**Microsoft:** DAT210x Programming with Python for Data Science

Bookmarks

5. Data Modeling > Lecture: Splitting Data > SciKit-Learn Implementation

Bookmark

## SciKit-Learn Implementation

SciKit-Learn helps you split your data:

```
>>> from sklearn.cross_validation import train_test_split
>>> data   = [0,1,2,3,4, 5,6,7,8,9]  # input dataframe samples
>>> labels = [0,0,0,0,0, 1,1,1,1,1]  # the function we're training is " >4 "

>>> data_train, data_test, label_train, label_test = train_test_split(data, labels,
test_size=0.5, random_state=7)

>>> data_train
[9, 7, 3, 6, 4]

>>> label_train
[1, 1, 0, 1, 0]

>>> data_test
[8, 5, 0, 2, 1]

>>> label_test
[1, 1, 0, 0, 0]
```

Notice how your data is held separately from your labels. This is important! It's difficult enough for your machines to figure out how to mathematically get from your raw data to your labels without the two being inter-jumbled. We've used `df` as a shorthand for dataframe but in the future we will start using a more conventional `X` for our data, and `y` for our answer labels.

Lab

Lecture: Regression
Quiz

Lab: Regression
Lab

Dive Deeper

6. Data Modeling II

Unless you specify otherwise, it'll hold back 25% of your data in the validation or testing set, and 75% will stay in the original training set. Each time you run `train_test_splot()` it holds back a randomly shuffled amount of data, so one thing you'll start noticing is that successive trials of your algorithm may actually produce slightly different accuracy levels. This is normal, so do not be alarmed. If you absolutely need the results to come back identically, such as if you're doing a demo, then you can pass in an optional random_state variable to make the centroid selection reproducible.

After you've trained your model against the **training** data (`data_train, label_train`), the next step is testing it. You'll use the `.predict()` method of your model, passing in the **testing** data (`data_test`) to come create an array of predictions. And then you'll gauge its accuracy against the *true* label_test answers. SciKit-Learn also has a method to help you do that:

```
from sklearn.metrics import accuracy_score

# Returns an array of predictions:
>>> predictions = my_model.predict(data_test)
>>> predictions
[0, 0, 0, 1, 0]

# The actual answers:
>>> label_train
[1, 1, 0, 1, 0]

>>> accuracy_score(label_train, predictions)
0.59999999999999998

>>> accuracy_score(label_train, predictions, normalize=False)
3
```

We covered a lot of important material here that you'll want to make sure gets added to your Evaluating phase on the **course map**. You'll only know if your models are performing well after evaluating them, so being able to reliably grade your models is a precursor to driving intelligence from your raw data.