Introduction to Python for Actuaries

John Bogaardt FCAS MAAA

Brian A. Fannin ACAS CSPA

March 20, 2018

What is Python?

What is Python?

Python is a programming language invented by …

… this guy

… and he named it Python because …

... he was into Monty Python when he wrote it

Zen of Python

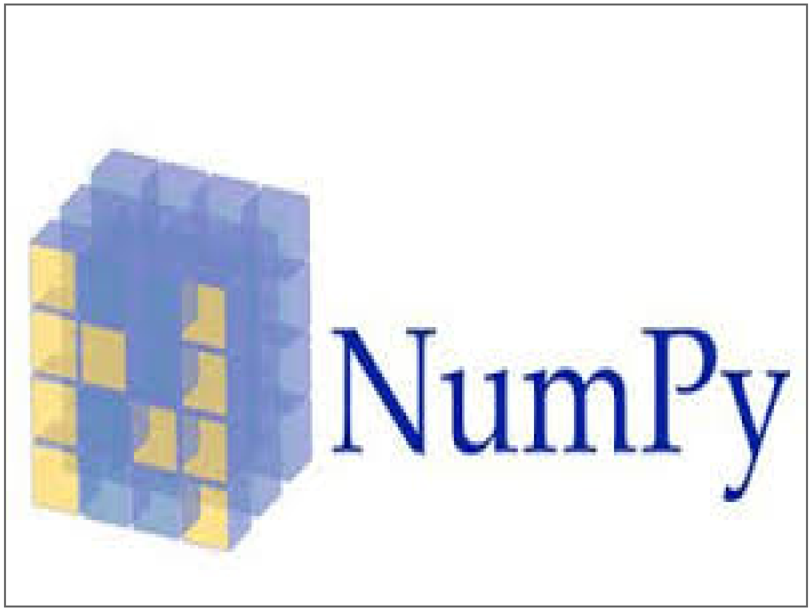
- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.

And more zen

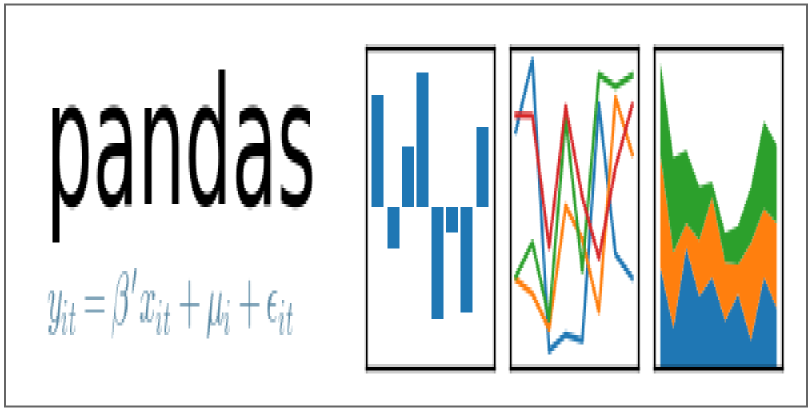
- Special cases aren’t special enough to break the rules.
- Although practicality beats purity.
- Errors should never pass silently.
- Unless explicitly silenced.
- In the face of ambiguity, refuse the temptation to guess.
- There should be one– and preferably only one –obvious way to do it.

Some of the popular packages

Data structures



- arrays
- vectorized operations



- data frames
- pivot operations

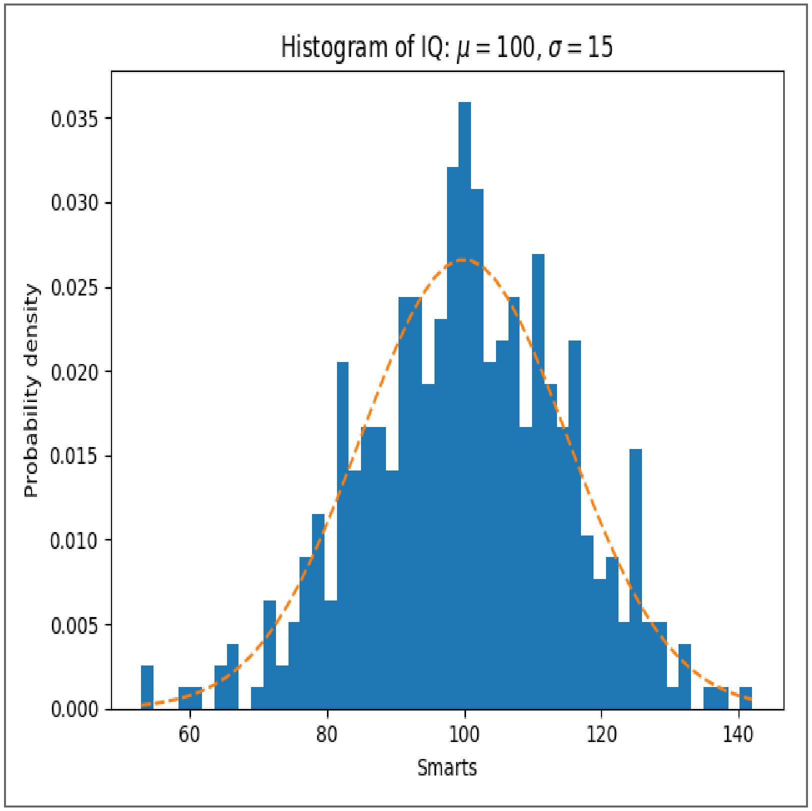
data visualization

matplotlib

- Not super exciting, but gets the job done

Other options:

- plotly
- ggplot, based on R’s ggplot2



Data from the web

Beautiful Soup

Web scraping

accessing web APIs

Stats/Machine learning

Supports common linear techniques

We'll have a lot to say about this soon!

Comparison with other platforms

Similarities between R and Python

- FLOSS - Free, Libre, Open Source Software
- Wide support (take that, Julia!)
- Interpreted, REPL
- Rich package ecosystem
- Not OS dependent
- Execution not always as fast as C, but possible to use C routines when needed
- Great database support - from RDBMS to Spark, Hadoop, etc.

What’s different from R?

R …

- loves statistics
- hasn’t really settled on OOP
- vector support out of the box
- does anyone really use `try()`?
- has a few actuarial packages
- RStudio!!!
- lacks consensus around machine learning - H2O, caret, ROCR?

Python …

- is general purpose
- has easy and consistent support for OOP
- vectors are available in `numpy`
- great exception handling
- not much actuarial focus
- no consensus on a FLOSS IDE (spyder?)
- scikit-learn!

Compare with Excel?

Excel …

- Closed source
- Data is visible, but logic is hidden
- Easy to produce ‘camera-ready’ exhibits

Python …

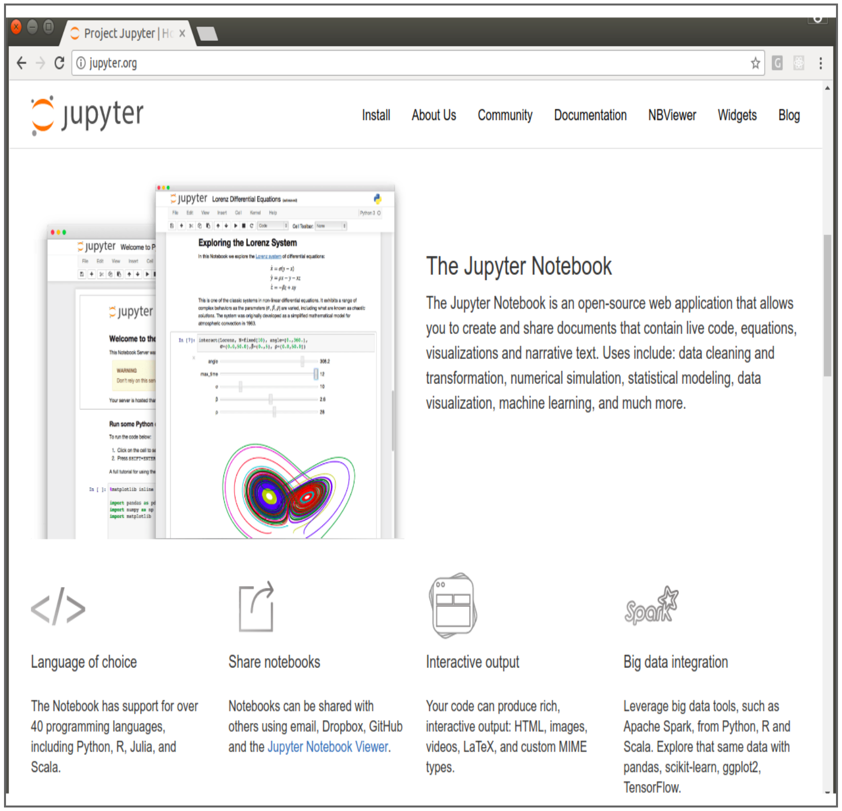
- FLOSS
- Logic is visible, data is abstract
- Emphasis on calculation rather than presentation

How can I incorporate Python into my workflow?

Practicalities

- Version 2 or 3?
  - Used to be an issue, but not much anymore
  - Version 2 is going away after a **VERY** long transition
- Editor?
  - Can just code at the console or copy and paste
  - A few decent FLOSS choices: Spyder, Rodeo
  - The best options are commercial
- Can I e-mail a python file to a co-worker?

Jupyter



The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Language of choice

The Notebook has support for over 40 programming languages, including Python, R, Julia, and Scala.

Share notebooks

Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.

Interactive output

Your code can produce rich, interactive output: HTML, images, videos, LaTeX, and custom MIME types.

Big data integration

Leverage big data tools, such as Apache Spark, from Python, R and Scala. Explore that same data with pandas, scikit-learn, ggplot2, TensorFlow.

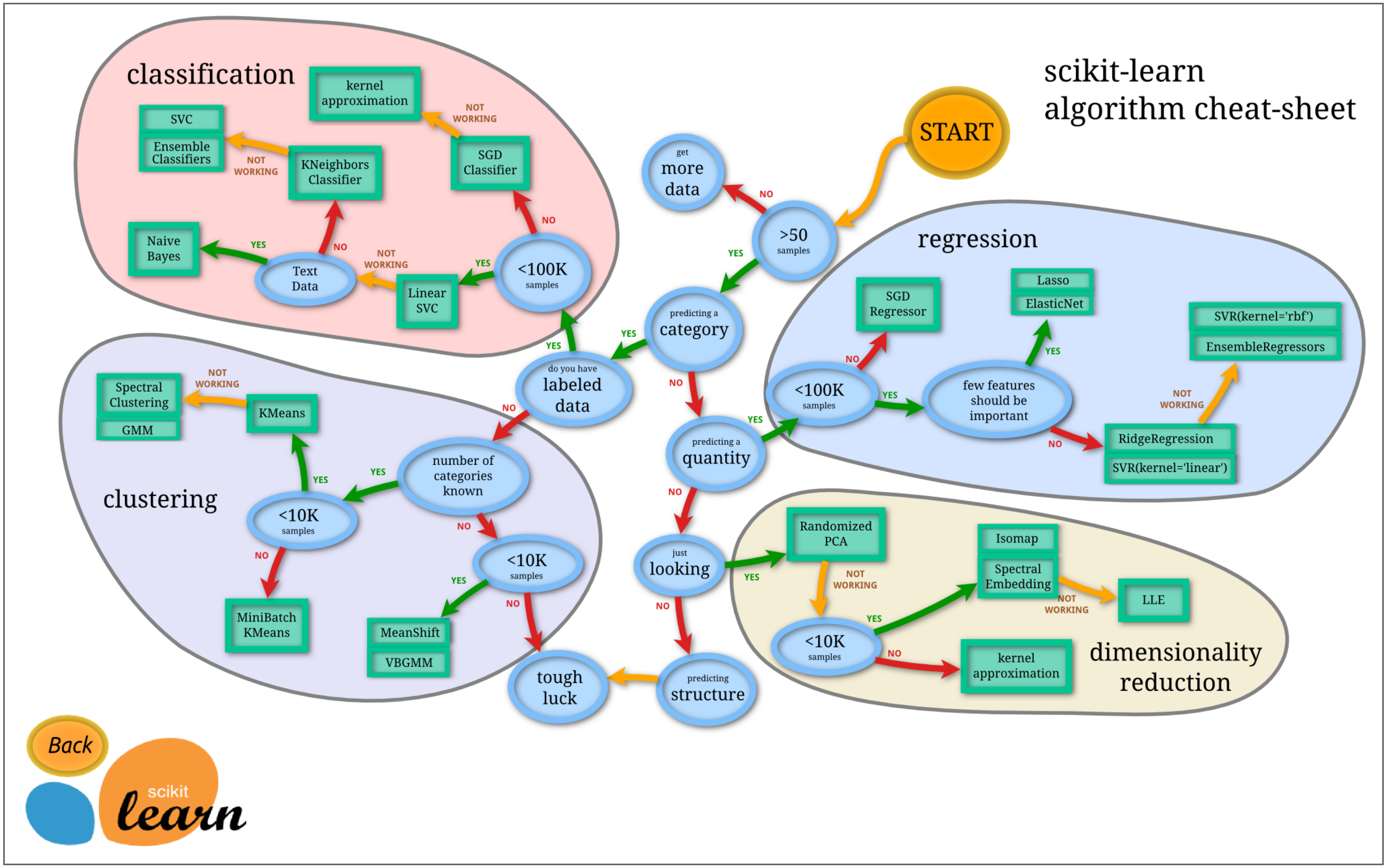
- Framework for interactive use of programming languages
- Supports for several different languages
- Files - “notebooks” - may be shared with other users
- Methodology is transparent and reproducible

Intro scikit learn

scikit-learn

- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license
- Common functional interface for:
  - Data splits, cross validations
  - Data transforms
  - Training, test, scoring
  - Pipelines to manage workflow
  - GridSearch for model tuning

But what algorithms does it support?



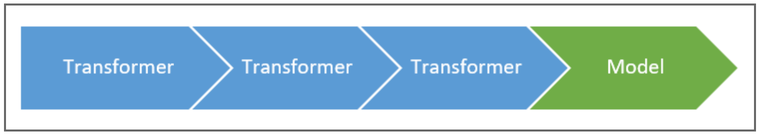
Estimators

- Estimators are the building block of scikit-learn. Almost everything is an estimator.
- All estimators have `fit()` and `predict()` methods.
- Supervised techniques generally have a `score()` method as well.

Transformers

- In addition to regressors, classifiers, or clustering techniques, scikit-learn includes transformers.
- These are used to generate new features, clean data, and extract information from your datasets.
- Transformers are estimators too and have fit() methods Transformers utilize the transform() method to ‘transform’ new data.

Pipelines



- Preprocess (`transform()`) my data
- Dimensionality reduction (PCA, cluster, etc.)
- Steps may be cached to save memory
- Awesome for grid search to shrink the potential parameter space
- The final step of the pipeline is `fit()`

Add it up

1. Estimators are objects which have a common function interface
2. I can transform my data to make it ready to fit
3. I can easily swap between estimators knowing that they all have `fit()` and `predict()` methods
4. I capture my workflow in a pipeline

How about a live demo

Wrapping up

Python and you

- If you already know R, check out Python.
- If you're ready for something beyond spreadsheets, Python is a great place to start.
- scikit-learn standardizes use of a wide variety of machine learning techniques. Learn once -> use many.

Thank you for your time!

Questions?

References

- **http://scikit-learn.org/stable/**
- **https://www.python.org/**
- **https://www.python.org/dev/peps/pep-0020/**
- **Data Camp numpy cheat sheet**
- **http://docs.python-requests.org/en/master**
- **https://pandas.pydata.org/about.html**
- **http://www.statsmodels.org/stable/index.html**
- **http://ggplot.yhathq.com**