How to split your dataset to train and test datasets using SciKit Learn

Sunny Srinidhi Follow
Jul 30, 2018 · 3 min read

> *Orignially published here: http://blog.contactsunny.com/data-science/how-to-split-your-dataset-to-train-and-test-datasets*

. . .

When you're working on a model and want to train it, you obviously have a dataset. But after training, we have to test the model on some test dataset. For this, you'll a dataset which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase.

In such cases, the obviously solution is to split the dataset you have into two sets, one for training and the other for testing; and you do this before you start training your model.

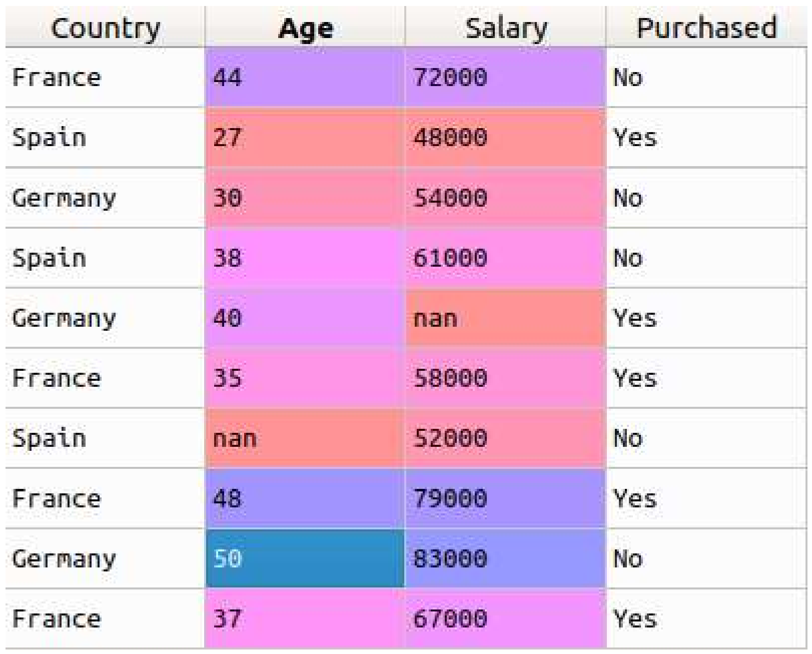
But the question is, how do you split the data? You can't possibly manually split the dataset into two. And you also have to make sure you split the data in a random manner. To help us with this task, the SciKit library provides a tool, called the Model Selection library. There's a class in the library which is, aptly, named 'train_test_split.' Using this we can easily split the dataset into the training and the testing datasets in various proportions.

There are a few parameters that we need to understand before we use the class:

- **test_size**—This parameter decides the size of the data that has to be split as the test dataset. This is given as a fraction. For example, if you pass 0.5 as the value, the dataset will be split 50% as the test dataset. If you're specifying this parameter, you can ignore the next parameter.

- **train_size**—You have to specify this parameter only if you're not specifying the test_size. This is the same as test_size, but instead you tell the class what percent of the dataset you want to split as the training set.
- **random_state**—Here you pass an integer, which will act as the seed for the random number generator during the split. Or, you can also pass an instance of the RandomState class, which will become the number generator. If you don't pass anything, the RandomState instance used by np.random will be used instead.

As an example, let's consider the same dataset that we've considered in our previous examples. I've given it here for reference:



Country	Age	Salary	Purchased
France	44	72000	No
Spain	27	48000	Yes
Germany	30	54000	No
Spain	38	61000	No
Germany	40	nan	Yes
France	35	58000	Yes
Spain	nan	52000	No
France	48	79000	Yes
Germany	50	83000	No
France	37	67000	Yes

We split this into two different datasets, one for the independent features—x, and one for the dependent variable—y (which is the last column). We'll now split the dataset x into two separate sets—xTrain and xTest. Similarly, we'll split the dataset y into two sets as well—yTrain and yTest. Doing this using the sklearn library is very simple. Let's look at the code:

```
from sklearn.model_selection import train_test_split
xTrain, xTest, yTrain, yTest = train_test_split(x, y,
test_size = 0.2, random_state = 0)
```

As you can see from the code, we have split the dataset in a 80–20 ratio, which is a common practice in data science. For a change, I'll not give the output here. Try this out for yourself and see how the new datasets are.