CHAPTER 27

Go Forth and Do Data Science

And now, once again, I bid my hideous progeny go forth and prosper.
—Mary Shelley

Where do you go from here? Assuming I haven't scared you off of data science, there are a number of things you should learn next.

IPython

I mentioned IPython earlier in the book. It provides a shell with far more functionality than the standard Python shell, and it adds "magic functions" that allow you to (among other things) easily copy and paste code (which is normally complicated by the combination of blank lines and whitespace formatting) and run scripts from within the shell.

Mastering IPython will make your life far easier. (Even learning just a little bit of IPython will make your life a lot easier.)

In the first edition, I also recommended that you learn about the IPython (now Jupyter) Notebook, a computational environment that allows you to combine text, live Python code, and visualizations.

I've since become a notebook skeptic, as I find that they confuse beginners and encourage bad coding practices. (I have many other reasons too.) You will surely receive plenty of encouragement to use them from people who aren't me, so just remember that I'm the dissenting voice.

Mathematics

Throughout this book, we dabbled in linear algebra (Chapter 4), statistics (Chapter 5), probability (Chapter 6), and various aspects of machine learning.

To be a good data scientist, you should know much more about these topics, and I encourage you to give each of them a more in-depth study, using the textbooks recommended at the ends of the chapters, your own preferred textbooks, online courses, or even real-life courses.

Not from Scratch

Implementing things "from scratch" is great for understanding how they work. But it's generally not great for performance (unless you're implementing them specifically with performance in mind), ease of use, rapid prototyping, or error handling.

In practice, you'll want to use well-designed libraries that solidly implement the fundamentals. My original proposal for this book involved a second "now let's learn the libraries" half that O'Reilly, thankfully, vetoed. Since the first edition came out, Jake VanderPlas has written the *Python Data Science Handbook* (O'Reilly), which is a good introduction to the relevant libraries and would be a good book for you to read next.

NumPy

NumPy (for "Numeric Python") provides facilities for doing "real" scientific computing. It features arrays that perform better than our `list`-vectors, matrices that perform better than our `list`-of-`list`-matrices, and lots of numeric functions for working with them.

NumPy is a building block for many other libraries, which makes it especially valuable to know.

pandas

pandas provides additional data structures for working with datasets in Python. Its primary abstraction is the `DataFrame`, which is conceptually similar to the NotQuiteABase `Table` class we constructed in Chapter 24, but with much more functionality and better performance.

If you're going to use Python to munge, slice, group, and manipulate datasets, pandas is an invaluable tool.

scikit-learn

scikit-learn is probably the most popular library for doing machine learning in Python. It contains all the models we've implemented and many more that we haven't. On a real problem, you'd never build a decision tree from scratch; you'd let scikit-learn do the heavy lifting. On a real problem, you'd never write an optimization algorithm by hand; you'd count on scikit-learn to already be using a really good one.

Its documentation contains many, many examples of what it can do (and, more generally, what machine learning can do).

Visualization

The matplotlib charts we've been creating have been clean and functional but not particularly stylish (and not at all interactive). If you want to get deeper into data visualization, you have several options.

The first is to further explore matplotlib, only a handful of whose features we've actually covered. Its website contains many examples of its functionality and a gallery of some of the more interesting ones. If you want to create static visualizations (say, for printing in a book), this is probably your best next step.

You should also check out seaborn, which is a library that (among other things) makes matplotlib more attractive.

If you'd like to create *interactive* visualizations that you can share on the web, the obvious choice is probably D3.js, a JavaScript library for creating "data-driven documents" (those are the three Ds). Even if you don't know much JavaScript, it's often possible to crib examples from the D3 gallery and tweak them to work with your data. (Good data scientists copy from the D3 gallery; great data scientists *steal* from the D3 gallery.)

Even if you have no interest in D3, just browsing the gallery is itself a pretty incredible education in data visualization.

Bokeh is a project that brings D3-style functionality into Python.

R

Although you can totally get away with not learning R, a lot of data scientists and data science projects use it, so it's worth getting at least familiar with it.

In part, this is so that you can understand people's R-based blog posts and examples and code; in part, this is to help you better appreciate the (comparatively) clean elegance of Python; and in part, this is to help you be a more informed participant in the never-ending "R versus Python" flamewars.

Deep Learning

You can be a data scientist without doing deep learning, but you can't be a *trendy* data scientist without doing deep learning.

The two most popular deep learning frameworks for Python are TensorFlow (created by Google) and PyTorch (created by Facebook). The internet is full of tutorials for them that range from wonderful to awful.

TensorFlow is older and more widely used, but PyTorch is (in my opinion) much easier to use and (in particular) much more beginner-friendly. I prefer (and recommend) PyTorch, but—as they say—no one ever got fired for choosing TensorFlow.

Find Data

If you're doing data science as part of your job, you'll most likely get the data as part of your job (although not necessarily). What if you're doing data science for fun? Data is everywhere, but here are some starting points:

- Data.gov is the government's open data portal. If you want data on anything that has to do with the government (which seems to be most things these days), it's a good place to start.
- Reddit has a couple of forums, r/datasets and r/data, that are places to both ask for and discover data.
- Amazon.com maintains a collection of public datasets that they'd like you to analyze using their products (but that you can analyze with whatever products you want).
- Robb Seaton has a quirky list of curated datasets on his blog.
- Kaggle is a site that holds data science competitions. I never managed to get into it (I don't have much of a competitive nature when it comes to data science), but you might. They host a lot of datasets.
- Google has a newish Dataset Search that lets you (you guessed it) search for datasets.

Do Data Science

Looking through data catalogs is fine, but the best projects (and products) are ones that tickle some sort of itch. Here are a few that I've done.

Hacker News

Hacker News is a news aggregation and discussion site for technology-related news. It collects lots and lots of articles, many of which aren't interesting to me.

Accordingly, several years ago, I set out to build a Hacker News story classifier to predict whether I would or would not be interested in any given story. This did not go over so well with the users of Hacker News, who resented the idea that someone might not be interested in every story on the site.

This involved hand-labeling a lot of stories (in order to have a training set), choosing story features (for example, words in the title, and domains of the links), and training a Naive Bayes classifier not unlike our spam filter.

For reasons now lost to history, I built it in Ruby. Learn from my mistakes.

Fire Trucks

For many years I lived on a major street in downtown Seattle, halfway between a fire station and most of the city's fires (or so it seemed). Accordingly, I developed a recreational interest in the Seattle Fire Department.

Luckily (from a data perspective), they maintain a Real-Time 911 site that lists every fire alarm along with the fire trucks involved.

And so, to indulge my interest, I scraped many years' worth of fire alarm data and performed a social network analysis of the fire trucks. Among other things, this required me to invent a fire-truck-specific notion of centrality, which I called TruckRank.

T-Shirts

I have a young daughter, and an incessant source of frustration to me throughout her childhood has been that most "girls' shirts" are quite boring, while many "boys' shirts" are a lot of fun.

In particular, it felt clear to me that there was a distinct difference between the shirts marketed to toddler boys and toddler girls. And so I asked myself if I could train a model to recognize these differences.

Spoiler: I could.

This involved downloading the images of hundreds of shirts, shrinking them all to the same size, turning them into vectors of pixel colors, and using logistic regression to build a classifier.

One approach looked simply at which colors were present in each shirt; a second found the first 10 principal components of the shirt image vectors and classified each

shirt using its projections into the 10-dimensional space spanned by the "eigenshirts" (Figure 27-1).

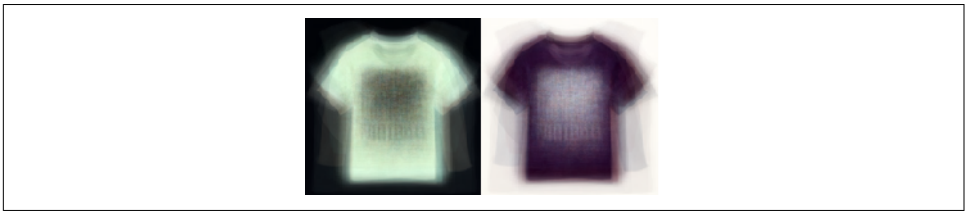

Figure 27-1. Eigenshirts corresponding to the first principal component

Tweets on a Globe

For many years I'd wanted to build a "spinning globe" visualization. During the 2016 election, I built a small web app that listened for geotagged tweets matching some search (I used "Trump," as it appeared in lots of tweets at that time), displayed them, and spun a globe to their location as they appeared.

This was entirely a JavaScript data project, so maybe learn some JavaScript.

And You?

What interests you? What questions keep you up at night? Look for a dataset (or scrape some websites) and do some data science.

Let me know what you find! Email me at *joelgrus@gmail.com* or find me on Twitter at *@joelgrus*.