	Presenza di malattia cardiaca (1 = vero, 0 = falso).

Sommario

Per raggiungere l'obiettivo preposto, ho progettato e analizzato diversi modelli di apprendimento supervisionato, tra cui Decision Tree, Random Forest e Regressione Logistica.
Per ottimizzare gli iperparametri, è stata utilizzata la tecnica di cross-validation, dopodiché i modelli sono stati confrontati in base all'accuratezza. Inoltre, per permettere il ragionamento probabilistico, si è scelto di discretizzare le feature continue, ingegnerizzandone nuove, utilizzando regole definite nella KB. Questo ha facilitato l'apprendimento della struttura della belief network impiegata per il ragionamento probabilistico.

Elenco argomenti di interesse

- **Apprendimento supervisionato**: Sono stati utilizzati modelli come Decision Tree, Random Foreste e Regressione Logistica per risolvere un task di classificazione binaria.
- **Rappresentazione e ragionamento relazionale:** È stato utilizzato Prolog per popolare la Knowledge Base (KB) con i fatti provenienti dal dataset e per creare regole che hanno permesso di ingegnerizzare nuove feature.
- **Apprendimento e Incertezza**: È stato applicato l'algoritmo K-Means per raggruppare i pazienti in dei cluster sulla base delle loro caratteristiche. Successivamente, è stata aggiunta una nuova feature, "Cluster", al dataset per indicare l'appartenenza di ciascun paziente a uno specifico gruppo. Dopodiché, sono stati riaddestrati i modelli supervisionati con lo scopo di verificare se esistesse una correlazione tra i cluster individuati e la presenza di malattie cardiache che potesse migliorare le performance dei modelli.
- **Ragionamento su Modelli di Conoscenza Incerta**: È stata utilizzata una belief network per effettuare ragionamento probabilistico sui dati.

APPRENDIMENTO SUPERVISIONATO

Sommario

L'apprendimento supervisionato è stato utilizzato per costruire modelli capaci di eseguire una classificazione binaria. In particolare, i modelli sono stati addestrati su un training set contenente sia le feature di input che la feature target. Nel mio caso, la feature target è rappresentata dalla label "Heart Disease", che può assumere due valori:

- 0 = il paziente non è affetto da malattie cardiache
- 1 = il paziente è affetto da malattie cardiache

L'obiettivo principale è quello di apprendere una funzione di predizione in grado di fare previsioni su dati mai visti prima, ossia, i modelli devono essere in grado di predire se un paziente sia affetto o meno da malattia cardiaca. Dopo aver addestrato i vari modelli, questi sono stati confrontati utilizzando la metrica di accuratezza.

Decisioni di Progetto

Per l'apprendimento automatico è stata utilizzata la libreria scikit-learn, mentre per la visualizzazione di grafici è stata utilizzata la libreria matplotlib.

Pre-processing:
Poiché i modelli basati sugli alberi di decisione e la Regressione Logistica implementati nella libreria scikit-learn non supportano direttamente le feature categoriche, ho trasformato queste feature in numeriche utilizzando la tecnica del One-Hot Encoding. Questa tecnica crea una nuova variabile binaria per ciascun valore distinto di una feature categorica, consentendo ai modelli di elaborarle correttamente.

Nel caso della Regressione Logistica, il One-Hot Encoding è particolarmente utile perché evita di introdurre un ordine implicito tra le categorie delle feature categoriche, permettendo al modello di trattarle come variabili indipendenti.

Inoltre, ho deciso di normalizzare i dati utilizzando il MinMaxScaler, che trasforma tutte le variabili sulla stessa scala, compresa tra 0 e 1. Questo è fondamentale per evitare che alcune feature, che potrebbero avere valori numerici più elevati, influenzino in modo eccessivo il modello di Regressione Logistica.

Ho deciso di utilizzare i seguenti modelli di apprendimento supervisionato:

- **Decision Tree Classifier**: è un classificatore che utilizza un albero decisionale, in cui i nodi interni rappresentano condizioni (test booleani) basate sui valori delle feature degli esempi nel training set. Le foglie dell'albero, invece, indicano le classi di appartenenza.
- **Random Forest Classifier**: è un classificatore basato su un insieme di alberi decisionali (foresta), in cui ogni albero è costruito utilizzando una porzione di dati e un sottoinsieme casuale delle feature. La predizione finale è determinata dalla classe votata della maggior parte degli alberi (per classificazione).
- **Regressione Logistica**: è un modello di apprendimento supervisionato utilizzato principalmente per problemi di classificazione binaria. La regressione logistica stima la probabilità che un'osservazione appartenga a una delle due classi, utilizzando una funzione logistica (sigmoide) per mappare una combinazione lineare delle feature di input in un valore compreso tra 0 e 1. Se la probabilità stimata è maggiore di una soglia (tipicamente 0.5),

l'osservazione viene classificata in una classe; altrimenti, viene classificata nell'altra. Il modello cerca di trovare i pesi che minimizzano l'errore di previsione, solitamente tramite il metodo della massima verosimiglianza.

Prima di addestrare i modelli, è necessario selezionare gli iperparametri, ossia parametri che non vengono appresi durante l'addestramento, ma devono essere definiti a priori. Tuttavia, scegliere gli iperparametri senza aver esaminato i dati può risultare complesso. Per questo motivo, ho utilizzato la tecnica di GridSearch con K-Fold Cross Validation per ottimizzare la selezione degli iperparametri. Il K-Fold Cross Validation consiste nel suddividere i dati in k sottoinsiemi (fold), in cui k-1 fold vengono utilizzati per l'addestramento, mentre il fold rimanente è impiegato per la validazione. Questo processo viene ripetuto variando il fold utilizzato per la validazione, garantendo che ogni fold venga usato una volta per la validazione e k-1 volte per l'addestramento.

Questo approccio consente di valutare le performance del modello su diverse porzioni del dataset, riducendo l'influenza di un singolo set di addestramento. Inoltre, aiuta a prevenire l'overfitting e a selezionare gli iperparametri che ottimizzano la generalizzazione del modello sui dati.

Per la ricerca degli iperparametri, come già accennato, ho utilizzato la tecnica di GridSearch integrata con Cross Validation (k = 5). GridSearch consente di definire una griglia di valori per gli iperparametri e per ogni combinazione di iperparametri esegue la cross-validation. GridSearch restituisce il miglior modello in base alla metrica di valutazione che ho impostato, ovvero l'accuratezza. Successivamente il miglior modello viene addestrato e valutato su tutto il dataset applicando la tecnica di k-fold cross validation con k = 5.
Di seguito si riporta il codice per l'applicazione di GridSearch:

```
def getBest_model_and_hyperparameters(estimator,param_grid,X_train,y_train):
    # Applichiamo GridSearchCV con 5-fold cross-validation
    grid_search = GridSearchCV(estimator, param_grid, cv=5, n_jobs=-1, scoring='accuracy')

    #Addestriamo il modello
    grid_search.fit(X_train, y_train)

    # Estraiamo il miglior modello da GridSearchCV
    best_model = grid_search.best_estimator_

    return best_model,grid_search.best_params_
```

Il codice per addestrare e valutare un modello:

```
def addestra(X_train, X_test, y_train, y_test,X,y,model):

    param = getGrid_params(model)

    best_model, best_hyperparameters = getBest_model_and_hyperparameters(model,param,X_train,y_train)

    print(f"Migliori parametri del modello {best_hyperparameters}")

    # Definisco la strategia di K-Fold Cross Validation (K=5)
    kf = KFold(n_splits=5, shuffle=True, random_state=42)

    # Eseguo la K-Fold Cross Validation sull'intero dataset
    scores_accuratezza = cross_val_score(best_model, X, y, cv=kf, scoring='accuracy')
    # Accuratezza media
    print(f"Accuratezza media : {scores_accuratezza.mean():.4f} ± {scores_accuratezza.std():.4f}")

    plot_curva_apprendimento(best_model,X,y)
```

Valutazione

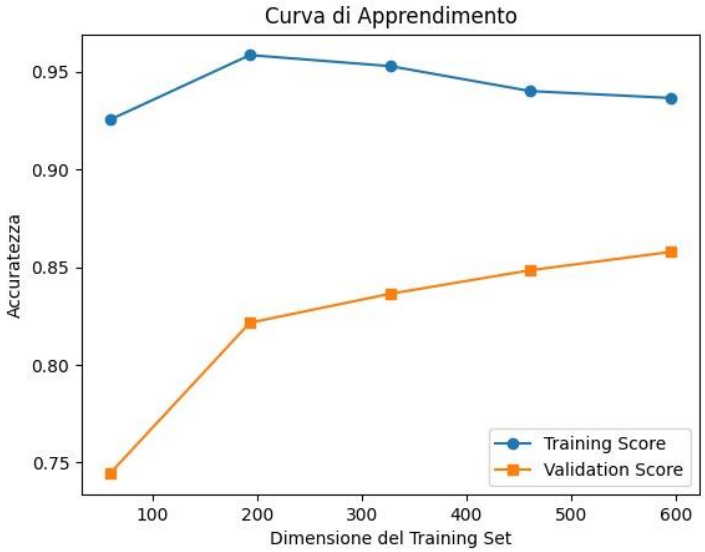
Come già accennato, per ogni tipo di classificatore, il miglior modello è stato selezionato ottimizzando l'accuratezza. In particolare, per ottimizzare gli iperparametri, ho utilizzato GridSearch con cross-validation, applicato esclusivamente sui dati di training. Una volta identificato il miglior modello, quest'ultimo è stato addestrato e valutato sull'intero dataset utilizzando la k-fold cross-validation con k=5.

Come è possibile vedere dalla seguente figura, la distribuzione delle classi della feature Heart Disease è quasi equa, quindi il dataset non è sbilanciato:

Per ogni classificatore di seguito è riportata l'accuratezza media con deviazione standard e le curve di apprendimento.

Random Forest Classifier:

L'accuratezza media del Random Forest è di 0.8726 con una deviazione standard di 0.0214

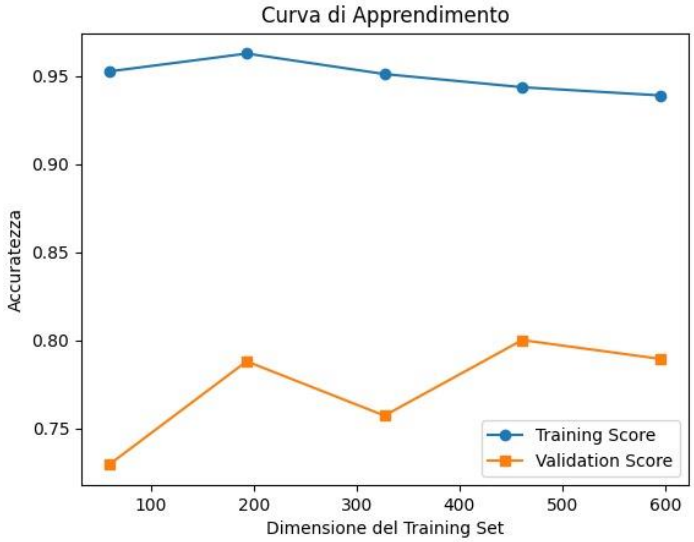


I miglior parametri restituiti dal GridSearch sono:

Iperparametro	Valore	Descrizione
Criterion	entropy	L'entropia misura il livello di disordine o incertezza in un dato set di dati. L'algoritmo cerca di massimizzare l'informazione guadagnata (cioè ridurre l'entropia) ad ogni passo, suddividendo i dati in modo che le osservazioni in ciascun sottoinsieme siano il più omogenee possibile.
Max_depth	10	Limita la profondità massima degli alberi. Un valore di 10 significa che ogni albero avrà al massimo 10 livelli.
Min_samples_leaf	1	Definisce il numero minimo di campioni che una foglia deve contenere. Un valore di 1 significa che ogni foglia dovrà avere almeno 1 campioni.
Min_samples_split	10	Definisce il numero minimo di campioni necessari per dividere un nodo in due. Un valore di 10 significa che un nodo non verrà diviso se non contiene almeno 10 campioni, evitando divisioni troppo specifiche su pochi dati.
n_estimators	200	Definisce il numero di alberi nella foresta. Un valore di 200 significa che il modello utilizza 200 alberi decisionali.

Tree Decision Classifier:

L'accuratezza media del Decision Tree è di 0.8270 con deviazione standard di 0.03



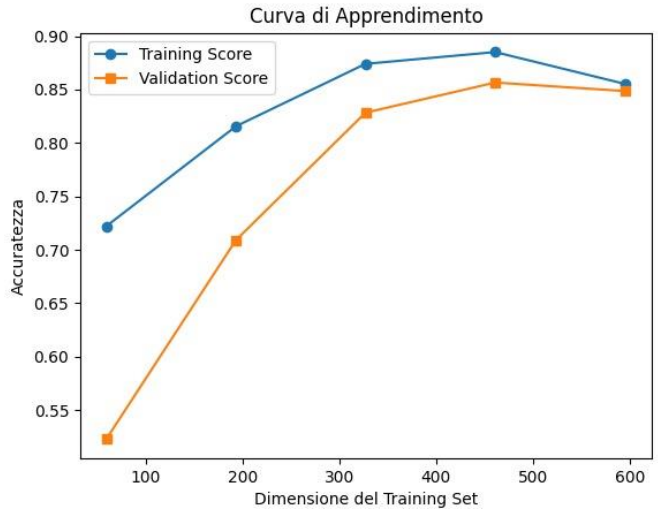
I miglior parametri restituiti dal GridSearch sono:

Iperparametro	Valore	Descrizione
Criterion	entropy	L'entropia misura il livello di disordine o incertezza in un dato set di dati. L'algoritmo cerca di massimizzare l'informazione guadagnata (cioè ridurre l'entropia) ad ogni passo, suddividendo i dati in modo che le osservazioni in ciascun sottoinsieme siano il più omogenee possibile.

Max_depth	10	Limita la profondità massima degli alberi.
Min_samples_leaf	2	Definisce il numero minimo di campioni che una foglia deve contenere. Un valore di 2 significa che ogni foglia dovrà avere almeno 2 campioni.
Min_samples_split	10	Definisce il numero minimo di campioni necessari per dividere un nodo in due. Un valore di 10 significa che un nodo non verrà diviso se non contiene almeno 10 campioni, evitando divisioni troppo specifiche su pochi dati.

Logistic Regression Classifier:

L'accuratezza media della regressione Logistica è di 0.8512 con deviazione standard 0.0183



I miglior parametri restituiti dal GridSearch sono:

Iperparametro	Valore	Descrizione
C	0.001	Il parametro di regolarizzazione. Un valore basso (come 0.001) implica una penalizzazione forte, che riduce la complessità del modello, favorendo una maggiore generalizzazione ma potenzialmente aumentando il bias.
Penalty	L2	Tipo di regolarizzazione applicata al modello. L'L2 penalizza i grandi coefficienti riducendo la loro grandezza per evitare l'overfitting, ma senza azzerare completamente i pesi.
Solver	liblinear	Algoritmo utilizzato per apprendere i pesi.

Conclusioni

Dalle metriche calcolate, si osserva che tutti i modelli ottengono ottime performance nella previsione della feature HeartDisease. Tuttavia, analizzando le curve di apprendimento, emergono differenze significative nel loro comportamento. Sia il Random Forest Classifier che il Decision Tree Classifier mostrano una tendenza a memorizzare eccessivamente le osservazioni del training set. Questo è evidente dal fatto che l'accuratezza sul training set supera costantemente il 90%. In particolare, per il Decision Tree, l'accuratezza sul validation set è significativamente inferiore rispetto a quella sul training set, suggerendo un chiaro caso di overfitting. Al contrario, la Regressione Logistica mostra curve di apprendimento più equilibrate tra training set e validation set, indicando una migliore

capacità di generalizzazione senza adattarsi eccessivamente ai dati di training. Un ulteriore aspetto da considerare è la deviazione standard delle performance. I modelli Decision Tree e Random Forest mostrano una varianza maggiore rispetto alla Regressione Logistica, il che indica una maggiore sensibilità alle variazioni nei dati di training. In altre parole, piccoli cambiamenti nei dati possono influenzare significativamente le loro prestazioni, migliorandole o peggiorandole. Questo rafforza l'ipotesi che Decision Tree e Random Forest possano soffrire di overfitting. La loro elevata sensibilità alle variazioni nei dati di training suggerisce che questi modelli si adattano troppo ai dati di partenza, perdendo capacità predittiva su dati non visti. In conclusione, il modello di Regressione Logistica risulta più robusto rispetto ai modelli basati su alberi decisionali, poiché offre prestazioni più stabili e una migliore generalizzazione.

Apprendimento e Incertezza

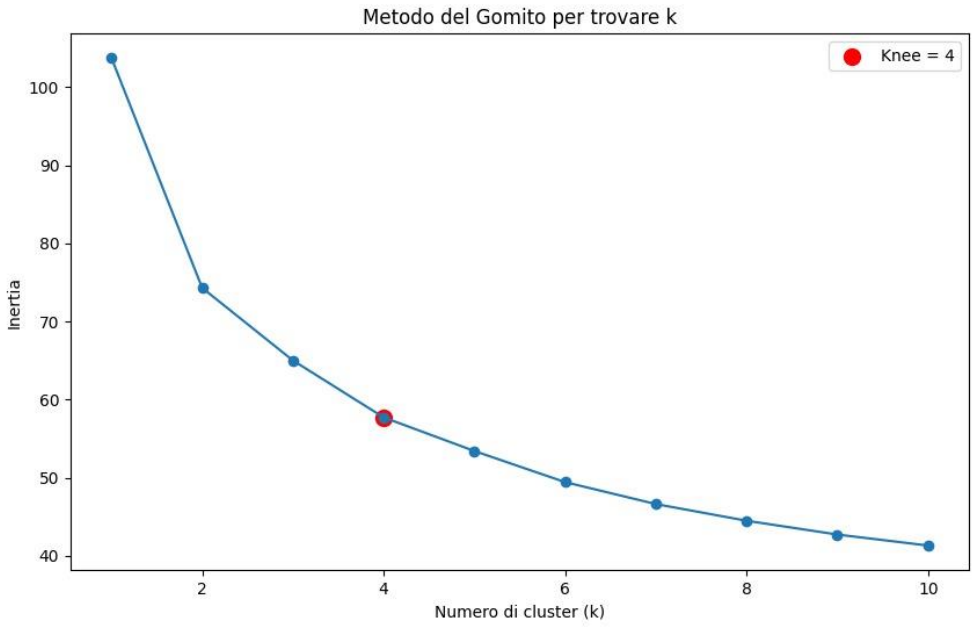
Sommario

Come modello di apprendimento non supervisionato è stato utilizzato il kmeans in quale ha permesso di effettuare hard clustering. In particolare, il kmeans è stato utilizzato per raggruppare i dati in base a delle caratteristiche in modo tale da ingegnerizzare una nuova feature 'Cluster' che permettesse di arricchire il dataset con lo scopo di provare a migliorare le performance dei modelli di apprendimento supervisionato.

Decisioni di progetto

Il K-Means è stato eseguito esclusivamente sulle variabili numeriche continue, le quali sono state preventivamente normalizzate mediante MinMaxScaler per garantire che tutte le variabili avessero la stessa importanza. Per determinare il numero ottimale di cluster con cui raggruppare i dati, è stata applicata la regola del gomito o del ginocchio. Questo metodo prevede l'esecuzione dell'algoritmo K-Means per diversi valori di K, calcolando per ciascuno l'inertia, ovvero la somma delle distanze quadratiche all'interno dei cluster. L'inertia misura quanto i punti all'interno di ogni cluster siano vicini al rispettivo centroide. Il numero ottimale di cluster corrisponde al punto in cui la riduzione dell'inertia inizia a rallentare significativamente, formando una curva a gomito nel grafico. Per trovare il numero ideale di cluster, ho deciso di eseguire l'algoritmo per valori di k da 1 a 11.

Di seguito viene riportata la curva del gomito:



Come si può vedere dal grafico il numero ideale di cluster è 4.

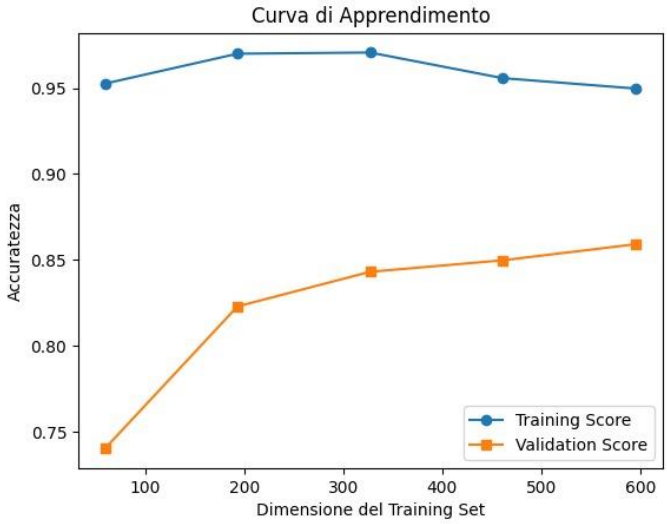
A questo punto eseguo il kmeans con k = 4 e arricchisco il dataset originale attraverso la feature 'Cluster', la quale indica a quale cluster ogni paziente è stato associato. Successivamente ho riaddestrato i modelli di apprendimento supervisionato per verificare se le performance miglioravano.

Valutazione

Di seguito vengono riportati i grafici relativi alle performance dei modelli di apprendimento supervisionato dopo l'aggiunta della feature 'Cluster':

Random Forest Classifier:

Il Random Forest ha un'accuratezza media di 0.8793 con deviazione standard di 0.0328

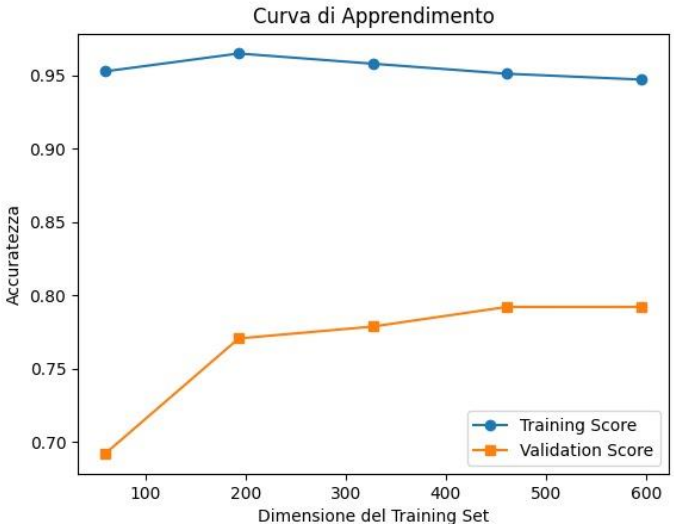


I miglior parametri restituiti dal GridSearch sono:

Iperparametro	Valore	Descrizione
Criterion	entropy	L'entropia misura il livello di disordine o incertezza in un dato set di dati. L'algoritmo cerca di massimizzare l'informazione guadagnata (cioè ridurre l'entropia) ad ogni passo, suddividendo i dati in modo che le osservazioni in ciascun sottoinsieme siano il più omogenee possibile.
Max_depth	10	Limita la profondità massima degli alberi. Un valore di 10 significa che ogni albero avrà al massimo 10 livelli.
Min_samples_leaf	2	Definisce il numero minimo di campioni che una foglia deve contenere. Un valore di 2 significa che ogni foglia dovrà avere almeno 2 campioni.
Min_samples_split	5	Definisce il numero minimo di campioni necessari per dividere un nodo in due. Un valore di 5 significa che un nodo non verrà diviso se non contiene almeno 5 campioni, evitando divisioni troppo specifiche su pochi dati.
n_estimators	100	Definisce il numero di alberi nella foresta. Un valore di 100 significa che il modello utilizza 100 alberi decisionali.

Decision Tree Classifier:

Il Decision Tree ha un'accuratezza media di 0.8230 con deviazione standard 0.0415

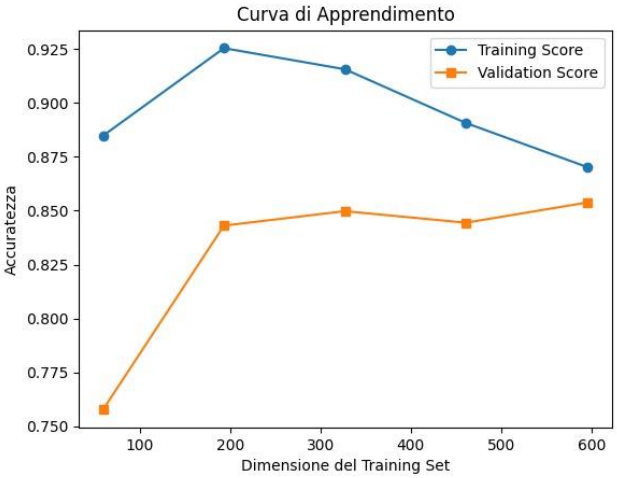


I miglior parametri restituiti dal GridSearch sono:

Iperparametro	**Valore**	**Descrizione**
Criterion	entropy	L'entropia misura il livello di disordine o incertezza in un dato set di dati. L'algoritmo cerca di massimizzare l'informazione guadagnata (cioè ridurre l'entropia) ad ogni passo, suddividendo i dati in modo che le osservazioni in ciascun sottoinsieme siano il più omogenee possibile.
Max_depth	None	Limita la profondità massima degli alberi.
Min_samples_leaf	1	Definisce il numero minimo di campioni che una foglia deve contenere. Un valore di 1 significa che ogni foglia dovrà avere almeno 1 campioni.
Min_samples_split	10	Definisce il numero minimo di campioni necessari per dividere un nodo in due. Un valore di 10 significa che un nodo non verrà diviso se non contiene almeno 10 campioni, evitando divisioni troppo specifiche su pochi dati.

Logistic Regression Classifier:

Il modello di regressione logistica ha un'accuratezza media di 0.8619 con deviazione standard 0.0240



I miglior parametri restituiti dal GridSearch sono:

Iperparametro	Valore	Descrizione
C	0.1	Il parametro di regolarizzazione.
Penalty	L2	Tipo di regolarizzazione applicata al modello. L'L2 penalizza i grandi coefficienti riducendo la loro grandezza per evitare l'overfitting, ma senza azzerare completamente i pesi.
Solver	saga	Algoritmo utilizzato per apprendere i pesi.

Come possibile verificare dalle metriche calcolate e dai grafici, possiamo notare come le performance dei modelli di apprendimento supervisionato tendono a rimanere pressoché le stesse con un leggero aumento della varianza. Quindi l'introduzione della feature Cluster non ha avuto un impatto positivo al fine del miglioramento dei modelli.

Rappresentazione e ragionamento relazionale

Sommario

Una Knowledge Base contiene assiomi, ovvero proposizioni assunte come vere. Gli assiomi possono essere fatti, cioè affermazioni prese per vere all'interno della base di conoscenza, oppure regole, che sono clausole definite in cui la testa della clausola è vera se lo sono tutti gli atomi nel corpo della clausola. Una Knowledge Base permette, quindi, di rappresentare un dominio di interesse mediante una semantica ben definita e di eseguire inferenze utilizzando la logica dei predicati. Nel mio caso, la Knowledge Base è stata utilizzata per inferire nuove feature da aggiungere al dataset, al fine di costruire successivamente una rete bayesiana. In particolare, a partire dalle feature con valori continui presenti nel dataset, “Cholesterol”, “Oldpeak” e “RestingBP”, ho applicato delle regole per inferire nuove feature che categorizano i valori di queste variabili. Ad esempio, ho determinato se un paziente ha un livello di colesterolo alto, moderato o basso, e ho creato una feature “Cholesterol_categoria” che contenesse queste nuove informazioni. Ho seguito un approccio analogo anche per le altre due feature indicate.

Decisioni di Progetto

Per popolare la knowledge base ed effettuare ragionamento relazionale ho usato la libreria pyswip per il linguaggio Prolog.

La knowledge base è popolata solo da una classe di individui ovvero i pazienti, i quali sono identificati da un indice che corrisponde al numero di riga in cui si trovano nel dataset:

- paziente(id): vero se id è un paziente.

Per ogni paziente ho inserito le seguenti proprietà:

- colesterolo(id,C)
- eta(id,A)
- pressione_sanguigna(id,B)
- sesso(id,S)
- chest_pain(id,Cp)
- glicemia_alta(id,F)
- result_ecg(id,E)
- fc_max(id,Max)
- exang(id,Ang)
- oldpeak(id,M)
- slope(id,S)
- presenza_malattia(id,T)

Le regole, utilizzate per inferire nuove feature, sono:

```
("colesterolo_alto(ID) :- colesterolo(ID, Col), Col > 250")
("colesterolo_moderato(ID) :- colesterolo(ID, Col), Col >= 200, Col =< 250")
("colesterolo_basso(ID) :- colesterolo(ID, Col), Col < 200")

("pressione_sanguigna_alta(ID) :- pressione_sanguigna(ID, Bps), Bps > 140")
("pressione_sanguigna_moderata(ID) :- pressione_sanguigna(ID, Bps), Bps >= 120, Bps =< 140")
("pressione_sanguigna_bassa(ID) :- pressione_sanguigna(ID, Bps), Bps < 120")

("oldpeak_alto(ID) :- oldpeak(ID, Millimetri), Millimetri > 2")
("oldpeak_moderato(ID) :- oldpeak(ID, Millimetri), Millimetri >= 1, Millimetri =< 2")
("oldpeak_basso(ID) :- oldpeak(ID, Millimetri), Millimetri < 1")
```

Per aggiungere i valori delle nuove feature al dataset, si calcolano questi valori per ogni esempio del dataset eseguendo delle query sulla knowledge base.

Ad esempio, per categorizzare i valori della variabile “Resting_BP” si esegue la seguente funzione:

```
def get_categoria_pressione_sanguigna(id,prolog):

    result = list(prolog.query(f"pressione_sanguigna_bassa({id})"))

    if(result):
        return 'Normal'
    else :
        result = list(prolog.query(f"pressione_sanguigna_moderata({id})"))
        if(result):
            return 'Moderate'
        else :
            return 'High'
```

Il valore restituito dalla funzione viene assegnato alla nuova feature “RestingBP_categoria”. Questo processo vene ripetuto per tutte le nuove feature di seguito indicate e per ogni esempio nel dataset.

Le nuove feature ingegnerizzate sono:

Nome	Descrizione
RestingBP_categoria	Indica il livello della pressione sanguigna (Normal,Moderate,High)
Cholesterol_categoria	Indica il livello di colesterolo (Normal,Moderate,High)
Oldpeak_categoria	Categorizza la depressione del tratto ST (Normal,Moderate,High)

Ragionamento su Modelli di Conoscenza Incerta

Sommario

Per effettuare ragionamento probabilistico è possibile utilizzare una rete bayesiana, rappresentata come un grafo aciclico diretto (DAG) in cui ogni nodo corrisponde a una variabile casuale. Un arco da una variabile X a una variabile Y indica che Y è condizionatamente dipendente da X, ovvero la probabilità di Y dipende dal valore assunto da X. In particolare, il ragionamento probabilistico in una rete bayesiana permette di calcolare la probabilità a posteriori di una variabile data una evidenza. In termini matematici, si calcola la probabilità condizionata di una variabile Y dato che si conoscono alcune evidenze (ovvero altre variabili nel modello). Questo si esprime tramite il teorema di Bayes.

Decisioni di Progetto

Per effettuare ragionamento probabilistico e apprendere la struttura bayesiana dai dati sono state utilizzate le librerie pgmpy e networkx.

Prima di effettuare ragionamento probabilistico è necessario apprendere la struttura della rete bayesiana dai dati.

Pre-processing:

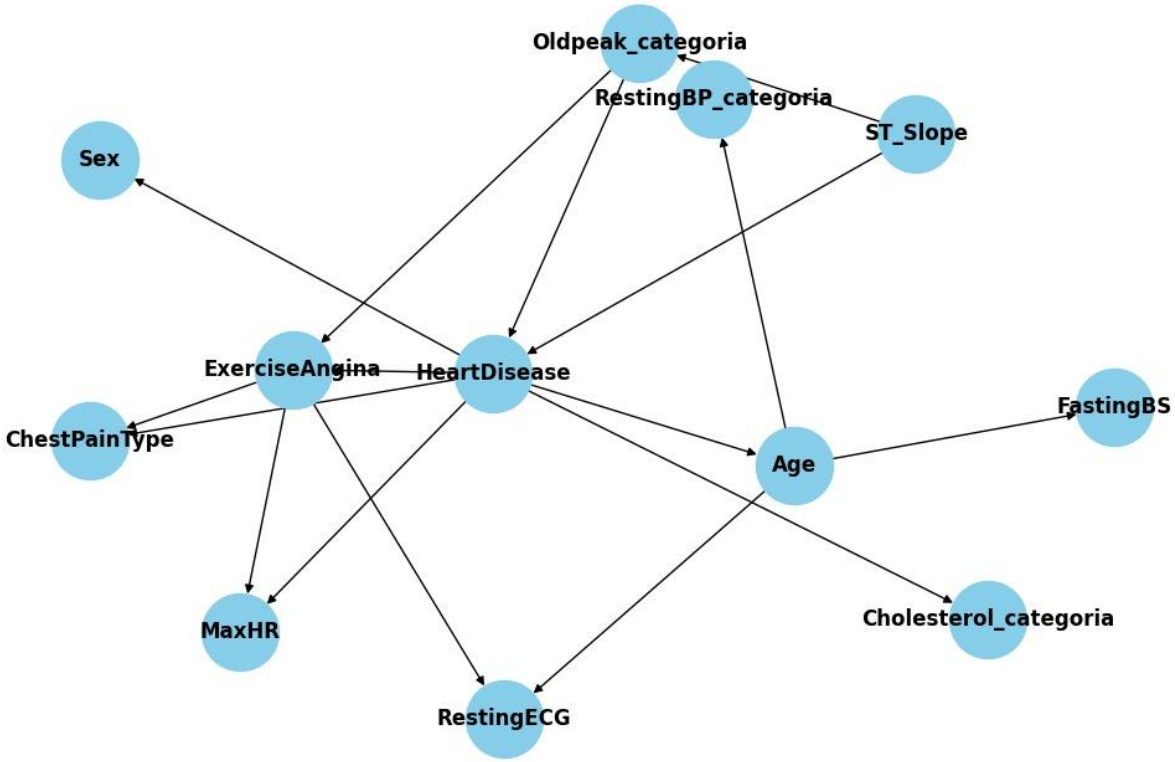
Per apprendere la struttura della rete bayesiana dal dataset, ho selezionato solo le feature discrete. Questa scelta è stata dettata dal fatto che l'apprendimento di una rete bayesiana con feature continue comporta un elevato carico computazionale. Infatti, per generare le CPT (tabelle di probabilità condizionate), sarebbe necessario esplorare tutte le possibili combinazioni di valori delle variabili continue. Poiché le variabili continue possono assumere un numero infinito di valori, questo approccio diventa impraticabile. Le feature continue originali, come RestingBP, Cholesterol e Oldpeak sono state quindi sostituite dalle feature discrete ovvero da RestingBP_categroia, Cholesterol_categoria e oldpeak_categoria, ottenute tramite tecniche di ingegnerizzazione attraverso ragionamento relazionale. Invece le feature Age e MaxHR sono state discretizzate rispettivamente attraverso la definizione di intervalli di fasce d'età e KBinsDescritizer con valore di k = 10.

```
bins = [18, 35, 50, 60, 150]
feature['Age'] = pd.cut(feature['Age'],bins=bins,labels=["Giovane","Adulto","50-60","Over 60"], right=False)
```

Per apprendere la struttura delle rete bayesiana dal dataset è stato utilizzato un algoritmo di ricerca greedy locale HillClimbSearch il cui obiettivo è massimizzare la likelihood.

Valutazione

Di seguito è riportato la rete bayesiana appresa:



Di seguito si riportano alcune CPT per delle variabili:

CPT per la variabile ExerciseAngina:

HeartDisease	HeartDisease(0)	...	HeartDisease(1)	HeartDisease(1)
Oldpeak_categoria	Oldpeak_categoria(High)	...	Oldpeak_categoria(Moderate)	Oldpeak_categoria(Normal)
ExerciseAngina(N)	0.75	...	0.2033898305084746	0.6310679611650486
ExerciseAngina(Y)	0.25	...	0.7966101694915254	0.36893203883495146

CPT per la variabile Heart Disease:

Oldpeak_categoria	Oldpeak_categoria(High)	...	Oldpeak_categoria(Normal)	Oldpeak_categoria(Normal)
ST_Slope	ST_Slope(Down)	...	ST_Slope(Flat)	ST_Slope(Up)
HeartDisease(0)	0.16	...	0.2857142857142857	0.9075907590759076
HeartDisease(1)	0.84	...	0.7142857142857143	0.0924092409240924

CPT per la variabile Cholesterol_categoria:

HeartDisease	HeartDisease(0)	HeartDisease(1)
Cholesterol_categoria(High)	0.35384615384615387	0.4691011235955056
Cholesterol_categoria(Moderate)	0.41794871794871796	0.3707865168539326
Cholesterol_categoria(Normal)	0.2282051282051282	0.1601123595505618

Dopo aver calcolato tutte le CPT è possibile effettuare delle query per effettuare ragionamento probabilistico. Ad esempio, si sono eseguite le seguenti query:

"*Qual è la probabilità di avere una malattia cardiaca sapendo di avere un Oldpeak alto, la pendenza del tratto ST verso il basso e un livello di colesterolo alto?*"

Questa query può essere scritta come:

P(HeartDisease = 1 | Oldpeak_categoria = 'High',ST_Slope = 'Down', Cholesterol_categoria = 'High')

Le probabilità calcolate sono:

HeartDisease	phi(HeartDisease)
HeartDisease(0)	0.1256
HeartDisease(1)	0.8744

È importante notare che tutte le variabili utilizzate come evidenza influenzano la probabilità di HeartDisease, poiché fanno parte della sua Markov Blanket. Questo insieme comprende i genitori di HeartDisease, i suoi figli e gli altri genitori di questi ultimi. Pertanto, una volta note le variabili nella Markov Blanket, esse rendono HeartDisease indipendente dal resto della rete.

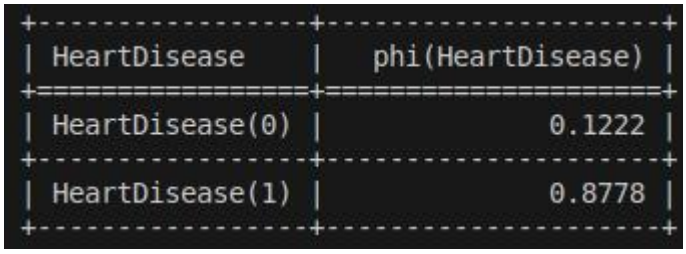
Un'altra query eseguita è:

“Qual è la probabilità di avere una malattia cardiaca sapendo *di avere un Oldpeak alto, la pendenza del tratto ST verso il basso, un livello di colesterolo alto e avere un età compresa tra i 50 e i 60?”*

Questa query corrisponde a:

P(HeartDisease = 1 | Oldpeak_categoria = ‘High’,ST_Slope = ‘Down’, Cholesterol_categoria = ‘High, Age = ’50-60’)

La distribuzione di probabilità è la seguente:



HeartDisease	phi(HeartDisease)
HeartDisease(0)	0.1222
HeartDisease(1)	0.8778

La probabilità di avere una malattia cardiaca è del 87,78 %

Se proviamo a modificare le evidenze con parametri teoricamente migliori per la stessa fascia d’età dovremmo ottenere una probabilità inferiore di avere una malattia cardiaca.

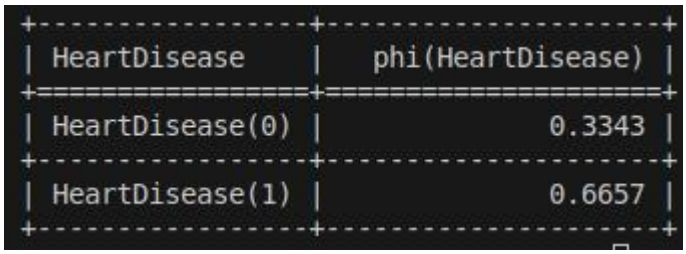
Eseguiamo una query migliorando i parametri nelle evidenze:

“Qual è la probabilità di avere una malattia cardiaca sapendo *di avere un Oldpeak moderato, la pendenza del tratto ST verso il basso, un livello di colesterolo normale e avere un età compresa tra i 50 e i 60?”*

Questa query corrisponde a:

P(HeartDisease = 1 | Oldpeak_categoria = ‘Moderate’ ,ST_Slope = ‘Down’, Cholesterol_categoria = ‘Normal’, Age = ’50-60’)

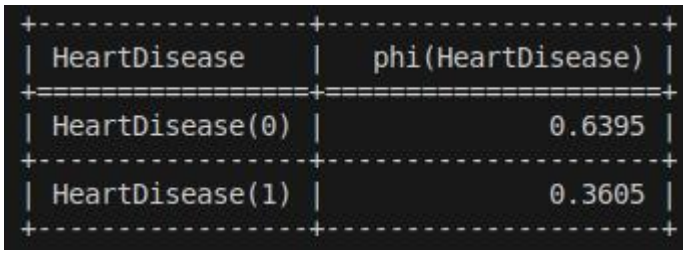
La distribuzione di probabilità è la seguente:



HeartDisease	phi(HeartDisease)
HeartDisease(0)	0.3343
HeartDisease(1)	0.6657

Otteniamo infatti una probabilità del 66,57 % di avere una malattia cardiaca.

Se invece proviamo ad eseguire la stessa query cambiando solo la fascia d’età da ’50-60’ a ‘Giovani’ otteniamo la seguente distribuzione di probabilità:



HeartDisease	phi(HeartDisease)
HeartDisease(0)	0.6395
HeartDisease(1)	0.3605

La probabilità di avere una malattia cardiaca si è ridotta al 36,05%.

Possibili estensioni future

Il progetto potrebbe essere esteso aggiungendo un modulo che, partendo dai dati di input del paziente, suggerisca delle possibili terapie. Per farlo, la base di conoscenze potrebbe essere arricchita con dati provenienti dal web semantico o da articoli scientifici.

Riferimenti Bibliografici

- Ragionamento relazionale: Poole, D., & Mackworth, A. (2017). *Artificial Intelligence: Foundations of Computational Agents* (3rd ed.). Cambridge University Press.

- Apprendimento supervisionato: Poole, D., & Mackworth, A. (2017). *Artificial Intelligence: Foundations of Computational Agents* (3rd ed.). Cambridge University Press. [Ch. 7].

- Apprendimento non supervisionato: Poole, D., & Mackworth, A. (2017). *Artificial Intelligence: Foundations of Computational Agents* (3rd ed.). Cambridge University Press. [Ch. 10].

- Ragionamento probabilistico e reti bayesiane: Poole, D., & Mackworth, A. (2017). *Artificial Intelligence: Foundations of Computational Agents* (3rd ed.). Cambridge University Press. [Ch. 9].

- Gruppo San Donato. (2024, settembre). *Colesterolo: i valori normali in base all'età*. https://www.grupposandonato.it/news/2024/settembre/colesterolo-valori-normali-seconda-eta#:~:text=I%20valori%20normali%20del%20colesterolo,mg%2F%20dl%20sono%20ritenuti%20pericolosi.

- Humanitas. (n.d.). *Pressione arteriosa: quando è alta e quando i valori sono nella norma*. https://www.humanitas.it/news/pressione-arteriosa-quando-e-alta-e-quando-i-valori-sono-nella-norma/

- Wikipedia contributors. *Sottolivellamento del tratto ST*. Wikipedia, L'enciclopedia libera. Disponibile su: https://it.wikipedia.org/wiki/Sottolivellamento_del_tratto_ST