

[Outline](#) > 2. Data And Features > Lecture: Feature Representation > Video

## Video

[Bookmark this page](#)

## Feature Representation

[Start of transcript. Skip to the end.](#)

(Caption will be displayed when you start playing the video.)

Alright humans, as humans we communicate data audibly with

our voices, textually by well, text, and creatively through

other avenues, such as painting and so on and so forth.

Computers really don't care about any of that and

they only speak numbers.

And in order to use many of Scikit Learning's machine



### Video

[Download video file](#)

### Transcripts

[Download SubRip \(.srt\) file](#)[Download Text \(.txt\) file](#)

Your features need be represented as quantitative (preferably numeric) attributes of the thing you're sampling. They can be real world values, such as the readings from a sensor, and other discernible, physical properties. Alternatively, your features can also be calculated derivatives, such as the presence of certain edges and curves in an image, or lack thereof.

If your data comes to you in a nicely formed, numeric, tabular format, then that's one less thing for you to worry about. But there is no guarantee that will be the case, and you will often encounter data in textual or other unstructured forms. Luckily, there are a few techniques that when applied, clean up these scenarios.

### Textual Categorical-Features

If you have a categorical feature, the way to represent it in your dataset depends on if it's ordinal or nominal. For ordinal features, map the order as increasing integers in a single numeric feature. Any entries not found in your designated categories list will be mapped to -1:

```
>>> ordered_satisfaction = ['Very Unhappy', 'Unhappy', 'Neutral', 'Happy', 'Very Happy']
>>> df = pd.DataFrame({'satisfaction':['Mad', 'Happy', 'Unhappy', 'Neutral']})
>>> df.satisfaction = df.satisfaction.astype("category",
    ordered=True,
    categories=ordered_satisfaction
).cat.codes

>>> df
   satisfaction
0             -1
1              3
2              1
3              2
```

On the other hand, if your feature is nominal (and thus there is no obvious numeric ordering), then you have two options. The first is you can encoded it similar as you did above. This would be a fast-and-dirty approach. While you're just getting accustomed to your dataset and taking it for its first run through your data analysis pipeline, this method might be best:

```
>>> df = pd.DataFrame({'vertebrates':[
...   'Bird',
...   'Bird',
...   'Mammal',
...   'Fish',
...   'Amphibian',
...   'Reptile',
...   'Mammal',
...   ]})

# Method 1)
>>> df['vertebrates'] = df.vertebrates.astype("category").cat.codes

>>> df
  vertebrates  vertebrates
0         Bird           1
1         Bird           1
2        Mammal           3
3         Fish           2
4  Amphibian           0
5        Reptile           4
6        Mammal           3
```

Notice how this time, `ordered=True` was not passed in, nor was a specific ordering listed. Because of this, Pandas encodes your nominal entries in alphabetical order. This approach is fine for getting your feet wet, but the issue it has is that it still introduces an ordering to a categorical list of items that inherently has none. This may or may not cause problems for you in the future. If you aren't getting the results you hoped for, or even if you *are* getting the results you desired but would like to further increase the result accuracy, then a more precise encoding approach would be to separate the distinct values out into individual boolean features:

```
# Method 2)
>>> df = pd.get_dummies(df, columns=['vertebrates'])

>>> df
  vertebrates_Amphibian  vertebrates_Bird  vertebrates_Fish \
0                    0.0                1.0                0.0
1                    0.0                1.0                0.0
2                    0.0                0.0                0.0
3                    0.0                0.0                1.0
4                    1.0                0.0                0.0
5                    0.0                0.0                0.0
6                    0.0                0.0                0.0

  vertebrates_Mammal  vertebrates_Reptile
0                    0.0                0.0
1                    0.0                0.0
2                    1.0                0.0
3                    0.0                0.0
4                    0.0                0.0
5                    0.0                1.0
6                    1.0                0.0
```

These newly created features are called boolean features because the only values they can contain are either 0 for non-inclusion, or 1 for inclusion. Pandas `.get_dummies()` method allows you to completely replace a single, nominal feature with multiple boolean indicator features. This method is quite powerful and has many configurable options, including the ability to return a `SparseDataFrame`, and other prefixing options. It's benefit is that no erroneous ordering is introduced into your dataset.

© All Rights Reserved



English ▼

© 2012-2017 edX Inc. All rights reserved except where noted. EdX, Open edX and the edX and Open EdX logos are registered trademarks or trademarks of edX Inc. | 粤ICP备17044299号-2

POWERED BY  
OPENedX®



