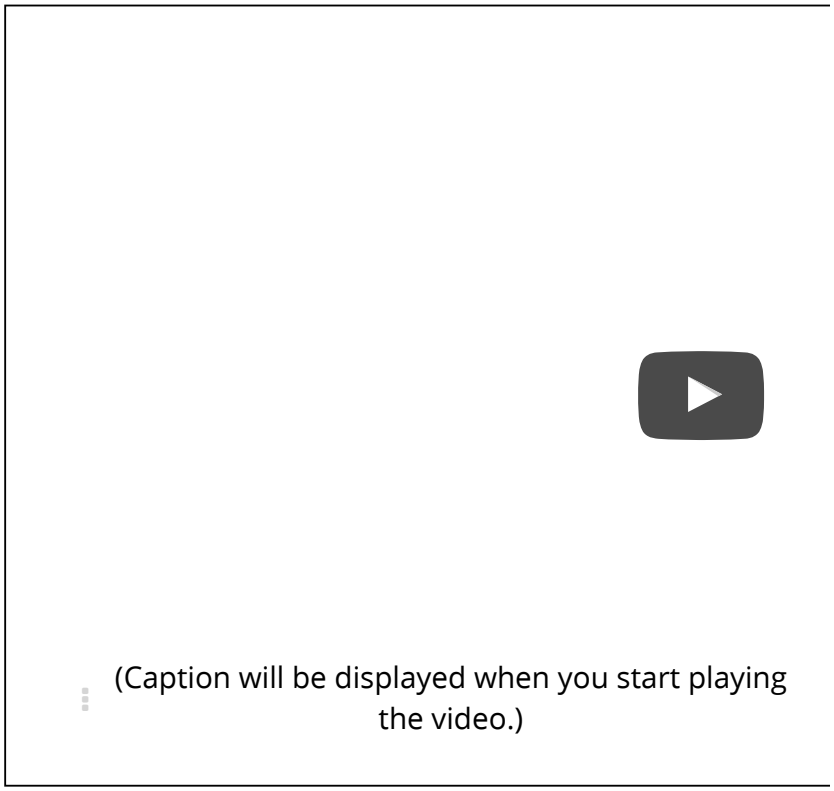
Video

Bookmark this page

SciKit-Learn and PCA



Start of transcript. Skip to the end.

Welcome to scikit-learn's website.

The way that this page is organized is in about

four to five different major categories.

So the first category that we're going to look at and

the first area of the website we're going to take a peek at,

is actually the user guide, the user manual section.

Video

Download video file

Transcripts

Download SubRip (.srt) file

Download Text (.txt) file

To get started, import PCA from `sklearn.decomposition` and then create a new instance of the model setting the `n_components` parameter to the number of dimensions you wish to keep. This value has to be less than or equal to the number of features in your original

dataset, since each computed component is a linear combination of your original features. The second parameter, svd_solver, dictates if a full singular value decomposition should be preformed on your data, or a randomized truncated one. If you decide to use randomized, be sure to seed the random_state variable whenever if you intend on producing replaceable results.

```
>>> from sklearn.decomposition import PCA
>>> pca = PCA(n_components=2, svd_solver='full')
>>> pca.fit(df)
PCA(copy=True, n_components=2, whiten=False)

>>> T = pca.transform(df)

>>> df.shape
(430, 6) # 430 Student survey responses, 6 questions..

>>> T.shape
(430, 2) # 430 Student survey responses, 2 principal components..
```

Once you've fit the model against your dataframe, you can use it to transform your dataset's observations (or any other observation that share its feature space) into the newly computed, principal component feature space with the `.transform()` method. This transformation is bidirectional, so you can recover your original feature values using `.inverse_transform()` so long as you don't drop any components. If even one component was removed, then after performing the inverse transformation back to the regular feature space, there will be some signs of information loss proportional to which component was dropped.

There are a few other interesting model attribute that SciKit-Learn exposes to you after you've trained your PCA model with the `.fit()` method:

- **components_** These are your principal component vectors and are linear combinations of your original features. As such, they exist within the feature space of your original dataset.
- **explained_variance_** This is the calculated amount of variance which exists in the newly computed principal components.
- **explained_variance_ratio_** Normalized version of explained_variance_ for when your interest is with probabilities.

English