ITIS-LS "Francesco Giordani" Caserta

prof. Ennio Ranucci

a.s. 2019-2020

Reti neurali e Machine Learning

Sintesi e semplificazione del contenuto dei seguenti video:

TED ED:

Introduzione Alle Reti Neurali 01: Cos'è una Rete Neurale?

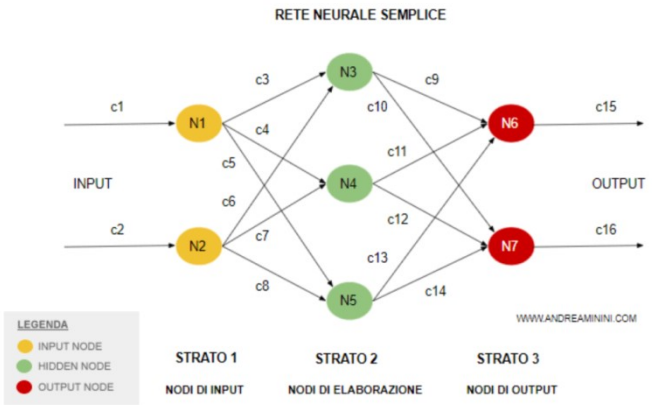
Qual è la differenza tra Intelligenza Artificiale, Machine Learning e Deep Learning?

Links ai due video:

https://ed.ted.com/on/2D1WzCXr

https://ed.ted.com/on/JM2EqPEb

è necessario seguire anche i links riportati nell'approfondimento delle lezioni TED ED



La successione degli archi crea tanti percorsi possibili tra un nodo iniziale e uno finale.

L'**intelligenza artificiale** possiamo definirla come l'abilità di un sistema tecnologico di risolvere problemi o svolgere compiti e attività tipici della mente e dell'abilità umane.
Il **Machine Learning** insegna ai computer e ai robot a fare azioni ed attività in modo naturale come gli esseri umani o gli animali: imparando dall'esperienza (o meglio, attraverso programmi di apprendimento automatico).
Il **Deep Learning**, la cui traduzione letterale significa apprendimento profondo, è una sottocategoria del Machine Learning (che letteralmente viene tradotto come apprendimento automatico) e indica quella branca dell'Intelligenza Artificiale che fa riferimento agli algoritmi ispirati alla struttura e alla funzione del cervello chiamate reti neurali artificiali.

Già oggi ci sono casi d'uso ed ambiti di applicazione che possiamo notare anche come "comuni" cittadini" non esperti di tecnologia.

-Dalla computer vision per le **auto senza conducente**, fino ai **droni e robot** impiegati per la consegna di pacchi o anche per l'assistenza in casi di emergenza (per esempio per la consegna di cibo o sangue per trasfusioni in zone terremotate, alluvionate o in zone che devono affrontare crisi epidemiologiche, ecc.);

-riconoscimento e sintesi vocale e linguistica per **chatbot e robot di servizio**; riconoscimento facciale per **sorveglianza** in paesi come la Cina;

-riconoscimento immagini per aiutare i radiologi a **individuare i tumori nei raggi X**, oppure per aiutare i ricercatori a **individuare le sequenze genetiche correlate alle malattie** e identificare **le molecole che potrebbero portare a farmaci più efficaci o addirittura personalizzati**;

-sistemi di analisi per la **manutenzione predittiva** su una infrastruttura o un impianto analizzando i dati dei sensori dell'IoT;

-la visione del computer che rende possibile il **supermercato Amazon Go senza cassa**.

Guardando invece ai **tipi di applicazione** (intesi come compiti che una macchina può svolgere grazie al Deep Learning), quelli di seguito sono quelli ad oggi più maturi:

1) **colorazione automatica delle immagini in bianco e nero** (per la rete neurale significa riconoscere bordi, sfondi, dettagli e conoscere i colori tipici di una farfalla, per esempio, sapendo esattamente dove collocare il colore corretto);

2) **aggiunta automatica di suoni a filmati silenziosi** (per il sistema di Deep Learning significa sintetizzare suoni e collocarli correttamente all'interno di una situazione particolare riconoscendo immagini ed azioni, per esempio inserendo il suono del martello pneumatico, della rottura dell'asfalto e i sottofondi di una strada cittadina trafficata in un video in cui si vedono dei lavoratori che stanno rompendo l'asfalto con il martello pneumatico);

3) **traduzione simultanea** (per il sistema di Deep Learnign significa ascoltare e riconoscere il linguaggio naturale, riconoscere la lingua parlata e tradurre il significato in un'altra lingua);

4) **classificazione degli oggetti all'interno di una fotografia** (il sistema in questo caso è in grado di riconoscere e classificare tutto ciò che vede in un'immagine, anche molto complessa dove per esempio c'è un paesaggio di sfondo, per esempio delle montagne, persone che camminano lungo un sentieri, degli animali al pascolo, ecc.);

5) **generazione automatica della grafia** (ci sono già oggi sistemi di Deep Learning capaci di utilizzare la grafia umana per scrivere addirittura apprendendo gli stili della scrittura a mano degli esseri umani ed imitandola);

6) **generazione automatica di testo** (sono sistemi che hanno imparato a scrivere correttamente in una determinata lingua rispettando ortografia, punteggiatura, grammatica e persino imparando ad usare stili di scrittura differenti a seconda dell'output da produrre, per esempio un articolo giornalistico o una novella);

7) **generazione automatica di didascalie** (in questo caso il riconoscimelo delle immagini, l'analisi del contesto e la capacità di scrittura consentono ad un sistema di scrivere in automatico le didascalie di una immagine descrivendone perfettamente la scena);

8) **gioco automatico** (abbiamo imparato a capire le potenzialità di un sistema in grado di imparare autonomamente come giocare ad un determinato gioco grazie a DeepMind – oggi parte di Google – che ha sviluppato un sistema di Deep Learning – AlphaGo – che non solo ha imparato a giocare al complessissimo gioco Go ma è riuscito anche a battere il campione mondiale, umano).

I sistemi di Deep Learning hanno compiuto enormi passi evolutivi e sono migliorati moltissimo negli ultimi cinque anni, soprattutto per la grandissima quantità di dati a disposizione ma soprattutto per la disponibilità di infrastrutture ultra performanti (CPU e GPU in particolare).

Nell'ambito della ricerca sull'Artificial Intelligence, l'apprendimento automatico ha riscosso un notevole successo negli ultimi anni, consentendo ai computer di superare o avvicinarsi alle prestazioni umane corrispondenti in aree che vanno dal riconoscimento facciale al riconoscimento vocale e linguistico. L'apprendimento profondo invece consente ai computer di fare un passo in avanti, in particolare di risolvere una serie di problemi complessi.

Nota:

L'*unità di elaborazione grafica*, detta comunemente GPU, dall'acronimo inglese
di ***g****raphics* ***p****rocessing* ***u****nit*, è un particolare circuito elettronico progettato appositamente per la creazione di grafica digitale.

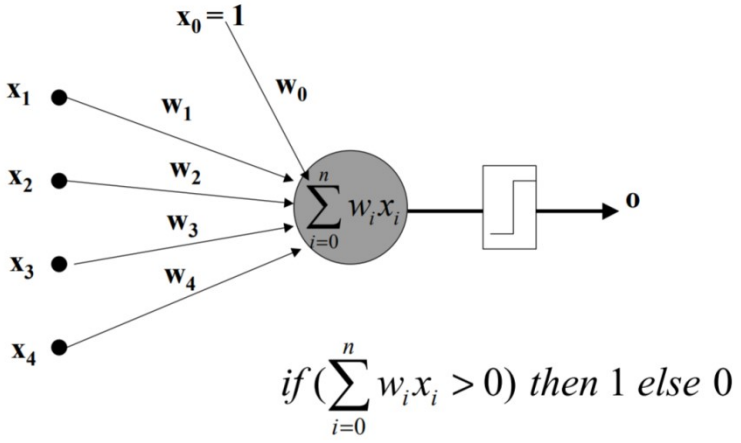
Reti Neurali

- Sono approssimatori di funzioni
- L'idea è ricostruire una funzione tramite la **composizione di unità elementari**, ciascuna delle quali è in grado di eseguire poche e semplici computazioni
- Le unità sono dette **neuroni** (per motivi storici)

Botta - Dip. di Informatica, Univ. Torino

I neuroni diventano funzioni che risolvono un sottoproblema, la rete neurale è costituita da almeno un neurone.

La rete più semplice: 1 solo neurone (Perceptron)



Botta - Dip. di Informatica, Univ. Torino

Il neurone riceve le x(input), i pesi e restituisce 0 oppure 1

Scrivere un algoritmo significa stabilire la relazione f tra ingressi ed uscite

$$y = f(x)$$

Explicit Programming

f è il codice del programmatore. Le istruzioni passo dopo passo trasformano gli input in output: la x in y

Ad esempio f è un algoritmo che calcola il doppio di x

Machine Learning

Alla macchina vengono forniti degli esempi di x e ralative y, l'algoritmo della macchina trova la **f** che traforma la x in y

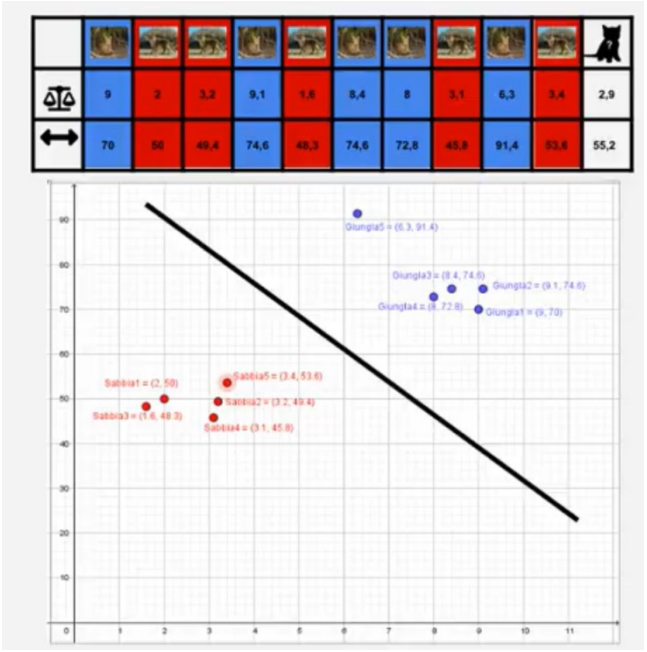
Ad esempio in una tabella sono riportati x e y dove y è il doppio di x. La macchina trova per apprendimento di calcolare il vaolore di y corrispondente ad un certo x

Quindi l'algoritmo si riformula sulla base del dataset (una tabella contenente x e y)

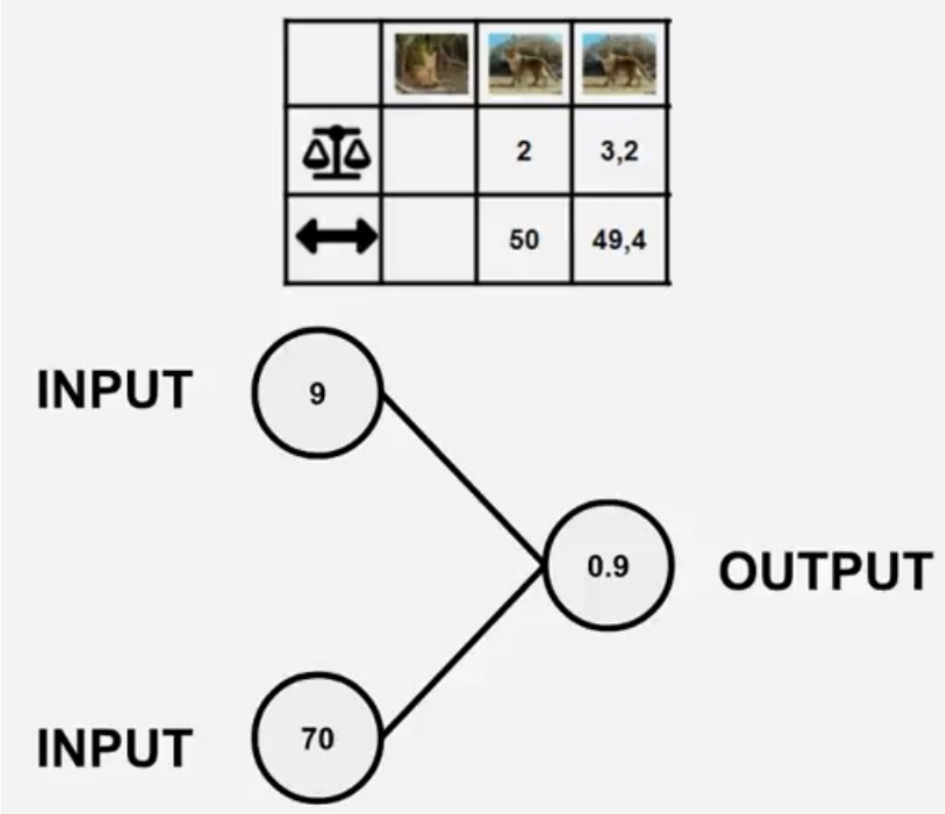
Esempio di dataset in cui sono riportati la massa e la lunghezza dei gatti della giungla e delle sabbie

											
⚖	9	2	3,2	9,1	1,6	8,4	8	3,1	6,3	3,4	
⟷	70	50	49,4	74,6	48,3	74,6	72,8	45,8	91,4	53,6	

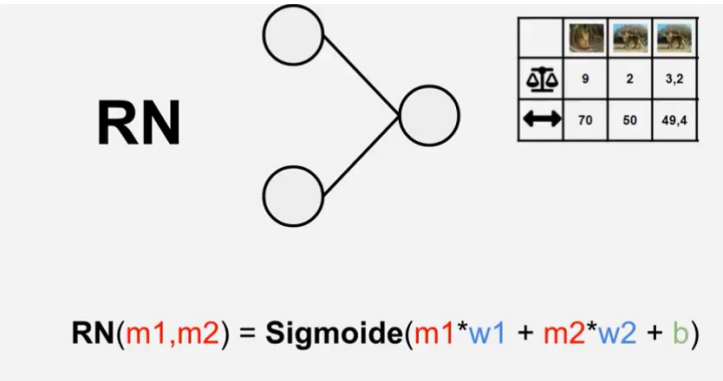
Possiamo riconoscere la specie del gatto dalle sue misure.
Distribuiamo in un grafico le misure:



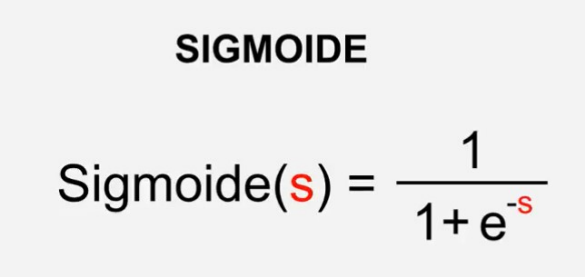
Invece di riconoscere la specie del gatto mediante il grafico, proviamo a farlo mediante una rete neurale (decidiamo che risultato vicino a 1 indica un gatto delle sabbie altrimenti se si avvicina a zero è un gatto della giungla):



Il risultato è stato ottenuto con la seguente formula:



RN=Rete Neurale
w1 e **w2** = pesi
m1=massa del gatto **m2**=lunghezza del gatto
Sigmoide= funzione che fa in modo che il risultato sia sempre compreso tra 0 e 1
b=BIAS



Avendo utilizzato valori casuali per w1, w2 e bias il risultato non ci permette di riconoscere il gatto.
Per addestrare la rete affinché restituisca risultati corretti si utilizzano due momenti forward propagation e backpropagation in cui una funzione matematica calcola i valori w1, w2 e b sulla base del dataset.

Codice Python (possiamo verificare il codice on line: https://repl.it/languages/python3)

Il seguente codice non consente il riconoscimento corretto della specie del gatto:

```
import math as mt
import random as rd

rd.seed(1)
def RN(m1,m2):
   t=m1*w1+m2*w2+b
   return sigmoide(t)
def sigmoide(t):
   return 1/(1+mt.exp(-t))
w1 = rd.random()
w2 = rd.random()
b = rd.random()
print(RN(9,7.0))
```

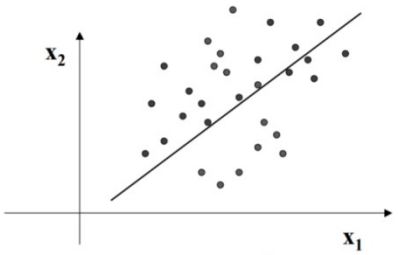
Codice Python

Il seguente codice consente il riconoscimento corretto della specie del gatto:

```
import math as mt
import random as rd

rd.seed(1)

def RN(m1,m2):
   t=m1*w1+m2*w2+b
   return sigmoide(t)

def sigmoide(t):
   return 1/(1+mt.exp(-t))

#dataset
dataset=[[9,7.0,0],
[2,5.0,1],
[3.2,4.94,1],
[9.1,7.46,0],
[1.6,4.83,1],
[8.4,7.46,0],
[8,7.28,0],
[3.1,4.58,1],
[6.3,9.14,0],
[3.4,5.36,1]]

#definisco la derivata della funzione sigmoide
def sigmoide_p(t):
   return sigmoide(t)*(1 - sigmoide(t))

def train():

   #pesi inizializzati inizialmente in modo casuale
   w1 = rd.random()
   w2 = rd.random()
   b = rd.random()

   iterazioni = 10000  #numero di iterazioni 10000
   learning_rate = 0.1 #imposto il learning rate 0.1

   for i in range(iterazioni):

      ri = rd.randint(0,len(dataset)-1) # genero un indice casuale
      point = dataset[ri] # prendo un gatto casuale dal dataset
```

```
        z = point[0] * w1 + point[1] * w2 + b
        pred = sigmoide(z) # previsione della rete

        target = point[2] #il mio valore obiettivo

        # costo del punto casuale attuale
        cost = (pred - target)**2

        #CALCOLO DELLE DERIVATE PARZIALI
        dcost_dpred = 2 * (pred - target) #derivata parziale del costo rispetto alla previsione
        dpred_dz = sigmoide_p(z) #derivata parziale della previsione rispetto a z

        dz_dw1 = point[0] #derivata parziale di z rispetto a w1
        dz_dw2 = point[1] #derivata parziale di z rispetto a w2
        dz_db = 1       #derivata parziale di z rispetto a b

        dcost_dz = dcost_dpred * dpred_dz #derivata parziale di z rispetto alla previsione (uso la
regola della catena)

        #REGOLA DELLA CATENA
        dcost_dw1 = dcost_dz * dz_dw1 #derivata parziale del costo rispetto a w1
        dcost_dw2 = dcost_dz * dz_dw2 #derivata parziale del costo rispetto a w2
        dcost_db = dcost_dz * dz_db #derivata parziale del costo rispetto a b

        #aggiornamento dei pesi e del bias
        w1 = w1 - learning_rate * dcost_dw1
        w2 = w2 - learning_rate * dcost_dw2
        b = b - learning_rate * dcost_db

    return w1, w2, b


#carichiamo i pesi e il bias
w1, w2, b = train()

pred=[] #array vuoto che conterrà le previsioni

for gatto in dataset:  #per ogni gatto nel dataset
    z = w1 * gatto[0] + w2 * gatto[1] + b
    prediction=sigmoide(z)    #previsione della rete
    if prediction <= 0.5: #se la previsione è minore o uguale a 0.5
        pred.append('giungla') #aggiungi la stringa "giungla" all'array pred
    else:
        pred.append('sabbie') #altrimenti aggiungi la stringa "sabbie" all'arrat pred

print(pred) #stampa a schermo l'array pred
```

Che cosa non può descrivere



Si può comunque cercare di posizionare la retta in modo da minimizzare l'errore.

Botta - Dip. di Informatica, Univ. Torino

Esempi di Machine Learning:

Link per verificarli on line: https://repl.it/languages/python3

Esempio 1 (viene importato il dataset "load_boston"):

from sklearn.datasets import load_boston

dataset = load_boston()

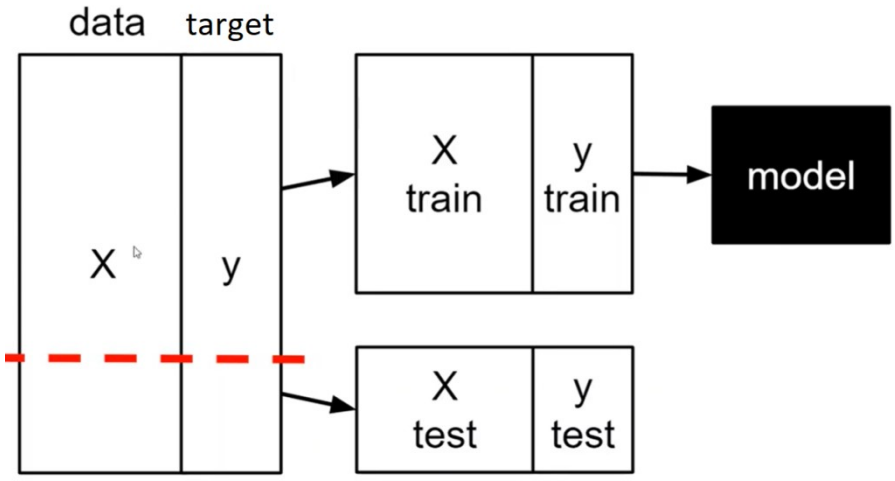
print (dataset ['DESCR'])

Descr restituisce la descrizione del dataset:

Instances: 506

Number of Attributes: 13 numeric/categorical predictive.

ecc.

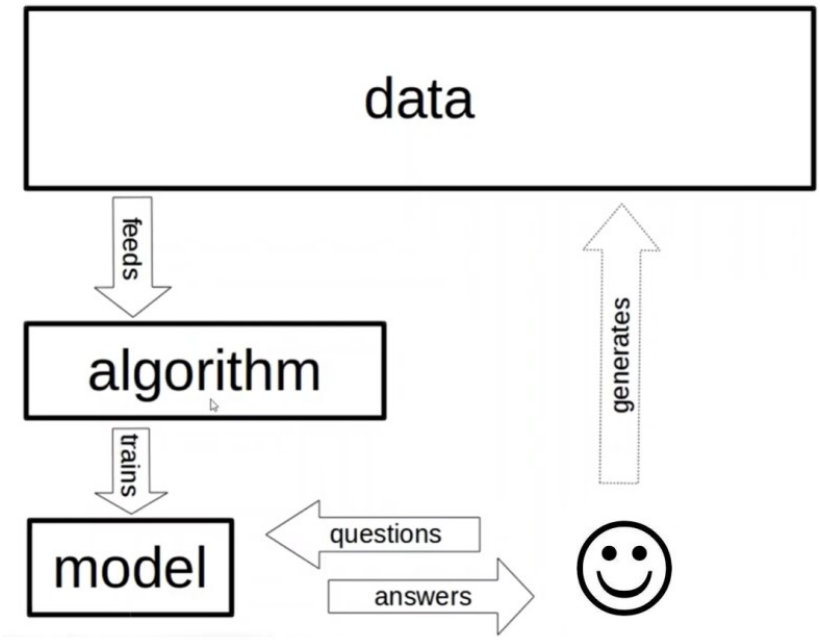


X è una matrice di records formata da varie colonne che contengono le varie informazioni dell'oggetto (misure dell'oggetto, dove si trova, ecc.)

Y è un vettore formato dai prezzi dei vari oggetti.

from sklearn.datasets import load_boston
dataset = load_boston()
print (dataset ['data'])
data restituisce i records di X
from sklearn.datasets import load_boston
dataset = load_boston()
print (dataset ['data'][0])
Restituisce il primo record di X
from sklearn.datasets import load_boston
dataset = load_boston()
print (dataset ['target'][0])
Restituisce il primo valore di y

from sklearn.datasets import load_boston
dataset = load_boston()
print (dataset ['target'])
Restituisce i valori del vettore y



Il dataset può essere costruito a partire dagli OPEN DATA

L'algoritmo di addestramento sulla base dei dati contenuti nel dataset (fotografia del mondo reale) produce il modello. Il modello è la **f** cioè l'algoritmo che produce le risposte.

Importiamo un algoritmo di addestramento detto “regressore”: la regressione lineare che restituisce un numero (una stima).
Gli algoritmi di addestramento “classificatori” restituiscono, invece, una categoria (maschio-femmina, compra-non compra…)

```
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression
dataset = load_boston()
X = dataset['data']
y = dataset['target']
model = LinearRegression()
model.fit(X,y)
```

Model è l’oggetto(istanza) creata dal costruttore LinearRegression()
model.fit(X,y) il modello viene addestrato e quindi deve ricevere sia la X che la y

```
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
import numpy as np
dataset = load_boston()
X = dataset['data']
y = dataset['target']
model = LinearRegression()
model.fit(X,y)
p = model.predict(X)
mae = mean_absolute_error(y,p)
print('MAE', mae)
```

predict fornisce la predizione, la risposta
print(‘Prezzo stimato’,p) fornisce per ogni oggetto il prezzo stimato
mae misura l’errore tra le y(le risposte reali) e le le predizioni(risposte stimate).

In statistica , l' **errore assoluto medio** (**MAE**) è una misura di errori tra osservazioni accoppiate che esprimono lo stesso fenomeno. Esempi di *Y* rispetto a *X* includono confronti tra previsto rispetto a osservato, tempo successivo rispetto a tempo iniziale e una tecnica di misurazione rispetto a una tecnica di misurazione alternativa. MAE è calcolato come:

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}.$$

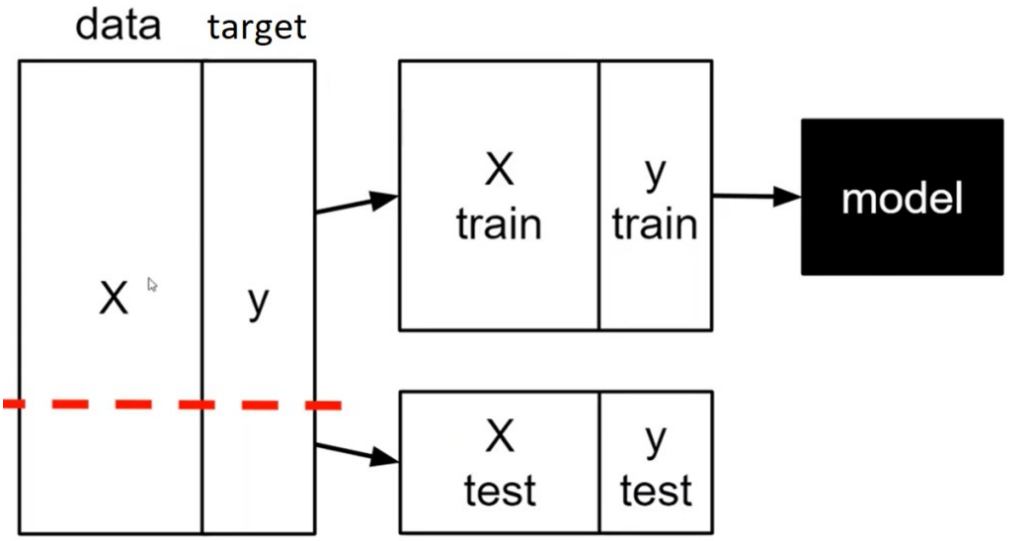
È quindi una media aritmetica degli errori assoluti $|e_i| = |y_i - x_i|$, dove y_i è la previsione e x_i il vero valore.

Per misurare l’errore è stato importato mean_absolute_error

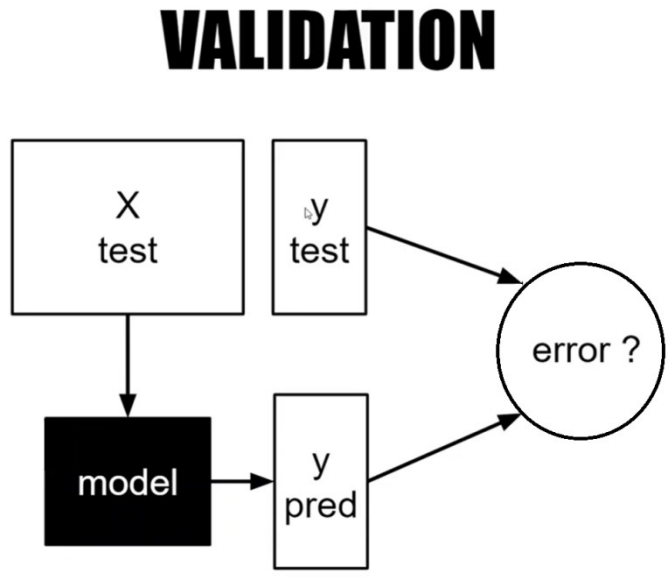
```
print('mean y',np.mean(y))
```

con questa istruzione otteniamo il prezzo medio del target.

Validazione



Per validare il risultato utilizzo il 30% circa del dataset
Mentre per l'addestramento utilizzo il 70% circa del dataset

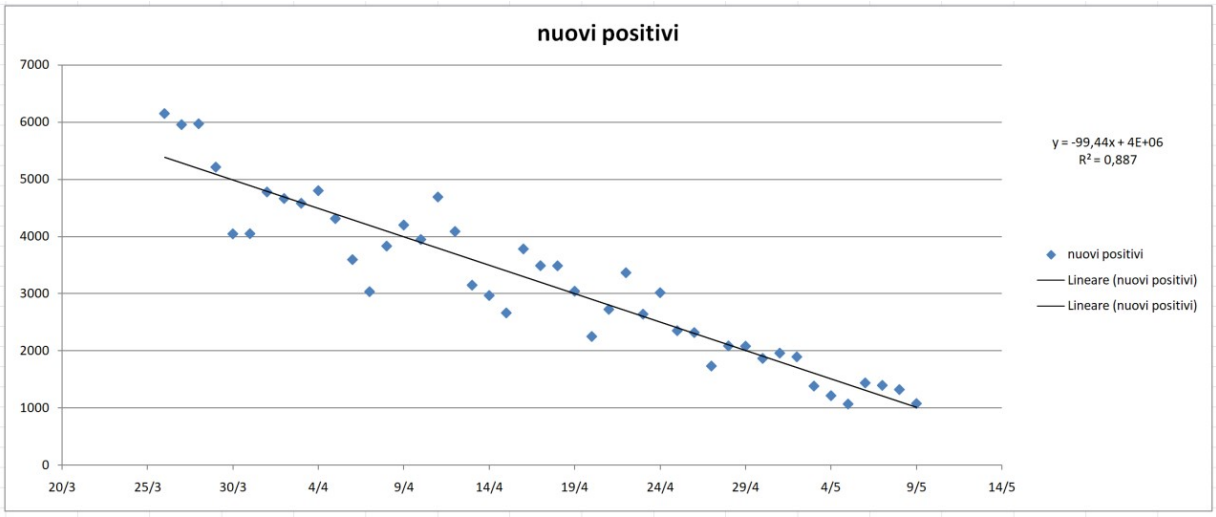


Per suddividere i dati del data set importiamo:
from sklearn.model_selection import train_test_split

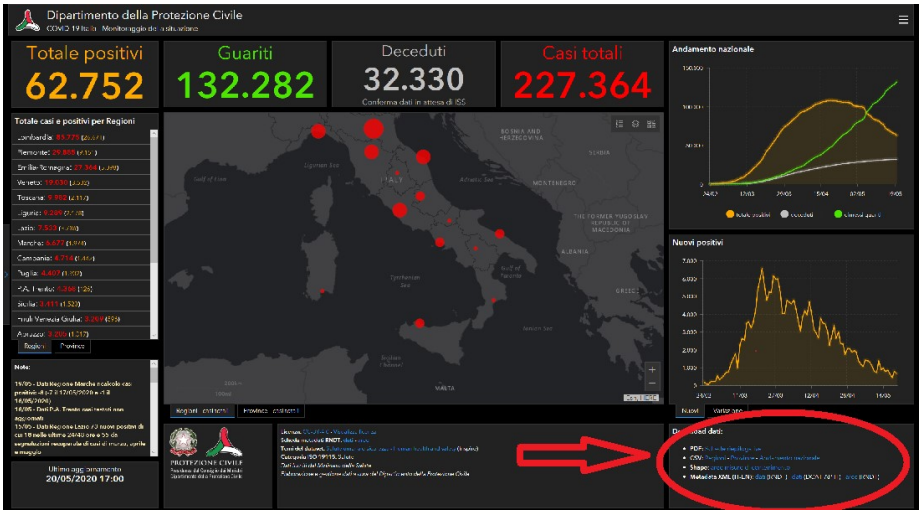
Limitiamo l'addestramento su X_train

```
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection  import train_test_split
import numpy as np
dataset = load_boston()
X = dataset['data']
y = dataset['target']
X_train, X_test, y_train, y_test= train_test_split(X,y)
model = LinearRegression()
model.fit(X_train,y_train)
p_train = model.predict(X_train)
p_test = model.predict(X_test)
mae_train = mean_absolute_error(y_train,p_train)
mae_test = mean_absolute_error(y_test,p_test)
print('MAE train', mae_train)
print('MAE test', mae_test)
```

Previsione nuovi Positivi al tempo del coronavirus
Foglio Excel e Open data Protezione Civile Italiana

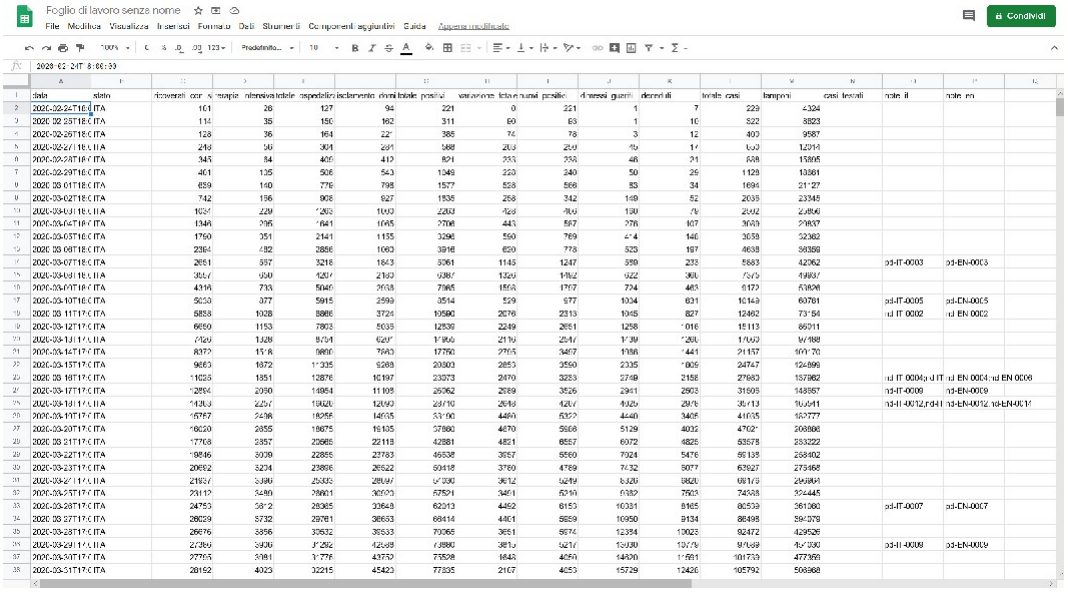


Linea di tendenza costruita con il foglio di lavoro elettronico che riporta il numero di Nuovi Positivi dal 26 marzo 2020 (Picco) al 9 maggio 2020 desunti dagli OPEN DATA della Protezione Civile.
Collegarsi alla pagina:
http://opendatadpc.maps.arcgis.com/apps/opsdashboard/index.html#/b0c68bce2cce478eaac82fe38d4138b1



Click su csv: andamento nazionale (nel riquadro indicato con la freccia e l'ovale rosso nella figura sopra riportata.
Selezionare l'ultimo file di aggiornamento (in fondo all'elenco) e poi RAW e copiare l'indirizzo nella barra dell'indirizzo.
Aprire GSuite ->fogli di Google->aprire un nuovo foglio di lavoro cliccando su +
Focus su A1 scrivere =importdata("https://raw.githubusercontent.com/pcm-dpc/COVID-19/master/dati-andamento-nazionale/dpc-covid19-ita-andamento-nazionale.csv")

L’indirizzo all’interno di importdata è quello copiato precedentemente nel sito della protezione civile.
Viene caricato il dataset:



Copiare le colonne “data e nuovi positivi” in un foglio EXCEL
Adesso dobbiamo estrarre la data dal formato dataOra:

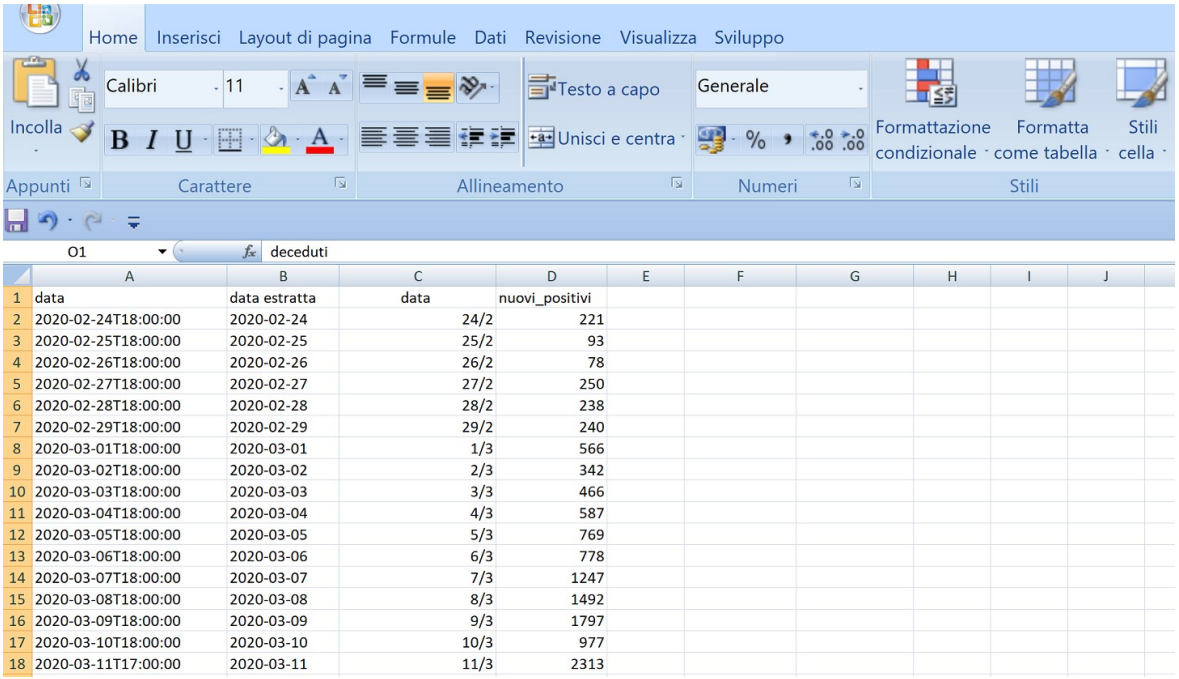
inderiamo una colonna tra “data” e “nuovi Positivi” e nella posizione b2 inseriamo la formula: =STRINGA.ESTRAI(A2;1;10) che preleva solo i primi 10 caratteri e quindi escude l’orario.

tirare il vertice dxbasso di A2 e copiamo la formula fino all’ultimo dato.

A questo punto copiamo la nuova colonna della data in blocco note e sostituiamo con “Sostituisci tutto” i trattini “-“ con la barra “/”.

Copiamo tutto e incolliamo in una colonna vuota del foglio excel inserita tra la data calcolata con la formula di estrazione e i “Nuovi Positivi” .

Cambiamo il formato della data scegliendo mese,giorno, otteniamo:



	A	B	C	D	E	F	G	H	I	J
1	data	data estratta	data	nuovi_positivi						
2	2020-02-24T18:00:00	2020-02-24	24/2	221						
3	2020-02-25T18:00:00	2020-02-25	25/2	93						
4	2020-02-26T18:00:00	2020-02-26	26/2	78						
5	2020-02-27T18:00:00	2020-02-27	27/2	250						
6	2020-02-28T18:00:00	2020-02-28	28/2	238						
7	2020-02-29T18:00:00	2020-02-29	29/2	240						
8	2020-03-01T18:00:00	2020-03-01	1/3	566						
9	2020-03-02T18:00:00	2020-03-02	2/3	342						
10	2020-03-03T18:00:00	2020-03-03	3/3	466						
11	2020-03-04T18:00:00	2020-03-04	4/3	587						
12	2020-03-05T18:00:00	2020-03-05	5/3	769						
13	2020-03-06T18:00:00	2020-03-06	6/3	778						
14	2020-03-07T18:00:00	2020-03-07	7/3	1247						
15	2020-03-08T18:00:00	2020-03-08	8/3	1492						
16	2020-03-09T18:00:00	2020-03-09	9/3	1797						
17	2020-03-10T18:00:00	2020-03-10	10/3	977						
18	2020-03-11T17:00:00	2020-03-11	11/3	2313						

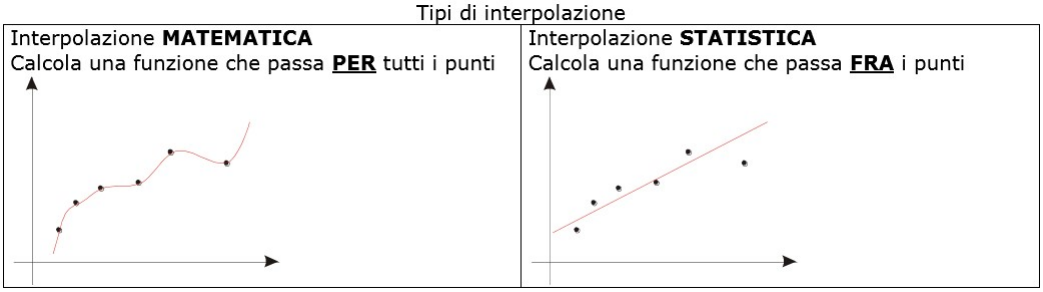
Selezionare la colonna “data” e la colonna “Nuovi positivi”->Inserisci grafico-> a dispersione(solo punti senza linea)->Layout->linea di tendenza->linea di tendenza lineare->opzioni linea di tendenza->spuntare le caselle: visualizza l’equazione del grafico e visualizza il valore R al quadrato del grafico.

***R quadrato**, chiamato coefficiente di determinazione, è un numero compreso tra zero e uno che determina quanto corrisponde la trendline di un grafico ai punti del grafico. R quadrato vicino ad uno rappresenta una linea di tendenza che molto vicina ai punti, il che significa che è possibile utilizzare la linea di tendenza per prevedere con precisione i valori aggiuntivi.*
In altre parole, il valore di R-quadro è il quadrato del coefficiente di correlazione. Il coefficiente di correlazione, R, ci dà una misura della adeguatezza della relazione lineare tra i valori di x e i valori di y. Un valore di R=1 indica una esatta relazione lineare tra x e y. Valori di R vicini a 1 indicano un’eccellente accordo tra i dati e la relazione scelta. Se il coefficiente di correlazione è relativamente lontano da 1 le previsioni basate sulla relazione lineare, y = mx + b, saranno poco attendibili.

***La funzione PREVISIONE** del foglio di lavoro elettronico*
Utile per calcolare o prevedere un valore futuro usando i valori esistenti. Il valore futuro è un valore y per un valore x specifico. I valori esistenti sono valori x e y noti e il valore futuro viene previsto usando la regressione lineare.
PREVISIONE.LINEARE(x; y_nota; x_nota)
x è il valore attuale di cui si desidera prevedere il valore futuro.
y_nota è la matrice o intervallo di dati dipendente.
x_nota è la matrice o intervallo di dati indipendente.

*Per **interpolazione** si intende la ricerca di una funzione matematica che approssima l'andamento di un insieme di punti.*
***Interpolazione Matematica**: tocca tutti i punti della distribuzione. Sebbene più precisa rispetto alla interpolazione statistica, potrebbe risultare complessa se non impossibile da calcolare; inoltre la funzione calcolata potrebbe risultare molto complessa e di difficile utilizzo.*
Applicazioni: l'interpolazione matematica viene utilizzata, in geometria analitica, per il calcolo di una retta passante per due punti (Interpolazione lineare) o di una parabola passante per tre punti non allineati (interpolazione parabolica o di 2° grado).
***Interpolazione Statistica**: consiste nel trovare una funzione che passa fra un insieme di punti in modo che "sia il più possibile vicina" ai vari punti. Sebbene possa sembrare imprecisa ed approssimativa rispetto a quella matematica permette di utilizzare funzioni piuttosto semplici (es. retta) e soprattutto facili da interpretare. L'interpolazione. Statistica è molto utile nei seguenti campi: Serie storiche; problemi di Perequazione, Estrapolazione, Studio del trend.*

Tipi di interpolazione



Metodo dei minimi quadrati
Questo metodo consiste nel determinare i parametri della funzione interpolante prescelta in modo che sia minima la somma dei quadrati degli scostamenti dei punti dalla funzione.
E' una tecnica di ottimizzazione (o regressione) che permette di trovare una funzione, rappresentata da una curva ottima (o curva di regressione), che si avvicini il più possibile ad un insieme di dati (tipicamente punti del piano). In particolare, la funzione trovata deve essere quella che minimizza la somma dei quadrati delle distanze tra i dati osservati e quelli della curva che rappresenta la funzione stessa. Possiamo distinguere parabola dei minimi quadrati e retta dei minimi quadrati.

https://repl.it/languages/python3
Nella prima colonna a dx del sito repl cliccare sui tre puntini per caricare il file: positivi24Feb.csv che contiene la data in formato numero e i nuovi positivi in quella data. (dal 24 feb al 9 mag)

Il codice che segue stampa il contenuto del file csv

```
import csv
with open('positivi24Feb.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            print(f' {row}')
            line_count += 1
        else:
            print(f'\t{row[0]}--{row[1]}')
            line_count += 1
    print(f'Processed {line_count} lines.')
```

Il codice che segue stampa il contenuto del secondo record di X

```
import pandas as pd
dataset = pd.read_csv('positivi24Feb.csv')
X = dataset['data']
y = dataset['nuovi_positivi']
print (dataset ['data'][1])
```

Il codice che segue stampa la previsione del 20 maggio (43960 numero corrispondente al 9 maggio+11)

```
import pandas as pd
from sklearn.linear_model import LinearRegression
dataset = pd.read_csv('positivi24Feb.csv')
X = dataset['data'].values.reshape(-1,1)
y = dataset['nuovi_positivi'].values.reshape(-1,1)
model = LinearRegression()
model.fit(X,y)
Xnew = [[43971]]
ynew = model.predict(Xnew)
print('previsione positivi del 20 maggio ',ynew)
```

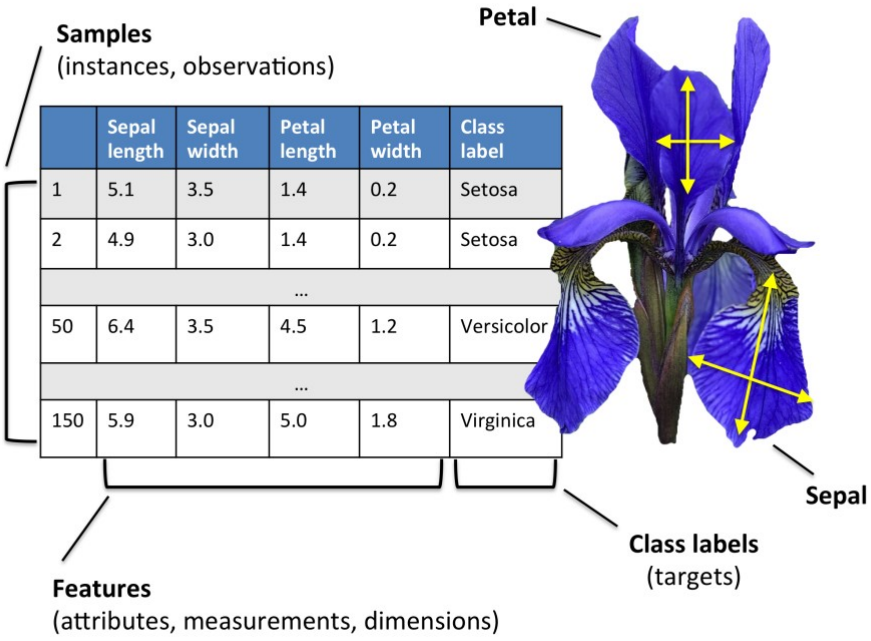
Il codice che segue stampa la previsione del 20 maggio (43960 numero corrispondente al 9 maggio+11) e il coefficiente di determinazione R quadrato

```
import pandas as pd
from sklearn.linear_model import LinearRegression
dataset = pd.read_csv('positivi24Feb.csv')
X = dataset['data'].values.reshape(-1,1)
y = dataset['nuovi_positivi'].values.reshape(-1,1)
model = LinearRegression()
model.fit(X,y)
Xnew = [[43971]]
ynew = model.predict(Xnew)
print('previsione positivi del 20 maggio ',ynew)
print(model.score(X, y))
```

Caricare il file: positivi26Mar.csv (dati a partire dal 26 marzo, PICCO)
Il codice che segue stampa la previsione del 20 maggio (43960 numero corrispondente al 9 maggio+11) e il coefficiente di determinazione R quadrato

```
import pandas as pd
from sklearn.linear_model import LinearRegression
dataset = pd.read_csv('positivi26Mar.csv')
X = dataset['data'].values.reshape(-1,1)
y = dataset['nuovi_positivi'].values.reshape(-1,1)
model = LinearRegression()
model.fit(X,y)
Xnew = [[43971]]
ynew = model.predict(Xnew)
print('previsione positivi del 20 maggio ',ynew)
print(model.score(X, y))
```

CLASSIFICATORE



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

```
from sklearn.datasets import load_iris
dataset = load_iris()
X= dataset ['data']
y= dataset ['target']
print (X[0],y[0])
```

```
from sklearn.datasets import load_iris
from sklearn.model_selection  import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
dataset = load_iris()
X= dataset ['data']
y= dataset ['target']
X_train, X_test, y_train, y_test= train_test_split(X,y)
model= DecisionTreeClassifier()
model.fit(X_train,y_train)
p_train = model.predict(X_train)
p_test = model.predict(X_test)
acc_train =accuracy_score(y_train, p_train)
acc_test =accuracy_score(y_test, p_test)
print(f'Train{acc_train},test{acc_test}')
```

RICONOSCIMENTO GATTI

											
⚖	9	2	3,2	9,1	1,6	8,4	8	3,1	6,3	3,4	
⟷	70	50	49,4	74,6	48,3	74,6	72,8	45,8	91,4	53,6	

```
from sklearn.tree import DecisionTreeClassifier
data=[[9,7.0],
[2,5.0],
[3.2,4.94],
[9.1,7.46],
[1.6,4.83],
[8.4,7.46],
[8,7.28],
[3.1,4.58],
[6.3,9.14],
[3.4,5.36]]
target=[[0],
[1],
[1],
[0],
[1],
[0],
[0],
[1],
[0],
[1]]

X= data
y= target
model= DecisionTreeClassifier()
model.fit(X,y)
p= model.predict(X)
print(p)
p1=model.predict([[3.2, 4.94]])
print (p1)
```

Split di dataAddestramento(70%) e dataRiconoscimento(30%) : l'addestramento viene fatto sui primi 7 elementi del dataset.
Si utilizza l'addestramento sui primi 7 elementi per riconoscere i 3 gatti le cui misure sono state escluse dall'addestramento.

```
from sklearn.tree import DecisionTreeClassifier
dataAddestramento=[[9,7.0],
[2,5.0],
[3.2,4.94],
[9.1,7.46],
[1.6,4.83],
[8.4,7.46],
[8,7.28]]

data=[[3.1,4.58],
[6.3,9.14],
[3.4,5.36]]

targetAddestramento=[[0],
[1],
[1],
[0],
[1],
[0],
[0]]
target=[[1],
[0],
[1]]

X= dataAddestramento
y= targetAddestramento
model= DecisionTreeClassifier()
model.fit(X,y)
p= model.predict(X)
print(p)
p1=model.predict(data)
print (p1)
```

I modelli addestrati da altri possono essere messi a disposizione per tutti.

https://www.youtube.com/watch?time_continue=305&v=EkStkZwwJxM&feature=emb_logo

RICONOSCIMENTO VOCALE

In ambiente repl.it (non funziona per il microfono)
Selezionare Packages (colonna a sx)
importare SpeechRecognition inserendo nella casella di ricerca il seguente url:
https://python.org/pypi/SpeechRecognition
cliccare su + per aggiungere il pacchetto

importare anche pyaudio

```
import speech_recognition as sr

recognizer_instance = sr.Recognizer()

with sr.Microphone() as source:

 recognizer_instance.adjust_for_ambient_noise(source)

 print("Sono in ascolto del tuo messaggio: ")

 audio = recognizer_instance.listen(source)

 print("Sto preparando la scrittura del messaggio: ")

try:

   text = recognizer_instance.recognize_google(audio, language="it-IT")

   print("Google: ", text)

except Exception as e:

 print (e)
```

RICONOSCIMENTO VOCALE in ambinte Python locale (funziona)

Creiamo la cartella progettiPython nella cartella
C:\Users\ennio\AppData\Local\Programs\Python\Python37

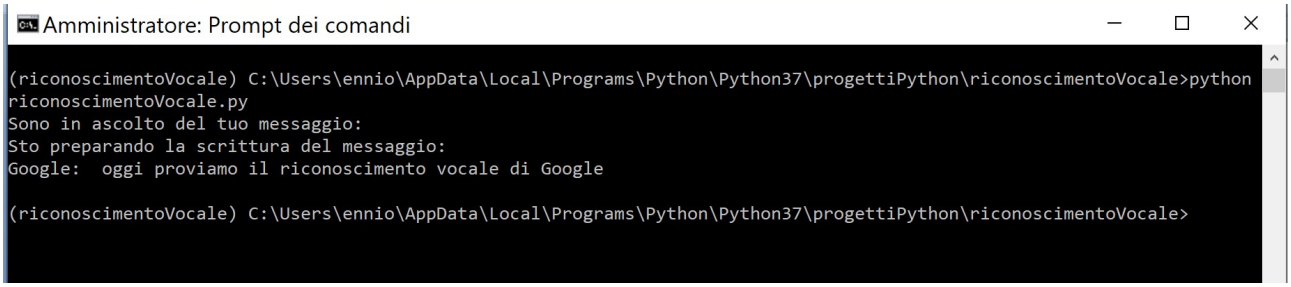
In ambiente CMD, andiamo nella cartella scripts della cartella Python37:
C:\Users\ennio\AppData\Local\Programs\Python\Python37>cd scripts
E diamo il seguente comando per creare la cartella virtuale riconoscimentoVocale (sottocartella di progettiPython):

C:\Users\ennio\AppData\Local\Programs\Python\Python37\Scripts>
virtualenv C:\Users\ennio\AppData\Local\Programs\Python\Python37/progettiPython/riconoscimentoVocale

Entriamo nella cartella scripts di riconoscimentoVocale:
C:\Users\ennio\AppData\Local\Programs\Python\Python37\progettiPython\riconoscimentoVocale>cd scripts
Diamo il comando: activate
Installiamo la libreria:
C:\Users\ennio\AppData\Local\Programs\Python\Python37\progettiPython\riconoscimentoVocale> pip install SpeechRecognition
Verifichiamo che l'installazione sia andata a buon fine:
pip list

Installare pyaudio (window 10):
pip install pipwin
pipwin install pyaudio

Salvare nella cartella sintesiVocale il file riconoscimentoVocale.py
Ed avviare il progetto nell'ambiente virtuale: python riconoscimentoVocale.py



(riconoscimentoVocale) questo prompt ci segnala che siamo nella cartella virtuale.

File "riconoscimentoVocale.py"

```
import speech_recognition as sr
recognizer_instance = sr.Recognizer()
with sr.Microphone() as source:
  recognizer_instance.adjust_for_ambient_noise(source)
  print("Sono in ascolto del tuo messaggio: ")
  audio = recognizer_instance.listen(source)
  print("Sto preparando la scrittura del messaggio: ")
try:
    text = recognizer_instance.recognize_google(audio, language="it-IT")
    print("Google: ", text)
except Exception as e:
  print (e)
```

SINTESI VOCALE

In ambiente repl.it
Selezionare Packages (colonna a sx)
importare gTTS

```
from gtts import gTTS
text="""Ciao a tutti. Oggi il professore di informatica vi fa ascoltare la voce di google"""
tts=gTTS(text=text,lang='it')
tts.save("tts_output_audio.mp3")
```