

Questions

Example: use argparse, when things get complicated or use argparse to control flow. How to find out how things can be modularized.

Two questions:

1. I'm using argparse, how do I search for relevant info
 - a. What I would do: search "Python argparse command line tutorial" "Python argparse integer"; "Python argparse alternatives" (Two alternatives: Click, rich...)
 - b. Python something.py --confidence 0.09 --flag true

My thoughts: if you research alternatives, you create less technical debt for the team down the road.

2. How to modularize code in Python
 - a. Search "how to package python code"

Sklearn non-linear regression

Spark, can't find a good way to do non-linear regression

SAS package has a nonlinear procedure, nonlinear regression, want to transition to Python with PySpark. Can't find this in PySpark. Can use SciPy.

1. Find how to do distributed computing for nonlinear regression.
2. Ideally, reuse the functions in SciPy without custom development. Only wrap it and distribute it.

My approach:

1. Python distribute compute function tutorial ([PySpark](#), dask, Ray)
2. Google "how do I turn a function into PySpark function" - let's say I found something and I don't really know how to use PySpark UDF
3. My strategy: build a prototype to test if this works
 - a. Reduce the data size. Create a small dataset (.csv) that resembles what I want to compute. 100 rows vs 1 billion rows
 - b. Instead of doing non linear regression. I want to if I can wrap SciPy function into PySpark function. Instead of using nonlinear, I'll use linear regression.

- c. Try to write script that performs linear regression on this 100 row dataset. Check if output makes sense
- d. Use the same structure, swap out the linear regression with nonlinear regression, run again on the 100 rows.
- e. Figure out how to run this script for 1 billion rows.

General question: how to efficiently code.

My thoughts:

1. Learn to use an IDE (VSCode, PyCharm Community) - highlights syntax errors, autocompletion if code is modular. Most useful: project wide search. VSCode can do remote development.
2. Google "how to use vscode with Python" "how to use PyCharm" "how to set up vscode for Python projects"
3. Learn to write simple python tests ->
 - a. Instead of just writing code, you need a "test script" that you can run frequently and tell if you broke the code
 - b. In Python, this is mostly done with the package "pytest"
 - c. Brian Okken's book is great

Question: Python feels overwhelming because it's open source. How do we choose the right framework given so many alternatives?

1. Consider the tradeoff in you work - is this thing urgent? Or I can take it slower and aim for better quality?
2. Don't over analyze, just start. Spend 5% of time researching alternatives. "Can I find something better with 5 mins of Googling"

What are some things that you want to do?

Write down the ideas and maybe I can help you get started