COME PASSARE DA EXCEL A PYTHON

GUIDA PER DTA SCIENTIST E DATA ANALYST

Intelligenza Artificiale (IA) e Python Come si Relazionano?

intelligenzaartificialeitalia.net/post/intelligenza-artificiale-ia-e-python-come-si-relazionano

2 novembre 2021

Intelligenza Artificiale (IA) e Python Come si Relazionano?

Python è uno dei linguaggi di programmazione più popolari utilizzati dagli sviluppatori oggi. Guido Van Rossum lo ha creato nel 1991 e sin dal suo inizio è stato uno dei linguaggi più utilizzati insieme a C++, Java, ecc.

Nel nostro tentativo di identificare quale sia il miglior linguaggio di programmazione per l'intelligenza artificiale e la rete neurale, Python ha preso un grande vantaggio.

Caratteristiche e vantaggi di Python per l'intelligenza artificiale

Python è un linguaggio interpretato che in parole povere significa che non ha bisogno di essere compilato in istruzioni in linguaggio macchina prima dell'esecuzione e può essere utilizzato direttamente dallo sviluppatore per eseguire il programma. Ciò lo rende abbastanza completo da consentire l'interpretazione del linguaggio da parte di un emulatore o di una macchina virtuale oltre al linguaggio macchina nativo, che è ciò che l'hardware comprende.

È un linguaggio di programmazione di alto livello e può essere utilizzato per scenari complicati. I linguaggi di alto livello trattano variabili, array, oggetti, complesse espressioni aritmetiche o booleane e altri concetti astratti di informatica per renderlo più completo, aumentando così in modo esponenziale la sua usabilità.

Python è anche un linguaggio di programmazione generico, il che significa che può essere utilizzato in più domini e tecnologie.

Python dispone anche di un sistema di tipi dinamici e di una gestione automatica della memoria che supporta un'ampia varietà di paradigmi di programmazione, inclusi quelli orientati agli oggetti, imperativi, funzionali e procedurali, per citarne alcuni.

Python è disponibile per tutti i sistemi operativi e ha anche un'offerta open source intitolata CPython che sta ottenendo una popolarità diffusa.

Diamo ora un'occhiata a come l'uso di Python per l'intelligenza Artificiale ci offre un vantaggio rispetto ad altri linguaggi di programmazione popolari.

Intelligenza Artificiale e Python Perché?

L'ovvia domanda che dobbiamo affrontare a questo punto è perché dovremmo scegliere Python per l'intelligenza artificiale rispetto ad altri .

Python offre maggior risultato con minor codice, infatti è stimato che serva solo 1/5 del codice per eseguire stesse operazioni rispetto ad altri linguaggi OOP. **Non c'è da stupirsi che sia uno dei più popolari oggi sul mercato.**

- Python ha librerie predefinite come Numpy per il calcolo scientifico, Scipy per il calcolo avanzato e Pybrain per l'apprendimento automatico (Python Machine Learning) che lo rendono uno dei migliori linguaggi per l'intelligenza artificiale.

- Gli sviluppatori Python di tutto il mondo forniscono supporto e assistenza completi tramite forum e tutorial, rendendo il lavoro del programmatore più semplice di qualsiasi altro linguaggio popolare.
- Python è indipendente dalla piattaforma ed è quindi una delle scelte più flessibili e popolari per l'utilizzo su diverse piattaforme e tecnologie con il minimo ritocco nella codifica di base.
- Python è il più flessibile di tutti gli altri con opzioni per scegliere tra l'approccio OOP e lo scripting. Puoi anche utilizzare l'IDE stesso per verificare la maggior parte dei codici ed è un vantaggio per gli sviluppatori alle prese con algoritmi diversi.

Python insieme all'intelligenza Artificiale

Python insieme a pacchetti come NumPy, scikit-learn, iPython Notebook e matplotlib costituiscono la base per avviare il tuo progetto di intelligenza artificiale.

NumPy viene utilizzato come contenitore per dati generici comprendenti un oggetto array N-dimensionale, strumenti per l'integrazione di codice C/C++, trasformata di Fourier, capacità di numeri casuali e altre funzioni.

Un'altra libreria utile è pandas, una libreria open source che fornisce agli utenti strutture dati di facile utilizzo e strumenti analitici per Python.

Matplotlib è un altro servizio che è una libreria di plottaggio 2D che crea figure di qualità di pubblicazione. È possibile utilizzare matplotlib fino a 6 toolkit di interfaccia utente grafica, server di applicazioni Web e script Python.

Il tuo prossimo passo sarà esplorare il clustering di k-means e anche raccogliere conoscenze su alberi decisionali, previsione numerica continua, regressione logistica, ecc.

Alcune delle librerie AI Python più comunemente utilizzate sono AIMA, pyDatalog, SimpleAI, EasyAi, ecc. Esistono anche librerie Python per l'apprendimento automatico come PyBrain, MDP, scikit, PyML.

Diamo un'occhiata un po' più in dettaglio alle varie librerie Python nell'IA e perché questo linguaggio di programmazione viene utilizzato per l'IA.

Librerie Python per l' Intelligenza Artificiale

- **AIMA** – Implementazione in Python di algoritmi da "Artificial Intelligence: A Modern Approach" di Russell e Norvig.
- **pyDatalog** – Motore di programmazione logica in Python
- **SimpleAI** – Implementazione in Python di molti degli algoritmi di intelligenza artificiale descritti nel libro “Artificial Intelligence, a Modern Approach”. Si concentra sulla fornitura di una libreria facile da usare, ben documentata e testata.
- **EasyAI** – Semplice motore Python per giochi a due giocatori con AI (Negamax, tabelle di trasposizione, risoluzione di giochi).

Python per l' Apprendimento Automatico (ML)

Diamo un'occhiata al motivo per cui Python viene utilizzato per l'apprendimento automatico e le varie librerie che offre allo scopo.

- **PyBrain** : un algoritmo flessibile, semplice ma efficace per le attività di **machine learning** . È anche una libreria modulare di Machine Learning per Python che fornisce una varietà di ambienti predefiniti per testare e confrontare algoritmi.
- **PyML** – Un framework bilaterale scritto in Python che si concentra su SVM e altri metodi del kernel. È supportato su Linux e Mac OS X.
- **Scikit-learn** – Scikit-learn è uno strumento efficiente per l'analisi dei dati durante l'utilizzo di Python. È open source e la libreria di machine learning generica più popolare.
- **MDP-Toolkit** – Un altro framework di elaborazione dati Python che può essere facilmente ampliato, ha anche una raccolta di algoritmi di apprendimento supervisionati e non supervisionati e altre unità di elaborazione dati che possono essere combinate in sequenze di elaborazione dati e architetture di rete feed-forward più complesse. L'implementazione di nuovi algoritmi è facile ed intuitiva. La base di algoritmi disponibili è in costante aumento e include metodi di elaborazione del segnale (analisi dei componenti principali, analisi dei componenti indipendenti e analisi delle caratteristiche lente), metodi di apprendimento multipli ([Hessian] Locally Linear Embedding), diversi classificatori, metodi probabilistici (analisi fattoriale, RBM), metodi di pre-trattamento dei dati e molti altri.

Librerie Python per il linguaggio naturale e l'elaborazione del testo

NLTK – Moduli Python open source, dati linguistici e documentazione per ricerca e sviluppo nell'elaborazione del linguaggio naturale e analisi del testo con distribuzioni per Windows, Mac OSX e Linux.

Python contro altri linguaggi popolari

Vediamo ora dove si trova Python con un altro linguaggio informatico per l'intelligenza artificiale come C++ e Java.

Python contro C++ per l'intelligenza Artificiale

- Python è un linguaggio più popolare rispetto a C++ per l'intelligenza artificiale e guida con un voto del 57% tra gli sviluppatori. Questo perché Python è facile da imparare e implementare. Con le sue numerose librerie, possono essere utilizzati anche per l'analisi dei dati.
- Per quanto riguarda le prestazioni, il C++ supera Python. Questo perché C++ ha il vantaggio di essere un linguaggio tipizzato staticamente e quindi non ci sono errori di digitazione durante il runtime. C++ crea anche codice runtime più compatto e veloce.
- Python è un linguaggio dinamico (al contrario di statico) e riduce la complessità quando si tratta di collaborare, il che significa che è possibile implementare funzionalità con meno codice. A differenza di C++, dove tutti i compilatori significativi tendono a eseguire ottimizzazioni specifiche e possono essere specifici della piattaforma, il codice Python può essere eseguito praticamente su qualsiasi piattaforma senza perdere tempo su configurazioni specifiche.
- Con l'aumento delle capacità di offerta di elaborazione accelerata da GPU per il parallelismo che ha portato alla creazione di librerie come CUDA Python e cuDNN, Python ha il vantaggio su C++. Ciò significa che una parte sempre maggiore del calcolo effettivo per i carichi di lavoro di machine learning viene scaricato sulle GPU e il risultato è che qualsiasi vantaggio prestazionale che il C++ può avere sta diventando sempre più irrilevante.
- Python vince su C++ per quanto riguarda la semplicità del codice, specialmente tra i nuovi sviluppatori. Il C++ essendo un linguaggio di livello inferiore richiede più esperienza e abilità da padroneggiare.
- La semplice sintassi di Python consente anche un processo ETL (Estrai, Trasforma, Carica) più naturale e intuitivo e significa che è più veloce per lo sviluppo rispetto a C++, consentendo agli sviluppatori di testare algoritmi di apprendimento automatico senza doverli implementare rapidamente.

Tra C++ e Python, quest'ultimo ha più margine ed è più adatto per l'IA. Con la sua semplice sintassi e leggibilità che promuovono il test rapido di complessi algoritmi di apprendimento automatico e una fiorente comunità supportata da strumenti collaborativi come Jupyter Notebooks e Google Colab, Python vince la corona.

Usare Java per programmare IA

Per capire come programmare l'intelligenza artificiale in Java, è essenziale sapere dove si trova rispetto a Python.

- Java è un linguaggio compilato mentre Python è un linguaggio interpretato.
- Le due lingue sono anche scritte in modo diverso. Una struttura in Java è racchiusa tra parentesi graffe. Python usa il rientro per eseguire le stesse attività.
- Java è anche più lento dal punto di vista delle prestazioni e per lo sviluppo di applicazioni di fascia alta in AI, Python è più preferito dagli sviluppatori.

Java Artificial Intelligence Library è la risposta di Java a Python, ma è ancora meno accessibile agli sviluppatori per evidenti ragioni. L'approccio moderno all'IA di Java Norvig Russell ha spianato la strada a molti per sedersi e notare perché potrebbe essere il linguaggio migliore per una rete neurale.

Piccolo Caso Studio

È stato condotto un esperimento per utilizzare l'intelligenza artificiale con un Internet of Things per creare un'applicazione IoT per l'analisi comportamentale dei dipendenti. Il software fornisce feedback utili ai dipendenti attraverso le emozioni dei dipendenti e l'analisi del comportamento, migliorando così i cambiamenti positivi nella gestione e nelle abitudini di lavoro.

Utilizzando le librerie di apprendimento automatico **Python**, **opencv** e **haarcascading** per la formazione delle applicazioni, è stato creato un **POC** di esempio per rilevare emozioni di base come felicità, rabbia, tristezza, disgusto, sospetto, disprezzo, sarcasmo e sorpresa attraverso telecamere wireless collegate in vari punti della baia.

I dati raccolti sono stati inviati a un database di cloud computing centralizzato in cui è possibile recuperare il quoziente emotivo giornaliero all'interno della baia o anche l'intero ufficio con un clic di un pulsante tramite un dispositivo Android o desktop.

Gli sviluppatori stanno facendo progressi graduali nell'analisi di ulteriori punti complessi sulle emozioni facciali e estrae maggiori dettagli con l'aiuto di algoritmi di deep learning e apprendimento automatico che possono aiutare ad analizzare le prestazioni dei singoli dipendenti e supportare il corretto feedback dei dipendenti/team.

Conclusione

Python svolge un ruolo fondamentale nel linguaggio di codifica AI fornendogli buoni framework come scikit-learn: machine learning in Python, che soddisfa quasi tutte le esigenze in questo campo e D3.js – Data-Driven Documents in JS, che è uno dei strumenti di visualizzazione più potenti e facili da usare.

Oltre ai framework, la sua rapida prototipazione lo rende un linguaggio importante da non ignorare. L'intelligenza artificiale ha bisogno di molte ricerche, e quindi è necessario non richiedere un codice standard di 500 KB in Java per testare una nuova ipotesi, che non porterà mai a termine il progetto. In Python, quasi ogni idea può essere rapidamente convalidata attraverso 20-30 righe di codice (lo stesso per JS con librerie). Pertanto, è un linguaggio piuttosto utile per il bene dell'intelligenza artificiale.

Quindi è abbastanza evidente che Python è il miglior linguaggio di programmazione per l'intelligenza Artificiale. Oltre ad essere il miglior linguaggio per l'intelligenza artificiale, Python è utile per molti altri obiettivi.

Una Guida Semplice e Completa per passare da Excel a Python Usando le Librerie Pandas e Numpy

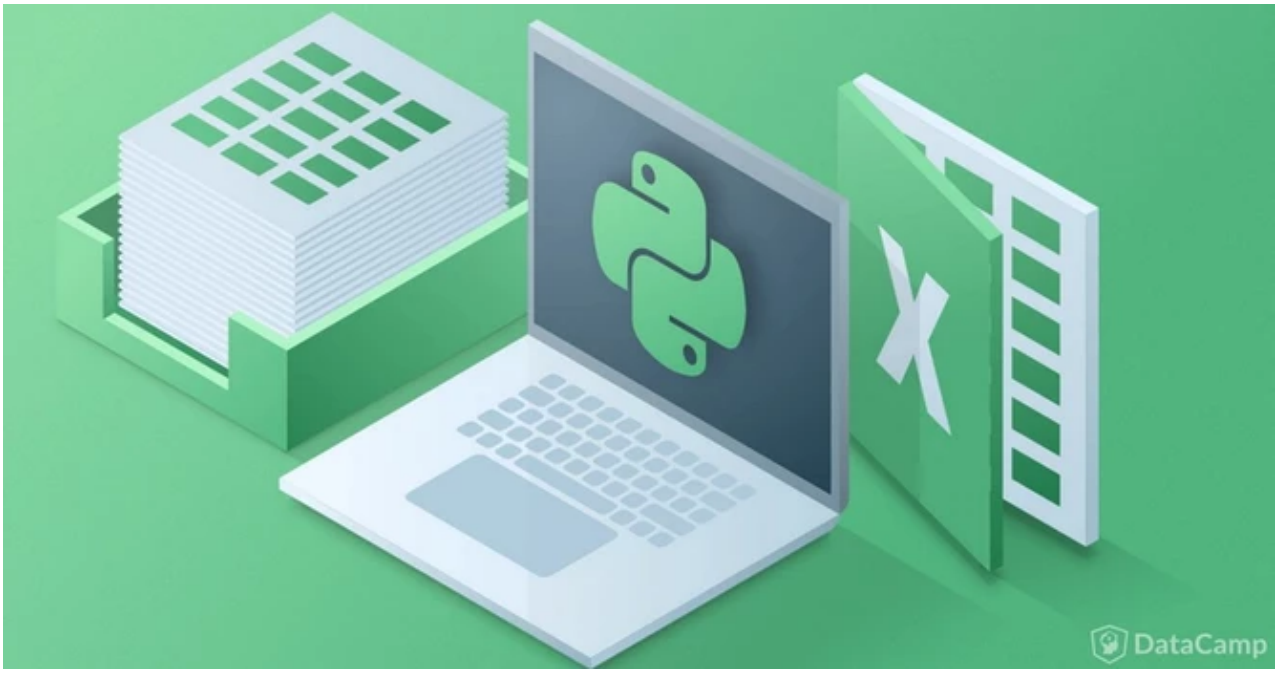
intelligenzaartificialeitalia.net/post/una-guida-semplice-e-completa-per-passare-da-excel-a-python-usando-le-librerie-pandas-e-numpy

26 maggio 2021

So che l'idea di imparare cose che puoi già fare in Excel / VBA in un ambiente completamente nuovo come Python non sembra così eccitante, tuttavia, i vantaggi offerti da Python, le molte librerie disponibili e l'alta richiesta delle aziende di data scientist rende ancora più interessante questo " **switch** ".

Per questo motivo, ho pensato a un utile e semplice guida che avrei voluto avere quando sono passato da Excel a Python. In questo articolo, utilizzeremo le librerie Panda e Numpy di Python per sostituire molti Excel funzioni che probabilmente hai usato in passato.

Sommario



Una Guida Semplice e Completa per passare da Excel a Python Usando le Librerie Pandas e Numpy

Prima di iniziare :

- Se non sai perchè utilizzeremo python, clicca qui
- Se non hai ancora installato Python, clicca qui
- Se non sai come scaricare e gestire le librerie, clicca qui
- Se non sai cosa sia un Dataset, clicca qui
- Se non sai quali sono le migliori librerie per l'I.A. , clicca qui

IMPORTARE I DATI

In questa guida, utilizzeremo un file Excel con formato .csv, ma non preoccuparti, un vantaggio di Python è la possibilità di poter leggere quasi ogni tipo di file, txt, json, xml, html e i vari formati excel. Ti lascio il link per vedere come importare i diversi tipi di file.

Nel caso in cui vuoi solo esercitarti e non sai che dataset usare ti lascio questo link per vedere dove trovare dataset gratis a scaricare.

Se vuoi seguire passo passo l'intero articolo ti lascio il file .csv che utilizzeremo in questa guida.

EsempioPython

.csv

Scarica CSV • 9.54MB

Per iniziare con questa guida, importiamo le librerie Pandas, Numpy e Matplotlib.

```
#importiamo le librerie
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Con questo, possiamo dare una prima occhiata ai dati. Per farlo, usiamo pd.read_csv (). Assicurati che il CSV e il tuo script Python si trovino nella stessa cartella.

```
dataset = pd.read_csv("EsempioPython.csv")
print(dataset)
```

Una volta letto questo file CSV, gli diamo un nome. In questo caso, l'ho chiamato **dataset**. Come consuetudine si usa **df** che sta per dataframe ed è il nome tipico dato dopo aver letto un file CSV. Dopo eseguendo il codice sottostante, otteniamo il seguente output.

Nel nostro caso stiamo usando **jupyter notebook** per scrivere il codice quindi se stai utilizzando il terminale o altri editor sarà stampato in modo diverso.

	Retailer_country	Order_method_type	Retailer_type	Product_line	Product_type	Product	Year	Quarter	Revenue	Quantity	Gross_margin
0	United States	Fax	Outdoors Shop	Camping Equipment	Cooking Gear	TrailChef Deluxe Cook Set	2012	1	59628.66	489.0	0.347548
1	United States	Fax	Outdoors Shop	Camping Equipment	Cooking Gear	TrailChef Double Flame	2012	1	35950.32	252.0	0.474274
2	United States	Fax	Outdoors Shop	Camping Equipment	Tents	Star Dome	2012	1	89940.48	147.0	0.352772
3	United States	Fax	Outdoors Shop	Camping Equipment	Tents	Star Gazer 2	2012	1	165883.41	303.0	0.282938
4	United States	Fax	Outdoors Shop	Camping Equipment	Sleeping Bags	Hibernator Lite	2012	1	119822.20	1415.0	0.291450
...	...	...	...	...	...	...	...	...	...	...	...
88470	Spain	Sales visit	Outdoors Shop	Mountaineering Equipment	Rope	Husky Rope 60	2014	3	30865.50	171.0	0.299114

Importare e stampare file CSV con python
Sembra simile a un foglio di calcolo Excel, ma in questo formato sarà più facile lavorare i dati i dati. Ora ti mostrerò in Python come eseguire alcune funzioni comuni che probabilmente utilizzato in Excel.

SOMMA, MEDIA, MAX, MIN e COUNT

Le popolari funzioni di Excel possono essere facilmente sostituite con i metodi Pandas.

Prima di farti vedere le singole funzioni, ci tengo a dirti che Se vogliamo ottenere la maggior parte delle funzioni elencate sopra, utilizziamo il metodo **.describe()**. Diamo un'occhiata.

```
print(dataset.describe())
```

Output:

```
print(dataset.describe())
```

	Year	Quarter	Revenue	Quantity	Gross_margin
count	88475.000000	88475.000000	8.847500e+04	88475.000000	87894.000000
mean	2012.855281	2.321300	4.263829e+04	780.586166	0.449718
std	0.778342	1.075003	6.578402e+04	1541.645422	0.123642
min	2012.000000	1.000000	0.000000e+00	1.000000	-12.853678
25%	2012.000000	1.000000	8.184360e+03	131.000000	0.369880
50%	2013.000000	2.000000	2.102628e+04	333.000000	0.450634
75%	2013.000000	3.000000	5.039060e+04	816.000000	0.520130
max	2014.000000	4.000000	1.635688e+06	67875.000000	0.770476

Come puoi vedere per ogni colonna ne verranno calcolate le principali operazioni statistiche di base. Ora procediamo nel vedere come sommare due o più colonne.

Per specificare su quali colonne operare utilizzeremo le parentisi subito dopo la nostra variabile contenente tutti i dati, con all'interno i nomi delle colonne che vuoi selezionare. Ne vediamo subito un esempio.

```
#Vogliamo calcolare le vendite totali dei negozi
vendite_totali = dataset["Quantity"].sum()
print(vendite_totali)
```

Output:

```
#Vogliamo calcolare le vendite totali dei negozi
vendite_totali = dataset["Quantity"].sum()
print(vendite_totali)
```

```
69062361.0
```

Somma di una Colonna con Python

Quindi, se vogliamo selezionare una colonna specifica, selezioniamo prima con parentesi quadre ["colonna"] e poi usa il metodo di cui abbiamo bisogno (.sum (), .mean (), .count (), ecc.). Ad esempio, calcoliamo la media, il max e il min delle quantità vendute dei negozi.

```
print("media : " + str( dataset["Quantity"].mean() ) )
print("MAX : " + str( dataset["Quantity"].max() ) )
print("MIN : " + str( dataset["Quantity"].min()) )

Output:
media : 780.5861655834982
MAX : 67875.0
MIN : 1.0
```

Come puoi vedere è veramente semplice, quasi più di Excel dopo che ci avrai preso mano. Nel caso in cui desideriamo contare il numero di dati specifici all'interno di una colonna, possiamo usare il .value_counts () metodo. Ora vediamo un esempio dove contiamo quale tipi di metodi sono stati usati negli ordini.

```
print(dataset["Order_method_type"].value_counts())

Output:

Web            69181
Sales visit     7168
Telephone       5814
E-mail          3092
Fax             1664
Mail            1024
Special          532
Name: Order_method_type, dtype: int64
```

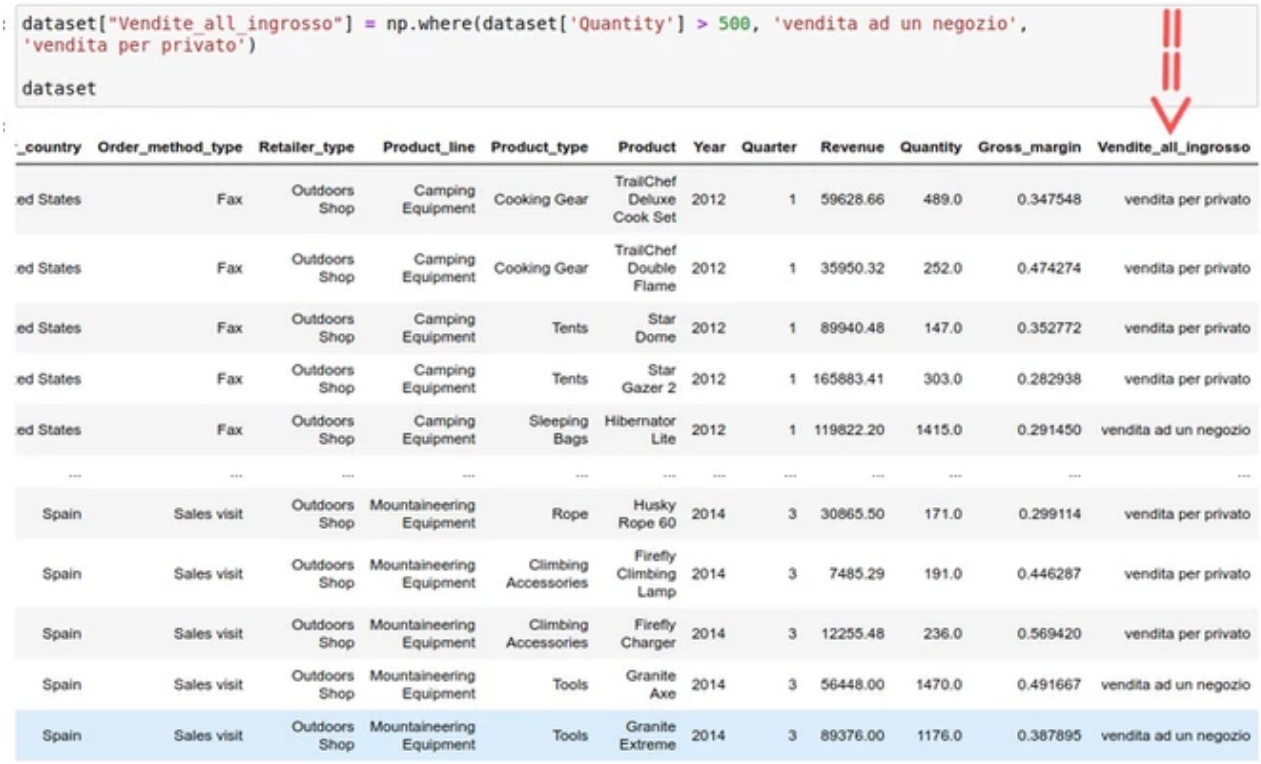
Come puoi vedere anche la gestione delle variabili di testo non è minimamente un problema per Python. Ora facciamo un salto di livello e vediamo come implementare l' IF di Excel con numpy, utilizzando sempre semplici funzioni.

SOSTITUIAMO L' IF

Possiamo facilmente sostituire la funzione IF di Excel utilizzando Numpy. Immaginiamo di voler sapere quali vendite sono superiori ad una quantità di 200 pezzi e creare quindi un nuovo attributo per ogni riga la dove la vendita risulta maggiore o minore di questa quantità.

Possiamo farlo facilmente con quanto segue codice.

```
dataset["Vendite_all_ingrosso"] = np.where(dataset['Quantity'] > 500,  'vendita ad
un negozio', 'vendita per privato')

print("dataset")

Output:
```



```
dataset["Vendite_all_ingrosso"] = np.where(dataset['Quantity'] > 500, 'vendita ad un negozio',
'vendita per privato')

dataset
```

_country	Order_method_type	Retailer_type	Product_line	Product_type	Product	Year	Quarter	Revenue	Quantity	Gross_margin	Vendite_all_ingrosso
ed States	Fax	Outdoors Shop	Camping Equipment	Cooking Gear	TrailChef Deluxe Cook Set	2012	1	59628.66	489.0	0.347548	vendita per privato
ed States	Fax	Outdoors Shop	Camping Equipment	Cooking Gear	TrailChef Double Flame	2012	1	35950.32	252.0	0.474274	vendita per privato
ed States	Fax	Outdoors Shop	Camping Equipment	Tents	Star Dome	2012	1	89940.48	147.0	0.352772	vendita per privato
ed States	Fax	Outdoors Shop	Camping Equipment	Tents	Star Gazer 2	2012	1	165883.41	303.0	0.282938	vendita per privato
ed States	Fax	Outdoors Shop	Camping Equipment	Sleeping Bags	Hibernator Lite	2012	1	119822.20	1415.0	0.291450	vendita ad un negozio
...	...	...	...	...	...	...	...	...	...	...	...
Spain	Sales visit	Outdoors Shop	Mountaineering Equipment	Rope	Husky Rope 60	2014	3	30865.50	171.0	0.299114	vendita per privato
Spain	Sales visit	Outdoors Shop	Mountaineering Equipment	Climbing Accessories	Firefly Climbing Lamp	2014	3	7485.29	191.0	0.446287	vendita per privato
Spain	Sales visit	Outdoors Shop	Mountaineering Equipment	Climbing Accessories	Firefly Charger	2014	3	12255.48	236.0	0.569420	vendita per privato
Spain	Sales visit	Outdoors Shop	Mountaineering Equipment	Tools	Granite Axe	2014	3	56448.00	1470.0	0.491667	vendita ad un negozio
Spain	Sales visit	Outdoors Shop	Mountaineering Equipment	Tools	Granite Extreme	2014	3	89376.00	1176.0	0.387895	vendita ad un negozio

Sostituire l' IF di Excel con python

Come puoi vedere np.where () necessita di 3 argomenti: la condizione, il valore se la condizione è Vera e il valore se la condizione è Falsa.

Supponiamo ora di avere una condizione più complessa, la quale richiede di classificare nello specifico a che tipo di struttura è stato l'acquisto basandosi sulle quantità. In questo case, abbiamo più di 2 valori, quindi usiamo **np.select ()** necessita di argomenti:

1. un elenco di condizioni
2. un elenco di valori

Un elenco in Python è rappresentato dalle parentesi quadre [].

Vediamo come si fa in modo pratico.

```
condizioni = [(dataset['Quantity']>=900),
            (dataset['Quantity']>=800)&(dataset['Quantity']<900),
            (dataset['Quantity']>=700)&(dataset['Quantity']<800),
            (dataset['Quantity']>=600)&(dataset['Quantity']<700),
            (dataset['Quantity']>=500)&(dataset['Quantity']<600),
            (dataset['Quantity']<500) ]

valori= ['Negozio vendita ingrosso',
 'Centro commerciale',
 'Catena di negozi',
 'Negozio Grande',
 'Negozio medio-piccolo',
 'Ptivato-Retailer']

dataset["Acquirente"] = np.select(condizioni, valori)

print("dataset")

Output:
```

```
condizioni = [(dataset['Quantity']>=900),
              (dataset['Quantity']>=800) & (dataset['Quantity']<900),
              (dataset['Quantity']>=700) & (dataset['Quantity']<800),
              (dataset['Quantity']>=600) & (dataset['Quantity']<700),
              (dataset['Quantity']>=500) & (dataset['Quantity']<600),
              (dataset['Quantity']<500) ]

valori= ['Negozio vendita ingrosso',
         'Centro commerciale',
         'Catena di negozi',
         'Negozio Grande',
         'Negozio medio-piccolo',
         'Ptivato-Retailer']

dataset["Acquirente"] = np.select(condizioni, valori)

dataset
```

rder_method_type	Retailer_type	Product_line	Product_type	Product	Year	Quarter	Revenue	Quantity	Gross_margin	Vendite_all_ingrosso	Acquirente
Fax	Outdoors Shop	Camping Equipment	Cooking Gear	TrailChef Deluxe Cook Set	2012	1	59628.66	489.0	0.347548	vendita per privato	Ptivato-Retailer
Fax	Outdoors Shop	Camping Equipment	Cooking Gear	TrailChef Double Flame	2012	1	35950.32	252.0	0.474274	vendita per privato	Ptivato-Retailer
Fax	Outdoors Shop	Camping Equipment	Tents	Star Dome	2012	1	89940.48	147.0	0.352772	vendita per privato	Ptivato-Retailer
Fax	Outdoors Shop	Camping Equipment	Tents	Star Gazer 2	2012	1	165883.41	303.0	0.282938	vendita per privato	Ptivato-Retailer
Fax	Outdoors Shop	Camping Equipment	Sleeping Bags	Hibernator Lite	2012	1	119822.20	1415.0	0.291450	vendita ad un negozio	Negozio vendita ingrosso
...	...	...	...	...	...	...	...	...	...	...	...
Sales visit	Outdoors Shop	Mountaineering Equipment	Rope	Husky Rope 60	2014	3	30865.50	171.0	0.299114	vendita per privato	Ptivato-Retailer
Sales visit	Outdoors Shop	Mountaineering Equipment	Climbing Accessories	Firefly Climbing Lamp	2014	3	7485.29	191.0	0.446287	vendita per privato	Ptivato-Retailer
Sales visit	Outdoors Shop	Mountaineering Equipment	Climbing Accessories	Firefly Charger	2014	3	12255.48	236.0	0.569420	vendita per privato	Ptivato-Retailer
Sales visit	Outdoors Shop	Mountaineering Equipment	Tools	Granite Axe	2014	3	56448.00	1470.0	0.491667	vendita ad un negozio	Negozio vendita ingrosso
Sales visit	Outdoors Shop	Mountaineering Equipment	Tools	Granite Extreme	2014	3	89376.00	1176.0	0.387895	vendita ad un negozio	Negozio vendita ingrosso

Sostituire l' IF di Excel con python

Come puoi vedere manipolare colonne o righe con Python non è molto difficile, ma andiamo avanti e vediamo quali altre funzioni possiamo replicare da Excel.

Immaginiamo di voler le vendite di prodotti solo di uno specifico Paese. Per farlo, in primo luogo, scrivi la condizione **dataset** ["**Retailer_country**"] == "**Italy**" e poi selezioniamo quella condizione all'interno del frame **dataset** utilizzando parentesi quadre []

```
dataset_italia = dataset[dataset["Retailer_country"]=="Italy"]
dataset_italia
```

Output:

```
dataset_italia = dataset[dataset["Retailer_country"]=="Italy"]
dataset_italia
```

	Retailer_country	Order_method_type	Retailer_type	Product_line	Product_type
7934	Italy	Fax	Outdoors Shop	Mountaineering Equipment	Rope
7935	Italy	Fax	Outdoors Shop	Mountaineering Equipment	Rope
7936	Italy	Fax	Outdoors Shop	Mountaineering Equipment	Rope
7937	Italy	Fax	Outdoors Shop	Mountaineering Equipment	Climbing Accessories
7938	Italy	Fax	Outdoors Shop	Mountaineering Equipment	Climbing Accessories
...	...	...	...	...	...
88276	Italy	Sales visit	Outdoors Shop	Camping Equipment	Sleeping Bags
88277	Italy	Sales visit	Outdoors Shop	Camping Equipment	Lanterns
88278	Italy	Sales visit	Outdoors Shop	Camping Equipment	Lanterns
88279	Italy	Sales visit	Outdoors Shop	Mountaineering Equipment	Rope
88280	Italy	Sales visit	Outdoors Shop	Mountaineering Equipment	Safety

4018 rows × 13 columns

Sostituire l' IF di Excel con python
Abbiamo selezionato solo le vendite avvenute in italia e le abbiamo inserite dentro un nuovo dataframe **dataset_italia** . Adesso possiamo eseguire qualsiasi calcolo che abbiamo visto prima "Sum, Average, Max, Min, Count" .

Se abbiamo due o più condizioni, il codice sarà simile a quello sopra, ma con alcuni cambiamenti. Immaginiamo di voler creare un ulteriore dataframe contenente solo le vendite fatte in Spagna nel 2014.

```
dataset_spagna_2014 = dataset[(dataset["Retailer_country"]=="Spain") &
(dataset["Year"]==2014)]

dataset_spagna_2014

Output:
```

```
dataset_spagna_2014 = dataset[(dataset["Retailer_country"]=="Spain") & (dataset["Year"]==2014)]
dataset_spagna_2014
```

	Retailer_country	Order_method_type	Retailer_type	Product_line	Product_type	Product	Year	Quarter	Revenue	Quantity	Gr
75187	Spain	Fax	Sports Store	Camping Equipment	Cooking Gear	TrailChef Canteen	2014	1	12160.14	1167.0	
75188	Spain	Fax	Sports Store	Camping Equipment	Cooking Gear	TrailChef Cup	2014	1	9264.20	4211.0	
75189	Spain	Fax	Sports Store	Camping Equipment	Cooking Gear	TrailChef Double Flame	2014	1	34747.38	241.0	
75190	Spain	Fax	Sports Store	Camping Equipment	Sleeping Bags	Hibernator Self - Inflating Mat	2014	1	59245.90	490.0	
75191	Spain	Fax	Sports Store	Camping Equipment	Sleeping Bags	Hibernator Camp Cot	2014	1	55387.08	558.0	
...	...	...	...	...	...	...	...	...	...	...	...
88470	Spain	Sales visit	Outdoors Shop	Mountaineering Equipment	Rope	Husky Rope 60	2014	3	30865.50	171.0	
88471	Spain	Sales visit	Outdoors Shop	Mountaineering Equipment	Climbing Accessories	Firefly Climbing Lamp	2014	3	7485.29	191.0	
88472	Spain	Sales visit	Outdoors Shop	Mountaineering Equipment	Climbing Accessories	Firefly Charger	2014	3	12255.48	236.0	
88473	Spain	Sales visit	Outdoors Shop	Mountaineering Equipment	Tools	Granite Axe	2014	3	56448.00	1470.0	
88474	Spain	Sales visit	Outdoors Shop	Mountaineering Equipment	Tools	Granite Extreme	2014	3	89376.00	1176.0	

890 rows × 13 columns

Sostituire l' IF di Excel con python
Poiché ci sono 2 condizioni, potremmo usare & | che rappresenta e / o rispettivamente.
Tieni a mente che ogni condizione dovrebbe essere tra parentesi.

PULIZIA DAI DI BASE

Vedremo alcuni metodi utilizzati per la pulizia dei dati. Continueremo a utilizzare il frame df_excel che abbiamo definito in precedenza.

STANDARDIZZAZIONE DEL TESTO

Potrebbe capitare, soprattutto se si opera con dati correlati a variabili non numeriche, di dover standardizzare il testo in modo da poterlo successivamente convertire in una variabile categorica numerica.

Cambia il faromato del testo con .str.lower, .str.upper o .str.title Per accedere alle stringhe contenute in una colonna, usiamo .str possiamo quindi cambiare maiuscole e minuscole di testo con quanto segue .

```
dataset["Product"].str.title()
dataset["Product"].str.upper()
dataset["Product"].str.lower()

Output:
```

```
dataset["Product"].str.title()

0        Trailchef Deluxe Cook Set
1           Trailchef Double Flame
2                       Star Dome
3                     Star Gazer 2
4                  Hibernator Lite
                   ...
88470                Husky Rope 60
88471         Firefly Climbing Lamp
88472               Firefly Charger
88473                  Granite Axe
88474               Granite Extreme
Name: Product, Length: 88475, dtype: object
```

```
dataset['Product'].str.upper()

0        TRAILCHEF DELUXE COOK SET
1           TRAILCHEF DOUBLE FLAME
2                       STAR DOME
3                     STAR GAZER 2
4                  HIBERNATOR LITE
                   ...
88470                HUSKY ROPE 60
88471         FIREFLY CLIMBING LAMP
88472               FIREFLY CHARGER
88473                  GRANITE AXE
88474               GRANITE EXTREME
Name: Product, Length: 88475, dtype: object
```

```
dataset['Product'].str.lower()

0        trailchef deluxe cook set
1           trailchef double flame
2                       star dome
3                     star gazer 2
4                  hibernator lite
                   ...
88470                husky rope 60
88471         firefly climbing lamp
88472               firefly charger
88473                  granite axe
88474               granite extreme
Name: Product, Length: 88475, dtype: object
```

Per sovrascrivere la colonna ti basterà assegnare il comando precedente sulla colonna.

Questo è il codice.

```
dataset['Product'] = dataset['Product'].str.lower()
```

RIMOZIONE VALORI NULLI

Ora vediamo come eliminare i valori nulli presenti nel dataset la sintassi della funzione dropna () è:

```
dropna(self, axis=0, how="any", thresh=None, subset=None, inplace=False)
```

- **axis** : i valori possibili sono {0 o "indice", 1 o "colonne"}, valore predefinito 0. Se 0, rilascia le righe con valori nulli. Se 1, rilascia le colonne con valori mancanti.
- **how** : i valori possibili sono {'any', 'all'}, il valore predefinito 'any'. Se "any", rilascia la riga / colonna se uno dei valori è nullo. Se "all", rilascia la riga / colonna se mancano tutti i valori.
- **thresh** : un valore int per specificare la soglia per l'operazione di eliminazione.
- **subset** : specifica le righe / colonne in cui cercare valori nulli.
- **inplace** : un valore booleano. Se True, il DataFrame di origine viene modificato e viene restituito None.

Per sovrascrivere il Dataset, senza valori nulli, ci basterà utilizzare questa riga di codice.

```
dataset.shape
```

```
(88475, 13)
```

```
dataset = dataset.dropna()
dataset.shape
```

```
(87894, 13)
```

Eliminare valori nulli dal Dataset con Python e pandas

SOSTITUIAMO I GRAFICI DI EXCEL CON MATPLOTLIB DI PYTHON

Questa parte dell'articolo ti introdurrà alla rappresentazione grafica in Python con Matplotlib, che è probabilmente la libreria di rappresentazione grafica e di visualizzazione dei dati più popolare per Python.

Installazione Il modo più semplice per installare matplotlib è usare pip. Digita il seguente comando nel terminale:

```
pip install matplotlib #python2
pip install matplotlib #python3
```

OPPURE, puoi scaricarlo da qui e installarlo manualmente.

Per iniziare (tracciamo una linea)

Il codice sembra autoesplicativo. Sono stati seguiti i seguenti passaggi:

- Definisci i valori dell'asse x e corrispondenti dell'asse y come Liste.
- **Stampali** usando la funzione **.plot ()** .
- Assegna un nome all'asse x e all'asse y usando le **funzioni .xlabel ()** e **.ylabel ()** .
- Dai un titolo alla tua trama usando la funzione **.title ()** .
- Infine, per visualizzare il grafico, utilizziamo la funzione **.show ()** .

Tracciare due o più linee sullo stesso grafico

- Qui, tracciamo due linee sullo stesso grafico. Li differenziamo dando loro un nome (**etichetta**) che viene passato come argomento della funzione .plot ().
- La piccola casella rettangolare che fornisce informazioni sul tipo di linea e sul suo colore è chiamata legenda. Possiamo aggiungere una legenda al nostro **grafico** usando la funzione **.legend ()** .

Customizzazione dei grafici Qui, discutiamo alcune personalizzazioni elementari applicabili su quasi tutti i grafici.

Come puoi vedere, abbiamo effettuato diverse personalizzazioni come :

- impostazione della larghezza della linea, dello stile della linea, del colore della linea.
- impostazione del pennarello, colore del viso del pennarello, dimensione del pennarello.
- sovrascrivendo l'intervallo degli assi x e y. Se l'override non viene eseguito, il modulo pyplot utilizza la funzione di scala automatica per impostare l'intervallo e la scala dell'asse.

Grafico a barre

- Qui, usiamo la funzione **plt.bar ()** per tracciare un grafico a barre.
- Le coordinate x del lato sinistro delle barre vengono trasmesse insieme alle altezze delle barre.
- puoi anche dare un nome alle coordinate dell'asse x definendo **tick_labels**

Istogramma

- Qui, usiamo la funzione **plt.hist ()** per tracciare un istogramma.
- le frequenze vengono passate come elenco delle **età** .
- L'intervallo può essere impostato definendo una tupla contenente il valore minimo e massimo.
- Il passaggio successivo consiste nel " **raggruppare** " l'intervallo di valori, ovvero dividere l'intero intervallo di valori in una serie di intervalli, quindi contare quanti valori rientrano in ciascun intervallo. Qui abbiamo definito **bin** = 10. Quindi, ci sono un totale di 100/10 = 10 intervalli.

Trama a dispersione

- Qui, usiamo la funzione **plt.scatter ()** per tracciare un **grafico** a dispersione.
- Come una linea, qui definiamo anche i valori dell'asse xe corrispondenti.

- **L'** argomento **marker** viene utilizzato per impostare il carattere da utilizzare come marker. La sua dimensione può essere definita utilizzando **il** parametro **s** .

Grafico a torta

- Qui, tracciamo un grafico a torta usando il metodo **plt.pie ()** .
- Prima di tutto, definiamo le **etichette** utilizzando un elenco chiamato **attività** .
- Quindi, una parte di ciascuna etichetta può essere definita utilizzando un altro elenco chiamato **slice** .
- Il colore per ciascuna etichetta viene definito utilizzando un elenco chiamato **colori** .
- **shadow = True** mostrerà un'ombra sotto ogni etichetta nel grafico a torta.
- **startangle** ruota l'inizio del grafico a torta di determinati gradi in senso antiorario dall'asse x.
- **explode** viene utilizzato per impostare la frazione di raggio con cui compensiamo ogni cuneo.
- **autopct** viene utilizzato per formattare il valore di ciascuna etichetta. Qui, l'abbiamo impostato per mostrare il valore percentuale solo fino a 1 cifra decimale.

Tracciare curve di una data equazione

- Per impostare i valori dell'asse x, usiamo il metodo **np.arange ()** in cui i primi due argomenti sono per l'intervallo e il terzo per l'incremento graduale. Il risultato è un array numpy.
- Per ottenere i valori dell'asse y corrispondenti, usiamo semplicemente il metodo **np.sin ()** predefinito **sull'array** numpy.
- Infine, tracciamo i punti passando gli array xey alla funzione **plt.plot ()** .

Quindi, in questa parte, abbiamo discusso i vari tipi di grafici che possiamo creare in matplotlib.

Come hai potuto vedere python offre le stesse se non più funzionalità di excel, spero che questa quida ti possa essere d'aiuto nella transizione. Per qualsiasi tipo di domanda o dubbio non esitare a lasciare un commento.

Trucchi Python per la data science

intelligenzaartificialeitalia.net/post/trucchi-python-per-la-data-science

23 gennaio 2022

Trucchi Python per la data science

introduzione

Grazie alla sua semplicità e facilità di apprendimento , Python è diventato molto popolare in questi giorni. Viene utilizzato per varie attività come data science, machine learning, sviluppo web, scripting, automazione, ecc . Python non è una delle competenze più impegnative per i data scientist. La semplicità di Python è il primo di numerosi vantaggi nella scienza dei dati. Sebbene alcuni data scientist abbiano un background in informatica o conoscano altri linguaggi di programmazione, moltiI data scientist provengono da statistica, matematica o altre discipline tecniche e potrebbero non avere la stessa conoscenza di programmazione quando entrano nell'industria . La sintassi di Python è facile da capire e da scrivere, il che lo rende un linguaggio di programmazione veloce e facile da imparare. In questo articolo, abbiamo raccolto più di **40 funzioni e metodi che possono aiutarti ad accelerare le tue attività di data science .**

Bene, iniziamo.....

Gestire Liste e dizionari con Python ?

Gli elementi di una lista possono essere ripetuti in modo abbastanza esteso in una singola riga. Mettiamolo in pratica usando il seguente esempio:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8]

even_numbers = [number for number in numbers if number % 2 == 0]

print(even_numbers)
```

Output: [1,3,5,7]

La stessa cosa può essere fatta usando dizionari, set e generatori. Diamo un'occhiata a un altro esempio, questa volta con i dizionari:

```
dictionary = {'first_num': 1, 'second_num': 2,
              'third_num': 3, 'fourth_num': 4}
oddvalues = {key: value for (key, value) in dictionary.items() if value % 2 != 0}
print(oddvalues)Output: {'first_num': 1, 'third_num': 3}
```

Come utilizzare la funzione enumerate() con Python ?

Enumerate è una funzione utile per scorrere un oggetto come un elenco, un dizionario o un file. La funzione produce una tupla che include i valori acquisiti dall'iterazione dell'oggetto e il contatore di loop (dalla posizione iniziale di 0). Quando si desidera scrivere codice in base all'indice, il contatore di loop è utile. Diamo un'occhiata a un esempio in cui il primo e l'ultimo elemento potrebbero essere trattati in modo diverso.

```
sentence = 'Just do It'
length = len(sentence)
for index, element in enumerate(sentence):
    print('{}: {}'.format(index, element))
    if index == 0:
        print('The first element!')
    elif index == length - 1:
        print('The last element!')
```

> **Output:** 0: J The first element! 1: u 2: s 3: t 4: 5: d 6: o 7: 8: I 9: t The last element!

I file possono anche essere enumerati con la funzione enumera. Prima di uscire dal ciclo, stamperemo le prime 10 righe del file CSV nell'esempio seguente. Non replicheremo il risultato perché è troppo lungo. Puoi, tuttavia, usarlo su qualsiasi file tu abbia.

```
with open('heart.csv') as f:
    for i, line in enumerate(f):
        if i == 10:
            break
        print(line)
```

Come Restituire più valori da una funzione con Python ?

Spesso desideriamo restituire più di un valore durante la progettazione di funzioni. Analizzeremo due approcci tipici qui:

Metodo 1:

Iniziamo con l'opzione più semplice: restituire una tupla. Questa tecnica viene spesso utilizzata solo quando sono presenti due o tre valori da restituire. Quando ci sono più valori in una tupla, è semplice perdere traccia dell'ordine degli elementi. Nella sezione del codice sotto get_student c'è una funzione di esempio che restituisce il nome e il cognome del dipendente come tuple in base ai loro numeri ID.

```
# returning a tuple.
def get_student(id_num):
    if id_num == 0:
        return 'Taha', 'Nate'
    elif id_num == 1:
        return 'Jakub', 'Abdal'
    else:
        raise Exception('No Student with this id: {}'.format(id_num))
```

Quando chiamiamo la funzione get_student con il numero 0, notiamo che restituisce una tupla con due valori: 'Taha' e 'Nate'

```
Student = get_student(0)
print('first_name: {}, last_name: {}'.format(Student[0], Student[1]))
```

> **Output:** first_name: Taha, last_name: Nate

Metodo 2:

Restituire un dizionario è la seconda opzione. Poiché i dizionari sono coppie di valori chiave, possiamo nominare i valori che vengono restituiti, il che è più intuitivo delle tuple. Il metodo 2 è lo stesso del metodo 1, restituisce solo un dizionario.

```
# returning a dictionary
def get_data(id_num):
    if id_num == 0:
        return {'first_name': 'Muhammad', 'last_name': 'Taha', 'title': 'Data
Scientist', 'department': 'A', 'date_joined': '20200807'}
    elif id_num == 1:
        return {'first_name': 'Ryan', 'last_name': 'Gosling', 'title': 'Data
Engineer', 'department': 'B', 'date_joined': '20200809'}
    else:
        raise Exception('No employee with this id: {}'.format(id_num))
```

È più facile fare riferimento a un valore specifico tramite la sua chiave quando un risultato è un dizionario. Stiamo chiamando la funzione con id_num = 0

```
employee = get_data(0)
print('first_name: {},nlast_name: {},ntitle: {},ndepartment: {},ndate_joined:
{}'.format(
    employee['first_name'], employee['last_name'], employee['title'],
employee['department'], employee['date_joined']))
```

> **Output:** first_name: Muhammad, last_name: Taha, title: Data Scientist, department: A, date_joined: 2020-08-07

Come Confrontare più numeri con Python ?

Se hai un valore e desideri confrontarlo con altri due valori, puoi utilizzare la seguente **espressione matematica di base: *1<x<30***

Questo è il tipo di espressione algebrica che impariamo alle elementari. Tuttavia, l'istruzione identica può essere utilizzata anche in Python. Sì, avete letto bene. Fino ad ora, presumibilmente hai eseguito confronti in questo formato:

1<x e x<30

In Python, tutto ciò che devi fare è usare quanto segue: *1<x<30*

```
x = 5
print(1<x<30)
```

> **Uscita:** true

For-Else Method con Python ?

Questo metodo viene utilizzato per applicare il ciclo su un elenco. In generale, quando vuoi scorrere un elenco che applichi, un ciclo for. Ma con questo metodo, puoi passare un'altra condizione in un ciclo, il che è estremamente raro. Altri linguaggi di programmazione non supportano questo metodo. Diamo un'occhiata a come funziona in generale. Se vuoi controllare se c'è un numero pari in una lista.

```
number_List = [1, 3, 7, 9,8]

for number in number_List:

if number % 2 == 0:

print(number)

break

else:

print("No even numbers!!")
```

Output: 8

Se viene trovato un numero pari, il numero verrà stampato e la parte else non verrà eseguita poiché passiamo un'istruzione break. Se l'istruzione break non viene mai eseguita, verrà eseguito il blocco else.

Come trovare l'elemento più grande o più piccolo da un elenco con Python ?

Usando il modulo "heapq" puoi trovare l'elemento n-più grande o n-più piccolo da un elenco. Vediamo un esempio:

```
import heapq
numbers = [80, 25, 68, 77, 95, 88, 30, 55, 40, 50]
print(heapq.nlargest(5, numbers))
print(heapq.nsmallest(5, numbers))
```

> Output: [95, 88, 80, 77, 68] [25, 30, 40, 50, 55]

Come passare tutti i valori di un elenco come argomento di funzioni?

È possibile accedere a tutti gli elementi di un elenco utilizzando un "*"

```
def Summation(*arg):
    sum = 0
    for i in arg:
        sum += i
    return sum
result = Summation(*[8,5,10,7])
print(result)
```

> Output: 30

Come ripetere un'intera stringa senza eseguire il loop con Python ?

Basta moltiplicare la stringa per un numero, il numero di volte in cui vuoi che la stringa venga ripetuta. Allora il tuo lavoro è fatto.

```
value = "Taha"
print(value * 5)
print("-" *21)
Output: TahaTahaTahaTahaTaha
```

Come trovare l'indice di un elemento da una lista con Python ?

Utilizzare ".index" per trovare l'indice di un elemento da un elenco

```
cities= ['Vienna', 'Amsterdam', 'Paris', 'Berlin']
print(cities.index('Berlin'))
```

> Output: 3

Come stampare più elementi sulla stessa linea con Python ?

```
print("Analytics", end="")
print("Vidhya")
print("Analytics", end=" ")
print("Vidhya")
print('Data', 'science', 'blogathon', '12', sep=', ')
```

> Output: AnalyticsVidhya Analytics Vidhya Data, science, logathon, 12

Qual è la differenza tra "is" e "==" in Python ?

Se vuoi controllare se due variabili puntano allo stesso oggetto, devi usare "is"

Ma se vuoi controllare se due variabili sono uguali o meno, devi usare "==".

```
list1 = [7, 9, 4]
list2 = [7, 9, 4]
print(list1 == list2)
print(list1 is list2)
list3 = list1
print(list3 is list1)
```

> Output: True False True

La prima affermazione è True, perché list1 e list2 hanno entrambi gli stessi valori, quindi sono uguali. La seconda istruzione è False perché i valori puntano a variabili diverse nella memoria e la terza istruzione è True perché list1 e list3 puntano entrambi a un oggetto comune in memoria.

Come unire 2 dizionari in una singola riga di codice con Python ?

```
first_dct = {"London": 1, "Paris": 2}
second_dct = {"Tokyo": 3, "Seol": 4}
merged = {**first_dct, **second_dct}
print(merged)
```

> Output: {‘London’: 1, ‘Paris’: 2, ‘Tokyo’: 3, ‘Seol’: 4}

Come identificare se una stringa inizia con un carattere specifico o meno con Python ?

Se hai bisogno di sapere se una stringa inizia con un alfabeto specifico, puoi utilizzare il metodo di indicizzazione che è comune. Ma puoi anche usare una funzione chiamata "*inizia* con", ti dirà se una stringa inizia con una parola specifica o meno, che passi alla funzione.

```
sentence = "Analytics Vidhya"
print(sentence.startswith("b"))
print(sentence.startswith("A"))
```

> Output: False True

Come ottenere l'Unicode di un carattere con Python ?

Se hai bisogno di conoscere l'Unicode di un carattere, devi usare una funzione chiamata *"ord"* e passare il carattere nella funzione, di cui vuoi conoscere l'Unicode. Vediamo un esempio:

```
print(ord("T"))
print(ord("A"))
print(ord("h"))
print(ord("a"))
```

> Uscita: 84 65 104 97

Come ottenere la coppia chiave-valore di un dizionario con Python ?

Se vuoi accedere alla chiave e al valore di un dizionario in modo diverso, puoi farlo usando una funzione chiamata *"items()"* .

```
cities = {'London': 1, 'Paris': 2, 'Tokyo': 3, 'Seol': 4}
for key, value in cities.items():
    print(f"Key: {key} and Value: {value}")
```

> Output: Key: London and Value: 1 Key: Paris and Value: 2 Key: Tokyo and Value: 3 Key: Seol and Value: 4

Come i valori booleani possono essere utilizzati nelle operazioni matematiche con Python ?

False è considerato 0 e **True** è considerato 1

```
x = 9
y = 3
outcome = (x - False)/(y * True)
print(outcome)
```

> Output: 3.0

Come aggiungere valore in una posizione specifica di una lista con Python ?

Se vuoi aggiungere un valore a una lista usando la funzione *“append”* , ma aggiungerà un valore nell'ultima posizione di una lista. Quindi, cosa succede se si desidera aggiungere valore in una posizione specifica di un elenco. Puoi anche farlo, puoi usare una funzione chiamata *"inserisci"* per inserire un valore in una posizione specifica di un elenco.

Sintassi:

list_name.insert(posizione, valore)

Vediamo un esempio.

```
cities = ["London", "Vienna", "Rome"]
cities.append("Seoul")
print("After append:", cities)
cities.insert(0, "Berlin")
print("After insert:", cities)
```

> Output: After append: [‘London’, ‘Vienna’, ‘Rome’, ‘Seoul’] After insert: [‘Berlin’, ‘London’, ‘Vienna’, ‘Rome’, ‘Seoul’]

Funzione Filter() con Python ?

Il funzionamento della funzione filtro risiede nel suo nome. Filtra un iteratore specifico in base a una funzione specifica passata al suo interno. Restituisce un iteratore.

Sintassi:

filtro (funzione, iteratore)

Vediamo un esempio con la funzione filtro:

```
mixed_number = [8, 15, 25, 30,34,67,90,5,12]
filtered_value = filter(lambda x: x > 20, mixed_number)
print(f"Before filter: {mixed_number}")
print(f"After filter: {list(filtered_value)}")
```

> Output: Before filter: [8, 15, 25, 30, 34, 67, 90, 5, 12] After filter: [25, 30, 34, 67, 90]

Come creare una funzione senza limite di parametri con Python ?

Puoi creare una funzione senza preoccuparti dei parametri. È possibile passare qualsiasi numero di parametri desiderati quando si chiama la funzione. Vediamo un esempio:

```
def multiplication(*arguments):
    mul = 1
    for i in arguments:
        mul = mul * i
    return mul
print(multiplication(3, 4, 5))
print(multiplication(5, 8, 10, 3))
print(multiplication(8, 6, 15, 20, 5))
```

> Output: 60 1200 72000

Come eseguire l'iterazione su due o più elenchi contemporaneamente con Python ?

Puoi scorrere un singolo elenco usando la funzione enumera, ma quando hai due o più elenchi, puoi anche scorrere su di essi usando la funzione *"zip()"* .

```
capital = ['Vienna', 'Paris', 'Seoul',"Rome"]
countries = ['Austria', 'France', 'South Korea',"Italy"]
for cap, country in zip(capital, countries):
    print(f"{cap} is the capital of {country}")
```

> Output: Vienna is the capital of Austria Paris is the capital of France Seoul is the capital of South Korea Amsterdam is the capital of Italy

---

Come modificare il "case" delle lettere in una frase con Python ?

Se vuoi cambiare le maiuscole delle lettere, cioè maiuscole in minuscole e minuscole in maiuscole, puoi farlo usando una funzione chiamata *"scambia maiuscole"*. Vediamo un esempio:

```
sentence = "Data Science Blogathon."
changed_sen = sentence.swapcase()
print(changed_sen)
```

> Output: dATA sCIENCE bLOGATHON.

Come controllare la dimensione della memoria utilizzata da un oggetto con Python ?

Per controllare la memoria utilizzata da un oggetto importa prima la libreria *sys* poi usa un metodo di questa libreria chiamato " *getsizeof* ". Restituirà la dimensione della memoria utilizzata dall'oggetto.

```
import sys
mul = 5*6
print(sys.getsizeof(mul))
```

> Output: 28

Funzione Map() con Python ?:

La funzione *map()* viene utilizzata per applicare una funzione specifica a un determinato iteratore.

Sintassi:

mappa(funzione, iteratore)

```
values_list = [8, 10, 6, 50]
quotient = map(lambda x: x/2, values_list)
print(f"Before division: {values_list}")
print(f"After division: {list(quotient)}")
```

> Output: Before division: [8, 10, 6, 50] After division: [4.0, 5.0, 3.0, 25.0]

Come invertire un'intera stringa con Python ?

Per invertire una stringa puoi usare il metodo di slicing. Vediamo un esempio:

```
value = "Analytics"
print("Reverse is:", value[::-1])
```

> Output: Reverse is: scitylanA

Come scoprire il tempo di esecuzione di un blocco con Python ?

Quando si addestra il modello di machine learning o deep learning o si esegue semplicemente un blocco di codice, è possibile verificare quanto tempo è stato necessario per eseguire il blocco di codice. Devi usare una funzione magica "%%time" nella parte superiore del blocco del tuo codice. Ti mostrerà la quantità di tempo necessaria per eseguire il blocco di codice. Vediamo un esempio:

```
%%time
sentence = "Data Science Blogathon."
changed_sen = sentence.swapcase()
print(changed_sen)
```

> Output: dATA sCIENCE bLOGATHON. Wall time: 998 µs

Come rilasciare il carattere sinistro o destro di una stringa con Python ?

Ci sono due funzioni chiamate *“rstrip()”* e *“lstrip()”* , *“rstrip”* è usato per eliminare alcuni caratteri dalla destra di una stringa e *“lstrip”* è usato per eliminare alcuni caratteri dalla sinistra di una stringa. Il valore predefinito di entrambe le funzioni è uno spazio vuoto. Ma puoi passare il tuo carattere specifico per rimuoverli dalla stringa.

```
sentence1 = "Data Science Blogathon       "
print(f"After removing the right space: {sentence1.rstrip()}")
sentence2 = "        Data Science Blogathon"
print(f"After removing the left space: {sentence2.lstrip()}")
sentence3 = "Data Science Blogathon .,bbblllg"
print("After applying rstrip:", sentence3.rstrip(".,blg"))
```

> Output: After removing the right space: Data Science Blogathon After removing the left space: Data Science Blogathon After applying rstrip: Data Science Blogathon

---

Come contare il numero di volte in cui un elemento appare in un elenco con Python ?

Puoi contare il numero di volte in cui un elemento appare in un elenco eseguendo un ciclo for tra di essi. Ma puoi farlo più facilmente, semplicemente chiamando un metodo nell'elenco chiamato *"count"* . Ecco un esempio:

```
cities= ["Amsterdam", "Berlin", "New York", "Seoul", "Tokyo", "Paris",
"Paris","Vienna","Paris"]
print("Paris appears", cities.count("Paris"), "times in the list")
```

> Output: Paris appears 3 times in the list

Come trovare l'indice di un elemento in una tupla o in una lista con Python ?

Puoi trovare l'indice di un elemento in una tupla o in una lista semplicemente chiamando un semplice metodo chiamato "index" su quella tupla o lista. Ecco un esempio:

```
cities_tuple = ("Berlin", "Paris", 5, "Vienna", 10)
print(cities_tuple.index("Paris"))
cities_list = ['Vienna', 'Paris', 'Seoul',"Amsterdam"]
print(cities_list.index("Amsterdam"))
```

> Output: 1 3

Come rimuovere ogni elemento da un elenco o da un set con Python ?

Puoi rimuovere tutti gli elementi da un elenco o da un insieme applicando un metodo chiamato "cancella" su quell'elenco o insieme.

```
cities_list = ['Vienna', 'Paris', 'Seoul',"Amsterdam"]
print(f"Before removing from the list: {cities_list}")
cities_list.clear()
print(f"After removing from the list: {cities_list}")
cities_set = {'Vienna', 'Paris', 'Seoul',"Amsterdam"}
print(f"Before removing from the set: {cities_set}")
cities_set.clear()
print(f"After removing from the set: {cities_set}")
```

> Output: Before removing from the list: [‘Vienna’, ‘Paris’, ‘Seoul’, ‘Amsterdam’] After removing from the list: [] Before removing from the set: {‘Vienna’, ‘Amsterdam’, ‘Seoul’, ‘Paris’}

Come unire 2 set di dati con Python ?

Per unire due insiemi puoi applicare il metodo chiamato “union()”. Si unirà alle due liste su cui hai applicato il metodo.

```
set1 = {'Vienna', 'Paris', 'Seoul'}
set2 = {"Tokyo", "Rome",'Amsterdam'}
print(set1.union(set2))
```

> Output: {‘Vienna’, ‘Tokyo’, ‘Seoul’, ‘Amsterdam’, ‘Rome’, ‘Paris’}

Come ordinare i valori di un elenco in base alla loro frequenza con Python ?

Innanzitutto, usa "counter" dal modulo chiamato collections per misurare la frequenza di ciascun valore, quindi applica un metodo chiamato "most_common" sul risultato del contatore per ordinare i valori dell'elenco in base alla loro frequenza.

```
from collections import Counter
count = Counter([7, 6, 5, 6, 8, 6, 6, 6])
print(count)
print("Sort values according their frequency:", count.most_common())
```

> Output: Counter({6: 5, 7: 1, 5: 1, 8: 1}) Sort values according their frequency: [(6, 5), (7, 1), (5, 1), (8, 1)]

Come eliminare i valori duplicati da un elenco con Python ?

Innanzitutto, converti l'elenco in un set, questo rimuoverà i valori duplicati perché un set non contiene valori duplicati. Quindi converti nuovamente il set in un elenco, in questo modo puoi facilmente eliminare i valori duplicati da un elenco.

```
cities_list = ['Vienna', 'Paris', 'Seoul',"Amsterdam","Paris","Amsterdam","Paris"]
cities_list = set(cities_list)
print("After removing the duplicate values from the list:",list(cities_list))
```

> Output: After removing the duplicate values from the list: [‘Vienna’, ‘Amsterdam’, ‘Seoul’, ‘Paris’]

Come fare una singola frase da tutti gli elementi di una lista con Python ?

Utilizzando un metodo chiamato “join” puoi unire tutti i singoli elementi di una lista e creare una singola stringa o frase.

```
words_list = ["Data", "science", "Blogathon"]
print(" ".join(words_list))
```

> Output: Data science Blogathon

Come restituire più valori da una funzione contemporaneamente con Python ?

Sì, puoi farlo in Python. È possibile restituire più valori da una funzione contemporaneamente. Vediamo un esempio:

```
def calculation(number):
    mul = number*2
    div = number/2
    summation = number+2
    subtract = number-2
    return mul, div, summation, subtract
mul, div, summation, subtract = calculation(10)
print("Multiplication:", mul)
print("Division:", div)
print("Summation:", summation)
print("Subtraction:", subtract)
```

> Output: Multiplication: 20 Division: 5.0 Summation: 12 Subtraction: 8

Come scoprire la differenza tra due liste con Python ?

Per prima cosa, converti le liste in insiemi, quindi applica il metodo chiamato "symmetric_difference" su questi insiemi. Ciò restituirà la differenza tra questi due elenchi.

```
cities_list1 = ['Vienna', 'Paris', 'Seoul',"Amsterdam", "Berlin", "London"]
cities_list2 = ['Vienna', 'Paris', 'Seoul',"Amsterdam"]
cities_set1 = set(cities_list1)
cities_set2 = set(cities_list2)
difference = list(cities_set1.symmetric_difference(cities_set2))
print(difference)
```

> Output: [‘Berlin’, ‘London’]

Come unire due elenchi diversi in un unico dizionario con Python ?

Innanzitutto, applica una funzione zip su questi due elenchi, quindi converti l'output della funzione zip in un dizionario. Il tuo lavoro è finito, è così facile convertire due elenchi in un unico dizionario.

```
number = [1, 2, 3]
cities = ['Vienna', 'Paris', 'Seoul']
result = dict(zip(number, cities))
print(result)
```

> Output: {1: ‘Vienna’, 2: ‘Paris’, 3: ‘Seoul’}

Come ottenere gli n elementi più grandi o gli n più piccoli di un elenco con Python ?

Prima importa il modulo "heap q" quindi applica il metodo "n più grande" e "n più piccolo" e passa il valore di n e il nome della lista, in questo modo puoi ottenere gli n elementi più grandi e gli n più piccoli di una lista.

```
import heapq
numbers = [100, 20, 8, 90, 86, 95, 9, 66, 28, 88]
print(heapq.nlargest(3, numbers))
print(heapq.nsmallest(3, numbers))
```

> Output: [100, 95, 90] [8, 9, 20]

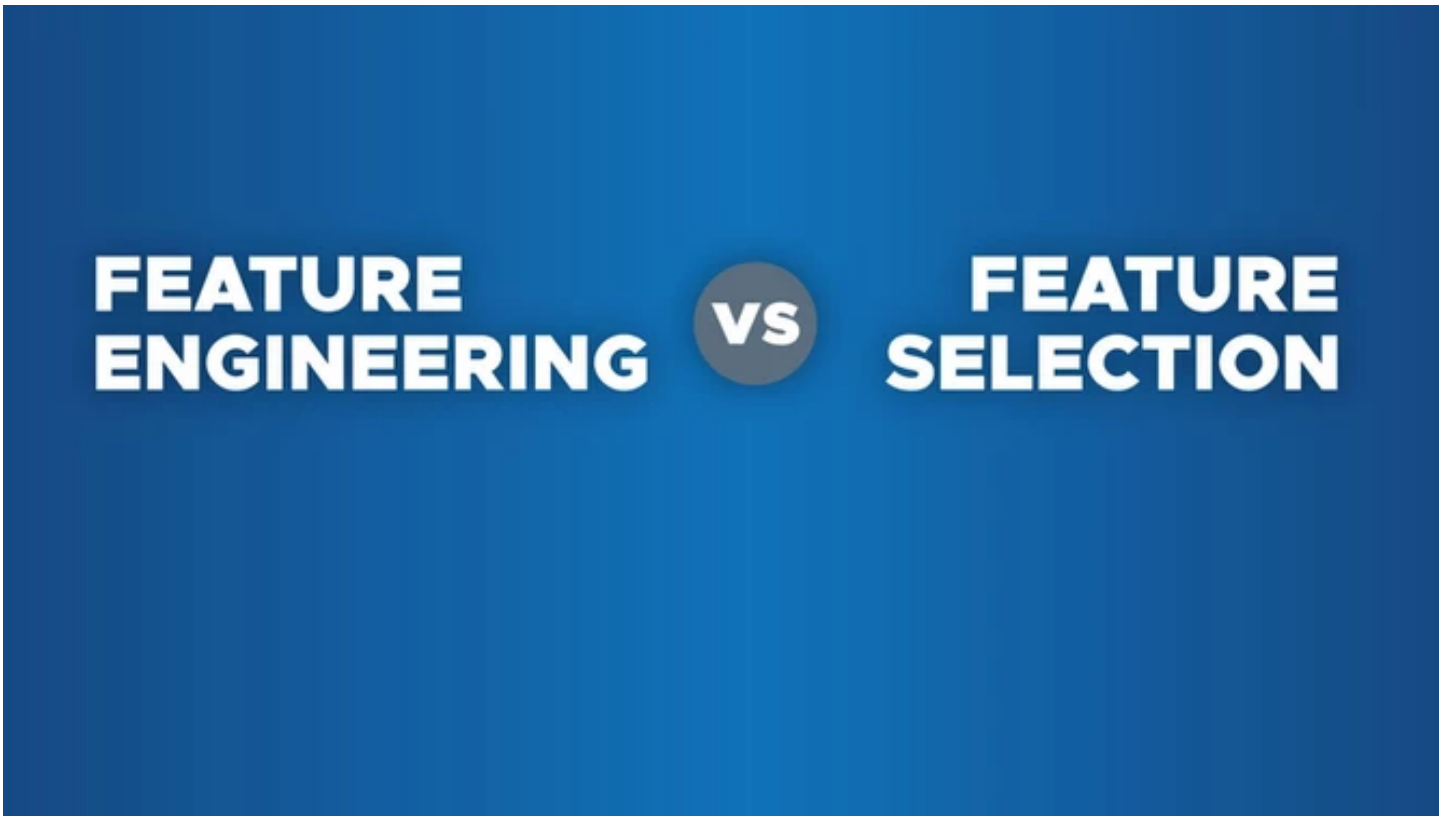
Team I.A. Italia Tempo di lettura: 9 min

Feature Engineering e Feature Selection per Principianti e non – Python e Machine Learning Tutorial

Dicono che i **dati** siano il nuovo **petrolio** , ma non usiamo il petrolio direttamente dalla sua fonte.

Deve essere elaborato e pulito prima di utilizzarlo per scopi diversi.

Lo stesso vale per i dati, non li usiamo direttamente dalla loro fonte. Deve anche essere elaborato.



Feature Engineering e Feature Selection per Principianti e non – Python e Machine Learning Tutorial

Questa può essere una sfida per i principianti nell'apprendimento automatico e nella scienza dei dati perché i dati provengono da fonti diverse con tipi di dati diversi. Pertanto non è possibile applicare lo stesso metodo di pulizia ed elaborazione a diversi tipi di dati.

e informazioni possono essere estratte dai dati così come

nergia può essere estratta dal petrolio."- Adeola Adesina

Devi imparare e applicare metodi a seconda dei dati che hai.
Dopo aver letto questo articolo, saprai:

- **Che cos'è l'ingegneria delle funzionalità e la selezione delle funzionalità.**
- **Diversi metodi per gestire i dati mancanti nel tuo set di dati.**
- **Diversi metodi per gestire le funzionalità continue.**
- **Diversi metodi per gestire le caratteristiche categoriche.**
- **Diversi metodi per la selezione delle caratteristiche.**

Cominciamo.🚀

Che cos'è il Feature Engineering l'ingegneria delle Feature?

L'ingegneria delle feature si riferisce a un processo di selezione e **trasformazione di** variabili/funzioni nel set di dati durante la creazione di un **modello predittivo** utilizzando l'apprendimento automatico.
Pertanto devi estrarre le funzionalità dal **set** di **dati** non **elaborato** che hai raccolto prima di addestrare i tuoi dati negli algoritmi di apprendimento automatico.
Altrimenti, sarà difficile ottenere buone informazioni sui tuoi dati.

rtura i dati e confesseranno qualsiasi cosa. — Ronald Coase

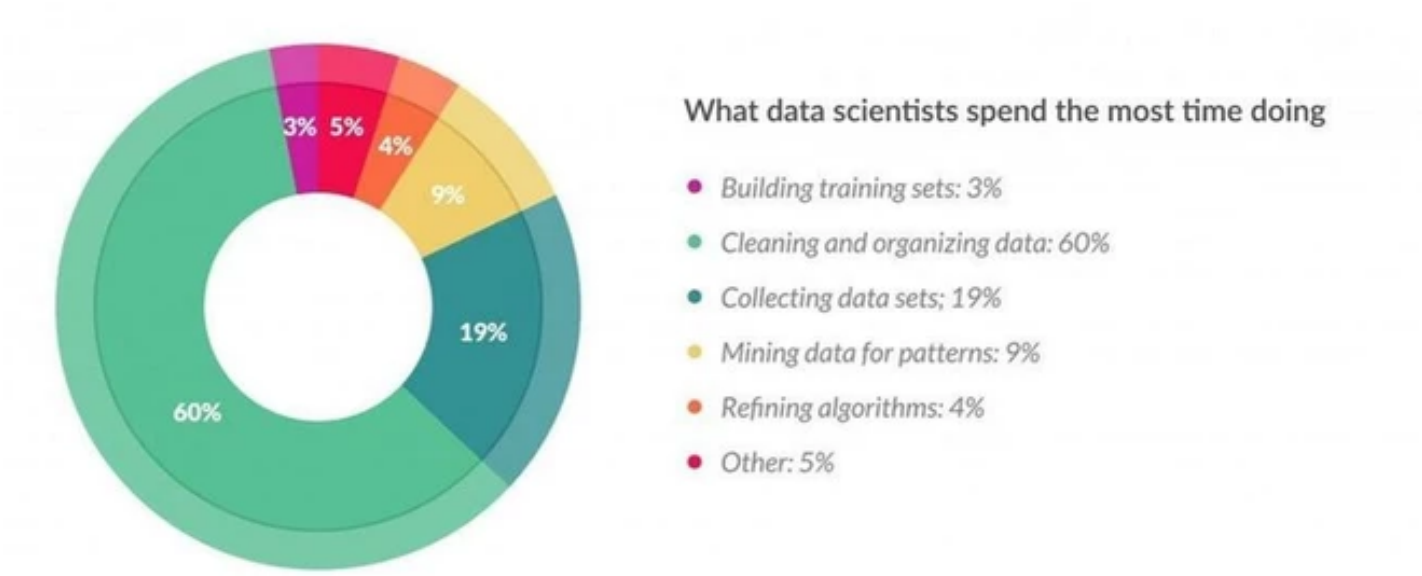
L'ingegneria delle feature ha due obiettivi:

- Preparazione del set di dati di input corretto, compatibile con i requisiti dell'algoritmo di apprendimento automatico.

- Migliorare le **prestazioni** dei modelli di machine learning.

Indagine CrowdFlower

Secondo un sondaggio condotto da CrowdFlower su 80 Data Scientist, i Data Scientist passano il **60%** del loro tempo a pulire e organizzare i dati. Questo è il motivo per cui avere competenze nell'ingegneria e nella selezione delle funzionalità è molto importante.



i Data Scientist passano il 60% del loro tempo a pulire e organizzare i dati

Alla fine della giornata, alcuni progetti di machine learning hanno successo e altri falliscono. Cosa fa la differenza? Facilmente il fattore più importante sono le ***feature*** *utilizzate. " — Prof. Pedro Domingos dell'Università di Washington*

Ora che sai perché è necessario apprendere tecniche diverse per la progettazione delle funzionalità, iniziamo imparando metodi diversi per gestire i dati mancanti.

Come gestire i dati mancanti

La gestione dei dati mancanti è molto importante poiché molti algoritmi di apprendimento automatico non supportano i dati con valori mancanti. La mancanza di valori nel set di dati può causare errori e prestazioni scadenti con alcuni algoritmi di apprendimento automatico.

Ecco l'elenco dei valori mancanti comuni che puoi trovare nel tuo set di dati.

- N / A
- nullo
- Vuoto
- ?
- nessuno
- sporco
- -
- NaN

Impariamo diversi metodi per risolvere il problema dei dati mancanti.

Cancellazione variabile

L'eliminazione delle variabili comporta l'eliminazione delle variabili (colonne) con valori mancanti caso per caso. Questo metodo ha senso quando ci sono molti valori mancanti in una variabile e se la variabile è di importanza relativamente minore.

L'unico caso in cui può valere la pena eliminare una variabile è quando i suoi valori mancanti superano il **60%** delle osservazioni.

```
importa packages
port numpy as np
port pandas as pd

leggi dataset
ta = pd.read_csv('path/to/data')

etta threshold
reshold = 0.7

droppa colonne con i valori mancani più alti del threshold
ta = data[data.columns[data.isnull().mean() < threshold]]
```

Nello snippet di codice sopra, puoi vedere come utilizzo NumPy e panda per caricare il set di dati e impostare una soglia su **0.7** . Ciò significa che qualsiasi colonna con valori mancanti superiori al **70%** delle osservazioni verrà eliminata dal set di dati.
Ti consiglio di impostare il valore di soglia in base alla dimensione del tuo set di dati.

Imputazione media o mediana

Un'altra tecnica comune consiste nell'utilizzare la media o la mediana delle osservazioni non mancanti. Questa strategia può essere applicata a una feature che contiene dati numerici.

```
Sostituisci i valori nulli con la media delle colonne di
partenenza
ta = data.fillna(data.median())
```

Nell'esempio sopra, usiamo il **metodo mediano** per riempire i valori mancanti nel set di dati.

Valore più comune

Questo metodo sostituisce i valori mancanti con il **valore massimo verificato** in una colonna/funzione. Questa è una buona opzione per la gestione di colonne/funzioni **categoriali** .

```
Sostituisci i valori nulli con la media delle colonne di
partenenza

ta['column_name'].fillna(data['column_name'].value_counts().idx
x(). inplace=True)
```

Qui utilizziamo il metodo **value_counts()** di panda per contare l'occorrenza di ciascun valore univoco nella colonna e quindi riempire il valore mancante con il valore più comune.

Come gestire le funzioni continue

Le feature continue nel set di dati hanno un intervallo di valori diverso. Esempi comuni di funzionalità continue sono età, stipendio, prezzi e altezza.

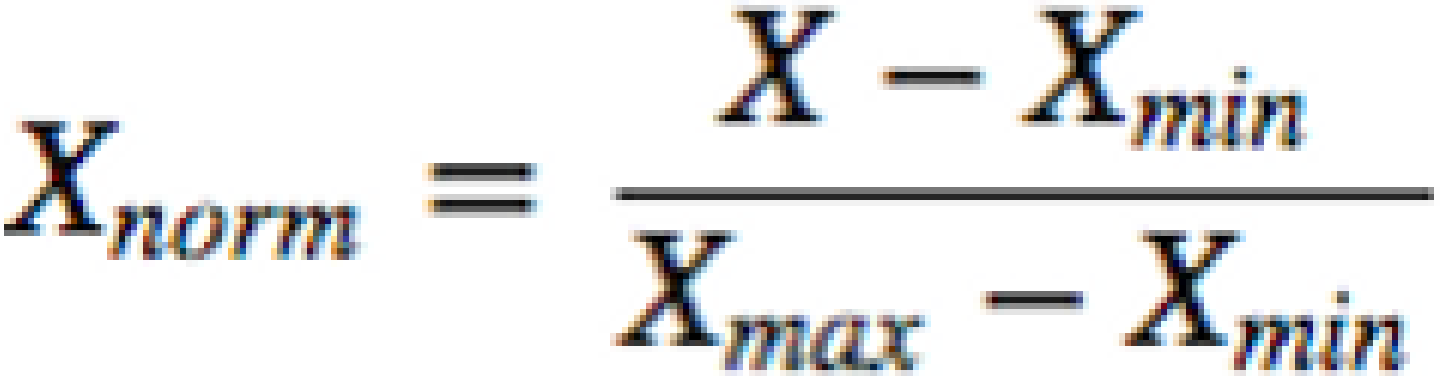
È molto importante gestire le funzionalità continue nel set di dati prima di addestrare gli algoritmi di apprendimento automatico. Se si addestra il modello con un intervallo di valori diverso, il modello non funzionerà bene.

Cosa intendo quando dico un diverso intervallo di valori? Supponiamo che tu abbia un set di dati con due funzioni continue, **età** e **stipendio** . La fascia di età sarà diversa dalla fascia di stipendio e ciò può causare problemi.

Di seguito sono riportati alcuni metodi comuni per gestire le funzionalità continue:

Normalizzazione Min-Max in Python

Per ogni valore in una caratteristica, la normalizzazione Min-Max sottrae il valore minimo nella caratteristica e quindi divide per il suo intervallo. L'intervallo è la differenza tra il massimo originale e il minimo originale.



$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Normalizzazione Min-Max

Infine, ridimensiona tutti i valori in un intervallo fisso tra 0 e 1.

È possibile utilizzare il metodo MinMaxScaler di Scikit -learn che trasforma le funzionalità ridimensionando ciascuna funzionalità in un determinato intervallo:

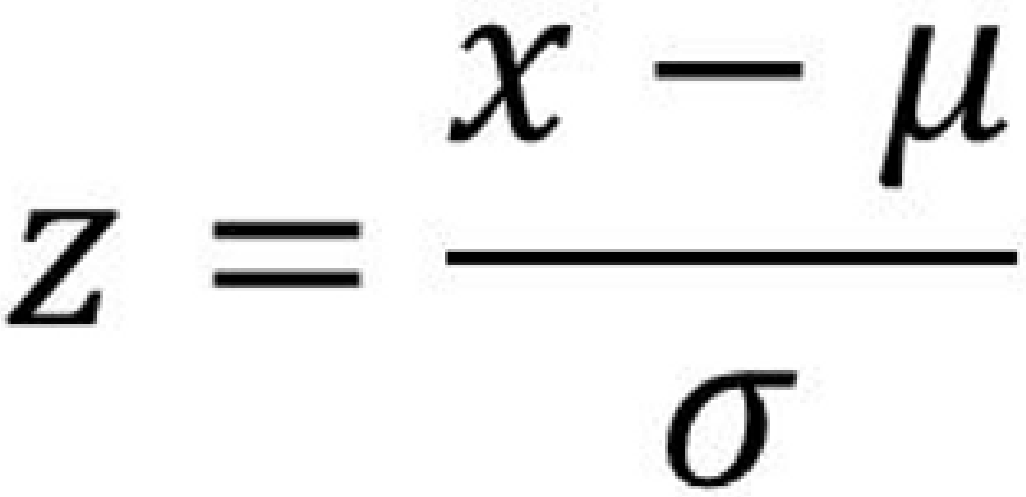
```
om sklearn.preprocessing import MinMaxScaler
port numpy as np

ta = np.array([[4, 6], [11, 34], [10, 17], [1, 5]])

creiamo lo scaler method
aler = MinMaxScaler(feature_range=(0,1))

Scaliamo e trasformiamo i dati
aled_data = scaler.fit_transform(data)

int(scaled_data)

[[0.3        0.03448276]
 [1.         1.        ]
 [0.9        0.4137931 ]
 [0.         0.        ]]
```

Come puoi vedere, i nostri dati sono stati trasformati e l'intervallo è compreso tra 0 e 1 .

Standardizzazione in Python

La Standardizzazione assicura che ogni caratteristica abbia una media di **0** e una deviazione standard di **1** , portando tutte le caratteristiche alla stessa grandezza.

Se la deviazione standard delle caratteristiche è ***diversa*** , anche il loro intervallo sarebbe diverso.



Standardizzazione

x = osservazione, μ = media, σ = deviazione standard

È possibile utilizzare il metodo **StandardScaler** di Scikit-learn per standardizzare le funzionalità rimuovendo la media e scalando a una deviazione standard di **1** :

```
om sklearn.preprocessing import StandardScaler
port numpy as np

ta = np.array([[4, 1], [11, 1], [10, 4], [1, 11]])

creiamo lo scaler method
aler = StandardScaler()

Scaliamo e trasformiamo i dati
aled_data = scaler.fit_transform(data)
```

```
int(scaled_data)
```

```
[[-0.60192927 -0.79558708]
 [ 1.08347268 -0.79558708]
 [ 0.84270097 -0.06119901]
 [-1.32424438  1.65237317]]
```

Verifichiamo che la media di ogni feature (colonna) sia **0** :

```
int(scaled_data.mean(axis=0))
```

[0. 0.]

E che la deviazione standard di ogni caratteristica (colonna) è **1** :

```
int(scaled_data.std(axis=0))
```

[1. 1.]

Come gestire le caratteristiche categoriche

Le caratteristiche categoriali rappresentano tipi di dati che possono essere suddivisi in gruppi. Ad esempio, generi e livelli di istruzione.

Tutti i valori non numerici devono essere *convertiti* in numeri interi o float per essere utilizzati nella maggior parte delle librerie di machine learning.I metodi comuni per gestire le caratteristiche categoriali sono:

Codifica etichetta

La codifica dell'etichetta converte semplicemente ogni valore categorico in una colonna in un numero.

Si consiglia di utilizzare la codifica delle etichette per convertirli in variabili binarie.

Nell'esempio seguente imparerai come utilizzare LableEncoder di Scikit -learn per trasformare i valori categoriali in binari:

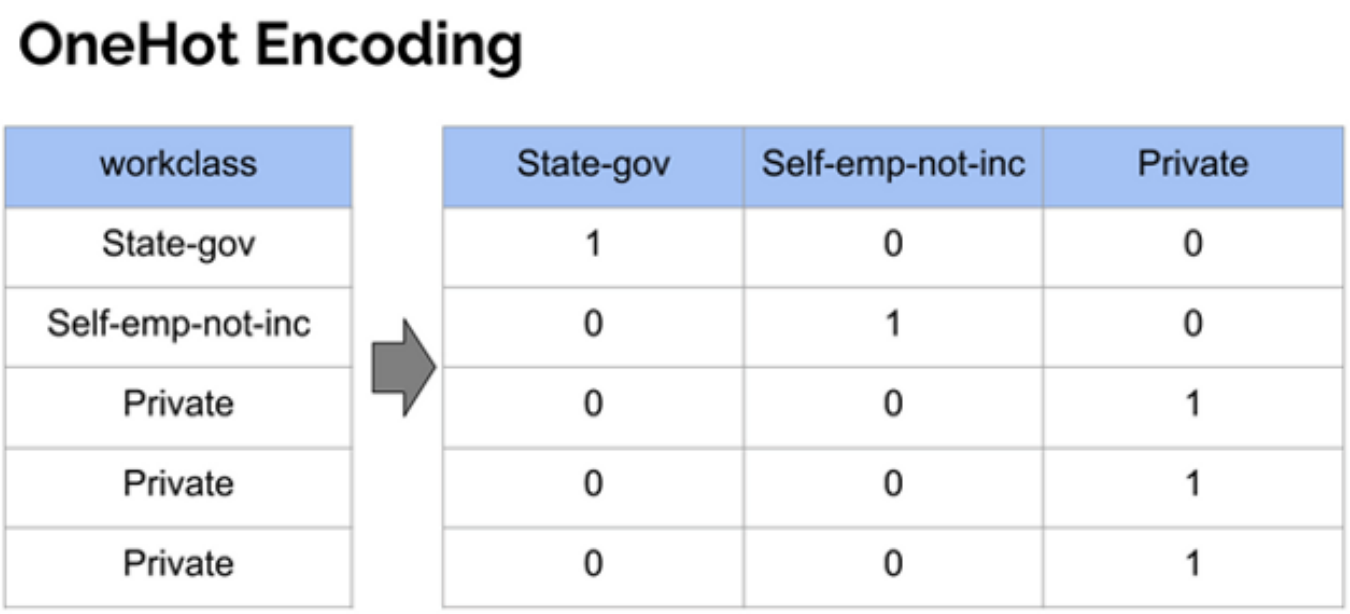
```
port numpy as np
port pandas as pd
```

```
om sklearn.preprocessing import LabelEncoder

ta = {'Gender':['male', 'female', 'female', 'male','male'],
      'Country':['Tanzania','Kenya', 'Tanzania',
anzania','Kenya']}

Creiamo il dataFrame
ta = pd.DataFrame(data)



creiamo il label encoder object
 = LabelEncoder()

ta['Gender']= le.fit_transform(data['Gender'])
ta['Country']= le.fit_transform(data['Country'])

int(data)
```

Dati trasformati

	Gender	Country
0	1	1
1	0	0
2	0	1
3	1	1
4	1	0

Codifica one-hot



Codifica one-hot

Il modo di gran lunga più comune per rappresentare le variabili categoriali è utilizzare la codifica one-hot, o metodi di codifica uno-su-N, noti anche come variabili fittizie.
L'idea alla base delle variabili fittizie è sostituire una variabile categoriale con una o più nuove funzionalità che possono avere i valori 0 e 1.

Nell'esempio seguente, utilizzeremo gli encoder della libreria Scikit-learn. LabelEncoder ci aiuterà a creare una codifica intera di etichette dai nostri dati eOneHotEncoder creerà una codifica one-hot di valori con codifica intera.

```
port numpy as np
om sklearn.preprocessing import OneHotEncoder, LabelEncoder


definiamo un esempio
ta = np.array(['cold', 'cold', 'warm', 'cold', 'hot', 'hot',
arm', 'cold', 'warm', 'hot'])

 creiamo integer encode
```

```
bel_encoder = LabelEncoder()

alleniamo e trasformia i dati
teger_encoded = label_encoder.fit_transform(data)
int(integer_encoded)

 creiamo one-hot encode
ehot_encoder = OneHotEncoder(sparse=False)

effttuiamo il reshape dei dati
teger_encoded = integer_encoded.reshape(len(integer_encoded),


alleniamo e trasformia i dati
ehot_encoded = onehot_encoder.fit_transform(integer_encoded)

int(onehot_encoded)
```

Questo è l' **output** di **integer_encoded** dal metodo **LabelEncoder** :

```
 0 2 0 1 1 2 0 2 1]
```

E questo è l'output di **onehot_encoded** dal metodo **OneHotEncoder** :

```
1. 0. 0.]
1. 0. 0.]
0. 0. 1.]
1. 0. 0.]
0. 1. 0.]
0. 1. 0.]
0. 0. 1.]
1. 0. 0.]
0. 0. 1.]
0. 1. 0.]]
```

Che cos'è la Feature Selection o selezione delle Feature ?

La selezione delle funzionalità è il processo in cui si selezionano automaticamente o manualmente le funzionalità che contribuiscono maggiormente alla variabile o all'output di previsione.

La presenza di funzionalità irrilevanti nei dati può *ridurre* l'accuratezza dei modelli di machine learning.
I motivi principali per utilizzare la selezione delle funzionalità sono:

- Consente all'algoritmo di apprendimento automatico di eseguire l'addestramento più velocemente.
- Riduce la complessità di un modello e ne facilita l'interpretazione.
- Migliora la precisione di un modello se viene scelto il sottoinsieme corretto.
- Riduce il sovradattamento.

lo preparato un modello selezionando tutte le caratteristiche e ottenuto una precisione di circa il **65%** che non è abbastanza ona per un modello predittivo e dopo aver effettuato alcune lezioni di funzionalità e progettazione delle funzionalità senza portare modifiche logiche al codice del mio modello, la mia ecisione è balzata a **81%** che è piuttosto impressionante "- Di heel Shaikh

I metodi comuni per la selezione delle funzionalità sono:

Selezione univariata

I test statistici possono aiutare a selezionare le caratteristiche indipendenti che hanno la relazione più forte con la caratteristica di destinazione nel set di dati. Ad esempio, il test del chi quadrato.

La libreria Scikit -learn fornisce la classe SelectKBest che può essere usata con una suite di diversi test statistici per selezionare un numero specifico di funzionalità.
Nell'esempio seguente utilizziamo la classe **SelectKBest** con il test chi-squired per trovare la funzionalità migliore per il set di dati Iris:

```
om sklearn.datasets import load_iris
om sklearn.feature_selection import SelectKBest
om sklearn.feature_selection import chi2

is_dataset = load_iris()

Dividiamo Target e features
= iris_dataset.data
= iris_dataset.target

= X.astype(int)

i2_features = SelectKBest(chi2, k = 2)
kbest_features = chi2_features.fit_transform(X, y)

features Ridotte
int('Original feature number:', X.shape[1])
int('Reduced feature number:', X_kbest_features.shape[1])
```

Numero di caratteristica originale: 4

Numero di caratteristica ridotto: 2

Come puoi vedere, il test del chi quadrato ci aiuta a selezionare **due importanti caratteristiche indipendenti** dalle 4 originali che hanno la relazione più forte con la caratteristica di destinazione.

Importanza delle caratteristiche

L'importanza delle funzionalità ti dà un punteggio per ogni caratteristica dei tuoi dati. Più alto è il punteggio, più **importante** o **rilevante è** quella caratteristica per la tua caratteristica target.
L'importanza delle funzionalità è una classe incorporata che viene fornita con classificatori basati su alberi come:

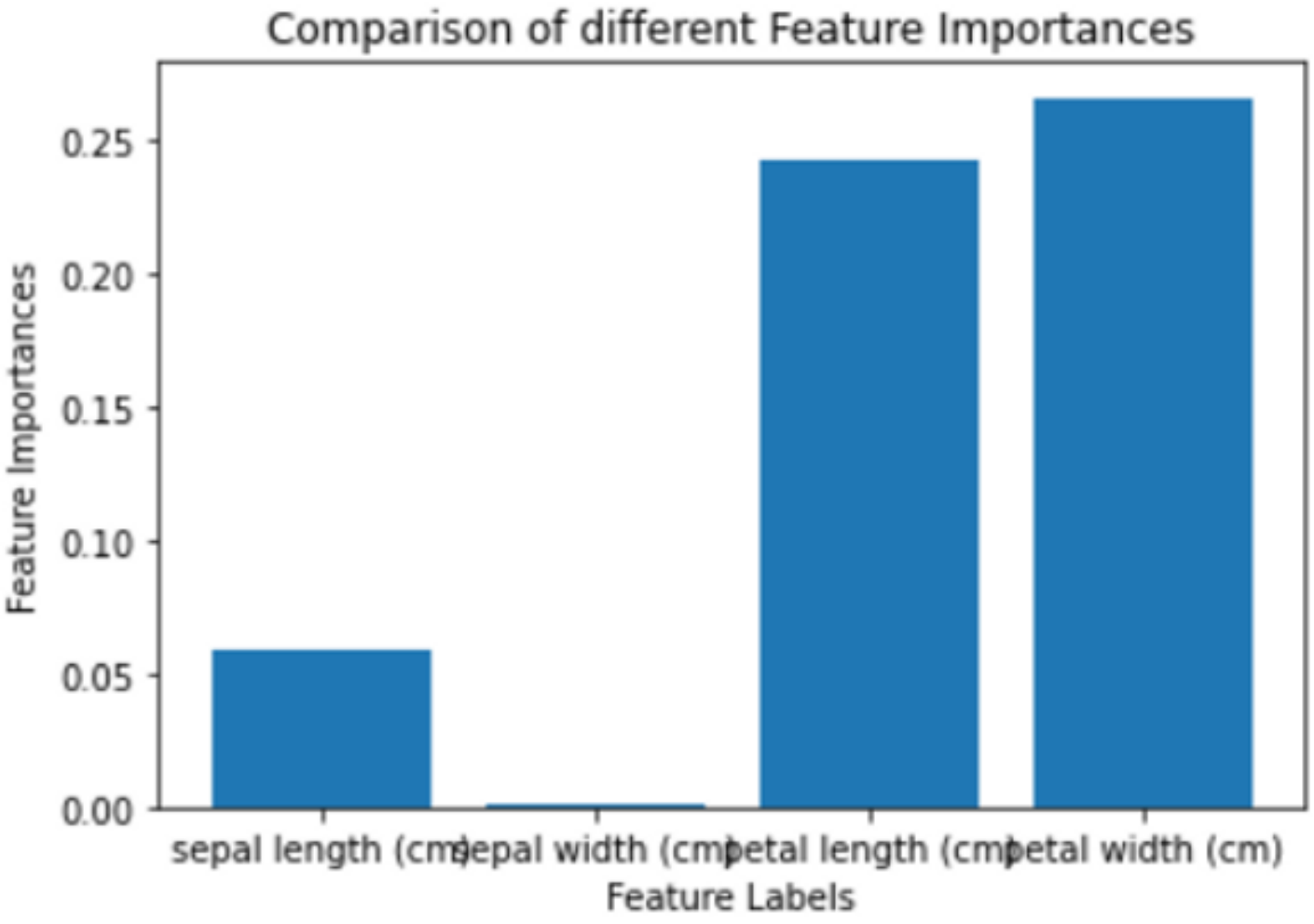
- Classificatori forestali casuali
- Classificatori albero extra

Nell'esempio seguente, addestreremo il classificatore ad albero aggiuntivo nel set di dati dell'iride e utilizzeremo la classe incorporata **.feature_importances_** per calcolare l'importanza di ogni caratteristica:

```
om sklearn.datasets import load_iris
port matplotlib.pyplot as plt
om sklearn.ensemble import ExtraTreesClassifier

is_dataset = load_iris()

Dividiamo Target e features
= iris_dataset.data
= iris_dataset.target

= X.astype(int)

 Creiamo il modello
tra_tree_forest = ExtraTreesClassifier(n_estimators = 5,
                                       criterion ='entropy',
x_features = 2)

Alleniamo il modello
tra_tree_forest.fit(X, y)

ature_importance = extra_tree_forest.feature_importances_

Normalizziamo
```

```
ature_importance_normalized = np.std([tree.feature_importances_
r tree in

tra_tree_forest.estimators_],
                                        axis = 0)

Mostriamo il grafico a barre
t.bar(iris_dataset.feature_names,
ature_importance_normalized)
t.xlabel('Feature Labels')
t.ylabel('Feature Importances')
t.title('Comparison of different Feature Importances')
t.show()
```

Caratteristiche importanti



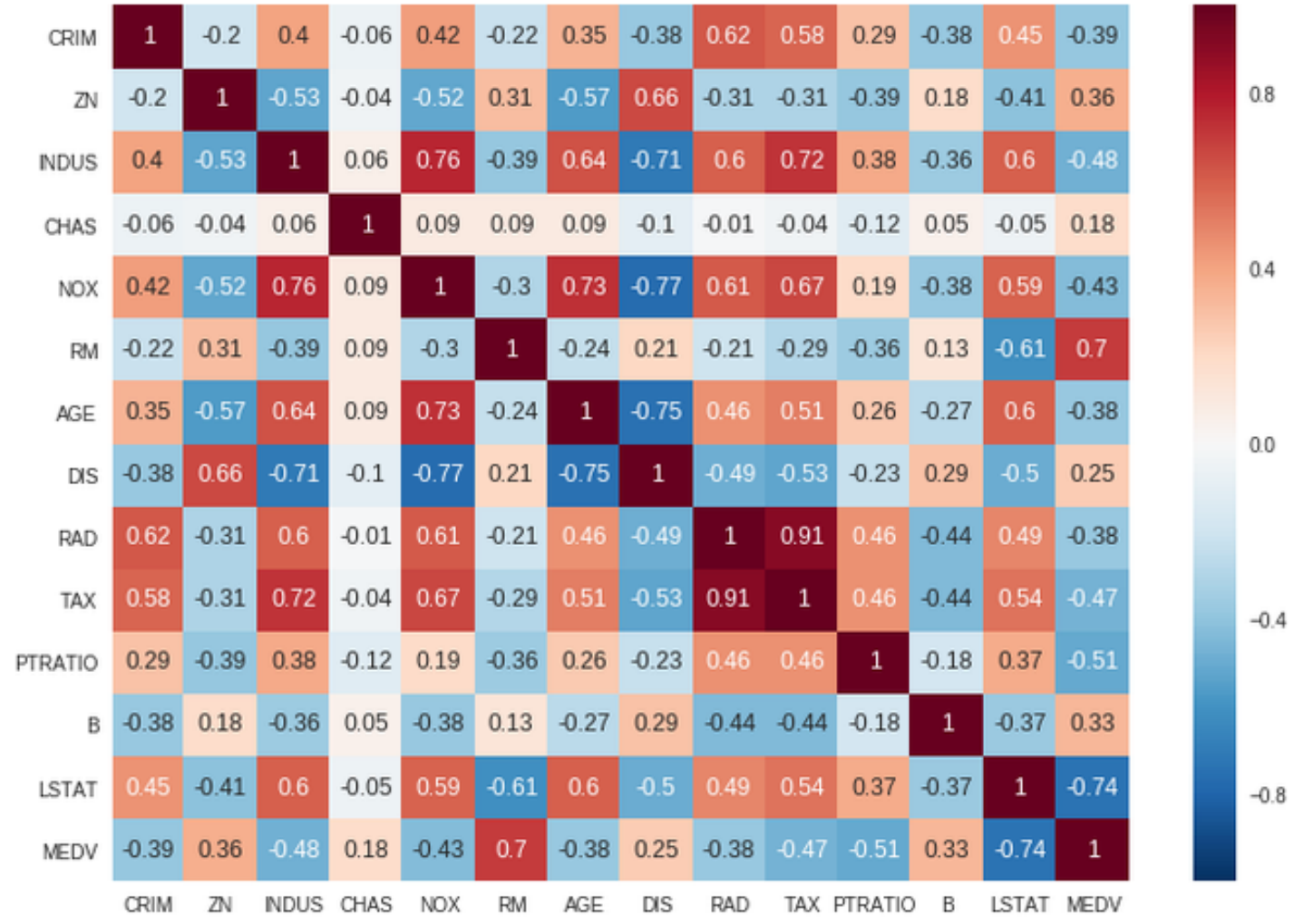
Il grafico mostra che le suddette caratteristiche più importanti sono *lunghezza petalo (cm)* e *larghezza petalo (cm)* , e che il minimo caratteristica importante è *la larghezza sepalo (cm)* . Ciò significa che puoi utilizzare le funzionalità più importanti per addestrare il tuo modello e ottenere le migliori prestazioni.

Matrice di correlazione Heatmap

La correlazione mostra come le caratteristiche sono correlate tra loro o con la caratteristica di destinazione.
La correlazione può essere positiva (un aumento di un valore della caratteristica aumenta il valore della variabile target) o negativa (un aumento di un valore della caratteristica diminuisce il valore della variabile target).

Nell'esempio seguente, utilizzeremo il dataset dei prezzi delle case di Boston dalla libreria Scikit-learn e il metodo corr() di pandas per trovare la correlazione a coppie di tutte le caratteristiche nel dataframe:

```
om sklearn.datasets import load_boston
port matplotlib.pyplot as plt
port seaborn as sns

ston_dataset = load_boston()

ston = pd.DataFrame(boston_dataset.data,
lumns=boston_dataset.feature_names)

 = X.astype(int)

ostriamo il grafico
 = sns.heatmap(boston.corr().round(2), annot=True)
```



Il coefficiente di correlazione va da -1 a 1. Se il valore è prossimo a 1, significa che c'è una forte correlazione positiva tra le due caratteristiche. Quando è vicino a -1, le caratteristiche hanno una forte correlazione negativa.

Nella figura sopra, si può vedere che **l'IVA** e **RAD** caratteristiche hanno come *TRONG correlazione positiva* e le **DIS** e **NOX** caratteristiche hanno una *forte correlazione negativa.*

Se scopri che ci sono alcune funzionalità nel tuo set di dati che sono correlate tra loro, significa che trasmettono le stesse informazioni. Si consiglia di rimuoverne uno.

Conclusione

I metodi che ho spiegato in questo articolo ti aiuteranno a preparare la maggior parte dei **set** di **dati strutturati** che hai. Ma se stai lavorando su set di dati non strutturati come immagini, testo e audio, dovrai imparare diversi metodi che non sono spiegati in questo articolo.