Data Science from Scratch

First Principles with Python

Stephen CUI

March 20, 2023

Contents

Chapter 1

Working with Data

1.1 Exploring Your Data

After you've identified the questions you're trying to answer and have gotten your hands on some data, you might be tempted to dive in and immediately start building models and getting answers. But you should resist this urge. Your first step should be to explore your data.

1.1.1 Exploring One-Dimensional Data

An obvious first step is to compute a few summary statistics. You'd like to know how many data points you have, the smallest, the largest, the mean, and the standard deviation.

But even these don't necessarily give you a great understanding. A good next step is to create a histogram, in which you group your data into discrete buckets and count how many points fall into each bucket.

1.1.2 Two Dimensions

1.1.3 Many Dimensions

With many dimensions, you'd like to know how all the dimensions relate to one another. A simple approach is to look at the *correlation matrix*, in which the entry in row i and column j is the correlation between the ith dimension and the jth dimension of the data.

A more visual approach (if you don't have too many dimensions) is to make a scatterplot matrix Figure 1.1 showing all the pairwise scatterplots.

1.2 Using NamedTuples

One common way of representing data is using dicts. There are several reasons why this is less than ideal, however. This is a slightly inefficient representation (a dict involves some overhead), so that if you have a lot of data they'll take up more memory than they have to. For the most part, this is a minor consideration. A larger issue is that accessing things by dict key is error-prone. Finally, while we can type-annotate uniform dictionaries, there's no helpful way to annotate dictionaries-as-data that have lots of different value types.

As an alternative, Python includes a namedtuple (Returns a new tuple subclass named typename.) class, which is like a tuple but with named slots. Like regular tuples, namedtuples are immutable, which means that you can't modify their values once they're created. (不能修改)

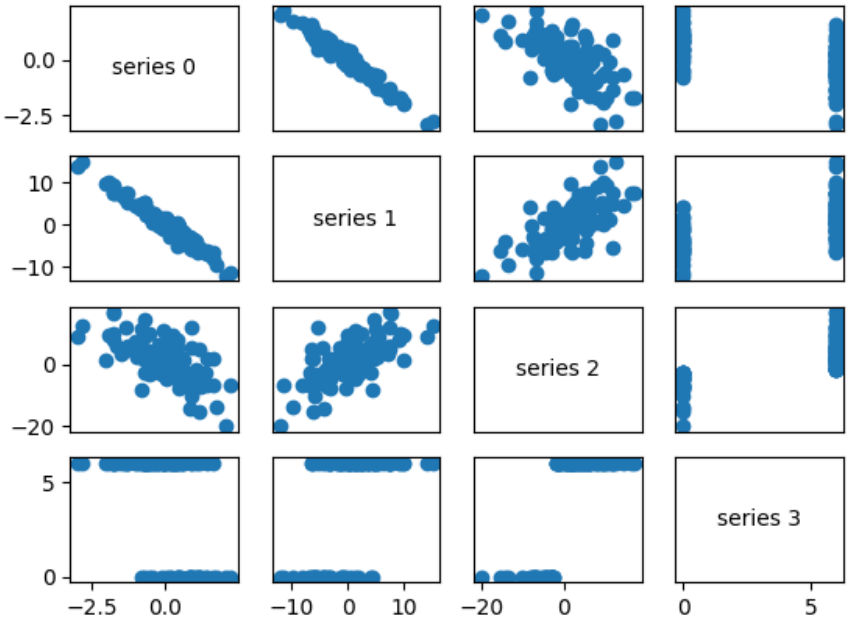

Figure 1.1: Scatterplot matrix

You'll notice that we still haven't solved the type annotation issue. We do that by using the typed variant, NamedTuple. namedtuple子类仍然不能解决代码提示问题

1.3 Dataclasses

Dataclasses are (sort of) a mutable version of NamedTuple. (“sort of” because NamedTuples represent their data compactly as tuples, whereas dataclasses are regular Python classes that simply generate some methods for you automatically.)

The syntax is very similar to NamedTuple. But instead of inheriting from a base class, we use a decorator. The big difference is that we can modify a dataclass instance's values. If we tried to modify a field of the NamedTuple version, we'd get an `AttributeError`.

This also leaves us susceptible to the kind of errors we were hoping to avoid by not using dicts.