Python Biella Group

Pandas 05

Re-introduzione

Indexing

Condizioni Filtro

MultiIndex – Stack - Unstack

Presenta

Mario

Python Biella Group

Obiettivi della serata

- Migliorare la conoscenza di pandas
  - In particolare, Dataframes (2d)

pandas

Python Biella Group

Cos' e' pandas?

pandas: high-performance open source library per l'analisi dei dati, in Python, sviluppata inizialmente da Wes McKinney (ora "Benevolent Dictator for Life") nel 2008. Free, distribuita con licenza 3-Clause BSD
https://pandas.pydata.org/

pandas sta per **panel data (pd)**, riferimento al formato tabellare con Il quale si processano i dati

Diventata nel corso degli anni **libreria standard de-facto library per data analysis** usando Python
C'è una larga community, con sviluppi rapidi e in continua evoluzione (al 09/02/2021, v. 1.2.2)

Alcune caratteristiche fondamentali:

- **caricamento/import dei dati** da files / database in diversi formati
- **processamento di datasets in diversi formati**: time series, tabelle eterogenee, matrici di dati
- **gestisce miriadi di operazioni sui datasets**: subsetting, slicing, filtering, merging, groupBy, re-ordering, and re-shaping
- **può gestire i "dati mancanti" con regole custom definite da e user/developer**, come ad esempio ignorarli, convertirli ad un valore di default e così via
- **può essere usato per fare parsing e conversioni dei dati** così come per **modellare e analisi statistiche**
- **si integra bene con altre librerie Python** come ad esempio statsmodels, SciPy e scikit-learn
- **offre ottime performance** e può essere ulteriormente velocizzato facendo uso di **Cython** (estensione C di Python)

Python Biella Group

NumPy, pandas e R

pandas fu creato da Wes McKinney nel 2008 come risposta alla sua frustrazione nell'uso delle time-series con il linguaggio **R**

Costruita "on top" di NumPy e fornisce features non disponibili in esso

Offre strutture dati veloci, di facile comprensione e copre il gap tra Python e linguaggi specifici per l'analisi dei dati come R

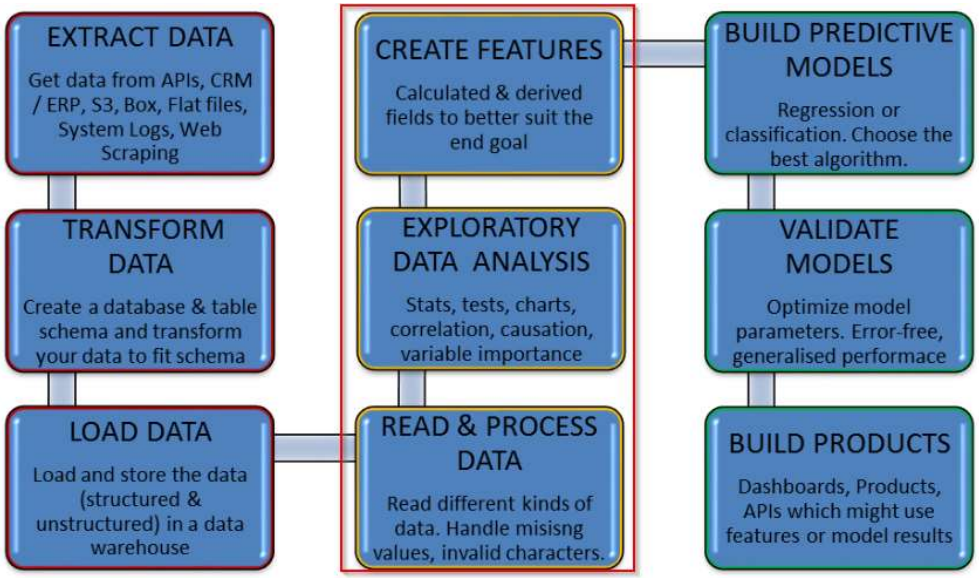
NumPy gestisce bene blocchi di dati **omogenei**.

Usare **pandas** aiuta a gestire dati, in forma tabellare, con **tipi dato differenti**

Python Biella Group

pandas

Utilizzo di pandas in Data Analytics / 1



Indispensabile per questi steps di “Data Science”:
- Lettura & processamento dei dati
- Analisi dei dati esplorativa
- Creazione “features” (colonne derivate)

Python Biella Group

Utilizzo di pandas in Data Analytics / 2

pandas è ottimale per questi tipi di dataset:
•Tabelle con colonne di tipo eterogeneo
•Time-series ordinate e non ordinate
•Matrici/array di dati con righe e colonne etichettate o non etichettate

pandas è ottimale per questi utilizzi sui dati:
•Facile gestione dei dati mancanti e NaN
•Aggiunta e cancellazione di colonne
•Allineamento automatico ed esplicito dei dati con le etichette
•GroupBy per aggregare e trasformare i dati usando la strategia dividi-applica-aggrega
•Convertire strutture Python o NumPy indicizzate in DataFrame
•Slicing, indexing, indexing gerarchico e subsetting dei dati
•Merging, joining e concatenazione dei dati
•Metodi I/O per files e formati speciali “HDF5”, “feather”, “parquet”…
•Time-series

Python Biella Group

Strutture dati

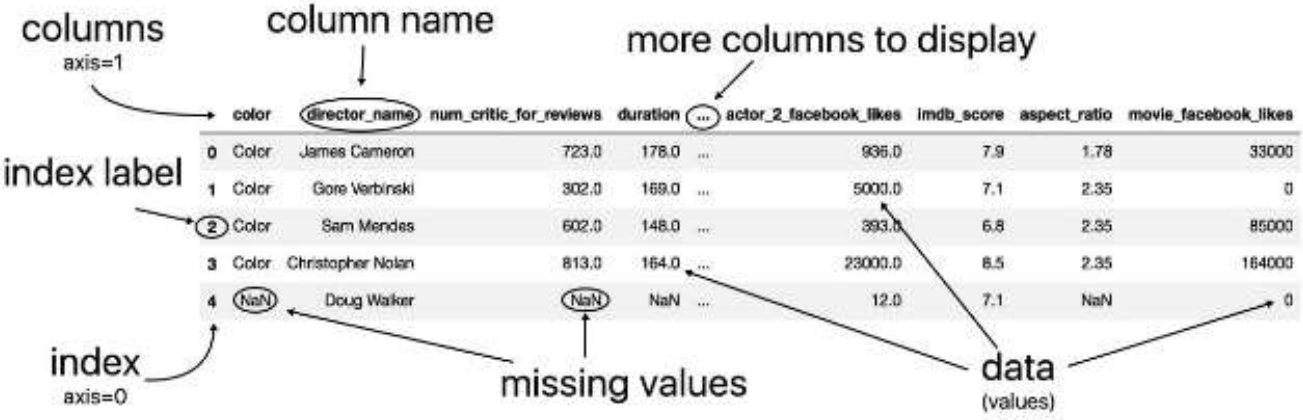
pandas.Series a 1 dimensione (1D) : 1D data and the index
In realtà 1D NumPy array (composto da qualsiasi tipo di dato) accoppiato da una array di etichette (**indice** della serie).

pandas.DataFrame a 2 dimensioni (2D): struttura dati a 2D composta da righe e colonne.
Ciascuna colonna è una Series.
Queste colonne dovrebbero essere della stessa lunghezza ma possono essere di tipo diverso (float, int, bool, e così via).
I DataFrames sono modificabili nei valori e nella dimensione (size): è possibile operare modificando i valori e/o aggiungere/cancellare righe o colonne.

Come le Series, che hanno un nome e un indice come attribute, un DataFrame ha nomi delle colonne e indice per le righe.
L'indice per le righe può essere numerico o di altro tipo (data, stringhe)
Gli indici sono necessari per ricerche veloci, per opportuni allineamenti e join di dati

~~**pandas.Panel** a 3 dimensioni (3D)~~ DEPRECATO! -> multi-indexing in DataFrames

Anatomia di un dataframe



	color	director_name	num_critic_for_reviews	duration	...	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes
0	Color	James Cameron	723.0	178.0	...	936.0	7.9	1.78	33000
1	Color	Gore Verbinski	302.0	169.0	...	5000.0	7.1	2.35	0
2	Color	Sam Mendes	602.0	148.0	...	393.0	6.8	2.35	85000
3	Color	Christopher Nolan	813.0	164.0	...	23000.0	8.5	2.35	164000
4	NaN	Doug Walker	NaN	NaN	...	12.0	7.1	NaN	0

Columns and the **index** is in **bold** font, which makes them easy to identify.
By convention, the terms **index label** and **column name** refer to the individual members of the index and columns, respectively.
The term index refers to all the index labels as a whole just as the term columns refers to all the column names as a whole

The columns and the index provide **labels** for the columns and rows of the DataFrame and allow for direct and easy access to different subsets of data.
When multiple Series or DataFrames are combined, the indexes align first before any calculation occurs.
Collectively, the columns and the index are known as the **axes**: a DataFrame has two axes--a vertical axis (the index) and a horizontal axis(the columns).
Pandas borrows convention from NumPy and uses the integers 0/1 as another way of referring to the vertical/horizontal axis.

DataFrame data (values) is always in regular font and is an entirely separate component from the columns or index.
Pandas uses **NaN** (not a number) to represent **missing values**.
*Notice that even though the color column has only string values, it uses **None** to represent a missing value.*

Python Biella Group

Indexing

pandas

(select & where)

Indicizzazione numerica -> Accesso/Selezione/Filtro per posizione:

.iloc[row_indexer, col_indexer]

col_indexer opzionale, default = :

Parametri:

- An integer, e.g. 5
- A list or array of integers, e.g. [4, 3, 0]
- A slice object with ints, e.g. 1:7
- A boolean array

Indicizzazione per etichetta -> Accesso/Selezione/Filtro per etichetta:

.loc[row_indexer, col_indexer]

col_indexer opzionale, default = :

Parametri:

- A single label, e.g. 5 or 'a'
- A list or array of labels, e.g. ['a', 'b', 'c']
- A slice object with labels, e.g. 'a':'f'. (start and stop included)
- A boolean array

Indicizzazione per etichetta -> Accesso/Selezione/Filtro per etichetta:

[indexer] (magic method: __getitem__)

Simile a .loc ma seleziona solo per colonna – filtra solo per riga

Parametri:

- A single label, e.g. 5 or 'a'
- A list or array of labels, e.g. ['a', 'b', 'c']
- A boolean array (len = num. of rows)
- A slice object with labels, e.g. 'a':'f'. (start and stop included)

Selezione di una colonna / riga -> Serie
Selezione di più colonne / righe -> DataFrame

Forte utilizzo dello slicing

jupyter

Python Biella Group

Condizioni di filtro

(where conditions)

<Pandas Serie> <operatore> <elemento di confronto>

OUTPUT = Pandas Serie (booleana) che può essere usata come filtro

Le condizioni possono essere combinate con operatori booleani ma non sono lazy (*)

DataFrame è un oggetto 2D – Sta allo sviluppatore decidere se accedere per riga o per colonna

(*) "Normalmente" Python valuta le espressioni in modo "**lazy**".
Nell'espressione (x and y), prima valuta x; se x è false, il suo valore è ritornato, altrimenti valuta anche y e ritorna il suo valore.
Nell'espressione (x or y), prima valuta x; se x è true, ritorna il suo valore altrimenti valuta y e il valore risultante è restituito.

Python Biella Group

MultiIndex

MultiIndex o **indice gerarchico**
E' un indice che usa valori multipli nella sua chiave

Quando si usa?

- Performance? (non ci sono benefici evidenti)

- «Autodocumentare» i dati, dare una struttura ai dati (*ove a senso! Attenzione: Pandas tiene i duplicati*)

- In combinazione con stack() e unstack()

Python Biella Group

Stack / Unstack

stack() «impila» i(l) livelli(o) trasformandoli da colonne a indice

Restituisce un DataFrame o una serie rimodellato/a con un indice a più livelli con uno o più nuovi livelli più interni rispetto al DataFrame corrente.

I nuovi livelli più interni vengono creati ruotando (**pivoting**) le colonne del dataframe corrente

unstack() operazione inversa

Ruota («pivot») un livello delle etichette indice (necessariamente gerarchiche).

...and JOIN US!

- Sito: https://pythonbiella.herokuapp.com/
- GitHub: https://github.com/PythonGroupBiella/MaterialeLezioni
- YouTube: https://www.youtube.com/channel/UCkvQcNjmC_duLhvDxeUPJAg
- Telegram: https://t.me/joinchat/AAAAAFGSWcxhSln_SRhseQ