

UNIVERSITÀ DI PISA

Corso di Laurea in Ingegneria Informatica

Progetto di Laboratorio di Informatica Applicata

Medical Diagnosis Aid with Data Science and Machine Learning

Professore:

Tiberio Uricchio

Studenti:

Luca Granucci

Paolo Walsh

Indice

1 Introduzione	3
1.1 Descrizione del problema	3
1.2 Obiettivi del Progetto	3
2 Analisi Esplorativa dei Dati	4
2.1 Dataset utilizzato	4
2.2 Pulizia e preparazione dei dati	5
2.3 Analisi statistica e visualizzazione	5
2.3.1 Visualizzazione dei dati	5
2.3.2 Osservazioni principali	7
2.3.3 Bias del dataset	11
3 Machine Learning	12
3.1 Preprocessing dei dati	12
3.2 Model Selection	13
3.3 Fine-tuning ed evaluation dei modelli	13
3.3.1 Fine-tuning del Classificatore K-Nearest Neighbors	13
3.3.2 Fine-tuning del Classificatore di Regressione Logistica	14
3.3.3 Valutazione modelli	15
3.4 Interpretazione del Modello con SHAP	16
3.4.1 Interpretazione dello <code>summary_plot</code> SHAP	16
3.5 Large Language Model per le diagnosi	18
3.6 Valutazione della Performance Diagnostica Umana	20
3.7 Considerazione Finali	21
4 Integrazione e sviluppo dell'applicazione web	22
4.1 Flask API	22
4.2 Interfaccia Streamlit	22
4.3 Controllo di versione con Git	23
4.4 Containerizzazione con Docker	23
4.5 Istruzioni per l'uso dell'applicazione web	23

Capitolo 1

Introduzione

Abbiamo scelto di sviluppare e approfondire il progetto "**Medical Diagnosis Aid with Data Science and Machine Learning**" poiché ci affascinava l'applicazione della data science e del machine learning a un ambito tanto pratico quanto rilevante come la medicina. La possibilità di utilizzare modelli predittivi per supportare diagnosi mediche rappresenta non solo una sfida tecnica interessante, ma anche un'opportunità concreta per contribuire, seppur in modo preliminare, a un settore che ha un impatto diretto sulla vita delle persone.

1.1 Descrizione del problema

Una diagnosi precoce e accurata di malattie cardiache è fondamentale per migliorare la qualità della vita dei pazienti e ridurre i costi sanitari. Tuttavia, l'analisi manuale dei dati clinici può essere soggettiva, dispendiosa in termini di tempo e suscettibile a errori.

1.2 Obiettivi del Progetto

- Sviluppare modelli predittivi in grado di stimare la probabilità di una malattia specifica (diabete o malattia cardiaca) a partire da dati clinici.
- Confrontare i risultati dei modelli di machine learning con l'approccio di un LLM (Large Language Model) e di uno studente di medicina del 5° anno in fase diagnostica.
- Analizzare la interpretabilità dei modelli per comprendere le variabili più influenti.

Capitolo 2

Analisi Esplorativa dei Dati

L’analisi esplorativa dei dati ha costituito la prima fase operativa del progetto ed è stata fondamentale per comprendere la struttura dei dataset, individuare eventuali problemi nei dati e iniziare a identificare relazioni utili per la fase di modellazione.

2.1 Dataset utilizzato

Abbiamo lavorato su un dataset ampiamente utilizzato nella letteratura scientifica:

- **Heart Disease UCI Dataset:** raccoglie informazioni relative a pazienti sottoposti a esami cardiaci (es. colesterolo, elettrocardiogramma, frequenza cardiaca massima), con lo scopo di prevedere la presenza di una malattia cardiaca.

Feature	Descrizione
<code>id</code>	Identificatore univoco del paziente nel dataset.
<code>age</code>	Età del paziente (in anni).
<code>sex</code>	Sesso del paziente (0 = femmina, 1 = maschio).
<code>dataset</code>	Origine del campione all’interno dei diversi sottodataset UCI.
<code>chest_pain_type</code>	Tipo di dolore toracico (es. tipico anginoso, atipico, non anginoso, asintomatico).
<code>blood_pressure_resting</code>	Pressione arteriosa a riposo (mm Hg).
<code>cholesterol</code>	Colesterolo sierico (mg/dl).
<code>fasting_blood_sugar</code>	Glicemia a digiuno > 120 mg/dl (1 = vero, 0 = falso).
<code>ecg_resting</code>	Risultati dell’elettrocardiogramma a riposo (0, 1 o 2, che indicano diverse anomalie).
<code>max_heart_rate</code>	Frequenza cardiaca massima raggiunta durante il test da sforzo.
<code>exercise_induced_angina</code>	Angina indotta da esercizio.
<code>st_depression_exercise</code>	Depressione del tratto ST causata dall’esercizio rispetto al riposo.
<code>st_slope_type</code>	Inclinazione del segmento ST durante l’esercizio (es. ascendente, piatto, discendente).
<code>major_vessels_colored</code>	Numero di vasi principali visualizzati con fluoroscopia colorata (da 0 a 3).
<code>thal_defect_type</code>	Tipo di difetto nel test del talio (normale, fisso, reversibile).
<code>heart_disease_gravity</code>	Grado di gravità della malattia cardiaca (0 = nessuna, fino a 4 = massima).
<code>sick</code>	Etichetta binaria: 1 = presenza di malattia cardiaca, 0 = assenza.

Tabella 2.1: Descrizione delle feature presenti nel dataset Heart Disease UCI.

id	age	sex	dataset	chest pain	BP	chol.	fbs	ECG	HR max	angina	ST dep.	ST slope	vessels	thal	pred.	sick
1	63	Male	Cleveland	typical angina	145.0	233.0	True	lv hypertrophy	150.0	False	2.3	downsloping	0.0	fixed defect	0	0
2	67	Male	Cleveland	asymptomatic	160.0	286.0	False	lv hypertrophy	108.0	True	1.5	flat	3.0	normal	2	1
3	67	Male	Cleveland	asymptomatic	120.0	229.0	False	lv hypertrophy	129.0	True	2.6	flat	2.0	reversible defect	1	1
4	37	Male	Cleveland	non-anginal	130.0	250.0	False	normal	187.0	False	3.5	downsloping	0.0	normal	0	0
5	41	Female	Cleveland	atypical angina	130.0	204.0	False	lv hypertrophy	172.0	False	1.4	upsloping	0.0	normal	0	0

Tabella 2.2: Esempio di campione dati dal dataset Heart Disease UCI.

2.2 Pulizia e preparazione dei dati

Nel dataset sono state individuate alcune anomalie, come valori pari a zero in variabili che non possono assumere tale valore (ad esempio, pressione diastolica). Questi casi sono stati trattati come valori mancanti e gestiti tramite rimozione per l’addestramento dei modelli di machine learning. Il dataset contiene **910** campioni di pazienti. Tuttavia, escludendo i valori nulli, si ottiene un dataset composto da **299** campioni. Questa riduzione della dimensione potrebbe portare i modelli a individuare pattern non rappresentativi della popolazione globale.

2.3 Analisi statistica e visualizzazione

Abbiamo effettuato un’analisi statistica descrittiva per tutte le variabili, calcolando la media, deviazione standard, valori minimi e massimi.

Variabile	count	mean	std	min	25%	50%	75%	max
age	920	53.51	9.42	28.0	47.0	54.0	60.0	77.0
blood_pressure_resting	861	132.13	19.07	0.0	120.0	130.0	140.0	200.0
cholesterol	890	199.13	110.78	0.0	175.0	223.0	268.0	603.0
max_heart_rate	865	137.55	25.93	60.0	120.0	140.0	157.0	202.0
st_depression_exercise	858	0.88	1.09	-2.6	0.0	0.5	1.5	6.2
major_vessels_colored	309	0.68	0.94	0.0	0.0	0.0	1.0	3.0
heart_disease_gravity	920	1.00	1.14	0.0	0.0	1.0	2.0	4.0

Tabella 2.3: Statistiche descrittive per le variabili principali del dataset Heart Disease UCI.

2.3.1 Visualizzazione dei dati

Per esplorare il dataset, abbiamo iniziato visualizzando un pairplot Figura 2.1 per esaminare visivamente le relazioni tra le variabili, distinguendo i dati in base alla presenza o meno della malattia.

Successivamente, abbiamo calcolato le correlazioni di Pearson tra le feature, dopo aver eseguito un **One Hot Encoding**, utile per individuare rapidamente le relazioni lineari più forti.

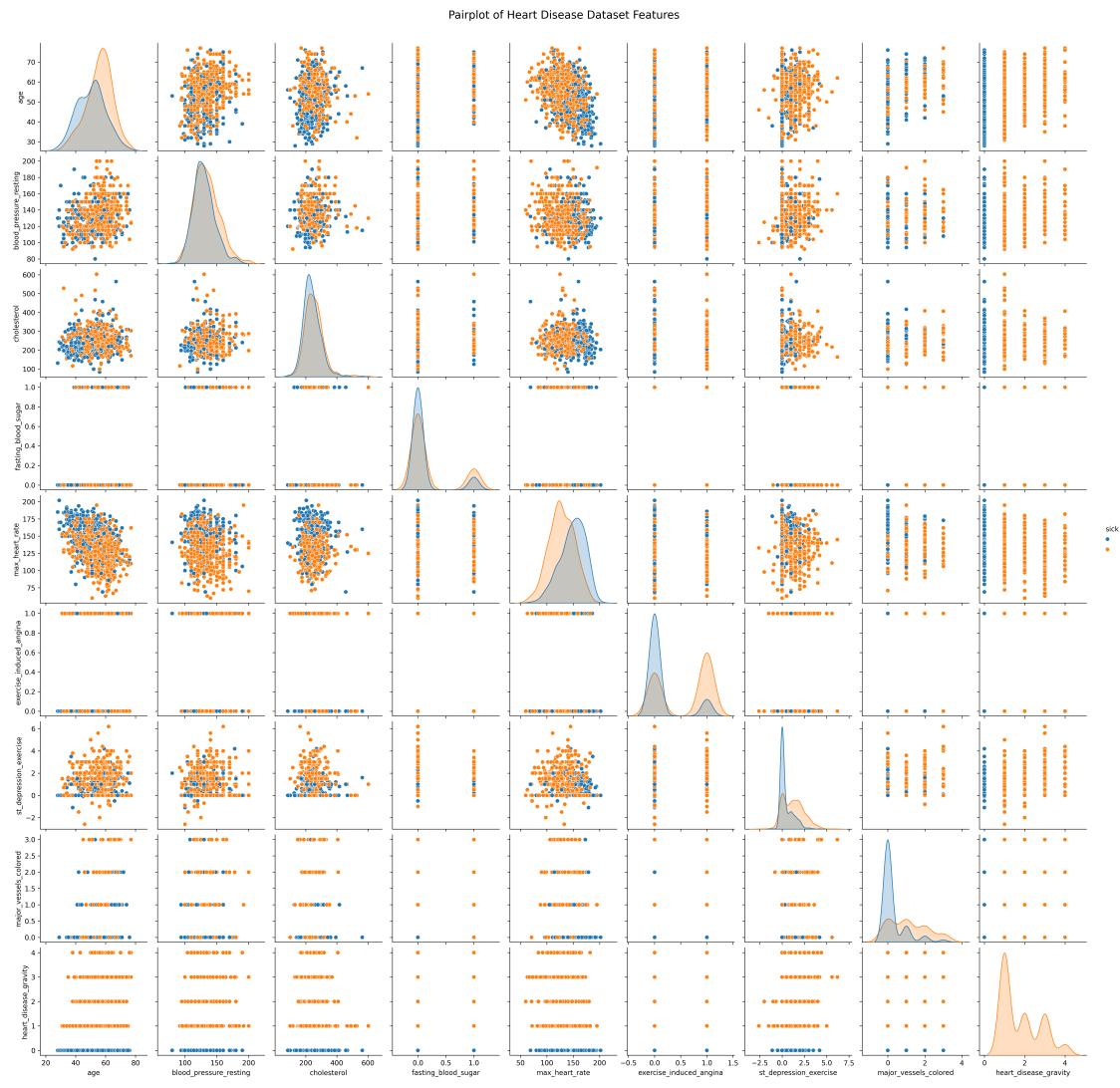


Figura 2.1: Pairplot delle feature principali, colorato in base allo stato di malattia.

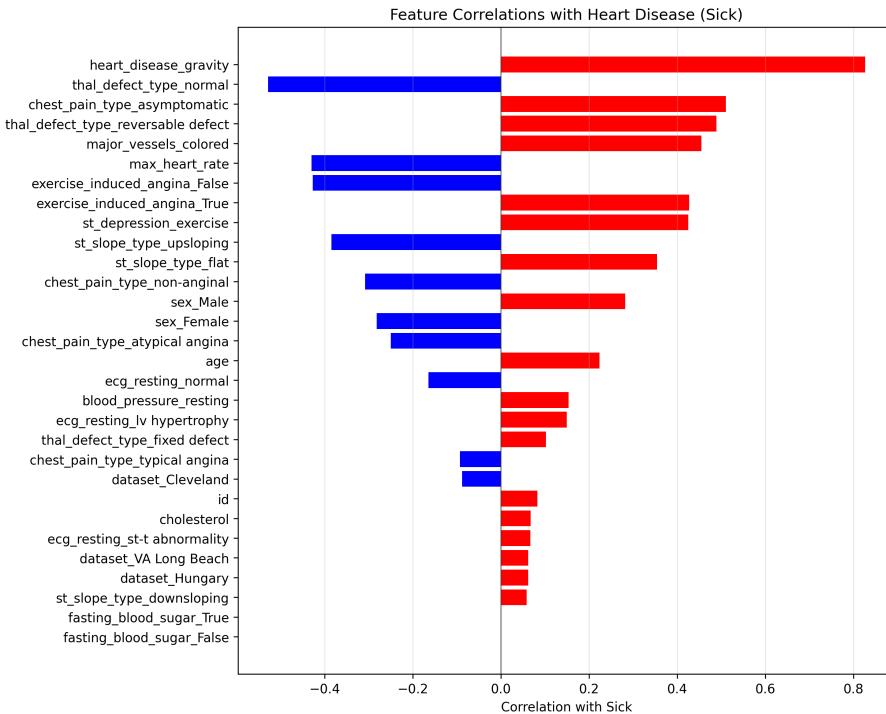


Figura 2.2: Correlazioni delle feature con la feature *sick*.

2.3.2 Osservazioni principali

Dall’analisi di Figura 2.2, emergono alcune feature con una correlazione significativa rispetto alla variabile *sick*, che indica la presenza di malattia.

La Tabella 2.4 riassume le variabili con coefficiente di correlazione (in valore assoluto) maggiore o uguale a 0.3.

I risultati indicano che alcune feature che nell’immaginario comune sono ritenute molto importanti come il *colesterolo*, sono nel nostro studio, in realtà meno correlate alla malattia, di altre come il *sesso*. Questo fenomeno è molto probabilmente dovuto alla ridotta dimensione del dataset e a diversi bias che andremo a discutere in maniera più esaustiva nella prossima sezione.

Infine per validare visivamente le correlazioni trovate, abbiamo generato alcuni grafici, Figura 2.3, Figura 2.4, Figura 2.5, Figura 2.6, Figura 2.7, che confrontano la distribuzione o frequenza di tali feature in base allo stato di malattia (*sick*).

In Figura 2.4 notiamo come una porzione significativa di pazienti con diagnosi di malattia cardiaca non manifesti il tipico dolore toracico anginoso. Sebbene questo possa apparire controtintuitivo, il risultato è clinicamente plausibile e riflette la nota eterogeneità dei sintomi della malattia cardiaca, che può presentarsi anche in forme asintomatiche o con dolore atipico.

Feature	Correlazione con sick
heart_disease_gravity ^a	0.78
chest_pain_type	-0.46
thal_defect_type	0.46
exercise_induced_angina	0.46
major_vessels_colored	0.46
st_slope_type	0.44
max_heart_rate	-0.39

^a Questa colonna rappresenta direttamente il grado di malattia diagnosticato, e quindi è naturalmente molto correlata con la variabile **sick**. Per questo motivo, non viene utilizzata come feature nei modelli predittivi.

Tabella 2.4: Feature con maggiore correlazione (in valore assoluto) con la variabile **sick** ($|\rho| \geq 0.3$).

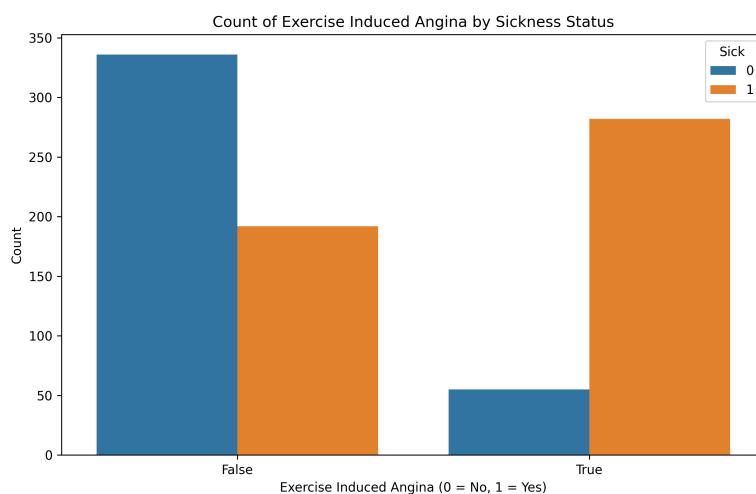


Figura 2.3: Presenza di angina da sforzo in funzione dello stato di malattia.

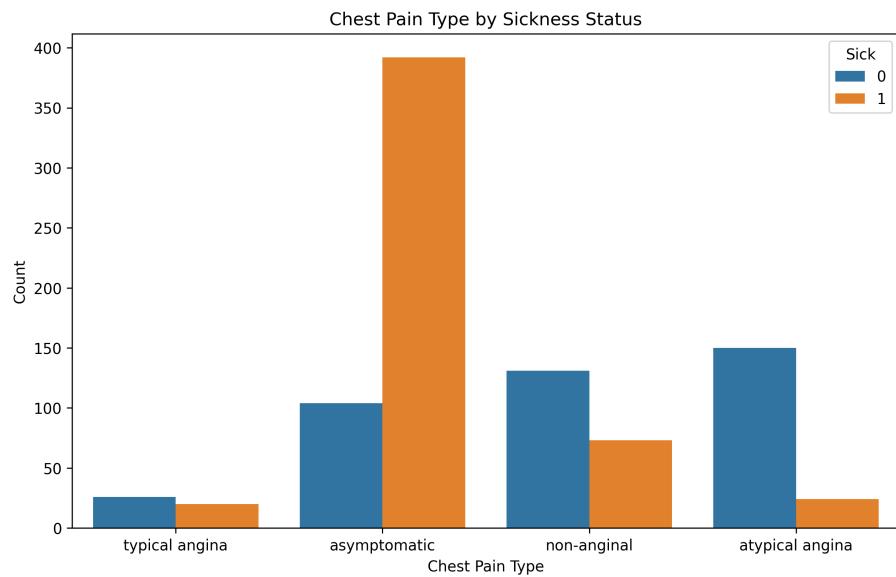


Figura 2.4: Tipologia di dolore toracico in funzione dello stato di malattia.

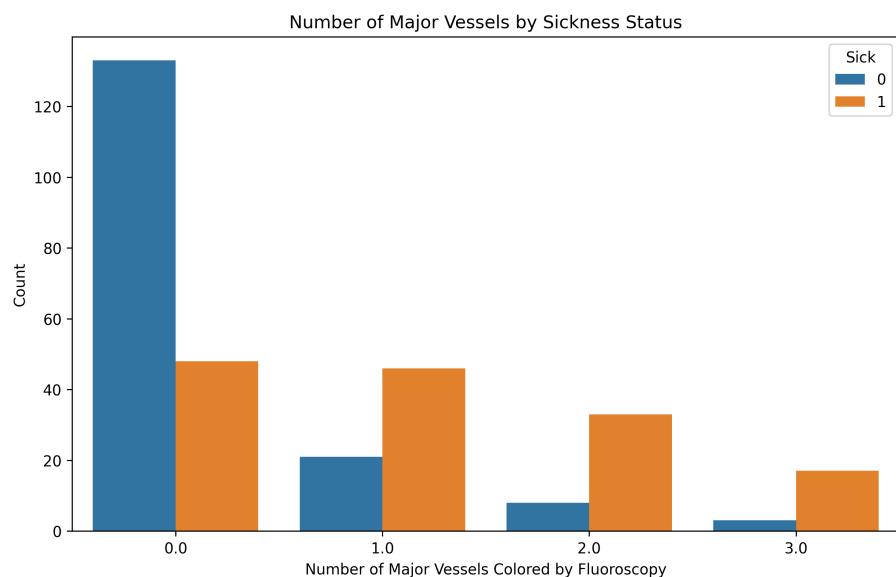


Figura 2.5: Numero di vasi principali colorati rispetto allo stato di malattia.

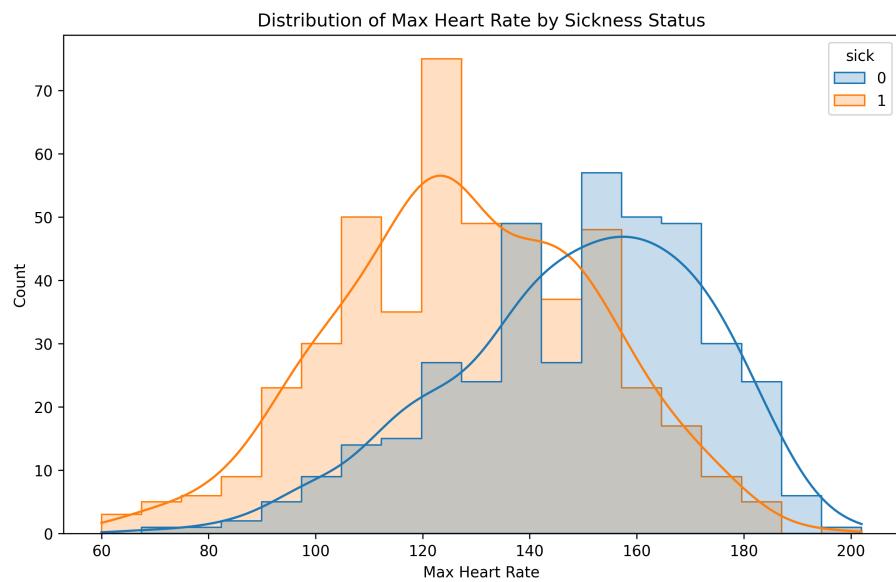


Figura 2.6: Distribuzione della frequenza cardiaca massima rispetto allo stato di malattia.

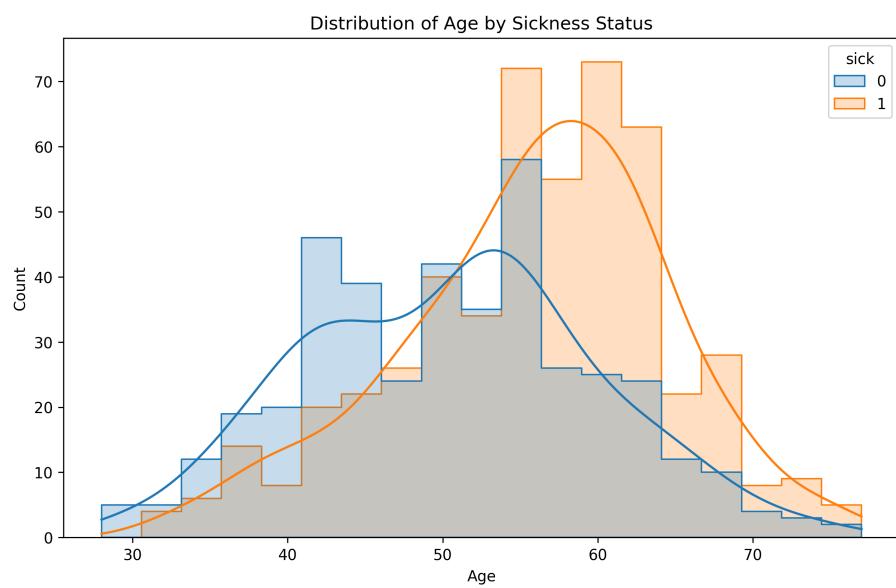


Figura 2.7: Distribuzione dell'età in funzione dello stato di malattia.

2.3.3 Bias del dataset

Il dataset impiegato in questo studio è composto da un totale di 910 campioni, di cui 726 relativi a soggetti di sesso maschile e 184 a soggetti di sesso femminile. Questa marcata disparità di genere ($\approx 80\%$ maschi) rappresenta un evidente squilibrio nella distribuzione dei dati, che può introdurre bias nel processo di apprendimento del modello e compromettere la sua capacità di generalizzare equamente tra i due gruppi. Inoltre possiamo osservare in Figura 2.8 che anche normalizzando in base al numero di soggetti maschili e femminili c'è un importante squilibrio nella presenza o meno di malattia. Un modello addestrato su dati fortemente sbilanciati tende infatti a sovra-adattarsi alla classe maggioritaria, riducendo le prestazioni predittive sui soggetti meno rappresentati, in questo caso le donne. Inoltre, i dati provengono esclusivamente da quattro centri clinici: Cleveland, Ungheria, Svizzera e VA Long Beach. Tale limitata diversità geografica restringe la capacità del modello di adattarsi a popolazioni provenienti da contesti socio-demografici differenti, visto che il tasso di malattie cardiovascolari varia in base all'etnia [1],

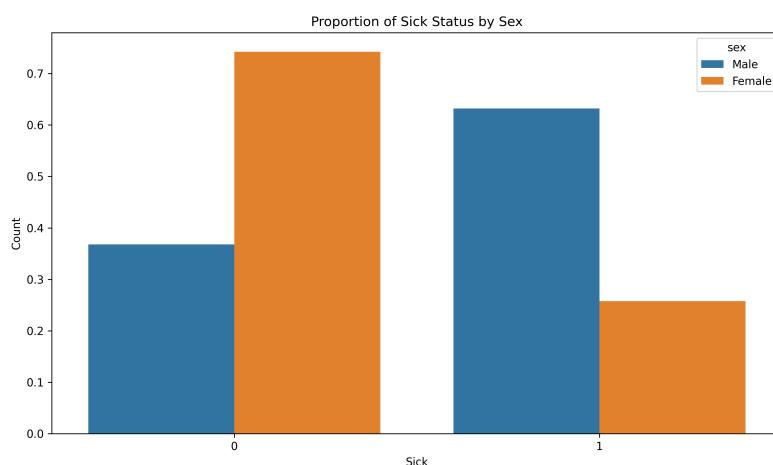


Figura 2.8: Numero di pazienti malati normalizzati per il sesso

Capitolo 3

Machine Learning

In questo capitolo andiamo a descrivere come sono stati scelti ed addestrati diversi modelli di machine learning per la predizione della malattia cardiaca, dati come input le feature del paziente.

3.1 Preprocessing dei dati

Prima di addestrare i modelli, abbiamo effettuato due operazioni fondamentali di preprocessing:

- **One Hot Encoding:** le feature categoriche o discrete (come `sex`, `chest_pain_type`, `thal_defect_type`, ecc.) sono state trasformate in un insieme di variabili binarie. Questa tecnica consente ai modelli di machine learning di trattare in modo corretto variabili non numeriche, evitando relazioni ordinarie spurie.

chest_pain_type	cp_typical	cp_atypical	cp_nonanginal	cp_asymptomatic
typical angina	1	0	0	0
asymptomatic	0	0	0	1
non-anginal pain	0	0	1	0
atypical angina	0	1	0	0

Tabella 3.1: Esempio di codifica One Hot per la feature `chest_pain_type`.

- **Standardizzazione:** tutte le feature numeriche sono state scalate utilizzando lo *Standard Scaler*, che trasforma i dati affinché abbiano media 0 e deviazione standard 1. Questo passaggio è cruciale per garantire che tutte le feature abbiano lo stesso peso durante l'addestramento, specialmente per modelli basati su distanze o ottimizzazione numerica.

La formula applicata per ogni valore x è la seguente:

$$x_{\text{scaled}} = \frac{x - \mu}{\sigma}$$

dove μ è la media della feature e σ la sua deviazione standard.

3.2 Model Selection

Abbiamo scelto diverse tipologie di modelli, ovvero:

- K-Nearest Neighbors Classifier
- Logistic Regression

Modello	Accuracy	Recall	Precision	F1-Score	ROC AUC
K-Nearest Neighbors	0.822	0.762	0.842	0.800	0.881
Logistic Regression	0.822	0.738	0.861	0.795	0.928

Tabella 3.2: Prestazioni iniziali dei modelli sul test set.

3.3 Fine-tuning ed evaluation dei modelli

Questa sezione descrive il processo di ottimizzazione degli iperparametri (fine-tuning) per i modelli K-Nearest Neighbors (KNN) e Regressione Logistica. L’obiettivo del fine-tuning è identificare la combinazione di iperparametri che massimizza la performance del modello su dati non visti, riducendo il rischio di overfitting o underfitting. Per entrambi i modelli, è stata utilizzata la tecnica della Grid Search con Cross-Validation, valutando le performance tramite la metrica *recall*.

Il *recall*, noto anche come Sensibilità, è una metrica di valutazione cruciale per i modelli di classificazione. Misura la proporzione di veri positivi che sono stati correttamente identificati dal modello rispetto a tutti i casi positivi reali ed è particolarmente rilevante in contesti dove è fondamentale non perdere nessun caso positivo, come nella diagnosi medica. La sua formula è la seguente:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

3.3.1 Fine-tuning del Classificatore K-Nearest Neighbors

Per il classificatore KNN, sono stati esplorati i seguenti iperparametri e i loro rispettivi valori:

- **n_neighbors**: il numero di vicini da considerare, con valori testati di [3, 5, 7, 9, 11].
- **weights**: la modalità di ponderazione dei vicini, con opzioni *uniform* (tutti i vicini hanno lo stesso peso) e *distance* (i vicini più vicini hanno un peso maggiore).
- **metric**: la metrica di distanza utilizzata per calcolare la prossimità tra i punti, con opzioni *euclidean* (distanza euclidea) e *manhattan* (distanza di Manhattan).

La ricerca è stata condotta tramite `GridSearchCV` con 5 fold di cross-validation (`cv=5`) e ottimizzazione della metrica *recall*.

Il processo di tuning ha identificato i seguenti migliori iperparametri:

Parametro	Valore
Distanza (<code>metric</code>)	manhattan
Numero di vicini (<code>n_neighbors</code>)	11
Pesi (<code>weights</code>)	distance
Recall in cross-validation	0.796

Tabella 3.3: Parametri ottimali trovati per K-Nearest Neighbors e relativo punteggio di *recall* in cross-validation.

3.3.2 Fine-tuning del Classificatore di Regressione Logistica

Per il classificatore di Regressione Logistica, la ricerca degli iperparametri è stata definita considerando le specificità dei diversi solutori:

- Per il solutore `liblinear`:
 - `penalty`: il tipo di regolarizzazione, con opzioni *l1* (Lasso) e *l2* (Ridge).
 - `C`: l'inverso della forza di regolarizzazione, con valori esplorati di [0.001, 0.01, 0.1, 1, 10, 100]. Valori più piccoli di `C` implicano una regolarizzazione più forte.
- Per il solutore `lbfgs`:
 - `penalty`: solo l'opzione *l2* è stata considerata, in quanto `lbfgs` non supporta la regolarizzazione *l1*.
 - `C`: l'inverso della forza di regolarizzazione, con gli stessi valori di [0.001, 0.01, 0.1, 1, 10, 100].

È stato inoltre specificato `random_state=42` per riproducibilità e `max_iter=1000` per garantire la convergenza del solutore. Anche in questo caso, la ricerca è stata effettuata con `GridSearchCV` su 5 fold di cross-validation, ottimizzando per la metrica *recall*.

Il fine-tuning ha determinato i seguenti migliori iperparametri per la Regressione Logistica:

Parametro	Valore
Regolarizzazione (<code>C</code>)	0.01
Penalità (<code>penalty</code>)	<code>l2</code>
Solver (<code>solver</code>)	<code>liblinear</code>
Recall in cross-validation	0.826

Tabella 3.4: Parametri ottimali per il modello di Logistic Regression e relativo punteggio di *recall* in cross-validation.

3.3.3 Valutazione modelli

Possiamo osservare, confrontando Tabella 3.5 e Tabella 3.2, come il processo di fine-tuning abbia portato ad un miglioramento del *recall* nel modello di **Règression Logistica** ma non in quello **KNN**. Il fine-tuning ha anche portato ad un miglioramento nell'*accuracy* e *F1-Score* di entrambi i modelli.

Modello	Accuracy	Recall	Precision	F1-Score	ROC AUC
K-Nearest Neighbors	0.833	0.762	0.865	0.810	0.881
Logistic Regression	0.833	0.786	0.846	0.815	0.928

Tabella 3.5: Prestazioni dei modelli sul test set post fine-tuning.

Mostriamo anche le **confusion matrix** per entrambi i modelli per mostrare la tipologia di errori commessi nella predizione.

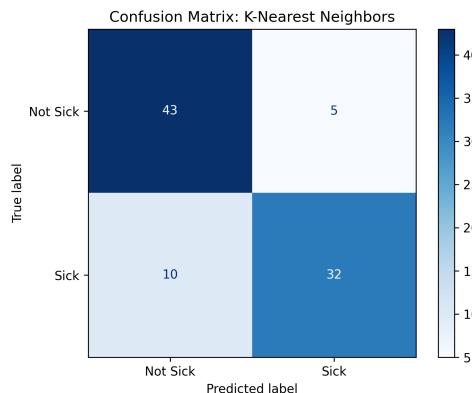


Figura 3.1: Confusion Matrix K-Nearest Neighbors

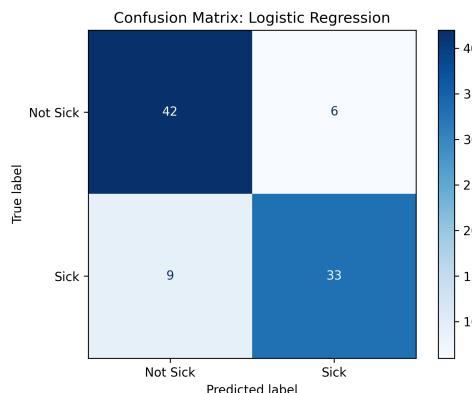


Figura 3.2: Confusion Matrix Logistic Regression

3.4 Interpretazione del Modello con SHAP

In questa sezione, abbiamo utilizzato le **SHapley Additive exPlanations (SHAP)** per interpretare il modello di Regressione Logistica ottimizzato. SHAP è un framework basato sulla teoria dei giochi che assegna a ciascuna feature un valore di importanza per una previsione specifica, spiegando come ogni feature contribuisce a spostare la previsione dal valore base (media delle previsioni).

3.4.1 Interpretazione dello summary_plot SHAP

Il comando `shap.summary_plot` genera un grafico riassuntivo che fornisce una panoramica dell'importanza di ogni feature e della loro distribuzione di impatto sulle previsioni del modello. Questo grafico è cruciale per la comprensione globale del modello.

Un tipico `summary_plot` (come quello in Figura 3.3) presenta:

- **Asse Y:** Elenca le feature in ordine di importanza decrescente. Le feature più in alto sono quelle che hanno il maggiore impatto medio sull'output del modello.
- **Asse X:** Indica il valore SHAP, che rappresenta l'impatto di quella feature sulla previsione del modello. Un valore SHAP positivo indica che la feature tende ad aumentare la previsione (probabilità di malattia), mentre un valore negativo tende a diminuirla.
- **Colore dei punti:** I punti sono colorati in base al valore della feature stessa (rosso per valori alti, blu per valori bassi). Questo permette di visualizzare se un valore alto o basso di una feature ha un impatto positivo o negativo sulla previsione.

Analizzando il `summary_plot`, è possibile identificare le feature più influenti e comprendere la direzione del loro impatto. Ad esempio, se i punti rossi (valori alti della feature) si trovano sul lato positivo dell'asse X, significa che valori elevati di quella feature tendono ad aumentare la probabilità della classe positiva. Viceversa per i punti blu. Questa visualizzazione fornisce insight potenti sulla relazione tra le feature e l'output del modello.

Feature	SHAP Importance
thal_defect_type_normal	0.078
thal_defect_type_reversible_defect	0.072
chest_pain_type_asymptomatic	0.057
major_vessels_colored	0.045
max_heart_rate	0.032
chest_pain_type_non-anginal	0.031

Tabella 3.6: Feature più importanti secondo i valori SHAP (SHapley Additive ex-Planations).

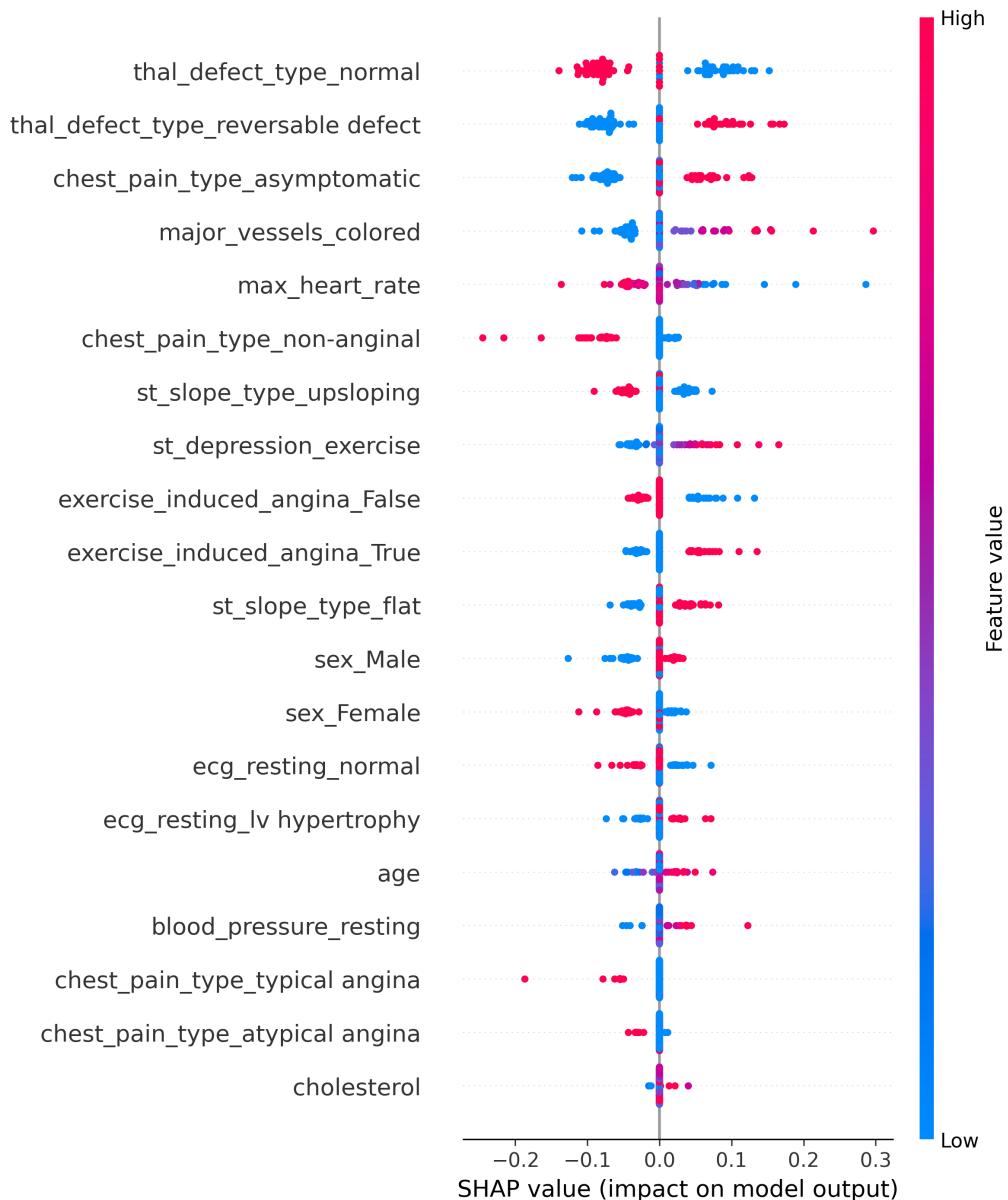


Figura 3.3: SHAP Summary Plot del modello di Regressione Logistica.

Si notano importanti differenze con la Tabella 2.4 che contiene le feature più correlate con la colonna *sick*. Ad esempio, la feature *exercise_induced_angina* nonostante fosse molto correlata con *sick* $\rho = 0.46$, non si trova tra le feature più rilevanti secondo i valori SHAP.

È inoltre interessante notare che il livello di colesterolo sia una feature considerata molto poco importante dei valori SHAP. In effetti i livelli di colesterolo di tutti i pazienti del dataset sono piuttosto uniformi, e non sembra essere molto correlato a *sick*, avendo anche un coefficiente di correlazione di Pearson di solo 0.12. Figura 2.2

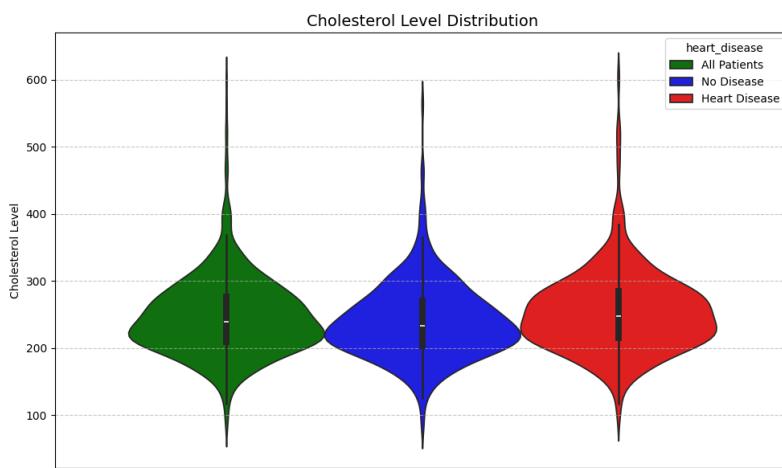


Figura 3.4: Distribuzione dei livelli di colesterolo nel dataset

3.5 Large Language Model per le diagnosi

Abbiamo anche condotto un'analisi esplorativa sulle capacità diagnostiche di un Large Language Model, nello specifico GPT-4 tramite l'API di OpenAI, per valutarne la performance nella predizione di patologie cardiache. L'obiettivo era verificare se il modello, basandosi unicamente sulla sua conoscenza interna e senza accesso a risorse esterne o ricerche su internet, potesse formulare una diagnosi binaria (malato/non malato).

A tal fine, è stato sviluppato uno script Python per interrogare GPT-4 con un prompt strutturato che includeva una serie dettagliata di parametri clinici del paziente. Di seguito è riportato un esempio del prompt utilizzato:

```
Without searching on the internet answer with a yes or no (y/n),
does this patient have heart disease?
```

```
age: 63
sex: Male
chest_pain_type: typical angina
```

```

blood_pressure_resting: 145.0
cholesterol: 233.0
fasting_blood_sugar: True
ecg_resting: lv hypertrophy
max_heart_rate: 150.0
exercise_induced_angina: False
st_depression_exercise: 2.3
st_slope_type: downsloping
major_vessels_colored: 0.0
thal_defect_type: fixed defect
heart_disease_gravity: 0

```

Le predizioni di GPT-4 sono state salvate e successivamente analizzate per calcolare le metriche di performance. I risultati ottenuti sono riassunti nella Tabella 3.7:

Modello	Accuracy	Recall	Precision	F1-Score
GPT-4	0.645	0.281	0.867	0.424

Tabella 3.7: Performance di Reg Log, GPT-4 e studente di medicina nella diagnosi di malattie cardiache.

Come evidenziato da Tabella 3.7, la performance di GPT-4, con un'accuracy del **64.5%** e un recall del **28.1%**, risulta nettamente inferiore rispetto ai modelli di machine learning tradizionali da noi addestrati per questo compito. Questo suggerisce che, sebbene gli LLM possiedano una vasta conoscenza, la loro capacità di ragionamento diagnostico isolato, senza feedback contestuale o accesso a strumenti di ricerca, presenta ancora significative limitazioni.

Infine, per una comprensione più approfondita degli errori di classificazione, è stata generata anche una **confusion matrix** per le predizioni di GPT-4.

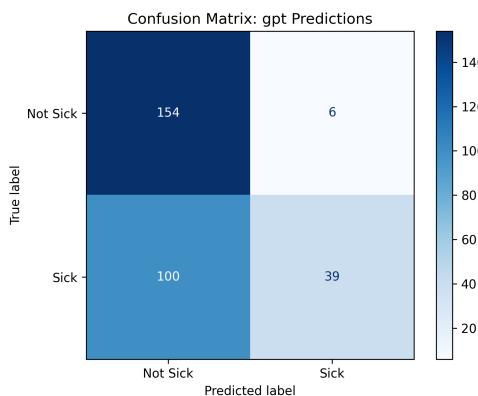


Figura 3.5: Confusion Matrix GPT-4

3.6 Valutazione della Performance Diagnostica Umana

Per stabilire un punto di riferimento umano e confrontare l'efficacia dei nostri modelli con la pratica clinica, abbiamo coinvolto una studentessa di medicina del quinto anno. Il suo compito consisteva nel fornire una diagnosi binaria (malato/non malato) per ciascun paziente, utilizzando lo stesso set di dati impiegato per i modelli di machine learning.

Abbiamo sviluppato uno script Python per la raccolta e l'organizzazione di queste diagnosi in un dataframe. Successivamente, abbiamo calcolato le metriche di accuracy e recall per quantificare le sue prestazioni diagnostiche.

Modello	Accuracy	Recall	Precision	F1-Score
Studente Medicina	0.699	0.604	0.651	0.795

Tabella 3.8: Performance di uno studente di medicina nella diagnosi di malattie cardiache.

Infine, per un'analisi dettagliata degli errori di classificazione abbiamo realizzato la confusion matrix delle sue predizioni.

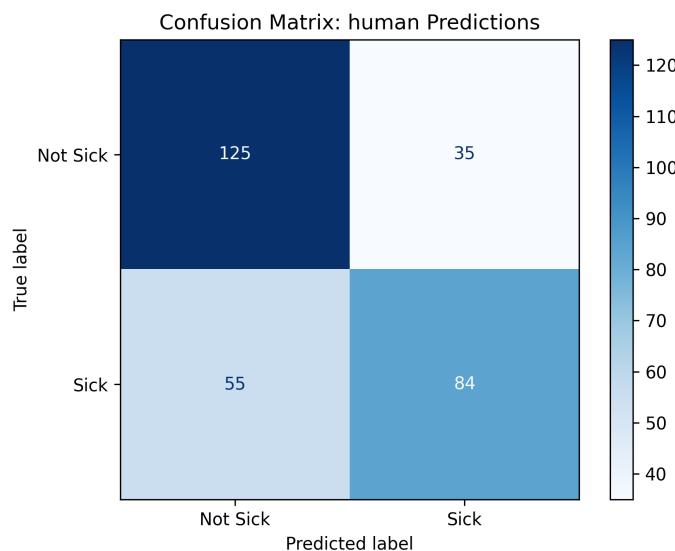


Figura 3.6: Human Prediction Confusion Matrix

Considerazioni sulla diagnosi umana

La studentessa di medicina ha affermato che il dataset non è completamente esplorativo. Alcune colonne non sono precise e alcune definizioni lasciano libera interpretazione.

Inoltre non esiste un'unica definizione di malattia cardiaca ma ne esistono molteplici. La studentessa ha dunque riscontrato difficoltà nel fare una diagnosi, data la poca documentazione che aveva il dataset.

3.7 Considerazione Finali

Osserviamo in Tabella 3.9, che riassume la performance di tutti i modelli utilizzati in questo studio, come i modelli classici di machine learning da noi addestrati siano più performanti per *accuracy*, *recall* ed *F1-score*. Il modello GPT-4 ha una *precision* più alta che è dovuta alla tendenza del modello a predire molto frequentemente l'assenza di malattia, come si può osservare in Figura 3.5.

Modello	Accuracy	Recall	Precision	F1-Score
Logistic Regression	0.833	0.786	0.846	0.815
K-Nearest Neighbors	0.833	0.762	0.865	0.810
Studente Medicina	0.699	0.604	0.651	0.795
GPT-4	0.645	0.281	0.867	0.424

Tabella 3.9: Performance dei modelli nella diagnosi di malattie cardiache.

Capitolo 4

Integrazione e sviluppo dell'applicazione web

Per completare il progetto e renderlo accessibile tramite interfaccia utente, abbiamo realizzato una semplice architettura software composta da componenti backend, frontend e strumenti di sviluppo moderni.

4.1 Flask API

Abbiamo sviluppato una **API REST** utilizzando il micro-framework **Flask** in Python. L'API espone 3 endpoint:

- `/model_list`: che risponde con la lista di modelli presenti
- `/models`: che risponde con le statistiche del modello specificato nella richiesta
- `/predict`: che risponde con la predizione del caso clinico specificato nella richiesta

Questa architettura rende possibile l'integrazione del modello in qualsiasi applicazione client.

4.2 Interfaccia Streamlit

Abbiamo anche realizzato un front-end con la libreria **Streamlit**. Il front-end comunica con l'API Flask e permette all'utente di visualizzare le statistiche e le previsioni dei modelli da un'interfaccia grafica intuitiva.

L'interfaccia mette anche a disposizione un form dove inserire i dati clinici del paziente per cui si vuole fare la previsione.

4.3 Controllo di versione con Git

Durante lo sviluppo del progetto abbiamo utilizzato **git** come sistema di controllo versione, organizzando il codice in un repository **github** e lavorando in modo collaborativo. Ciò ha permesso una gestione efficace delle modifiche, creando opportunamente diversi *branch* per le varie funzionalità, il tracciamento dello storico e la suddivisione dei compiti.

4.4 Containerizzazione con Docker

Per facilitare la distribuzione e l'esecuzione del progetto in ambienti diversi, abbiamo utilizzato **Docker**. L'intero sistema (modello, API e interfaccia) è stato containerizzato all'interno di immagini Docker, consentendo l'avvio con un semplice comando, senza necessità di installazioni locali complesse.

4.5 Istruzioni per l'uso dell'applicazione web

Dopo aver clonato il repository dal seguente link utilizzando il comando `git clone https://github.com/PaoloWalsh/medical-diagnosis-aid.git`, è sufficiente eseguire nel terminale il comando `docker compose up --build`. Una volta completato l'avvio dei container, sarà possibile accedere all'applicativo aprendo un browser e navigando all'indirizzo `http://0.0.0.0:8501/`. A questo punto verrà visualizzata l'interfaccia front-end realizzata con Streamlit ,Figura 4.1, che comunica in modo trasparente con l'API back-end sviluppata in Flask.

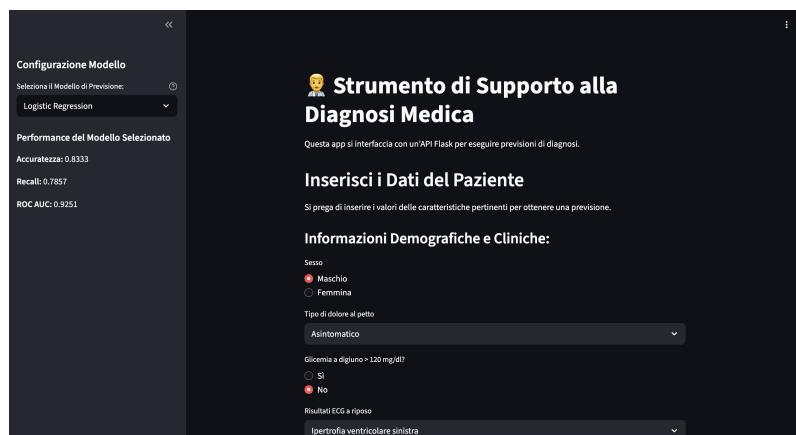


Figura 4.1: Interfaccia streamlit

Bibliografia

- [1] Cameron Razieh, Francesco Zaccardi, Joanne Miksza, Melanie J Davies, Anna L Hansell, Kamlesh Khunti, and Thomas Yates. Differences in the risk of cardiovascular disease across ethnic groups: Uk biobank observational study. *Nutrition, Metabolism and Cardiovascular Diseases*, 32(11):2594–2602, 2022.