

**Heart Disease con tecniche di ML**

Di Andrea de Donato e Paolo Di Simone

Indice

[Introduzione 3](#_Toc109742235)

[Tecnologie Utilizzate 4](#_Toc109742236)

[Data Analysis e Data Visualization 4](#_Toc109742237)

[Data Visualization 4](#_Toc109742238)

[Matrice di correlazione 6](#_Toc109742239)

[Features Importances con RandomForestClassifier e ExtraTreesClassifier 6](#_Toc109742240)

[Classificazione 7](#_Toc109742241)

[DecisionTreeClassifier 7](#_Toc109742242)

[K-NN 8](#_Toc109742243)

[SVM 9](#_Toc109742244)

[Naïves Bayes Classifier 9](#_Toc109742245)

[Logistic Regression 10](#_Toc109742246)

[RandomForestClassifier (MIGLIOR MODELLO) 10](#_Toc109742247)

[Features Selection 11](#_Toc109742248)

[Min Max Scaling 12](#_Toc109742249)

[Conclusioni 12](#_Toc109742250)

Indice delle figure

[Figura 1: Countplot feature "sex” 5](#_Toc109742174)

[Figura 2: Displot feature "age" 6](#_Toc109742175)

[Figura 3: Boxplot feature "age" 6](#_Toc109742176)

[Figura 4: Matrice di correlazione 7](#_Toc109742177)

[Figura 5: Confronto RandomForestClassifier e ExtraTreesClassifier 8](#_Toc109742178)

[Figura 6: Risultati Accuracy su Decision Tree con profondità di 5 nodi 9](#_Toc109742179)

[Figura 7: Visualizzazione Decision Tree con profondità di 5 nodi (libreria export\_graphviz) 9](#_Toc109742180)

[Figura 8: Visualizzazione di un modello Random Forest ed esecuzione della stima 12](#_Toc109742181)

# 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.

Ogni anno le malattie cardiovascolari causano 3.9 milioni di morti in Europa e più di 1.8 milioni di morti nell’Unione europea; quindi, queste malattie contano il 45% delle morti totali in Europa e il 37% delle morti totali nell’Unione europea [1].

Il datasets utilizzato per lo studio di queste malattie cardiache è *“Heart Disease Dataset (Comprehensive)”* di *IEEE Dataport* [2]. Questo dataset sulle malattie cardiache è ottenuto combinando 5 dataset sulle malattie cardiache già disponibili in modo indipendente con l’obiettivo di far progredire la ricerca sul Machine Learning relativo al CAD (Coronary Artery Disease) e sugli algoritmi di data mining e per far progredire la diagnosi clinica (di queste malattie cardiache). I 5 dataset utilizzati 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 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 [2].

## 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, mentre l’ultima è relativa al target (etichetta della 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:

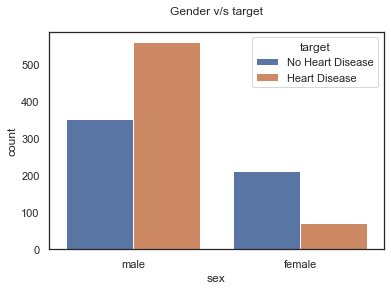


Figura 1: Countplot feature "sex”

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

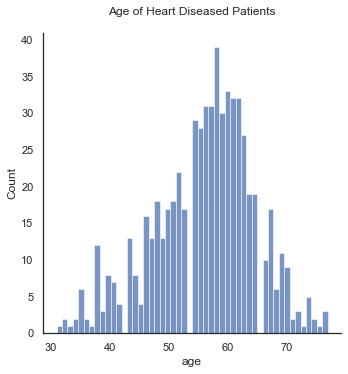


Figura 2: Displot feature "age"

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; quindi, permette di individuare gli “outlier” (che se non opportunamente gestiti possono portare alla degradazione del sistema):

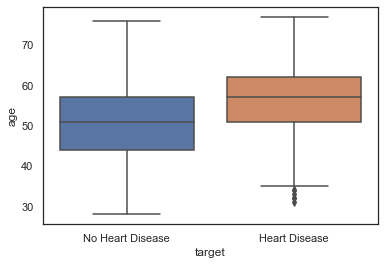


Figura 3: Boxplot feature "age"

## 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.

Possiamo osservare come c’è una correlazione positiva tra le feature ***chest pain type***, ***exercise angina***, ***oldpeak*** e ***ST slope***, con il ***target***; mentre c’è una correlazione negativa tra la feature ***max heart rate*** e il ***target***.

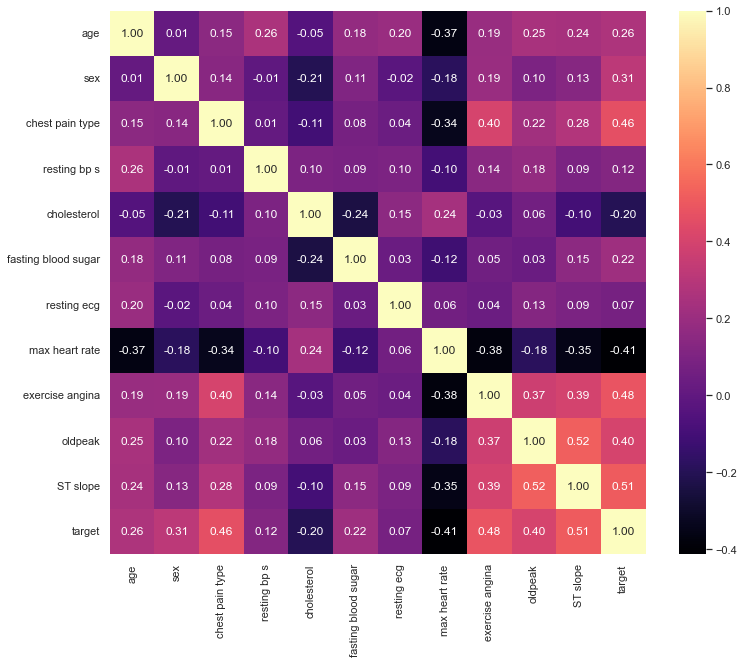
******

Figura 4: 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 [3].

RandomForestClassifier è il più conveniente e il più ottimizzato tra i Decision Trees, tuttavia, l’ExtraTreesClassifier è molto più veloce per l’addestramento rispetto al Random Forest e utilizza le stesse API.

Questi Decision Tree permettono di misurare features importance 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:

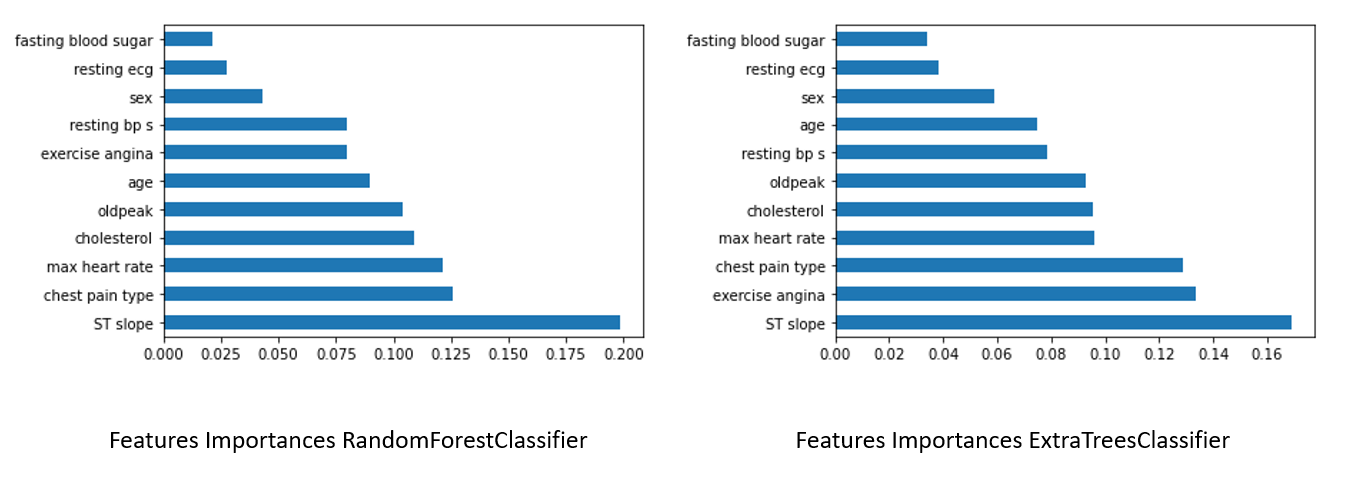


Figura 5: Confronto RandomForestClassifier e ExtraTreesClassifier

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 in ExtraTreesClassifier)

# Classificazione

## DecisionTreeClassifier

Il primo tipo di classificatore utilizzato per effettuare delle predizioni di malattie cardiache è il DecisionTreeClassifier. Questa scelta è dovuta in primo luogo per la semplicità del modello ma allo stesso tempo per il buon potere espressivo. I DecisionTreeClassifier 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 DecisionTreeClassifier 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 a bassa profondità dell’albero.

Nel nostro progetto sono stati applicati diversi alberi per la classificazione con varie profondità. La profondità può essere specificata attraverso l’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 **max\_depth=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 con **max\_depth=5** sono:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.83 | 0.81 | 0.82 | 144 |
| 1 | 0.82 | 0.85 | 0.84 | 154 |

|  |  |
| --- | --- |
| Accuracy sul Training Set | 0.86532 |
| Accuracy sul Test Set | 0.82886 |

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.

L’albero di decisione ottenuto è il seguente:

Figura 6: Risultati Accuracy su Decision Tree con profondità di 5 nodi

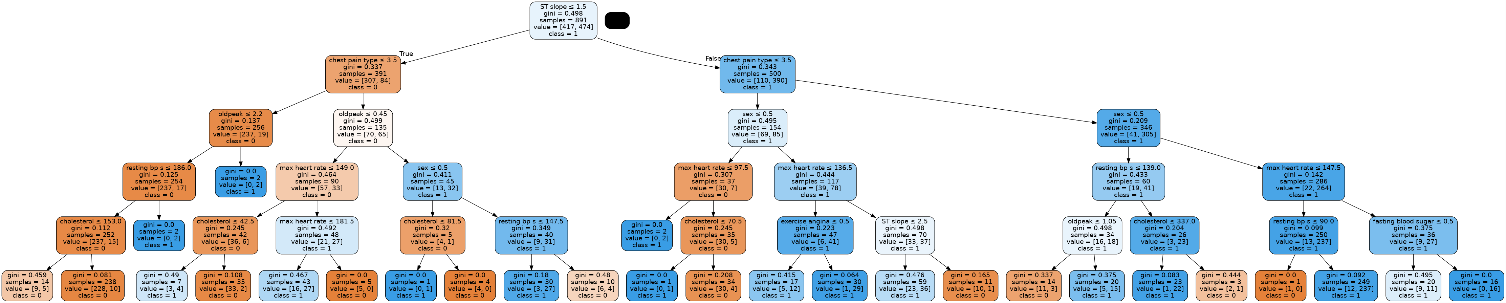


Figura 7: Visualizzazione Decision Tree con profondità di 5 nodi (libreria export\_graphviz)

Per il DecisionTreeClassifier senza limiti sulla profondità dell’albero si è ottenuto:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.84 | 0.94 | 0.89 | 144 |
| 1 | 0.93 | 0.83 | 0.88 | 154 |

|  |  |
| --- | --- |
| Accuracy sul Training Set | 1.00000 |
| Accuracy sul Test Set | 0.88255 |

Come si può vedere, in questo caso, 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**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.74 | 0.72 | 0.73 | 144 |
| 1 | 0.75 | 0.77 | 0.76 | 154 |

|  |  |
| --- | --- |
| Accuracy sul Training Set | 0.77441 |
| Accuracy sul Test Set | 0.74497 |

## 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=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:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.86 | 0.84 | 0.85 | 144 |
| 1 | 0.85 | 0.88 | 0.87 | 154 |

|  |  |
| --- | --- |
| Accuracy sul Training Set | 0.82043 |
| Accuracy sul Test Set | 0.85906 |

## Naïves Bayes Classifier

I metodi Naïve 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 GaussianNBche implementa l'algoritmo Gaussian Naïve Bayes per la classificazione.

Si presume che la probabilità delle feature sia gaussiana:

I risultati ottenuti attraverso il classificatore di Naïve Bayes sono:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.84 | 0.90 | 0.87 | 144 |
| 1 | 0.90 | 0.84 | 0.87 | 154 |

|  |  |
| --- | --- |
| Accuracy sul Training Set | 0.83277 |
| Accuracy sul Test Set | 0.86577 |

## 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, in questo caso, utilizzare la penalty L2 o non utilizzare alcuna penalty porta agli stessi risultati.

Per la regressione logistica con penalty L2 si è ottenuto:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.82 | 0.88 | 0.85 | 144 |
| 1 | 0.88 | 0.82 | 0.85 | 154 |

|  |  |
| --- | --- |
| Accuracy sul Training Set | 0.82828 |
| Accuracy sul Test Set | 0.84899 |

## RandomForestClassifier (MIGLIOR MODELLO)

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

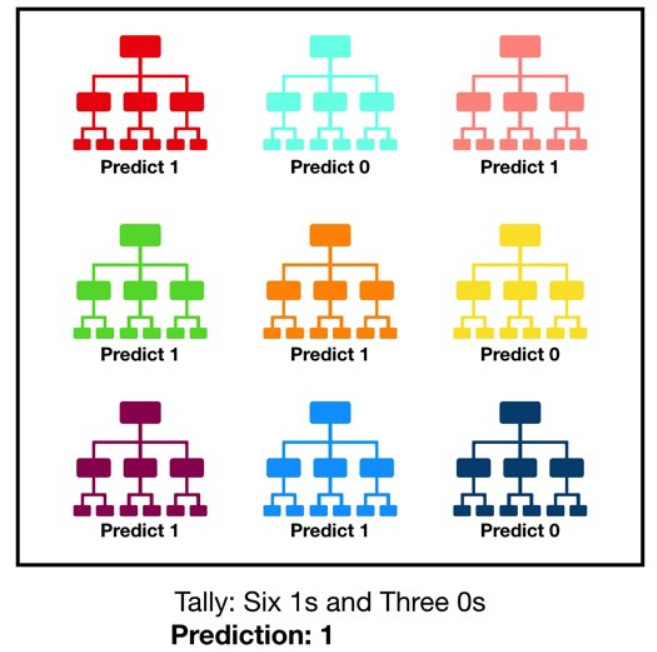


Figura 8: Visualizzazione di un modello Random Forest ed esecuzione della stima

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.92 | 0.90 | 0.91 | 144 |
| 1 | 0.91 | 0.92 | 0.91 | 154 |

|  |  |
| --- | --- |
| Accuracy sul Training Set | 0.89113 |
| Accuracy sul Test Set | 0.91275 |

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

## Features Selection

Per valutare un eventuale miglioramento dei modelli utilizzati con i migliori risultati (RandomForestClassifier e LogisticRegression) è stata applicata la tecnica di Features Selection, attraverso la quale sono stati rimossi dal dataset 6 features, per mantenere le features più significative: ***chest pain type, max heart rate, exercise angina, oldpeak, ST slope.***

In primo luogo, la Features Selection è stata utilizzata con RandomForestClassifier con la profondità degli alberi pari a 5 (quella risultata migliore nei test precendenti) ottenendo dei risultati peggiori rispetto a quelli ottenuti senza Features Selection.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.86 | 0.86 | 0.86 | 144 |
| 1 | 0.87 | 0.86 | 0.87 | 154 |

|  |  |
| --- | --- |
| Accuracy sul Training Set | 0.86532 |
| Accuracy sul Test Set | 0.86242 |

I risultati ottenuti con Features Selection applicata a LogisticRegression sono, anche in questo caso, peggiori alla Logistic Regression senza la Features Selection.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.80 | 0.85 | 0.82 | 144 |
| 1 | 0.85 | 0.81 | 0.83 | 154 |

|  |  |
| --- | --- |
| Accuracy sul Training Set | 0.81706 |
| Accuracy sul Test Set | 0.82550 |

## Min Max Scaling

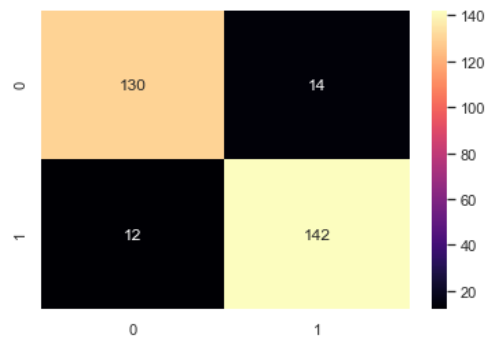
Una delle più importanti trasformazioni che devono essere applicate è il ridimensionamento. Di solito, gli algoritmi di Machine Learning non funzionano bene quando gli attributi numeri hanno delle scale molto diverse tra di loro. È stata utilizzata la Min Max Scaling che effettua la normalizzazione delle feature applicando la seguente formula:

Utilizzando la Features Scaling abbinata alla Features Selection per i modelli Random Forest e Logistic Regression si sono ottenuti, in entrambi i modelli, dei risultati peggiori rispetto agli addestramenti iniziali.

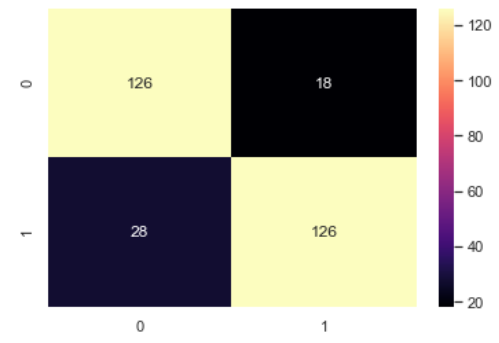
# Conclusioni

In conclusione, sono state calcolate le matrici di confusione per i due classificatori:

* RandomForestClassifier



* LogisticRegression



La matrice di confusione conferma quello che si evince dalle accuracy ottenute precedentemente. Infatti, il numero di falsi positivi e falsi negativi nel RandomForestClassifier è decisamente minore.

Stessa cosa può essere vista attraverso il test di significatività di McNemar’s che permette di confrontare le differenze fra due classificatori, in particolare sulle istanze classificate in modo diverso.

Calcolando la matrice di contingenza:

Con:

* : numero di istanze classificate male da entrambi i classificatori;
* : numero di istanze classificate male dal primo classificatore, ma non dal secondo;
* : numero di istanze classificate male dal secondo classificatore, ma non dal primo;
* : numero di istanze classificate correttamente da entrambi i classificatori.

In questo caso si ha:

Nel test di McNemar’s:

* L’ipotesi nulla è che cioè che i due classificatori “commettano gli stessi errori”;
* L’ipotesi alternativa è che le performance dei due modelli siano diverse.

Il test di McNemar’s può essere visto come una chi-quadro:

L’ipotesi nulla è accettata se il p-value è minore di una soglia specificata (in questo caso 0.05).

In questo caso, ovviamente l’ipotesi nulla è rigettata, dal momento che i due classificatori hanno un’accuracy molto diversa (in generale questo test è utilizzato per classificatori che hanno un accuracy molto simile).

Come ultima cosa sono state fatte delle predizioni a partire da delle istanze come:

* ***age = 58***
* ***sex = 1***
* ***chest pain type = 4***
* ***resting blood pressure = 130***
* ***serum cholesterol = 0***
* ***fasting blood sugar = 0***
* ***resting electrocardiogram results = 1***
* ***maximum heart rate achieved = 100***
* ***exercise induced angina = 1***
* ***oldpeak ST = 1***
* ***the slope of the peak exercise ST segment = 2***

Il RandomForestClassifier classifica questa istanza come 1, cioè con possibile Heart Diseases.