USING PYTHON FOR BIG DATA AND ANALYTICS

If you have been reading our blog posts here, you will see that most of our big data analytics and data science examples are written using Python code. Why is that? Why do most data scientists seem to prefer Python?

Why Use Python for DataScience?

There are several reasons why Python is so popular with data scientists:

- Python is easy and a lot less wordy than other languages (in particular compared to Java, which is too wordy say many programmers.). But **easy** and **simple** do not mean Python is in any way limited. In fact, because of its open source nature, other programmers can add features to the language. Oracle maintains control over Java, so its distribution mechanism for third-party code is somewhat more complicated, requiring extra steps to download and include JAR files, which is how those are distributed with Java.
- Closely related to the comment above, mathematicians and programmers have code in Python some very complex structures and algorithms so that regular programmers can find them easier to use.
- Python is built into the command line shells of some big data products, a testament to its wide use.
- It's an alternative for data scientists who in the past used more scientific tools, tools more geared to graduate level mathematics and statistics. Before big data, data science was something only statisticians, operations researchers, and applied mathematicians could

do. Their tools include Wolfram Alpha and Mathematics, Matplotlib, Scikit-learn, and R. The first two are mainly to solve complex math equations and draw graphs. The last two are still used by data scientists, as they have been ported to big data environments, with only a few limits on scalability.
- As a corollary to what I just wrote above, Python is easy enough that data scientists can start using it instead of Microsoft Excel or Google Sheets, which is where they often start. Those are easy tools to get started looking at a data set. But as the user becomes more familiar with Python he or she can learn that with just a couple of lines of code they can also browse, summarize, and arrange data with just a few commands, just like Excel. That is especially true if they use one Zeppelin or Jupyter (see below) as scratch pads for their work, as they support creating graphs and tables with one button. You cannot make complex graphs in the Python shell, since it is not a graphical environment.
- Many Machine Learning algorithms are written in Python. These include Google TensorFlow, Microsoft Cognitive ToolKit, Scikit-learn, and Spark ML (Spark Machine Learning). We should give credit where credit is due and say that much of the mathematics programmed here come from university researchers, working at places like the Princeton Institute for Advanced Study working in the 1940s to the present day. So Google, Microsoft, etc. are all using the same algorithms. Of course Stanford has moved into that stratosphere of great thinkers with the Page Rank and other algorithms but neural networks, regression, and classification and the algorithms used to find solutions to these came decades earlier, before there were databases and languages that could handle those in the computer.

Python is Easy for Beginners yet Advanced for Those Who Need It

You can write a Python program in one simple line. For example, enter **1*2** into the Python shell and it responds **2**.

Python code can be run in the interactive Python shell or be submitted to the Python interpreter as a batch job. The added advantage with an interactive shell is when you are writing pieces of a larger program you can type individual sections into the command line shell and it will execute those. Then as you get that small piece of logic working you paste it into the larger program. So it too is a scratchpad towards building something larger.

But this simplicity does not mean that Python is in any way limited. It supports modern data structures, like sets and maps, as well as primitive types like integers and even complex numbers. But, as we see below, Python includes Numpy, which is the main API used for what is called "scientific computing ecosystem." Don't worry. You won't need much science here except for an understanding of matrices and linear algebra, which is what the vast majority of ML algorithms use, even neural networks. Numpy handles linear algebra and matrix mathematics on a very large scale. Most machine learning algorithms operate on these n-dimensional matrices. Other than that the mathematics is not very complicated. Neural networks, in fact, spend much of their time guessing at solutions, trying that, then guessing again. So it's not very elegant as far as more advanced mathematics go.

Python Command Line Shell

Apache Spark has a Python shell. That means you can open datasets, do transformations, and run algorithms in one easy command line. Without that you would have to package your program and

then submit it to Spark using spark-submit. The disadvantage with **spark-submit**, as with any batch job, is you cannot inspect variables in real time. So you do what mainframe programmers have been doing for decades which is print values to a log. That's OK for text, but when you use the Python shell that text is an object, which means you can further work with it. It's not a static non-entity.

The Python Pip Toolkit

Python is made rich by letting programmers contribute to its open source repository, the Python Package Index (PIP).

Sample pip packages read and write to JSON and **requests** to work with web services (It is called **HTTP for Humans**, which should give an idea of why these things exist: to make complex tasks simpler.) And more complex handle handle, well, machine learning.

Then there is Pandas and other tools to transform data from one format to another and to run these algorithms at scale, meaning across a cluster. For example, older algorithms that existed before distributed computing (i.e., big data) like scikit-learn would not work with distributed data frames and other objects run across a cluster. They are designed to work with one file on one computer. So that is an issue to keep in mind as you figure out which framework to use. But your data scientists are going to be used to it's rich set of tools. So for very large data sets you might have a hybrid of tools, depending on the skill sets of your programmers.

Data Transformation Made (Easier) Easy

The number one contribution of open source programmers to Python and data science is Pandas. Just behind that, is the Databricks CSV parser, which has now been added to Apache Spark.

This means you can type a few lines or code and have Python read a JSON, CSV, or other file type and then transform that into a Pandas table, which flattens a table out so that it is easier to read, plus it lets you apply column headings, further making things easier to read. It's as if you took a hard-to-read CSV file and laid it out in familiar spreadsheet formation. And then you can take slices of this data to rotate rows and columns, make the data smaller and thus simpler to understand, apply functions to individual cells, etc.

In short Pandas and Numpy makes working with n-dimensional matrices easier. First of all, the vast majority of machine learning problems involved some rather mundane arithmetics. This is doing matrix multiplication, aka linear algebra. While this is mundane for the computer, humans cannot even visualize once these matrices run when they have more than 3 dimensions.

Python Notebooks

Finally, Python works will in a notebook environment, due to, as we said, it's brevity and interpreted nature.

We wrote about using Jupyter here. And we have used Zeppelin in lots of other examples.

Notebooks are scratch pads for programmers. Plus it solves some logistical problems. You put the Python Notebook code (many other languages are supported in Notebooks) and into a web page and then click **run** and it runs the code. That code can include matplotlib graphs, which absent a web browser and websockets you would need an X environment (notoriously complicated) set up in

order for the programmer to share their graphs with their users.

Data scientists can distribute their graphs and summaries with these notebooks as well. This is because it lets you share these with users yet hide the code from them. And you can make real time displays.

There are of course plenty of tools and frameworks for with ML. Python just might be the most widely used one, for the many good reasons outlined above.