Università degli Studi di Bari Aldo Moro
Dipartimento di Informatica
Ingegneria Della Conoscenza

Diagnosi delle Malattie Cardiache

Cannito Antonio

Matr. 777859

a.a 2024/2025

Link Repository GitHub :
Acannito2003/ICON: Diagnosi malattie cardiache per Esame Ingegneria della Conoscenza

INDICE

Capitolo 1

Introduzione

Secondo la World Health Organization (WHO), le malattie cardiovascolari rappresentano la principale causa di morte a livello globale, responsabili di circa 17,9 milioni di decessi ogni anno. Tra le patologie più comuni si annoverano infarto del miocardio, ictus, ipertensione e insufficienza cardiaca, spesso causate da una combinazione di fattori genetici, stili di vita poco salutari e condizioni mediche preesistenti come diabete e obesità.

La diagnosi precoce delle malattie cardiache è cruciale per un intervento tempestivo, poiché queste patologie possono progredire silenziosamente per anni, portando a gravi complicazioni o eventi fatali. Tecniche avanzate di Machine Learning possono essere impiegate per migliorare la capacità di previsione e identificazione precoce di queste condizioni, analizzando dati clinici e identificando pattern nascosti nei fattori di rischio.

Questo studio si propone di sviluppare strumenti basati su modelli di intelligenza artificiale per la diagnosi e la previsione del rischio cardiovascolare, con l'obiettivo di supportare i medici nel processo decisionale, migliorare l'efficienza della diagnosi e ridurre il tasso di mortalità associato a queste patologie.

STRUMENTI UTILIZZATI 1.1

Come linguaggio di programmazione si è scelto di usare Python, data la sua semplicità di scrittura e l'ampia disponibilità di librerie per il Machine Learning e l'analisi dei dati. Per l'esecuzione degli esperimenti è stato utilizzato PyCharm, un ambiente di sviluppo integrato (IDE) che offre strumenti avanzati per il debugging, la gestione delle dipendenze e l'organizzazione del codice.
Le principali librerie utilizzate includono:

- Pandas e NumPy per la gestione e manipolazione dei dati.
- Scikit-learn per l'implementazione di modelli di apprendimento supervisionato e non supervisionato.
- Imbalanced-learn per il bilanciamento delle classi nei dataset.
- PyTorch per la costruzione e l'addestramento di reti neurali.
- Pgmpy per la modellazione di reti bayesiane.
- Matplotlib per la visualizzazione dei dati e l'analisi esplorativa.

Tutte le librerie necessarie sono state installate e gestite tramite pip all'interno di un ambiente virtuale di PyCharm, al fine di garantire l'isolamento del progetto ed evitare conflitti con altre dipendenze. Le librerie specifiche utilizzate saranno descritte più nel dettaglio nelle rispettive sezioni del progetto.

Capitolo 2

Data Preprocessing

INFORMAZIONI SUL DATASET 2.1

Il dataset **heart.csv** trovato su Keggle contiene **1.025 campioni** di pazienti con **14 variabili cliniche** utili per la diagnosi delle malattie cardiache. Ecco la descrizione delle variabili presenti:

- **age**: Età del paziente (espressa in anni).
- **sex**: Sesso del paziente (0 = femmina, 1 = maschio).
- **cp**: Tipo di dolore toracico (da 0 a 3, dove 0 indica assenza di dolore e valori più alti indicano dolori più intensi e tipici di problemi cardiaci).
- **trestbps**: Pressione sanguigna a riposo (misurata in mmHg).
- **chol**: Livello di colesterolo sierico (mg/dL).
- **fbs**: Glicemia a digiuno (1 se ≥126 mg/dL, 0 altrimenti).
- **restecg**: Risultati dell'elettrocardiogramma a riposo (valori da 0 a 2).
- **thalach**: Frequenza cardiaca massima raggiunta durante un test da sforzo.
- **exang**: Angina indotta da esercizio (1 = sì, 0 = no).
- **oldpeak**: Depressione del segmento ST (misurata in deviazione rispetto alla linea di base, indica ischemia).
- **slope**: Pendenza del segmento ST durante l'esercizio (0 = discendente, 1 = piatta, 2 = ascendente).
- **ca**: Numero di vasi principali colorati con fluoroscopia (da 0 a 4).
- **thal**: Stato della talassemia (1 = normale, 2 = difettoso fisso, 3 = difettoso reversibile).
- **target**: Indicatore della presenza di malattia cardiaca (1 = presenza, 0 = assenza).

PRE PROCESSING DEL DATASET 2.2

Il dataset è stato inizialmente processato come segue:

1. Controllo valori nulli
2. Controllo valori duplicati: il dataset è stato analizzato per verificare la presenza di righe duplicate. Sono state rimosse 723 righe duplicate per evitare bias nell'addestramento e ridurre ridondanze nei dati.
3. Mapping delle variabili categoriche:
    - La feature sex è stata trasformata in una variabile binaria:
        - 0 → Female
        - 1 → Male
    - La feature cp (tipo di dolore toracico) è stata mantenuta con i suoi 4 livelli, ma potrebbe essere categorizzata ulteriormente in caso di necessità.
    - La feature thal è stata trasformata in categorie comprensibili:
        - 1 → Normale
        - 2 → Difettoso fisso
        - 3 → Difettoso reversibile
4. Encoding: per rendere utilizzabili le variabili categoriche nei modelli di apprendimento automatico, è stato applicato il One Hot Encoding alle feature sex, cp, restecg, slope, ca e thal. Questa tecnica permette di evitare che il modello interpreti erroneamente le relazioni numeriche tra categorie ordinali.
5. Scaling: le feature numeriche age, trestbps, chol, thalach e oldpeak saranno normalizzate o standardizzate in base al tipo di modello utilizzato.

Capitolo 3

Apprendimento Supervisionato

L'apprendimento supervisionato è una branca del Machine Learning in cui un modello viene addestrato utilizzando un insieme di esempi etichettati. Ogni esempio è descritto da un insieme di features (caratteristiche) di input e da una corrispondente feature target, che rappresenta la risposta corretta. L'obiettivo dell'apprendimento supervisionato è predire i valori del target per nuovi esempi, dato un set di features di input.

In questo contesto, ogni esempio è già associato alla sua risposta corretta, chiamata ground truth. Nei task di classificazione, la risposta target è un'etichetta discreta (ad esempio, "malattia cardiaca presente" o "malattia cardiaca assente"), mentre nei task di regressione, la risposta è un valore numerico continuo.

Nel caso specifico di questo studio, l'apprendimento supervisionato viene utilizzato per la classificazione delle malattie cardiache. Ogni esempio nel dataset rappresenta un paziente, con un insieme di caratteristiche mediche, come l'età, la pressione sanguigna, il colesterolo, e altre variabili cliniche. La feature target indica se il paziente è affetto da malattia cardiaca o meno, con valori tipicamente etichettati come "0" per l'assenza di malattia e "1" per la presenza di malattia.

L'obiettivo di questo modello è quello di prevedere, sulla base delle features di input (ad esempio, dati clinici del paziente), la probabilità che un paziente sviluppi o abbia una malattia cardiaca. In questo modo, il modello può essere utilizzato per effettuare previsioni accurate su nuovi pazienti, aiutando i medici a diagnosticare in modo più rapido e preciso, con un impatto significativo sulla prevenzione e gestione delle malattie cardiache.

METODOLOGIA 3.1

MODELLI DI APPRENDIMENTO SUPERVISIONATO

Per il task di apprendimento supervisionato, sono stati scelti due modelli di base e due modelli di ensemble, che includono un modello di **bagging** e uno di **boosting**:

- **Decision Tree**: Si tratta di un classificatore ad albero in cui le foglie rappresentano le classi di appartenenza, mentre la radice e i nodi interni rappresentano condizioni basate sulle **features di input**. A seconda che tali condizioni siano soddisfatte o meno, il percorso seguito dal modello varia. Alla fine, si giunge alla foglia, che fornisce la classificazione.

- **Random Forest**: Questo classificatore si ottiene creando numerosi **Decision Tree** indipendenti. La classificazione finale è determinata dalla maggioranza delle classi predette dai vari alberi del modello.

- **Logistic Regression**: Si tratta di un modello statistico che modella la probabilità di appartenenza a una classe utilizzando una funzione sigmoide. La funzione sigmoide trasforma una combinazione lineare delle **features** in un valore compreso tra 0 e 1. Il valore risultante viene quindi sogliato per assegnare l'osservazione a una delle due classi.

- **Gradient Boosting Classifier**: Questo è un modello di ensemble che combina diversi **Decision Tree** in modo sequenziale. Ogni nuovo albero è progettato per correggere gli errori commessi dai precedenti, dando maggiore peso agli esempi difficili da classificare. Il processo di apprendimento si basa sulla minimizzazione della funzione di perdita tramite il gradiente, migliorando progressivamente le performance del modello.

Per l'implementazione di questi modelli, è stata utilizzata la libreria **scikit-learn (sklearn)**.

Ricerca degli iperparametri ottimali

Per la selezione degli iperparametri, è stata adottata la metodologia **K-Fold Cross Validation**, che suddivide il dataset in *k* sottoinsiemi distinti (*fold*). Il modello viene addestrato *k* volte, utilizzando in ogni iterazione *k-1* fold per il training e il restante fold per il testing. Questo approccio permette di valutare il modello su dati diversi, migliorandone l'affidabilità.

In combinazione con questa tecnica, è stato applicato **GridSearch**, un metodo che esplora tutte le possibili configurazioni degli iperparametri, valutandole singolarmente mediante la Cross Validation. Alla fine, viene selezionata la combinazione di iperparametri con le prestazioni migliori per l'addestramento finale del modello.

Iperparametri

Gli iperparametri sono valori che influenzano l'addestramento del modello e devono essere impostati prima della fase di training, poiché non vengono appresi direttamente dal modello. La loro selezione è una fase critica, in quanto determina la complessità e la precisione delle previsioni.
Gli iperparametri ricercati per ciascun modello sono:

- **Decision Tree**:
    - *criterion*: metodo per valutare la qualità della suddivisione dei nodi (*"gini"*, *"entropy"*). Il primo misura la probabilità di errore di classificazione, il secondo quantifica il grado di disordine nei dati.
    - *max depth*: profondità massima dell'albero (*7, 10, 12*).
    - *min samples split*: numero minimo di campioni richiesti per dividere un nodo (*8, 10, 15*).
    - *min samples leaf*: numero minimo di campioni richiesti per formare una foglia (*5, 7, 10*).
    - *splitter*: criterio di suddivisione dei nodi, impostato su *"best"* per selezionare la partizione ottimale.

- **Random Forest**:
    - *n estimators*: numero di alberi nella foresta (*50, 100, 200*).
    - *criterion*: metodo per valutare la qualità della suddivisione dei nodi (*"gini"*, *"entropy"*).

    - *max depth*: profondità massima degli alberi (*5, 10, 15*).
    - *min samples split*: numero minimo di campioni richiesti per dividere un nodo (*5, 10, 15*).
    - *min samples leaf*: numero minimo di campioni richiesti per formare una foglia (*3, 5, 10*).

- **Logistic Regression**:
    - *C*: parametro di regolarizzazione; valori più bassi aumentano la regolarizzazione (*0.001, 0.01, 0.1, 1, 10, 100*).
    - *penalty*: tipo di penalizzazione applicata, fissata su *l2* (norma 2, che penalizza pesi elevati e previene l'overfitting).
    - *solver*: algoritmo di ottimizzazione (*"lbfgs"*, *"liblinear"*). Il primo utilizza derivate parziali per un consumo ridotto di memoria, il secondo si basa su programmazione lineare.
    - *max iter*: numero massimo di iterazioni per la convergenza (*100000, 150000*).

- **Gradient Boosting**:
    - *n estimators*: numero di alberi nella sequenza (*50, 100, 150*). Un numero maggiore migliora le prestazioni, ma aumenta il rischio di overfitting.
    - *learning rate*: velocità di apprendimento. Valori più bassi riducono il rischio di overfitting, mentre valori più alti accelerano l'addestramento (*0.01, 0.05, 0.1*).
    - *max depth*: profondità massima degli alberi (*5, 7, 10*).
    - *min samples split*: numero minimo di campioni richiesti per dividere un nodo (*5, 8, 12*).
    - *min samples leaf*: numero minimo di campioni richiesti per formare una foglia (*3, 5, 7, 10*).

Fase di addestramento e test

Dopo aver individuato la configurazione ottimale degli iperparametri, i modelli sono stati addestrati utilizzando la tecnica **Repeated K-Fold Cross Validation**, con i parametri *n_splits* e *n_repeats* impostati a 5. In questo metodo, il dataset viene suddiviso in 5 sottoinsiemi (folds) e il processo viene ripetuto 5 volte. A ogni iterazione, 4 fold vengono usati per il training e il restante per il testing, cambiando il fold di test a rotazione. Questo approccio consente di ottenere una valutazione più robusta delle prestazioni del modello.

Valutazione delle prestazioni

Per analizzare l'accuratezza dei modelli addestrati, sono state adottate le seguenti metriche di valutazione:

- **Accuracy**: misura complessiva della correttezza del modello, calcolata come il rapporto tra le previsioni corrette e il totale delle previsioni effettuate.

- **Precision Macro**: media delle precisioni calcolate per ciascuna classe. La precisione per una classe è data dal rapporto tra il numero di istanze correttamente classificate per quella classe e il totale delle istanze classificate in essa.

- **Recall Macro**: media delle recall calcolate per ogni classe. La recall per una classe è il rapporto tra il numero di istanze effettivamente appartenenti a quella classe e il numero totale di istanze di quella classe classificate correttamente.

- **F1 Macro**: media armonica della precisione e della recall per ogni classe. L'F1-score è calcolato come il rapporto tra 2 moltiplicato per il prodotto tra precisione e recall, diviso la loro somma.

Tuttavia, queste metriche, in alcune circostanze, potrebbero fornire una visione parziale delle prestazioni del modello. Per evitare valutazioni fuorvianti, sono stati considerati ulteriori indicatori:

- **Varianza e deviazione standard**: misurano la dispersione delle previsioni del modello. La varianza indica quanto le predizioni si discostano dalla loro media, mentre la deviazione standard, essendo la radice quadrata della varianza, fornisce una misura diretta della variabilità delle previsioni.

- **Curve di apprendimento**: rappresentano graficamente l'andamento dell'errore nelle fasi di training e test, consentendo di identificare problemi di underfitting, overfitting o un corretto apprendimento del modello.

Un modello con bassa varianza e deviazione standard è generalmente preferibile, poiché le sue previsioni risultano più stabili e meno soggette a fluttuazioni casuali. In altre parole, un modello con queste caratteristiche è più robusto e meno incline all'overfitting, permettendogli di generalizzare meglio su dati mai visti prima.

PREPARAZIONE DEL DATASET 3.2

Per l'addestramento dei modelli di Machine Learning, la gestione della scala delle feature è un aspetto fondamentale.
Nel caso di modelli come **Decision Tree, Random Forest e Gradient Boosting**, il dataset può essere utilizzato direttamente senza necessità di modifiche alla scala dei valori. Questo perché tali modelli sono basati su suddivisioni gerarchiche dei dati e non vengono influenzati dalla magnitudine assoluta delle feature.
Diversamente, la **Logistic Regression** assegna pesi a ciascuna feature durante l'addestramento e, di conseguenza, risente della scala dei valori. Se le feature hanno scale molto diverse, alcune potrebbero dominare il processo di apprendimento, influenzando il modello e ostacolando la sua capacità di apprendere le relazioni corrette tra i dati.
Per evitare questo problema, si applica la **normalizzazione delle feature**, che riduce i valori di ciascuna feature in un intervallo compreso tra **0 e 1**. La trasformazione segue la formula:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Dove:
- x' è il valore normalizzato,
- x è il valore originale della feature,

- min(x) e max(x) sono rispettivamente il minimo e il massimo valore osservato per quella feature.

Questa operazione consente alla Logistic Regression di trattare tutte le feature con lo stesso peso relativo, migliorando la stabilità numerica e l'efficienza dell'addestramento.
Per effettuare la normalizzazione, è stato utilizzato l'oggetto **MinMaxScaler** della libreria **scikit-learn**, che applica automaticamente la trasformazione ai dati prima dell'addestramento del modello.

ESPERIMENTO 3.3

Iperparametri trovati con GridSearch :

Modello	Parametro	Valore
Decision Tree	criterion	gini
	max_depth	7
	min_samples_split	10
	min_samples_leaf	10
Random Forest	n_estimators	50
	max_depth	10
	min_samples_split	15
	min_samples_leaf	5
	criterion	Gini
Logistic Regression	C	10
	penalty	l2
	solver	liblinear
	max_iter	100000
Gradient Boosting Classifier	n_estimators	150
	learning_rate	0.05
	max_depth	5
	min_samples_split	8
	min_samples_leaf	10

Tabella 3.1: Tabella degli iper-parametri trovati con Grid Search

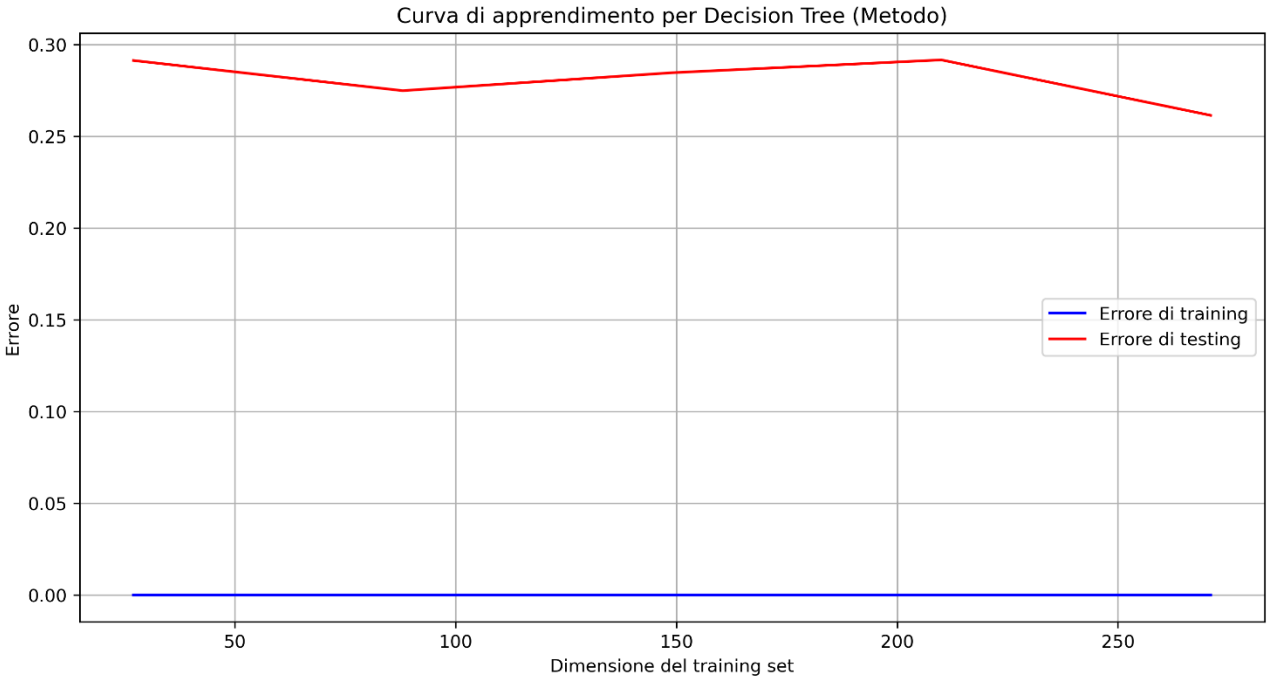


Figura 3.1: Curva di Apprendimento Decision Tree

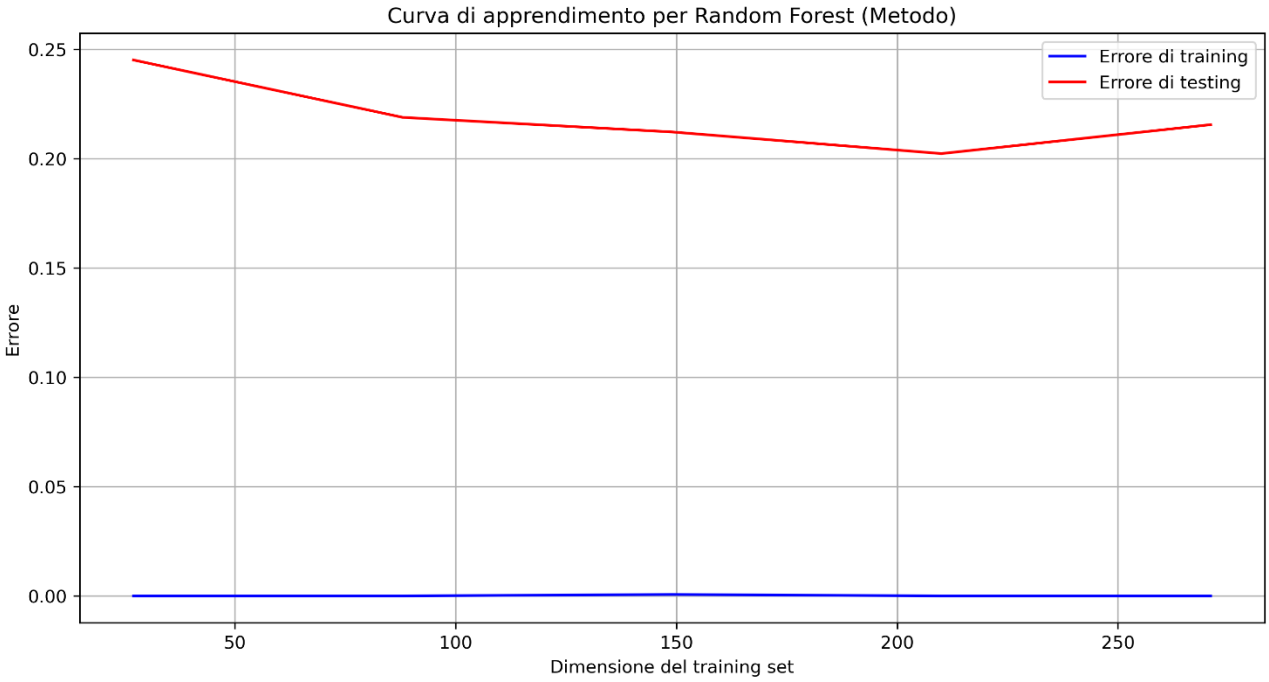


Figura 3.2: Curva di Apprendimento Random Forest

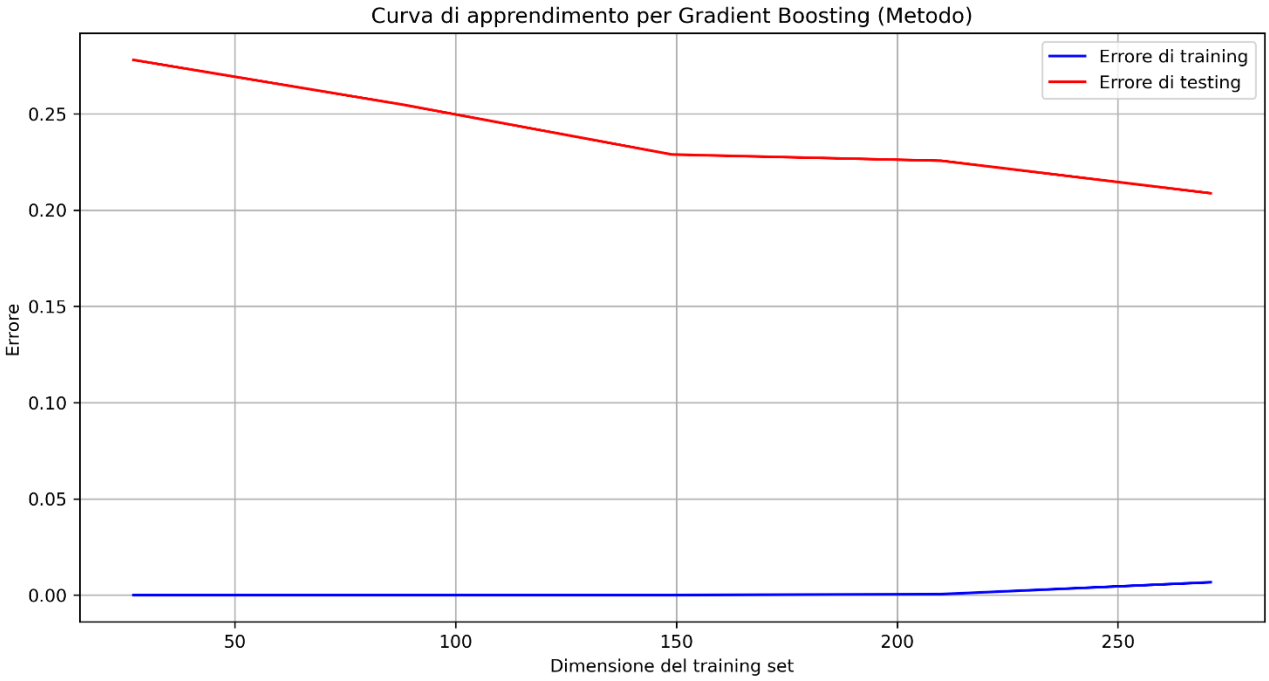


Figura 3.3: Curva di Apprendimento Gradient Boosting Classifier

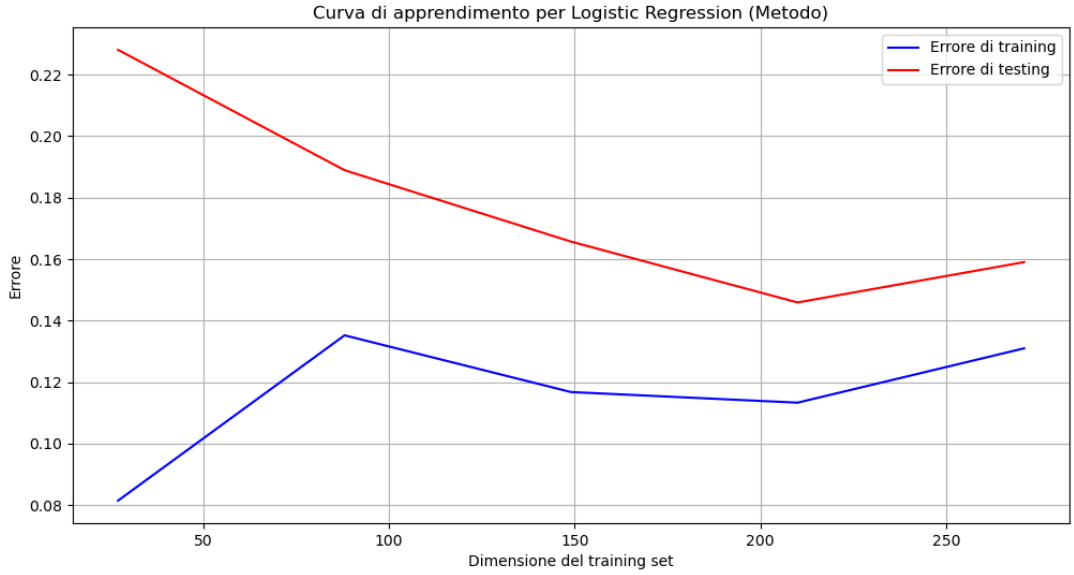
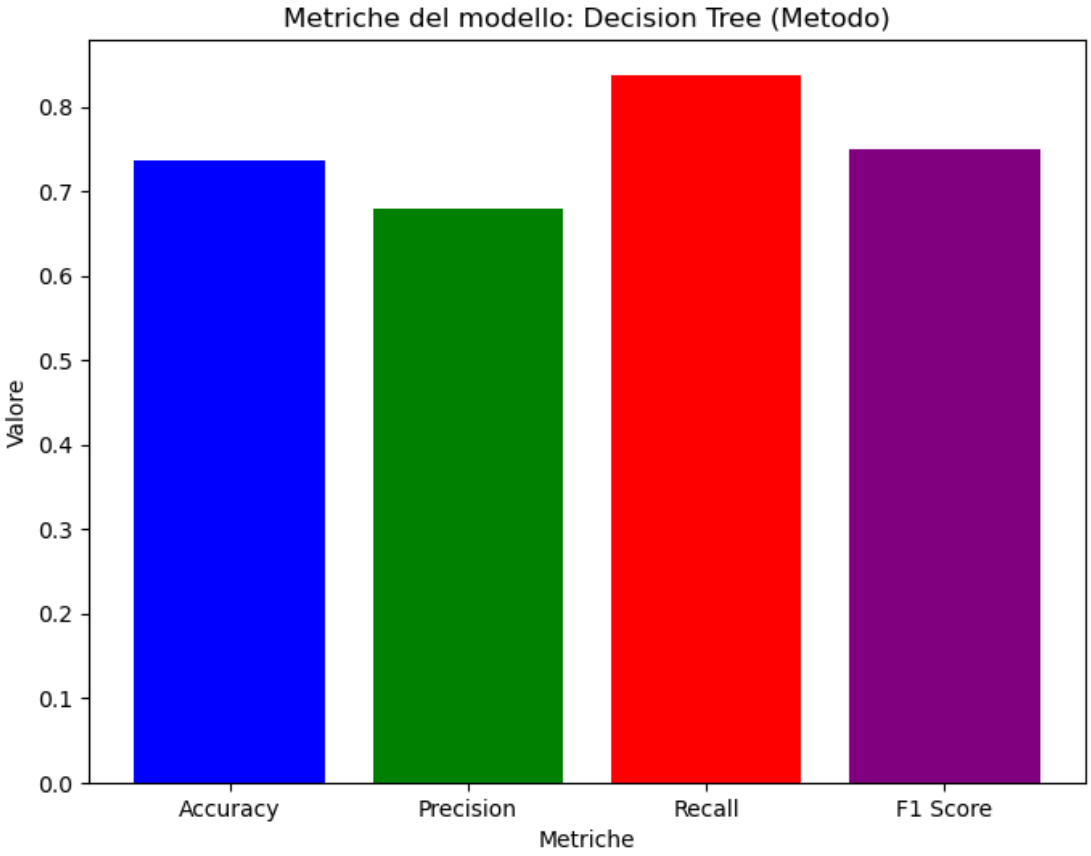
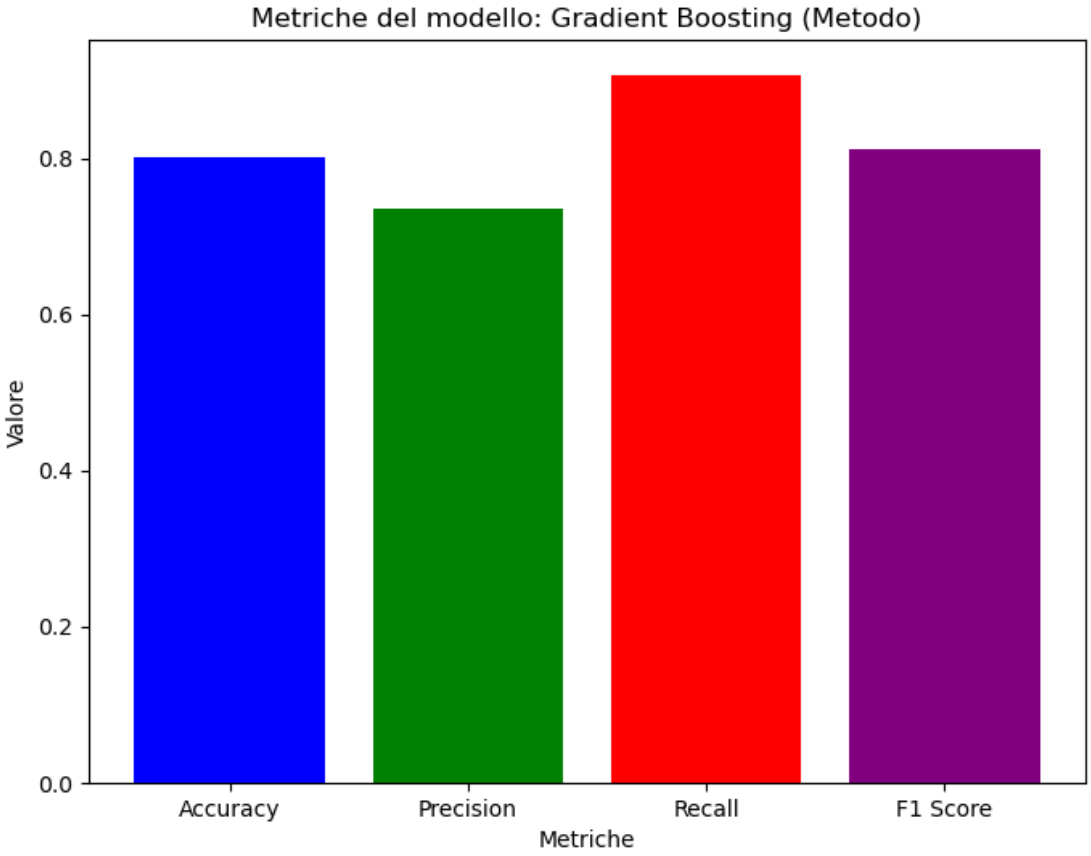
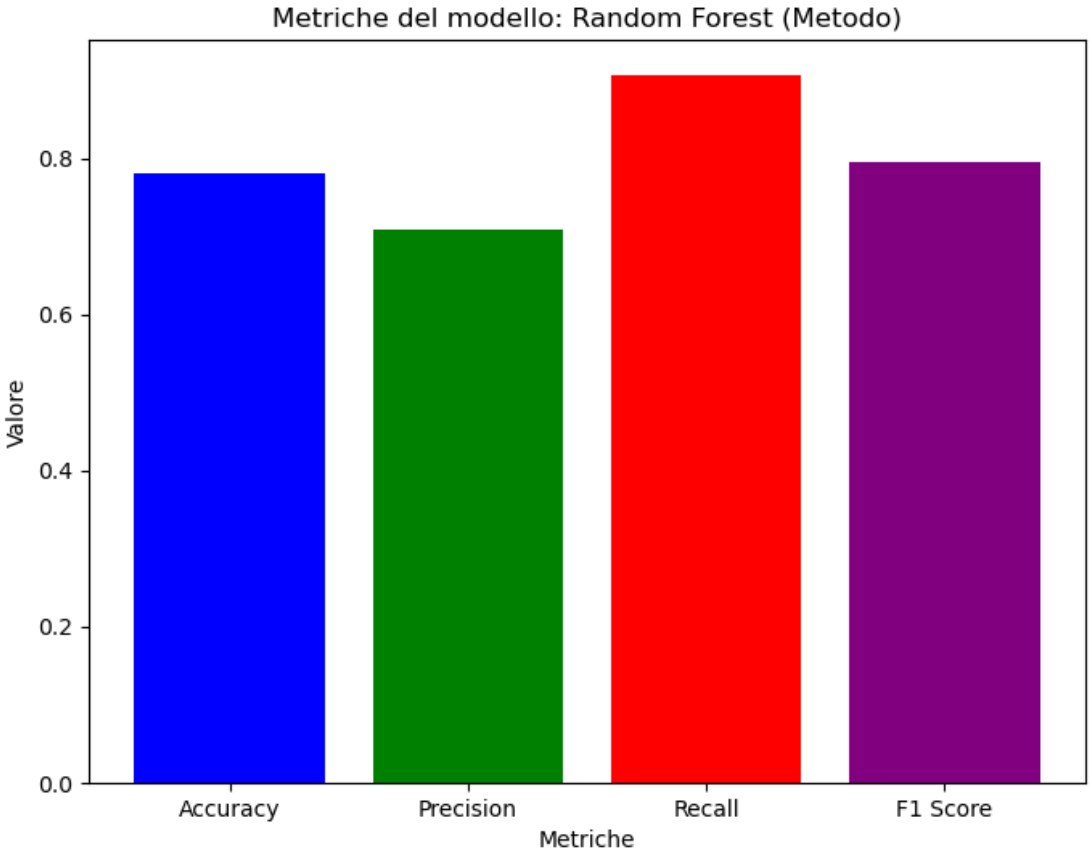


Figura 3.4: Curva di Apprendimento Logistic Regression


Metriche del modello: Decision Tree (Metodo)
Valore
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Accuracy
Precision
Recall
F1 Score
Metriche


Metriche del modello: Gradient Boosting (Metodo)
Valore
0.0
0.2
0.4
0.6
0.8
Accuracy
Precision
Recall
F1 Score
Metriche


Metriche del modello: Random Forest (Metodo)
Valore
0.0
0.2
0.4
0.6
0.8
Accuracy
Precision
Recall
F1 Score
Metriche

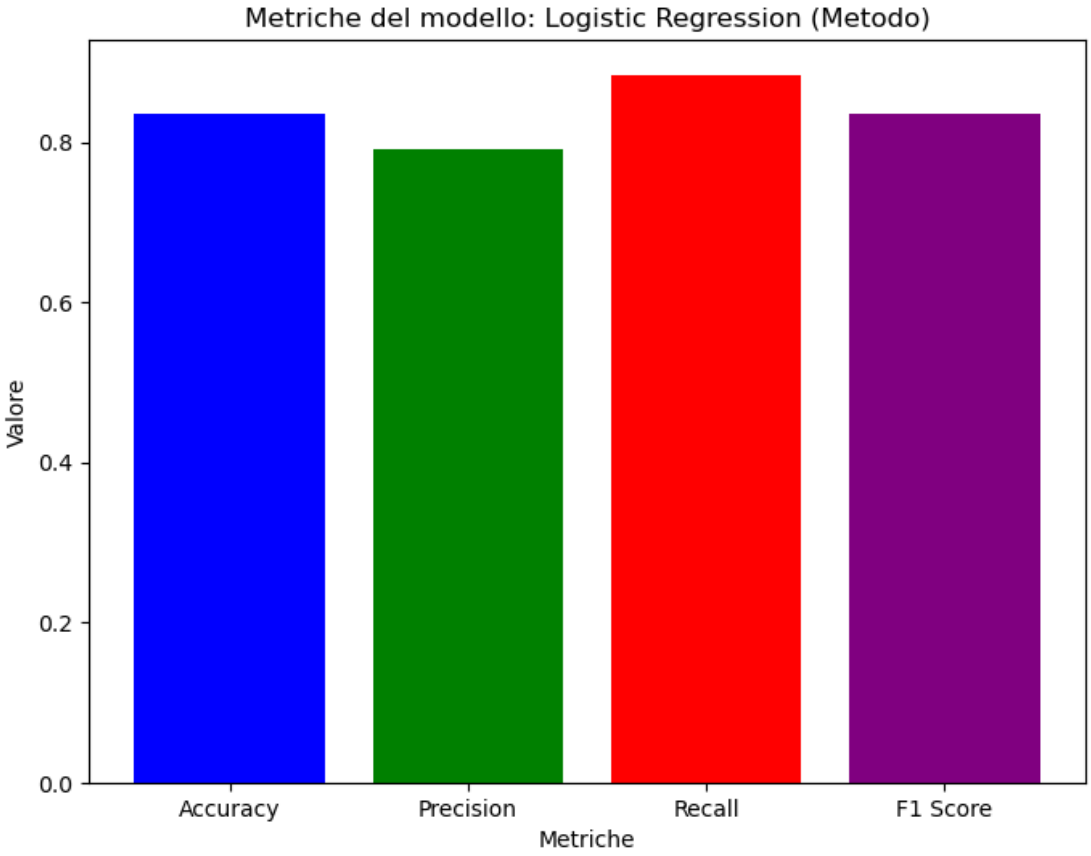


Figure 3.5: Metriche di valutazione

Varianza E Deviazione Standard Degli Errori Per Modello

	Modello	Varianza degli errori	Deviazione standard degli errori
1	Decision Tree	0.2244	0.4737
2	Random Forest	0.2249	0.4742
3	Gradient Boosting	0.1857	0.431
4	Logistic Regression	0.1618	0.4023

Tabella 3.2: Confronto Varianza degli Errori per i Modelli e Deviazione Standard dell'errore per i Modelli

CONSIDERAZIONI 3.3.1

Decision Tree

- Curve di apprendimento: La curva di training mostra un rapido calo iniziale dell'errore, seguita da una stabilizzazione. La curva di testing diminuisce più gradualmente. Le due curve si avvicinano, suggerendo che il modello sta iniziando a generalizzare abbastanza bene, ma potrebbe ancora trarre beneficio da ulteriori dati.
- Metriche: L'accuracy e la precision sono elevate, ma il recall è più basso, indicando che il modello fatica a riconoscere la classe minoritaria (pazienti con malattia cardiaca). Questo potrebbe suggerire un problema di sbilanciamento delle classi.

Random Forest

- Curve di apprendimento: La curva di training diminuisce rapidamente, stabilizzandosi successivamente. La curva di testing è più alta ma in leggera diminuzione. La distanza tra le due curve è ampia, ma tende a convergere, il che indica che il modello potrebbe migliorare con più dati per generalizzare meglio.
- Metriche: L'accuracy e la precision sono elevate, con un lieve miglioramento del recall rispetto al Decision Tree, ma il modello continua ad avere difficoltà con la classe minoritaria.

Logistic Regression

- Curve di apprendimento: L'errore di training è leggermente più alto rispetto agli altri modelli, segnalando che la Logistic Regression fatica a adattarsi ai dati. La curva di testing cresce dopo una fase iniziale, suggerendo che il modello potrebbe avere difficoltà a generalizzare sui nuovi dati.
- Metriche: L'accuracy, la precision, il recall, e la F1-score sono più basse rispetto agli altri modelli. Ciò indica che la Logistic Regression non è in grado di riconoscere efficacemente la classe minoritaria e ha prestazioni inferiori.

Gradient Boosting

- Curve di apprendimento: L'errore di training è il più basso tra tutti i modelli e la curva di testing scende in modo graduale. La distanza tra le curve è ridotta, indicando una buona capacità di generalizzazione. Tuttavia, la curva di testing mostra una lieve fluttuazione, segnalando una possibile variazione nei dati.
- Metriche: Il recall è ancora inferiore rispetto alla accuracy e alla precision, ma le prestazioni generali sono superiori agli altri modelli, indicando che Gradient Boosting potrebbe essere il modello migliore per questo dataset.

Considerazioni Finali:

- Modello migliore: Gradient Boosting si distingue per le sue elevate prestazioni generali, con un'alta accuracy e precision. Sebbene il recall sia più basso, potrebbe essere migliorato con tecniche di bilanciamento delle classi, come l'oversampling della classe minoritaria.
- Modelli da migliorare: Logistic Regression è il modello con le performance più basse e potrebbe non essere la scelta ideale per questo tipo di dataset. Gli altri modelli, come Decision Tree e Random Forest, mostrano buone performance, ma presentano problemi simili nel riconoscere correttamente la classe minoritaria.

DISTRIBUZIONE MALATTIA CARDIACA

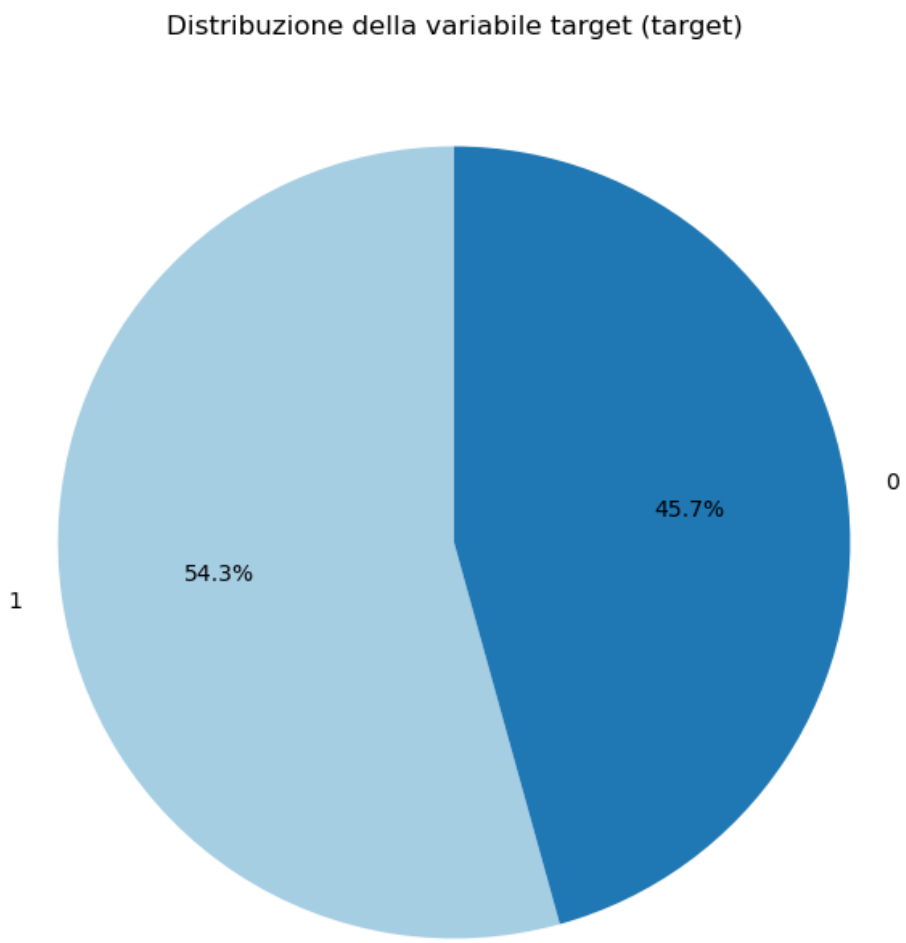


Figura 3.6: Distribuzione Target con colore celeste diagnosi positiva e blu negativa

Possibili problemi derivanti dalla distribuzione della variabile target

- I modelli di apprendimento supervisionato potrebbero tendere a predire sempre la classe maggioritaria, ottenendo un'alta accuratezza senza realmente apprendere la classe minoritaria. Questo comportamento è tipico dei cosiddetti **modelli "dumb"**.
- Le metriche di valutazione potrebbero risultare ingannevoli, facendo sembrare i modelli più accurati di quanto lo siano realmente.
- I modelli, non ricevendo abbastanza esempi della classe minoritaria, potrebbero non essere in grado di **apprendere a fondo** le caratteristiche di tale classe.
- Il rischio di **overfitting** aumenta, portando a scarse capacità di generalizzazione e a prestazioni insoddisfacenti su nuovi dati contenenti la classe minoritaria.

Bilanciamento della variabile target con OverSampling

Per affrontare questi problemi, è stato effettuato un bilanciamento della variabile target utilizzando la tecnica **SMOTE (Synthetic Minority Over-sampling Technique)** della libreria imblearn.over_sampling.
Questa tecnica genera **esempi sintetici** della classe minoritaria (Diagnosi Positiva) fino a ottenere una distribuzione equilibrata, ovvero **50% Diagnosi Positiva e 50% Diagnosi Negativa**.

Dopo l'applicazione di OverSampling, la nuova distribuzione della variabile target risulta essere la seguente:

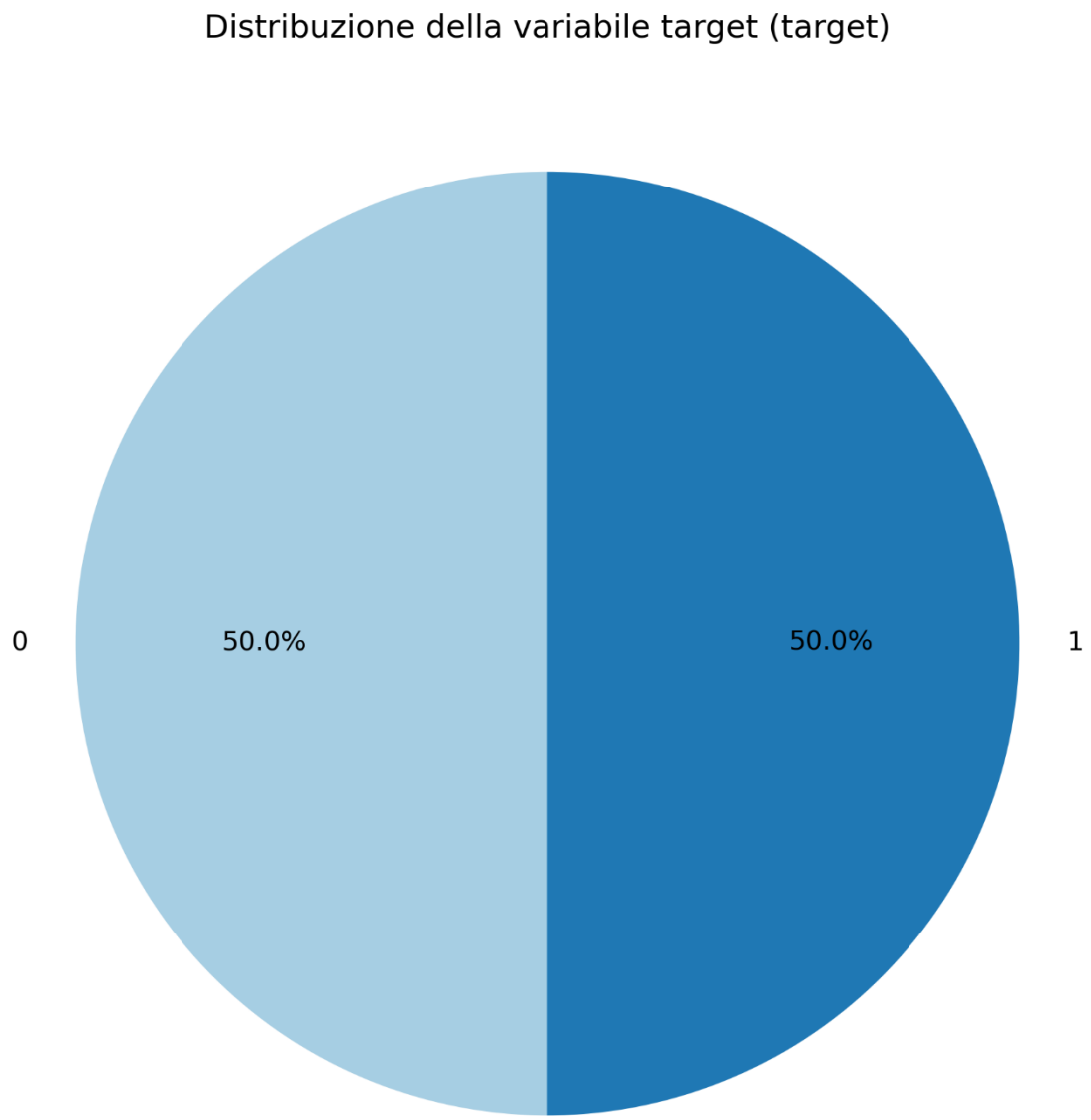


Figura 3.7: Distribuzione Target con OverSampling con celeste diagnosi positiva e blu diagnosi negativa

Capitolo 4

Apprendimento Non Supervisionato

L’apprendimento non supervisionato è una disciplina del Machine Learning in cui un modello viene addestrato esclusivamente sulle feature di input, senza disporre di un'etichetta target (ground truth), a differenza dell’apprendimento supervisionato. L'obiettivo principale è identificare schemi nascosti e raggruppamenti nei dati.

Nel nostro caso, l’apprendimento non supervisionato non è stato applicato all’intero dataset, poiché la presenza delle diagnosi renderebbe inutile tale approccio. Al contrario, è stato impiegato solo sul sottoinsieme di pazienti con diagnosi positiva, con l'intento di individuare eventuali schemi e suddivisioni tra i pazienti affetti da malattie cardiache. Questo processo mira a riconoscere gruppi con caratteristiche comuni, per esempio distinguendo sottogruppi in base a diversi fattori di rischio o tipologie specifiche della patologia.

METODOLOGIA 4.1

Per l’apprendimento non supervisionato è stato scelto l'algoritmo K-Means, un metodo di clustering che suddivide i dati in k gruppi in base alla similarità tra gli esempi e i centroidi dei cluster. Inizialmente, i centroidi vengono selezionati in modo casuale (init = "random"). Successivamente, ogni dato viene assegnato al cluster del centroide più vicino, calcolando la distanza tra essi utilizzando la distanza euclidea. Dopo ogni assegnazione, i centroidi vengono aggiornati come la media dei punti assegnati al rispettivo cluster. Il processo si ripete iterativamente fino alla convergenza, ossia quando i centroidi si stabilizzano o viene raggiunto il numero massimo di iterazioni.

La scelta del numero di cluster k è cruciale e deve essere determinata con attenzione. Per questo, è stato adottato il metodo Elbow Rule, che consente di identificare il valore ottimale di k analizzando l’inerzia del modello, ossia la somma delle distanze quadratiche tra ciascun punto e il proprio centroide. L'inerzia tende a diminuire all'aumentare di k, ma oltre un certo punto la riduzione diventa meno evidente, creando una curva che presenta un “gomito”. Il valore di k corrispondente a questa svolta è considerato ottimale, in quanto rappresenta il miglior compromesso tra la compattezza dei cluster e la riduzione della varianza intra-cluster, senza introdurre complessità eccessiva.

Per l'implementazione è stato utilizzato KMeans dalla libreria sklearn.cluster, mentre per determinare il miglior valore di k è stata impiegata la funzione KneeLocator della libreria kneed.

Valutazione

Poiché nell’apprendimento non supervisionato non esistono etichette di riferimento per valutare direttamente le prestazioni, si utilizzano metriche che analizzano la coerenza

interna, la separabilità dei cluster e l'adattabilità alla struttura dei dati. Le principali metriche utilizzate sono:

- **Silhouette Score**: valuta quanto un punto è vicino al cluster di appartenenza rispetto agli altri cluster. Il punteggio varia tra -1 e +1, dove -1 indica una pessima assegnazione e +1 indica un clustering ottimale.

- **Within-Cluster Sum of Squares (WCSS):** misura la somma delle distanze quadratiche tra i punti e il proprio centroide all'interno di ciascun cluster. Un valore basso indica che i punti di un cluster sono ravvicinati (clustering più compatto), mentre un valore alto suggerisce cluster dispersi e meno efficaci. Il WCSS è fondamentale per determinare il numero ottimale di cluster tramite la Elbow Rule.

PREPARAZIONE DEL DATASET 4.2

Come primo passo, sono stati **rimossi i pazienti senza diagnosi di malattia cardiaca** (ovvero quelli con la feature target = 0), poiché l'obiettivo dell'analisi è individuare possibili sottogruppi all'interno dei pazienti affetti da patologie cardiache. Successivamente, è stata eliminata la variabile target, in quanto, avendo lo stesso valore per tutti i pazienti considerati, non avrebbe fornito informazioni utili per la differenziazione tra i vari casi.

Poiché l'algoritmo di clustering **K-Means** utilizza la **distanza euclidea** per raggruppare i dati, la presenza di feature con scale di valori molto diverse tra loro avrebbe potuto compromettere l'efficacia del modello. Per questo motivo, è stata applicata la **normalizzazione** delle feature, scalando tutti i valori nell'intervallo **[0,1]** con la seguente formula:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Questa trasformazione è stata realizzata utilizzando **MinMaxScaler** della libreria sklearn.preprocessing, che garantisce che tutte le feature abbiano la stessa scala, evitando che variabili con valori numerici più elevati influenzino eccessivamente il clustering. Questo passaggio è fondamentale per garantire un raggruppamento più accurato e significativo dei pazienti con **malattie cardiache**, identificando possibili sottogruppi con caratteristiche comuni.

ESPERIMENTO 4.3

Il numero ottimale k di cluster è 4, trovato tramite l'Elbow Rule e mostrato nel seguente grafico

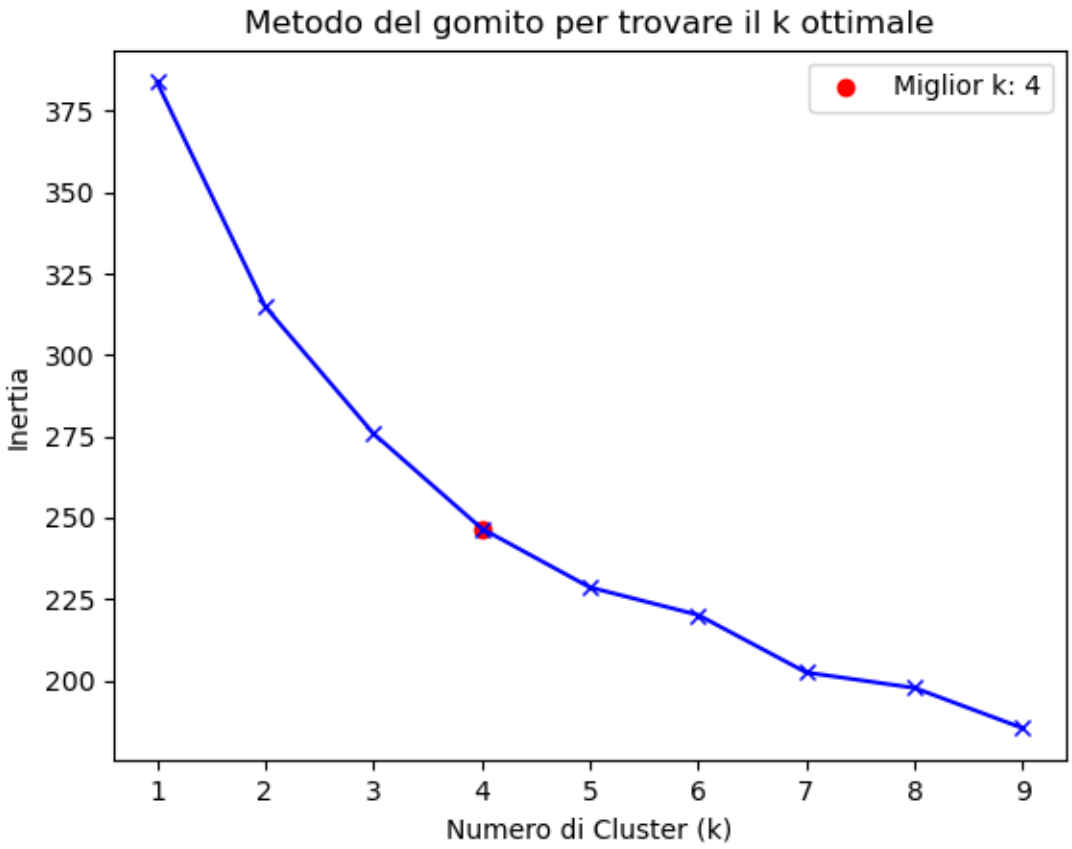


Figura 4.1: Metodo del gomito

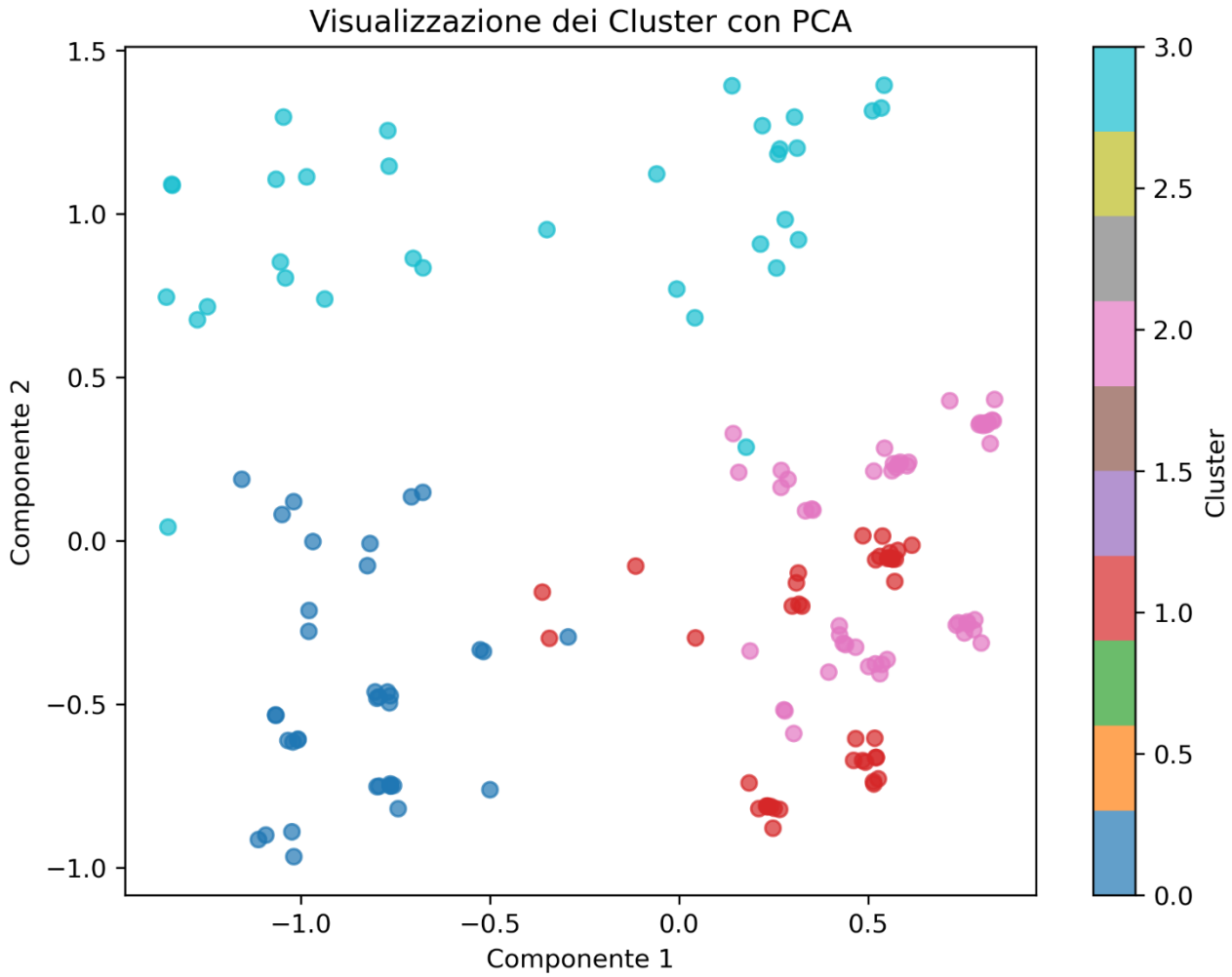
Dopo aver suddiviso i dati in quattro cluster utilizzando l'algoritmo di clustering K-Means (k=4), come determinato dal Metodo del Gomito, è stato utile rappresentarli in uno spazio bidimensionale per valutare la separabilità e la distribuzione dei pazienti nei gruppi individuati. Poiché il dataset originale contiene numerose feature, sono state impiegate tecniche di riduzione della dimensionalità, come PCA e t-SNE, per ottenere una rappresentazione visiva chiara dei cluster senza alterarne la struttura.

- **PCA (Principal Component Analysis)** riduce la dimensionalità dei dati trovando le componenti principali, ossia le combinazioni lineari delle feature che spiegano la maggior parte della varianza. Questo metodo proietta i dati in uno spazio bidimensionale, offrendo una rappresentazione lineare dei cluster. Questa visualizzazione aiuta ad analizzare la distribuzione globale dei pazienti, evidenziando eventuali sovrapposizioni tra gruppi e somiglianze nelle caratteristiche cliniche.

- **t-SNE (t-Distributed Stochastic Neighbor Embedding)** conserva le **relazioni di vicinanza** tra i pazienti, mettendo in evidenza la **struttura locale** dei dati e rivelando **possibili**

sottogruppi all’interno dei cluster. Questo approccio è particolarmente utile quando i cluster non hanno una forma **lineare**, permettendo di individuare gruppi con caratteristiche simili in termini di fattori di rischio o manifestazioni della malattia cardiaca.

Questa analisi visiva permette di comprendere meglio come i pazienti affetti da malattie cardiache si distribuiscano nei vari cluster, fornendo spunti per identificare sottogruppi con caratteristiche comuni che potrebbero essere rilevanti per la ricerca clinica o per la personalizzazione dei trattamenti.


Visualizzazione dei Cluster con PCA
Componente 2
Componente 1
Cluster
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

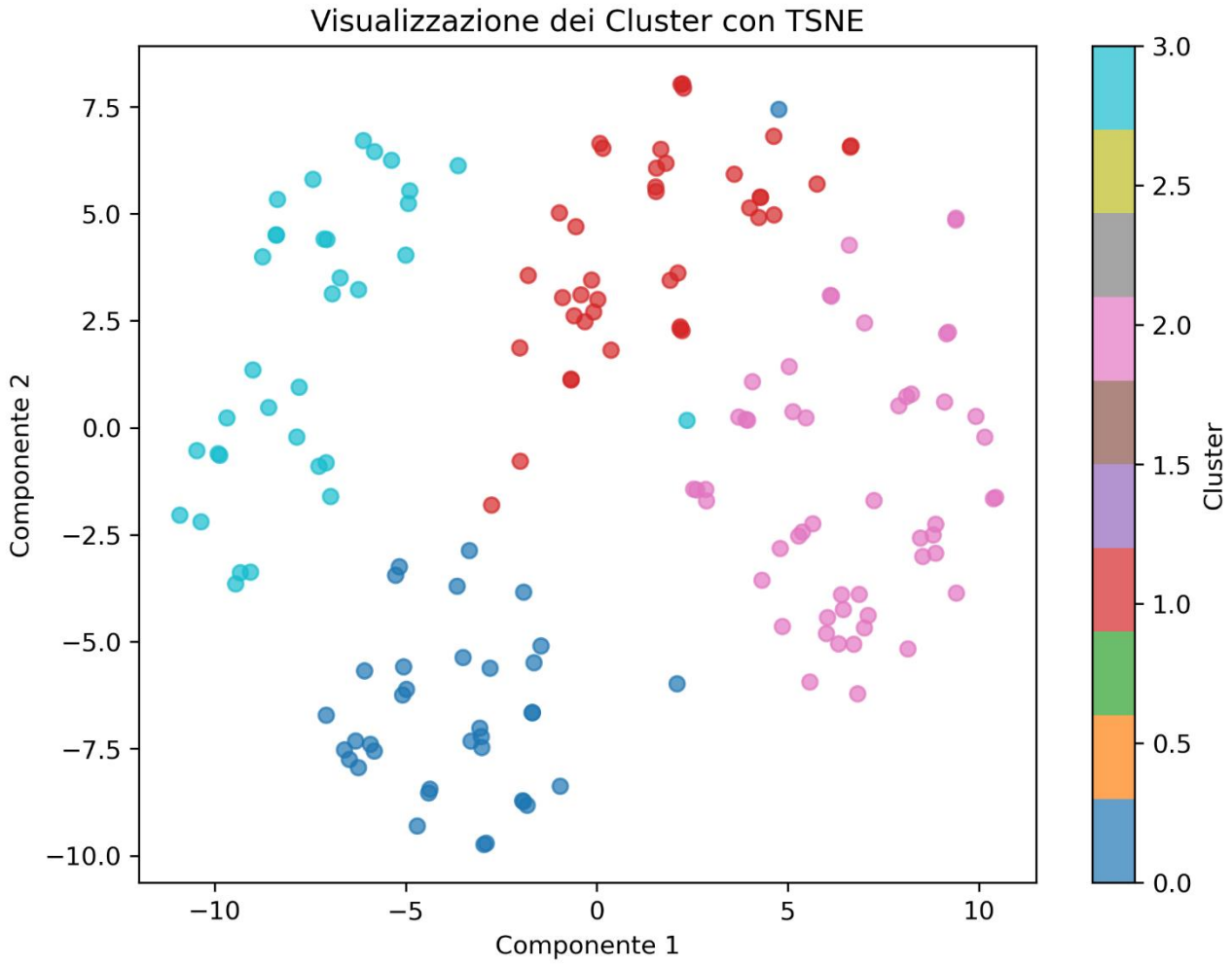


Figure 4.2: Visualizzazone clustering PCA e TSNE

Dopo aver eseguito il clustering sui pazienti affetti da malattie cardiache, sono stati ottenuti i seguenti risultati:

- WCSS (Within-Cluster Sum of Squares): 244.3819903553186
- Silhouette Score: 0.17597052527490636

Interpretazione delle Metriche

- **WCSS (Compattezza dei Cluster)**

  Il valore di WCSS relativamente basso indica che i cluster sono più compatti, suggerendo che i pazienti raggruppati insieme presentano una certa omogeneità nei loro dati clinici. Tuttavia, il valore da solo non è sufficiente per determinare la qualità del clustering e deve essere interpretato insieme ad altre metriche.

- **Silhouette Score (Separabilità dei Cluster)**

Il punteggio di 0.17597052527490636i indica che i cluster sono moderatamente distinti, con una separazione parziale tra i gruppi. Questo suggerisce che esiste una certa sovrapposizione tra alcuni cluster, il che potrebbe essere dovuto a caratteristiche condivise tra sottogruppi di pazienti.

Analisi Grafica con PCA e t-SNE

Le rappresentazioni bidimensionali con PCA e t-SNE mostrano una segmentazione parzialmente efficace:

- PCA evidenzia gruppi distinti ma con una certa sovrapposizione tra alcuni cluster.
- t-SNE fornisce una visione più dettagliata della struttura locale, mettendo in evidenza sottogruppi più chiari rispetto a PCA, anche se non perfettamente separati.

Capitolo 5

Ragionamento Probabilistico e Bayesian Network

Il ragionamento probabilistico è un approccio basato sulla teoria della probabilità, che analizza la dipendenza e l'indipendenza tra variabili e sfrutta il Teorema di Bayes. In questo contesto, si associano probabilità a eventi e ipotesi, utilizzando le probabilità a posteriori per trarre conclusioni. Un esempio applicativo di questo tipo di ragionamento è rappresentato dalle Reti Bayesiane, che permettono di modellare relazioni probabilistiche tra variabili in modo strutturato ed efficace.

PREPARAZIONE AL DATASET 5.1

Abbiamo risolto il problema della gestione delle variabili continue nella Rete Bayesiana applicando una discretizzazione automatica. In particolare, abbiamo utilizzato il KBinsDiscretizer della libreria sklearn.preprocessing per trasformare le variabili continue in categorie discrete.

Modifiche effettuate:

- Conversione delle feature continue in dati discreti per permettere il calcolo delle distribuzioni di probabilità condizionate (CPD).
- Uso di KBinsDiscretizer con:
    - Strategia "uniform": le feature sono state suddivise in bin di ampiezza uguale.
    - Encoding ordinale: ogni bin è stato rappresentato da un numero intero, mantenendo l'ordine tra i valori.

Questa soluzione ha permesso di ridurre il carico computazionale ed evitare il problema del segmentation fault, rendendo il modello applicabile senza comprometterne l'efficacia.

APPRENDIMENTO STRUTTURA 5.2

Per l'implementazione della **Rete Bayesiana**, è stata utilizzata la libreria **pgmpy**, mentre per la visualizzazione della struttura è stata impiegata **networkx**.

La struttura della rete è stata appresa attraverso **HillClimbSearch**, un algoritmo di **ricerca locale** che esplora diverse configurazioni possibili della rete. Il processo inizia con una struttura iniziale casuale e la ottimizza progressivamente attraverso miglioramenti iterativi. Per la stima dei parametri, è stato adottato il metodo della **Massima Verosimiglianza (Maximum Likelihood Estimation, MLE)**, che consente di calcolare le probabilità condizionate tra le variabili in base ai dati osservati.

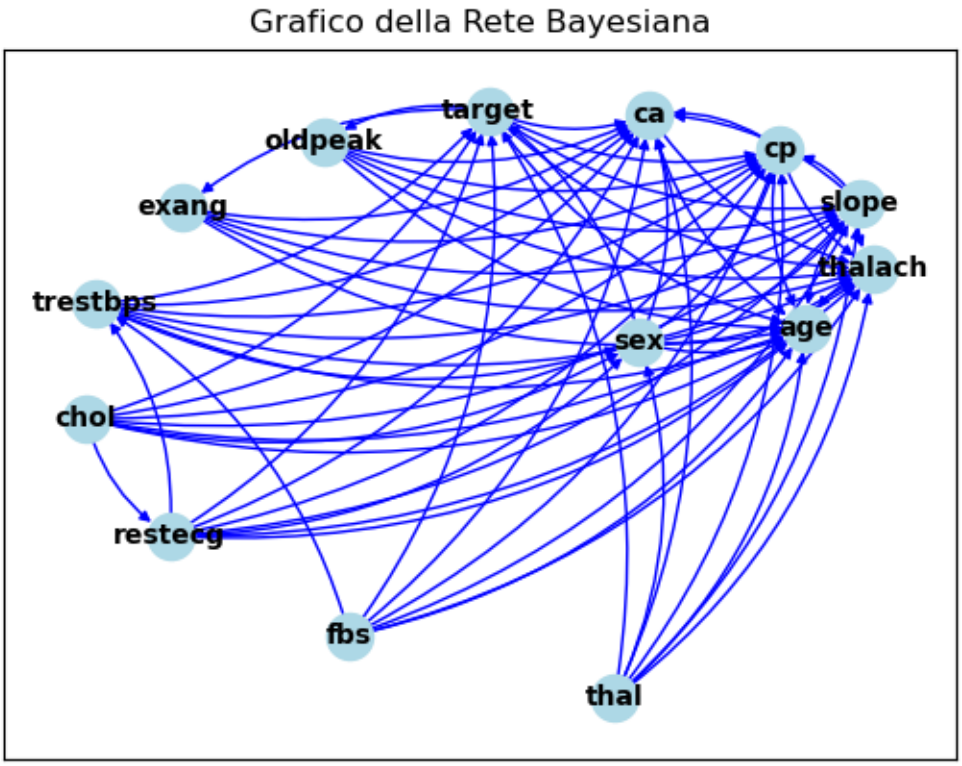


Figura 5.1: Grafico della Rete Bayesiana

Dopo aver appreso la **struttura della Rete Bayesiana**, uno degli utilizzi principali è l'**inferenza probabilistica**, che consente di calcolare la **probabilità di una o più variabili incognite** sulla base dei valori osservati di altre variabili.

Nel contesto della **Rete Bayesiana per le malattie cardiache**, questo significa poter stimare la probabilità di sviluppare una **patologia cardiaca** in base a fattori di rischio come **età, pressione sanguigna e colesterolo**. L'inferenza si basa sul **Teorema di Bayes**, che permette di aggiornare le probabilità degli eventi quando vengono fornite nuove evidenze cliniche, migliorando la capacità del modello di rappresentare relazioni causali tra le variabili.

Questo approccio è utile per supportare decisioni cliniche, identificare sottogruppi di pazienti a **maggiore rischio** e comprendere meglio l'impatto di **diversi fattori di salute** sulla probabilità di malattia.

ESEMPIO : GENERAZIONE DI CAMPIONI E GESTIONE DI DATI MANCANTI 5.3

Una volta appresa la struttura della Rete Bayesiana, è possibile utilizzarla per generare nuovi campioni di dati. Questa funzionalità consente di simulare nuove configurazioni di variabili, fornendo un metodo efficace per analizzare scenari ipotetici o supportare inferenze su dati non osservati.

La generazione di nuovi campioni sfrutta le probabilità apprese dal dataset durante l'addestramento del modello, creando esempi sintetici che rispettano le distribuzioni probabilistiche delle variabili.

Dati Esempio Generato

Variabile	Valore
chol	0
thal	4
trestbps	3
slope	4
thalach	4
Sex	4
age	4
fbs	4
cp	4
target	4
ca	4
restecg	2
exang	4
oldpeak	0

Tabella 5.1: Dati esempio generato

I valori delle feature per l'esempio randomico vengono generati già discretizzati.
Le reti bayesiane sono in grado di gestire casi in cui una variabile di input non è nota. Questo è utile in quanto i dati reali spesso presentano valori mancanti. La Rete Bayesiana è in grado di fare inferenza su nuovi dati, anche se non tutte le variabili sono note.

Ad esempio, eliminando la variabile **trestbps** dal precedente esempio generato e provandone a predire il valore per inferenza sulla Rete Bayesiana, si ottiene:

Inferenza sulla variabile trestbps

trestbps	Probabilità
trestbps(0)	0.1147
trestbps(1)	0.5108
trestbps(2)	0.0000
trestbps(3)	0.3746
trestbps(4)	0.0000

Tabella 5.2: Inferenza sulla variabile trestbps

Possiamo notare che il risultato non è consistente con il valore effettivo, in quanto il valore effettivo è quello con probabilità (**trestbps(3) = 0.3746**), facendoci notare che la rete potrebbe essere migliorata.

ESEMPIO : QUERY 5.4

Dopo aver costruito la rete bayesiana, possiamo eseguire interrogazioni probabilistiche per stimare la probabilità di una variabile sconosciuta, dato un insieme di informazioni note.

Tuttavia, calcolare queste probabilità è un'operazione computazionalmente onerosa, poiché richiede l'applicazione del teorema di Bayes, la moltiplicazione di fattori probabilistici e la marginalizzazione sulle variabili nascoste. Il problema principale è che il numero di combinazioni cresce esponenzialmente con il numero di nodi nella rete, rendendo l'inferenza diretta impraticabile su reti complesse.

Per rendere il processo più efficiente, utilizziamo Variable Elimination, un metodo implementato nella libreria pgmpy.inference. Questo approccio semplifica i calcoli eliminando iterativamente le variabili latenti e aggregando le probabilità, riducendo così il carico computazionale rispetto all'inferenza tradizionale.

Esempio di interrogazione sulla rete bayesiana

"Se sappiamo che un paziente ha un'età discretizzata pari a (3), un livello di colesterolo (chol) pari a (2) dolore toracico (cp= 3) e ha un valore di slope = 0, qual è la distribuzione di probabilità della sua frequenza cardiaca massima (thalach)?

```
query_report(infer,
   variables=['thalach'],
   evidence={
      'age': 3,
      'chol': 2,
      'cp': 3,
      'slope': 0
  }
```

Risultati dell'interrogazione sulla variabile thalach

thalach	Probabilità
thalach (0)	0.2
thalach (1)	0.2
thalach (2)	0.2
thalach (3)	0.2
thalach (4)	0.2

Tabella 5.3: Risultato query sulla Rete Bayesiana

L'output ottenuto dalla query sulla rete bayesiana mostra una distribuzione uniforme delle probabilità per i possibili stati della variabile thalac.

Interpretazione dei Risultati

1. Distribuzione uniforme delle probabilità
    - La rete non sta trovando relazioni forti tra le variabili osservate (age=3, chol=2, cp=3, slope=0) e la variabile target thalach.
    - Questo potrebbe indicare che le evidenze fornite non forniscono abbastanza informazione per discriminare tra i diversi valori di thalach.
2. Possibili cause della distribuzione uniforme
    - Modello non sufficientemente addestrato, infatti se la rete bayesiana non è stata appresa su dati con una forte correlazione tra le variabili, l'inferenza potrebbe risultare meno precisa.
    - Potrebbe essere utile includere altre variabili, come exang (angina da sforzo) o oldpeak (depressione del segmento ST), che potrebbero avere un maggiore impatto sulla frequenza cardiaca massima.
    - Se nel dataset i valori di thalach sono distribuiti in modo uniforme, è possibile che la rete stia semplicemente riflettendo questa distribuzione.

Capitolo 6

Rete Neurale

Nel contesto dell'apprendimento supervisionato, la Regressione Logistica ha prodotto risultati accettabili, ma non ottimali. Questo potrebbe essere dovuto a un alto bias del modello, ovvero alla sua struttura troppo semplice che potrebbe non riuscire a catturare relazioni non lineari presenti nei dati.

Per affrontare questa limitazione, si è deciso di introdurre l'uso delle Reti Neurali Artificiali (ANN), che rappresentano un'estensione della regressione logistica. Le reti neurali consentono di aggiungere strati nascosti e funzioni di attivazione non lineari, con l'obiettivo di apprendere pattern più complessi e migliorare la capacità di generalizzazione del modello.

Le reti neurali offrono inoltre maggiore scalabilità e sono in grado di gestire dataset di grandi dimensioni, caratteristica utile nel contesto della diagnosi delle malattie cardiache.

In questo studio, si è valutata l'applicazione delle reti neurali per la classificazione del rischio di malattie cardiache nei pazienti. L'obiettivo è stato verificare se un'architettura più avanzata possa migliorare le prestazioni rispetto alla regressione logistica, ottenendo un modello più preciso e affidabile per supportare il processo diagnostico

METODOLOGIA 6.1

Per migliorare le prestazioni rispetto alla Regressione Logistica, è stata progettata una Rete Neurale Artificiale (ANN) composta da un layer di input, due hidden layer e un layer di output. L'obiettivo è verificare se, introducendo due hidden layer con funzioni di attivazione non lineari, si possa migliorare la capacità di apprendimento del modello, catturando relazioni più complesse nei dati clinici dei pazienti.

Struttura della Rete Neurale

La rete è stata progettata nel seguente modo:

- Numero di unità nel layer di input: 22, corrispondenti al numero di feature utilizzate nel dataset.
- Funzione di attivazione degli hidden layer: ReLU, che introduce non linearità nel modello e migliora la capacità di apprendimento.
- Funzione di attivazione del layer di output: Sigmoid, poiché il problema è di classificazione binaria (presenza o assenza di malattia cardiaca).
- Funzione di loss: Binary Crossentropy, utilizzata per problemi di classificazione binaria, penalizzando maggiormente le previsioni errate.
- Numero di epoche: 100.
- Batch size: 32, per garantire un equilibrio tra velocità di addestramento e stabilità del modello.

- Learning rate: 0.001, scelto come compromesso tra velocità di convergenza e precisione del modello.

Ricerca degli Iperparametri
Per ottimizzare le prestazioni del modello, sono stati selezionati alcuni iperparametri da esplorare mediante Grid Search in combinazione con Cross Validation:

- Model_hidden_units 1: Numero di unità nel primo hidden layer: [4, 8, 16, 24, 32]
- Model_hidden_units 2: Numero di unità nel secondo hidden layer: [4, 8, 16, 24, 32]
- Model_optimizer: ["adam", "sgd", "rmsprop"]
    - SGD: aggiorna i pesi del modello dopo ogni batch usando il gradiente della funzione di loss.
    - RMSprop: adatta dinamicamente il learning rate per ogni parametro, migliorando la stabilità dell'addestramento.
    - Adam: combina i vantaggi di SGD con Momentum e RMSprop, adattando dinamicamente il learning rate per un'ottimizzazione più efficiente

Implementazione e Valutazione
L'implementazione della rete neurale è stata realizzata utilizzando PyTorch, sfruttando la flessibilità e le potenzialità di questa libreria per l'addestramento e la valutazione del modello di deep learning.
Per la valutazione delle prestazioni del modello, sono state considerate due metriche principali:

1. **Classification Report:**
    - Sono state calcolate le metriche di precision, recall e F1-score per entrambe le classi (diagnosi positiva e negativa), per valutare le performance del modello.
    - Il report è stato salvato in formato CSV per ulteriori analisi.

2. **Curve di Accuracy e Loss:**
    - Per analizzare l'andamento dell'addestramento e della validazione nel corso delle epoche, sono state tracciate le curve di accuracy e loss durante le fasi di training e validation.
    - I grafici sono stati salvati per facilitare il confronto tra diverse configurazioni di iperparametri.

L'analisi dei risultati ottenuti ha permesso di selezionare la configurazione ottimale della rete neurale, garantendo il miglior bilanciamento tra accuratezza e generalizzazione.

PREPARAZIONE DEL DATASET 6.2

Poiché le Reti Neurali utilizzano la Discesa del Gradiente, è importante che le feature siano normalizzate per evitare che variabili con scale molto diverse causino problemi di convergenza.
Il dataset è stato preprocessato e normalizzato affinché tutte le feature abbiano valori compresi tra 0 e 1, garantendo un apprendimento più stabile ed efficiente.

Suddivisione del Dataset
Per garantire una valutazione robusta del modello, il dataset è stato suddiviso nel seguente modo:

- 70% per il training
    - 80% training effettivo (usato per aggiornare i pesi del modello)

    - 20% validation (usato per ottimizzare gli iperparametri e monitorare l'addestramento)
- 30% per il test (valutazione finale del modello su dati mai visti)

Questa suddivisione consente di ottenere una stima affidabile delle prestazioni del modello, evitando problemi di overfitting o scarsa generalizzazione.

Nel contesto della diagnosi delle **malattie cardiache**, l'addestramento della **Rete Neurale** avviene attraverso la **Discesa del Gradiente**, un algoritmo di ottimizzazione che aggiorna i pesi del modello iterativamente per minimizzare l'errore.

Tuttavia, se le **feature** hanno scale molto diverse tra loro, questo può causare problemi nell'aggiornamento dei pesi. Le feature con valori numerici più grandi tenderebbero ad avere **aggiornamenti maggiori**, mentre quelle con valori più piccoli potrebbero essere **trascurate**. Questo può portare a:

- **Oscillazioni nei gradienti**, rendendo l'addestramento instabile.
- **Rallentamento della convergenza**, prolungando il tempo necessario per trovare un modello ottimale.

Per evitare questi problemi, è stata applicata la **normalizzazione delle feature**, che ridimensiona i dati in un intervallo uniforme.

ESPERIMENTO 6.3

Iperparametri trovati con GridSearch

model _hidden_units_1	4
model _hidden_units_2	16
model_optimizer	rmsprop

Tabella 6.1: Iperparametri

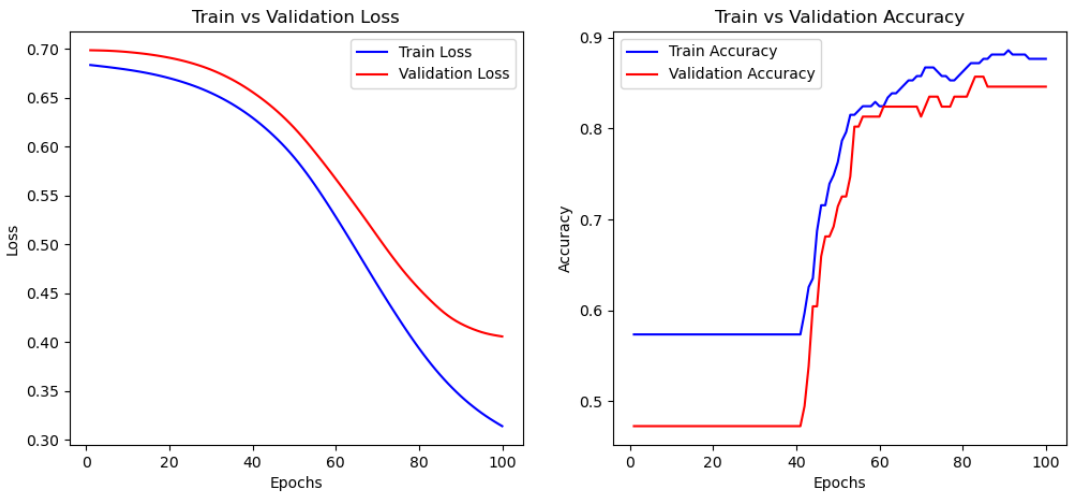


Figure 6.1: Curve dell'accuratezza e della loss al variare delle epoche

📊 Report di classificazione per il Validation Set:

	Precision	Recall	F1 Score	Support
Diagnosi Negativa	0.94	0.85	0.89	20.0
Diagnosi Positiva	0.88	0.96	0.92	23.0
macro avg	0.91	0.90	0.91	43.0
weighted avg	0.91	0.91	0.91	43.0
Accuracy	0.91			

Tabella 6.2: Report di classificazione per il validation set

📊 Report di classificazione per il Test Set:

	Precision	Recall	F1 Score	Support
Diagnosi Negativa	0.83	0.81	0.82	42.0
Diagnosi Positiva	0.84	0.86	0.85	49.0
macro avg	0.83	0.83	0.83	91.0
weighted avg	0.84	0.84	0.84	91.0
Accuracy	0.84			

Tabella 6.3: Report di classificazione per il test set

Considerazioni sui Risultati del Modello

Curva della Loss (Training vs Validation)

- La loss di training e validation diminuisce progressivamente, il che indica che il modello sta apprendendo correttamente.
- Il calo della loss è più marcato fino a circa 60 epoche, dopodiché la curva si stabilizza.
- La loss di training è leggermente inferiore alla loss di validation, il che è normale. Non c'è un grande divario, quindi non sembrano esserci segni evidenti di overfitting.
- La progressione liscia delle curve suggerisce che l'ottimizzazione avviene in modo stabile, senza oscillazioni o instabilità.

Accuratezza e Precision

Validation Set

- Accuracy: 91%
- Diagnosi Negativa: Precision 0.94, Recall 0.85, F1-score 0.89, Support 20
- Diagnosi Positiva: Precision 0.88, Recall 0.96, F1-score 0.92, Support 23
- Macro avg e Weighted avg sono vicine all'accuracy, il modello non dà troppa importanza a una classe rispetto all'altra.

Test Set

- Accuracy: 84%

- Diagnosi Negativa: Precision 0.83, Recall 0.81, F1-score 0.82, Support 42
- Diagnosi Positiva: Precision 0.84, Recall 0.86, F1-score 0.85, Support 49
- Le metriche sono leggermente più basse rispetto al validation set, ma comunque abbastanza buone, segno che il modello generalizza bene.

Considerazioni sulle metriche

- La precisione è simile tra le due classi, il che significa che il modello non favorisce una classe rispetto all'altra.
- Il recall della diagnosi positiva è più alto (0.96 nel validation set, 0.86 nel test set) rispetto a quello della diagnosi negativa. Ciò significa che il modello è più bravo a identificare i pazienti con malattia cardiaca rispetto ai sani.
- L'accuracy rimane alta, quindi il modello è affidabile.

Confronto tra Validation e Test Set

- L'accuracy del validation set (91%) è più alta rispetto a quella del test set (84%), il che potrebbe suggerire un leggero overfitting.
- La differenza tra validation e test non è eccessiva, ma potrebbe essere utile monitorare le prestazioni su nuovi dati.
- Possibili soluzioni per migliorare la generalizzazione:
    - Aumentare i dati di training, se disponibili.
    - Utilizzare tecniche di regolarizzazione (es. Dropout o L2 Regularization).
    - Effettuare data augmentation per migliorare la robustezza del modello.

Sbilanciamento del Dataset

- Il numero di diagnosi positive e negative è abbastanza bilanciato nel dataset, quindi il modello non è sbilanciato verso una delle due classi.
- Tuttavia, la recall della diagnosi negativa è leggermente più bassa rispetto alla recall della diagnosi positiva. Ciò significa che alcuni pazienti sani vengono erroneamente classificati come malati.
- Strategie per migliorare la diagnosi negativa:
    - Bilanciare la funzione di perdita: Assegnare un peso leggermente maggiore agli errori sulle diagnosi negative.
    - Ottimizzare la soglia di classificazione: Provare a variare la soglia di decisione per migliorare il recall della diagnosi negativa.
    - Aggiungere più dati negativi se possibile, per migliorare l'apprendimento del modello su questa classe.

CONCLUSIONI FINALI 6.4

- Nonostante la Rete Neurale possa catturare relazioni più complesse rispetto alla Regressione Logistica, nel caso delle malattie cardiache, i risultati suggeriscono che il modello di Regressione Logistica potrebbe essere più efficiente in termini di costi computazionali e performance, specialmente se ottimizzato per gestire l'sbilanciamento delle classi. La Rete Neurale, pur mostrando un buon livello di accuratezza e precisione, non è riuscita a migliorare significativamente le prestazioni rispetto alla Regressione Logistica.

Capitolo 7

Sviluppi Futuri

I modelli e le tecniche adottate nel caso di studio hanno già dimostrato una buona capacità predittiva, ma in un contesto medico reale, le **performance** diventano un fattore cruciale. In effetti, non basta ottenere un buon risultato, ma si deve puntare all'**ottimizzazione** quanto più possibile, poiché anche minimi errori potrebbero avere conseguenze significative sulla salute dei pazienti. Ad esempio, i **falsi negativi** (pazienti che non sono risultati positivi alla diagnosi ma che in realtà lo sono) potrebbero portare a diagnosi tardive, con il rischio di complicazioni per la salute del paziente. D'altro canto, i **falsi positivi** (pazienti che sono risultati positivi ma che in realtà non lo sono) potrebbero portare a trattamenti non necessari, esponendo il paziente a rischi inutili.

Altri sviluppi futuri potrebbero essere:

Utilizzo di Tecniche di Transfer Learning
Il transfer learning potrebbe essere applicato per migliorare l'efficienza e l'accuratezza del modello. Questo approccio consente di sfruttare modelli pre-addestrati su grandi dataset generali. Ad esempio, un modello pre-addestrato su un ampio dataset di immagini mediche potrebbe essere riutilizzato per affinare le sue capacità sui dati specifici delle malattie cardiache, riducendo significativamente i tempi di addestramento e migliorando le performance predittive.

2. **Implementazione di Modelli di Apprendimento Profondo (Deep Learning) per Dati Temporali**
Se il dataset include sequenze temporali di dati (come misurazioni periodiche della pressione sanguigna, ECG o altri segnali biologici), l'uso di Reti Neurali Ricorrenti (RNN) o Long Short-Term Memory (LSTM) potrebbe essere vantaggioso. Questi modelli sono in grado di catturare e analizzare le dipendenze temporali nei dati, migliorando la capacità del sistema di fare previsioni più precise riguardo a eventi futuri (come attacchi di cuore o aritmie) in base ai dati precedenti.

Capitolo 8

Guida all’uso del codice

I file supervised.py, unsupervised.py, bayesian_network.py, neural_network.py hanno ognuno un main che avviato mostra delle scelte per effettuare ogni passaggio dell’esperimento.

BIBLIOGRAFIA

[1] D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. 3/e. Cambridge University Press