Heartcare

Progetto di Fondamenti di Intelligenza Artificiale
Università degli Studi di Salerno

Mario Cicalese, Leopoldo Todisco, Carlo Venditto

Index

Abstract

L'obiettivo del progetto è quello di usare metodi di Machine Learning per aiutare gli iscritti alla piattaforma HeartCare a prevedere un possibile infarto in base alle misurazioni precedentemente effettuate mediante dispositivi medici. Nella fattispecie, è stato adottato un algoritmo di classificazione, nel corso del documento saranno esplicati i tre modelli generati e l'algoritmo genetico che consente di ottimizzare la scelta dei parametri. Nella fase finale del documento vengono esposti prima i risultati di ogni modello, poi eventuali risvolti futuri.

1 Introduzione

Le malattie cardiovascolari sono la prima causa di morte nella regione Oms Europa, ma negli ultimi anni la situazione è cambiata: il numero di decessi è salito drammaticamente nei Paesi dell'Europa centrorientale.
Stime dell'ISS contano che circa la metà di tutti i decessi sono dovuti a problematiche cardiovascolari.

1.1 Situazione attuale

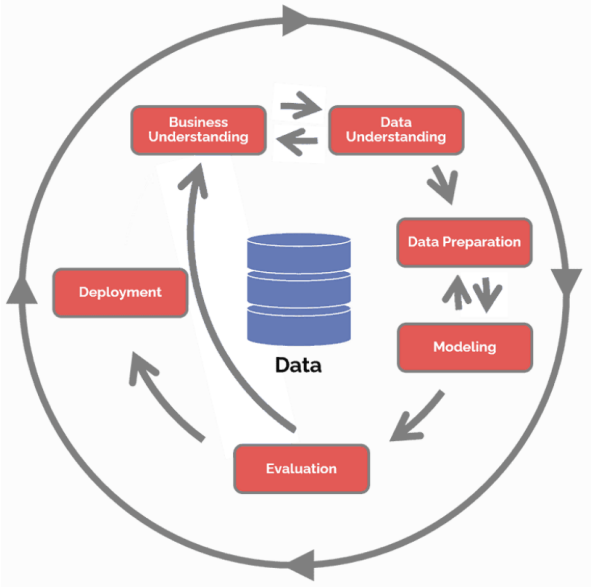
Al giorno d'oggi l'infarto è una malattia non considerata sopratutto tra le persone di giovane età. Quest'ultimo è sottovalutato a causa della mancanza di consapevolezza delle persone che non conoscono i sintomi o semplicemente li ignorano non effettuando visite mediche. Grazie al classificatore implementato nel sistema HeartCare è possibile effettuare una previsione in pochi secondi permettendo alle persone di rimanere nella propria abitazione, risolvendo problemi critici legati alla mobilità.

1.2 Specifica PEAS

- **P**erformance: è data dal valore del f1-score della classificazione;
- **E**nvironment: paziente, casa;
- **A**ctuators: schermo per visualizzare parametri e domande;
- **S**ensors: dispositivo medico, computer per inserire eventuali dati.

1.3 CRISP-DM

Il CRISP-DM (Cross-Industry Standard Process for Data Mining) è il modello di ciclo di vita del software standard per progetti di Machine Learning, si compone di fasi che possono essere eseguite un numero illimitato di volte, il che lo rende un modello non sequenziale.



1.4 Business Understanding

Questa è la prima fase del modello CRISP-DM, e consiste nella definizione dei requisiti, e nella stima di ipotetici rischi. L'obiettivo di business è quello di stimare, a partire da misurazioni relative ai battiti per minuto, pressione e colesterolo, se un paziente è a rischio infarto o no.
Essendo il risultato un valore discreto (1 o 0) è idoneo utilizzare un algoritmo di classificazione.

2 Metodologia per la classificazione

2.1 Data Understanding

2.1.1 Acquisizione dei dati

Il dataset utilizzato risale al 1988, ed è il risultato di quattro database: Cleveland, Ungheria, Svizzera e Long beanch V ed è possibile ritrovarlo su Kaggle e scaricarlo in formato csv.

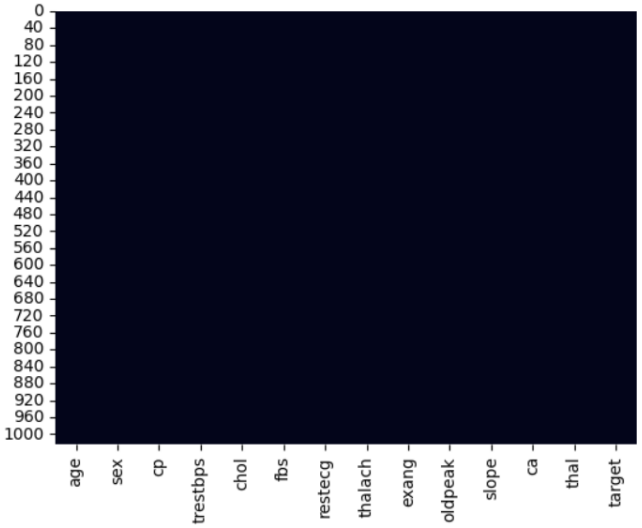
2.1.2 Esplorazione dei dati

I dati si presentano in formato tabellare per un totale di 1026 righe e 14 colonne, e ogni colonna è divisa da una virgola. Le quattordici caratteristiche del dataset sono:

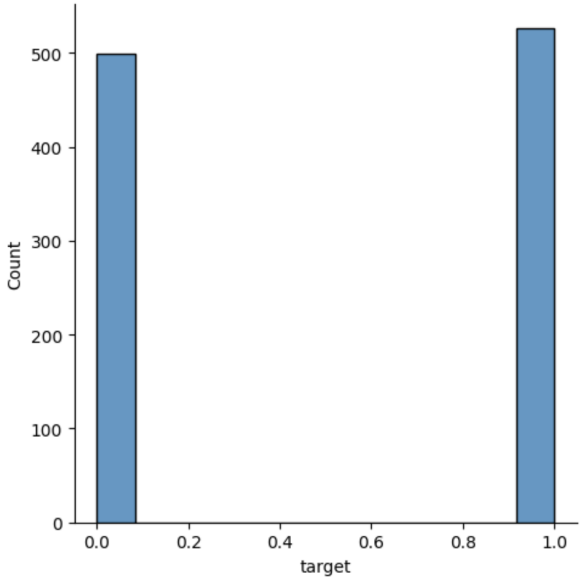
1. age;
2. sex, può valere 0 o 1. 1 indica sesso maschile e 0 sesso femminile;
3. cp, indica il dolore al petto in scala 0 a 3, a seconda della forza;
4. trestbps, indica la pressione media;
5. chol, indica il livello di colesterolo;
6. fbs, indica il livello di zuccheri nel sangue;
7. restecg, indica l'esito degli ultimi risultati di una misurazione elettrocardiogramma, può avere valori 0 o 1;
8. thalach, indica i battiti per minuto;
9. exang, indica il dolore al braccio sinistro o al petto post esercizio fisico. Può valere 0 o 1, in base al fatto che esso ci sia o meno;
10. oldpeak
11. slope
12. ca, numero di vene che sono evidenziate dalla fluoroscopia;
13. thal, indica se la persona non ha mai avuto un infarto, se ha avuto un infarto ed è guarita completamente o se ha avuto un infarto ma non è del tutto guarita. Può assumere valori che vanno da 1 a 3;
14. target, vale 1 se l'entry è relativa a un malato, 0 altrimenti.

2.1.3 Qualità dei dati

È stata condotta un'analisi della qualità dei dati per individuare i possibili dati mancanti.



Come emerge dal grafico non sono presenti dati mancanti, se questi fossero presenti ci sarebbero state dei punti bianchi in corrispondenza della riga.



Il dataset appare già di default quasi totalmente bilanciato.

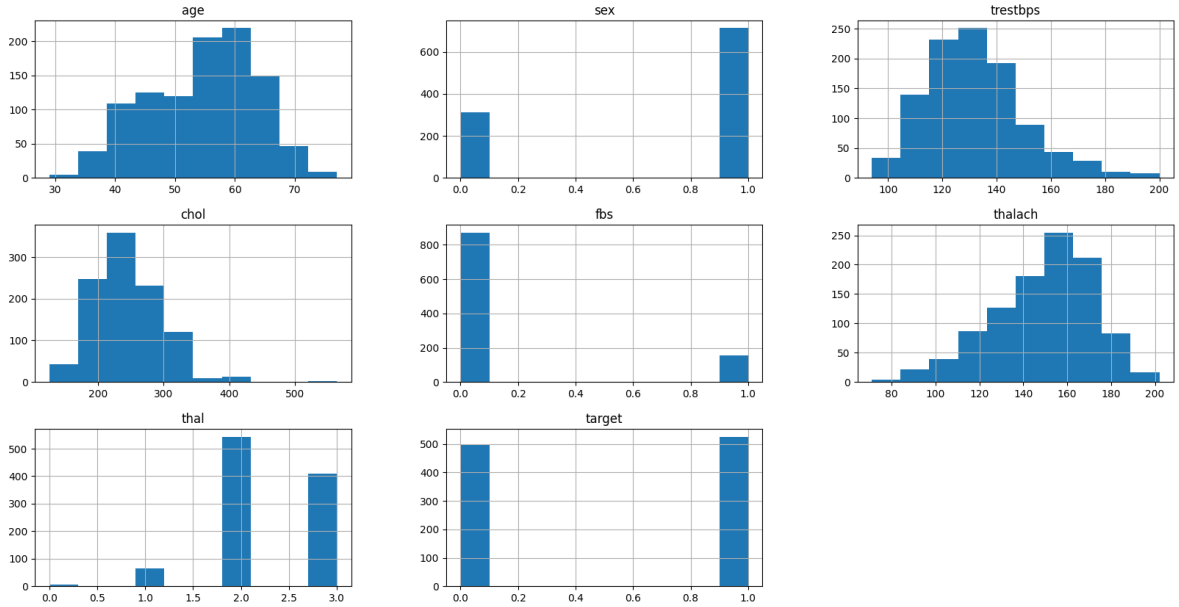
2.2 Data Preparation

2.2.1 Data cleaning

Poichè il dataset non presenta istanze vuote non è stato necessario eseguire operazioni di data cleaning.

2.2.2 Feature Selection

Siccome la piattaforma HeartCare non comprende alcuni attributi presenti nel dataset, le seguenti features sono state eliminate: exang, ca, cp, slope, oldpeak, restecg. Ecco come compare il dataset al termine di questa fase:



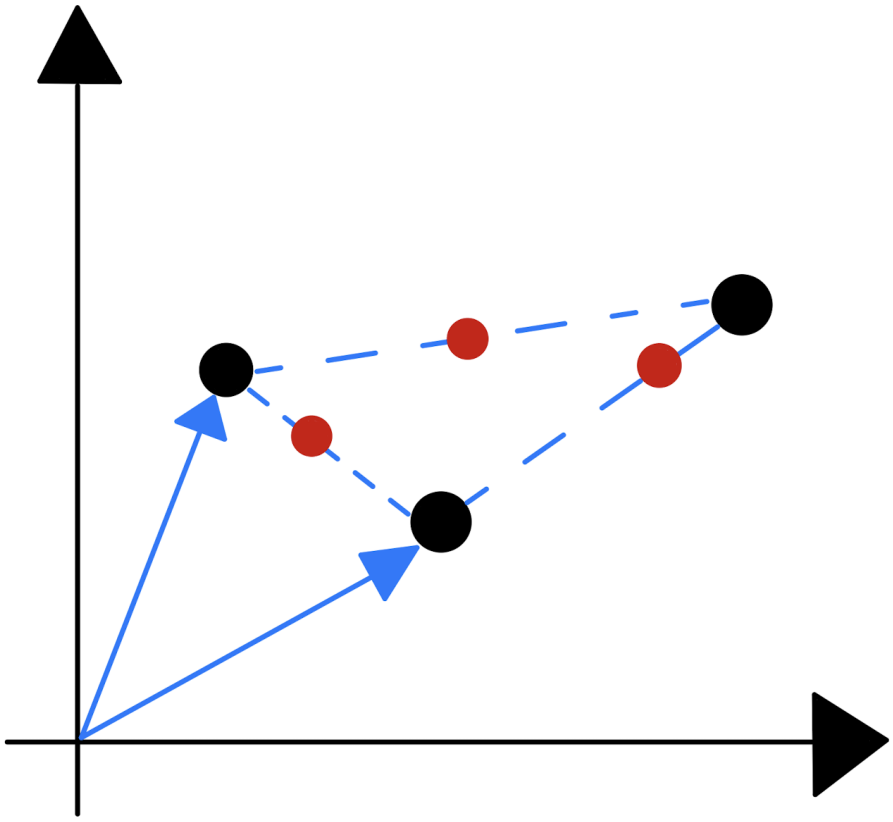
2.2.3 Feature Scaling

Per la normalizzazione dei dati si è fatto ricorso alla minmax normalization usando il MinMaxScaler di sklearn.

2.2.4 Data Balancing

Sebbene il dataset fosse quasi ben bilanciato, si è deciso di applicare tecniche di oversampling per raggiungere un bilanciamento perfetto. L'oversampling è stato reso possibile dalla libreria imblearn con l'algoritmo SMOTE.
Come funziona SMOTE?

1. dato un punto della classe di minoranza si calcolano i k-vicini e si identifica il "feature vector";
2. si calcola la distanza fra il vettore e il vicino a minor distanza;
3. si moltiplica la distanza con un numero casuale fra 0 e 1;
4. si identifica un nuovo punto lungo il segmento aggiungendo il numero casuale al vettore.
5. si ripete il processo per tutti i vettori identificati



2.3 Algoritmo genetico

Per ottimizzare le prestazioni degli alberi decisionali si è pensato di utilizzare un algoritmo genetico per la scelta dei parametri. Gli algoritmi genetici sono una procedura di alto livello ispirata alla genetica per definire un algoritmo di ricerca. Partendo da una popolazione di individui (soluzioni candidate) si procede con l'evoluzione di quest'ultima al fine di produrre soluzioni migliori rispetto ad una funzione di fitness. Tramite l'utilizzo di operatori genetici è possibile creare nuove generazioni.

2.3.1 Inizializzazione

Si è scelto di utilizzare una codifica che prevede l'utilizzo di un array di interi al cui suo interno sono presenti:

1. max-depth: che indica la profondità massima dell'albero e può assumere valori in un range tra 1 e 20.
2. min-sample-split: che indica il minimo numero di campioni richiesti per dividere un nodo interno e può assumere valori tra 1 e 1000.
3. min-sample-leaf: il minimo numero di campioni richiesti per trovarsi in un nodo foglia e può assumere valori tra 1 e 1000.
4. max-features: il numero di feature da considerare quando si cerca la divisione migliore e può assumere valore tra 1 e 9.

Population size: la taglia della popolazione è fissata a 100 per non rendere la funzione di fitness troppo complessa da calcolare.

2.3.2 Selezione

Un algoritmo di selezione indica come avviene la selezione degli individui da poter portare nella prossima generazione. È stato utilizzato un **tournament selection** dove gli individui sono selezionati casualmente andando a formare il torneo e il migliore di tra questi individui passa la selezione definitivamente.

2.3.3 Crossover

È stato deciso di utilizzare l'algoritmo **one-point** in cui si va a selezionare un punto del patrimonio genetico dei genitori e si procede alla generazione di due figli tramite scambio di cromosomi.
Il razionale dietro questa scelta è dato dal fatto che altri algoritmi come Arithmetic, calcolando la media aritmetica, potevano fornire dei valori che non erano interi e quindi non accettati come parametri del classificatore.

2.3.4 Mutazione

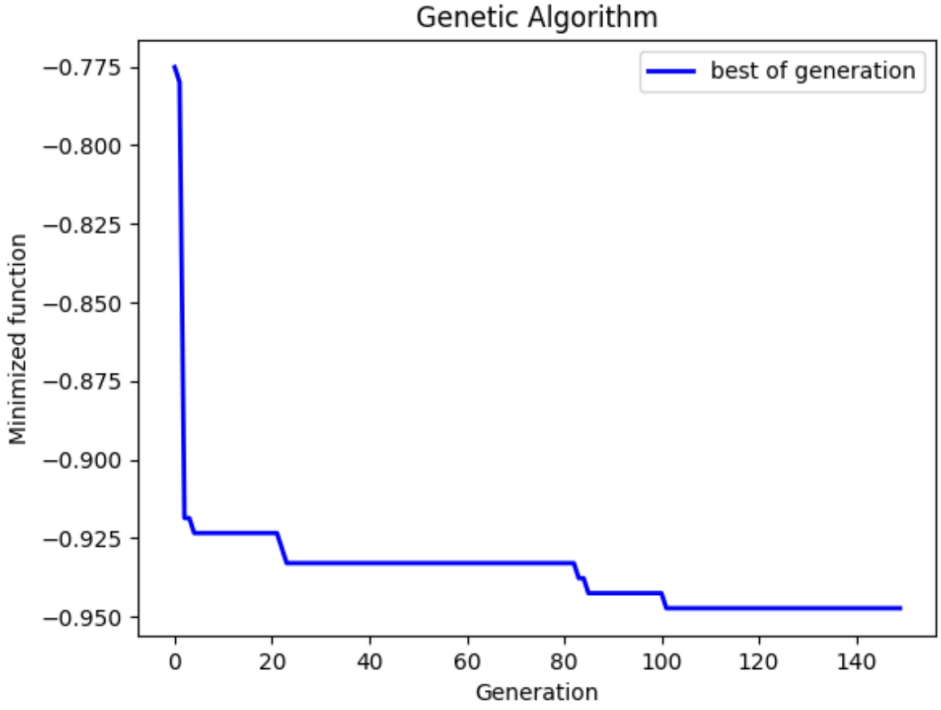
Per la mutazione si è utilizzato l'algoritmo **uniform by center**. Questa strategia consiste nel generare nuovi valori casuali per i geni di un individuo scegliendo un valore centrale (spesso il valore attuale del gene) e generando un intervallo uniforme intorno ad esso. I nuovi valori vengono quindi scelti casualmente all'interno di questo intervallo.
Quando si dice che un valore viene scelto in modo uniforme significa che ogni valore all'interno di un determinato intervallo ha la stessa probabilità di essere selezionato.

2.3.5 Stopping condition

L'algoritmo termina dopo 150 iterazioni.

2.3.6 Funzione di fitness

La funzione di fitness misura il grado di conformità di una soluzione rispetto al problema. Nel nostro caso la funzione di fitness restituisce il valore dell'f-measure (media ponderata tra accuracy e recall) che il modello assume quando viene inizializzato con i parametri dell'algoritmo genetico.



I migliori valori individuati sono stati:

12 per max-depth

5 per min-samples-split

1 per min-samples-leaf

6 per max-fatures

2.4 Modelling

Terminata la fase di preprocessing, il prossimo step è quello di creare un modello per l'apprendimento.

2.4.1 Gaussian Naive Bayes

Gaussian Naive Bayes è un algoritmo di classificazione probabilistico che considera le caratteristiche della nuova istanza da classificare e calcola la probabilità che queste facciano parte di una classe tramite l'applicazione del teorema di Bayes. Il problema principale di tale algoritmo è che assume che le caratteristiche non siamo correlate l'una all'altra. Di conseguenza, l'algoritmo non andrà a valutare, in fase di classificazione, la potenziale utilità data dalla combinazione di più caratteristiche. Tutto questo perchè la classificazione viene eseguita sulla base del teorema di Bayes :

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

P(A) e P(B) rappresentano le probabilità di osservare A e B indipendentemente dall'altro.
P(B|A) rappresenta la probabilità di osservare B dato che A si è già verificato.
P(A|B) rappresenta la probabilità di osservare A dato che B si è già verificato.

La classificazione avviene secondo tre step:
1. Calcolo della probabilità della classe: le probabilità di classe è il numero di istanze che appartengono a quella classe diviso il numero di istanze totali;
2. Calcolo della probabilità condizionata: si applica il teorema di Bayes, per determinare le probabilità condizionate delle caratteristiche del problema;
3. Decisione: identificata nella classe che ottiene il valore di probabilità più elevata.
Tale algoritmo è stato implementato tramite la libreria sklearn che mette a disposizione il classificatore GaussianNB.

2.4.2 Decision Tree

Decision Tree (albero decisionale) è un algoritmo di classificazione che mira a creare un albero i cui nodi rappresentano un sotto-insieme di caratteristiche del problema e i cui archi rappresentano delle decisioni.
Ad ogni passo, viene calcolata, tra tutte quelle non ancora considerate, la caratteristica migliore (più discriminatoria) in base al suo grado di purezza attraverso l'information gain.
Quindi, per ogni attributo del dataset calcoleremo il suo gain:

$$Gain(D, A) = H(D) - \sum_{v \in values(A)} \frac{|D_v|}{|D|} * H(D_v)$$

A rappresenta l'attributo,
D il dataset,
D_v il sottoinsieme di D per cui l'attributo A ha il valore v;
$|D_v|$ è il numero di elementi di D_v,
$|D|$ è il numero di elementi nel dataset;
H(D) è l'entropia.

$$H(D) = -\sum_{c} p(c) * log_2 p(c)$$

L'entropia misura quanto rumore fa un attributo, ed è inversamente proporzionato alla sua purezza. La predizione di una nuova istanza viene effettuata navigando l'albero, effettuando delle decisioni sul percorso da seguire in base alle caratteristiche del nuovo dato fino ad arrivare a un nodo foglia che rappresenta la classe di appartenenza. Infatti, la costruzione dell'albero termina quando si definisce un nodo foglia per ogni classe.
Un nodo foglia si crea nel momento in cui si ha un sottoinsieme puro, cioè, un sottoinsieme di istanze che appartengono alla stessa classe.

Il classificatore è stato implementato utilizzando il DecisionTreeClassifier della libreria scikit-learn, impostando l'entropia come funzione per calcolare la qualità di una divisione, poichè il classificatore utilizza Gini di default. L'entropia è leggermente migliore rispetto Gini, anche se più costosa, ma avendo un dataset non troppo grande, si è preferito puntare ai risultati piuttosto che ai costi. I restanti parametri sono stati calcolati tramite l'algoritmo genetico visto in precedenza.

2.4.3 Random Forest

Random Forest è un algoritmo di classificazione formato da un largo numero di alberi decisionali che operano come un ensamble. Ogni singolo albero nella foresta produce una classe di predizione e la classe con il maggior numero di voti diventa la predizione del modello. Anche in questo caso si è deciso di utilizzare l'entropia come funzione per calcolare la qualità di una divisione. Per l'implementazione si è utilizzato il classificatore RandomForestClassifier della libreria sklearn.
Il numero di alberi nella foresta è di 100 mentre per gli altri parametri come max-depth, min-samples-split, min-samples-leaf e max-feature sono stati utilizzati i valori generati dall'algoritmo genetico.

2.5 Validazione

Chiaramente, non si può valutare un classificatore su dati che non si conoscono, altrimenti non si potrebbero misurare le sue prestazioni, ma, avendo un dataset di partenza etichettato, si può sfruttare questa conoscenza per capire come un classificatore classifica questi dati e di conseguenza poter misurare le sue prestazioni. Ma nello stesso tempo, se si addestrasse e si validasse il modello sullo stesso dataset si avrebbero risultati totalmente inaffidabili. Si può però dividere il dataset di partenza in maniera tale da considerare alcune delle istanze come non note. Si creano quindi due insiemi:

- Il **training set**, che sarà composto da istanze che l'algoritmo utilizzerà per l'addestramento;
- Il **test set**, che sarà composto dalle istanze per cui l'algoritmo addestrato dovrà predire la classe di appartenenza e sul quale verrà valutato la sua performance.

La tecnica adottata è stata la K-fold cross validation, la quale consente di dividere il dataset in k parti e a ogni iterazione, di usare 1 parte per la validazione e le k-1 parti restanti per l'addestramento; ovviamente, a termine della validazione, il modello viene eliminato. Al parametro k è stato assegnato valore 10, mentre la k fold è stata ripetuta 50 volte al fine di minimizzare il Mean Absolute Error.
Abbiamo implementato tale algoritmo tramite la libreria sklearn che mette a disposizione il modulo KFold.

2.6 Valutazione

A prescindere dalla procedura di validazione che si utilizzerà, si avrà bisogno di strumenti adatti per valutare la bontà delle predizioni. Per avere una rappresentazione dell'accuratezza dei 3 classificatori si fa impiego della **matrice di confusione**. La matrice di confusione è una matrice con cui poter indicare se ed in quanti casi il classificatore ha predetto correttamente o meno il valore di un'etichetta del test set.

	Istanze realmente positive	Istanze realmente negative
Istanze predette come positive	Veri positivi	Falsi positivi
Istanze predette come negative	Falsi negativi	Veri negativi

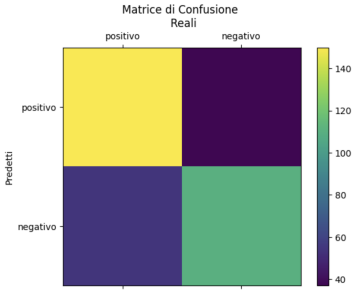
Sulla base dei valori della matrice di confusione, si procede a calcolare diverse metriche di valutazione; le principali sono Precision, Recall, Accuracy e F-score

$$\text{Precision} = \frac{TP}{TP+FP} \quad \text{Recall} = \frac{TP}{TP+FN} \quad \text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{F-score} = 2 * \frac{precision * recall}{precision + recall}$$

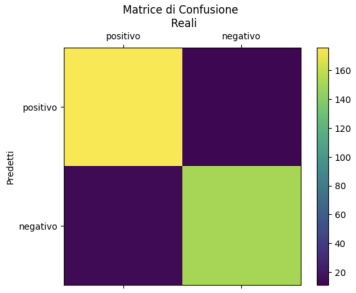
La **precision** indica il numero di predizioni corrette rispetto tutte le predizioni fatte dal classificatore. In altri termini, indica il numero di errori che ci saranno nella lista delle predizioni fatte dal classificatore.
La **recall** indica il numero di predizioni corrette rispetto a tutte le istanze positive di quella classe. In altri termini, indica quante istanze positive nell'intero dataset il classificatore può determinare.
L'**accuracy** indica il numero totale di predizioni corrette, considerando anche i veri negativi al numeratore. Questo potrebbe creare un problema di interpretazione nel caso di dataset sbilanciati dove il numero di casi positivi è molto basso.
F-score rappresenta la media ponderata della precision e della recall.

2.6.1 Valutazione dei 3 modelli

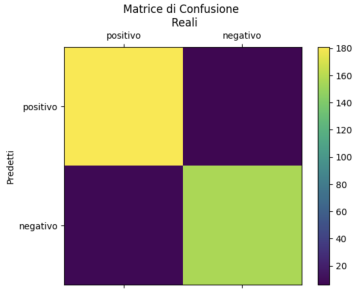
Sono state calcolate le matrice di confusione e le quattro metriche di valutazione sui 3 modelli per determinare e scegliere il migliore in termini di performance. Questo perchè per lo stesso problema ci sono più algoritmi possibili da poter utilizzare ma solo uno deve essere scelto. Ci si affida al metodo empirico, misurando l'errore commesso da ciascun algoritmo. Il metodo empirico e la misura dell'errore offrono una base di confronto tra i vari modelli, andando ad eseguire gli algoritmi nel contesto del problema, e misurando le loro prestazioni in base agli errori commessi.



(a) Bayes



(b) Decision Tree



(c) Random Forest

Dalle matrice di confusione sono state calcolate le quattro metriche di valutazione.

```
recall =  0.6707317073170732
precision =  0.7284768211920529
accuracy =  0.7293447293447294
f1 =  0.6984126984126985
```

(a) Bayes

```
recall =  0.9512195121951219
precision =  0.9454545454545454
accuracy =  0.9515669515669516
f1 =  0.9483282674772037
```

(b) Decision Tree

```
recall =  0.9451219512195121
precision =  0.9567901234567902
accuracy =  0.9544159544159544
f1 =  0.9509202453987731
```

(c) Random Forest

Così come si evince dalle metriche di valutazione, l'algoritmo migliore è il Random Forest, con un f1-score di 0.95, che ha portato alla sua scelta nonostante presenti degli svantaggi rappresentati nella seguente tabella:

Classificatore	Vantaggi	Svantaggi
Gaussian Naive Bayes	L'algoritmo è il più semplice da implementare poichè non serve definire dei parametri di input, quindi è anche il più veloce non dipendendo dall'algoritmo genetico	È il peggiore in quanto a performance poichè in fase di predizione considera tutte le caratteristiche indipendenti tra di loro
Decision Tree	Ha performance molto simili al random forest, in alcuni punti anche migliori, per esempio nella recall, nonostante sia molto meno costoso da realizzare rispetto ad esso	Piccole variazioni nei dati possono rendere instabile l'albero decisionale
Random Forest	È il migliore tra i tre, raggiungendo un'accuracy e una precision molto alta	È il più costoso e lento da realizzare andando a creare un insieme di alberi decisionali

3 Conclusioni

In questo progetto sono stati utilizzati metodi di apprendimento supervisionato su un dataset. In una prima parte ci si è occupati di tutte le fasi che comprendono la preparazione dei dati affinchè possano essere utilizzati dai classificatori. Nella seconda parte si è fatto un focus sulle varie tipologie di algoritmi di classificazione, sperimentando empiricamente allo scopo di determinare il migliore; tuttavia alcuni di questi algoritmi necessitavano di parametri da definire e proprio per questo si è andato a creare un algoritmo genetico che ne ottimizzasse le prestazioni in termini di precisione. È emerso che il random forest è l'algoritmo migliore in questo caso.
Nonostante gli ottimi risultati emerge comunque la necessità di migliorare il modello utilizzando dataset più precisi e vasti.

3.1 Repository Github

https://github.com/leotodisco/HeartCare_AI

References

[1] David Lapp, *Heart disease dataset*, https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset (2019).

[2] Lawrence O. Hall W. Philip Kegelmeyer Nitesh V. Chawla, Kevin W. Bowyer, *Smote: Synthetic minority over-sampling technique*, Journal of Artificial Intelligence Research 16 (2002).