ARTIFICIAL INTELLIGENCE

8 Alternatives to Pandas for Processing Large Datasets

Stop using Pandas

Edwin Tan

Jan 17, 2022 3 min read

Photo by Erik Mclean on Unsplash

Pandas library has became the de facto library for data manipulation in python and is widely used by data scientist and analyst. However, there are times when the dataset is too large and Pandas may run into memory errors. Here are 8 alternatives to Pandas for dealing with large datasets. For each alternative library, we will examine how to load data from CSV and perform a simple `groupby` operation. Fortunately many of these libraries have similar syntax as Pandas hence making the learning curve less steep.

Dask

> *Dask provides multi-core and distributed parallel execution on larger-than-memory datasets.*

A Dask DataFrame is a large parallel DataFrame composed of many smaller Pandas DataFrames, split along the index. These Pandas DataFrames may live on disk for larger-than-memory computing on a single machine, or on many different machines in a cluster. One Dask DataFrame operation triggers many operations on the constituent Pandas DataFrames.

```
pip install dask
```

```
import dask.dataframe as dd
df = dd.read_csv('../input/yellow-new-york-taxi/yellow_tripdata_2009-
df2 = df.groupby('vendor_name').agg({'Passenger_Count':'mean'})
```

Up to this point no computation has been executed yet. To compute the results for `df2`

```
df2.compute()
```

Dask is more than just a data manipulation library. Dask-ML provides scalable machine learning which can be used along side with popular machine learning libraries such as Scikit-learn, Xgboost and LightBGM.

Modin

> *Modin uses Ray or Dask to provide an effortless way to speed up your pandas notebooks, scripts, and libraries.*

```
pip install modin
```

```
import modin.pandas as pd
df = pd.read_csv('../input/yellow-new-york-taxi/yellow_tripdata_2009-
df2 = df.groupby('vendor_name').agg({'Passenger_Count':'mean'})
```

Data Table

> *Datatable is a python library for manipulating tabular data. It supports out-of-memory datasets, multi-threaded data processing, and flexible API.*

```
pip install datatable
```

```
from datatable import dt, f, by
df = dt.fread('../input/yellow-new-york-taxi/yellow_tripdata_2009-01.
df2 = df[:, dt.mean(f.Passenger_Count), by('vendor_name')]
```

Polars

> *Polars is a blazingly fast DataFrames library implemented in Rust using Apache Arrow Columnar Format as memory model.*

```
pip install polars
```

```
import polars as pl

df = pl.read_csv('../input/yellow-new-york-taxi/yellow_tripdata_2009-
df2 = df.groupby('vendor_name').agg([pl.mean('Passenger_Count')])
```

Vaex

> *Vaex is a python library for lazy Out-of-Core DataFrames (similar to Pandas), to visualize and explore big tabular datasets*

```
pip install vaex
```

```
df = vaex.read_csv('../input/yellow-new-york-taxi/yellow_tripdata_200
df2 = df.groupby('vendor_name').agg({'Passenger_Count':'mean'})
```

Pyspark

> *Pyspark is a Python API for Apache Spark used to process large dataset through distributed computation.*

```
pip install pyspark
```

```
from pyspark.sql import SparkSession, functions as f
spark = SparkSession.builder.appName("SimpleApp").getOrCreate()

df = spark.read.option('header', True).csv('../input/yellow-new-york-
df2 = df.groupby('vendor_name').agg(f.mean('Passenger_Count'))
```

Up to this point no computation has been executed yet. To compute the results for `df2`

```
df2.show()
```

Koalas

> *The Koalas project makes data scientists more productive when interacting with big data, by implementing the pandas DataFrame API on top of Apache Spark.*

As Koalas runs on top of Apache Spark, Spark has to be installed as well.

```
pip install pyspark
pip install koalas
```

```
import databricks.koalas as ks
from pyspark.sql import SparkSession
df = ks.read_csv('../input/yellow-new-york-taxi/yellow_tripdata_2009-
df2 = df.groupby('vendor_name').agg({'Passenger_Count':'mean'})
```

cuDF

> _cuDF is a Python GPU DataFrame library built on the Apache Arrow columnar memory format for data manipulation. cuDF also provides a pandas-like API that will be familiar to data engineers & data scientists, so they can use it to easily accelerate their workflows without going into the details of CUDA programming._

cuDF can be installed via conda, take a look at this guide.

```
import cudf
df = cudf.read_csv('../input/yellow-new-york-taxi/yellow_tripdata_200
df2 = df.groupby('vendor_name').agg({'Passenger_Count':'mean'})
```

Conclusion

In this article we looked at 8 python libraries for manipulating large datasets. Many of them have similar syntax as Pandas which makes the makes the learning curve less steep. If you are wondering about the execution speed for different packages, H2O.ai has created a useful ops benchmark on some of these packages. It compares the speed of execution of various packages on dataset of different size and for different common operations such as joins and groupbys.

I hope this is useful for a start, leave a comment below your go-to library for dealing with large datasets in python.