Images haven't loaded yet. Please exit printing, wait for images to load, and try to print again.

Sunny Srinidhi Follow
Jul 29 · 4 min read

> *Originally published here: http://blog.contactsunny.com/data-science/label-encoder-vs-one-hot-encoder-in-machine-learning*

. . .

If you're new to Machine Learning, you might get confused between these two—Label Encoder and One Hot Encoder. These two encoders are parts of the SciKit Learn library in Python, and they are used to convert categorical data, or text data, into numbers, which our predictive models can better understand. Today, let's understand the difference between the two with a simple example.

. . .

Label Encoding

To begin with, you can find the SciKit Learn documentation for Label Encoder here. Now, let's consider the following data:

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

Data from SuperDataScience

In this example, the first column is the country column, which is all text. As you might know by now, we can't have text in our data if we're going to run any kind of model on it. So before we can run a model, we need to make this data ready for the model.

And to convert this kind of categorical text data into model-understandable numerical data, we use the Label Encoder class. So all we have to do, to label encode the first column, is import the LabelEncoder class from the sklearn library, fit and transform the first column of the data, and then replace the existing text data with the new encoded data. Let's have a look at the code.

```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
x[:, 0] = labelencoder.fit_transform(x[:, 0])
```

We've assumed that the data is in a variable called 'x'. After running this piece of code, if you check the value of x, you'll see that the three countries in the first column have been replaced by the numbers 0, 1, and 2.

```
array([[0, 44.0, 72000.0],
       [2, 27.0, 48000.0],
       [1, 30.0, 54000.0],
       [2, 38.0, 61000.0],
       [1, 40.0, nan],
       [0, 35.0, 58000.0],
       [2, nan, 52000.0],
       [0, 48.0, 79000.0],
       [1, 50.0, 83000.0],
       [0, 37.0, 67000.0]], dtype=object)
```

That's all label encoding is about. But depending on the data, label encoding introduces a new problem. For example, we have encoded a set of country names into numerical data. This is actually categorical data and there is no relation, of any kind, between the rows.

The problem here is, since there are different numbers in the same column, the model will misunderstand the data to be in some kind of order, $0 < 1 < 2$. But this isn't the case at all. To overcome this problem, we use One Hot Encoder.

• • •

One Hot Encoder

If you're interested in checking out the documentation, you can find it here. Now, as we already discussed, depending on the data we have, we might run into situations where, after label encoding, we might confuse our model into thinking that a column has data with some kind of order or hierarchy, when we clearly don't have it. To avoid this, we 'OneHotEncode' that column.

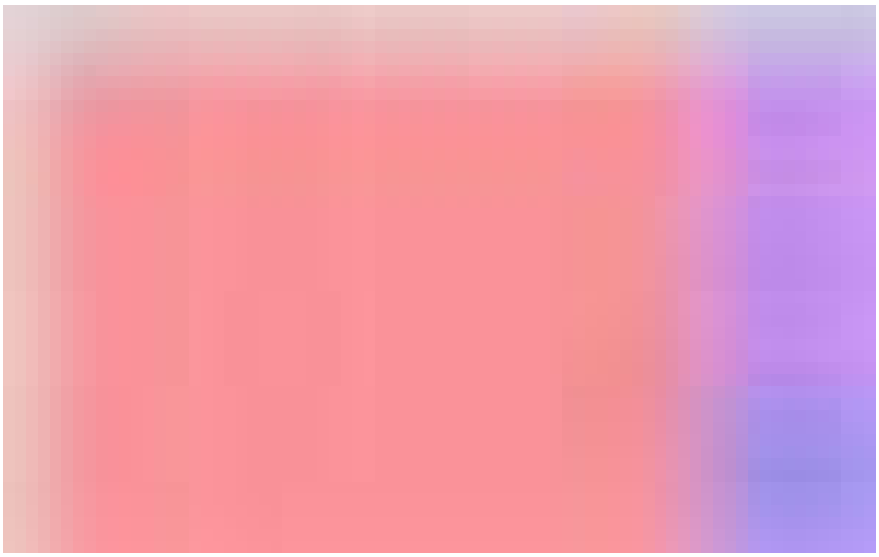
What one hot encoding does is, it takes a column which has categorical data, which has been label encoded, and then splits the column into multiple columns. The numbers are replaced by 1s and 0s, depending on which column has what value. In our example, we'll get three new columns, one for each country—France, Germany, and Spain.

For rows which have the first column value as France, the 'France' column will have a '1' and the other two columns will have '0's. Similarly, for rows which have the first column value as Germany, the 'Germany' column will have a '1' and the other two columns will have '0's.

The Python code for one hot encoding is also pretty simple:

```
from sklearn.preprocessing import OneHotEncoder
onehotencoder = OneHotEncoder(categorical_features = [0])
x = onehotencoder.fit_transform(x).toarray()
```

As you can see in the constructor, we specify which column has to be one hot encoded, [0] in this case. Then we fit and transform the array 'x' with the onehotencoder object we just created. And that's it, we now have three new columns in our dataset:



As you can see, we have three new columns with 1s and 0s, depending on the country that the rows represent.

So, that's the difference between Label Encoding and One Hot Encoding.