* [A Quick Guide to Reading Technical Documentation](#gjdgxs)
  + [Tutorials](#30j0zll)
  + [Reference Pages](#1fob9te)
* [Example](#3znysh7)

# CSE 450 - Machine Learning & Data Mining

## Reading Technical Documentation

## A Quick Guide to Reading Technical Documentation

Reading technical documentation isn't usually like reading a novel or news article. It can take time before you're comfortable enough to be able to quickly scan through a reference page and quickly glean what you need for a particular situation.

Broadly speaking, there are three types of technical documentation: tutorials, reference pages, and help sites.

We'll cover the first two types here, and you can read more about how to use help sites on the [Getting Help page](http://docs.google.com/getting-help.html).

Framework vs Library vs Whatever

Some people get pretty hung up on the differences between "frameworks", "libraries", "APIs", etc... For our purposes (and frankly for most purposes), these differences don't matter that much.

In order to save some typing, I'll just refer to everything as a "library".

### Tutorials

Tutorials go by a variety of names: quick starts, getting started guides, variations on "Learn the Basics", anything that says "X minutes" or "X hours".

Tutorials are usually designed to introduce you to the overall philosophy behind a library's design. They focus on teaching you how to use the library to solve a specific problem.

To get the most out of a tutorial, you need to go beyond the surface a bit. Particularly, you should pay attention to the general flow of how the different parts of a library fit together. Well designed libraries usually have a variety of functions that follow the same general pattern of use.

3rd Party Tutorials

There are usually a ton of "3rd party" tutorials for the more popular libraries, including online tutorials, books, and videos.

3rd Party Tutorials are those created by someone who isn't directly involved in the library's design or maintenance. You'll see a lot of them on YouTube. These tutorials can be quite good, but there are two things you have to be careful of:

1. Sometimes, 3rd party tutorials will teach you bad practices or shortcuts, because the person creating the tutorial has only a basic understanding of the concepts behind the library's design.
2. Often, 3rd party tutorials are outdated or are based on outdated techniques, especially in a field that moves as quickly as machine learning. Quite often, libraries will add new techniques, pipelines, or capabilities to make some of the most common uses of that library easier or more powerful.

The best way to avoid these problems is to focus on official tutorials created by the individuals directly involved in the creation and maintenance of the library.

### Reference Pages

Reference pages are sometimes also called documentation, manual, user manual, API reference, and a few others.

Reference manuals are not really designed to be read straight through like a tutorial. You *can* do that, and will likely learn a lot, but chances are you won't have much fun.

Instead, once you've learned the general design philosophy of a library, you can look up particular functions when you need them.

When reading a reference page, the most critical thing is not to gloss over words you don't understand. Unlike tutorials, reference pages aren't usually chatty, and typically every word matters.

If you come a cross a term you don't understand, stop and look it up before continuing. As time goes on, the number of terms you have to look up will decrease.

## Example

Let's say I wanted to perform a Linear Regression with Python and someone tells me to use the SciKit-Learn library.

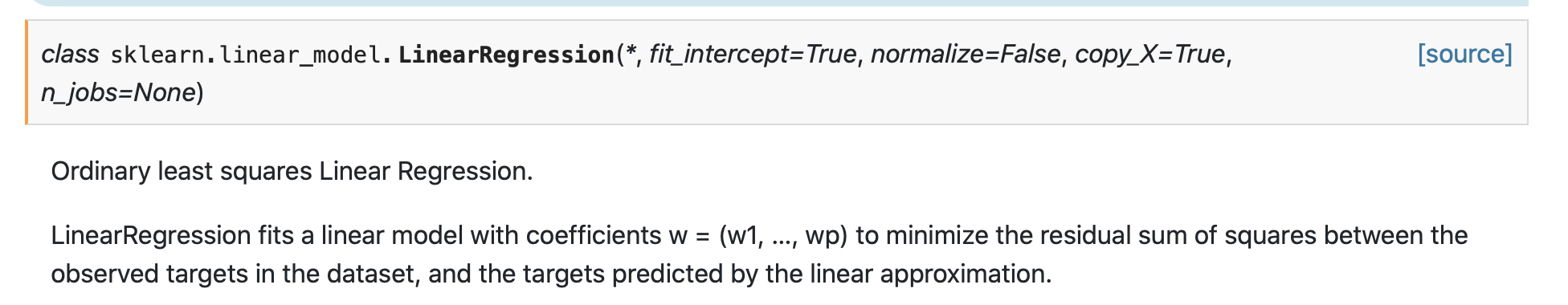
I'd start by reading through the [Getting Started Guide](https://scikit-learn.org/stable/getting_started.html) and [Introductory Tutorial](https://scikit-learn.org/stable/tutorial/basic/tutorial.html) to understand how the library worked.

Once I felt like I understood the library's overall design philosophy, I'd jump over to the [tutorial page for Linear Regression](https://scikit-learn.org/stable/modules/linear_model.html#ordinary-least-squares).

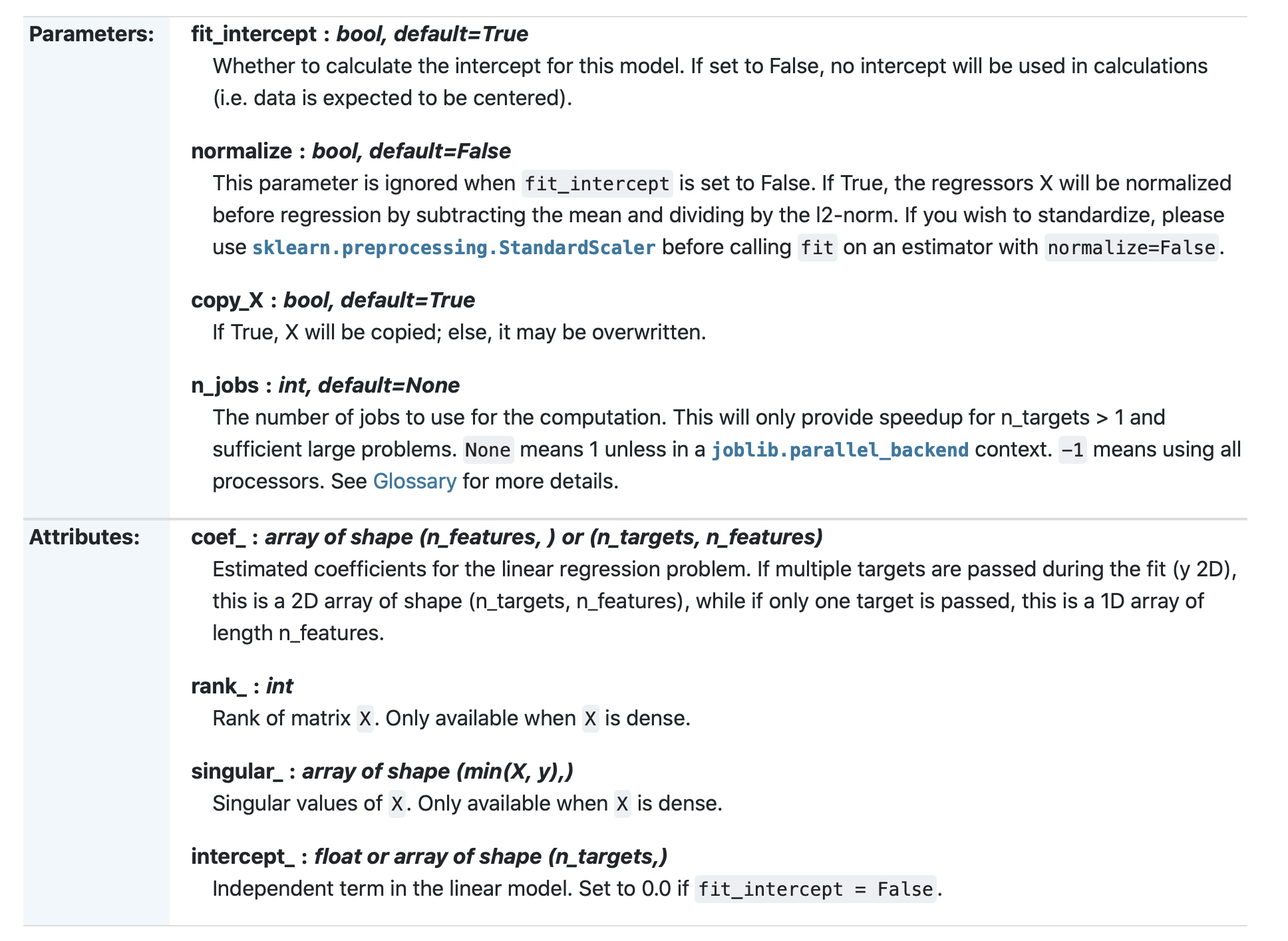
That page links to two import things. First, I'd look at [the more detailed example](https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py).

Next, I'd go to the [Reference Page for the Linear Regression API](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html).

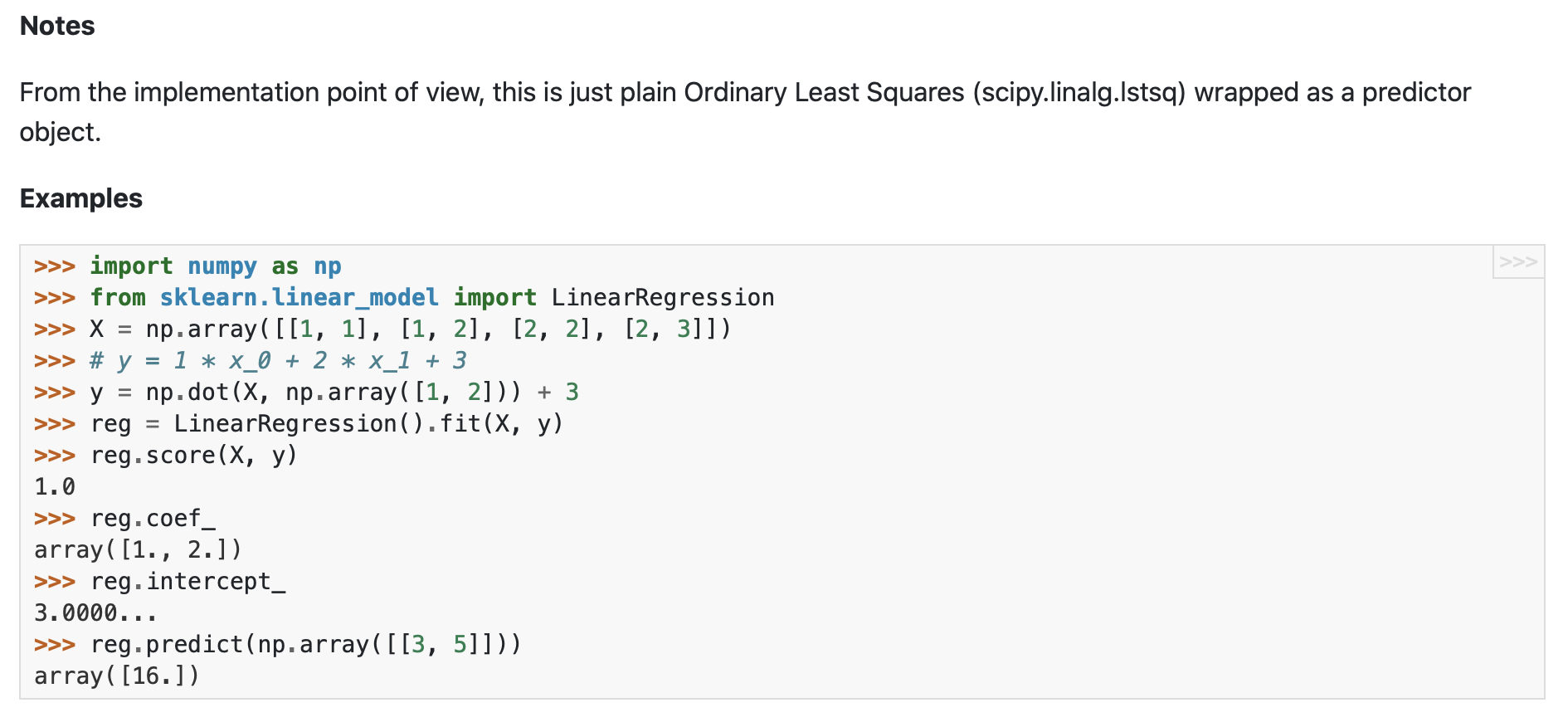
On the reference page, I'd first note that this is a class:



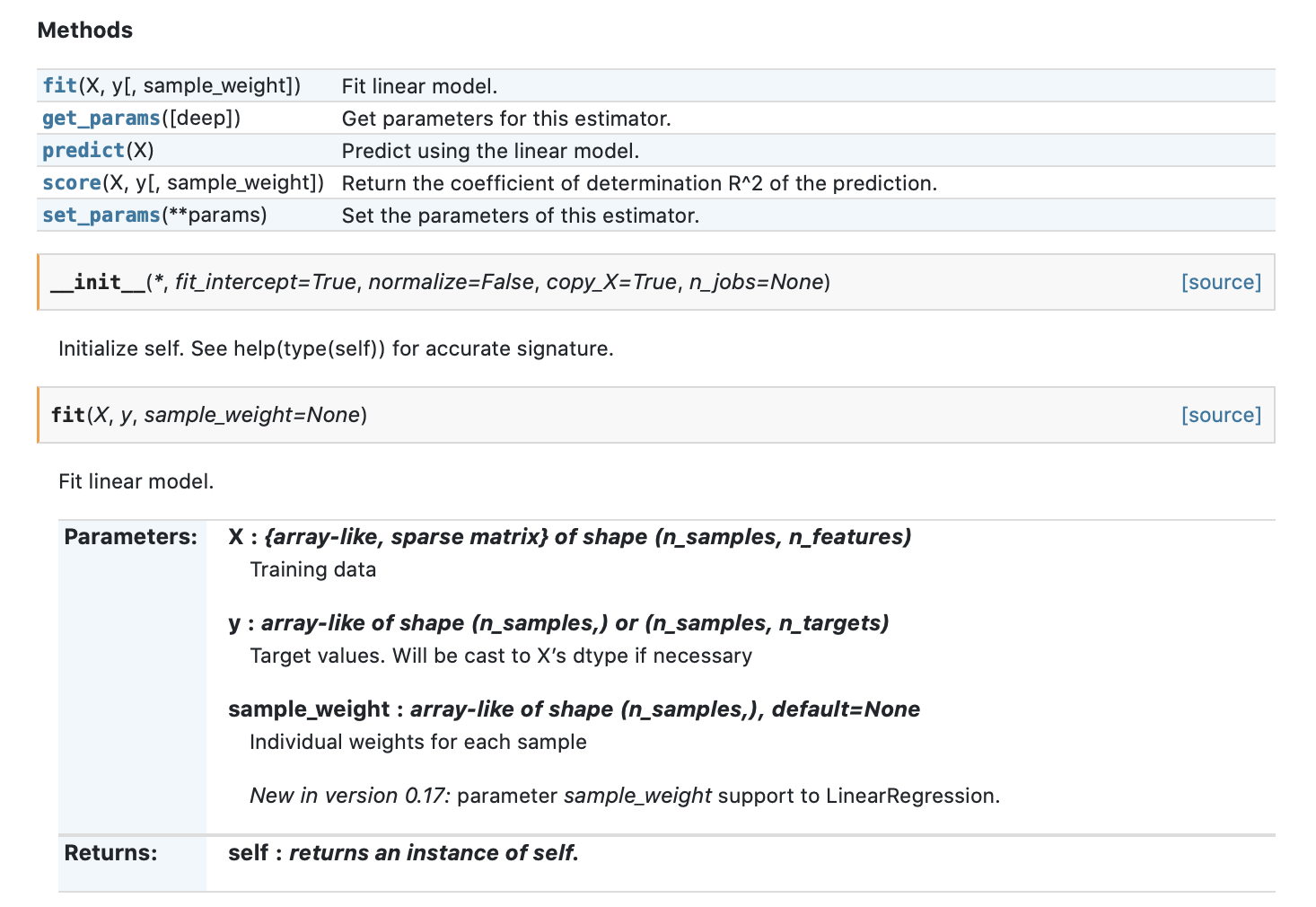
This means I'd create an instance of the class with certain parameters, and have access to a variety of attributes. I'd read through those and pay special attention to what the default values were, and how they affected the regression.



I'd then read the notes and the little example to see a simple example of how this library worked:

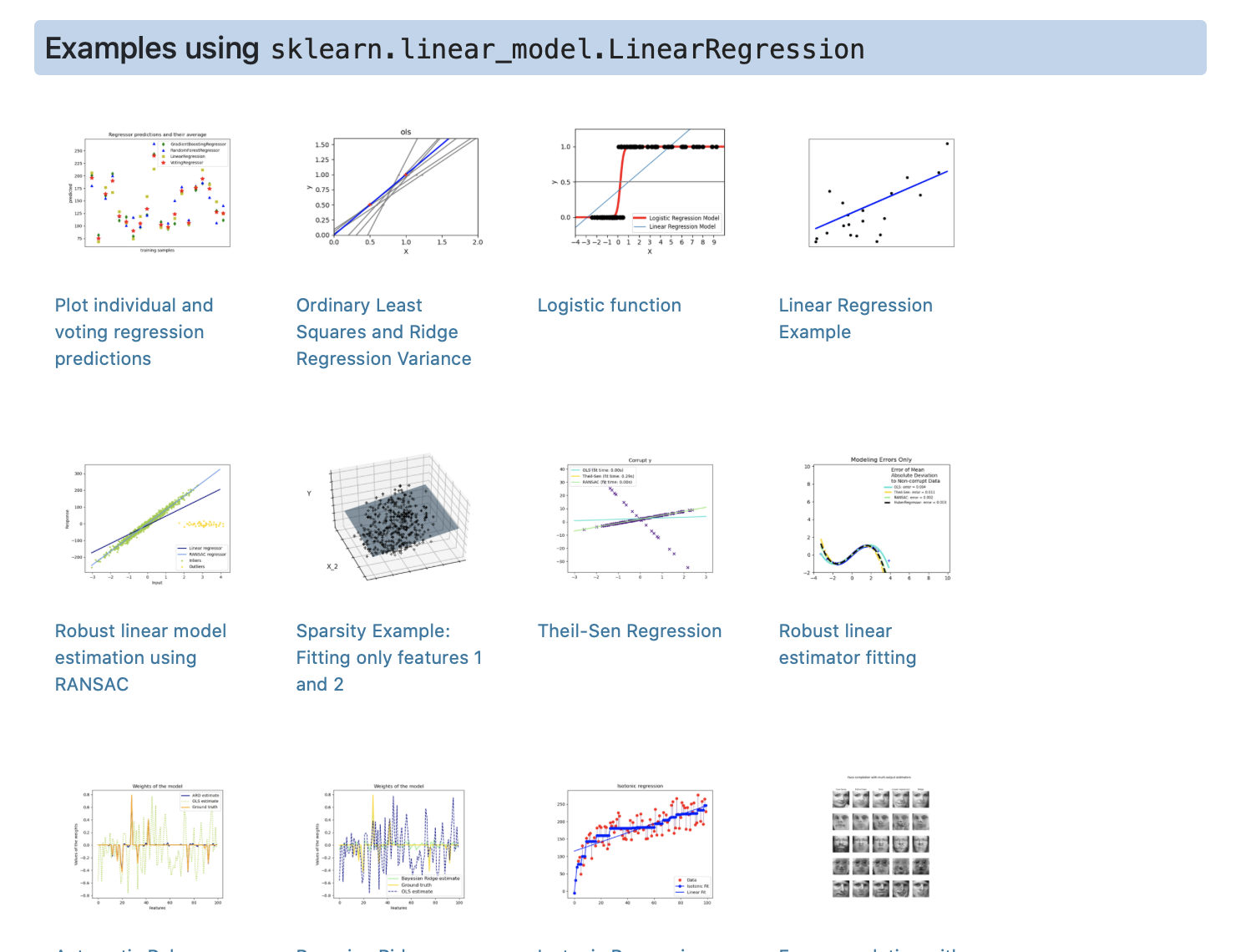


I'd then read through the method list and for each method I'd read what the method did, what parameters and default values it used, and what it returned.



Once I had a good understanding of the methods and parameters, I'd go *back* to [the more detailed example](https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py) I looked at earlier, and pay more attention to what values are being passed to (and returned from) different methods.

Finally, I'd browse through the more extended examples to see if there were any variations that looked interesting or relevant. If so, I'd click on each of those and read through what they had to say.



Only then would I start writing my own code.