DIPARTIMENTO
DI INFORMATICA

DOCUMENTAZIONE CASO DI STUDIO

ICON_HEART

Corso di studi: Ingegneria della Conoscenza

Docente: Nicola Fanizzi

Progetto realizzato da: Giacomo Pagliara

Matricola:722894

E-mail: g.pagliara8@studenti.uniba.it

Link al repository del progetto: https://github.com/giaco2000/Icon_HEART

INTRODUZIONE

Il progetto consiste nella realizzazione di un sistema esperto, basato su una base di conoscenza, utilizzato al fine di riconoscere e diagnosticare condizioni cardiache causate da malattie cardiovascolari (CVD). L'idea del progetto si basa su un sistema che una volta riconosciuti una serie di sintomi, ha la possibilità di diagnosticare una particolare condizione cardiaca in corso. Ovviamente, il processo non è banale e richiede l'ausilio di dottori ed esperti nel settore e nell'ambito medico, quindi il sistema non vuole sostituire la figura del medico ma può essere utilizzato come assistenza e supporto temporaneo. Ho preso in considerazione, durante lo sviluppo del sistema, di tre condizioni cardiache causate da CVD che sono: ATTACCO CARDIACO, ARITMIA e INSUFFICIENZA CARDIACA (spiegherò nel dettaglio nelle prossime slide i particolari). Mi sono soffermato sull'addestramento del sistema solo per alcuni sintomi chiave, consultando e informandomi su internet. Anche le condizioni cardiache implementate non sono tutte ma solo quelle più frequenti secondo dati statistici. Quando l'utente esprime i sintomi il sistema oltre a diagnosticare la condizione suggerisce una particolare azione che l'utente dovrebbe effettuare per tutelare la propria salute. Utilizzando il sistema, l'utente può anche leggere e consultare nel dettaglio i vari sintomi legati alle tre condizioni cardiache. La funzionalità è stata implementata con l'utilizzo di un Ontologia. Quest'ultima funzionalità va a chiudere la prima parte del progetto legata alla realizzazione del sistema esperto basato su conoscenza. Mentre, nella seconda ed ultima parte del progetto, viene presa in considerazione solo la condizione dell'insufficienza cardiaca, e riguarda l'analisi delle prestazioni di vari modelli di apprendimento supervisionato e non supervisionato utilizzati per predire, date una serie di caratteristiche (feature), la sopravvivenza di pazienti affetti proprio da insufficienza cardiaca.

INFORMAZIONI GENERALI

Per sviluppare il progetto è stato utilizzato il linguaggio di programmazione **Python** versione 3.10.2.

La base di conoscenza e le due fasi di apprendimento sono state realizzate attraverso l'IDE *Visual Studio Code*. Si consiglia di utilizzare quest'ultimo per avviare il sistema. Prima di avviare il sistema è necessario installare le librerie attraverso il file *requirements.txt* così come spiegato nel file *Read.me* su GitHub.

Per avviare la base di conoscenza è sufficiente avviare il file ***"main.py".*** Una volta avviato vengono proposte una serie di domande e l'utente risponde inserendo i sintomi che avverte. L'utente può rispondere inserendo l'intero che corrisponde alla risposta che vuole dare. Il sistema gestisce eventuali errori di input. Dopo aver risposto a tutte le domande il sistema diagnosticherà il problema suggerendo un'azione da compiere. Se non viene identificato nessun problema l'utente può riavviare il sistema di diagnostica o consultare i sintomi tramite l'Ontologia.

Mentre per avviare il sistema di apprendimento supervisionato e no, è sufficiente avviare il file ***"model_prediction.ipynb".*** Ho deciso di utilizzare file Jupyter Notebook per la fase di apprendimento dei classificatori perché permette di effettuare una programmazione interattiva e permette di esplorare le anteprime molto utili in una fase di apprendimento automatico.

Prima di avviare il file ".ipynb" per l'apprendimento automatico bisogna cambiare il percorso per poter leggere il set di dati, ovvero il file ***"heart.csv".*** Per fare ciò bisogna cambiare il percorso nella riga di codice presente nella prima cella del file:

```
data = pd.read_csv('C:/Users/giaco/Desktop/Icon_HEART/heart.csv')
data.head()
```

Quindi aggiungere il proprio percorso → *'C:/…./…./Icon_Heart/heart.csv'*

SCELTE PROGETTUALI:

INFORMAZIONI UTILIZZATE NEL SISTEMA DI DIAGNOSTICA:

Il termine "malattie cardiache" si riferisce a diversi tipi di condizioni cardiache. A volte le malattie cardiache possono essere "silenziose" e non diagnosticate fino a quando una persona non manifesta segni o sintomi di INFARTO, INSUFFICIENZA CARDIACA o ARITMIA.

Che cos'è l'INFARTO (o attacco di cuore)? → si verifica quando una parte del muscolo cardiaco non riceve abbastanza sangue.

Che cos'è l'ARITMIA? → si verifica quando il cuore batte troppo lentamente, troppo velocemente o in modo irregolare.

Che cos'è l'INSUFFICIENZA CARDIACA? → si verifica quando il cuore non riesce a pompare sangue e ossigeno a sufficienza per supportare altri organi del corpo.

Per il sistema di diagnostica mi sono soffermato sull'implementazione di alcuni *sintomi chiave* che riguardano queste tre condizioni cardiache, in particolare i sintomi sono:

- *Respiro corto*
- *Affaticamento*
- *Dolore o fastidio al torace*
- *Problemi di stomaco*
- *Palpitazioni*
- *Gonfiore nella parte inferiore del corpo*

Per il sintomo "*problemi di stomaco*" ho inglobato all'interno di quest'ultimo "*nausea, vomito, bruciore di stomaco e indigestione*".
Quindi "problemi di stomaco" comprende → {nausea, vomito, bruciore di stomaco, indigestione}.

Ciascuno di questi sintomi può rilevare o meno una determinata condizione cardiaca. Infatti, si verifica:

- L'INFARTO: quando sono presenti questi sintomi → *respiro corto, affaticamento, dolore o fastidio al torace, problemi di stomaco.*
- L'ARITMIA: quando sono presenti questi sintomi → *palpitazioni*
- L'INSUFFICIENZA CARDIACA: quando sono presenti questi sintomi → *respiro corto, affaticamento, gonfiore nella parte inferiore del corpo.*

Il sistema di diagnostica non si sofferma solo nell'individuazione di queste tre condizioni cardiache. Consultando internet e vari articoli scientifici, molte volte i sintomi dell'infarto possono essere confusi con quelli di un ATTACCO DI PANICO, con un PROBLEMA LEGATO ALLO STOMACO oppure con un eventuale PROBLEMA CARDIACO GENERICO.

L'attacco di panico, infatti, può creare diversi sintomi tra cui il *respiro corto* e il *dolore o fastidio al torace* che sono gli stessi tra quelli generati da un infarto. Il sistema è addestrato nel riconoscimento e nella differenziazione tra le due condizioni.

Siccome all'interno dei *sintomi chiave* ci sono i *problemi di stomaco,* ho voluto addestrare il sistema anche nell'individuazione e nella differenziazione tra condizioni cardiache e problemi legati allo stomaco.

Oppure, il sistema potrebbe rilevare determinati sintomi che non riesce ad associare né a una delle tre condizioni cardiache né a un problema legato allo stomaco e neanche a un attacco di panico, e diagnostica quindi che potrebbe esserci un problema cardiaco generico.

Infine, se il sistema non rivela alcun sintomo rilevante da associare a una delle tre condizioni cardiache, oppure a un problema legato allo stomaco, oppure a un attacco di panico oppure a un problema cardiaco generico, propone di avviare nuovamente il sistema di diagnostica.

SISTEMA DI DIAGNOSTICA

Il sistema mira a diagnosticare una delle tre *condizioni cardiache* oppure un *attacco di panico* oppure un *problema legato allo stomaco* oppure semplicemente un

problema cardiaco generico, basandosi sui dati forniti dall'utente, il quale viene coinvolto nella risposta a una serie di domande, mirate a individuare i sintomi e le possibili relazioni che questi sintomi hanno con le problematiche citate al primo rigo di questa sezione, al fine di suggerire una particolare azione da compiere per tutelare la salute. Se il sistema non riconosce nessuna problematica, l'utente può riavviare il sistema di diagnostica oppure consultare i sintomi tramite l'Ontologia. Più avanti con le slide verranno illustrati esempi di output tratti dal terminale.

La procedura adottata per un sistema di diagnostica è logicamente un ***backward chaining***. In un linguaggio logico (come può essere il *Prolog*) ci limiteremmo a chiedere i sintomi avvertiti, e il sistema, notando che ha degli atomi dal valore che non conosce, andrebbe in automatico a porre le domande a run-time sugli atomi che abbiamo definito '*askable'*; tuttavia per le limitazioni della libreria utilizzata (**Experta**) siamo costretti ad operare con il ***forward chaining*** andando a porre prima le domande all'utente, le cui risposte sono considerate come fatti osservati, per poi determinare se è presente o meno una determinata problematica(cardiaca o non cardiaca) a seconda della regola considerata vera.

L'idea generale si basa su una regola di derivazione, una forma generalizzata della regola di inferenza chiamata ***modus ponens***:

Se "$h \leftarrow a1 \wedge \ldots \wedge am$" è una clausola definita proposizionale nella base di conoscenza e ogni ai è stato derivato, allora h può essere derivato.

Dove:

- h è la **testa** dell'atomo
- $a1 \wedge \ldots \wedge am$ è il **corpo** della clausola, formato da ai **atomi**

Se $m > 0$ la clausola è detta regola, se $m = 0$ il corpo è vuoto e la clausola è detta clausola atomica o **fatto**, e tutte le clausole atomiche nella base di conoscenza sono sempre derivate in maniera diretta.

BASE DI CONOSCENZA

Nella base di conoscenza le regole sono state formulate prendendo spunto da articoli/blog scientifici che riguardano la medicina in particolare che riguardano la cardiologia. I vari blog scientifici possono essere consultati nella cartella *"Blog".* Ho inserito le caratteristiche reali più rilevanti, ovviamente la base di conoscenza potrebbe essere migliorata ma soprattutto ampliata tramite l'aiuto di esperti nel settore come ad esempio cardiologi.

LEGENDA PAROLE:

- ***dft = dolore o fastidio al torace***
- ***attacco cardiaco = infarto***
- ***no_heart significa che la problematica individuata è legata allo stomaco e non al cuore***

La base di conoscenza è strutturata nel seguente modo:

Regole:

- aritmia ← respiro_corto=2 ∧ affaticamento=2 ∧ dft=2 ∧ problemi_stomaco=2 ∧ palpitazioni=1 ∧ gonfiore=2
- keep_calm ← aritmia

- attacco_cardiaco ← respiro_corto=1 ∧ affaticamento=1 ∧ dft=1 ∧ problemi_stomaco=1 ∧ palpitazioni=2 ∧ gonfiore=2
- call_118 ← attacco_cardiaco

- no_heart ← respiro_corto=1 ∧ affaticamento=1 ∧ problemi_stomaco=1 ∧ dft=2
- no_heart ← respiro_corto=1 ∧ affaticamento=2 ∧ problemi_stomaco=1 ∧ dft=2
- no_heart ← respiro_corto=2 ∧ affaticamento=1 ∧ problemi_stomaco=1 ∧ dft=2
- no_heart ← respiro_corto=2 ∧ affaticamento=2 ∧ problemi_stomaco=1 ∧ dft=2
- stomach ← no_heart

- attacco_di_panico ← respiro_corto=1
- attacco_di_panico ← dft=1 ∧ respiro_corto=1
- respira ← attacco_di_panico
- insufficienza_cardiaca ← respiro_corto=1 ∧ affaticamento=1 ∧ gonfiore=1 ∧ problemi_stomaco=2 ∧ dft=2 ∧ palpitazioni=1
- insufficienza_cardiaca ← respiro_corto=1 ∧ affaticamento=1 ∧ gonfiore=1 ∧ problemi_stomaco=2 ∧ dft=2 ∧ palpitazioni=2

- immediate_assistance ← insufficienza_cardiaca

- no_malattia ← affaticamento=2 ∧ dft=2 ∧ respiro_corto=2
- no_malattia ← affaticamento=1 ∧ dft=2 ∧ problemi_stomaco=2 ∧ gonfiore=2
- no_malattia ← affaticamento=1 ∧ dft=2 ∧ problemi_stomaco=2 ∧ gonfiore=1
- no_malattia ← respiro_corto=1 ∧ dft=2 ∧ affaticamento=1 ∧ gonfiore=2 ∧ problemi_stomaco=2
- no_malattia ← problemi_stomaco=2 ∧ palpitazioni=2 ∧ gonfiore=2 ∧ respiro_corto=2 ∧ dft=2 ∧ affaticamento=2
- no_sintomi ← no_malattia

- problema_cardiaco_generico ← dft=1
- problema_cardiaco_generico ← respiro_corto=2 ∧ palpitazioni=1 ∧ affaticamento=2 ∧ dft=1 ∧ gonfiore=1
- problema_cardiaco_generico ← respiro_corto=2 ∧ palpitazioni=1 ∧ affaticamento=2 ∧ dft=1 ∧ gonfiore=2
- problema_cardiaco_generico ← respiro_corto=1 ∧ affaticamento=1 ∧ dft=1 ∧ problemi_stomaco=1 ∧ palpitazioni=1 ∧ gonfiore=1
- sii_specifico ← problema_cardiaco_generico

Fatti:

Fatti dichiarati dall'utente rispondendo alle domande:

- respiro_corto:
  - 1 = si
  - 2 = no
- affaticamento:
  - 1 = si
  - 2 = no
- dft (dolore o fastidio al torace):
  - 1 = si
  - 2 = no

- problemi_stomaco:
  - ❖ 1 = si
  - ❖ 2 = no
- palpitazioni:
  - ❖ 1 = si
  - ❖ 2 = no
- gonfiore (gonfiore nella parte inferiore del corpo):
  - ❖ 1 = si
  - ❖ 2 = no

Fatti derivati da regole (rappresentano i problemi emersi, hanno dominio binario):

- aritmia
- attacco_cardiaco
- no_heart
- attacco_di_panico
- insufficienza_cardiaca
- no_malattia
- problema_cardiaco_generico

Le regole permettono di individuare sei diverse problematiche ovvero: **aritmia**, **attacco cardiaco**, **problema allo stomaco**, **attacco di panico**, **insufficienza cardiaca** o un **problema cardiaco generico.** Per ognuna di queste regole, derivate a partire dai fatti, verrà derivata una possibile azione da compiere suggerita all'utente per tutelare la salute. Per le regole che non identificano alcun sintomo rilevante da associare alle problematiche citate sopra verrà chiesto all'utente di riavviare il sistema di diagnostica oppure consultare e leggere le descrizioni dei vari sintomi nel dettaglio tramite l'Ontologia oppure terminare il programma.

Le regole sono derivate prendendo in considerazione i fatti. Questi si dividono in fatti dichiarati prendendo in considerazione le risposte dell'utente (**osservazioni**) e fatti derivati da regole. Le osservazioni sono: *respiro_corto*, *affaticamento*, *dft, problemi_stomaco, palpitazioni, gonfiore*. Queste osservazioni verranno spiegate nel dettaglio avviando, dal menu principale del sistema, l'Ontologia.

I fatti derivati hanno valore binario (True, False), quest'ultimi nel momento in cui la regola corrispondente risulta vera vengono dichiarati veri per identificare il problema e suggerire una particolare azione da compiere per tutelare la salute e

sono*: aritmia* che rappresenta una delle tre condizioni cardiache e quindi bisognerebbe contattare il proprio medico di base e cercare di stare tranquilli, *attacco_cardiaco* che rappresenta una delle tre condizioni cardiache e quindi bisognerebbe recarsi immediatamente presso un pronto soccorso o chiamare il 118, *no_heart* che rappresenta una problematica legata allo stomaco e non riguardante il cuore e quindi bisognerebbe effettuare una gastroscopia*, attacco_di_panico* che rappresenta una condizione di ansia non legata al cuore ma che spesso viene confusa con un attacco cardiaco e quindi bisognerebbe iniziare a respirare lentamente e chiedere aiuto, *insufficienza_cardiaca* che rappresenta una delle tre condizioni cardiache e quindi bisognerebbe contattare il medico di base e la guardia medica per ricevere assistenza immediata, *no_malattia* che indica che non è stata identificata nessuna problematica dal sistema, *problema_cardiaco_generico* che rappresenta la presenza di un problema cardiaco generico e quindi bisognerebbe essere più specifici nel valutare i sintomi. Infine, ci sono fatti come *question* e *order_question* utilizzati per gestire il flusso delle domande.

SISTEMA ESPERTO E SUA IMPLEMENTAZIONE

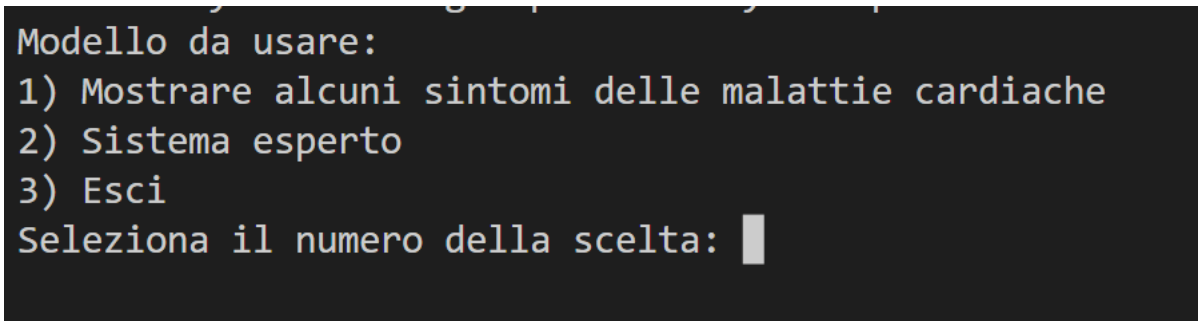
Un **sistema esperto** è un'applicazione dell'intelligenza artificiale che vede un programma cercare di risolvere dei problemi, provando a riprodurre i comportamenti di persone esperte in un determinato campo di attività, per fare inferenza.
È principalmente composto da una "***knowledge base***" (o base di conoscenza), che rappresenta e memorizza fatti e regole riguardanti il mondo, un "***inference engine***" che si occupa di mettere in pratica le nozioni apprese dalla base di conoscenza, e da una "*user interface*" che permette una facile interazione tra il sistema e l'utente.

L' implementazione del sistema si basa su un sistema esperto realizzato tramite il linguaggio **Python** utilizzando la libreria **Experta**, che permette di associare un insieme di fatti ad un insieme di regole relativi agli stessi, ed eseguire azioni in base alle regole di abbinamento. Le regole sono formate da due componenti chiamati **LHS** (Left-Hand-Side) e **RHS** (Right-Hand-Side). Il primo descrive le condizioni che si devono verificare affinché la regola venga applicata, mentre il secondo è l'insieme di azioni che vengono compiute quando viene applicata la regola. Nel momento in cui si avvia il sistema, un fatto denominato "*question*" viene impostato a *True* per iniziare a porre all'utente le domande per comprendere i sintomi riscontrati dall'utente. A seconda della domanda l'utente può rispondere inserendo il numero della risposta che vuole dare ovvero "si" oppure "no". Il sistema è tollerante agli errori gestendo i casi in cui l'utente immette input errati.

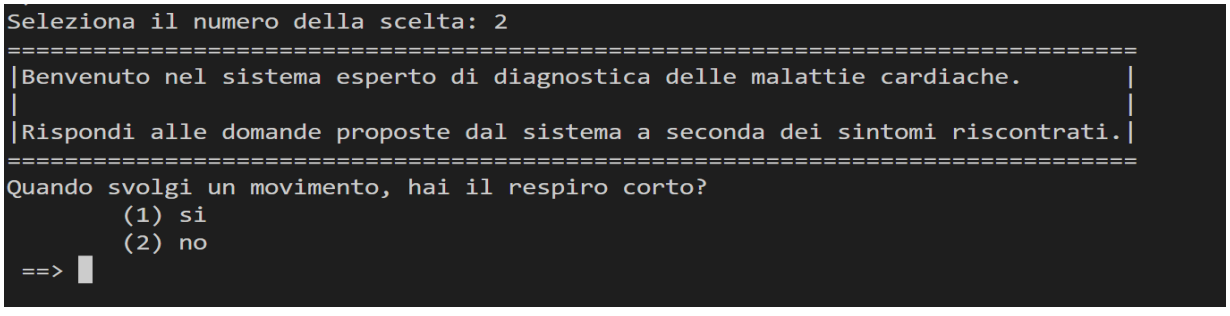
Per ogni sintomo è stata associata una regola che, quando viene applicata, si occupa di domandare all'utente se riscontra il sintomo e, in base alla sua risposta, il fatto relativo al sintomo viene avvalorato. Nel momento in cui tutti i fatti relativi ai sintomi sono avvalorati, il sistema applica le regole opportune per derivare le problematiche emerse. Quando un problema è stato derivato viene impostato a *True* il fatto relativo e di conseguenza verrà derivata la regola per suggerire una particolare azione da effettuare all'utente a seconda della problematica emersa.
Ad esempio, se l'utente risponde dicendo che avverte il respiro corto oppure se avverte il respiro corto e dolore o fastidio al torace insieme, verrà derivata la regola che indica che l'utente sta avendo un attacco di panico, avvalorando il fatto corrispondente al problema e di conseguenza suggerendo all'utente di iniziare a respirare lentamente e di chiedere aiuto.
Ad esempio, se l'utente risponde dicendo che avverte il respiro corto, un senso di affaticamento, gonfiore nella parte inferiore del corpo, palpitazioni e non avverte problemi allo stomaco e non avverte dolore o fastidio al torace, verrà derivata la regola che indica che l'utente potrebbe essere affetto da insufficienza cardiaca, avvalorando il fatto corrispondente al problema e di conseguenza suggerendo all'utente di contattare il medico e la guardia medica per ricevere assistenza.

SCHERMATA INIZIALE ED ESEMPI TRATTI DAL TERMINALE

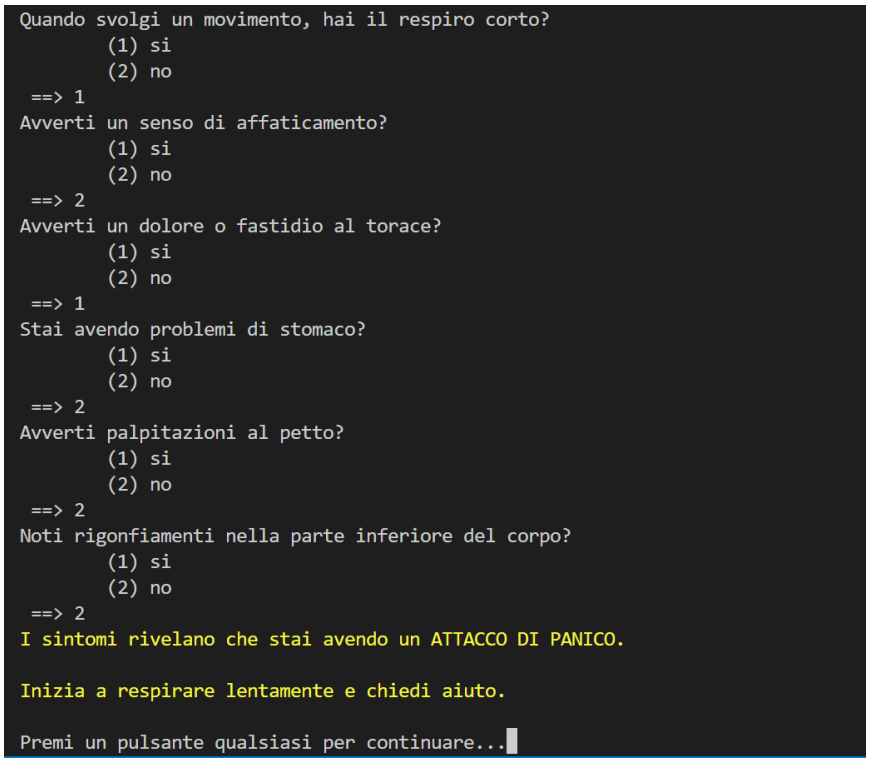
Questa immagine rappresenta la schermata inziale del sistema esperto:



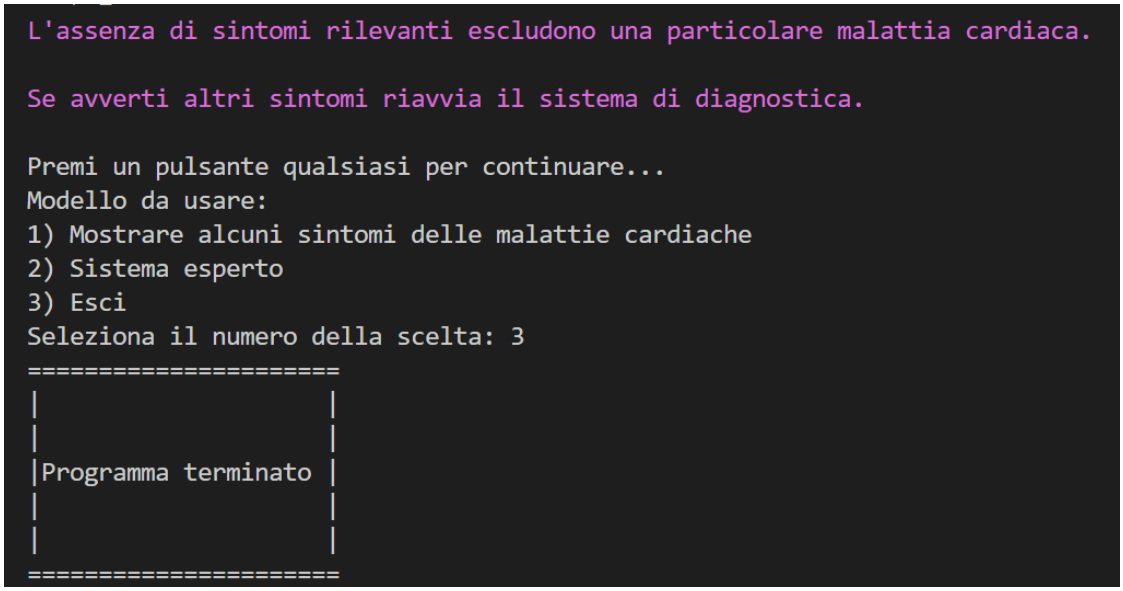
Inserendo l'opzione "2" si avvierà il sistema di diagnostica:



Una volta risposto alle domande il sistema risponderà con il problema identificato e un'azione suggerita:



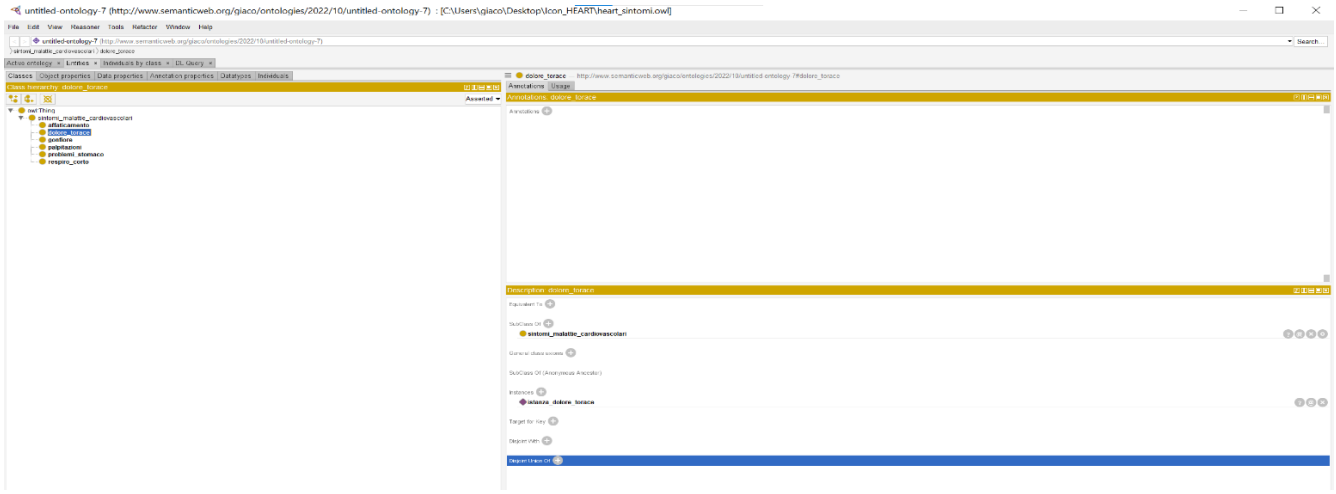
Se il sistema non rivela alcun sintomo rilevante, l'utente può riavviare il sistema di diagnostica oppure leggere le descrizioni dei sintomi tramite Ontologia oppure uscire dal programma:



ONTOLOGIE

In informatica, un'ontologia `e una rappresentazione formale, condivisa ed esplicita di una concettualizzazione di un dominio di interesse. Più nel dettaglio, si tratta di una teoria assiomatica del primo ordine esprimibile in una logica descrittiva. Il termine ontologia formale è entrato in uso nel campo dell'intelligenza artificiale e della rappresentazione della conoscenza, per descrivere il modo in cui diversi schemi vengono combinati in una struttura dati contenente tutte le entità rilevanti e le loro relazioni in un dominio. I programmi informatici possono poi usare l'ontologia per una varietà di scopi, tra cui il ragionamento induttivo, la classificazione, e svariate tecniche per la risoluzione di problemi. L'ontologia è stata creata mediante il tool **protégé**, e la lettura avviene tramite la libreria python **Owlready2**.

Questa immagine rappresenta la creazione dell'ontologia su protégé:



Dal menu iniziale del sistema esperto inserendo l'opzione numero "1" si avvierà l'Ontologia nel seguente modo:

```
Modello da usare:
1) Mostrare alcuni sintomi delle malattie cardiache
2) Sistema esperto
3) Esci
Seleziona il numero della scelta: 1
Sintomo [1]: Nome: affaticamento
Sintomo [2]: Nome: dolore_torace
Sintomo [3]: Nome: gonfiore
Sintomo [4]: Nome: palpitazioni
Sintomo [5]: Nome: problemi_stomaco
Sintomo [6]: Nome: respiro_corto

Seleziona il sintomo di cui vuoi conosere la descrizione, inserisci il numero del sintomo
6
Sintomo: respiro_corto, descrizione: "Il fiato corto (o dispnea) è un sintomo che indica una soggettiva difficoltà respiratoria.Le cause più frequenti del fiato corto comprendono crisi
 d'asma, polmonite, broncopneumopatia cronica ostruttiva (BPCO) e infarto miocardico.Si manifesta sotto forma di respirazione difficile, paragonabile alla sensazione di non poter rifia
tare oppure come affanno. Può presentarsi in modo graduale o improvvisamente."
Modello da usare:
1) Mostrare alcuni sintomi delle malattie cardiache
2) Sistema esperto
3) Esci
Seleziona il numero della scelta:
```

<u>Dall'Ontologia viene creato un dizionario con i sintomi e la relativa descrizione</u>.

APPRENDIMENTO AUTOMATICO

Il dataset utilizzato in questo report è il "Heart Failure Clinical Records" disponibile al seguente link: UCI Machine Learning Repository: Heart failure clinical records Data Set

Le malattie cardiovascolari (CVD) sono la prima causa di morte a livello globale, con una stima di 17,9 milioni di vittime ogni anno, che rappresentano il 31% di tutti i decessi nel mondo.

L'insufficienza cardiaca è un evento comune causato da CVD e questo set di dati contiene 12 caratteristiche che possono essere utilizzate per prevedere la mortalità per insufficienza cardiaca. Le persone con malattie cardiovascolari o ad alto rischio cardiovascolare (per la presenza di uno o più fattori di rischio come ipertensione, diabete, iperlipidemia o malattie già accertate) necessitano di una diagnosi precoce e di una gestione in cui un modello di machine learning può essere di grande aiuto. Questo set di dati contiene le cartelle cliniche di 299 pazienti con insufficienza cardiaca raccolti presso il Faisalabad Institute of Cardiology e presso l'Allied Hospital di Faisalabad (Punjab, Pakistan), nel periodo aprile-dicembre 2015.

Il set di dati contiene 13 features, che riportano informazioni cliniche, corporee e sullo stile di vita. Le prime 12 features sono le features di input e la tredicesima è la variabile target:

1. Età: Età del paziente
2. Anemia: diminuzione dei globuli rossi o dell'emoglobina (Booleana)- 0=No, 1=Sì
3. Creatinina fosfochinasi: livello dell'enzima CPK nel sangue (mcg/L)
4. Diabete: Se il paziente ha il diabete (Booleano)- 0=No, 1=Sì
5. Frazione di eiezione: percentuale di sangue che lascia il cuore ad ogni contrazione (percentuale)
6. Alta pressione sanguigna: se il paziente ha ipertensione (booleano)- 0=No, 1=Sì
7. Piastrine: Piastrine nel sangue (kilopiastrine/mL)
8. Creatinina sierica: livello di creatinina sierica nel sangue (mg/dl)
9. Sodio sierico: livello di sodio sierico nel sangue (mEq/L)
10. Sesso: donna o uomo (binario) - 0=femmina, 1=maschio
11. Fumo: se il paziente fuma o meno (Booleano)- 0=No, 1=Sì

12. Tempo: periodo di follow-up (giorni)
13. EVENTO DI MORTE: Se il paziente è deceduto durante il periodo di follow-up (Booleano)- 0=No, 1=Yes. Questa è la variabile target.

ANALISI DEI DATI

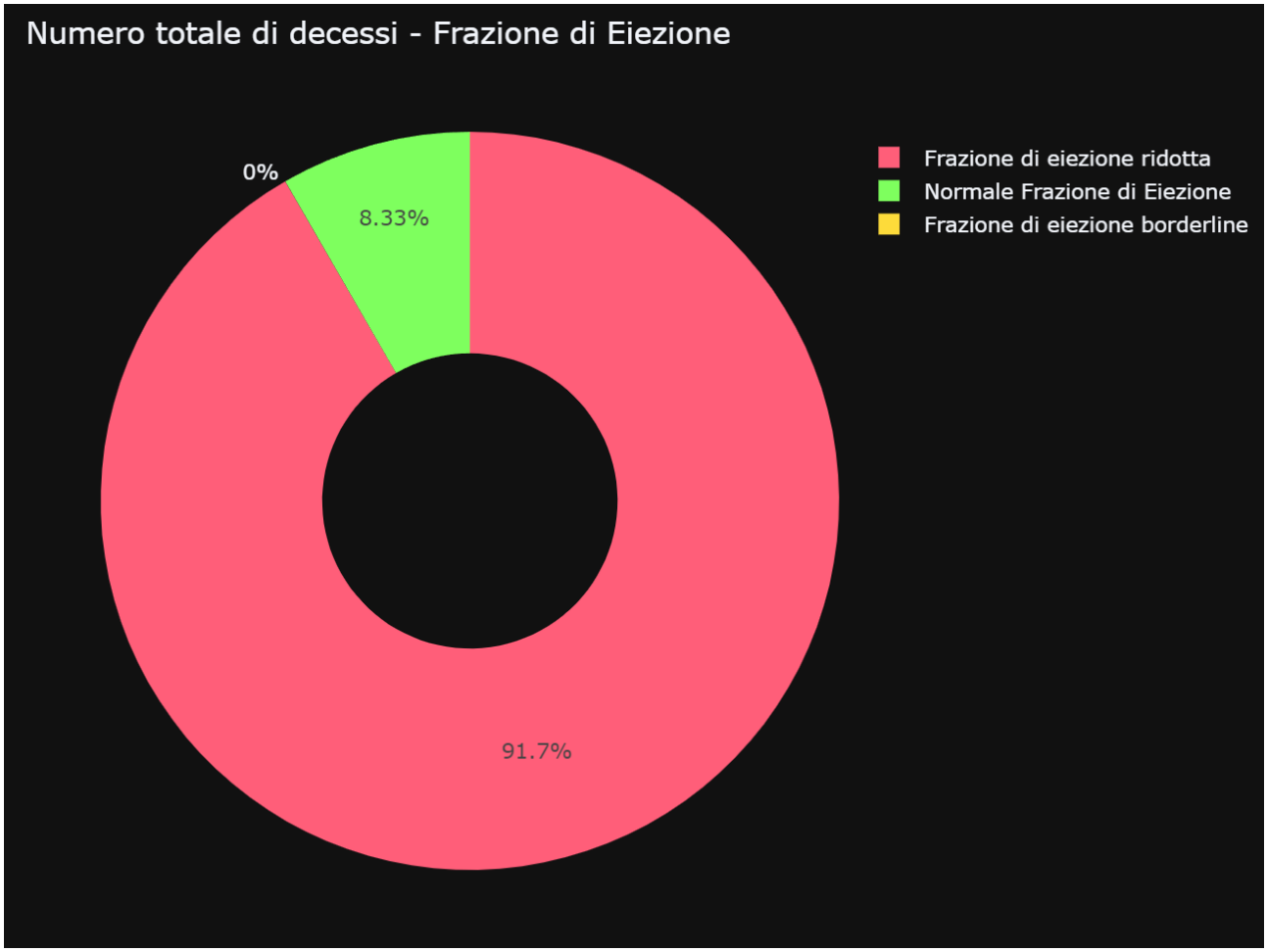
All'interno del dataset non sono presenti valori nulli.

All'interno del codice sono stati implementati i valori reali delle features del dataset.

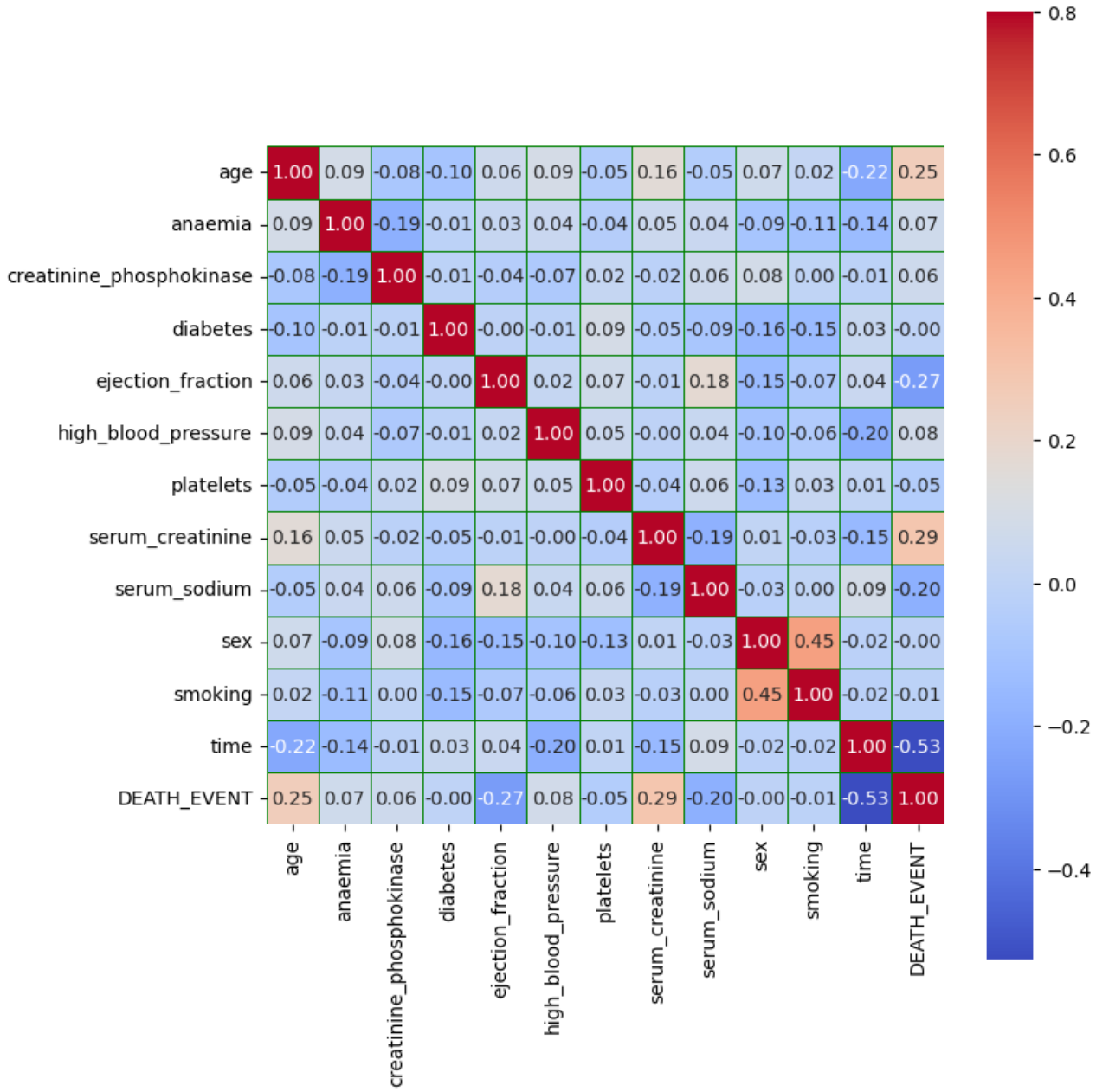
Ad esempio, i valori normali delle piastrine devono essere compresi tra 150000 e 400000. E cosi via per ciascuna feature che indica il valore di un enzima del sangue.

Sono stati implementati i vari grafici che rappresentano, per ciascuna feature, il numero totale di decessi.

Questa immagine rappresenta il grafico per il numero totale di decessi considerando la frazione di eiezione:



E’ stata generata una heatmap che ci permetterà di visualizzare quali sono i sintomi più comuni correlati tra di loro:



Dalla heatmap si nota, dunque, che ci sono sintomi mediamente o altamente correlati tra di loro. Se ne deduce quindi che per gli esempi del dataset che avranno valore 1 in corrispondenza di feature mediamente o altamente correlate tra di loro, la classificazione sarà più attendibile e veritiera. Dato che l’obiettivo del nostro lavoro è quello di realizzare un sistema che possa essere di supporto ad un medico (e non ad uso e consumo di un privato), abbiamo concentrato i nostri sforzi nell’addestrare una serie di classificatori e nel determinarne i migliori iper-parametri al fine di poter fornire al medico una serie di prognosi piuttosto che una sola.

APPRENDIMENTO SUPER VISIONATO

Si è optato per l'addestramento dei classificatori solo sull'età, frazione di eiezione, creatinina sierica, tempo e pressione del sangue.

Il dataset è stato quindi suddiviso in due insiemi: per la fase di training e per la fase di testing con una proporzione rispettivamente del 70% e del 30%.

La scelta dei classificatori è ricaduta su:

- Decision Tree
- K-Nearest Neighbors
- Logistic Regression
- Random Forest
- Neural Network

La ricerca degli iperparametri è stata effettuata tramite la Grid Search, andando a definire per ogni classificatore una lista di quelli che sono gli attributi e per ogni attributo è stato specificato un insieme di valori assumibili. Ciò che fa la Grid Search è testare il classificatore con ogni possibile combinazione degli attributi specificatigli.

Per ogni combinazione effettua una k-fold cross validation (con un valore k settato arbitrariamente a cinque in quanto è la profondità massima che si può raggiungere) e calcola lo score di accuratezza relativo all'iterazione corrente.

Una volta che sono state testate tutte le combinazioni viene restituita la combinazione con l'accuracy score maggiore. La scelta per la valutazione dei classificatori è ricaduta su: Matrice di confusione e 0-1 Loss per la valutazione delle classificazioni rispetto alle ground truth

Classification Report per visualizzare precision, recall, f1-score, accuracy, macro average e weighted average per ogni classificatore

Cross Validation Score per calcolare lo score della cross-validation

Learning Curve per valutare l'accuratezza rispetto all'aumentare del numero di esempi sottoposti ai classificatori.

RANDOM FOREST

L'algoritmo dell'estimatore Random Forest è basato sull'addestramento di un numero N di Decision Tree, ognuno dei quali effettua una classificazione per ogni esempio. Quando tutti gli alberi (o più precisamente tutta la foresta) hanno

classificato l'esempio, si effettua una conta di qual è stata la classe maggiormente stimata e la si assume come predizione della foresta. Questo per l'appunto viene effettuato per tutti gli esempi. Dati gli ottimi risultati ottenuti con la Grid Search abbiamo ripetuto la ricerca degli iper-parametri anche per questo estimatore ed è risultato che il numero di alberi più adatto per la foresta è pari a 70, per ogni albero che compone la foresta si evince che il miglior criterio sia "entropy" (diversamente dagli alberi di decisione dove è risultato più valido il criterio "gini"). La formula relativa all'entropia è la seguente:

$$Entropy = \sum_{x} -P(x) * \log_2 P(x)$$

I parametri migliori indicati dal Grid Search sono:

```
{'bootstrap': True,
 'criterion': 'entropy',
 'max_depth': 8,
 'max_features': 'sqrt',
 'n_estimators': 70}
```

Classification report senza ottimizzazione dei parametri:

```
Train score:  0.9916317991631799
Test score:  0.8
Cross Validated Score:  0.8621212121212121
Standard Deviation:  0.0800166419788539
Variance:  0.006402662993572084
0-1 Loss:  0.19999999999999996
Accuracy: 0.8921694480102697
              precision    recall  f1-score   support

           0       0.84      0.88      0.86        41
           1       0.71      0.63      0.67        19

    accuracy                           0.80        60
   macro avg       0.77      0.75      0.76        60
weighted avg       0.80      0.80      0.80        60
```

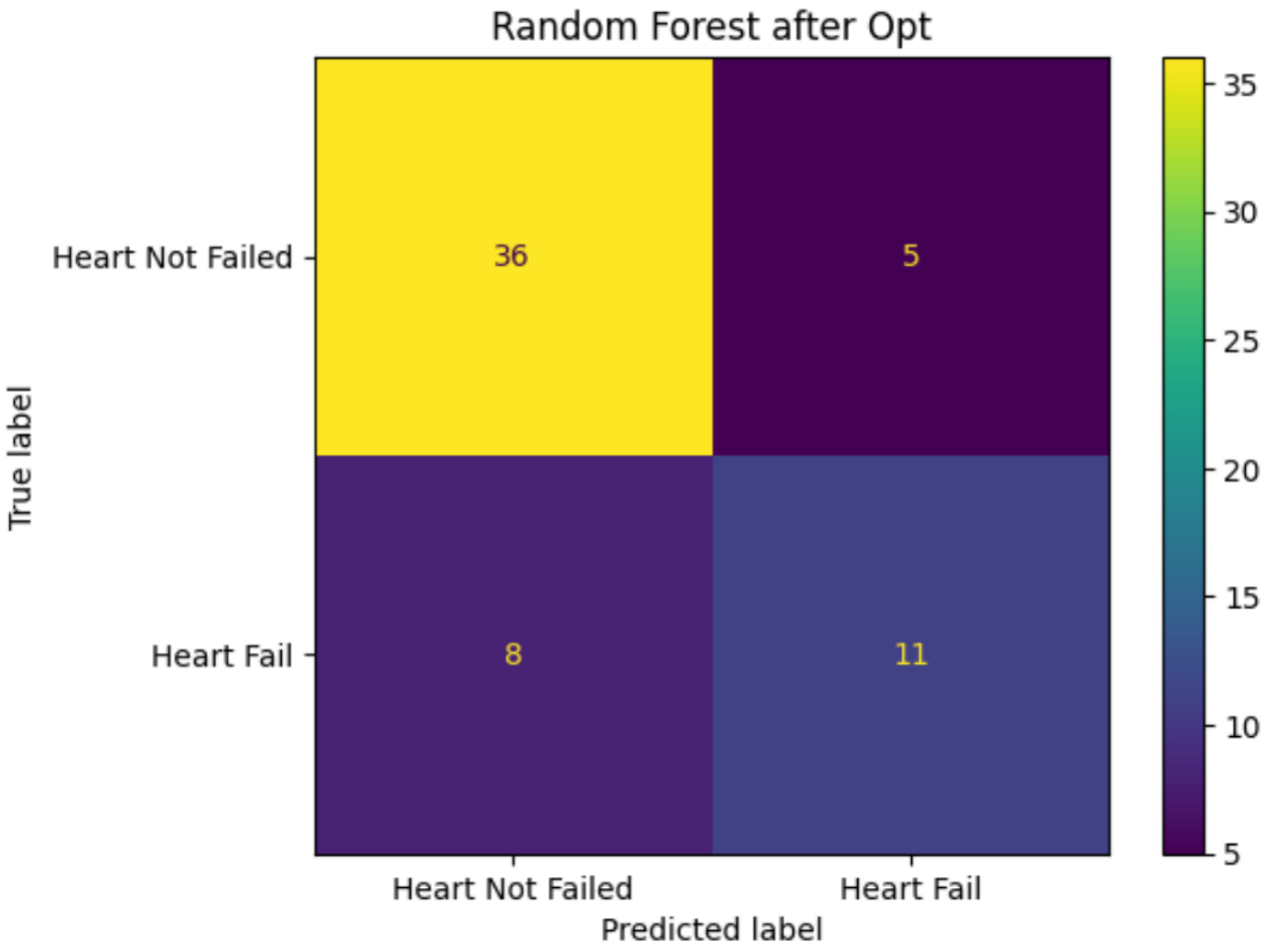
Classification report senza ottimizzazione dei parametri:

```
Train score:  0.9790794979079498
Test score:  0.8666666666666667
Cross Validated Score:  0.8454545454545455
Standard Deviation:  0.07071879412190886
Variance:  0.00500114784205693
0-1 Loss:  0.1333333333333333
Accuracy: 0.9679075738125803
              precision    recall  f1-score   support

           0       0.90      0.90      0.90        41
           1       0.79      0.79      0.79        19

    accuracy                           0.87        60
   macro avg       0.85      0.85      0.85        60
weighted avg       0.87      0.87      0.87        60
```

Per avere una rassicurazione circa il corretto apprendimento del classificatore, è stata creata una matrice di confusione, la quale ci ha fornito una rappresentazione grafica dell'accuratezza del classificatore:



DECISION TREE

Il funzionamento degli alberi di decisione prevede che il valore di una feature obiettivo venga classificato sulla base di una serie di regole di decisione basate sui dati di input a disposizione. Nello specifico ogni nodo interno dell'albero indica una condizione ed i valori derivati dagli esempi di input costituiscono dei sottoalberi. Le foglie dell'albero invece contengono il valore della feature obiettivo. Per quanto riguarda la nostra implementazione di questo modello di classificazione, a seguito degli insoddisfacenti risultati iniziali, abbiamo effettuato subito una Grid Search (operazione che abbiamo effettuato per prima a partire da questo classificatore) dalla quale è risultato che il miglior criterio tra "entropy" e "gini" è proprio quest'ultimo. Il criterio "gini" permette di minimizzare la probabilità di classificazioni errate e la sua formula è:

$$Gini = 1 - \sum_{x} P(x)^2$$

I parametri migliori indicati dal Grid Search sono:

```
{'criterion': 'gini', 'max_depth': 1}
```

Classification report senza ottimizzazione dei parametri:

```
Train score:  1.0
Test score:  0.7666666666666667
Cross Validated Score:  0.7742424242424242
Standard Deviation:  0.10202270223959047
Variance:  0.010408631772268136
0-1 Loss:  0.2333333333333328
Accuracy: 0.7163029525032092
              precision    recall  f1-score   support

           0       0.81      0.85      0.83        41
           1       0.65      0.58      0.61        19

    accuracy                           0.77        60
   macro avg       0.73      0.72      0.72        60
weighted avg       0.76      0.77      0.76        60
```

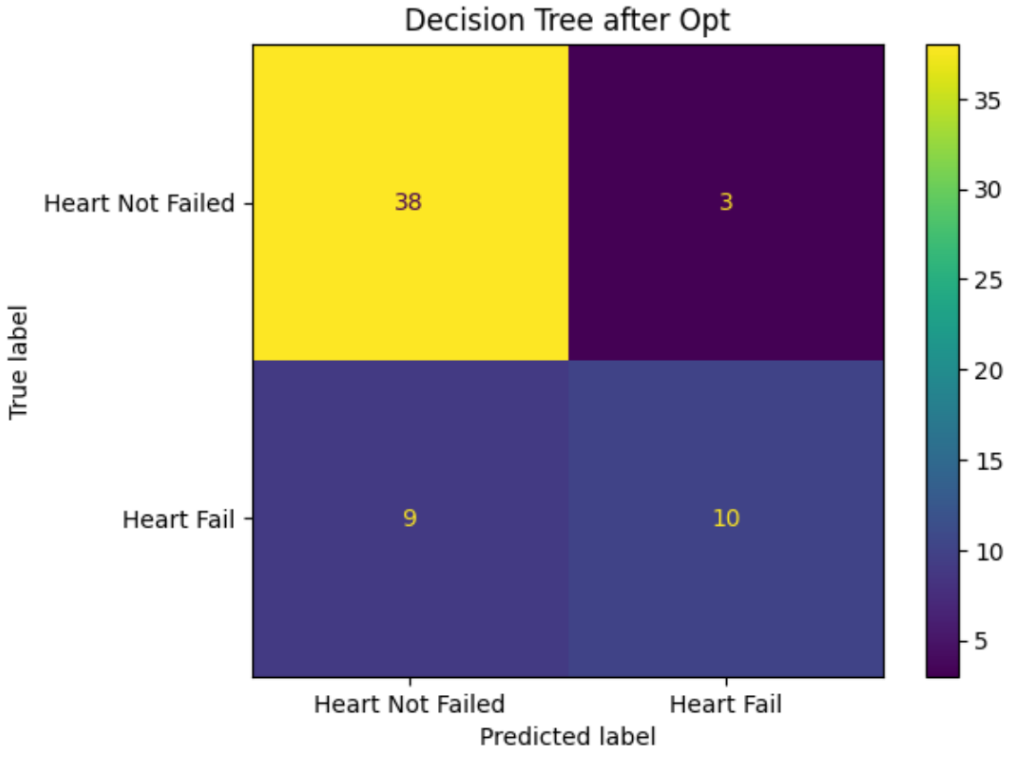
Classification report con ottimizzazione dei parametri:

```
Train score:  0.8577405857740585
Test score:  0.8
Cross Validated Score:  0.8496212121212123
Standard Deviation:  0.08551802257706428
Variance:  0.007313332185491276
0-1 Loss:  0.19999999999999996
Accuracy: 0.7265725288831836
              precision    recall  f1-score   support

           0       0.81      0.93      0.86        41
           1       0.77      0.53      0.62        19

    accuracy                           0.80        60
   macro avg       0.79      0.73      0.74        60
weighted avg       0.80      0.80      0.79        60
```

Per avere una rassicurazione circa il corretto apprendimento del classificatore, è stata creata una matrice di confusione, la quale ci ha fornito una rappresentazione grafica dell'accuratezza del classificatore:



K-Nearest Neighbors

Il funzionamento dell'algoritmo relativo al classificatore K-NN si basa sulla semplice memorizzazione degli esempi del dataset (comprendendo la/le feature obiettivo), senza che venga appreso un modello. La classificazione avviene confrontando il nuovo esempio con un insieme formato per l'appunto da k vicini che voteranno per decretare la classe di appartenenza del nuovo esempio. La votazione può avvenire come calcolo della moda, della media o a seguito dell'interpolazione dei k vicini.

I parametri migliori indicati dal Grid Search sono:

```
{'algorithm': 'kd_tree', 'n_neighbors': 10, 'p': 1, 'weights': 'distance'}
```

Classification report senza ottimizzazione dei parametri:

```
Train score:  0.8661087866108786
Test score:  0.7666666666666667
Cross Validated Score:  0.7492424242424243
Standard Deviation:  0.10812121636771292
Variance:  0.011690197428833793
0-1 Loss:  0.23333333333333328
Accuracy: 0.8260590500641849
              precision    recall  f1-score   support

           0       0.75      0.98      0.85        41
           1       0.86      0.32      0.46        19

    accuracy                           0.77        60
   macro avg       0.81      0.65      0.66        60
weighted avg       0.79      0.77      0.73        60
```

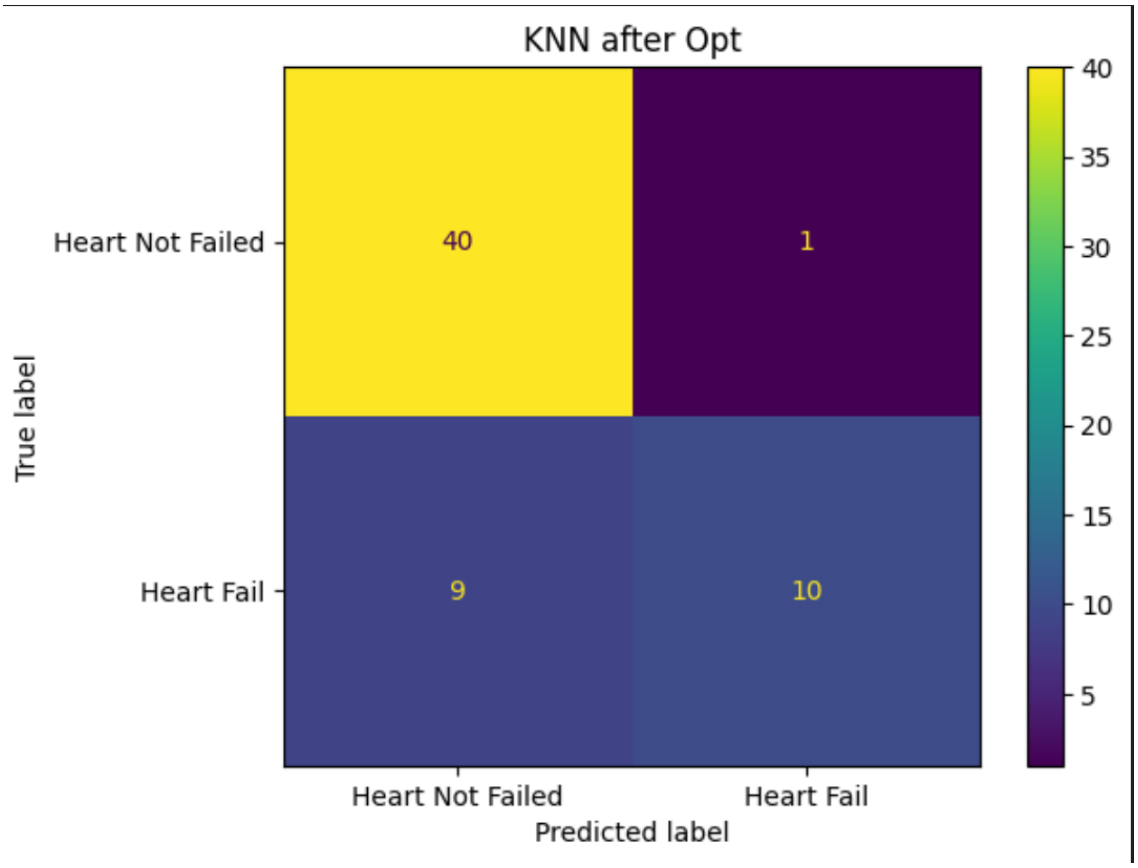
Classification report con ottimizzazione dei parametri:

```
Train score:  1.0
Test score:  0.8333333333333334
Cross Validated Score:  0.8121212121212121
Standard Deviation:  0.1012604221528934
Variance:  0.010253673094582184
0-1 Loss:  0.16666666666666663
Accuracy: 0.876765083440308
              precision    recall  f1-score   support

           0       0.82      0.98      0.89        41
           1       0.91      0.53      0.67        19

    accuracy                           0.83        60
   macro avg       0.86      0.75      0.78        60
weighted avg       0.85      0.83      0.82        60
```

Per avere una rassicurazione circa il corretto apprendimento del classificatore, è stata creata una matrice di confusione, la quale ci ha fornito una rappresentazione grafica dell'accuratezza del classificatore:



LOGISTIC REGRESSION

Il modello di classificazione della regressione logistica si basa su dei pesi di una funzione lineare appiattita dalle sigmoidee, minimizzando un errore su E. È importante specificare che parte come un modello di regressione, ma successivamente viene appiattito con la log loss.

I parametri migliori indicati dal Grid Search sono:

```
{'C': 0.1, 'penalty': 'l2'}
```

Classification report senza ottimizzazione dei parametri:

```
Train score:  0.8410041841004184
Test score:  0.8333333333333334
Cross Validated Score:  0.840530303030303
Standard Deviation:  0.09942658716398774
Variance:  0.00988564623507805
0-1 Loss:  0.16666666666666663
Accuracy: 0.889602053915276
              precision    recall  f1-score   support

           0       0.82      0.98      0.89        41
           1       0.91      0.53      0.67        19

    accuracy                           0.83        60
   macro avg       0.86      0.75      0.78        60
weighted avg       0.85      0.83      0.82        60
```

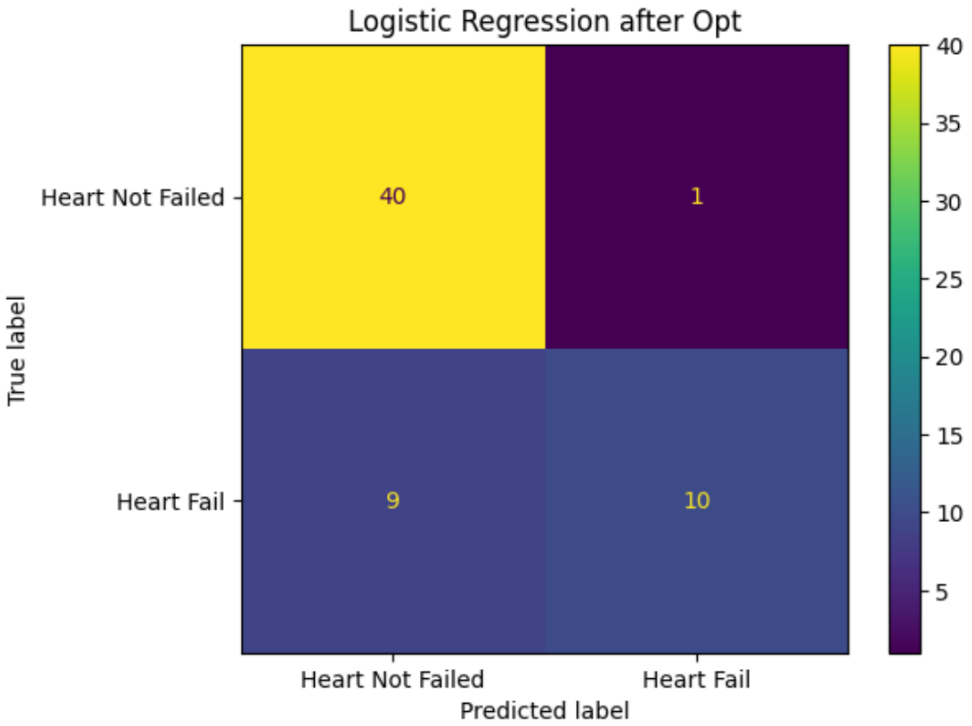
Classification report con ottimizzazione dei parametri:

```
Train score:  0.8368200836820083
Test score:  0.8333333333333334
Cross Validated Score:  0.8196969696969697
Standard Deviation:  0.10049235999266926
Variance:  0.010098714416896231
0-1 Loss:  0.16666666666666663
Accuracy: 0.8921694480102695
              precision    recall  f1-score   support

           0       0.82      0.98      0.89        41
           1       0.91      0.53      0.67        19

    accuracy                           0.83        60
   macro avg       0.86      0.75      0.78        60
weighted avg       0.85      0.83      0.82        60
```

Per avere una rassicurazione circa il corretto apprendimento del classificatore, è stata creata una matrice di confusione, la quale ci ha fornito una rappresentazione grafica dell'accuratezza del classificatore:



Multi Layer Perceptron

Una Rete Neurale è un modello di apprendimento che si basa sul funzionamento dei neuroni cerebrali biologici, per il nostro problema abbiamo utilizzato una Rete Neurale di tipo feed-forward che si basa su una gerarchia di funzioni lineari intervallate da funzioni di attivazione. In genere prende in input una serie di feature e le sottopone agli strati nascosti (detti anche feature non osservate) che le mappano secondo una funzione di attivazione e restituiscono in output uno o più feature obiettivo. Formalmente è composta da tre strati o layer: Un layer di input (nel nostro caso tutte le feature che rappresentano i sintomi). Un layer lineare completo, il cui modello matematico è:

$$y = \sum_{i} w_i * x_i$$

Dove y è la classe da predire, x è il numero di feature di input e w sono i pesi da assegnare ad ogni feature di input. Una funzione di attivazione f.

Per ottenere una classificazione accurata si fa uso della back-propagation che ricalcola ed aggiorna i pesi w. Ho optato per l'uso di una Rete Neurale poiché questa tipologia di modello è flessibile e si comporta molto bene nel caso si disponga di molti dati. Dato l'elevato costo computazionale che una rete neurale comporta, mi sono limitato ad effettuare dei test per la ricerca della migliore funzione di attivazione per la nostra rete.

I parametri migliori indicati dal Grid Search sono:

```
{'activation': 'relu'}
```

Classification report senza ottimizzazione dei parametri:

```
Train score:  0.8368200836820083
Test score:  0.85
Cross Validated Score:  0.8155303030303029
Standard Deviation:  0.10156248896305653
Variance:  0.01031493916437098
0-1 Loss:  0.15000000000000002
Accuracy: 0.852374839537869
              precision    recall  f1-score   support

           0       0.88      0.90      0.89        41
           1       0.78      0.74      0.76        19

    accuracy                           0.85        60
   macro avg       0.83      0.82      0.82        60
weighted avg       0.85      0.85      0.85        60
```

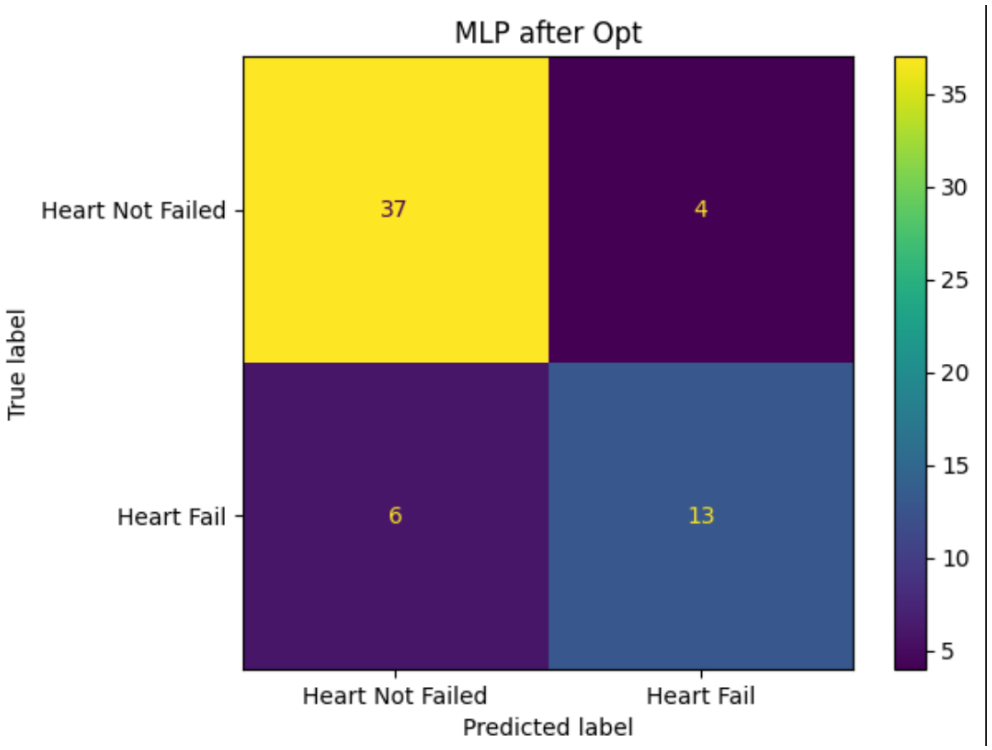
Classification report con ottimizzazione dei parametri:

```
Train score:  0.8702928870292888
Test score:  0.8333333333333334
Cross Validated Score:  0.8155303030303032
Standard Deviation:  0.11741874376177426
Variance:  0.013787161386593203
0-1 Loss:  0.16666666666666663
Accuracy: 0.8575096277278561
              precision    recall  f1-score   support

           0       0.86      0.90      0.88        41
           1       0.76      0.68      0.72        19

    accuracy                           0.83        60
   macro avg       0.81      0.79      0.80        60
weighted avg       0.83      0.83      0.83        60
```

Per avere una rassicurazione circa il corretto apprendimento del classificatore, è stata creata una matrice di confusione, la quale ci ha fornito una rappresentazione grafica dell'accuratezza del classificatore:



Apprendimento non supervisionato

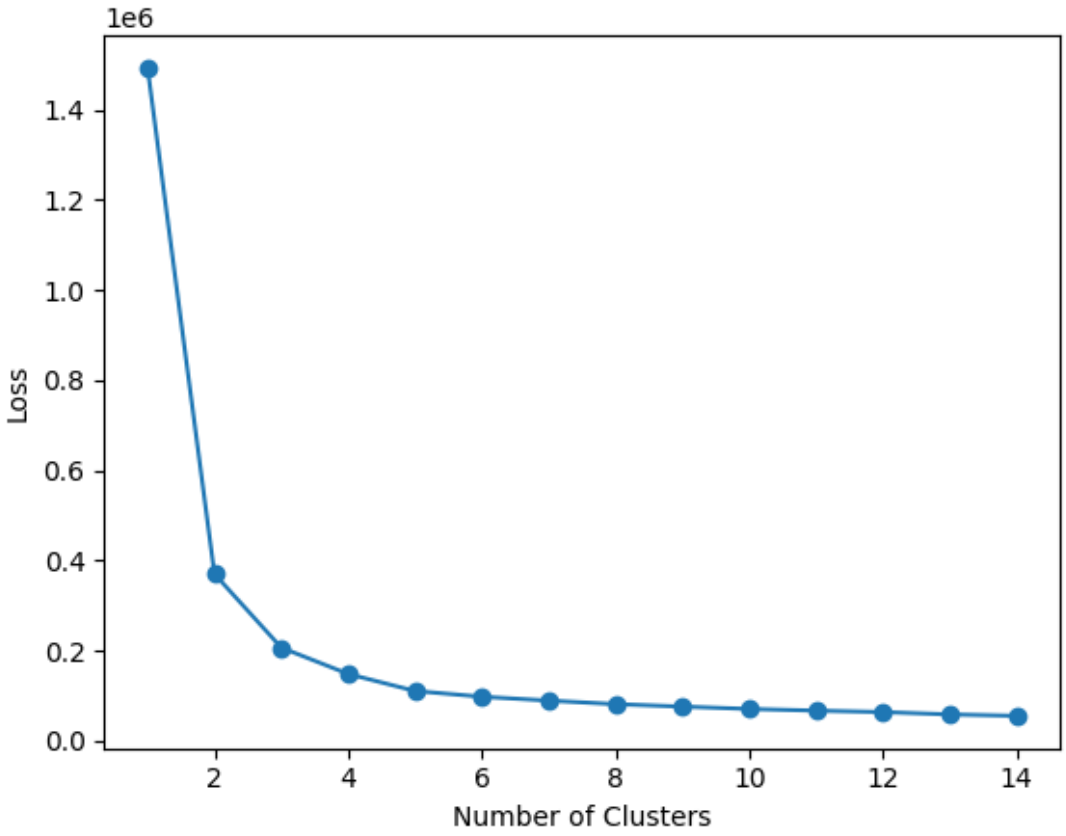
Per apprendimento non supervisionato si intende un apprendimento in cui non ci sono feature-obiettivo negli esempi di training.

Clustering K-Means

K-Means è un algoritmo per raggruppare i dati senza etichetta in k gruppi e ogni datapoint è assegnato al centroide più vicino. In questa sezione, assumiamo che il nostro set di dati non abbia un'etichetta, quindi costruiamo l'algoritmo con solo 12 variabili indipendenti, nessuna DEATH_EVENT include. La performance del modello dipende dal valore di k. Per trovare il valore ottimale di k, viene eseguito il **"metodo del gomito"**.

Il metodo del gomito traccia il valore dell'inerzia, o la funzione di perdita, e il numero di cluster (k). l'inerzia indica quanto sono lontani i punti all'interno di un cluster. Il numero di cluster aumenta mentre la perdita diminuisce. Ciò significa che ogni cluster ha meno punti dati e i punti dati sono più vicini ai loro centriodi. Il valore di k in cui la perdita diminuisce di più è chiamato gomito, ed è dove dividiamo

più cluster. In questo esperimento, è molto difficile identificare il punto di rottura del gomito poiché il grafico ha un andamento decrescente regolare.

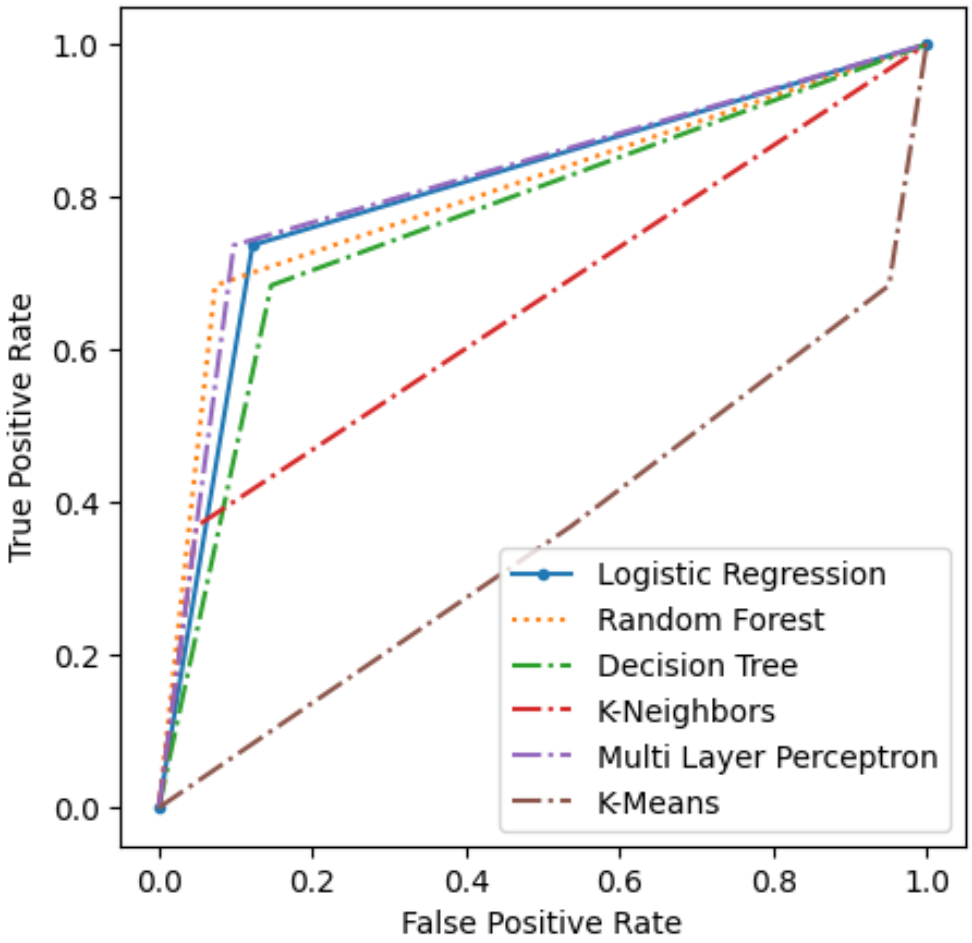


Confrontiamo l'output del clustering "K-means" e l'output originale. Dato che vorremmo raggruppare i nostri pazienti in due gruppi: pazienti morti e sopravvissuti. Quindi usiamo k=2 per il modello. La tabella seguente mostra l'output di K-means e il set di dati originale.

	K-Means label	DEATH_EVENT
0	1	1
1	1	1
2	1	1
3	1	1
4	0	1

L'accuratezza del modello K-means con due cluster è 0,4448.

FINE



Alla luce di quanto emerge dallo studio di questo grafico, possiamo notare come il Random Forest ha un apprendimento rapido in una fase iniziale ed una crescita lineare dopo alcuni test, restituendo fin da subito un minore numero di falsi positivi. La stessa curva di apprendimento è seguita anche dalla rete neurale e dall'algoritmo Logistic Regression, anche se questi ultimi tendono a sbagliare in una fase iniziale, e dopo poco apprendono correttamente restituendo sempre meno casi di falsi positivi. L'algoritmo Decision Tree ha un apprendimento più lento rispetto agli altri citati finora. Invece, l'algoritmo di apprendimento non supervisionato K-Means ha un apprendimento molte lento, e non molto preciso, dato che si inizia ad affinare alla fine dei nostri test.

Questa figura rappresenta l’accuratezza dei vari modelli:

	Model	Accuracy Score
0	RFC	0.915918
1	KNN	0.821566
2	DT	0.768935
3	LR	0.854942
4	MLP	0.857510