Business analytics with Python-based tools

Paris Open Source Summit - 7 December 2017

Stéfane Fermigier
Founder & CEO, Abilian - Enterprise Social Software
Intro
Who am I?

- Stefane Fermigier, Python developer since 1996
- Founder of Abilian SAS
  - Python shop, developing business application (collaboration, CRM, workflow…)
  - R&D activity (Wendelin -> Olapy)
Why use Python for business data analysis?

- Why not? :)
- Python is one of the leading languages for data science/data processing, and also a leading language for web & business apps.
- As a Python shop, we’d like to leverage this leadership in data processing tools to build exploration/reporting features in our business applications using a familiar language.

Source: KDnuggets
Our goal today

• Overview and demo a few useful tools related to business data analytics

• Use a very common dataset, called “Black Friday” (sales for a variety of products, over a variety of categories, locations, etc.), as a starting point for our explorations
“Black Friday” dataset
Pandas & Jupyter
Jupyter Notebooks

• Originally called iPython notebooks
• Very simple to use
• Web based notebook
• Great environment for exploration
• Rich text (markdown) inline comments
• Figures embed into the documents

Install
• pip install jupyter

Run
• jupyter notebook
Data analysis tools library

Built on NumPy, inspired by R

Provides built-in data structures which simplify the manipulation and analysis of data sets.

https://pandas.pydata.org/

Use the following import convention:

```python
In [1]: import pandas as pd
```
Pandas Data Structures

Series

- A one-dimensional labeled array capable of holding any data type

```
In [2]: pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```
• A two-dimensional labeled data structure with columns of potentially different types

<table>
<thead>
<tr>
<th>Country</th>
<th>Capital</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Belgium</td>
<td>Brussels</td>
<td>11190846</td>
</tr>
<tr>
<td>India</td>
<td>New Delhi</td>
<td>1303171035</td>
</tr>
<tr>
<td>Brazil</td>
<td>Brasília</td>
<td>207847528</td>
</tr>
</tbody>
</table>

```
In [6]: data = {'Country': ['Belgium', 'India', 'Brazil'],
               'Capital': ['Brussels', 'New Delhi', 'Brasília'],
               'Population': [11190846, 1303171035, 207847528]}

df = pd.DataFrame(data, columns=['Country', 'Capital', 'Population'])
```
Working with files

Read

```python
In [4]:
products_df = pd.read_csv('Black_Friday/Product.csv', sep=';')
products_df
```

```
<table>
<thead>
<tr>
<th>Company</th>
<th>Category</th>
<th>Sub_Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>Tech/electronics</td>
<td>TV</td>
</tr>
<tr>
<td>Apple</td>
<td>Tech/electronics</td>
<td>Iphone</td>
</tr>
<tr>
<td>Amazon</td>
<td>Household items</td>
<td>Fridge</td>
</tr>
<tr>
<td>Amazon</td>
<td>Household items</td>
<td>Washer</td>
</tr>
<tr>
<td>zara</td>
<td>Clothing/accessories</td>
<td>jackets</td>
</tr>
<tr>
<td>h&amp;m</td>
<td>Clothing/accessories</td>
<td>shirts</td>
</tr>
<tr>
<td>Walmart</td>
<td>Other</td>
<td>Other</td>
</tr>
<tr>
<td>Amazon</td>
<td>Game</td>
<td>Playstation</td>
</tr>
<tr>
<td>Toys R Us</td>
<td>Game</td>
<td>Kids_game</td>
</tr>
</tbody>
</table>
```

Write

```python
In [5]:
# pip install openpyxl
df.to_excel('myDataFrame.xlsx', sheet_name='Sheet1')
```
**Advanced manipulations**

### Combining Data

```python
In [5]:
facts_df = pd.read_csv('Black_Friday/Facts.csv', sep=';')
facts_df.head()
```

```python
Out[5]:
<table>
<thead>
<tr>
<th>Month</th>
<th>City</th>
<th>Sub_Category</th>
<th>average_sales_M</th>
<th>Average Spent By Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 2015</td>
<td>Madrid</td>
<td>jackets</td>
<td>110</td>
<td>84</td>
</tr>
<tr>
<td>November 2015</td>
<td>Barcelona</td>
<td>shirts</td>
<td>190</td>
<td>141</td>
</tr>
<tr>
<td>November 2015</td>
<td>Paris</td>
<td>jackets</td>
<td>300</td>
<td>100</td>
</tr>
<tr>
<td>November 2015</td>
<td>Lausanne</td>
<td>Washer</td>
<td>220</td>
<td>231</td>
</tr>
<tr>
<td>November 2015</td>
<td>Lausanne</td>
<td>TV</td>
<td>120</td>
<td>365</td>
</tr>
</tbody>
</table>
```

```python
In [6]:
combined_df = facts_df.merge(products_df)
combined_df.head()
```

```python
Out[6]:
<table>
<thead>
<tr>
<th>Month</th>
<th>City</th>
<th>Sub_Category</th>
<th>average_sales_M</th>
<th>Average Spent By Person</th>
<th>Company</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 2015</td>
<td>Madrid</td>
<td>jackets</td>
<td>110</td>
<td>84</td>
<td>zara</td>
<td>Clothing/accessories</td>
</tr>
<tr>
<td>November 2015</td>
<td>Paris</td>
<td>jackets</td>
<td>300</td>
<td>100</td>
<td>zara</td>
<td>Clothing/accessories</td>
</tr>
<tr>
<td>November 2014</td>
<td>Madrid</td>
<td>jackets</td>
<td>90</td>
<td>72</td>
<td>zara</td>
<td>Clothing/accessories</td>
</tr>
<tr>
<td>November 2014</td>
<td>Paris</td>
<td>jackets</td>
<td>200</td>
<td>95</td>
<td>zara</td>
<td>Clothing/accessories</td>
</tr>
<tr>
<td>November 2015</td>
<td>Barcelona</td>
<td>shirts</td>
<td>190</td>
<td>141</td>
<td>h&amp;m</td>
<td>Clothing/accessories</td>
</tr>
</tbody>
</table>
Grouping Data

In [7]: `combined_df.groupby(by=['Company', 'Category']).sum()`

Out[7]:

<table>
<thead>
<tr>
<th>Company</th>
<th>Category</th>
<th>average_sales_M</th>
<th>Average Spent By Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>Game</td>
<td>2138</td>
<td>1281</td>
</tr>
<tr>
<td></td>
<td>Household items</td>
<td>849</td>
<td>1366</td>
</tr>
<tr>
<td></td>
<td>Tech/electronics</td>
<td>1252</td>
<td>1306</td>
</tr>
<tr>
<td>Apple</td>
<td>Tech/electronics</td>
<td>1308</td>
<td>800</td>
</tr>
<tr>
<td>Toys R Us</td>
<td>Game</td>
<td>1204</td>
<td>436</td>
</tr>
<tr>
<td>Walmart</td>
<td>Other</td>
<td>294</td>
<td>940</td>
</tr>
<tr>
<td>h&amp;m</td>
<td>Clothing/accessories</td>
<td>330</td>
<td>246</td>
</tr>
<tr>
<td>zara</td>
<td>Clothing/accessories</td>
<td>700</td>
<td>351</td>
</tr>
</tbody>
</table>
Pivot Table

- Spread rows into columns

```python
In [8]: pt = pd.pivot_table(combined_df, values=['average_sales_M', 'Average Spent By Person'],
                        index=['Month', 'City', 'Category'],
                        columns='Company')
pt.fillna('')
# fillna to replace NaN values
```

<table>
<thead>
<tr>
<th>Month</th>
<th>City</th>
<th>Category</th>
<th>Amazon</th>
<th>Apple</th>
<th>Toys R Us</th>
<th>Walmart</th>
<th>h&amp;m</th>
<th>zara</th>
<th>Amazon</th>
<th>Apple</th>
<th>Toys R Us</th>
<th>Walmart</th>
<th>h&amp;m</th>
<th>zara</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 2014</td>
<td>Barcelona</td>
<td>Clothing/accessories</td>
<td>105</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Geneva</td>
<td>Household Items</td>
<td>220</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lausanne</td>
<td>Household Items</td>
<td>210</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tech/electronics</td>
<td>235</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Los Angeles</td>
<td></td>
<td>Game</td>
<td>231</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Madrid</td>
<td></td>
<td>Clothing/accessories</td>
<td>72</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New York</td>
<td></td>
<td>Game</td>
<td>103</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tech/electronics</td>
<td>303</td>
<td>420</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paris</td>
<td></td>
<td>Clothing/accessories</td>
<td>95</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Game</td>
<td>200</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zurich</td>
<td></td>
<td>Other</td>
<td>440</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>November 2015</td>
<td>Barcelona</td>
<td>Clothing/accessories</td>
<td>141</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Apache Superset (incubating)
“A modern, enterprise-ready business intelligence tool”

- Data exploration and visualisation platform
  - A Rich SQL IDE
  - A Data Exploration Interface

- Create and share interactive dashboards

- Flexible authentication and authorisation

- Customisable and hackable (based on Flask)!

- Supports many backends (MySQL, Postgres, Redshift, SparkSQL...)
• Table by Table (report/analyses)

Black Friday dataset

https://github.com/apache/incubator-superset
Black Friday dataset

- Table by Table (report/analyses)

https://github.com/apache/incubator-superset
Olapy
Olapy

- Developed since 2016 by Abilian
- In-memory data processing using Pandas
- Aggregated data browsing
- MDX support
- XMLA interface (-> Excel)
- Multiple back-ends (CSV, SQL)
- Simple web front-end

https://github.com/abilian/olapy
MDX = Multi Dimensional Expressions

SQL extension for querying a multi-dimensional database

```
Select
    [Geography].[Geo].[Country] on Rows,
    [Time].[Calendar].[Year].[2010] on Columns
From Sales
Where [Measures].[Count]
```
MDX ??

❖ MDX = Multi Dimensional Expressions
❖ SQL extension for querying a multi-dimensional database

Select

[Geography].[Geo].[Country] on Rows,
[Time].[Calendar].[Year].[2010] on Columns

From Sales

Where [Measures].[Count]
MDX = Multi Dimensional Expressions

SQL extension for querying a multi-dimensional database

```
Select [Geography].[Geo].[Country] on Rows,
      [Time].[Calendar].[Year].[2010] on Columns
From Sales
Where [Measures].[Count]
```
MDX

- MDX = Multi Dimensional Expressions
- SQL extension for querying a multi-dimensional database

```sql
Select  
  [Geography].[Geo].[Country]  on Rows,  
  [Time].[Calendar].[Year].[2010]  on Columns
From  Sales
Where  [Measures].[Count]
```
MDX = Multi Dimensional Expressions

SQL extension for querying a multi-dimensional database

Select

[Geography].[Geo].[Country] on Rows,
[Time].[Calendar].[Year].[2010] on Columns

From Sales

Where [Measures].[Count]
MDX ??

- MDX = Multi Dimensional Expressions
- SQL extension for querying a multi-dimensional database

Select

[Geography].[Geo].[Country] on Rows,
[Time].[Calendar].[Year].[2010] on Columns

From Sales

Where [Measures].[Count]
Extensible Markup Language for Analysis - XMLA

- Data Access Protocol
- Supports exchange of analytical data between clients and servers
  - Available on Any Device or Platform
  - Using Any Programming Language
- Just SOAP
  - Discover
  - Execute
On-Line Analytical Processing (OLAP) & Multidimensional Databases

- A multidimensional BD is a hypercube:
- Axes are called user-defined dimensions
- Cells contain measures calculated from more or less complex formulas.
- Operators on the cube are algebraic (return a cube) and can thus be combined.

\[\text{Multi-dimensional database} = \text{"super-spreadsheet"}\]
Olapy as server

Install:
• pip install olapy

Run:
• olapy runserver

➢ From excel go to: Data/from other sources/from analyses services
➢ Use URL http://127.0.0.1:8000/xmla
Olapy as server

Install:
- pip install olapy

Run:
- olapy runserver

➢ From excel go to:

Data/from other sources/
from analyses services

➢ Use URL

http://127.0.0.1:8000/xmla
olapy-web

- Web client for olapy-core (very basic)
- Interactive data exploration
- Dashboard based on configuration file
- Based on pivottable.js and Plotly

1. git clone https://github.com/abilian/olapy-web.git
2. cd olapy-web
3. pip install -r requirements.txt
4. export FLASK_APP=manage.py
5. flask run
6. Use URL 127.0.0.1:5000 on your web browser
olapy-web

• Web client for olapy-core (very basic)
• Interactive data exploration
• Dashboard based on configuration file
• Based on pivottable.js and Plotly

1. git clone https://github.com/abilian/olapy-web.git
2. cd olapy-web
3. pip install --r requirements.txt
4. export FLASK_APP=manage.py
5. flask run
6. Use URL 127.0.0.1:5000 on your web browser
Use olapy as library

• Execute MDX queries

```python
In [1]: from olapy.core.mdx.executor.execute import MdxEngine

mdx_query = """SELECT Hierarchize({[Measures].[average_sales_M]}) ON COLUMNS FROM [Black_Friday] """

egressor = MdxEngine('Black_Friday')

execution_result = executor.execute_mdx(mdx_query)['result']

Out[1]:

average_sales_M
0    8875
```
Roadmap

- Version 0.5 just released!
- WIP
  - Benchmarking & performance tuning
  - Web front-end / OnlyOffice integration
  - Integration in real projects
  - Multi-core / multi-server scalability using the wendelin-core out-of-core computation engine
Bonobo ETL
Bonobo

- python 3.5+

```python
import bonobo

data = bonobo.Graph(
    bonobo.CsvReader('employees.csv'),
    bonobo.Filter(
        lambda **row: row['position'] != 'CEO',
    ),
    bonobo.CsvWriter('employees.output.csv'),
)
```
from olapy.etl.etl import run_olapy_etl

dims_infos = {
    'Geography': ['geography_key'],
    'Product': ['product_key']
}

facts_ids = ['geography_key', 'product_key']

# source_type = 'csv' | 'file' | 'db'
run_olapy_etl(dims_infos=dims_infos,
              facts_ids=facts_ids,
              facts_table='sales_facts',
              source_type='csv')
More tools
Redash

• Query all your data sources in one place
• Convert your queried data into visualisations

Online Demo:
http://demo.redash.io
Redash

• Query all your data sources in one place
• Convert your queried data into visualisations

Online Demo:
http://demo.redash.io
Cubes & friends

• Light-weight Python framework and OLAP HTTP server
• OLAP and aggregated browsing
• Multiple hierarchies in a dimension
• Authentication and authorisation of cubes and their data

• Data exploration and visualisation tool for Cubes

Online demo:
http://www.cubesviewer.com/studio.html
Conclusion
More info

- Slides will appear soon on https://speakerdeck.com/sfermigier/
- Repo for this talk: https://github.com/abilian/talks
- Repo Olapy: https://github.com/abilian/olapy
- Contact: sf@abilian.com