

**Heart Disease con tecniche di ML**

Di Andrea de Donato e Paolo Di Simone

Indice

[Introduzione 2](#_Toc108188093)

[Data Analysis e Data Visualization 2](#_Toc108188094)

[Matrice di correlazione 3](#_Toc108188095)

[Features Importances con RandomForestClassifier e ExtraTreesClassifier 3](#_Toc108188096)

[Classificazione 4](#_Toc108188097)

[Conclusioni 4](#_Toc108188098)

Indice delle figure

[Figura 1:CountPlot su feature "sex” 4](#_Toc108190095)

[Figura 3: Matrice di correlazione 5](#_Toc108190096)

[Figura 4: Confronto RandomForestClassifier e ExtraTreesClassifier 6](#_Toc108190097)

**Heart Diseases**

# 

# Introduzione

Lo scopo di questo progetto è quello di predire attraverso tecniche di Machine Learning quali individui sono più esposti e soggetti ad avere delle malattie cardiache. L’attenzione per le malattie cardiache è data dal fatto che sono una delle principali cause di morte per gli uomini. Ad esempio, negli Stati Uniti un quarto dei decessi è riconducibile a qualche tipo di malattia cardiaca. Il datasets che è utilizzato per lo studio delle malattie cardiche è “Heart Disease Dataset (Comprehensive)” di IEEE Dataport ([Heart Disease Dataset (Comprehensive) | IEEE DataPort (ieee-dataport.org)](https://ieee-dataport.org/open-access/heart-disease-dataset-comprehensive)). Questo dataset sulle malattie cardiache è ottenuto combinando 5 dataset popolari sulle malattie cardiache già disponibili in modo indipendente ma non combinati prima aiutare con l’obiettivo di far progredire la ricerca sull'apprendimento automatico relativo al CAD (Coronary artery disease) e sugli algoritmi di data mining e per far progredire la diagnosi clinica. I cinque set di dati utilizzati per la sua curatela sono:

1. Cleveland
2. Hungarian
3. Switzerland
4. Long Beach VA
5. Statlog (Heart) Data Set.

Il dataset contiene 12 attributi e 1190 istanze. I principali attributi presenti all’interno di questo dataset che contribuiscono ad un’analisi di questo tipo di malattia sono:

* ***age***
* ***sex***
* ***chest pain type***
* ***resting blood pressure***
* ***serum cholesterol***
* ***fasting blood sugar***
* ***resting electrocardiogram results***
* ***maximum heart rate achieved***
* ***exercise induced angina***
* ***oldpeak =ST***
* ***the slope of the peak exercise ST segment***
* ***class (target)***

Una descrizione più esaustiva degli attributi è reperibile presso il sito di IEEE Dataport.

## Tecnologie Utilizzate

Il progetto è sviluppato in linguaggio Python, tale scelta dipende principalmente dalle librerie che il linguaggio offre per il Machine Learning. Tra queste librerie sono state utilizzate:

* Pandas: per la gestione e la visualizzazione del dataset
* NumPy: per operare più facilmente su matrici e array di grandi dimensioni
* Scikit-learn: per gli algoritmi di Machine Learning
* MatPlotLib e Seaborn: per la visualizzazione grafica dei dati

# Data Analysis e Data Visualization

Prima di analizzare il dataset è necessario suddividere le 12 colonne tra features e target. Le prime 11 colonne fanno riferimento alle features messe a disposizione nel dataset, mentre l’ultima è relativa al target (etichetta classe di appartenenza).

## Data Visualization

Nella fase di Data Visualization sono state rappresentate le istanze del dataset sulla base delle diverse features disponibili. Questa fase è utile per vedere quali sono i casi relativi al Heart Diseases al variare delle features. I vari tipi di grafico che sono stati utilizzati per visualizzare i dati sono: Countplot, Displot e Boxplot. Ciascuna di queste tipologie di grafico consente di mettere in risalto diversi aspetti di ciascuna features. Ad esempio, il Countplot per la feature “sex” permette di capire quanti uomini rispetto alle donne soffrono di malattie cardiache:

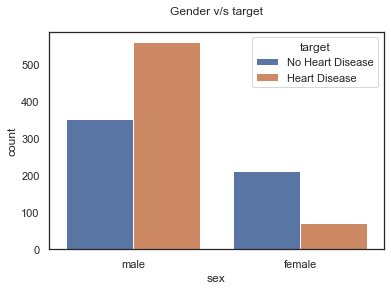
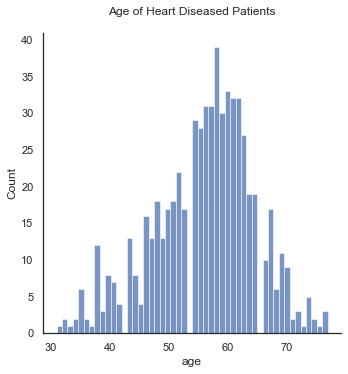
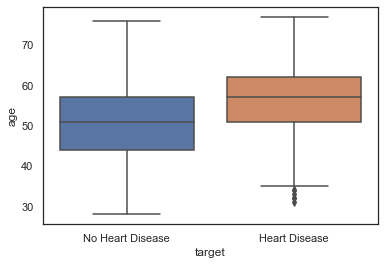


Figura 1:CountPlot su feature "sex”

Ad esempio, il Displot per la feature “age” consente di visualizzare quali sono le fasce di età che sono più soggette a malattie cardiache:



Ad esempio, il Boxplot per la feature “age” consente di visualizzare quale è il range di età degli individui che rischiano di avere malattie cardiache:



## Matrice di correlazione

Il primo passo effettuato nell’ambito della Data Analysis è la creazione della matrice di correlazione per individuare quali features sono più o meno correlate tra di loro. I valori all’interno della matrice di correlazione possono variare da -1 a +1. Come possiamo vedere nella matrice ottenuta, le features che sono più correlate con il target sono: ***chest pain type, exercise angina, oldpeak, ST slope, max hearth rate.***

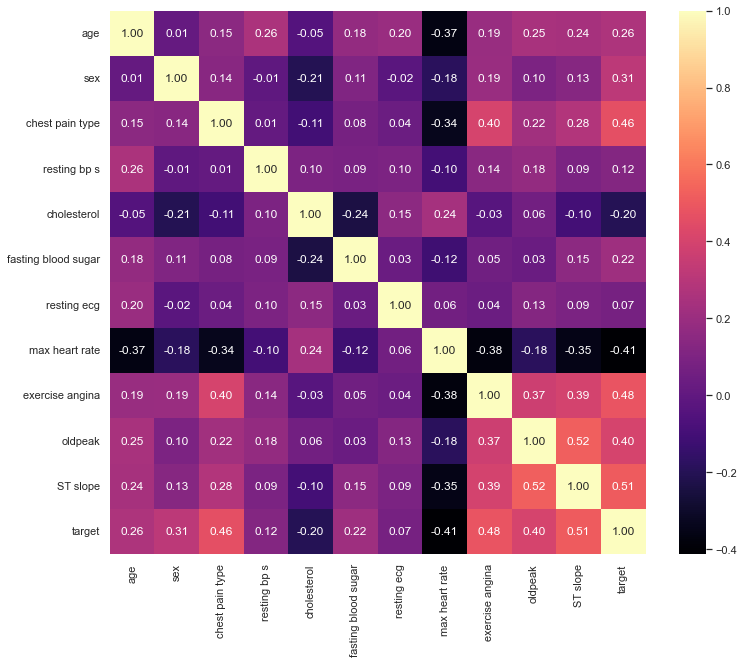
******

Figura 2: Matrice di correlazione

## 

## Features Importances con RandomForestClassifier e ExtraTreesClassifier

Per misurare l’importanza di ogni features sono stati utilizzati due classificatori: RandomForestClassifier e ExtraTreesClassifier. È difficile dire in anticipo se un RandomForestClassifier funzionerà meglio o peggio di un ExtraTreesClassifier. Generalmente, l'unico modo per sapere è provare entrambi e confrontarli. RandomForestClassifier è il più conveniente e il più ottimizzato tra i Decision Trees. L’ExtraTreesClassifier è molto più veloce per l’addestramento rispetto al Random Forest e utilizza le stesse API. Questi Random Forests permettono di misurare features importances in maniera semplice. Scikit-Learn misura l'importanza di una features osservando quanto i nodi dell'albero che utilizzano quella features riducono l’errore di classificazione.

Utilizzando le due diverse tecniche sul dataset in questione sono stati ottenuti i seguenti risultati:

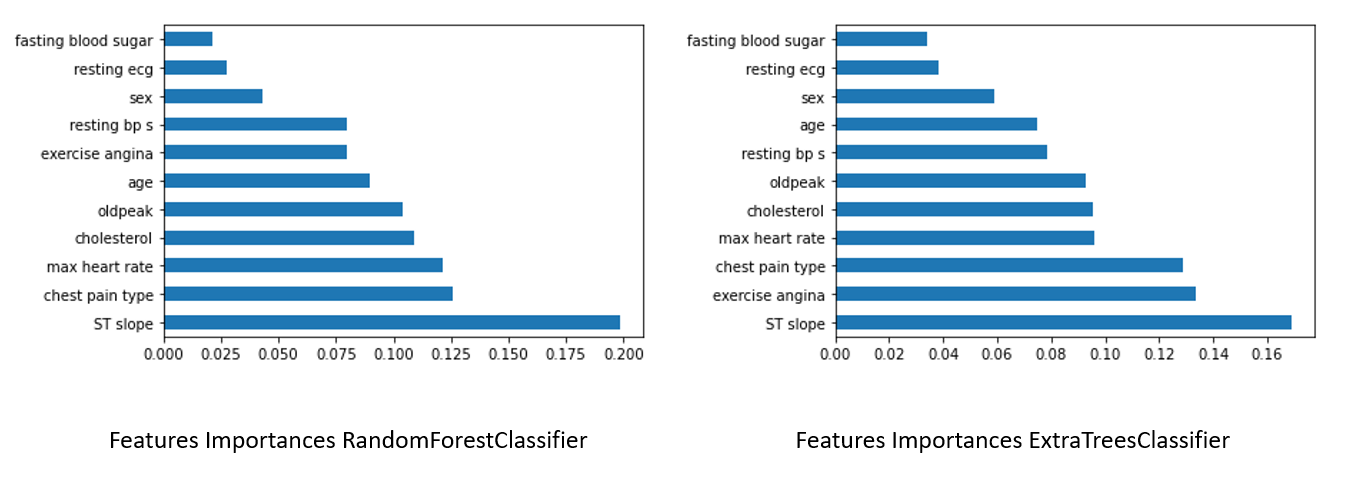


Figura 3: Confronto RandomForestClassifier e ExtraTreesClassifier

In conclusione, si può vedere che con tutti i metodi utilizzati in questa fase di analisi dei dati le features più discriminanti ai fini della classificazione sono le stesse ma con leggere differenze (ad esempio, la features exercise angina in RandomForestClassifier è molto meno importante rispetto a ExtraTreeClassifier)

# Classificazione

## Decision Tree Classifier

Il primo tipo di classificatore utilizzato per effettuare delle predizioni di malattie cardiache è il Decision Tree. Questa scelta è dovuta in primo luogo per la semplicità del modello ma allo stesso tempo per il buon potere espressivo. I classificatori **Decision tree** possono essere visti come un modello che separa i dati prendendo delle decisioni basate su una serie di domande che vengono poste durante il processo decisionale. Il Decision Tree si basa sul concetto di Information Gain, ovvero si scelgono tra le features a disposizione quelle che offrono un guadagno maggiore per essere posizionate nei nodi ad bassa profondità dell’albero.

Nel nostro progetto sono stati applicati diversi alberi per la classificazione con varie profondità. La profondità può essere specificata attraverso un iperparametro (max\_depth) all’interno del costruttore dell’albero. In scikit-learn, nel caso in cui non si specifica la profondità, si genera un albero con un profondità necessaria per avere tutte le foglie con delle istanze pure (esempi appartenenti alla stessa classe target), quindi ci si troverà ad avere un caso di overfitting sul training set. Le ulteriori profondità utilizzate per l’addestramento sono state 5, 10 e 15 nodi.

Per poter visualizzare l’albero che viene ottenuto attraverso l’addestramento è stata utilizzata la libreria export\_graphviz del modulo sklearn.tree

Si può notare attraverso le accuracy che sono state ottenute dai diversi classificatori che l’albero con profondità pari a 5 si dimostra il migliore poiché riduce l’overfitting sul training test mantenendo dei risultati molto simili nell’accuracy del test set. I risultati ottenuti sull’albero di profondità sono:

Immagine che contiene testo, arancia

Descrizione generata automaticamente

Figura 4: Risultati Accuracy su Decision Tree con profondità di 5 nodi

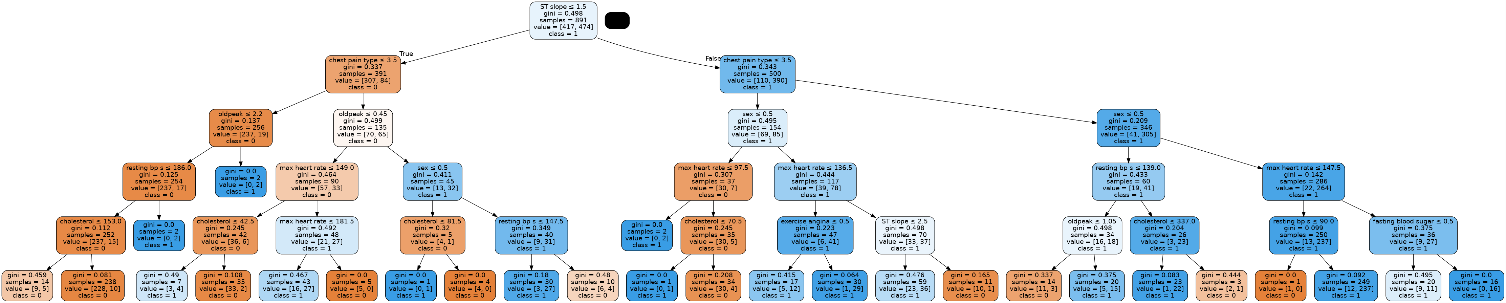


Figura 5: Visualizzazione Decision Tree con profondità di 5 nodi (libreria export\_graphviz)

I risultati ottenuti dagli altri Decision Tree sono:

Immagine che contiene testo

Descrizione generata automaticamente

Figura 6: Confronto risultati Decision Tree con varie profondità

Come si può vedere nel caso di Decision Tree senza profondità specificata abbiamo che tutte le istanze del training set vengono classificate correttamente poiché l’albero costruito ha tutti gli split necessari per avere un completo adattamento.

## K-NN

K-NN è basato su un algoritmo molto semplice, ovvero attribuisce all’istanza che si vuole classificare l’etichetta della classe appartenente all’esempio più vicino (simile). In scikit-learn, uno degli iperparametri più importanti che può essere modificato è **n\_neighbors.** Questo iperparametro consente di indicare quanti sono gli esempi più vicini che contribuiscono alla classificazione all’istanza. In questo caso, l’iperparametro non è stato modificato lasciando il valore di default (n\_neighbors = 5) poiché a seguito di variazioni del valore del parametro sono state ottenute dei risultati peggiori. Il risultato, quindi, è ottenuto utilizzando il valore di default di n\_neighbors:

Immagine che contiene tavolo

Descrizione generata automaticamente

Figura 7: Risultati classificatore K-NN

La **f1-score** rappresenta la media armonica tra la precision e recall, mentre **support** rappresenta il numero di casi individuati per ciascun valore della classe di appartenenza. I risultati ottenuti sono risultati peggiori rispetto alla classificazione attraverso Decision Tree.

## SVM

L'obiettivo dell'algoritmo SVM è trovare un iperpiano in uno spazio N-dimensionale (N — il numero di features) che classifichi distintamente i punti dati. Nel costruttore di svm si hanno diversi iperparametri importanti:

* **C** è un parametro di regolarizzazione. All’aumentare del valore di C diminuisce il numero di support vector (i.e., complessità del problema), diminuisce il numero di errori sul Training Set e diminuisce il margine di separazione (i.e., capacità di generalizzazione)
* **kernel** specifica il tipo di kernel da utilizzare nell'algoritmo. Se non viene specificato, verrà utilizzato 'rbf'.
* **Random state** controlla la generazione di numeri pseudo casuali per mescolare i dati per le stime di probabilità. Consente, quindi, di avere gli stessi risultati nelle diverse esecuzioni.
* **E molti altri…**

I migliori modelli SVM impiegati sono stati ottenuti variando i parametri **C** e **kernel**. Il migliore classificatore SVM è stato ottenuto impostando l’iperparametro C pari a 200 e come tipo di kernel lineare. Con la tipologia di kernel RBF si ottengono dei risultati del tutto comparabili a quello con tipologia lineare.

I risultati ottenuti con parametri C=200 e kernel='linear' sono:

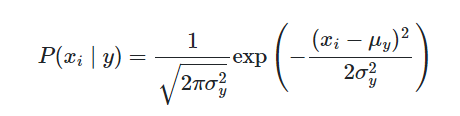
Immagine che contiene tavolo

Descrizione generata automaticamente

Figura 8: Risultati classificatore SVM

## Naïves Bayes Classifier

I metodi Naive Bayes sono un insieme di algoritmi di apprendimento supervisionato basati sull'applicazione del teorema di Bayes con l'assunzione "ingenua" dell'indipendenza condizionale tra ogni coppia di caratteristiche dato il valore della variabile di classe. Come sappiamo l’approccio Bayesiano si basa sulla teoria delle probabilità per assegnare ad un esempio la relativa classe di appartenenza. In questo contesto è stato utilizzato **GaussianNB** che implementa l'algoritmo Gaussian Naive Bayes per la classificazione. Si presume che la probabilità delle caratteristiche sia gaussiana:



I risultati ottenuti attraverso GaussianNB sono:

Immagine che contiene tavolo

Descrizione generata automaticamente

Figura 9: Risultati Classificatore Naives Bayes

## Logistic Regression

La regressione logistica stima la probabilità del verificarsi di un evento, come ad esempio voto espresso o non espresso. la regressione logistica viene utilizzata anche per stimare la relazione tra una variabile dipendente e una o più variabili indipendenti, ma viene utilizzata per fare una previsione circa una variabile categoriale rispetto a una continua. Una variabile categoriale può essere true o false, sì o no, 1 o 0 eccetera.

Uno degli iperparametri più importanti è la penalty. La penalty consente di definire quale tipo di regolarizzazione utilizzare. Per default viene utilizzata la L2 (ridge). A seguito dell’addestramento di diversi modelli si giunti alla conclusione che tra la penalty L2 e None (assenza di regolarizzazione), in questo caso, offrono dei risultati simili e comparabili. I risultati sono i seguenti:

Immagine che contiene testo

Descrizione generata automaticamente

Figura 10: Confronto risultati Logistic Regression con diverse penalty

## Random Forest (MIGLIOR MODELLO)

Il Random Forest, come suggerisce il nome, è costituito da un gran numero di alberi decisionali (Decision Tree) che operano come un insieme. Ogni singolo albero nel Random Forest effettua una previsione di classe e la classe con il maggior numero di voti diventa la previsione del nostro modello.

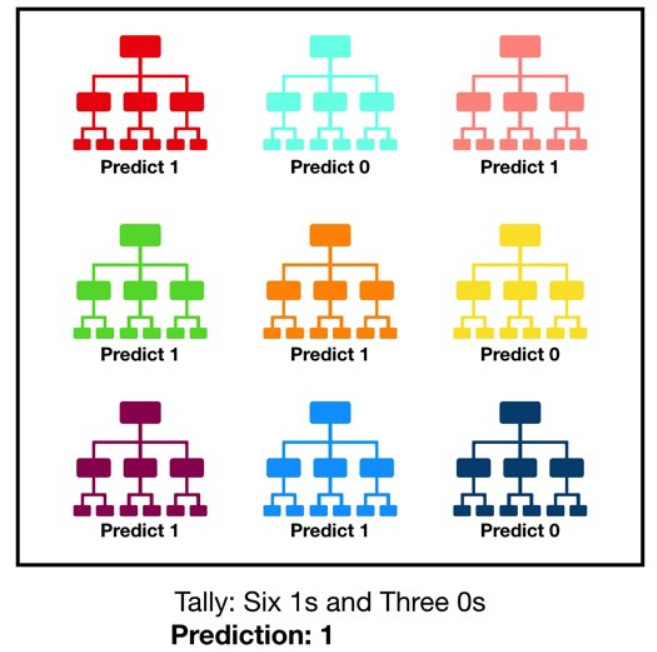


Figura 11: Visualizzazione di un modello Random Forest ed esecuzione della stima

In questo caso è stato utilizzato un Random Forest con alberi di profondità pari a 5 (uguale al migliore modello individuato nei Decision Tree). I risultati ottenuti sono i seguenti:

Immagine che contiene tavolo

Descrizione generata automaticamente

Figura 12: Risultati Classificatore Random Forest

Come si può vedere rispetto a tutti gli altri modelli discussi precedentemente, il Random Forest rappresenta il miglior classificatore poiché riesce ad ottenere dei buoni risultati sul test set senza avere overfitting sul training set.

# Conclusioni