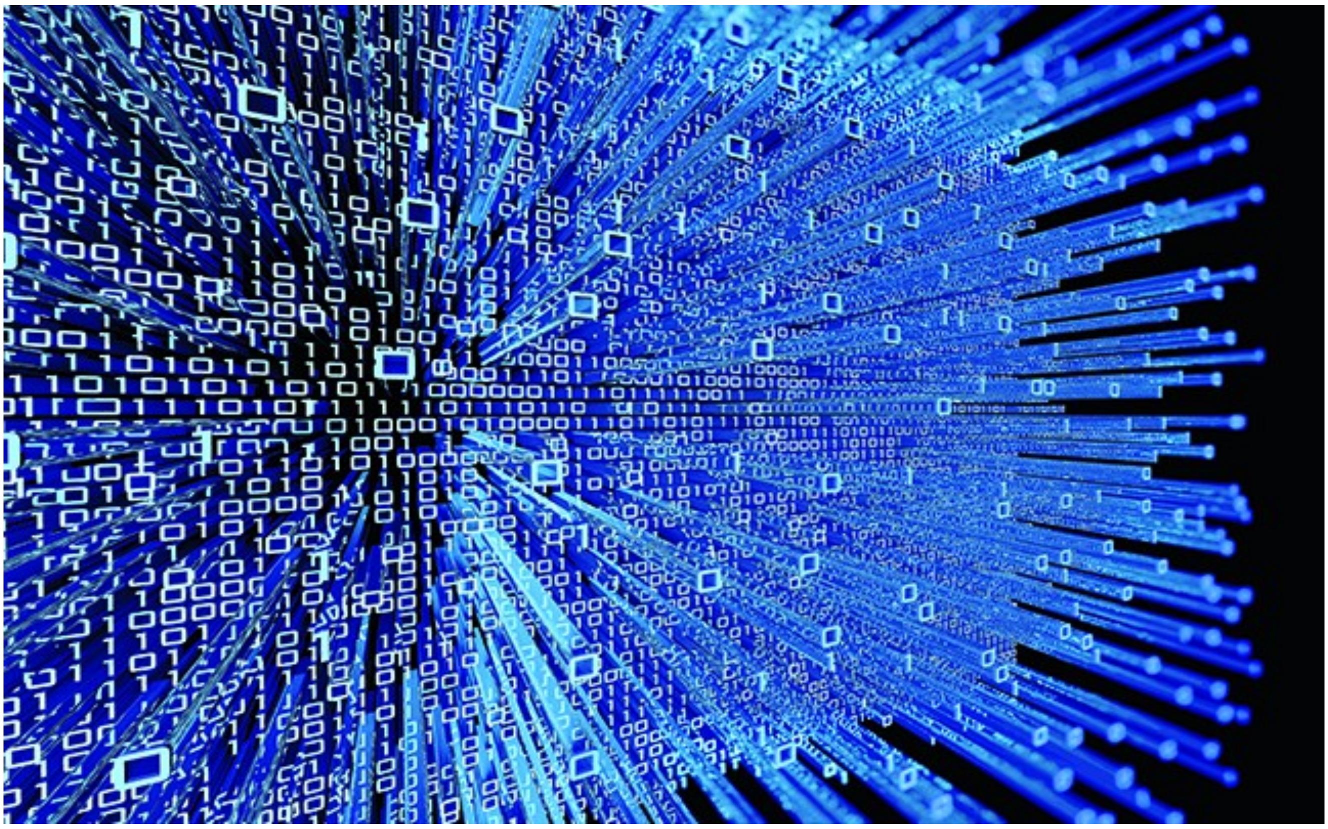
Why and How to Use Pandas with Large Data

But Not Big Data...



Pandas has been one of the most popular and favourite data science tools used in Python programming language for data wrangling and analysis.

Data is unavoidably messy in real world. And Pandas is *seriously* a game changer when it comes to cleaning, transforming, manipulating and analyzing data. In simple terms, Pandas helps to **clean the mess.**

My Story of NumPy & Pandas

When I first started out learning Python, I was naturally introduced to NumPy (Numerical Python). It is the fundamental package for scientific computing with Python that provides an abundance of useful features for operations on n-arrays and matrices in Python.

In addition, the library provides vectorization of mathematical operations on the NumPy array type, which significantly optimizes computation with high performance and enhanced speed of execution.

NumPy is cool.

But therein still lies some underlying needs for more higher level of data analysis tools. And this is where Pandas comes to my rescue.

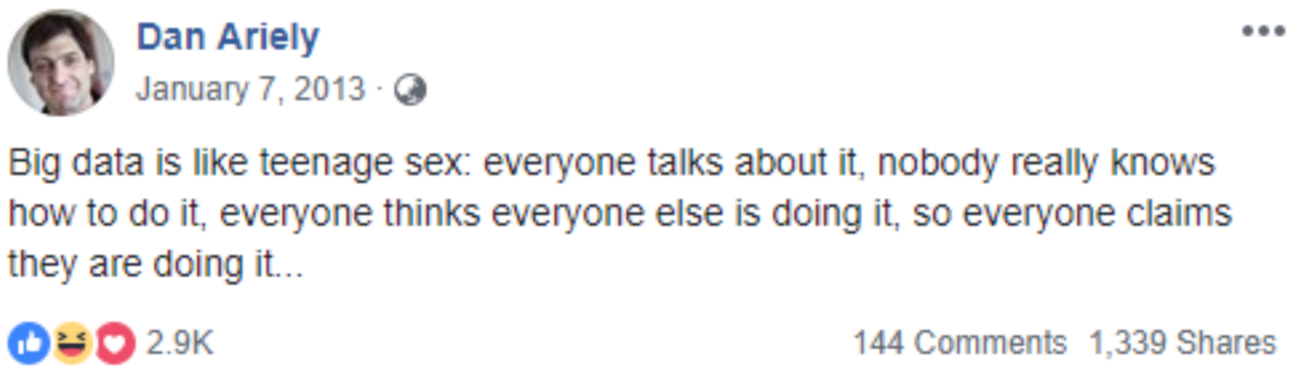
Fundamentally, the functionality of Pandas is built on top of NumPy and both libraries belong to the SciPy stack. This means that Pandas relies heavily on NumPy array to implement its objects for manipulation and computation — but used in a more convenient fashion.

In practice, NumPy & Pandas are still being used interchangeably. The high level features and its convenient usage are what determine my preference in Pandas.

Why use Pandas with Large Data — Not BIG Data?

There is a stark difference between large data and big data. With the hype around big data, it is easy for us to consider everything as "big data" and just go with the flow.

A famous joke by Prof. Dan Ariely:


Dan Ariely
January 7, 2013 ·

Big data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it...

2.9K   144 Comments 1,339 Shares

(Source)

The word large and big are in themselves 'relative' and in my humble opinion, large data is data sets that are less than 100GB.

Pandas is very efficient with small data (usually from 100MB up to 1GB) and performance is rarely a concern.

However, if you're in data science or big data field, chances are you'll encounter a common problem sooner or later when using Pandas — low performance and long runtime that ultimately result in insufficient memory usage — when you're dealing with large data sets.

Indeed, Pandas has its own limitation when it comes to big data due to its algorithm and local memory constraints. Therefore, big data is typically stored in computing clusters for higher scalability and fault tolerance. And it can often be accessed through big data ecosystem (AWS

EC2, Hadoop etc.) using Spark and many other tools.

Eventually, one of the ways to use Pandas with large data on local machines (with certain memory constraints) is to reduce memory usage of the data.

How to use Pandas with Large Data?

(Source)

So the question is: **How to reduce memory usage of data using Pandas?**

The following explanation will be based my experience on an anonymous large data set (40–50 GB) which required me to reduce the memory usage to fit into local memory for analysis (even before reading the data set to a dataframe).

1. Read CSV file data in chunk size

To be honest, I was baffled when I encountered an error and I couldn't read the data from CSV file, only to realize that the memory of my local machine was too small for the data with 16GB of RAM.

Here comes the good news and the beauty of Pandas: I realized that **pandas.read_csv** has a parameter called **chunksize**!

The parameter essentially means the number of rows to be read into a dataframe at any single time in order to fit into the local memory. Since the data consists of more than 70 millions of rows, I specified the chunksize as 1 million rows each time that broke the large data set into many smaller pieces.

Read CSV file data in chunksize

The operation above resulted in a TextFileReader object for iteration. Strictly speaking, **df_chunk** is not a dataframe but an object for further operation in the next step.

Once I had the object ready, the basic workflow was to perform operation on each chunk and concatenate each of them to form a dataframe in the end (as shown below). By iterating each chunk, I performed data filtering/preprocessing using a function — **chunk_preprocessing** before appending each chunk to a list. And finally I concatenated the list into a final dataframe to fit into the local memory.

Workflow to perform operation on each chunk

2. Filter out unimportant columns to save memory

Great. At this stage, I already had a dataframe to do all sorts of analysis required.

To save more time for data manipulation and computation, I further filtered out some unimportant columns to save more memory.

Filter out unimportant columns

3. Change dtypes for columns

The simplest way to convert a pandas column of data to a different type is to use `astype()`.

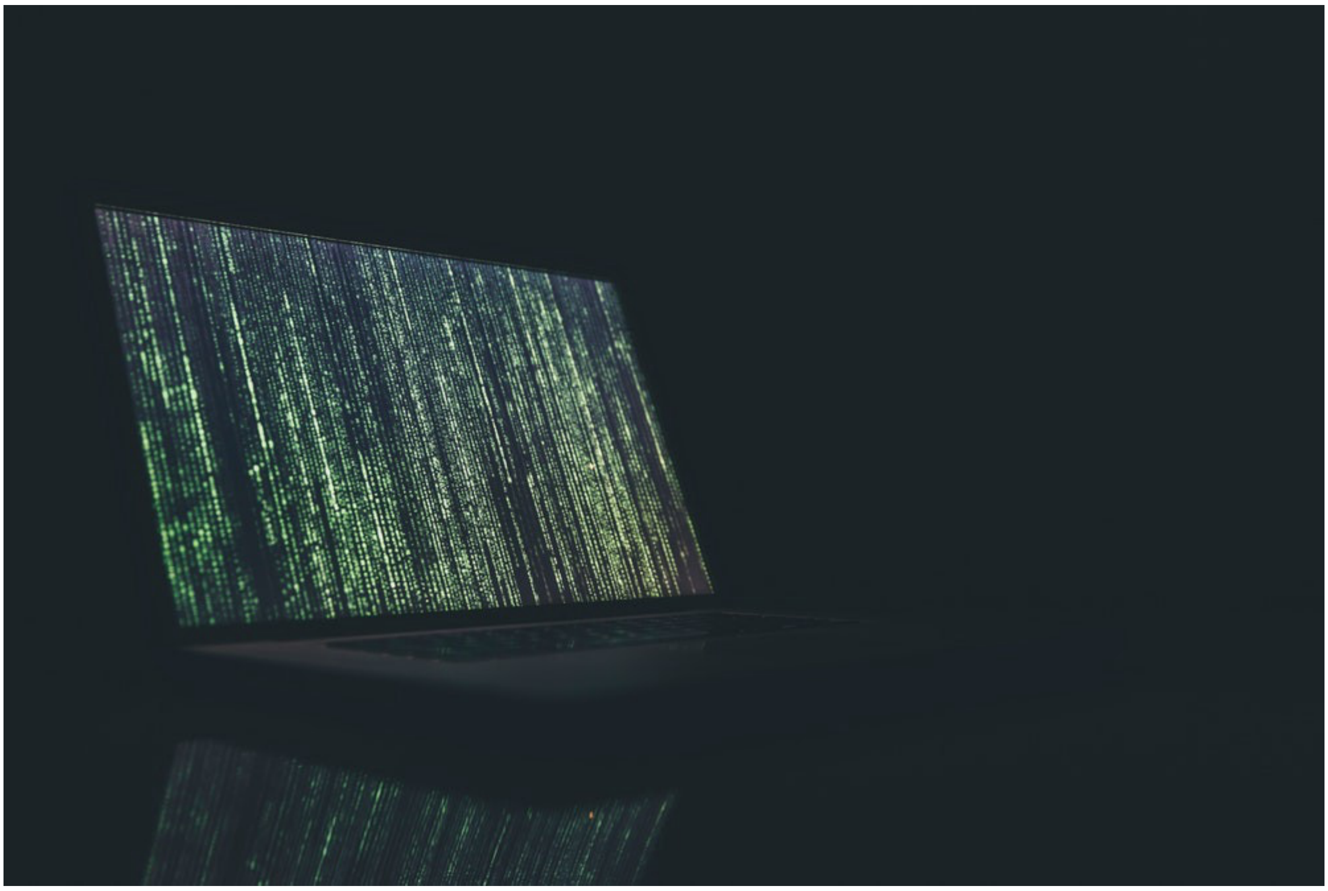
I can say that changing data types in Pandas is extremely helpful to save memory, especially if you have large data for intense analysis or computation (For example, feed data into your machine learning model for training).

By reducing the bits required to store the data, I reduced the overall memory usage by the data up to 50% !

Give it a try. And I believe you'll find that useful as well! Let me know how it goes. 😊

Change data types to save memory

Final Thoughts


(Source)

There you have it. Thank you for reading.

I hope that sharing my experience in using Pandas with large data could help you explore another useful feature in Pandas to deal with large data by reducing memory usage and ultimately improving computational efficiency.

Typically, Pandas has most of the features that we need for data wrangling and analysis. I strongly encourage you to check them out as they'd come in handy to you next time.

Also, if you're serious about learning how to do data analysis in Python, then this book is for you — **Python for Data Analysis**. With complete instructions for manipulating, processing, cleaning, and crunching datasets in Python using Pandas, the book gives a comprehensive and step-by-step guides to effectively use Pandas in your analysis.

Hope this helps!