Intelligenza Artificiale

Università Roma Tre
Dipartimento di Ingegneria

Anno Accademico 2021 - 2022

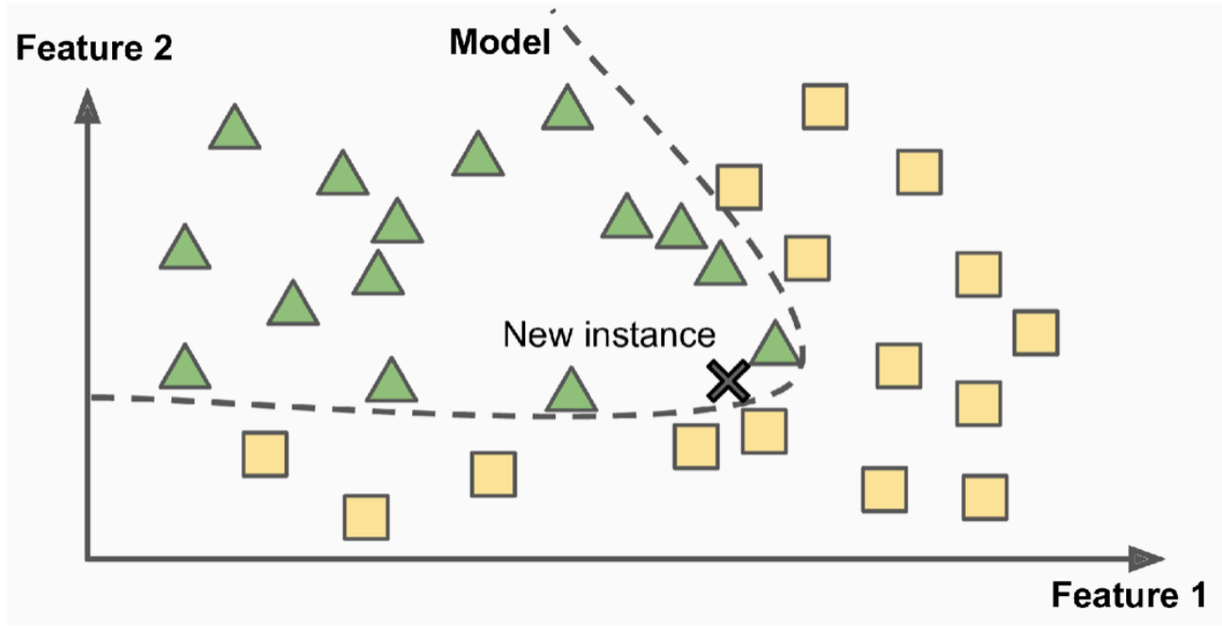
Esercitazione: Regressione (Ex02)

Sommario

- Dataset Better Life Index
- Richiami: Model Selection, Simple Linear Regression, Funzione di Costo
- Libreria Scikit-learn
- Linear Regression in Python
- Esempio: Dataset Diabete
- Linear Regression e Predizione
- Esercitazione su dataset Better Life Index

Richiami

- Le tecniche di *Regressione* ricadono nell'ambito dell'apprendimento *Model-based*, dove
  - Si costruisce un modello che rappresenta le caratteristiche dei dati in ingresso (es. andamento).
  - Tale modello verrà poi impiegato nella fase di *predizione* su istanze in ingresso distinte da quelle impiegate durante l'apprendimento.



Datasets

- Durante le esercitazioni faremo uso di vari datasets, alcuni reali, altri sintetici che ci permetteranno di mettere in evidenza vari aspetti e problematiche rilevanti nell'ambito dell'apprendimento automatico.

Esempio: dataset Better Life Index (1)

Dataset che lega il benessere (life satisfaction) con indicatori giudicati essenziali nella vita quotidiana (es. salario, livello istruzione), suddivisi per nazione.

https://stats.oecd.org/index.aspx?DataSetCode=BLI

			Community	Education			Environment		Civic engagement		Health		Life Satisfaction	
Indicator	ong-term mployment rate	Personal earnings	Quality of support network	Educational attainment	Student skills	Years in education	Air pollution	Water quality	Stakeholder engagement for developing regulations	Voter turnout	Life expectancy	Self-reported health	Life satisfaction	
Unit	ercentage	US Dollar	Percentage	Percentage	Average score	Years	Micrograms per cubic metre	Percentage	Average score	Percentage	Years	Percentage	Average score	
Country														
Colombia	1.1	..	80	59	406	14	22.6	82	1.4	53	76.7	80	5.7	
Czech Republic	0.6	29 885	96	94	495	18	17	89	1.6	62	79.3	62	6.9	
Denmark	0.9	58 430	95	82	501	19	10	93	2	85	81.5	70	7.5	
Estonia	1.2	30 720	95	91	526	18	5.9	86	2.7	64	78.8	57	6.5	
Finland	1.2	46 230	96	91	516	20	5.5	97	2.2	69	82.1	68	7.9	
France	2.9	45 581	94	81	494	17	11.4	78	2.1	75	82.9	67	6.7	
Germany	1.2	53 745	90	86	500	18	12	91	1.8	76	81.4	66	7.3	
Greece	10.8	27 207	78	76	453	19	14.5	67	1.8	58	81.7	79	5.8	
Hungary	1.2	25 409	94	86	479	16	16.7	81	1.2	70	76.4	58	6	
Iceland	0.7	67 488	98	76	481	19	6.4	97	2.1	81	83.2	77	7.6	
Ireland	1.2	49 474	96	85	505	18	7.8	80	1.3	63	82.8	84	7	
Israel	0.2	39 322	95	88	465	16	19.7	77	2.5	67	82.9	74	7.2	
Italy	4.8	37 769	89	63	477	17	15.9	77	2.5	73	83.6	73	6.5	
Japan	0.8	38 515	89	..	520	16	13.7	87	1.4	53	84.4	37	6.1	

Esempio: dataset Better Life Index (2)

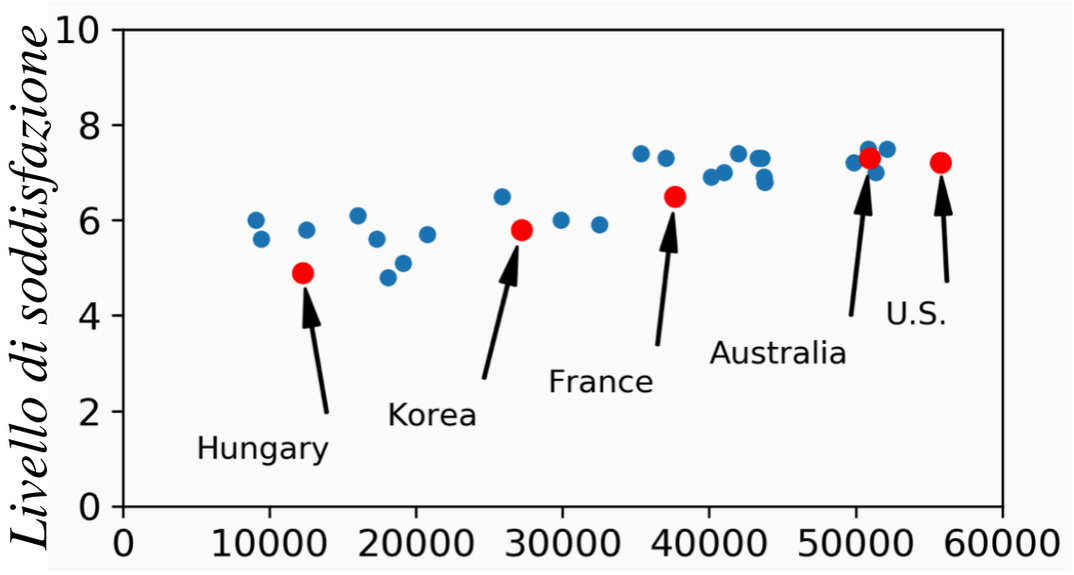
Valore del PIL (GDP) annuale

https://www.imf.org/external/datamapper/NGDP_RPCH@WEO/OEMDC/ADVEC/WEOWORLD

Richiami: Model selection

- Se prendiamo i due dataset e creiamo un join, possiamo mettere in correlazione due variabili, es. PIL pro capite e livello di soddisfazione percepito.

Country	GDP per capita (USD)	Life satisfaction
Hungary	12,240	4.9
Korea	27,195	5.8
France	37,675	6.5
Australia	50,962	7.3
United States	55,805	7.2



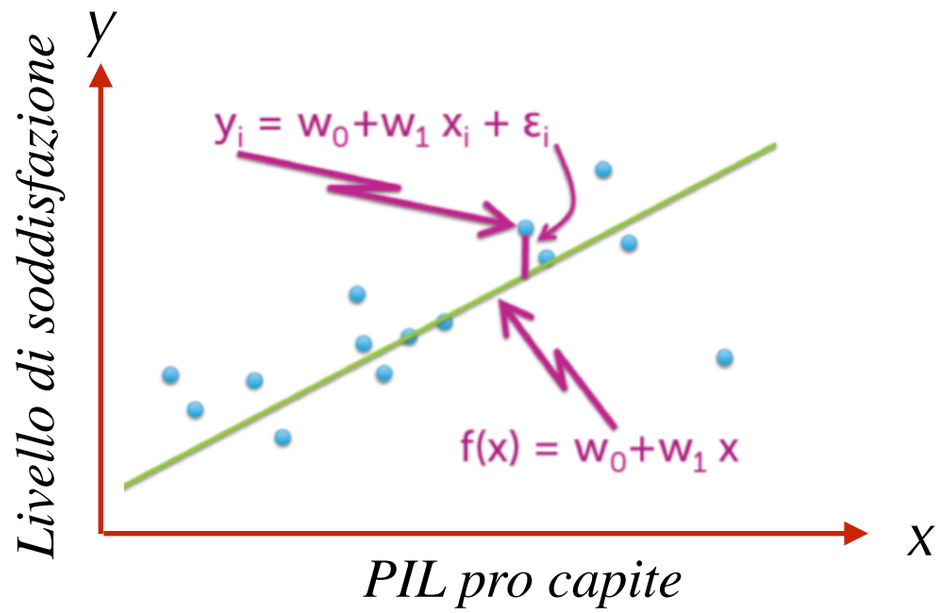
- Si nota come i due valori siano correlati, sebbene non esattamente, con un legame lineare.
- Possiamo supporre che esista un modello *lineare* (o *ordinary least squares*) che leghi la soddisfazione con il valore del PIL (fase di **model selection**).

Richiami: Simple Linear Regression Model

- I parametri del modello lineare sono w_0 e w_1.
  - Attenzione: non esiste un formalismo standard per rappresentare i parametri, a volte si impiega θ o altri simboli.

$$y_i = w_0 + w_1 x_i + \epsilon_i$$

$$\hat{y}_i = f(x_i) = w_0 + w_1 x_i$$



- I parametri del modello lineare sono w_0 e w_1.
  Adattando tali valori possiamo definire qualsiasi modello lineare.

Richiami: funzione di costo

- Tipicamente is impiega una misura di costo basata sulla distanza tra valore esatto e valore determinato dal modello, come la *Residual Sum of Squares* (RSS), sempre positiva, chiamata anche *Sum of Squared Residuals* (SSR) o *Sum of Squared estimate of Errors* (SSE):

$$\mathrm{RSS}(w_0, w_1) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{N} [y_i - (w_0 + w_1 x_i)]^2$$

- Valori prossimi allo *0* indicano un modello ideale.
- La *Mean Square Error* (MSE), chiamata anche *Mean Squared Deviation* (MSD), corrisponde alla RSS normalizzata sul numero di campioni.
  È utile per valutare il modello finale dopo l'addestramento.

Richiami: la libreria Scikit-learn

- È la libreria con licenza aperta (BSD) più conosciuta di machine learning in Python. La prima release risale al 2010.
  - https://scikit-learn.org/stable/
- Include gli algoritmi più popolari di classificazione, regressione, clustering (es. support-vector machines, random forests, gradient boosting, k-means e DBSCAN).
- Alcune parti del codice sono state scritte in modo altamente efficiente con varie tecnologie (vedi Cython)
- È facilmente interfacciabile con altre librerie per la gestione e il calcolo numerico di dati, es. NumPy (algebra lineare), SciPy (ottimizzazione, algebra lineare, analisi dei segnali, etc) e Pandas.

Linear Regression in Python

- Il modulo linear_model di **sklearn** implementa l'addestramento basandosi su un modello lineare. La funzione di costo di default è la **RSS**.
- La funzione fit() prende come parametri due arrays X e y, effettua l'addestramento (o *fitting*) e memorizza i coefficienti nella variabile coef_.
- Ad esempio:

```
>>> from sklearn import linear_model
>>> reg = linear_model.LinearRegression()
>>> reg.fit([[0, 0], [1, 1], [2, 2]], [0, 1, 2])
LinearRegression()
>>> reg.coef_
array([0.5, 0.5])
```

- Nell'esempio si impiegano 2 valori (cioè 2 features) per punto, e la retta ha 2 coefficienti w_1. Il valore di w_0 si ottiene con la variabile intercept_ del modello.
- Nota: l'underscore nel nome delle variabili indica che i valori sono ottenuti durante l'addestramento, e perciò non sono iperparametri del modello.

Scikit-learn e le misure di performance

- Il modulo metrics di **sklearn** implementa varie misure di performance.

https://scikit-learn.org/stable/modules/model_evaluation.html

Regression	
'explained_variance'	`metrics.explained_variance_score`
'max_error'	`metrics.max_error`
'neg_mean_absolute_error'	`metrics.mean_absolute_error`
'neg_mean_squared_error'	`metrics.mean_squared_error`
'neg_root_mean_squared_error'	`metrics.mean_squared_error`
'neg_mean_squared_log_error'	`metrics.mean_squared_log_error`
'neg_median_absolute_error'	`metrics.median_absolute_error`
'r2'	`metrics.r2_score`
'neg_mean_poisson_deviance'	`metrics.mean_poisson_deviance`
'neg_mean_gamma_deviance'	`metrics.mean_gamma_deviance`
'neg_mean_absolute_percentage_error'	`metrics.mean_absolute_percentage_error`

- Nota: troviamo MSE ma non RSS.

Esempio: dataset diabete

- Un dataset diabete è un dataset *toy* (cioè utile per scopi didattici e per testare il codice) disponibile all'interno della libreria scikit-learn.
  - URL: https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html https://scikit-learn.org/stable/datasets/toy_dataset.html
  - 442 istanze
  - 10 features reali normalizzate ove richiesto -.2 < x < .2
    - age age in years
    - sex
    - bmi body mass index
    - bp average blood pressure
    - s1 tc, total serum cholesterol
    - s2 ldl, low-density lipoproteins
    - s3 hdl, high-density lipoproteins
    - s4 tch, total cholesterol / HDL
    - s5 ltg, possibly log of serum triglycerides level
    - s6 glu, blood sugar level
  - Target: intero nell'intervallo 25 - 346 che indica quanto la malattia sia accresciuta dopo 1 anno

Esempio Python: diabete (1)

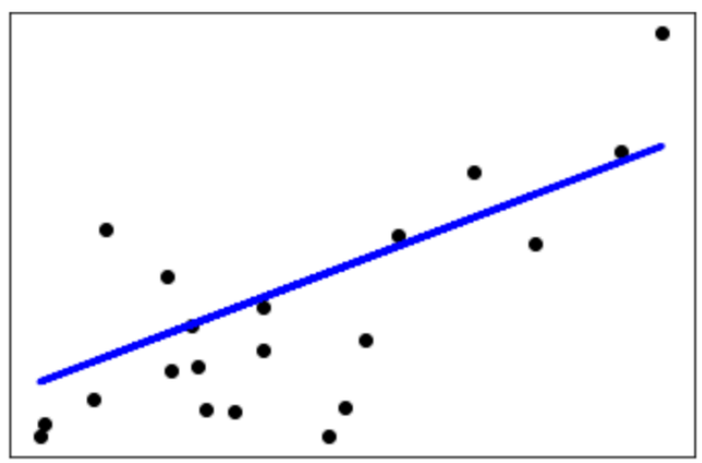
- Codice per impiegare il dataset:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score


# Carico il dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)

# Mantengo solo la terza feature
diabetes_X = diabetes_X[:, np.newaxis, 2]

# Suddivido il dataset in training/test 80/20%
diabetes_X_train,diabetes_X_test,diabetes_y_train, diabetes_y_test =
  train_test_split(diabetes_X, diabetes_y,test_size=0.2)

...
```

- Esercizio: completa il codice impiegando un modello lineare, visualizzando il valore dei coefficienti e l'errore MSE.

Esempio Python: diabete (2)

```
# Istanzia un modello di regressione lineare
regr = linear_model.LinearRegression()

# Addestramento con funzione di costo RMSE
regr.fit(diabetes_X_train, diabetes_y_train)

# Ricava le predizioni sul test set
diabetes_y_pred = regr.predict(diabetes_X_test)

# Stampa i parametri del modello
print("Coefficients: \n", regr.coef_)
# Valuto il MSE
print("Mean squared error: %.2f" % mean_squared_error(diabetes_y_test, diabetes_y_pred))


plt.scatter(diabetes_X_test, diabetes_y_test, color="black")
plt.plot(diabetes_X_test, diabetes_y_pred, color="blue", linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()

# Coefficients:   [938.23786125]
# Mean squared error: 2548.07
```



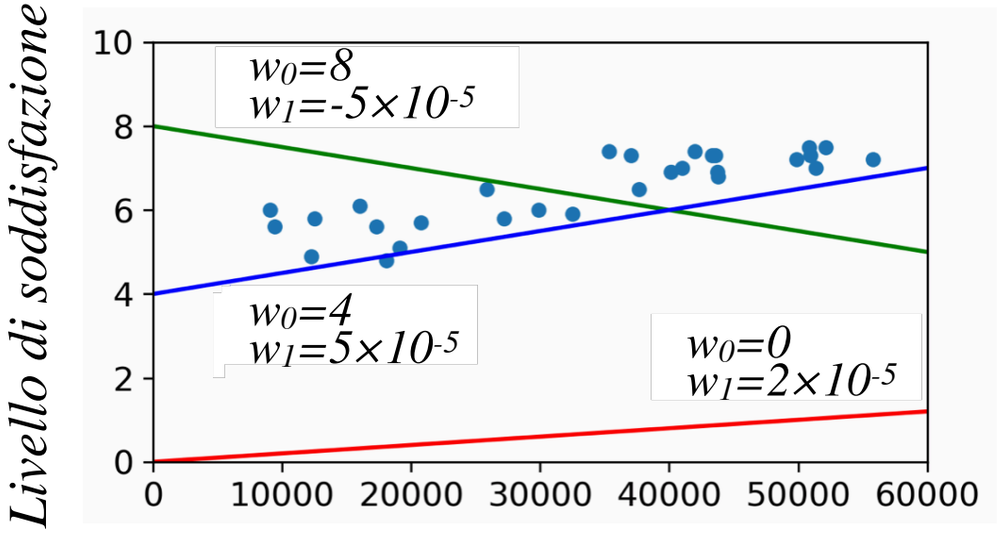
Linear Regression e predizione

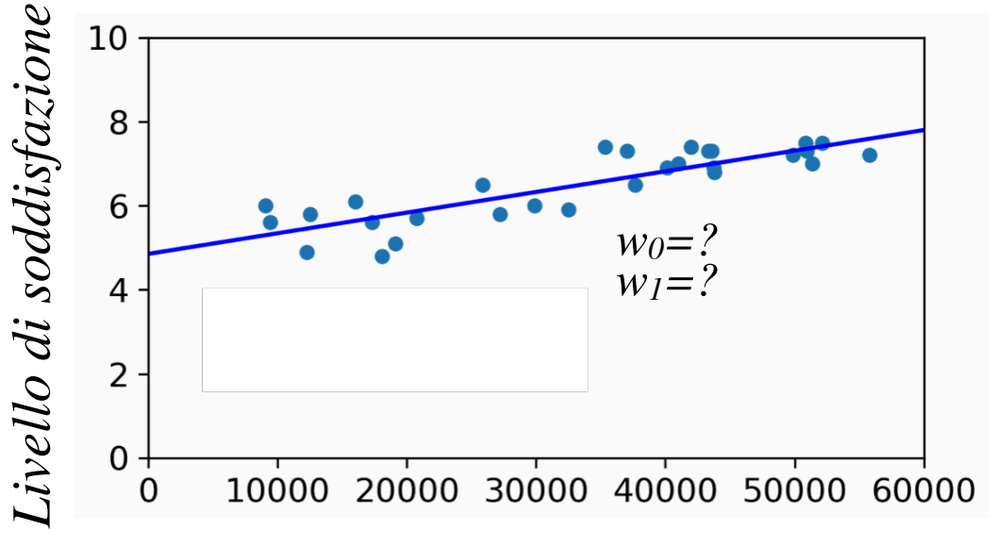
- Il modulo linear_model permette facilmente di fare predizione sui dati.
- La funzione *predict()* prende un array di istanze (una o più features) e ricava il valore in base al modello addestrato.

```
X_new = [[22587]]
print(model.predict(X_new))
```

Esempio: dataset Better Life Index (3)

- Il problema consiste nel determinare i due parametri *w*. Chiaramente il modello può solo approssimare la correlazione tra i valori.
- Se facciamo più ipotesi, cioè creiamo più modelli, ci occorre una misura di performance (o di costo) per confrontarli e scegliere il più adatto.





Esercizio Python: Better Life Index

- Accedi al seguente notebook dove i dati sono già caricati e formattati per gli step successivi:
  - https://colab.research.google.com/drive/1apLqC0KAveOkkCT8JuO5kNQyj1GT5Hz9?usp=sharing
- Risolvi i seguenti esercizi:
  - *Esercizio #1*: crea e addestra un modello lineare con funzione di costo RSS
  - *Esercizio #2*: visualizza i parametri del modello
  - *Esercizio #3*: prendi tre campioni random dal dataset e ricava la predizione in base al modello addestrato
  - *Esercizio #4*: calcola RSS MSE e RMSE valutando i tre campioni
  - *Esercizio #5*: suddividi il dataset in input in train e test con un rapporto 80/20
  - *Esercizio #6*: addestra nuovamente il modello, e ricava RSS MSE e RMSE sui test set
  - *Esercizio #7*: suddividi nuovamente il dataset ma con un rapporto 50/50. Valuta nuovamente le performance del modello e discuti eventuali differenze nei valori ottenuti.

Testi di Riferimento

- Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media 2017
- Andreas C. Müller, Sarah Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media 2016