Relazione Progetto Machine Learning

Matteo Ianeri

1 Obiettivo

L'obiettivo è predire l'aspettativa di vita.

2 Informazioni Dataset

Il dataset *Life Expectancy 2000-2015.csv* contiene dati relativi alle caratteristiche demografiche ed allo stile di vita di un campione della popolazione. Le grandezze riportate nelle colonne sono le seguenti:

- **Country**: 119 nomi di nazioni.
- **Year**: l'anno, dal 2000 al 2015 (inclusi).
- **Continent**: Continente.
- **Least Developed**
- **Life Expectancy**: valore dell'aspettativa di vita.
- **Population**: numerosità della popolazione.
- **CO2 emissions**: emissioni di anidride carbonica.
- **Health expenditure**: spesa per la salute.
- **Electric power consumption**: consumi elettrici.
- **Forest area**: aree verdi.
- **GDP per capita**: prodotto interno lordo per abitante.
- **Individuals using the Internet**: numero di utenti Internet.
- **Military expenditure**: spesa militare.
- **People practicing open defecation**: da non usare.
- **People using at least basic drinking water services**
- **Obesity among adults**: tasso di obesità.
- **Beer consumption per capita**: consumo di birra per abitante.

3 Relazione - Python

3.1 Preprocessing dei Dati

Per migliorare la precisione nella previsione dell'aspettativa di vita, ho implementato un processo di preprocessing sui dati. Questo processo è volto all'eliminazione di anomalie quali valori mancanti o duplicati.

3.2 Correlazione

ATTENZIONE: A causa di un errore che non sono riuscito a risolvere su Latex, i grafici verranno mostrati nelle ultime slide della relazione, pertanto ho fatto dei reindirizzamnti dove c'è scritto (Vedi figura ...) per vederli subito, all'interno della relazione)
In seguito, ho generato un corrplot per esaminare come varie caratteristiche demografiche e relative allo stile di vita influenzino l'aspettativa di vita nei diversi paesi del mondo, nel periodo 2000-2015.

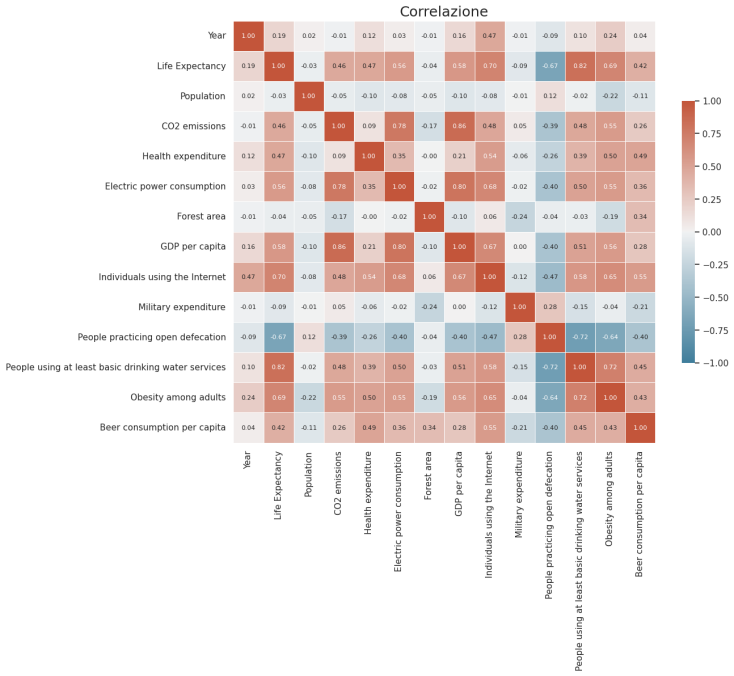

Figure 1:

3.3 Confronto tra variabili

Dopo aver esaminato il correlogramma, creo diversi scatterplot per poter analizzare meglio la correlazione tra aspettativa di vita e le caratteristiche che hanno una forte correlazione per continente. (Vedi Figura 13)
Dai seguenti scatterplot possiamo concludere che le persone che vivono in paesi a basse emissioni di carbonio possono raggiungere un'aspettativa di vita ragionevolmente elevata, ma non possono raggiungere alti livelli di reddito.
Fino a poco tempo fa, si riteneva vere le seguenti correlazioni: lo sviluppo umano dipende dalla crescita economica, la crescita economica richiede energia aggiuntiva e quindi porta a un aumento delle emissioni di gas serra.
Di conseguenza, l'umanità non può evolversi senza sfruttare ulteriormente le risorse della Terra.
Tuttavia, queste correlazioni non sono né così forti, né universalmente valide come si pensava inizialmente.
Julia K. Steinberger (University of Leeds e Alpen-Adria-Universität), J. Timmons Roberts, Glen P. Peters e Giovanni Baiocchi hanno pubblicato congiuntamente uno studio su Nature Climate Change, in cui esplorano i legami tra le emissioni di carbonio e lo sviluppo umano.
Durante le loro indagini, l'aspettativa di vita, il reddito e le emissioni di carbonio sono stati visti in termini di come si relazionano tra loro.
Inoltre, le emissioni sono state classificate come emissioni territoriali (ad esempio dovute alla produzione industriale nel rispettivo paese) o basate sul consumo (i valori netti sono derivati sommando i valori delle emissioni importate e sottraendo i valori delle emissioni esportate in base al carbonio incorporato nei beni e nei servizi). Nel complesso, è possibile dimostrare – in accordo con quest'ultimo approccio – che la maggior parte dei paesi esportatori di carbonio, come quelli dell'ex Unione Sovietica, dell'Europa orientale, del Medio Oriente o del Sud Africa, si trovano nella fascia media, sia in termini di aspettativa di vita che in termini di reddito.
I paesi importatori di carbonio, tuttavia, rappresentano un gruppo eccezionalmente eterogeneo, costituito da due estremi: da un lato, questi sono i paesi più poveri, costretti a importare costosi combustibili fossili e beni prodotti ad alta intensità di carbonio.
D'altra parte, questo gruppo comprende quei paesi con lo status socio-economico più sviluppato, la più alta aspettativa di vita e un reddito medio pro capite più elevato.
La ricerca ha rivelato che, sebbene le emissioni di carbonio sia territoriali che basate sul consumo siano altamente correlate con lo sviluppo umano, la forma e la forza della relazione tra emissioni di carbonio e reddito è completamente diversa dalla relazione tra emissioni di carbonio e aspettativa di vita.
Il confronto tra paesi mostra che, sebbene sia possibile ottenere contemporaneamente basse emissioni di carbonio e un'elevata aspettativa di vita, ciò vale solo quando il reddito della popolazione è moderato. Gli obiettivi economici e ambientali sembrano contraddirsi a vicenda; questo sembra certamente essere il caso dei valori più elevati del PIL pro capite.

3.4 Gestione Outliers

Prima di procedere con la previsione dell'aspettativa di vita, esaminerò le variabili correlate con l'aspettativa di vita per identificare la presenza di valori anomali (outliers).
In presenza di outliers, applicherò il metodo di winsorizzazione, che modifica i valori estremi al di fuori del primo e terzo quantile per adattarli più strettamente alla distribuzione centrale. Questo processo mira a minimizzare l'impatto degli outliers sui risultati finali ottenuti dai modelli di regressione. (Vedi Grafici 9 e 10)

```
def winsorize(data, percentile_lower=0.05, percentile_upper=0.95):
    """Winsorizes numerical columns in a DataFrame.

    Args:
        data (pandas.DataFrame): The DataFrame to winsorize.
        percentile_lower (float, optional): The percentile threshold for lower winsorization. Defaults to 0.05.
        percentile_upper (float, optional): The percentile threshold for upper winsorization. Defaults to 0.95.

    Returns:
        pandas.DataFrame: The DataFrame with winsorized numerical columns.
    """

    numerical_columns = data.select_dtypes(include=[np.number])

    for col in numerical_columns:
        lower_bound = data[col].quantile(percentile_lower)
        upper_bound = data[col].quantile(percentile_upper)


        data.loc[data[col] < lower_bound, col] = lower_bound


        data.loc[data[col] > upper_bound, col] = upper_bound

    return data

Life_Expectancy_00_15_winsorized = winsorize(Life_Expectancy_00_15)
```

Listing 1: Codice Python per utilizzare la Winsorizzazione

3.5 Codifica dei dati

Il dataset contiene diverse variabili categoriche che necessitano di essere trasformate per le analisi successive.
Il procedimento che adotterò è la codifica numerica: questa operazione consiste nell'assegnare a ciascuna categoria un corrispondente valore numerico unico, garantendo così che ogni categoria sia rappresentata da un numero specifico.

```
#Creo un oggetto labelEncoder
le = LabelEncoder()
Life_Expectancy_00_15_winsorized['Country'] = le.
    fit_transform(Life_Expectancy_00_15_winsorized['Country'
    ]) #Trasformo la colonna "Country" in numeri interi.
Life_Expectancy_00_15_winsorized['Least Developed'] = le.
    fit_transform(Life_Expectancy_00_15_winsorized['Least
    Developed']) #Trasformo la colonna "Least Developed" in
    numeri interi
Life_Expectancy_00_15_winsorized['Continent'] = le.
    fit_transform(Life_Expectancy_00_15_winsorized['Continent
    ']) #Trasformo la colonna "Continent" in numeri interi
```

Listing 2: Codice Python per codificare dati categorici

3.6 Suddivisione dei dati

```
# Definisco una lista delle colonne da escludere in X
colonne_da_escludere = [col for col in
    Life_Expectancy_00_15_winsorized.columns if col not in ['
    Life Expectancy', 'Population', 'Military Expenditure', '
    People Defecation', 'Forest Area']]

# Seleziono solo le colonne da includere in X
X = Life_Expectancy_00_15_winsorized[colonne_da_escludere]

# Definisco una variabile y contenente solo la colonna "Life
     Expectancy" del DataFrame life_exp.
y = Life_Expectancy_00_15_winsorized['Life Expectancy']

# Importazione della funzione train_test_split utilizzata
    per dividere il dataset in set di addestramento e test
from sklearn.model_selection import train_test_split

# Divisione del dataset 'X' e 'y' in set di addestramento e
    test:
# X_train, y_train: subset di dati e etichette per l'
    addestramento
# X_test, y_test: subset di dati e etichette per il test
# test size = 0.2: il 20% del dataset sar  utilizzato come
    set di test
# random state = 1: seed per il generatore di numeri casuali
     per garantire la riproducibilit  dei risultati di
    divisione
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=1)
```

Listing 3: Codice Python per suddividere il dataset in training e testing

3.7 Normalizzazione dei dati

Prima di procedere con la creazione dei modelli di regressione, è essenziale normalizzare i dati. La necessità di questo passaggio deriva dall'esistenza di variabili nel dataset che sono espresse in unità di misura diverse.
Per garantire che ogni variabile contribuisca equamente al modello, procederemo con la standardizzazione dell'intero dataset, portando ogni variabile ad avere una media di 0 e una deviazione standard di 1. Questo approccio facilita l'applicazione di tecniche di regressione, migliorando l'efficacia e l'accuratezza dei modelli predittivi.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Listing 4: Codice Python per normalizzare i dati

3.8 Modelli di machine learning

Per prevedere la variabile "Aspettativa di vita", impiego diversi modelli di regressione. Analizzo, per ciascun modello, un confronto tra i valori osservati e quelli previsti, esaminando le metriche di performance sia sulla fase di addestramento (training) che di validazione (test).
Infine, confronto i modelli per determinare quale offre le migliori previsioni, basandomi sul coefficiente R2 e assicurandomi l'assenza di overfitting.
La metodologia che adotterò per i seguenti modelli è la stessa.

3.8.1 GradientBoostingRegressor

Inizializzo il modello di regressione con specifiche tecniche quali il numero di stadi di boosting (100), il tasso di apprendimento (0.1), e un seed per la riproducibilità (30).
Il modello è addestrato utilizzando un set di dati di training (X train, y train), e successivamente è utilizzato per generare previsioni sia sui dati di training che di test (X test, y test).

Table 1: Metriche per il training set (Gradient Boosting)

Metrica	Valore
R^2	0.98
MAE	0.96
MSE	1.55
RMSE	1.24
EVS	0.98

Table 2: Metriche per il test set (Gradient Boosting)

Metrica	Valore
R^2	0.97
MAE	1.20
MSE	2.42
RMSE	1.55
EVS	0.97

Table 3: Confronto tra valori osservati e previsti per i primi 10 elementi

Osservato	Previsto
77.82	77.82
69.48	68.96
67.89	67.76
79.60	79.92
74.18	73.25
80.54	80.63
77.58	77.08
72.68	70.48
74.70	73.64
73.65	73.54

3.8.2 SVR

Inizializzo il modello SVR con specifiche tecniche quali il kernel RBF (Radial Basis Function), un parametro di regolarizzazione C pari a 100, un parametro gamma di 0.1 per definire l'ampiezza dell'RBF, e un epsilon di 0.1 che stabilisce la larghezza del tubo entro cui le previsioni sono considerate accettabili senza penalità.

Table 4: Metriche per il training set (SVR)

Metrica	Valore
R^2	1.00
MAE	0.25
MSE	0.24
RMSE	0.49
EVS	1.00

Table 5: Metriche per il test set (SVR)

Metrica	Valore
R^2	0.99
MAE	0.43
MSE	0.50
RMSE	0.71
EVS	0.99

Table 6: Confronto tra valori osservati e previsti per i primi 10 elementi

Osservato	Previsto
77.82	77.55
69.48	69.45
67.89	67.76
79.60	79.86
74.18	74.11
80.54	80.66
77.58	77.79
72.68	72.07
74.70	74.93
73.65	73.77

3.8.3 KNN

Inizializzo il modello KNeighborsRegressor specificando il numero di vicini da considerare, che in questo caso è impostato a 5. Questo parametro è fondamentale per determinare come il modello calcola i valori previsti basandosi sulla media dei valori dei k vicini più prossimi nel set di training.

Table 7: Metriche per il training set (KNN)

Metrica	Valore
R^2	0.99
MAE	0.40
MSE	0.50
RMSE	0.71
EVS	0.99

Table 8: Metriche per il test set (KNN)

Metrica	Valore
R^2	0.98
MAE	0.62
MSE	1.07
RMSE	1.04
EVS	0.98

Table 9: Confronto tra valori osservati e previsti per i primi 10 elementi

Osservato	Previsto
77.82	78.44
69.48	69.21
67.89	67.37
79.60	80.06
74.18	74.00
80.54	80.80
77.58	76.84
72.68	72.91
74.70	74.74
73.65	73.59

3.8.4 XGB

Inizializzo il modello XGBRegressor specificando parametri chiave come n estimators=100, che indica il numero di stadi di boosting da utilizzare. Il learning rate di 0.1 modula la velocità con cui il modello si adatta alle caratteristiche del problema durante l'addestramento, mentre random state=30 assicura la riproducibilità dei risultati, essenziale per la validazione sperimentale e la comparazione di modelli.

Table 10: Metriche per il training set (XGBoost)

Metrica	Valore
R^2	0.999
MAE	0.19
MSE	0.06
RMSE	0.25
EVS	1.00

Table 11: Metriche per il test set (XGBoost)

Metrica	Valore
R^2	0.99
MAE	0.52
MSE	0.57
RMSE	0.76
EVS	0.99

Table 12: Confronto tra valori osservati e previsti per i primi 10 elementi

Osservato	Previsto
77.82	77.64
69.48	69.58
67.89	67.75
79.60	79.88
74.18	73.94
80.54	80.30
77.58	77.29
72.68	72.13
74.70	73.05
73.65	73.75

3.8.5 Random Forest

Inizializzo il modello RandomForest con:
n estimators=100: Specifica il numero di alberi nel "bosco", con 100 alberi che contribuiscono a una media robusta e a una generalizzazione efficace.
random state=30: Assicura che i risultati siano riproducibili. Questo parametro fissa il seme del generatore di numeri casuali usato nella costruzione degli alberi, facilitando la comparazione diretta di modifiche e ottimizzazioni al modello.

Table 13: Metriche per il training set (Random Forest)

Metrica	Valore
R^2	1.00
MAE	0.24
MSE	0.14
RMSE	0.38
EVS	1.00

Table 14: Metriche per il test set (Random Forest)

Metrica	Valore
R^2	0.99
MAE	0.62
MSE	0.92
RMSE	0.96
EVS	0.99

Table 15: Confronto tra valori osservati e previsti per i primi 10 elementi

Osservato	Previsto
77.82	78.54
69.48	69.20
67.89	67.67
79.60	80.29
74.18	73.82
80.54	80.15
77.58	76.85
72.68	71.87
74.70	73.70
73.65	73.55

3.8.6 Decision Tree

Il modello DecisionTreeRegressor è configurato con random state=30 che fornisce un seme per il generatore di numeri casuali usato nelle divisioni binarie dell'albero. Questo assicura che i risultati siano riproducibili, essenziale per la validazione sperimentale e comparazione di modelli.

Table 16: Metriche per il training set (Decision Tree)

Metrica	Valore
R^2	1.00
MAE	0.00
MSE	0.00
RMSE	0.00
EVS	1.00

Table 17: Metriche per il test set (Decision Tree)

Metrica	Valore
R^2	0.95
MAE	0.83
MSE	3.45
RMSE	1.86
EVS	0.95

Table 18: Confronto tra valori osservati e previsti per i primi 10 elementi

Osservato	Previsto
77.82	78.60
69.48	69.07
67.89	65.38
79.60	80.40
74.18	74.59
80.54	80.39
77.58	77.39
72.68	72.83
74.70	74.12
73.65	73.23

3.8.7 AdaBoost

Il modello AdaBoostRegressor è configurato con i seguenti parametri:
base estimator=LinearRegression(): Indica che il modello base utilizzato per il boosting è la regressione lineare.
n estimators=100: Determina il numero di modelli successivi che vengono addestrati. Qui, 100 modelli di regressione lineare vengono potenzialmente migliorati durante il processo di boosting.
random state=30: Fissa il seme per la riproducibilità dei risultati, essenziale per la validazione sperimentale e comparazione di modelli.

Table 19: Metriche per il training set (AdaBoost)

Metrica	Valore
R^2	0.82
MAE	2.72
MSE	11.86
RMSE	3.44
EVS	0.82

Table 20: Metriche per il test set (AdaBoost)

Metrica	Valore
R^2	0.83
MAE	2.67
MSE	11.58
RMSE	3.40
EVS	0.84

Table 21: Confronto tra valori osservati e previsti per i primi 10 elementi

Osservato	Previsto
77.82	75.73
69.48	72.96
67.89	62.19
79.60	80.22
74.18	72.09
80.54	80.11
77.58	74.36
72.68	68.40
74.70	74.97
73.65	74.46

3.8.8 Linear

Inizializzo un modello che assume una relazione lineare tra le variabili indipendenti e la variabile dipendente, adattando una linea che minimizza la somma dei quadrati degli errori (metodo dei minimi quadrati).

Table 22: Metriche per il training set (Regressione Lineare)

Metrica	Valore
R^2	0.82
MAE	2.62
MSE	11.36
RMSE	3.37
EVS	0.82

Table 23: Metriche per il test set (Regressione Lineare)

Metrica	Valore
R^2	0.84
MAE	2.54
MSE	10.95
RMSE	3.31
EVS	0.84

Table 24: Confronto tra valori osservati e previsti per i primi 10 elementi

Osservato	Previsto
77.82	76.38
69.48	72.75
67.89	62.60
79.60	80.34
74.18	71.80
80.54	80.73
77.58	74.60
72.68	69.06
74.70	74.30
73.65	73.89

3.8.9 Polinomial

Il modello PolynomialFeatures viene configurato con:
degree=2: Imposta il grado del polinomio a 2, permettendo al modello di catturare non solo relazioni lineari, ma anche quadratiche tra le variabili indipendenti e la variabile dipendente.

Table 25: Metriche per il training set (Regressione Polinomiale)

Metrica	Valore
R^2	0.93
MAE	1.58
MSE	4.27
RMSE	2.07
EVS	0.93

Table 26: Metriche per il test set (Regressione Polinomiale)

Metrica	Valore
R^2	0.92
MAE	1.72
MSE	5.39
RMSE	2.32
EVS	0.92

Table 27: Confronto tra valori osservati e previsti per i primi 10 elementi

Osservato	Previsto
77.82	79.11
69.48	69.89
67.89	69.19
79.60	79.42
74.18	75.30
80.54	79.17
77.58	78.81
72.68	69.12
74.70	72.57
73.65	77.08

3.8.10 Ridge

Il modello Ridge viene configurato con:
alpha=1.0: controlla l'intensità della regolarizzazione L2. Un valore maggiore di alpha aumenta la penalità sulla grandezza dei coefficienti del modello, spingendo verso soluzioni con coefficienti più piccoli e potenzialmente più robusti, a scapito della complessità del modello.
random state=30: Fornisce un seme per il generatore di numeri casuali, garantendo così la riproducibilità dei risultati.

Table 28: Metriche per il training set (Regressione Ridge)

Metrica	Valore
R^2	0.82
MAE	2.62
MSE	11.36
RMSE	3.37
EVS	0.82

Table 29: Metriche per il test set (Regressione Ridge)

Metrica	Valore
R^2	0.84
MAE	2.54
MSE	10.94
RMSE	3.31
EVS	0.84

Table 30: Confronto tra valori osservati e previsti per i primi 10 elementi

Osservato	Previsto
77.82	76.39
69.48	72.71
67.89	62.60
79.60	80.33
74.18	71.79
80.54	80.71
77.58	74.60
72.68	69.07
74.70	74.30
73.65	73.88

3.8.11 Lasso

Il modello Lasso viene configurato con:
alpha=1.0: Questo parametro regola l'intensità della regolarizzazione L1. Un alpha più alto aumenta la penalità sui coefficienti, spingendo più coefficienti verso zero e promuovendo un modello più semplice e potenzialmente con una migliore generalizzazione in presenza di multicollinearità o dati di alta dimensione.
random state=30: Fornisce un seme per il generatore di numeri casuali.

Table 31: Metriche per il training set (Regressione Lasso)

Metrica	Valore
R^2	0.77
MAE	2.91
MSE	14.64
RMSE	3.83
EVS	0.77

Table 32: Metriche per il test set (Regressione Lasso)

Metrica	Valore
R^2	0.80
MAE	2.91
MSE	14.30
RMSE	3.78
EVS	0.80

Table 33: Confronto tra valori osservati e previsti per i primi 10 elementi

Osservato	Previsto
77.82	76.47
69.48	70.59
67.89	63.59
79.60	77.72
74.18	72.49
80.54	77.19
77.58	74.13
72.68	70.85
74.70	73.17
73.65	72.58

3.8.12 Extratree

Il modello ExtraTreesRegressor viene configurato con:
n estimators=100: aumenta il numero di alberi che aiuta a migliorare l'accuratezza del modello fino a un certo punto, ma anche ad incrementare il costo computazionale.
random state=30: Questo parametro fissa il seme del generatore di numeri casuali utilizzato nella selezione delle caratteristiche e nella divisione dei nodi, garantendo che la costruzione degli alberi sia consistente tra diverse esecuzioni.

Table 34: Metriche per il training set (Extra Trees)

Metrica	Valore
R^2	1.00
MAE	0.00
MSE	0.00
RMSE	0.00
EVS	1.00

Table 35: Metriche per il test set (Extra Trees)

Metrica	Valore
R^2	1.00
MAE	0.34
MSE	0.25
RMSE	0.50
EVS	1.00

Table 36: Confronto tra valori osservati e previsti per i primi 10 elementi

Osservato	Previsto
77.82	78.02
69.48	69.46
67.89	67.73
79.60	79.97
74.18	74.08
80.54	80.57
77.58	77.06
72.68	72.27
74.70	74.30
73.65	73.71

3.8.13 HistGradientBoosting

Table 37: Metriche per il training set (HistGradientBoosting)

Metrica	Valore
R^2	1.00
MAE	0.25
MSE	0.12
RMSE	0.35
EVS	1.00

Table 38: Metriche per il test set (HistGradientBoosting)

Metrica	Valore
R^2	0.99
MAE	0.55
MSE	0.55
RMSE	0.74
EVS	0.99

Table 39: Confronto tra valori osservati e previsti per i primi 10 elementi

Osservato	Previsto
77.82	78.48
69.48	69.20
67.89	67.84
79.60	80.11
74.18	74.13
80.54	80.24
77.58	77.53
72.68	71.94
74.70	73.50
73.65	74.04

3.8.14 SGD

Inizializzo SGDRegressor che utilizza il metodo di discesa stocastica del gradiente.
max iter=1000 specifica il numero massimo di passaggi attraverso i dati di addestramento prima di fermare l'addestramento.
tol=1e-3 stabilisce il criterio di arresto basato sul miglioramento dell'ottimizzazione.
random state=30 assicura la riproducibilità dei risultati facendo uso del generatore di numeri casuali per inizializzare i pesi.

Table 40: Metriche per il training set (SGDRegressor)

Metrica	Valore
R^2	0.82
MAE	2.62
MSE	11.37
RMSE	3.37
EVS	0.82

Table 41: Metriche per il test set (SGDRegressor)

Metrica	Valore
R^2	0.84
MAE	2.54
MSE	10.92
RMSE	3.30
EVS	0.84

Table 42: Confronto tra valori osservati e previsti per i primi 10 elementi

Osservato	Previsto
77.82	76.42
69.48	72.68
67.89	62.59
79.60	80.24
74.18	71.81
80.54	80.61
77.58	74.55
72.68	69.14
74.70	74.33
73.65	73.87

3.8.15 Elastic Net

Inizio importando ElasticNet dalla libreria sklearn.linear model, un modello avanzato di regressione che combina le penalizzazioni L1 e L2.
alpha=1.0 controlla l'intensità complessiva della regolarizzazione.
l1 ratio=0.5 determina il bilanciamento tra la regolarizzazione L1 (lasso) e L2 (ridge), offrendo un compromesso tra riduzione della dimensionalità (L1) e penalizzazione della complessità (L2).
random state=30 assicura la riproducibilità dei risultati.

Table 43: Metriche per il training set (Elastic Net)

Metrica	Valore
R^2	0.75
MAE	3.02
MSE	16.28
RMSE	4.04
EVS	0.75

Table 44: Metriche per il test set (Elastic Net)

Metrica	Valore
R^2	0.77
MAE	3.04
MSE	16.04
RMSE	4.00
EVS	0.77

Table 45: Confronto tra valori osservati e previsti per i primi 10 elementi

Osservato	Previsto
77.82	76.86
69.48	66.72
67.89	63.50
79.60	77.76
74.18	70.04
80.54	76.84
77.58	74.29
72.68	71.27
74.70	73.21
73.65	72.47

3.9 Confronto tra modelli

Dopo aver sviluppato i vari modelli di regressione, procedo a confrontarli attentamente per determinare quale tra essi predice i risultati più accuratamente. Questa analisi si basa sulle seguenti metriche valutative calcolate sul set di test dei dati:

- **R-squared (R^2)**: indica la percentuale della varianza della variabile dipendente che è predicibile dalle variabili indipendenti.
  Un R^2 di 1 indica che il modello predice perfettamente i dati senza alcun errore residuo, mentre un R^2 di 0 indica che il modello non predice meglio di un modello semplicemente basato sulla media dei dati.

- **Mean Absolute Error (MAE)**: misura la grandezza media degli errori in un set di previsioni, senza considerare la loro direzione (ignora se sono sovrastime o sottostime).
  È meno sensibile agli outlier rispetto al Mean Squared Error, perché non eleva gli errori al quadrato. Questo lo rende particolarmente adatto in contesti dove gli outlier non dovrebbero contribuire in modo eccessivo alla misura complessiva dell'errore.
  Un MAE di 0 significa che non ci sono errori nelle previsioni, il che è il caso ideale. Valori più alti indicano errori maggiori.

- **Mean Squared Error (MSE)**: è la media dei quadrati degli errori; cioè, la media delle differenze al quadrato tra i valori predetti e quelli reali. Elevare al quadrato gli errori significa che le discrepanze più grandi hanno un impatto proporzionalmente maggiore sul MSE rispetto alle discrepanze più piccole.
  Un MSE di 0 indica che il modello predice i valori osservati senza errori, il che è perfetto. Poiché MSE eleva gli errori al quadrato, esso penalizza più fortemente gli errori più grandi, quindi un valore basso in un contesto di dati con potenziali outlier è particolarmente desiderabile.

- **Explained Variance Score (EVS)**: misura la proporzione di varianza dei dati che è stata spiegata dal modello. In termini pratici, questo score valuta quanto bene il nostro modello può ricostruire i dati reali. Un punteggio più alto indica una migliore capacità di ricostruzione.
  È simile a R^2, ma mentre R^2 si basa sulle differenze rispetto alla media osservata, EVS si concentra sulle varianze spiegate rispetto alla varianza totale osservata, offrendo una visione leggermente diversa e complementare della performance del modello.

Utilizzo queste metriche per identificare il modello che non solo adatta meglio i dati, ma che offre anche la migliore generalizzazione su nuovi insiemi di dati non visti durante l'addestramento. (I valori che inserisco sono quelli ottenuti dai risultati dei codici precedenti).

Model-Name	R^2	MAE	MSE	RMSE	EVS
Elastic Net	0.75	3.02	16.28	4.04	0.75
Polynomial	0.77	2.91	14.64	3.83	0.77
Ridge	0.82	2.62	11.36	3.37	0.82
SGD	0.82	2.62	11.37	3.37	0.82
Linear	0.82	2.62	11.36	3.37	0.82
AdaBoost	0.82	2.72	11.86	3.44	0.82
Lasso	0.93	1.58	4.27	2.07	0.93
Gradient Boosting	0.98	0.96	1.55	1.24	0.98
XGBoost	0.99	0.19	0.06	0.25	1.00
KNN	0.99	0.40	0.50	0.71	0.99
Decision Tree	1.00	0.00	0.00	0.00	1.00
HistGradientBoosting	1.00	0.25	0.12	0.35	1.00
Random Forest	1.00	0.24	0.14	0.38	1.00
Extratree	1.00	0.00	0.00	0.00	1.00
SVM	1.00	0.25	0.24	0.49	1.00

Table 46: Confronto delle prestazioni dei modelli di regressione di training

Model-Name	R^2	MAE	MSE	RMSE	EVS
Elastic Net	0.77	3.04	16.04	4.00	0.77
Lasso	0.80	2.91	14.30	3.78	0.80
AdaBoost	0.83	2.67	11.58	3.40	0.84
SGD	0.84	2.54	10.92	3.30	0.84
Linear	0.84	2.54	10.95	3.31	0.84
Ridge	0.84	2.54	10.94	3.31	0.84
Polynomial	0.92	1.72	5.39	2.32	0.92
Decision Tree	0.95	0.83	3.45	1.86	0.95
Gradient Boosting	0.97	1.20	2.42	1.55	0.97
KNN	0.98	0.62	1.07	1.04	0.98
Random Forest	0.99	0.62	0.92	0.96	0.99
XGBoost	0.99	0.52	0.57	0.76	0.99
HistGradientBoosting	0.99	0.55	0.55	0.74	0.99
SVM	0.99	0.43	0.50	0.71	0.99
Extratree	1.00	0.34	0.25	0.50	1.00

Table 47: Confronto delle prestazioni dei modelli di regressione di testing

3.10 Conclusioni sul modello migliore

Analizzando entrambe le tabelle, i modelli con punteggi R2 bassi **(Elastic Net, Polynomial)** mostrano scarsa capacità predittiva, suggerendo underfitting.
Inoltre modelli come **Decision Tree, Extratree, Random Forest, HistGradientBoosting, e SVM** con punteggi R2 di 1.00 sul training e vicini a 1.00 sul testing possono indicare overfitting. Sono altamente specifici ai dati di training e possono perdere generalizzazione sui dati non visti.
Il miglior modello dovrebbe bilanciare tra alte prestazioni e generalizzazione, quindi in questo caso il **Gradient Boosting** ha alte prestazioni sia nei dati di training (R2: 0.98) che di testing (R2: 0.97), suggerendo un buon equilibrio.
Quindi posso concludere che il **Gradient Boosting** risulta il miglior modello, dato che ha alte prestazioni in entrambi i set di dati con R2 molto alto e bassi errori **(MAE, MSE, RMSE)**. Inoltre, non raggiunge il punteggio massimo di 1.00, riducendo il rischio di overfitting e rendendolo il più adatto per generalizzare su dati non visti.
XGBoost, Random Forest, e SVM sono valide alternative ma con una leggera inclinazione verso l'overfitting.

3.11 Deep Learning

Poichè la maggior parte dei modelli predice troppo bene l'aspettativa di vita, ho deciso di costruire un modello di ANN (Artificial Neural Network). Il modello implementato è una rete neurale sequenziale, una struttura composta da layer disposti in serie, dove ogni layer trasforma l'input ricevuto in un output mediante una funzione di attivazione, passando poi l'output come input al layer successivo.
Primo Layer (Dense): Comprende 64 unità con funzione di attivazione 'ReLU' (Rectified Linear Unit). La ReLU è utilizzata per introdurre non-linearità nel modello, consentendo di catturare relazioni complesse nei dati. Questo layer prende in input direttamente le feature di X train, la cui dimensione determina dinamicamente input shape, garantendo flessibilità e adattabilità del modello a vari tipi di dataset.
Secondo Layer (Dense): Identico al primo per numero di neuroni e tipo di attivazione, questo layer lavora sui dati trasformati dal primo layer per elaborare ulteriormente le relazioni tra le features.
Layer di Output: Consiste in un singolo neurone e, a differenza dei layer precedenti, non utilizza una funzione di attivazione. Questa configurazione è tipica nei problemi di regressione, dove l'obiettivo è predire un valore continuo. La mancanza di funzione di attivazione consente di produrre un range di valori reale non limitato. Vado a compilare il modello, con i seguenti parametri:
Ottimizzatore: 'Adam', un algoritmo basato sulla discesa stocastica del gradiente che si adatta automaticamente al contesto dei dati grazie alla sua capacità di regolare la velocità di apprendimento. Questo lo rende ideale per lavorare con dataset di dimensioni e caratteristiche variabili.
Funzione di Perdita: 'mean squared error', scelta per enfatizzare e penalizzare più severamente gli errori più grandi, una caratteristica desiderabile in molte applicazioni pratiche di regressione.
Metriche: Include 'MAE' (Mean Absolute Error) e 'MSE' (Mean Squared Error) per fornire una valutazione diretta e quantificabile degli errori del modello.

Dopo aver costruito e lanciato il modello, vado a stampare i primi 10 risultati per avere un confronto tra valori previsti ed osservati, e calcolo l'R.Squared.

Osservato	Previsto
77.82	79.40
69.48	40.91
67.89	66.96
79.60	80.34
74.18	73.20
80.54	80.82
77.58	73.52
72.68	67.33
74.70	73.21
73.65	76.28

Table 48: Confronto tra valori osservati e previsti dal modello

Metric	Value
R-squared Score	0.914

Table 49: Valore dello score R-squared

Creo un grafico della loss di addestramento (training loss) e della loss di validazione (validation loss) su diverse epoche durante il processo di addestramento della rete neurale.
Il grafico presenta due curve, una rossa e una verde, che rappresentano rispettivamente la loss di addestramento e la loss di validazione attraverso un numero di epoche durante l'addestramento di un modello.
Entrambe le curve mostrano un rapido declino nei primi stadi dell'addestramento, suggerendo che il modello sta imparando rapidamente e migliorando la sua capacità di adattarsi ai dati.
Dopo la fase iniziale di declino, entrambe le curve sembrano stabilizzarsi, indicando che il modello sta raggiungendo una convergenza, ovvero un punto in cui ulteriori apprendimenti migliorano minimamente la performance del modello sul set di addestramento e di validazione.
Si nota che le due curve restano vicine l'una all'altra per tutto il processo di addestramento, il che è un segnale positivo. Se la curva rossa (training loss) iniziasse a distaccarsi significativamente verso il basso rispetto alla curva verde (validation loss), questo sarebbe un segnale di overfitting.
C'è un punto marcato con un cerchio blu sulla curva di validazione, contrassegnato come "best epoch= 100". Questo suggerisce che il modello ha raggiunto la sua migliore performance sulla loss di validazione all'epoca 100. Questo punto può essere utilizzato come indicazione per fermare l'addestramento per evitare l'overfitting o per effettuare il rollback del modello a questa particolare (epoca)

Infine vado a creare un grafico per vedere quanto i valori osservati si distaccano da quelli previsti. (Vedi Figura 12)

4 Relazione - R

Senza ulteriori spiegazioni, presenterò i codici implementati in R accompagnati dai rispettivi output, allo scopo di confrontare le eventuali differenze con le implementazioni analoghe in Python.
Omettendo la parte dedicata all'EDA, mi concentro per lo più sulla creazione dei modelli di regressione.

4.1 Gestione Outliers

Esamino le variabili correlate con l'aspettativa di vita per identificare la presenza di valori anomali (outliers).
In presenza di outliers, applicherò il metodo di winsorizzazione, che modifica i valori estremi al di fuori del primo e terzo quantile per adattarli più strettamente alla distribuzione centrale. Questo processo mira a minimizzare l'impatto degli outliers sui risultati finali ottenuti dai modelli di regressione.

```r
winsorize <- function(data, percentile_lower=0.05,
    percentile_upper=0.95) {
  # Winsorizes numerical columns in a DataFrame.

  # Arguments:
  #   data (data.frame): The DataFrame to winsorize.
  #   percentile_lower (numeric, optional): The percentile
    threshold for lower winsorization. Defaults to 0.05.
  #   percentile_upper (numeric, optional): The percentile
    threshold for upper winsorization. Defaults to 0.95.

  numerical_columns <- data[, sapply(data, is.numeric)]

  for (col in names(numerical_columns)) {
    lower_bound <- quantile(data[[col]], percentile_lower)
    upper_bound <- quantile(data[[col]], percentile_upper)

    data[[col]][data[[col]] < lower_bound] <- lower_bound
    data[[col]][data[[col]] > upper_bound] <- upper_bound
  }

  return(data)
}

Life_Expectancy_00_15_winsorized <- winsorize(Life_
    Expectancy_00_15)
```

Listing 5: Codice R per utilizzare la Winsorizzazione

4.2 Codifica dei dati

Il dataset contiene diverse variabili categoriche che necessitano di essere trasformate per le analisi successive.
Il procedimento che adotterò è la codifica numerica: questa operazione consiste nell'assegnare a ciascuna categoria un corrispondente valore numerico unico, garantendo così che ogni categoria sia rappresentata da un numero specifico.

```
# Trasformo le colonne "Country", "Least Developed" e "
    Continent" in numeri interi
Life_Expectancy_00_15_winsorized$Country <- as.integer(as.
    factor(Life_Expectancy_00_15_winsorized$Country))
Life_Expectancy_00_15_winsorized$Least.Developed <- as.
    integer(as.factor(Life_Expectancy_00_15_winsorized$Least.
    Developed))
Life_Expectancy_00_15_winsorized$Continent <- as.integer(as.
    factor(Life_Expectancy_00_15_winsorized$Continent))
str(Life_Expectancy_00_15_winsorized)
```

Listing 6: Codice R per utilizzare la Winsorizzazione

4.3 Suddivisione dei dati

```r
# Definisco una lista delle colonne da escludere in X
colonne_da_escludere <- c('Life.Expectancy', 'Population', '
    Military.expenditure', 'People.practicing.open.defecation
    ', 'Forest.area')

# Seleziono solo le colonne da escludere in X
X <- Life_Expectancy_00_15_winsorized[, !names(
    Life_Expectancy_00_15_winsorized) %in%
    colonne_da_escludere]

# Definisco la variabile y contenente solo la colonna "
    Life_Expectancy" del DataFrame
y <- Life_Expectancy_00_15_winsorized$Life.Expectancy

# Divido il dataset in training set e test set
set.seed(1) # Imposto il seed per la riproducibilit  dei
    risultati
train_indices <- sample(1:nrow(X), 0.8*nrow(X)) # 80% dei
    dati per il training set
X_train <- X[train_indices, ]
X_test <- X[-train_indices, ]
y_train <- y[train_indices]
y_test <- y[-train_indices]
cat("Dimensioni del training set:", dim(X_train), dim(
    y_train), "\n")
cat("Dimensioni del test set:", dim(X_test), dim(y_test), "\
    n")
```

Listing 7: Codice Python per suddividere il dataset in training e testing

4.4 Normalizzazione dei dati

Prima di procedere con la creazione dei modelli di regressione, è essenziale normalizzare i dati. La necessità di questo passaggio deriva dall'esistenza di variabili nel dataset che sono espresse in unità di misura diverse.
Per garantire che ogni variabile contribuisca equamente al modello, procederemo con la standardizzazione dell'intero dataset, portando ogni variabile ad avere una media di 0 e una deviazione standard di 1. Questo approccio facilita l'applicazione di tecniche di regressione, migliorando l'efficacia e l'accuratezza dei modelli predittivi.

```
# Calcolo delle medie e deviazioni standard per la
    standardizzazione
preProcessParams <- preProcess(X_train, method = c("center",
    "scale"))

# Applica la standardizzazione ai dati di addestramento
X_train <- predict(preProcessParams, X_train)

# Applica la stessa standardizzazione ai dati di test
X_test <- predict(preProcessParams, X_test)
```

Listing 8: Codice Python per normalizzare i dati

4.5 Modelli di machine learning

Per prevedere la variabile "Aspettativa di vita", impiego diversi modelli di regressione. Analizzo, per ciascun modello, un confronto tra i valori osservati e quelli previsti, esaminando le metriche di performance sia sulla fase di addestramento (training) che di validazione (test).
Infine, confronto i modelli per determinare quale offre le migliori previsioni, basandomi sul coefficiente R2 e assicurandomi l'assenza di overfitting.
La metodologia che adotterò per i seguenti modelli è la stessa.

4.5.1 GradientBoostingRegressor

Table 50: Metriche per il training set (GradientBoostingRegressor)

Metrica	Valore
R^2	0.9512578
MAE	1.398079
MSE	3.213459
RMSE	1.792612
EVS	0.9216937

Table 51: Metriche per il test set (GradientBoostingRegressor)

Metrica	Valore
R^2	0.943776
MAE	1.508827
MSE	4.012703
RMSE	2.003173
EVS	0.9150318

Table 52: Confronto tra valori osservati e previsti

Osservato	Previsto
74.288	70.71636
75.228	72.70361
76.221	75.02651
76.914	75.16493
74.311	74.50845
74.644	74.70862
75.199	74.69713
75.661	75.39703
76.09	76.20508
52.6852	53.9798

4.5.2 SVR

Table 53: Metriche per il training set (SVR)

Metrica	Valore
R^2	0.9949475
MAE	0.5118864
MSE	0.3328655
RMSE	0.576945
EVS	0.9895819

Table 54: Metriche per il test set (SVR)

Metrica	Valore
R^2	0.9929195
MAE	0.5753134
MSE	0.4881315
RMSE	0.6986641
EVS	0.9847607

Table 55: Confronto tra valori osservati e previsti

Osservato	Previsto
74.288	74.87094
75.228	74.99933
76.221	76.20717
76.914	76.79738
74.311	73.56596
74.644	73.40296
75.199	74.66815
75.661	74.98479
76.09	74.66127
52.6852	51.85077

4.5.3 KNN

Table 56: Metriche per il training set (KNN)

Metrica	Valore
R^2	0.9910372
MAE	0.4283022
MSE	0.5833809
RMSE	0.7637938
EVS	0.9905187

Table 57: Metriche per il test set (KNN)

Metrica	Valore
R^2	0.9832741
MAE	0.6575753
MSE	1.155844
RMSE	1.075102
EVS	0.9803439

Table 58: Confronto tra valori osservati e previsti

Osservato	Previsto
74.288	74.1026
75.228	75.103
76.221	76.5852
76.914	77.4412
74.311	73.6122
74.644	74.1292
75.199	74.0318
75.661	73.621
76.09	73.228
52.6852	52.6852

4.5.4 XGB

Table 59: Metriche per il training set (XGBoost)

Metrica	Valore
R^2	0.9976924
MAE	0.2893762
MSE	0.1521488
RMSE	0.3900626
EVS	0.9950553

Table 60: Metriche per il test set (XGBoost)

Metrica	Valore
R^2	0.9891158
MAE	0.6265699
MSE	0.7675299
RMSE	0.8760879
EVS	0.9805733

Table 61: Confronto tra valori osservati e previsti

Osservato	Previsto
74.288	73.05078
75.228	74.00745
76.221	76.49613
76.914	76.7205
74.311	74.04183
74.644	74.94984
75.199	74.75462
75.661	75.23148
76.09	76.15604
52.6852	52.48174

4.5.5 Random Forest

Table 62: Metriche per il training set (Random Forest)

Metrica	Valore
R^2	0.9974431
MAE	0.3011079
MSE	0.1819499
RMSE	0.4265558
EVS	0.9964726

Table 63: Metriche per il test set (Random Forest)

Metrica	Valore
R^2	0.9895759
MAE	0.6344116
MSE	0.7894101
RMSE	0.8884875
EVS	0.9851999

Table 64: Confronto tra valori osservati e previsti

Osservato	Previsto
74.288	72.74806
75.228	74.06992
76.221	75.4549
76.914	76.28392
74.311	73.64073
74.644	74.03203
75.199	73.89213
75.661	75.08223
76.09	74.82412
52.6852	52.80888

4.5.6 Decision Tree

Table 65: Metriche per il training set (Decision Tree)

Metrica	Valore
R^2	1
MAE	3.732332e-16
MSE	1.060793e-29
RMSE	3.256981e-15
EVS	0.9999208

Table 66: Metriche per il test set (Decision Tree)

Metrica	Valore
R^2	0.9699003
MAE	0.6912037
MSE	2.082534
RMSE	1.443099
EVS	0.9538612

Table 67: Confronto tra valori osservati e previsti

Osservato	Previsto
74.288	74.579
75.228	75.423
76.221	76.562
76.914	76.562
74.311	73.936
74.644	74.938
75.199	74.938
75.661	75.878
76.09	75.878
52.6852	52.6852

4.5.7 AdaBoost

Table 68: Metriche per il training set (AdaBoost)

Metrica	Valore
R^2	0.952
MAE	1.373
MSE	3.179
RMSE	1.783
EVS	0.921

Table 69: Metriche per il test set (AdaBoost)

Metrica	Valore
R^2	0.946
MAE	1.489
MSE	3.891
RMSE	1.973
EVS	0.921

Table 70: Confronto tra valori osservati e previsti

Osservato	Previsto
74.288	70.662
75.228	73.119
76.221	75.547
76.914	75.547
74.311	74.146
74.644	74.146
75.199	75.019
75.661	75.363
76.090	74.795
52.685	53.200

4.5.8 Linear

Table 71: Metriche per il training set (Regressione Lineare)

Metrica	Valore
R^2	0.8234648
MAE	2.622742
MSE	11.43268
RMSE	3.381225
EVS	0.8088625

Table 72: Metriche per il test set (Regressione Lineare)

Metrica	Valore
R^2	0.8478511
MAE	2.50955
MSE	10.67666
RMSE	3.267515
EVS	0.8314863

Table 73: Confronto tra valori osservati e previsti

Osservato	amp; Previsto
74.288	68.66134
75.228	69.34084
76.221	72.10013
76.914	73.01854
74.311	69.14041
74.644	69.38144
75.199	69.70257
75.661	70.20805
76.09	71.29082
52.6852	55.68944

4.5.9 Polinomial

Table 74: Metriche per il training set (Regressione Polinomiale)

Metrica	Valore
R^2	0.867942
MAE	2.248271
MSE	8.552277
RMSE	2.924428
EVS	0.840998

Table 75: Metriche per il test set (Regressione Polinomiale)

Metrica	Valore
R^2	0.880455
MAE	6.546039
MSE	65.025535
RMSE	8.063841
EVS	0.849182

Table 76: Confronto tra valori osservati e previsti

Osservato	Previsto
74.288000	68.844379
75.228000	71.214861
76.221000	76.950305
76.914000	79.533240
74.311000	67.310278
74.644000	68.663422
75.199000	69.670593
75.661000	70.541761
76.090000	71.470410
52.685200	37.318948

4.5.10 Ridge

Table 77: Metriche per il training set (Regressione Ridge)

Metrica	Valore
R^2	0.8116144
MAE	2.660701
MSE	12.3518
RMSE	3.514513
EVS	0.8068337

Table 78: Metriche per il test set (Regressione Ridge)

Metrica	Valore
R^2	0.8378731
MAE	2.583453
MSE	11.76226
RMSE	3.429615
EVS	0.8286462

Table 79: Confronto tra valori osservati e previsti

Osservato	Previsto
74.288	68.7242
75.228	69.53759
76.221	71.86225
76.914	72.747
74.311	68.97469
74.644	69.34477
75.199	69.77064
75.661	70.45916
76.09	71.60333
52.6852	56.44502

4.5.11 Lasso

Table 80: Metriche per il training set (Regressione Lasso)

Metrica	Valore
R^2	0.7949703
MAE	2.907593
MSE	14.65499
RMSE	3.828184
EVS	0.7978642

Table 81: Metriche per il test set (Regressione Lasso)

Metrica	Valore
R^2	0.8202588
MAE	2.926152
MSE	14.74383
RMSE	3.839769
EVS	0.809778

Table 82: Confronto tra valori osservati e previsti

Osservato	Previsto
74.288	68.93874
75.228	69.83326
76.221	72.30001
76.914	72.88797
74.311	69.84581
74.644	69.96051
75.199	70.30028
75.661	; 70.79259
76.09	71.67998
52.6852	57.69889

4.5.12 ExtraTree

Table 83: Metriche per il training set (Extra Trees)

Metrica	Valore
R^2	0.9969381
MAE	0.288067
MSE	0.2078128
RMSE	0.4558649
EVS	0.9957968

Table 84: Metriche per il test set (Extra Trees)

Metrica	Valore
R^2	0.9868438
MAE	0.6502148
MSE	0.939559
RMSE	0.9693085
EVS	0.982528

Table 85: Confronto tra valori osservati e previsti

Osservato	Previsto
74.288	73.59034
75.228	73.40339
76.221	75.30501
76.914	76.34815
74.311	73.84282
74.644	73.99493
75.199	73.3445
75.661	74.90684
76.09	75.35151
52.6852	52.83776

4.5.13 HistGradientBoosting

Table 86: Metriche per il training set (HistGradientBoosting)

Metrica	Valore
R^2	0.973
MAE	0.999
MSE	1.785
RMSE	1.336
EVS	0.956

Table 87: Metriche per il test set (HistGradientBoosting)

Metrica	Valore
R^2	0.964
MAE	1.192
MSE	2.616
RMSE	1.618
EVS	0.948

Table 88: Confronto tra valori osservati e previsti

Osservato	Previsto
74.288	72.796
75.228	74.172
76.221	75.958
76.914	76.043
74.311	74.820
74.644	74.820
75.199	74.783
75.661	75.693
76.090	76.345
52.685	52.573

4.5.14 SGD

Table 89: Metriche per il training set (SGDRegressor)

Metrica	Valore
R^2	0.8234229
MAE	2.619456
MSE	11.43564
RMSE	3.381662
EVS	0.8091154

Table 90: Metriche per il test set (SGDRegressor)

Metrica	Valore
R^2	0.8476893
MAE	2.507328
MSE	10.69715
RMSE	3.27065
EVS	0.8315497

Table 91: Confronto tra valori osservati e previsti

Osservato	Previsto
74.288	68.66665
75.228	69.36943
76.221	72.08385
76.914	73.00334
74.311	69.12045
74.644	69.36887
75.199	69.69725
75.661	70.21228
76.09	71.28921
52.6852	55.65972

4.5.15 ElasticNet

Table 92: Metriche per il training set (Elastic Net)

Metrica	Valore
R^2	0.8234258
MAE	2.619416
MSE	11.43547
RMSE	3.381637
EVS	0.8091207

Table 93: Metriche per il test set (Elastic Net)

Metrica	Valore
R^2	0.847689
MAE	2.507503
MSE	10.69774
RMSE	3.27074
EVS	0.8316021

Table 94: Confronto tra valori osservati e previsti

Osservato	Previsto
74.288	68.66661
75.228	69.3667
76.221	72.08279
76.914	73.0023
74.311	69.1189
74.644	69.36775
75.199	69.69585
75.661	70.21112
76.09	71.28948
52.6852	55.66645

4.6 Confronto tra modelli

Dopo aver sviluppato i vari modelli di regressione, procedo a confrontarli attentamente per determinare quale tra essi predice i risultati più accuratamente. Questa analisi si basa sulle seguenti metriche valutative calcolate sul set di test dei dati:

- **R-squared (R^2)**: indica la percentuale della varianza della variabile dipendente che è predicibile dalle variabili indipendenti.
  Un R^2 di 1 indica che il modello predice perfettamente i dati senza alcun errore residuo, mentre un R^2 di 0 indica che il modello non predice meglio di un modello semplicemente basato sulla media dei dati.

- **Mean Absolute Error (MAE)**: misura la grandezza media degli errori in un set di previsioni, senza considerare la loro direzione (ignora se sono sovrastime o sottostime).
  È meno sensibile agli outlier rispetto al Mean Squared Error, perché non eleva gli errori al quadrato. Questo lo rende particolarmente adatto in contesti dove gli outlier non dovrebbero contribuire in modo eccessivo alla misura complessiva dell'errore.
  Un MAE di 0 significa che non ci sono errori nelle previsioni, il che è il caso ideale. Valori più alti indicano errori maggiori.

- **Mean Squared Error (MSE)**: è la media dei quadrati degli errori; cioè, la media delle differenze al quadrato tra i valori predetti e quelli reali. Elevare al quadrato gli errori significa che le discrepanze più grandi hanno un impatto proporzionalmente maggiore sul MSE rispetto alle discrepanze più piccole.
  Un MSE di 0 indica che il modello predice i valori osservati senza errori, il che è perfetto. Poiché MSE eleva gli errori al quadrato, esso penalizza più fortemente gli errori più grandi, quindi un valore basso in un contesto di dati con potenziali outlier è particolarmente desiderabile.

- **Explained Variance Score (EVS)**: misura la proporzione di varianza dei dati che è stata spiegata dal modello. In termini pratici, questo score valuta quanto bene il nostro modello può ricostruire i dati reali. Un punteggio più alto indica una migliore capacità di ricostruzione.
  È simile a R^2, ma mentre R^2 si basa sulle differenze rispetto alla media osservata, EVS si concentra sulle varianze spiegate rispetto alla varianza totale osservata, offrendo una visione leggermente diversa e complementare della performance del modello.

Utilizzo queste metriche per identificare il modello che non solo adatta meglio i dati, ma che offre anche la migliore generalizzazione su nuovi insiemi di dati non visti durante l'addestramento. (I valori che inserisco sono quelli ottenuti dai risultati dei codici precedenti).

Model-Name	R^2	MAE	MSE	RMSE	EVS
Lasso	0.79	2.90	14.65	3.82	0.79
Ridge	0.81	2.66	12.35	3.51	0.80
Linear	0.82	2.62	11.43	3.38	0.80
Elastic Net	0.83	2.61	11.43	3.38	0.80
SGD	0.83	2.61	11.43	3.38	0.80
Polynomial	0.86	2.24	8.55	2.92	0.8
Gradient Boosting	0.95	1.39	3.21	1.79	0.92
AdaBoost	0.95	1.37	3.17	1.78	0.921
HistGradientBoosting	0.97	0.99	1.78	1.33	0.95
KNN	0.99	0.42	0.58	0.76	0.99
Random Forest	0.99	0.30	0.18	0.42	0.99
XGBoost	0.99	0.28	0.15	0.39	0.99
Extratree	0.99	0.28	0.20	0.45	0.99
SVM	0.99	0.51	0.33	0.57	0.98
Decision Tree	1.00	0.00	0.00	0.00	1.00

Table 95: Confronto delle prestazioni dei modelli di regressione di training

Model-Name	R^2	MAE	MSE	RMSE	EVS
Lasso	0.82	2.92	14.74	3.83	0.80
Ridge	0.83	2.58	11.76	3.42	0.82
Elastic Net	0.84	2.50	10.69	3.27	0.83
SGD	0.84	2.50	10.69	3.27	0.83
Linear	0.84	2.50	10.67	3.26	0.83
Polynomial	0.88	6.54	65.02	8.06	0.84
Gradient Boosting	0.94	1.50	4.01	2.00	0.915
AdaBoost	0.94	1.48	3.89	1.97	0.92
Decision Tree	0.96	0.69	2.08	1.44	0.95
HistGradientBoosting	0.96	1.19	2.61	1.61	0.94
KNN	0.98	0.65	1.15	1.07	0.98
Random Forest	0.98	0.63	0.78	0.88	0.98
XGBoost	0.98	0.62	0.76	0.87	0.98
Extratree	0.98	0.65	0.93	0.96	0.98
SVM	0.99	0.57	0.48	0.69	0.98

Table 96: Confronto delle prestazioni dei modelli di regressione di testing

4.7 Conclusioni sul modello migliore

In questo caso modelli con punteggi R2 bassi **(Lasso, Ridge)** mostrano una minore capacità predittiva rispetto ad altri modelli, suggerendo un possibile underfitting.
Inoltre modelli come **Decision Tree, Extratree, Random Forest, HistGradientBoosting, e SVM** con punteggi R2 molto vicini a 1.00 sul training e testing possono indicare overfitting, sebbene abbiano eccellenti prestazioni sui dati di testing.
Come sempre bisogna tenere bene a mente che il miglior modello dovrebbe bilanciare alte prestazioni con il rischio di overfitting, quindi:

- **Elastic Net, SGD, Linear, Ridge:** Offrono buone prestazioni con un equilibrio tra accuratezza e rischio di overfitting. Elastic Net e SGD sembrano leggermente migliori con un R2 di 0.84 sui dati di testing.

- **Polynomial:** Mostra un significativo peggioramento sui dati di testing (R2: 0.88, MSE: 65.02), suggerendo overfitting.

- **Gradient Boosting e AdaBoost:** Hanno alte prestazioni in entrambi i set di dati con R2 molto alto e bassi errori (MAE, MSE, RMSE). Questi modelli sono buoni compromessi.

- **KNN, Random Forest, XGBoost, Extratree, SVM:** Prestazioni molto alte ma con un rischio di overfitting leggermente maggiore rispetto ai modelli di boosting.

Posso concludere che **Gradient Boosting** è il miglior modello, dato che ha alte prestazioni sia nei dati di training (R2: 0.95) che di testing (R2: 0.94), suggerendo un buon equilibrio tra capacità predittiva e rischio di overfitting.
Alternative al Gradient Boosting sono **AdaBoost e SVM**, con eccellenti prestazioni e un rischio di overfitting moderato.

4.8 Deep Learning

Anche in questo caso poichè la maggior parte dei modelli predice troppo bene l'aspettativa di vita, ho deciso di costruire un modello di ANN (Artificial Neural Network). Il modello implementato è una rete neurale sequenziale, una struttura composta da layer disposti in serie, dove ogni layer trasforma l'input ricevuto in un output mediante una funzione di attivazione, passando poi l'output come input al layer successivo.
Primo Layer (Dense): Comprende 64 unità con funzione di attivazione 'ReLU' (Rectified Linear Unit). La ReLU è utilizzata per introdurre non-linearità nel modello, consentendo di catturare relazioni complesse nei dati. Questo layer prende in input direttamente le feature di X train, la cui dimensione determina dinamicamente input shape, garantendo flessibilità e adattabilità del modello a vari tipi di dataset.
Secondo Layer (Dense): Identico al primo per numero di neuroni e tipo di attivazione, questo layer lavora sui dati trasformati dal primo layer per elaborare ulteriormente le relazioni tra le features.
Layer di Output: Consiste in un singolo neurone e, a differenza dei layer precedenti, non utilizza una funzione di attivazione. Questa configurazione è tipica nei problemi di regressione, dove l'obiettivo è predire un valore continuo. La mancanza di funzione di attivazione consente di produrre un range di valori reale non limitato. Vado a compilare il modello, con i seguenti parametri:
Ottimizzatore: 'Adam', un algoritmo basato sulla discesa stocastica del gradiente che si adatta automaticamente al contesto dei dati grazie alla sua capacità di regolare la velocità di apprendimento. Questo lo rende ideale per lavorare con dataset di dimensioni e caratteristiche variabili.
Funzione di Perdita: 'mean squared error', scelta per enfatizzare e penalizzare più severamente gli errori più grandi, una caratteristica desiderabile in molte applicazioni pratiche di regressione.
Metriche: Include 'MAE' (Mean Absolute Error) e 'MSE' (Mean Squared Error) per fornire una valutazione diretta e quantificabile degli errori del modello.

Dopo aver costruito e lanciato il modello, vado a stampare i primi 10 risultati per avere un confronto tra valori previsti ed osservati, e calcolo l'R.Squared.

Table 97: Metriche per il training set (ANN Model)

Metrica	Valore
R^2	0.913
MAE	1.845
MSE	5.678
RMSE	2.383

Table 98: Metriche per il test set (ANN Model)

Metrica	Valore
R^2	0.900
MAE	2.047
MSE	6.889
RMSE	2.625

Table 99: Confronto tra valori osservati e previsti per i primi 10 elementi

Osservato	Previsto
74.29	67.60
75.23	67.80
76.22	71.92
76.91	73.55
74.31	70.90
74.64	71.55
75.20	72.34
75.66	73.47
76.09	75.01
52.69	52.30

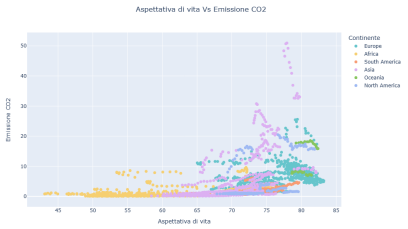

Figure 2: Confronto tra aspettativa di vita e CO2

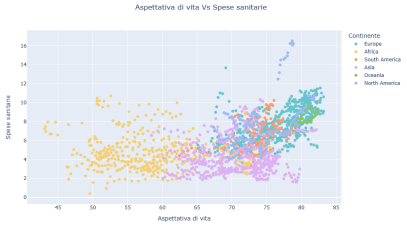

Figure 3: Caption for newplot(14)

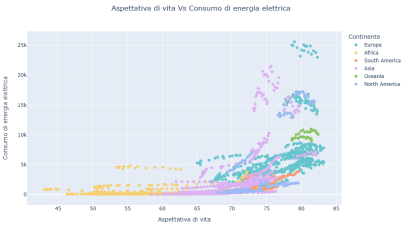

Figure 4: Caption for newplot(15)

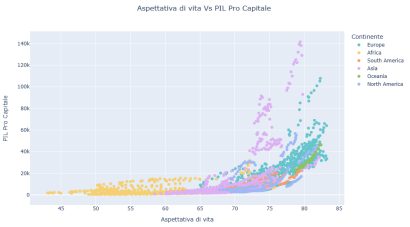

Figure 5: Caption for newplot(16)

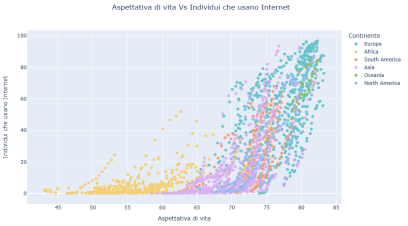

Figure 6: Caption for newplot(17)

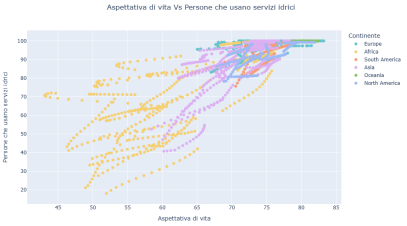

Figure 7: Caption for newplot(18)

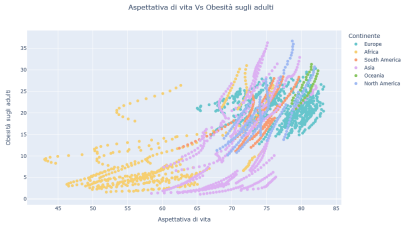

Figure 8: Caption for newplot(19)

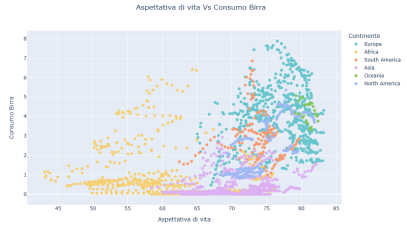

Figure 9: Caption for newplot(20)

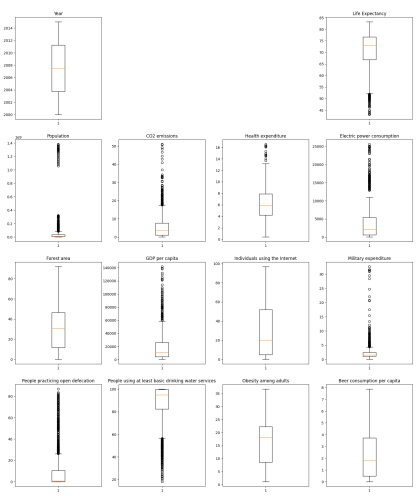

Figure 10: Distribuzione variabili prima del metodo Winsor

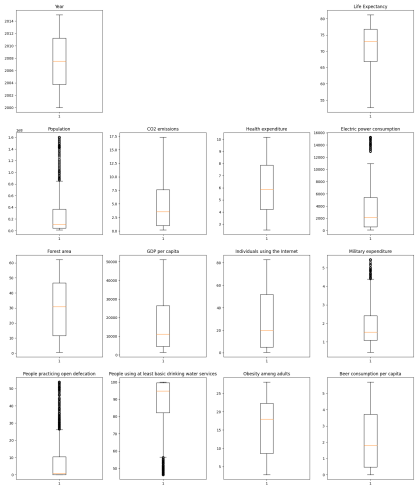

Figure 11: Distribuzione variabili dopo aver eseguito il metodo Winsor

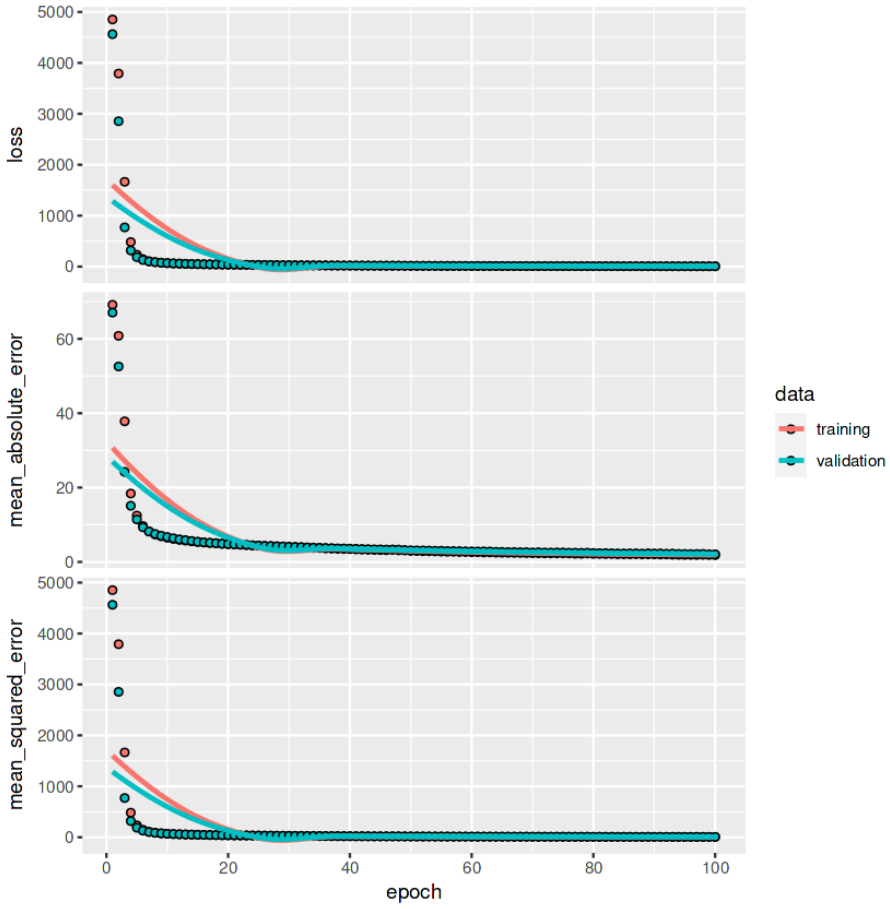

Figure 12: Esempio di Immagine Scaricata

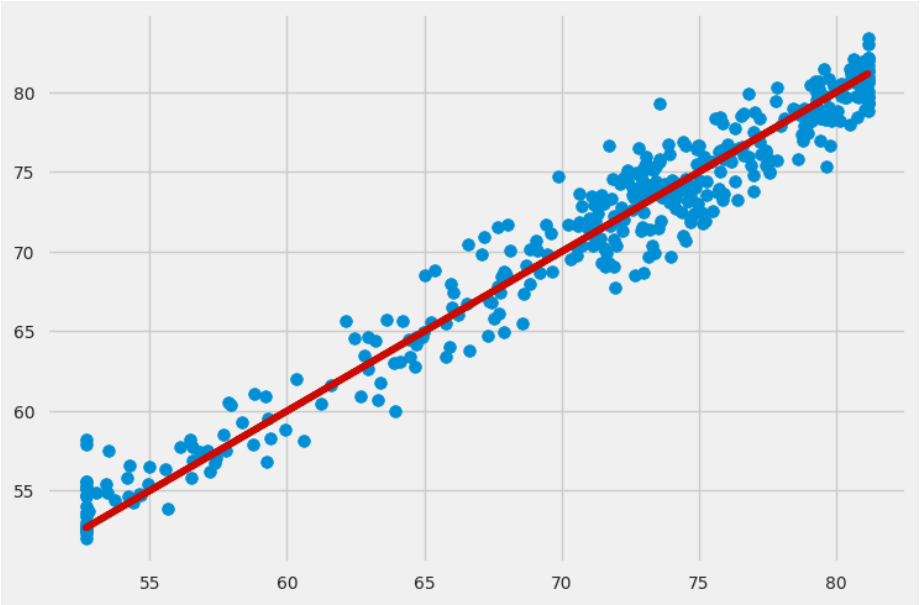

Figure 13: Esempio di Immagine Scaricata

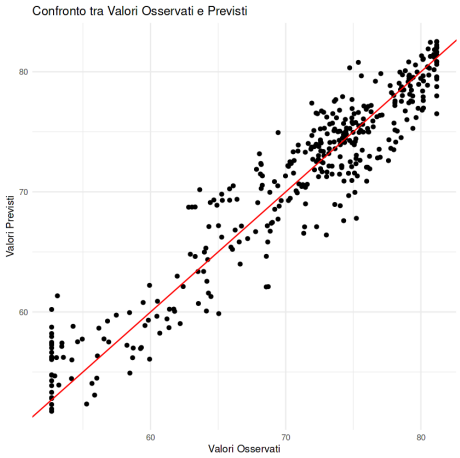

Figure 14: Enter Caption