Universitá degli Studi di Salerno

Dipartimento di Informatica
Relazione Artificial Intelligence

HeartGuard
Analisi comparativa di modelli di Machine Learning per la predizione dell'attacco cardiaco

Professori:
Prof.ssa Loredana Caruccio
Prof.ssa Genoveffa Tortora

Candidato
Armando Imbimbo
Matr. NF22700004

Anno Accademico 2025 / 2026

Abstract

Negli ultimi anni si è registrato un aumento dei decessi per attacco cardiaco, esso è uno dei più gravi eventi cardiovascolari. Una diagnosi preventiva può essere fondamentale per la sopravvivenza dei pazienti, ma identificare un attacco cardiaco è molto complesso e richiede l'analisi di molteplici fattori poiché i sintomi possono variare da persona a persona e possono manifestarsi anche in forme atipiche.
Con l'introduzione del Machine Learning, è possibile valutare l'efficacia di diversi modelli di apprendimento automatico nel prevedere un attacco cardiaco, classificando i pazienti in due gruppi attacco cardiaco e non attacco cardiaco. Per valutare i modelli verranno usate le metriche come: Recall, Precision, Accuracy e F1-Score oltre a strumenti di analisi avanzata quali la matrice di confusione e la curva AUC-ROC. L'obiettivo è quello di individuare il miglior modello per la predizione degli attacchi di cuore, confrontando i risultati dei modelli, oltre ad individuare i punti di forza e le limitazioni delle diverse tecniche.

Indice

Capitolo 1

Introduzione

La morte per attacco cardiaco negli ultimi anni ha avuto un aumento a livello globale. L'attacco cardiaco si può rilevare attraverso la presenza di sintomi quali oppressione toracica e/o mancanza di respiro oppure attraverso le analisi della tropanina ed il Ck-Mb. La tropanina è una proteina presente sia nei muscoli scheletrici che in quello cardiaco, livelli elevati di tale proteina mostrano un eventuale attacco cardiaco. Invece il Ck-Mb è un tipo di cheratina presente principalmente nel muscolo cardiaco, essa viene analizzata principalmente per valutare i danni al cuore, livelli alti di Ck-Mb possono essere indicativi di infarto o danni al cuore.
Inoltre come riportato in [1] i fattori di rischio legati all'attacco di cuore sono l'età, il sesso e la familiarità sui quali non si può agire efficacemente ma si possono monitorare per prevenzione. Invece i fattori modificabili sono legati alle abitudini quotidiane le quali possono essere corrette, tali fattori sono lo stile di vita, l'alimentazione, la pressione alta e il diabete.
Gli attacchi di cuore si possono prevenire se si conduce uno stile di vita sano in particolare se si evita il fumo, si svolge attività fisica, si ha un alimentazione sana, si controlla il proprio peso forma e si limita il consumo dell'alcol, questi comportamenti permettono di tenere sotto controllo il fattore di rischio cioè quelle malattie e condizioni che favoriscono l'insorgenza di attacchi di cuore [2].
L'informatica può contribuire alla prevenzione degli attacchi di cuore mediante la loro previsione, resa possibile dall'impiego di metodologie di apprendimento automatico. L'obiettivo del progetto è quello di utilizzare due algoritmi sul dataset "*Medicaldataset*" disponibile su Kaggle, contenente dati relativi agli attacchi di cuore provenienti dall'ospedale Zheen di Erbil, in Iraq, e comprendere quale dei due è il migliore, nello specifico verranno utilizzati il "*Logistic Regression*" ed il "*Decision Tree*". Il documento è composto da 6 capitoli, il secondo capitolo

discuterà dello stato dell'arte, concentrandosi sulla tematica della previsione degli attacchi di cuore con l'utilizzo di metodologie di apprendimento automatico. Il terzo capitolo intitolato "*obiettivo*" verterà in particolare sull'ambito progettuale, sulle librerie usate, sul workflow e sul dataset, per poi concludere con l'obiettivo finale. Nel quarto capitolo verrà discussa la metodologia in particolare l'analisi del dataset e preparazione della variabile target, la rimozione degli outlier con annessi grafici, un'analisi della matrice di correlazione che mostra la differenza con e senza gli outliers e l'implementazione dei due algoritmi di Machine Learning selezionati per raggiungere l'obiettivo del progetto. Nel quinto capitolo verrano mostrati i risultati dei due modelli presi in considerazione. Infine, il sesto capitolo presenterà le conclusioni del progetto, riassumendo i risultati ottenuti.

Capitolo 2

Stato dell'arte

La letteratura scientifica mette a disposizione numerose ricerche riguardanti la tematica della previsione degli attacchi di cuore con l'utilizzo di metodologie di apprendimento automatico. In tale sezione verranno presentati e analizzati due casi di studio che esaminano questa tematica.
Il primo caso di studio [3] analizza il problema dell'aumento del numero di persone che perdono la vita a causa di infarti. In questi casi, il trattamento tempestivo e l'intervento precoce sono fondamentali per aumentare le possibilità di sopravvivenza. Quando viene fornita assistenza medica immediata e viene somministrata una terapia adeguata, le probabilità di sopravvivenza crescono in modo significativo.

Per questo motivo, lo studio ha avuto come obiettivo l'individuazione precoce del rischio di infarto attraverso l'uso di tecniche di Machine Learning. Il dataset utilizzato comprende i dati di 303 pazienti e 14 caratteristiche. I dati sono stati analizzati e addestrati utilizzando due algoritmi il "*Logistic Regression*" e il "*Decision Tree*".

Dai risultati ottenuti emerge che la "*Logistic Regression*" ha raggiunto un'accuratezza dell'83,8%, mentre il "*Decision Tree*" ha raggiunto un'accuratezza del 77%. In base a questi risultati, il modello di "*Logistic Regression*" si è dimostrato più efficace nel predire il rischio di infarto.
Il secondo caso di studio [4] si concentra sull'importanza della previsione dell'infarto tramite tecniche di apprendimento automatico, considerate fondamentali per favorire interventi preventivi e un'assistenza sanitaria più personalizzata. La ricerca ha l'obiettivo di prevedere il rischio di infarto in ambito sanitario utilizzando un'ampia varietà di dati.

Inoltre, vengono presentati diversi modelli di Machine Learning che cercano

di bilanciare accuratezza e facilità di interpretazione dei risultati, ponendo particolare attenzione alla diagnosi precoce e all'intervento tempestivo. Nel complesso, questo approccio interdisciplinare mette in evidenza il ruolo chiave dell'apprendimento automatico nel ridurre l'impatto delle malattie cardiache e nel migliorare l'uso delle risorse dedicate all'assistenza sanitaria.

Gli autori dello studio hanno analizzano l'applicazione di diverse tecniche di Machine Learning per prevedere il rischio di infarto utilizzando dati clinici strutturati. Sono stati scelti vari modelli di classificazione, tra cui "*LogisticRegression*", "*Support Vector Machine*", "*Random Forest*" e "*Decision Tree*", selezionati per la loro efficacia già dimostrata in studi precedenti in ambito sanitario e per il buon compromesso tra accuratezza e interpretabilità.

La metodologia ha previsto una fase approfondita di pre-processing, la gestione dello sbilanciamento delle classi e l'ottimizzazione degli iperparametri, con l'obiettivo di migliorare le prestazioni dei modelli. Per la valutazione sono state utilizzate diverse metriche, tra cui accuratezza, precisione, recall, F1-score e AUC-ROC.

Capitolo 3

Obiettivo

In questo capitolo verrà illustrato l'ambito progettuale cioè la sviluppo di modelli di Machine Learning per la classificazione dell'attacco di cuore con l'uso del dataset "*Medicaldataset*". Inoltre verranno illustrate le librerie python utilizzate come scikit-learn, Pandas, matplotlib, verrà descritto il workflow, la struttura del dataset e la suddivisione del dataset in trainig set e test set. Infine verrà presentato l'obiettivo finale del progetto il quale si focalizza sullo sviluppo di un modello di Machine Learning efficace ed accurato.

3.1 Ambito del progetto

Il progetto si concentra sullo sviluppo e la valutazione dei modelli di Machine Learning per la classificazione dell'attacco di cuore, utilizzando il dataset "*Medicaldataset*" disponibile su Kaggle. Il dataset contiene dati sugli attacchi cardiaci raccolti presso l'ospedale Zheen di Erbil, in Iraq, da gennaio a maggio 2019.

3.2 Librerie Python

Per la realizzazione del progetto è stato utilizzato il linguaggio di programmazione Python, supportato da librerie specifiche utilizzate nel campo del Machine Learning. Nello specifico sono state utlizzate le seguenti librerie mostrate nella figura 3.1:

- "*scikit-learn*": utilizata per la creazione e la valutazione dei modelli di Machine Learning.

- "*Pandas e Numpy*": le due librerie sono state utilizzate per la manipolazione e l'analisi del dataset.
- "*Seaborn e matplotlib*": entrambe le librerie sono state utilizzate per la visualizzazione grafica delle analisi.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import (accuracy_score, f1_score, precision_score, recall_score
, confusion_matrix, ConfusionMatrixDisplay, roc_curve, auc)
import warnings
import numpy as np
from sklearn.preprocessing import StandardScaler
```

Figura 3.1: Librerie Python utilizzate

3.3 Workflow

Il progetto si sviluppa nelle seguenti fasi:

- Pre-elaborazione dei dati: come prima fase è stata effettuata un'analisi esplorativa del dataset, la preparazione della variabile target e la rimozione degli outlier. Tali punti verranno discussi nel prossimo capitolo.
- Suddivisione del dataset: la seconda fase è la suddivisione del dataset con 75% training set e 25% test set. Tale suddivisione ci ha permesso di ottenere un buon equilibrio tra l'addestramento del modello e la valutazione delle sue prestazioni sui dati non visti.
- Feature Scaling: la terza fase è lo "*StandardScaler*" anch'esso rientrante nella pre-elaborazione dei dati, ci ha permesso di standardizzare le feature. Esso viene effettuato solo sul train split poiché se venisse effettuato sull'intero dataset includendo anche il test set, introdurrebbe informazioni provenienti dai dati di test durante la fase di addestramento, causando una stima eccessivamente ottimistica delle prestazioni del modello.

- Iperparametri: nella quarta fase per entrambi i modelli sono stati ottimizzati gli iperparametri con l'utilizzo di "*RandomizedSearchCV*".
- Valutazione del modello: nell'ultima fase per entrambi i modelli sono state calcolate le metriche standard: Recall, Precision, Accuracy, F1-Score e l'AUC-ROC. Inoltre per comprendere la distribuzione degli errori sono state anilizzate le confusion matrix.

3.4 Descrizione del dataset

Il dataset utilizzato è "*Medicaldataset*" [5] presente su kaggle, esso contiene un set di dati sugli infarti i quali sono stati raccolti presso l'ospedale Zheen di Erbil, in Iraq, da gennaio a maggio 2019. Gli attributi di questo dataset sono: età, sesso, frequenza cardiaca, pressione arteriosa sistolica, pressione arteriosa diastolica, glicemia, ck-mb, troponina e result che può essere negativo o positivo inoltre esso è composto da 9 colonne e 1319 istanze, nella tabella 3.1 sottostante sono illustrate nel dettaglio le colonne del dataset.

Age	L'età del paziente.
Gender	1 maschio, 0 femmina.
Heart Rate	Numero di battiti cardiaci al minuto.
Systolic Blood Pressure	Pressione nelle arterie quando il cuore si contrae.
Diastolic Blood Pressure	Pressione nelle arterie tra un battito cardiaco e l'altro.
Blood Sugar	Livello di glucosio nel sangue del paziente.
Ck-mb	Enzima cardiaco rilasciato durante il danno al muscolo cardiaco.
Troponin	Biomarcatore proteico altamente specifico per le lesioni del muscolo cardiaco.
Result	Etichetta dell'esito che indica se il paziente ha avuto o meno un infarto.

Tabella 3.1: Colonne del dataset "*Medicaldataset*"

Inoltre il dataset come mostrato nella figura 3.2 sottostante ha la seguente distribuzione: 61,49% di pazienti positivi all'attacco di cuore e 38.59% negativi all'attacco di cuore.

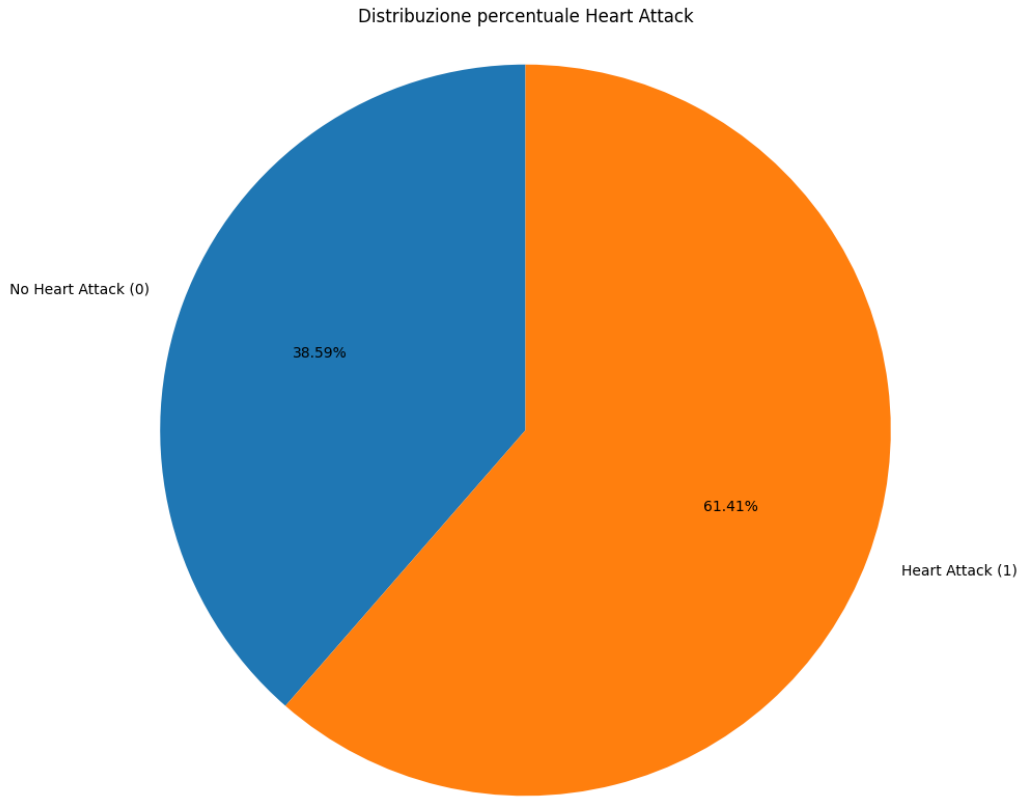

Figura 3.2: Distribuzione percentuale attacco di cuore

Tale dataset è stato scelto poichè affronta un tema delicato e di grande rilevanza negli ultimi anni, caratterizzato da un aumento significativo della mortalità. La possibilità di prevedere un attacco cardiaco può contribuire in modo concreto alla riduzione del numero di decessi. Inoltre il datset ci permmette di effettuare un'a-

nalisi statitisca significativa per il traning dei modelli grazie alla sua completezza e all'origine dei dati provenienti da cartelle cliniche.

3.5 Suddivisione del dataset

Il dataset è stato suddiviso in due parti: il 75% per il training set ed il 25% per il test set. La quantità di dati del training set ha permesso ai modelli di apprendere le relazioni tra le variabili indipendenti e la variabile target. Il test set è stato usato per valutare le prestazioni dei modelli su dati non visti durante l'addestramento.

La suddivisione 75/25 ha garantito sia una quantita adeguata di dati per addestrare i modelli che un numero adeguato di dati per testare le loro prestazioni. Tale suddivisione ci permette di evitare sia overfitting che underfitting. L'overfitting si verifica quando un modello si adatta troppo ai dati di addestramento, dando luogo ad un modello non in grado di effettuare previsioni accurate su dati diversi da quelli di addestramento. Invece l'underfitting si verifica quando un modello è troppo semplice, il che può essere il risultato di un modello che ha bisogno di più tempo di addestramento oppure di più caratteristiche di input o di una regolarizzazione minore.

Nella figura 3.3 viene mostrato il codice per la suddivisione del dataset. Il primo passo è stato l'eliminazione della variabile target "*Result*" per la creazione del set features (X), il passaggio successivo consiste nell'estrazione della variabile "*Result*" come vettore delle etichette (y). Grazie all'utilizzo della funzione train_test_split si è suddiviso il dataset nel seguente modo: training set 75% e test set 25%.

```
X = ds.drop( labels: 'Result', axis=1)
y = ds['Result']

X_train, X_test, y_train, y_test = train_test_split(
    *arrays: X, y, test_size=0.25, random_state=42, stratify=y
)
```

Figura 3.3: Suddivisione del dataset

3.6 Obiettivo finale del progetto

L'obiettivo principale del progetto è quello di sviluppare un modello di machine learning che abbia:

- Una buona accuratezza nel classificare gli attacchi di cuore tra i pazienti.
- Una robustezza nel garantire delle ottime performance su dati acquisiti dal mondo reale.
- Una buona rilevanza clinica riducendo al minimo i falsi negativi, cioè i casi in cui un paziente con attacco di cuore viene erroneamente classificato come paziente che non ha un attacco di cuore.

Per compredere quale modello tra i due è il migliore ed ha le caratteristiche citate in precedenza verranno usate le metriche come: Recall, Precision, Accuracy e F1-Score oltre a strumenti di analisi avanzata quali la matrice di confusione e la curva AUC-ROC.

Capitolo 4

Metodologia

In questa sezione viene mostrata la metodologia utilizzata in particolare, l'analisi del dataset e preparazione della variabile target, la rimozione degli outlier con annessi grafici, un'analisi della matrice di correlazione che mostra la differenza con e senza gli outliers e l'implementazione delle tecniche utilizzate nel progetto.

4.1 Analisi del dataset e preparazione della variabile target

Pulizia dei dati Per ottenere un quadro generale del dataset, sono state estratte informazioni sulla qualità dei dati, verificando la presenza di valori mancanti e duplicati con l'eventuale rimozione degli stessi.

```
RangeIndex: 1319 entries, 0 to 1318
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1319 non-null   int64
 1   Gender                    1319 non-null   int64
 2   Heart rate                1319 non-null   int64
 3   Systolic blood pressure   1319 non-null   int64
 4   Diastolic blood pressure  1319 non-null   int64
 5   Blood sugar               1319 non-null   float64
 6   CK-MB                     1319 non-null   float64
 7   Troponin                  1319 non-null   float64
 8   Result                    1319 non-null   object
dtypes: float64(3), int64(5), object(1)
```

Figura 4.1: Informazioni Dataset

```
🔍 Missing Values Check:
No missing values found! ✅
🔍 Duplicate Check:
No duplicates found! ✅
```

Figura 4.2: Check valori nulli e duplicati

L'immagine 4.1 mostra il numero di valori non nulli per ciascuna colonna, il numero di colonne e di entries e il tipo di dati presenti nel dataset. Inoltre nella figura 4.2 viene mostrato con una maggiore chiarezza l'assenza di valori nulli e duplicati nel dataset.

Codifica della variabile target (Mapping) La variabile target "*Result*", inizialmente categorica e con due possibili valori positivo o negativo, è stata mappata in valori numerici secondo il codice riportato nella figura 4.3 sottostante. Tale trasformazione è essenziale per la fase di modellazione.

```
d = {'positive': 1, 'negative': 0} # Mappatura target
ds['Result'] = ds['Result'].map(d)
```

Figura 4.3: Mappatura target

Distribuzione delle classi Per comprendere meglio la composizione del dataset, è stata analizzata la distribuzione dei pazienti con e senza attacco di cuore, sia in termini numerici che percentuali, con il supporto di un grafico 3.2 discusso nel capitolo precedente.

4.2 Identificazione e rimozione degli outliers

Prima della rimozione degli outliers sono stati generati dei grafici che mostrano come i dati sono strutturati all'interno delle colonne. Nei grafici 4.4 sottostanti si può notare che ci sono diversi valori fuori dal range previsto, tali dati assumono valori superiori alla media o errati. Per evitare che questi valori distorcano le analisi successive, si è quindi proceduto alla rimozione degli outlier.

Per identificare e rimuovere gli outliers si è utilizzato il metodo Interquartile, che si basa sul calcolo del primo quartile (Q1) e del terzo quartile (Q3). L'Interquartile (IQR) è definito come la differenza tra il terzo e il primo quartile, quindi IQR = Q3 – Q1. Le soglie per individuare gli outlier sono state fissate come 1,5 volte l'IQR al di sotto di Q1 e al di sopra di Q3. Tutti i valori inferiori alla soglia inferiore o superiori alla soglia superiore sono considerati outlier e possono essere rimossi dai dati.

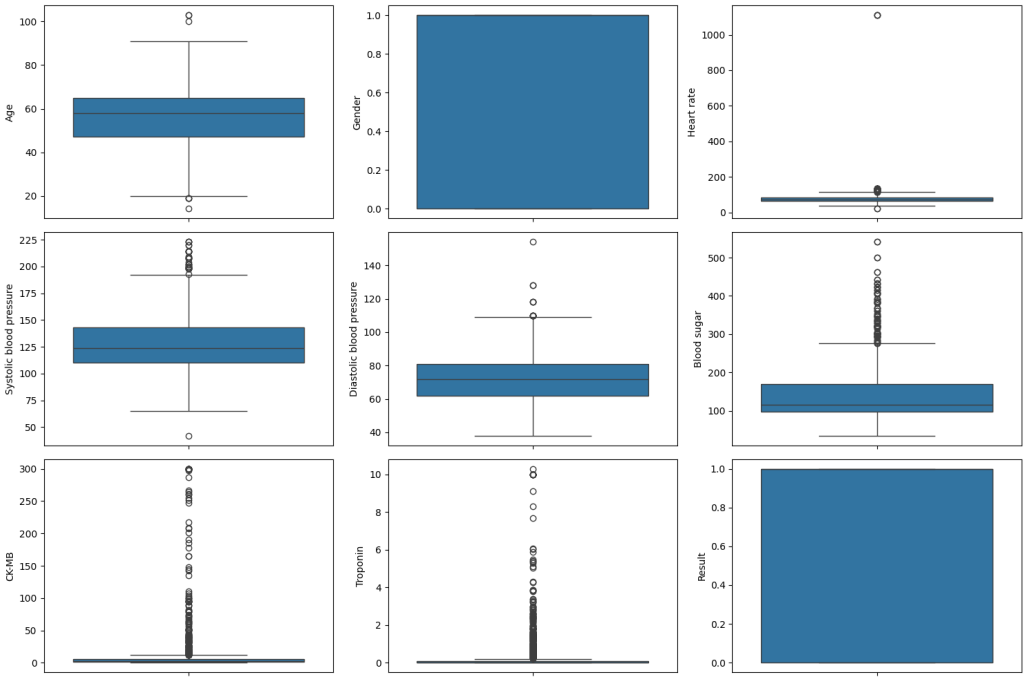

Figura 4.4: Grafici degli outliers

4.3 Analisi della matrice di correlazione

La matrice di correlazione è una matrice quadrata che riporta i coefficienti di correlazione tra tutte le coppie di variabili, permettendo di analizzarne le relazioni lineari, essa è compresa sempre fra 1 e -1.

- -1 significa che le due variabili hanno una relazione inversa, cioè all'aumentare di una diminuisce l'altra.
- 1 significa che le due variabili hanno una relazione diretta, cioè che al'aumentare di una aumenta anche l'altra.
- 0 significa che non si può stabilire un andamento lineare tra le due variabili.

Nella figura 4.5 sottostante viene mostrata la matrice di correlazione prima della rimozione degli outlier, la quale riporta i coefficienti di correlazione tra tutte le coppie di variabili. Inoltre per ottenere una visuale più chiara è stato creato un secondo grafico 4.6 che mostra le feature più correlate con l'attacco di cuore.

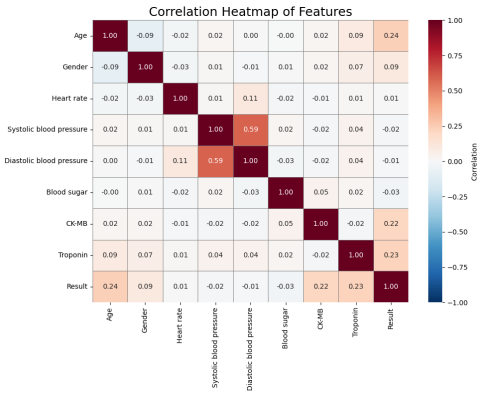

Figura 4.5: Matrice di correlazione

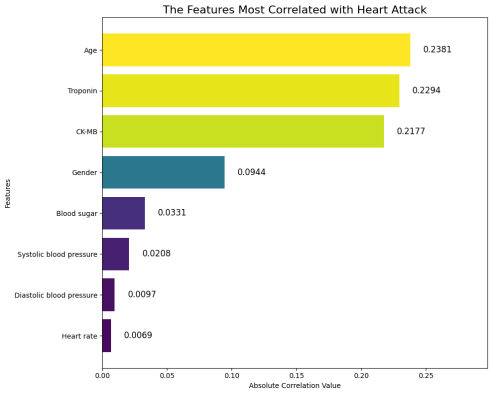

Figura 4.6: Grafico delle feature più correlate con l'attacco di cuore

Come mostrato in figura 4.6 le features maggiormente correlate con l'attaco di cuore sono age, tropanin e ck-mb.

Invece nel grafico 4.7 sottostante possiamo vedere come con la rimozione degli outliers abbiamo una rimozione del rumore che ha portato ad una aumento dei coefficienti di correlazione tra le coppie di variabili. Inoltre il secondo grafico 4.8 che mostra le feature più correlate con l'attacco di cuore fa comprendere maggiormente tale aumento.

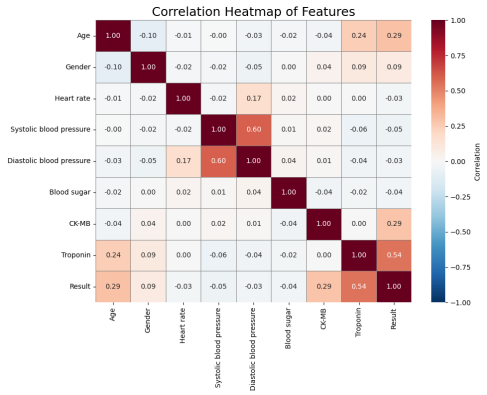

Figura 4.7: Matrice di correlazione

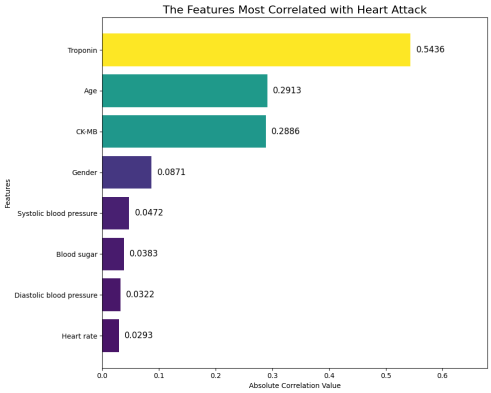

Figura 4.8: Grafico delle feature più correlate con l'attacco di cuore

4.4 Feature Scaling

Il Feature Scaling noto anche come normalizzazione Z-score è un passaggio fondamentale di pre-elabolazione per molti algoritmi di apprendimento automatico. Esso consiste nel ridimensionare le variabili in modo che presentino media pari a zero e deviazione standard pari a uno. Sebbene i modelli basati su alberi decisionali risultino poco sensibili alla scala delle feature, molti altri algoritmi necessitano di dati normalizzati. In particolare, la normalizzazione aiuta alcuni modelli a convergere più facilmente, come nel caso della regressione logistica. Nella figura 4.9 sottostante viene mostrato il codice per il Feature Scaling.

```
#selezione delle feature numeriche
numeric_features = ['Age', 'Gender', 'Heart rate', 'Systolic blood pressure',
                    'Diastolic blood pressure', 'Blood sugar', 'CK-MB', 'Troponin']
scaler = StandardScaler() #inizailizza scaler

scaler.fit(X_train[numeric_features])

X_train_scaled = scaler.transform(X_train[numeric_features])
X_test_scaled = scaler.transform(X_test[numeric_features])
```

Figura 4.9: Feature Scaling

4.5 Implementazione

In tale sezione verrà illustrata l'implementazione dei due algoritmi di Machine Learning selezionati per raggiungere l'obiettivo del progetto.

4.5.1 RandomizedSearchCV

"*RandomizedSearchCV*" è una tecnica di ottimizzazione degli iperparametri in scikit-learn. Un vantaggio di "*RandomizedSearchCV*" rispetto a "*GridSearchCV*" è la maggiore efficienza quando lo spazio di ricerca degli iperparametri è molto esteso. Infatti, invece di valutare tutte le possibili combinazioni, "*RandomizedSearchCV*" ne esamina solo un numero limitato scelto casualmente. Questo approccio risulta particolarmente utile quando il modello richiede elevati costi computazionali o quando gli iperparametri assumono valori continui, rendendo poco praticabile un'esplorazione completa tramite GridSearchCV. Inoltre, "*RandomizedSearchCV*" può ridurre il rischio di overfitting. Un'analisi esaustiva dello spazio degli iperparametri, come quella effettuata da "*GridSearchCV*", può portare a individuare configurazioni molto performanti sui dati di training ma meno efficaci su dati nuovi. Al contrario, il campionamento casuale adottato da "*RandomizedSearchCV*" favorisce una ricerca meno rigida e può contribuire a una migliore capacità di generalizzazione del modello.
Nel contesto del progetto l'utillizzo di "*RandomizedSearchCV*" rappresenta una scelta adeguata in quanto consente di esplorare in modo efficiente un ampio spazio di iperparametri, mantenendo un buon compromesso tra costi computazionali e qualità dei risultati. Considerate le dimensioni contenute del dataset, questo approccio permette di ottimizzare i modelli in tempi ridotti, garantendo al contempo buone capacità di generalizzazione. Di seguito viene illustrato l'utilizzo di "*RandomizedSearchCV*" all'interno dei modelli utilizzati.
Per il modello "*Logistic Regression*" sono stati presi in input i seguenti parametri:

- penalty: definisce il tipo di regolarizzazione da usare, la regolarizzazione "*l1*" (Lasso) tende ad annullare alcuni coefficienti, invece la regolarizzazione "*l2*" (Ridge) riduce la grandezza dei coefficienti senza annullarli completamente.

- solver: determina l'algoritmo di ottimizzazione utilizzato per la regressione logistica. "*Saga*" supporta diverse forme di regolarizzazione, inclusa L1 e L2 mentre "*Liblinear*" è efficiente per dataset di piccole e medie dimensioni e problemi di classificazione binaria, inoltre supporta L1 e L2.

- C: controlla la forza della regolarizzazione inversa. I valori bassi di C implicano una regolarizzazione più forte, mentre i valori alti di C riducono l'effetto della regolarizzazione.

Inoltre vengono definiti:

- estimator: Il modello Logistic Regression da ottimizzare.
- param_distributions: parametri da testare definiti in precedenza.
- scoring="*accuracy*": utilizza l'accuratezza per valutare i modelli.
- cv=10: si utilizza la cross-validazione con 10 fold per una stima più robusta.
- random_state=42: al fine di garantire la riproducibilità della ricerca casuale.

Invece per il modello "*Decision Tree*" sono stati presi in input i seguenti parametri:

- max_depth: definisce la profondità massima consentita per l'albero decisionale, ovvero il numero massimo di livelli di suddivisione.
- min_samples_split: indica il numero minimo di campioni richiesti affinché un nodo interno possa essere ulteriormente suddiviso.
- min_samples_leaf: stabilisce il numero minimo di campioni che devono essere presenti in un nodo foglia.
- max_features: definisce il numero di feature da considerare durante la ricerca della migliore suddivisione in ogni nodo. L'opzione "*sqrt*" indica che viene utilizzata la radice quadrata del numero totale di feature.
- criterion: specifica la funzione utilizzata per valutare la qualità di una suddivisione (Gini Impurity o entropy).

Inoltre vengono definiti:

- estimator: Il modello Decision Tree da ottimizzare.
- param_distributions: parametri da testare definiti in precedenza.
- n_iter indica il numero di combinazioni casuali di iperparametri da valutare.
- scoring="*accuracy*": utilizza l'accuratezza per valutare i modelli.
- cv=10: si utilizza la cross-validazione con 10 fold per una stima più robusta.
- random_state=42: al fine di garantire la riproducibilità della ricerca casuale.

4.5.2 Logistic Regression

La "*Logistic Regression*" è un algoritmo di apprendimento supervisionato ampiamente utilizzato nella data science. Appartiene alla categoria dei metodi di classificazione e viene impiegata per prevedere un output discreto o categorico. Inoltre la "*Logistic Regression*" viene utilizzata per problemi di previsione e classificazione come ad esempio per la previsione di malattie, infatti in ambito medico, questo approccio di analytics può essere impiegato per stimare la probabilità di insorgenza di patologie o malattie all'interno di una determinata popolazione. Le organizzazioni sanitarie possono così pianificare interventi e strategie di prevenzione mirate per i soggetti che presentano una maggiore predisposizione a specifiche condizioni patologiche come riportato in [6].
La "*Logistic Regression*" è stata scelta poiché, oltre al suo utilizzo in ambito medico, presenta i seguenti vantaggi:

- Interpretabilità: i coefficienti del modello sono facilmente interpretabili, permettendo una chiara comprensione dell'influenza delle variabili esplicative sul risultato.
- Efficienza computazionale: rispetto ad altri modelli di classificazione, la regressione logistica richiede un costo computazionale ridotto, consentendo tempi di addestramento e di utilizzo contenuti.
- Versatilità: il modello è in grado di gestire variabili di risposta binarie, multinomiali e ordinali, rendendolo applicabile a un'ampia varietà di contesti.

Tuttavia, la "*Logistic Regression*" presenta anche alcune limitazioni: in particolare, richiede una fase di pre-processing dei dati più accurata, che include operazioni come la rimozione degli outlier e l'applicazione di tecniche di feature scaling. Infatti nel prossimo capitolo verranno mostrare le metriche con e senza l'utilizzo di "*StandardScaler*"

Di seguito è riportato nella figura 4.10 il codice utilizzato per l'addestramento del modello di Logistic Regression con dati scalati. In questa fase viene applicato RandomizedSearchCV sui dati di training precedentemente normalizzati, al fine di individuare la combinazione ottimale degli iperparametri. Al termine del processo, viene selezionato il modello migliore, già addestrato, che viene successivamente utilizzato per effettuare le predizioni. Vengono quindi calcolate e stampate le metriche, insieme ai migliori parametri individuati. Infine, viene tracciata la curva AUC-ROC, al fine di valutare le prestazioni complessive del modello.

```
random_search_lr_scaled.fit(X_train_scaled, y_train)
best_logreg_lr_scaled = random_search_lr_scaled.best_estimator_

# Predizioni train e test con scaled
y_train_pred_log_scaled = best_logreg_lr_scaled.predict(X_train_scaled)
y_test_pred_log_scaled = best_logreg_lr_scaled.predict(X_test_scaled)

print("\n=== Logistic Regression ===")
print("Migliori parametri trovati:", random_search_lr_scaled.best_params_)
print_metrics(y_train, y_train_pred_log_scaled, dataset_name: "Train")
print_metrics(y_test, y_test_pred_log_scaled, dataset_name: "Test")
plot_roc_curve(y_test, model_name: "Logistic Regression (Scaled)",
               y_score=best_logreg_lr_scaled.predict_proba(X_test_scaled)[:,1])
```

Figura 4.10: Codice per la Logistic Regression

4.5.3 Decision Tree

Il "*Decision Tree*" è un algoritmo di apprendimento supervisionato di tipo non parametrico, impiegato sia nei problemi di classificazione sia in quelli di regressione. La sua rappresentazione assume una struttura gerarchica ad albero, composta da un nodo radice, nodi intermedi che rappresentano le decisioni, rami che collegano i nodi e nodi foglia che forniscono l'output finale del modello. Il "*Decision Tree*" ha la seguente struttura:

- Nodo radice: rappresenta il nodo di partenza.
- Nodi interni: rappresentano le scelte decisionali o le suddivisioni basate sui valori delle caratteristiche di input.
- Nodi foglia: rappresentano i valori previsti o le classi.

Esso è stato scelto per le seguiti motivazioni:

- Facilità di interpretazione: gli alberi decisionali sono algoritmi intuitivi e facilmente interpretabili, in quanto possono essere rappresentati graficamente. Infatti ogni decisione o previsione può essere ricostruita seguendo il percorso definito all'interno dell'albero.
- Trattamento di diversi tipi di dati: è in grado di trattare sia caratteristiche di tipo categorico sia continuo, rendendolo flessibile rispetto alla natura dei dati in ingresso.

- Velocità: un ulteriore aspetto rilevante è rappresentato dall'elevata rapidità degli alberi decisionali, sia durante la fase di addestramento del modello sia nella fase di inferenza.
- Ridotta o assente fase di preparazione dei dati: gli alberi decisionali richiedono meno pre-processing dei dati ad esempio non richiedono l'applicazione di tecniche di scaling delle caratteristiche, poiché non sono sensibili alla varianza dei dati.

Di seguito è riportato nella figura 4.11 il codice utilizzato per il "*Decions Tree*". In questa fase viene eseguita la ricerca dei migliori iperparametri tramite "*RandomizedSearchCV*", durante la quale il modello viene addestrato sui dati di training. Al termine della ricerca, viene selezionato il modello ottimale, che risulta già addestrato. Tale modello viene quindi utilizzato per effettuare il calcolo delle metriche. Vengono inoltre stampati i migliori parametri individuati e, infine, viene tracciata la curva AUC-ROC.

```
#random_search_dt.fit(X_train_scaled, y_train)
random_search_dt.fit(X_train, y_train) #addestra il modello
best_dt=random_search_dt.best_estimator_



# Predizioni
#y_train_pred_dt = best_dt.predict(X_train_scaled) #non ha necessità di essere scalato
#y_test_pred_dt = best_dt.predict(X_test_scaled)
y_train_pred_dt = best_dt.predict(X_train)
y_test_pred_dt = best_dt.predict(X_test)

print("\n=== Decision Tree ===")
print("Best parameters found:", random_search_dt.best_params_)
print_metrics(y_train, y_train_pred_dt, dataset_name: "Train")
print_metrics(y_test, y_test_pred_dt, dataset_name: "Test")
plot_roc_curve(y_test, model_name: "Decision Tree", y_score=best_dt.predict_proba(X_test)[:,1])
```

Figura 4.11: Codice per il Decision Tree

Capitolo 5

Valutazione Sperimentale

5.1 Indicatori di Performance del Modello

In tale sezione verrano mostrati i risultati dei due modelli di Machine Learning utilizzati, analizzando le prestazioni sia sul training set che sul test set.

Confusion Matrix La matrice di confusione riassume le istanze di test in base ai valori veri e predetti, presentandoli sotto forma di tabella di contingenza.

- True Positive (TP): indica il numero di casi di previsione corretta di una classe positiva.
- True Negative (TN): indica il numero di casi in cui il modello ha correttamente predetto una classe negativa.
- False Positive (FP): indica il numero di casi in cui il modello ha erroneamente predetto una classe positiva, quando in realtà era negativa.
- False Negative (FN): indica il numero di casi in cui il modello ha erroneamente predetto una classe negativa quando in realtà era positiva.

Metriche standard Le metriche di performance del machine learning vengono utilizzate per valutare e misurare l'efficacia di un modello predittivo. La scelta delle metriche è fondamentale, poiché permette di capire quanto il modello sia in grado di risolvere una specifica problematica e di confrontare in modo oggettivo approcci diversi. Sebbene ogni modello di machine learning possa richiedere

metriche specifiche, esistono numerosi indicatori di performance comuni, ognuno dei quali offre una diversa prospettiva sulle prestazioni complessive del modello.

- Recall: misura la capacità di un modello di identificare correttamente i casi positivi, quantificando il numero di veri positivi. È noto anche come tasso di sensibilità o tasso di veri positivi (True Positive Rate, TPR). Un modello di machine learning con un valore di recall elevato riesce a individuare la maggior parte dei casi positivi, riducendo al minimo i falsi negativi, ovvero i casi positivi classificati erroneamente come negativi.

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

- Precision: indica la percentuale di predizioni corrette di classe positiva rispetto al totale delle predizioni di classe positiva. Un modello di machine learning con un'elevata precisione è in grado di ridurre il numero di falsi positivi, ovvero quei casi negativi che vengono erroneamente classificati come positivi.

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

- Accuracy: indica il numero di volte in cui il modello ha effettuato una classificazione corretta rispetto al numero totale di occorrenze da classificare.

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- F1 score: è la media F1 è la media armonica di recall e precision, che fonde entrambe le metriche in un unico valore.

$$\textbf{F1 score} = 2 * \frac{precision * recall}{precision + recall}$$

AUC-ROC curve L'Area under the curve (AUC) del receiver operating characteristic (ROC) è una misura consolidata utilizzata nella classificazione binaria. La curva ROC rappresenta graficamente il rapporto tra il tasso di veri positivi e il tasso di falsi positivi, calcolati a diverse soglie di probabilità comprese tra 0 e 1.

- Il True Positive Rate (TPR): noto anche come Recall, è definito come il rapporto tra il numero di veri positivi (TP) e la somma dei veri positivi e dei falsi negativi.
- Il False Positive Rate (FPR): è invece calcolato come il rapporto tra il numero di falsi positivi (FP) e la somma dei falsi positivi e dei veri negativi (TN).

5.2 Logistic Regression

Il modello di "*Logistic Regression*" è stato valutato in due configurazioni differenti: senza l'utilizzo dello StandardScaler e con l'utilizzo dello StandardScaler.

Nel primo caso i migliori iperparametri individuati sono stati solver = "*liblinear*", penalty = "*l2*" e C = 8. Le metriche ottenute mostrano prestazioni discrete, con un'accuratezza pari al 0.7750 sul training set e al 0.7563 sul test set come mostrato nelle tabelle 5.1 5.2.

Nel secondo caso è stata invece applicata la standardizzazione delle feature tramite StandardScaler. In questo caso, i migliori parametri trovati sono solver = "*saga*", penalty = "*l1*" e C = 8. I risultati mostrano un miglioramento significativo delle prestazioni rispetto al modello non scalato. L'accuratezza raggiunge lo 0.9306 sul training set e lo 0.9289 sul test set, con valori elevati anche per precisione, recall e F1-score come mostrato nelle tabelle 5.3 5.4.

Nel complesso, il confronto evidenzia come l'utilizzo dello StandardScaler abbia un impatto rilevante sulle prestazioni della Logistic Regression, migliorando la generalizzazione del modello e rendendo le metriche di valutazione più stabili ed elevate.

Modello	**Accuracy**	**Precision**	**Recall**	**F1 score**
Logistic Regression	0.7750	0.7422	0.7393	0.7407

Tabella 5.1: Metriche dei dati di training sul modello Logistic Regression non scalato

Modello	**Accuracy**	**Precision**	**Recall**	**F1 score**
Logistic Regression	0.7563	0.7342	0.6824	0.7073

Tabella 5.2: Metriche dei dati di test sul modello Logistic Regression non scalato

Modello	**Accuracy**	**Precision**	**Recall**	**F1 score**
Logistic Regression	0.9306	0.9286	0.9105	0.9194

Tabella 5.3: Metriche dei dati di training sul modello Logistic Regression scalato

Modello	**Accuracy**	**Precision**	**Recall**	**F1 score**
Logistic Regression	0.9289	0.8989	0.9412	0.9195

Tabella 5.4: Metriche dei dati di test sul modello Logistic Regression scalato

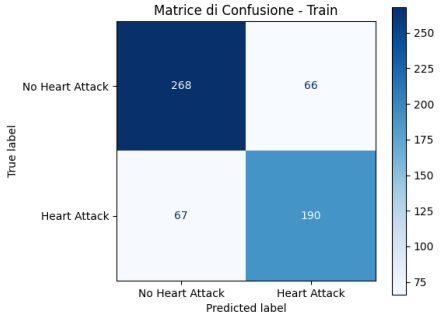

Figura 5.1: Confusion Matrix sui dati di Training non scalati

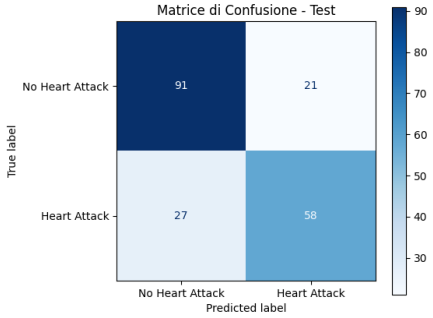

Figura 5.2: Confusion Matrix sui dati di Test non scalati

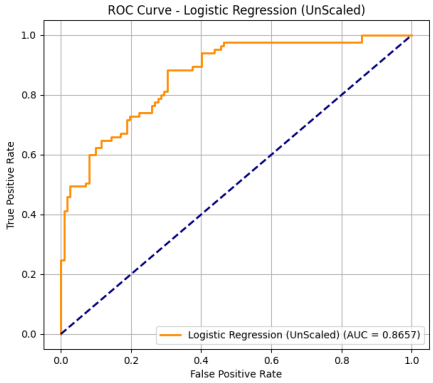

Figura 5.3: AUC-ROC curve del modello Logistic Regression non scalato

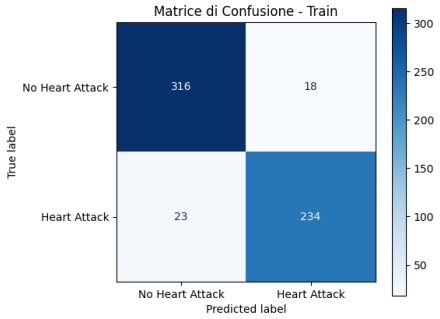

Figura 5.4: Confusion Matrix sui dati di Training scalati

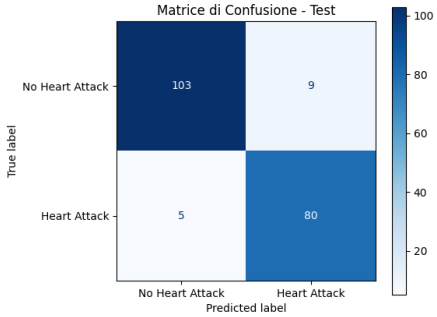

Figura 5.5: Confusion Matrix sui dati di Test scalati

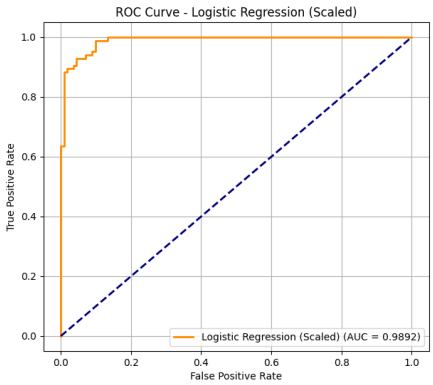

Figura 5.6: AUC-ROC curve del modello Logistic Regression scalato

Nelle figure 5.1,5.2 vengono mostrate le matrici di confusione del training set e del test set del modello non scalato. La matrice di confusione sul training set mostra che il modello ha identificato correttamente 268 casi negativi e 190 casi positivi, mentre i falsi negativi e falsi positivi ammontano rispettivamente a 67 e 66. Sul test set, il modello ha correttamente predetto 91 veri negativi e 58 veri positivi, con 27 falsi negativi e 21 falsi positivi. Inoltre, nella figura 5.3 è riportata la ROC curve del modello sul test set, con un valore di AUC pari a 0,8657, che indica una buona capacità discriminativa del modello nel distinguere tra le classi.

Invece nelle figure 5.4, 5.5 vengono mostrate le matrici di confusione del training set e del test set del modello scalato. La matrice di confusione sul training set mostra che il modello ha identificato correttamente 316 casi negativi e 234 casi positivi, mentre i falsi negativi e falsi positivi ammontano rispettivamente a 23 e 18. Sul test set, il modello ha predetto correttamente 103 veri negativi e 80 veri positivi, con 5 falsi negativi e 9 falsi positivi. Inoltre, nella Figura 5.6 è riportata la ROC curve del modello sul test set, con un AUC pari a 0,9892, indicante un'ottima capacità discriminativa tra le classi.

In conclusione si osserva che l'applicazione dello StandardScaler ha portato ad un significativo miglioramento delle prestazioni. Il modello scalato mostra una maggiore accuratezza e una riduzione dei falsi positivi e falsi negativi sia sul training che sul test set. L'incremento dell'AUC ROC da 0,8657 a 0,9892 conferma una migliore capacità discriminativa e una generalizzazione più solida sulle nuove osservazioni.

5.3 Decision Tree

Per il Decision Tree sono stati individuati come migliori iperparametri: min_samp les_split = 15, min_samples_leaf = 1, max_features = "*sqrt*", max_depth = 11, criterion = "*gini*". Le metriche ottenute sul training set mostrano un'accuratezza molto elevata pari a 0.9797 e un F1 score di 0.9764. Sul test set, le prestazioni rimangono alte, con accuratezza del 0.9746 ed F1 score del 0.9704 come riportato nelle tabelle 5.5, 5.6. Questi risultati indicano che il modello riesce a generalizzare bene sui dati di test, mostrando elevate capacità predittive e un buon equilibrio tra precisione e recall.

Modello	Accuracy	Precision	Recall	F1 score
Decision Tree	0.9797	0.9880	0.9650	0.9764

Tabella 5.5: Metriche dei dati di training sul modello Decision Tree

Modello	Accuracy	Precision	Recall	F1 score
Decision Tree	0.9746	0.9762	0.9647	0.9704

Tabella 5.6: Metriche dei dati di test sul modello Decision Tree

Nelle figure 5.7, 5.8 vengono mostrate le matrici di confusione del training set e del test set del modello. La matrice di confusione sul training set mostra che il Decision Tree ha identificato correttamente 331 casi negativi e 248 casi positivi, mentre i falsi negativi e falsi positivi ammontano rispettivamente a 9 e 3. Sul test set, il modello ha predetto correttamente 110 veri negativi e 82 veri positivi, con 3 falsi negativi e 2 falsi positivi. Inoltre, la ROC curve del modello sul test set 5.9 mostra un valore di AUC pari a 0,9893, confermando un'eccellente capacità discriminativa tra le classi.

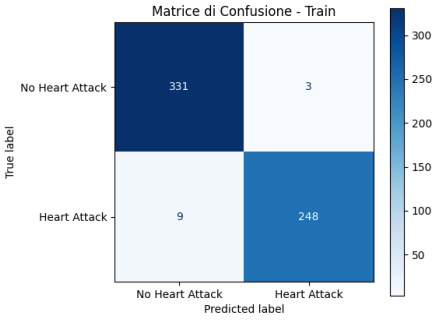

Figura 5.7: Confusion Matrix sui dati di Training

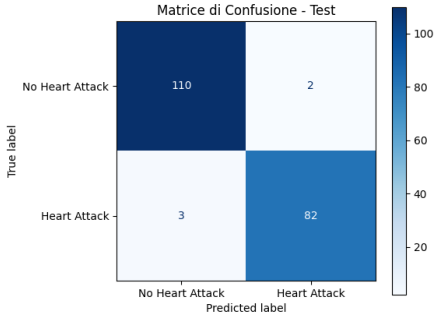

Figura 5.8: Confusion Matrix sui dati di Test

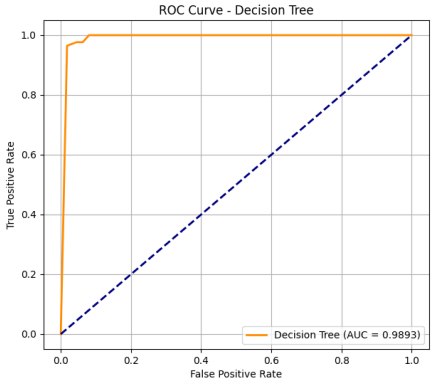

Figura 5.9: AUC-ROC curve del modello Decisoin Tree

Capitolo 6

Conclusioni

Dalle metriche dei due modelli, mostrate nel capitolo precedente, emergono differenze significative. Il modello di "*Logistic Regression*" senza "*StandardScaler*" mostra prestazioni moderate, con un'accuratezza di 0,7563 sul test set e un recall relativamente basso.

La "*Logistic Regression*" scalata con StandardScaler ottiene invece un notevole miglioramento delle prestazioni: l'accuratezza sul test set sale al 0.9289 e il recall raggiunge lo 0.9412, riducendo significativamente i falsi negativi. L'AUC ROC pari a 0,9892 conferma un'eccellente capacità discriminativa tra le classi.

Il "*Decision Tree*" ottiene prestazioni complessivamente più elevate: accuratezza dello 0.9746, F1-score 0.9704 e recall 0.9647 sul test set, indicando una migliore capacità di identificare correttamente i casi positivi e minimizzare i falsi negativi. L'AUC ROC pari a 0,9893 conferma un'eccellente capacità discriminativa tra le classi.

Quindi in conclusione, il "*Decision Tree*" risulta il modello migliore, grazie all'elevata accuratezza, precisione e F1-score. Inoltre, il Decision Tree è anche il modello più efficace nel ridurre i falsi negativi, superando la "*Logistic Regression*" con "*StandardScaler*" in termini di recall.

Bibliografia

[1] Infarto del miocardio Saper riconoscere i sintomi di un infarto è fondamentale per agire tempestivamente e salvare la vita https://www.fondazioneveronesi.it/educazione-alla-salute/glossario/infarto-del-miocardio 1

[2] Infarto – Prevenzione primaria https://www.gavazzeni.it/enciclopedia/prevenzione/infarto-prevenzione-primaria/ 1

[3] Asli Orgerim and Adnan Kalkan, Diagnosing heart attack risk with logistic regression and decision tree algorithms, October 9, 2025, https://doi.org/10.1504/IJHTM.2024.149019 3

[4] Hanaa Albanna a, Madhav Raj Theeng Tamang b, Chandan Patel b, Mhd Saeed Sharif b, Revolutionizing heart attack prognosis: Introducing an innovative regression model for prediction, https://doi.org/10.1016/j.imu.2025.101664 3

[5] Rashid, Tarik A.; Hassan, Bryar (2022), "Heart Attack Dataset", Mendeley Data, V1, doi: 10.17632/wmhctcrt5v.1 https://www.kaggle.com/datasets/fatemehmohammadinia/heart-attack-dataset-tarik-a-rashid/data 7

[6] Fangfang Lee, Cos'è la regressione logistica? https://www.ibm.com/it-it/think/topics/logistic-regression#684929715 18