*Pandas vs Apache Spark? La soluzione è **Koalas**!*

Pandas, come già accennato, è una *libreria* scritta per il linguaggio di programmazione *Python* che ci permette di manipolare i dati e anche analizzarli in maniera approfondita, semplificando, spesse volte, le operazioni di scrittura del codice che altrimenti sarebbero meno compatte. Pandas infatti è considerata *una delle quattro librerie principali dell'ecosistema per il datascience di Python insieme a **NumPy**, **SciKit Learn** e **MatPlotLib***.

Apache Spark, o più semplicemente Spark, è un *framework open source* basato sul paradigma *MapReduce* e consente ottime prestazioni quando si tratta di caricare dati in larga scala e interrogarli ripetutamente, questo grazie al fatto che è stato studiato e realizzato per un uso orientato al *machine learning*. L'API per *Python* di Spark è ***PySpark***.

Le principali differenze tra Pandas e Spark Vediamo ora le prime differenze tra *Pandas* e *Spark*:

- *Pandas* è rivolto all'elaborazione di collezioni dati di dimensioni relativamente piccole ed è lo standard d'uso su singole macchine;
- *Spark* è rivolto all'elaborazione di collezioni dati di grandi dimensioni ed è lo standard d'uso per l'elaborazione su archiviazione distribuita.

Ma c'è da dire che, come detto all'inizio, *Pandas* consente una evidente compattezza del codice e consente anche un'elaborazione più "snella e svelta" dei dati, infatti, ad esempio, un DataFrame in *Pandas*, è mutabile, mentre in *PySpark* no e per modificarlo bisognerebbe crearne uno nuovo partendo dal vecchio.

Aiutiamoci nella comprensione delle differenze tra l'utilizzo di *Pandas* e *PySpark* con le seguenti porzioni di codice:

Pandas `import pandas as pd df = pd.read_csv("my_data.csv) df.columns = ['x', 'y', 'z1'] df['x2'] ? df.x * df.x`

PySpark `df = (spark.read.option("inferSchema", "true"). option("comment", True). csv("my_data.csv")) df = df.toDF('x', 'y', 'z1') df = df.withColumn('x2', df.x*df.x)`

Com'è possibile notare dal codice qui sopra, l'utilizzo di *Pandas* è molto più intuitivo e veloce a differenza dell'uso di *PySpark*.

Ed è proprio qui che ***Koalas*** prova a "sistemare le cose": *Koalas* infatti è un toolkit, annunciato nell'Aprile del 2019, che effettua una sorta di ""porting" di *Pandas* sulla piattaforma *Spark*, permettendo quindi l'elaborazione di collezioni di dati molto grandi e con un'archiviazione distribuita in modo smart e compatto. La svolta apportata da *Koalas* è quella di poter offrire il medesimo codebase sia per l'utilizzo su *Pandas* che per l'utilizzo su *PySpark* senza dover effettuare appunto modifiche al codice (**Nota**: vanno opportunamente modificate le import).

Anche stavolta sfruttiamo due porzioni di codice per capire meglio quanto appena detto:

Pandas `import pandas as pd df = pd.read_csv("my_data.csv) df.columns = ['x', 'y', 'z1'] df['x2'] ? df.x * df.x`

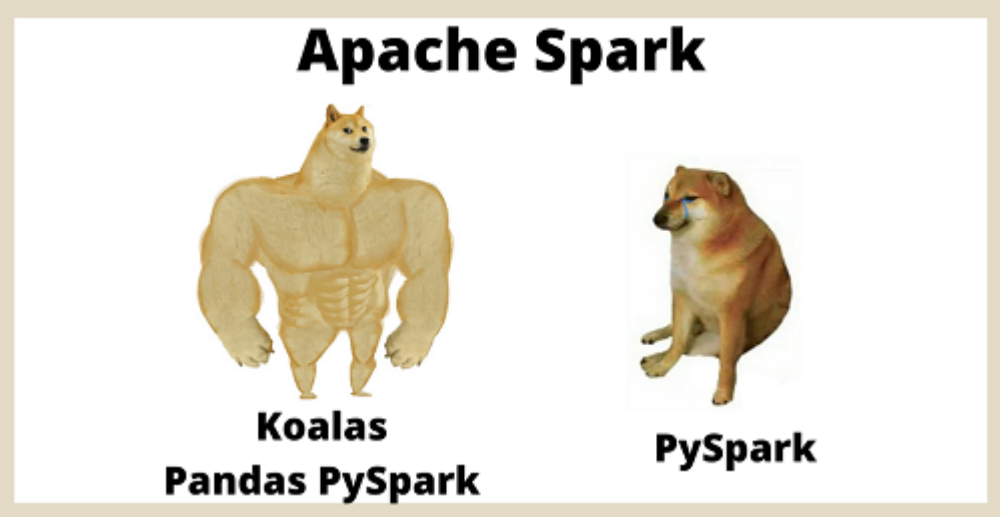
Koalas `import databricks.koalas as ks df = ks.read_csv("my_data.csv) df.columns = ['x', 'y', 'z1'] df['x2'] ? df.x * df.x`

Come si può notare, a meno degli import per ovvi motivi differenti, dato che stiamo usando due librerie diverse, il codice è esattamente il medesimo.

Conclusioni

Come visto durante questo breve versus, l'utilizzo della piattaforma *Apache Spark* offre numerosi vantaggi rispetto all'uso della libreria *Pandas* ma certo tra essi non vi è la semplicità di scrittura del codice. A porre rimedio a tutto questo ci pensa *Koalas* che permette lo switch tra *Pandas* e *Apache Spark*, bypassando *PySpark* , e sfruttando il medesimo codice.

Stavolta, ironicamente, potremmo usare per capire meglio, al posto di porzioni di codice, un meme per concludere al meglio il nostro discorso:


Apache Spark
Koalas
Pandas PySpark
PySpark