# Obesity System Prediction

Gruppo di Lavoro: Bruno Michele, 775612, m.bruno150@studenti.uniba.it

AA 2024/25

Repository Github: `https://github.com/Miki004/progettoIcon`

## Contents

# 1 Introduzione

Il progetto si propone di sviluppare un sistema per la previsione del livello di obesità e la stima del peso corporeo di un individuo, combinando tecniche di apprendimento automatico con un sistema basato su conoscenza. L'applicazione integra modelli di classificazione e regressione per l'analisi dei dati, unitamente a un motore inferenziale sviluppato in Prolog per dedurre informazioni sullo stile di vita e il rischio di malattie correlate all'obesità. L'interfaccia utente è realizzata in Python utilizzando la libreria Streamlit, offrendo un'interazione intuitiva per l'inserimento dei dati e la visualizzazione dei risultati.

La rappresentazione della conoscenza avviene tramite una Knowledge Base (KB) in Prolog, che definisce regole inferenziali su abitudini alimentari, attività fisica, consumo calorico e fattori di rischio per malattie metaboliche. Le informazioni raccolte vengono trasformate in fatti Prolog per l'inferenza automatica, permettendo la valutazione del rischio di obesità e patologie associate, come diabete, ipertensione e sindrome metabolica.

L'analisi predittiva utilizza un modello addestrato su un dataset contenente dati su età, peso, altezza, abitudini alimentari, attività fisica e altre variabili rilevanti. Il preprocessing include tecniche di standardizzazione e encoding delle variabili categoriche, mentre la selezione del modello è stata ottimizzata tramite Grid Search con validazione incrociata. I migliori modelli individuati sono un Random Forest Classifier per la classificazione del livello di obesità e un Random Forest Regressor per la previsione del peso corporeo. L'architettura del sistema evidenzia l'integrazione di tre componenti principali:

- Modulo di apprendimento automatico: utilizza algoritmi supervisionati per la classificazione e la regressione.

- Motore inferenziale basato su Prolog: deduce informazioni qualitative dallo stile di vita dell'utente.

- Interfaccia interattiva in Streamlit: permette l'inserimento dei dati e la visualizzazione delle previsioni e delle inferenze.

In relazione agli argomenti del corso, il progetto affronta i seguenti temi chiave:

**Ragionamento automatico**: utilizzo di modelli di classificazione e regressione per la previsione dell'obesità e peso.

**Rappresentazione e ragionamento relazionale:**: inferenza logica basata su clausole di Horn per dedurre rischi e correlazioni tra abitudini e malattie.

Integrazione di sistemi basati su conoscenza e machine learning: sfruttando Prolog per la logica dichiarativa e Python per il machine learning e l'interfaccia utente.

Il progetto dimostra come sia possibile integrare tecniche di intelligenza artificiale basate su dati e su conoscenza per ottenere un sistema predittivo robusto ed esplicabile, capace di fornire sia previsioni quantitative che interpretazioni qualitative del rischio di obesità e delle condizioni di salute correlate. Il dataset utilizzato per questo studio contiene informazioni dettagliate su diversi fattori che possono influenzare l'obesità. Di seguito una descrizione delle feature presenti nel dataset:

- **Gender**: Sesso del soggetto (Maschio/Femmina).
- **Age**: Età del soggetto in anni.
- **Height**: Altezza in metri.
- **Weight**: Peso in kg.
- **family_history**: Presenza di obesità in famiglia (Sì/No).
- **FAVC**: Consumo frequente di cibo ad alto contenuto calorico (Sì/No).
- **FCVC**: Frequenza di consumo di verdure (da 1 a 3).
- **NCP**: Numero di pasti principali al giorno (da 1 a 4).
- **CAEC**: Consumo di cibo tra i pasti (Always, Frequently, Sometimes, No).
- **SMOKE**: Abitudine al fumo (Sì/No).
- **CH2O**: Consumo giornaliero di acqua (in litri).
- **SCC**: Monitoraggio del consumo calorico (Sì/No).
- **FAF**: Frequenza di attività fisica (da 0 a 3).
- **TUE**: Ore giornaliere trascorse utilizzando la tecnologia (da 0 a 10).
- **CALC**: Frequenza di consumo di alcol (Sometimes, Frequently, Always, No).
- **MTRANS**: Metodo di trasporto utilizzato (Public Transportation, Automobile, Walking, Motorbike, Bike).
- **Obesity**: Classe di obesità del soggetto (vari livelli di sovrappeso e obesità).

# 2 Argomenti di Interesse

- **Rappresentazione e Ragionamento Proposizionale**
- **Apprendimento Supervisionato**

# 3 Rappresentazione e Ragionamento Proposizionale

## 3.1 Sommario

Nel sistema sviluppato, il profilo dell'utente è rappresentato secondo la logica proposizionale, utilizzando clausole definite proposizionali e vincoli d'integrità per identificare eventuali incongruenze nei dati,il tutto rappresentato attraverso una Knowledge Base. L'acquisizione delle informazioni avviene tramite l'interfaccia grafica definita nel file **Interface.py**, la quale permette di raccogliere input dall'utente e generare dinamicamente fatti in formato Prolog. Questi fatti vengono poi utilizzati per inferire conoscenze e verificare condizioni logiche predefinite, supportando analisi come la valutazione del rischio di malattie cardiovascolari.

L'integrazione con i modelli di machine learning consente inoltre di effettuare previsioni sullo stato di obesità e sul peso dell'utente, combinando tecniche statistiche con un ragionamento basato su conoscenza dichiarativa.

## 3.2 Strumenti Utilizzati

Il sistema utilizza diversi strumenti software per l'implementazione e l'inferenza logica:

- **Prolog**: per la definizione di regole e vincoli logici.
- **Pyswip**:Per implementare il Prolog in Python.
- **Streamlit**: per l'implementazione di un'interfaccia web che consente l'interazione con il sistema.

## 3.3 Decisioni di Progetto

Nel processo di sviluppo del sistema, sono state prese alcune decisioni progettuali fondamentali. È stato scelto di utilizzare Prolog per la rappresentazione della conoscenza, poiché questo linguaggio permette un approccio dichiarativo all'inferenza logica. Inoltre, è stata implementata una serie di vincoli di integrità per garantire la coerenza e l'accuratezza delle informazioni inserite dagli utenti. Infine, il sistema è stato progettato con un'architettura modulare, separando la logica di inferenza dal livello applicativo, così da permettere una maggiore scalabilità e adattabilità a contesti diversi.

## 3.4 Definizione dello Stile di Vita

All'interno del sistema, il concetto di stile di vita viene analizzato attraverso due principali categorie. Un individuo viene considerato sedentario se risulta esplicitamente indicato tale o se presenta una vita sociale ridotta accompagnata da un elevato tempo dedicato all'uso di tecnologie. Al contrario, un individuo non sedentario viene considerato sportivo. L'identificazione dello stile di vita a rischio avviene attraverso la valutazione combinata di fattori come la sedentarietà, il fumo e la ridotta vita sociale.

### 3.4.1 Definizione del Metabolismo

Il metabolismo di un individuo è valutato tenendo conto di diversi elementi. Un metabolismo elevato è solitamente associato a una giovane età, alla pratica regolare di attività fisica, a una dieta equilibrata con una buona idratazione, e a un sonno adeguato.

### 3.4.2 Definizione della Dieta

La qualità dell'alimentazione di un individuo viene determinata in base al consumo calorico giornaliero, al livello di idratazione e al numero di pasti consumati.Una dieta può essere considerata scorretta quando presenta un apporto calorico elevato senza un'adeguata compensazione metabolica. Nel sistema viene inoltre analizzata la capacità dell'individuo di bilanciare un'alimentazione non ottimale attraverso il metabolismo e l'attività fisica.

```
% Una dieta scorretta dipende dalla sedentarietà e dal numero di pasti
dieta_scorretta(P) :- mangia_male(P), beve_poco(P).
dieta_scorretta(P) :- pasti(P, troppi), vita_sedentaria(P).

% Se una persona ha metabolismo alto e fa attività fisica, compensa la dieta
compensa_dieta(P) :- metabolismo_alto(P), sportivo(P).
compensa_dieta(P) :- metabolismo_alto(P), dieta_scorretta(P), beve_molto(P).
```

## 3.5 Classificazione del Rischio di Obesità

L'analisi del rischio di obesità si basa sull'interazione tra diversi fattori. Un individuo viene considerato a rischio quando possiede uno stile di vita non salutare abbinato a una dieta scorretta, quando l'alimentazione non è compensata da un metabolismo elevato oppure quando, nonostante lo stile di vita a rischio, non dispone di fattori protettivi sufficienti.

```
% ========= Classificazione del rischio di obesità =========
rischio_obesita(P) :- stile_vita_a_rischio(P), dieta_scorretta(P).      You, 3 days ago • f
rischio_obesita(P) :- dieta_scorretta(P), \+ compensa_dieta(P).
rischio_obesita(P) :- stile_vita_a_rischio(P), \+ protezione_obesita(P).
```

Il simbolo \+ rappresenta la negazione per fallimento (negation as failure), uno dei tratti distintivi di Prolog. In termini teorici, Prolog adotta il "closed world assumption": ciò che non può essere provato come vero, viene considerato falso. Se, dunque, Prolog non trova prove per compensa-dieta(P) o protezione-obesita(P), assume che tali predicati non valgano e conclude la presenza di rischio di obesità.

## 3.6 Associazione tra Obesità e Malattie

Il sistema stabilisce una correlazione tra l'obesità e diverse patologie. Il diabete è strettamente legato a un'alimentazione non equilibrata, mentre l'ipertensione è influenzata dalla combinazione di obesità, fumo e sedentarietà. Le malattie cardiovascolari sono associate all'eccesso di peso, all'abuso di fumo e a una dieta inadeguata, mentre l'apnea notturna è spesso conseguenza di disturbi del sonno aggravati dall'obesità. Infine, la sindrome metabolica emerge come un quadro complesso derivante dall'interazione tra obesità, ipertensione e diabete.

## 3.7 Vincoli di Integrità

Per assicurare la coerenza dei dati inseriti, sono stati definiti alcuni vincoli di integrità. Il sistema impedisce che un individuo possa essere contemporaneamente classificato come sedentario e sportivo, o che possa avere un elevato consumo calorico pur seguendo una dieta sana. Viene inoltre evitata la contraddizione di classificare una persona come dormiente sia poco che molto, e si assicura che un individuo non possa essere considerato a rischio obesità se presenta un profilo di vita sano.

```
% ========= Vincoli di integrità =========
% Una persona non può essere sedentaria e sportiva contemporaneamente
errore(P, 'Contraddizione: sedentario e sportivo insieme') :- sedentario(P, si), sportivo(P).

% Una persona non può mangiare bene e avere un alto consumo calorico
errore(P, 'Contraddizione: mangia bene ma ha consumo calorico alto') :- mangia_bene(P), consumo_calorico(P, alto).

% Una persona non può dormire poco e molto contemporaneamente
errore(P, 'Contraddizione: dorme poco e molto') :- dorme_poco(P), dorme_molto(P).

% Una persona non può avere stile di vita sano e essere a rischio obesità
errore(P, 'Contraddizione: stile di vita sano ma a rischio obesità') :-
    protezione_obesita(P),
    rischio_obesita(P).
```

## 3.8 Fatti nel Sistema

Nel sistema vengono definiti vari individui, ciascuno caratterizzato da parametri specifici riguardanti lo stile di vita, la dieta e le abitudini quotidiane, ogni volta dinamicamente aggiunti.

## 3.9 Conclusioni e Miglioramenti Futuri

Il sistema fornisce un modello di valutazione del rischio di obesità e delle malattie associate, basandosi su regole logiche che combinano diversi fattori di stile di

vita e salute. Il controllo di integrità aiuta a rilevare contraddizioni nei dati, garantendo coerenza nell'analisi.

- Si potrebbero aggiungere regole che suggeriscono azioni concrete per migliorare lo stato di salute (ad esempio, raccomandazioni su dieta ed esercizio fisico).
- Attualmente, ci sono vincoli di integrità che rilevano errori logici, ma potrebbero essere migliorati introducendo meccanismi di auto-correzione, come la ridefinizione automatica di un parametro contraddittorio.
- Invece di analizzare solo singoli individui, si potrebbero creare classi di utenti (ad esempio, adolescenti, adulti, anziani) con regole specifiche per ciascuna categoria.

# 4 Apprendimento Automatico

## 4.1 Sommario

L'apprendimento supervisionato è un ramo del machine learning in cui un modello viene istruito utilizzando un dataset con etichette, cioè dati già associati alle loro risposte corrette. Attraverso questo processo, l'algoritmo apprende schemi e relazioni tra input e output, così da effettuare previsioni su dati nuovi. Esistono due categorie principali: la classificazione, in cui il modello assegna ogni input a una classe predefinita (come nella classificazione binaria, che distingue semplicemente tra vero/falso o sì/no), e la regressione, in cui l'output è un valore numerico continuo che può assumere qualunque valore in un certo intervallo.

Nel mio progetto ho fatto uso dell'apprendimento automatico per predire sia la tipologia di obesità di un individuo (problema di classificazione), sia il peso (problema di regressione). I dati utilizzati per l'addestramento e la previsione vengono catturati attraverso un'interfaccia realizzata con Streamlit, che permette di inserire i valori delle variabili di interesse, inviarli al modello e visualizzare i risultati in modo interattivo.

## 4.2 Strumenti Utilizzati

I principali strumenti impiegati includono:

- **Scikit-learn**: libreria per il machine learning in Python, utilizzata per la creazione, addestramento e valutazione dei modelli.
- **Pandas e NumPy**: per la gestione e la manipolazione dei dati.
- **Matplotlib e Seaborn**: per la visualizzazione dei risultati.

## 4.3 Decisioni di Progetto

Gli iper-parametri sono valori impostati prima dell'addestramento che influenzano le prestazioni e la complessità del modello. A differenza dei parametri interni (come i pesi di una rete neurale), gli iper-parametri non vengono appresi dal modello, ma devono essere definiti a priori. La selezione ottimale di questi valori è un passaggio cruciale per garantire un buon bilanciamento tra underfitting e overfitting.

Per ottimizzare gli iper-parametri, ho utilizzato la tecnica della K-Fold Cross Validation (CV), che prevede la suddivisione del dataset in k sottoinsiemi (fold). Il modello viene quindi addestrato k volte, utilizzando a ogni iterazione un fold per il test e i restanti k-1 per l'addestramento. Questo approccio consente di valutare la capacità del modello di generalizzare su dati nuovi.

Per affinare ulteriormente la scelta degli iper-parametri, ho impiegato GridSearch con Cross Validation, una strategia che esplora tutte le combinazioni possibili di valori predefiniti per gli iper-parametri, individuando quella che garantisce le migliori prestazioni.

- **Addestramento** All'interno del progetto ho utilizzato la RepeatedKFold, che è una tecnica di Cross Validation per valutare le prestazioni di un modello di Machine Learning.

  In dettaglio, il parametro $n_splits = 5$ indica che i dati vengono suddivisi in 5 fold, ovvero il dataset viene diviso in 5 parti uguali. Ogni volta, 4 di queste parti vengono usate per addestrare il modello e la parte rimanente per testarlo, ripetendo il processo fino a coprire tutti i fold.

  Il parametro $n_repeats = 10$ specifica che l'intero processo di suddivisione in fold viene ripetuto 10 volte, rimescolando i dati ogni volta. Questo aiuta a ridurre la varianza nelle valutazioni del modello e fornisce una stima più affidabile delle sue prestazioni.

  In sintesi, questa configurazione permette di eseguire una validazione incrociata su 50 iterazioni (5 fold $\times$ 10 ripetizioni), offrendo una stima più robusta delle prestazioni del modello rispetto alla semplice K-Fold Cross Validation.

**Preprocessing dei dati**

- Valori Mancanti I dati mancanti nelle colonne categoriche sono sostituiti con la categoria più frequente, mentre quelli nelle colonne numeriche con la media. Questo evita che il modello perda informazioni escludendo intere righe o colonne, e al tempo stesso rende il dataset privo di valori NaN.

- La variabile "Obesity" è trasformata in valori numerici (0, 1, 2, ...) tramite un encoder apposito, poiché molti algoritmi di machine learning non accettano etichette testuali. In questo modo, il modello può apprendere correttamente le diverse classi di obesità.

- Si identificano le feature di tipo categorico (testo) e quelle numeriche, in modo da applicare trasformazioni differenti su ciascun gruppo. Le variabili categoriche richiedono infatti tecniche di codifica come One-Hot Encoding, mentre quelle numeriche vengono solitamente scalate per uniformarne i range.Nelle variabili categoriche non ordinarie (senza un ordine naturale), la One-Hot Encoding è una scelta standard in quanto trasforma le categorie in coordinate ortonormali, evitando di introdurre un ordine numerico artificioso (come accadrebbe con un LabelEncoder usato direttamente sulle feature predittive). Così, il modello non "pensa" che una categoria codificata come 2 sia "maggiore" di una codificata come 1; vengono invece trattate come diverse dimensioni binarie. Se le categorie sono molte, questa tecnica produce un'espansione del numero di feature, che però risulta più corretta per la maggior parte degli algoritmi.

Al termine di questi step, si ottiene un dataset privo di valori mancanti, con feature numeriche correttamente scalate e feature categoriche convertite in forma numerica appropriata, pronto per l'addestramento del modello.

## 4.4 Modelli utilizzati:

Il codice definisce due insiemi di modelli di Machine Learning, due per la classificazione e due per la regressione, specificando anche gli iperparametri su cui effettuare la ricerca a griglia.

I modelli di classificazione vengono utilizzati per prevedere categorie o classi a partire dai dati in input. Nel mio progetto vengono utilizzati per classificare tra le diverse categorie di obesità, mentre i modelli di regressione per predire il peso.

## 4.5 RandomForestClassifier

Questo modello è basato su una collezione di alberi decisionali (ensemble learning). Ogni albero viene addestrato su un sottoinsieme casuale dei dati e prende una decisione indipendente. La classificazione finale avviene con un voto di maggioranza tra tutti gli alberi. Il vantaggio principale del Random Forest è la sua robustezza: grazie alla combinazione di più alberi, riduce il rischio di overfitting e migliora la generalizzazione.

Nel codice, il modello viene ottimizzato variando:

- Il numero di alberi nella foresta $n_{estimators}$.
- La profondità massima degli alberi $max_{depth}$, che controlla la complessità del modello.
- Il numero minimo di campioni richiesti per dividere un nodo interno $min_{samples_split}$.
- Il numero minimo di campioni richiesti per una foglia $min_{samples_leaf}$.

## 4.6 LogisticRegression

La Regressione Logistica è un modello statistico che utilizza la funzione sigmoide per prevedere la probabilità di appartenenza a una classe. È particolarmente efficace per problemi di classificazione binaria, ma può essere estesa a più classi con strategie come softmax.

Nel codice, la regressione logistica viene configurata con:

- Il solver newton-cg, un metodo avanzato per ottimizzare la funzione di costo.
- Un limite massimo di iterazioni $max_{iter} = 500$ per garantire la convergenza.
- Il parametro di regolarizzazione C, che controlla la penalizzazione sui coefficienti per evitare overfitting. La penalizzazione l2, che aggiunge un termine di regolarizzazione per ridurre la varianza del modello.

I modelli di regressione vengono utilizzati per prevedere valori numerici continui.

## 4.7 RandomForestRegressor

Il Random Forest Regressor è la versione per regressione del Random Forest Classifier. Anche in questo caso, il modello costruisce più alberi decisionali, ma invece di classificare, ogni albero fornisce una stima numerica. Il valore finale della previsione è dato dalla media delle predizioni dei singoli alberi, rendendo il modello più stabile e meno sensibile ai dati rumorosi.

Gli iperparametri ottimizzati nel codice includono:

- Il numero di alberi nella foresta $n_{estimators}$.
- La profondità massima degli alberi $max_{depth}$.
- Il numero minimo di campioni richiesti per dividere un nodo $min_{samples_split}$.
- Il numero minimo di campioni richiesti per una foglia $min_{samples_leaf}$.

## 4.8 GradientBoostingRegressor

Il Gradient Boosting Regressor è un modello basato sulla tecnica del boosting, in cui gli alberi decisionali vengono costruiti in sequenza. Ogni nuovo albero cerca di correggere gli errori del precedente, minimizzando progressivamente la funzione di perdita. Questo approccio consente di ottenere un modello molto potente, particolarmente efficace con dataset strutturati.

Nel codice, il modello viene ottimizzato su:

- Il numero di alberi $n_{estimators}$.

- Il tasso di apprendimento $learning_{rate}$, che controlla l'impatto di ogni nuovo albero sulla previsione finale.
- La profondità massima degli alberi $max_{depth}$, che bilancia complessità e capacità di generalizzazione.

Gli iperparametri sono stati ottimizzati tramite la Gridsearch con Cross validation

## 4.9 Valutazione

La funzione plot-learning-curves ha il compito di mostrare come cambiano le prestazioni di un modello al variare della quantità di dati di addestramento. La funzione utilizza la K-Fold Cross Validation, suddividendo più volte il dataset in diverse partizioni per testare il modello e generare, a seconda del tipo di problema, differenti metriche di valutazione: nel caso di classificazione, plotta ad esempio l'accuratezza (convertita in 1 - accuracy), oltre a precision e recall; se invece il modello è di regressione, mostra metriche come MSE, MAE o $R^2$. Ripetendo il processo con diverse suddivisioni, si ottiene una stima più robusta, poiché il comportamento del modello viene esaminato su molteplici campioni e combinazioni di training e test.. Ciò rende la valutazione molto più solida, perché diminuisce la dipendenza da una singola suddivisione casuale dei dati e fornisce una panoramica più completa di come il modello si comporta su diverse porzioni e combinazioni di training e test.

## 4.10 Metriche per la Classificazione

Le metriche di classificazione servono a valutare la capacità del modello di distinguere correttamente tra le classi.

## 4.11 Accuracy

L'accuracy misura la percentuale di istanze correttamente classificate sul totale:

$$Accuracy = \frac{\#Predizionicorrette}{\#Totaledelleistanze} \tag{1}$$

Questa metrica è utile quando le classi sono bilanciate, ma può essere fuorviante in caso di dataset sbilanciati.

## 4.12 Precision

La precisione indica la percentuale di predizioni positive che sono effettivamente corrette:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Dove:

- $TP$ (True Positive) sono i campioni correttamente classificati come positivi.
- $FP$ (False Positive) sono i campioni classificati come positivi, ma che in realtà appartengono alla classe negativa.

## 4.13 Recall (Sensibilità)

Il recall misura la capacità del modello di catturare tutti i campioni effettivamente positivi:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

Dove:

- $FN$ (False Negative) sono i campioni della classe positiva classificati erroneamente come negativi.

## 4.14 F1-score

L'F1-score è la media armonica tra precision e recall:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

L'accuracy misura la quota di campioni classificati correttamente sul totale e risulta particolarmente intuitiva in dataset ben bilanciati. Se però una classe è molto più numerosa dell'altra, l'accuracy può risultare poco affidabile, perché anche un modello che ignora la classe rara può ottenere una percentuale di correttezza apparente elevata. In questi casi si passa a metriche di dettaglio come la precision e il recall: la prima (TP / (TP + FP)) rileva la capacità del modello di evitare falsi allarmi, mentre il secondo (TP / (TP + FN)) evidenzia la propensione a catturare effettivamente tutti i campioni della classe positiva. L'F1-score, essendo una media armonica di precision e recall, bilancia le due grandezze e risulta utile quando le classi sono sbilanciate.

## 4.15 Metriche per la Regressione

Le metriche di regressione misurano la capacità del modello di prevedere valori numerici con precisione.

## 4.16 Mean Squared Error (MSE)

L'MSE calcola la media dei quadrati delle differenze tra le predizioni e i valori reali:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{5}$$

Questa metrica penalizza fortemente gli errori grandi, rendendola sensibile agli outlier.

## 4.17 Mean Absolute Error (MAE)

Il MAE calcola la media dei valori assoluti delle differenze tra le predizioni e i valori reali:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{6}$$

A differenza dell'MSE, non eleva al quadrato gli errori e quindi è meno sensibile agli outlier.

## 4.18 $R^2$-score (Coefficiente di Determinazione)

L'$R^2$-score misura quanto bene il modello spiega la varianza dei dati:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{7}$$

Nella regressione, l'errore del modello nel prevedere valori numerici viene quantificato mediante metriche come l'MSE (media dei quadrati delle differenze tra predizione e valore reale) e l'MAE (media dei valori assoluti delle stesse differenze). L'MSE enfatizza maggiormente gli errori ampi, risultando più sensibile alla presenza di outlier, mentre l'MAE, non elevando lo scarto al quadrato, è meno influenzato da errori estremamente grandi. L'$R^2$-score, invece, confronta la bontà del modello rispetto a una semplice previsione basata sulla media dei valori osservati: valori prossimi a 1 indicano che l'algoritmo spiega gran parte della varianza dei dati, mentre valori vicini a 0 suggeriscono scarsa capacità di previsione.

## 4.19 Deviazione Standard

La deviazione standard misura la dispersione dei punteggi ottenuti nelle varie fold della cross-validation:

Deviazione Standard $= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$(8) Un valore alto indica una variabilità elevata nelle prestazioni del modello.

## 4.20 Varianza

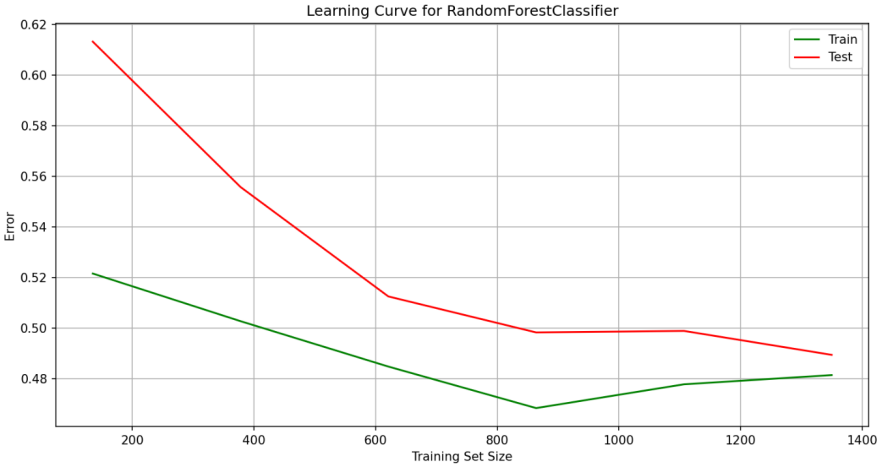
La varianza è il quadrato della deviazione standard:

$$Varianza = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{9}$$

Un modello con alta varianza può essere instabile e molto influenzato dai dati di training. La deviazione standard e la varianza descrivono quanto le prestazioni del modello fluttuano nelle diverse fold della cross-validation. Se la deviazione standard è elevata, le metriche ottenute su ciascuna suddivisione dei dati variano

molto: il modello è più sensibile al training set che riceve, e ciò può riflettere un certo grado di instabilità (alta varianza). Se invece la deviazione standard e la varianza sono ridotte, il comportamento del modello si mantiene coerente sui vari split, suggerendo che è meno condizionato dalle peculiarità del campione di addestramento e che le sue prestazioni sono più affidabili.

## 4.21 Valutazione Grafici




La prima figura (Learning Curve) mostra l'andamento dell'errore di classificazione (asse verticale) al crescere della quantità di dati di addestramento (asse orizzontale).

- Linea verde (Train): errore del modello sul sottoinsieme di training. In questo caso parte intorno a 0.52 e scende leggermente verso 0.48, indicando che il RandomForestClassifier riesce a ridurre l'errore sulle istanze di cui dispone man mano che ha più dati.
- Linea rossa (Test): errore sui dati di validazione, ossia i fold che il modello non vede in addestramento. Parte più alta (0.62) e si abbassa gradualmente fino a un valore vicino a 0.49. Il fatto che la curva rossa decresca in modo netto e si avvicini a quella verde suggerisce che, con un numero sufficiente di campioni, il modello non è vittima di pesante overfitting e beneficia effettivamente di più dati.

Nel complesso, l'errore finale di training e di test risulta abbastanza vicino, il che indica che il RandomForestClassifier è in grado di generalizzare discretamente al di fuori del training set, specie quando raggiunge la dimensione massima di campioni.

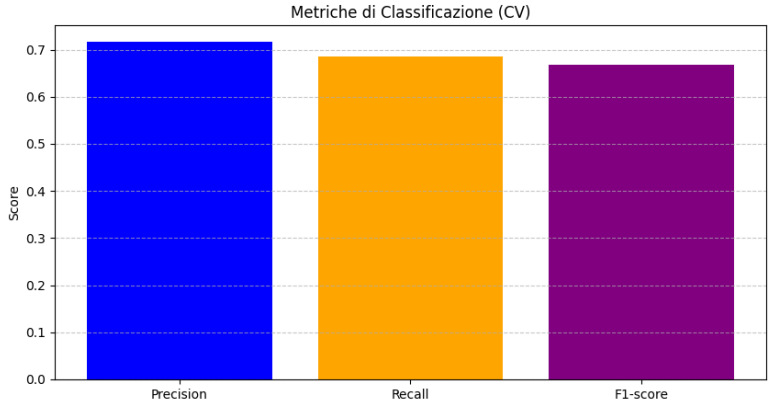
La seconda figura (Metriche di Classificazione in Cross Validation) riporta i valori medi di Precision, Recall e F1-score calcolati con un approccio di GridSearchCV e Cross Validation (in cui, nel codice, ogni modello viene addestrato

più volte su fold diversi). Dopo il tuning degli iperparametri e la scelta del modello migliore, si osserva:

- Precision circa 0.7: quando il modello predice una classe positiva, è corretta il 70% delle volte circa.
- Recall circa 0.65: riesce a recuperare correttamente il 65% dei casi positivi effettivi.
- F1-score circa 0.67: media armonica tra precision e recall, che risulta su un valore intermedio.

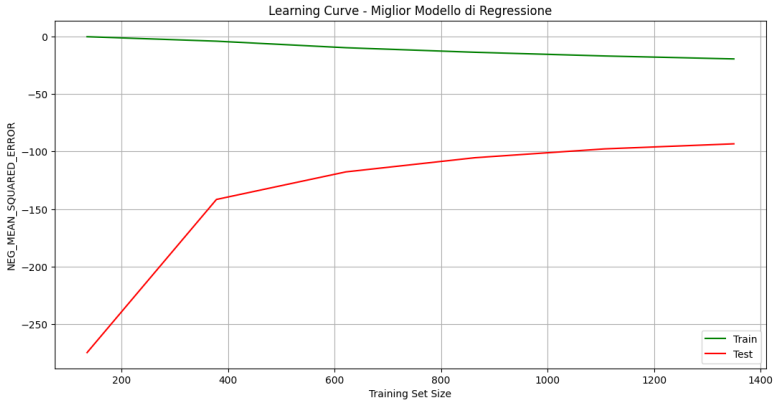
ogni modello è stato valutato in GridSearchCV (ricercando la combinazione migliore di iperparametri) e poi validato con $cross - val - score$ su più fold. Dalle varie configurazioni testate, è stato selezionato il modello con la massima accuracy media; in questo caso, il RandomForestClassifier con i parametri ottimali ha offerto i punteggi più elevati. Le barre nel grafico mostrano che il modello mantiene valori di precision, recall e F1 non troppo distanti fra loro, a indicare un comportamento bilanciato tra la capacità di riconoscere i positivi e quella di evitare i falsi allarmi.

La deviazione standard pari a 0.0243 e la varianza di 0.0006 indicano che i punteggi di accuracy ottenuti dal modello, nei vari fold di cross validation, sono tra loro molto simili. In altre parole, il modello si comporta in modo abbastanza stabile (poco dispersione) su diverse partizioni di training e di test, e questo solitamente è un buon segnale di robustezza.

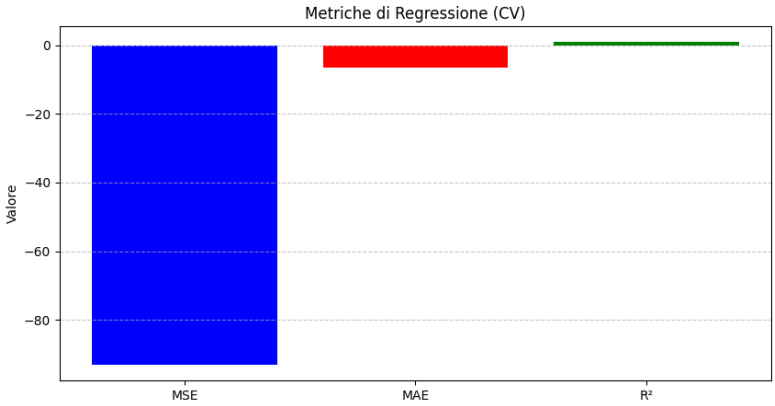



Questi valori, abbastanza allineati fra loro, indicano che il modello ha una buona ma non eccellente capacità di distinzione tra classi. Il fatto che precision sia leggermente più alta di recall segnala che il modello tende a sbagliare di più non riconoscendo alcuni positivi (falsi negativi), piuttosto che confondendo negativi come positivi (falsi positivi). In sintesi, si tratta di un risultato abbastanza stabile e bilanciato, ma con un margine di miglioramento nella capacità di individuare tutti i positivi.

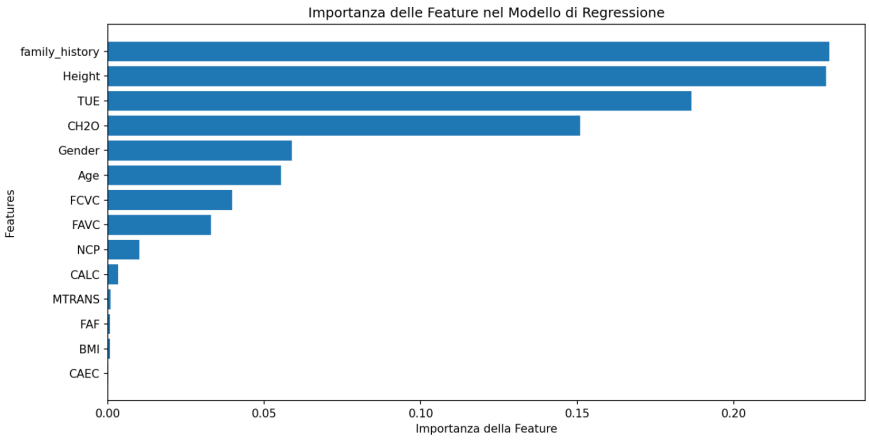
- Precision circa 0.70: quando il modello prevede “positivo”, ha ragione circa il 70% delle volte.
- Recall circa 0.65: recupera il 65% dei veri positivi totali, cioè a volte ancora ne “perde” qualcuno (falsi negativi).
- F1-score circa 0.67: media armonica tra precision e recall, che riassume il bilanciamento tra le due metriche.




Il primo grafico (Learning Curve) mostra come l’errore del modello di regressione, misurato con il neg-mean-squared-error, si comporti all’aumentare del numero di campioni di addestramento. La linea verde (train) parte da un errore vicino allo zero, segno che il modello riesce a “memorizzare” bene i dati quando il dataset è piccolo, ma il suo errore cresce leggermente con l’espandersi della dimensione del training set. La curva rossa (test) invece migliora in modo sensibile con l’aggiunta di più dati: parte da valori attorno a -250 e risale progressivamente verso -100. Questo andamento suggerisce che il modello soffra inizialmente di sovradattamento (overfitting) sul train, ma, acquisendo più informazioni, si stabilizza e ottiene risultati migliori sul test. Nonostante la riduzione del gap tra train e test, resta una certa differenza fra le due curve, indicando che, pur migliorando con un maggior numero di campioni, il modello non si comporta ancora in modo ottimale.




Nel grafico delle Metriche di Regressione (MSE, MAE, $R^2$) si nota che il modello ha ancora ampio margine di miglioramento. Il MSE (espresso come valore negativo in scikit-learn) arriva intorno a -90 in media, il MAE si aggira su valori intorno a 7-10, mentre l'$R^2$ resta moderato e lontano da 1. Guardando anche la deviazione standard dell'MSE, pari a circa 16.27, e la relativa varianza di 264.81, ci si rende conto che i risultati del modello possono variare in modo sensibile a seconda della suddivisione del dataset. Questo riflette una stabilità non perfetta e un certo livello di incertezza sulle previsioni.




l modello di regressione, pur avendo evidenziato in modo chiaro che "family-history" e "Height" siano le variabili più determinanti, non ha mostrato miglioramenti significativi neppure dopo aver generato nuove feature ispirate alle variabili più importanti e ridotto l'overfitting eliminando feature poco informative (come SMOKE e SCC). Questo suggerisce che il margine di miglioramento potrebbe essere limitato dalle caratteristiche stesse del dataset: anche ottimiz-

zando iperparametri e snellendo le feature, la curva di errore non converge a valori soddisfacenti, e la distanza tra train e test rimane sensibile. L'aggiunta di dati, la ricerca di interazioni più profonde o l'adozione di modelli più complessi potrebbero essere strade per spingersi oltre, ma con le attuali informazioni non è bastato puntare sulle feature già identificate come centrali per alzare in modo concreto le prestazioni complessive.Nel complesso, i risultati indicano un modello che, sebbene catturi bene alcuni fattori chiave, può essere ulteriormente perfezionato. Si potrebbero valutare nuove strategie di feature engineering, un ridimensionamento del set di feature e il ricorso a modelli di ensemble più avanzati per ridurre ancora l'errore e stabilizzare le previsioni. Con un'ulteriore ottimizzazione degli iperparametri e l'eventuale aggiunta di dati, ci si aspetterebbe di diminuire ulteriormente il divario tra la curva verde e la rossa, migliorando così il punteggio anche nei grafici di valutazione finale.

Nemmeno l'uso di tecniche di bilanciamento delle classi per le partizioni di classificazione non hanno portato i risultati sperati. In sostanza, i grafici rivelano che l'errore rimane elevato e la distanza tra l'errore di training e test non si è ridotta in modo soddisfacente, indicando che, con le informazioni attualmente disponibili, l'overfitting resta un problema e le performance non riescono a progredire oltre un certo limite.

# 5 Sviluppi Futuri

Oltre ai classici approcci di machine learning, si può guardare a forme di ragionamento più avanzate, come :

- Deep Learning con Embedding di Feature Complesse Nel caso si disponga di dati di natura diversa (immagini, testo, segnali di sensori), si potrebbe integrare un approccio deep learning che apprenda automaticamente feature ad alto livello. Se si hanno, ad esempio, dati relativi a immagini (pattern di distribuzione del grasso corporeo) o dati testuali (diari alimentari), una rete neurale potrebbe estrarre informazioni non immediatamente codificabili a mano e metterle in relazione con le variabili classiche (età, altezza, storia familiare, ecc.).
- Apprendimento Rinforzato per Consigli Personalizzati Oltre alla previsione del peso, l'apprendimento rinforzato (RL) potrebbe fornire raccomandazioni di azioni (dieta, piani di allenamento, etc.) e "imparare" quali interventi producono i risultati migliori a lungo termine. In tal modo si passerebbe da un modello puramente predittivo a uno decisionale, che sperimenta strategie e apprende dall'effetto cumulato.

# 6 Guida all'utilizzo

Assicurasi di avere tutti i requirements, scaricare il progetto,porsi all'interno della cartella del progetto ed eseguire il comando:

python -m streamlit run Interface.py

# 7 Riferimenti bibliografici


- Ragionamento logico: D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. 3/e. Cambridge University Press [Ch.3]
- Apprendimento supervisionato: D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. Cambridge University Press. 3rd Ed. [Ch.7]