feature-grouper

Release 0.1.0

Dec 07, 2020

Contents:

CHAPTER 1

Overview

The `feature-grouper` package provides functions and a scikit-learn transformer class for applying a simple yet effective form of dimensionality reduction based on hierarchical clustering of correlated features.

Example usage:

```
import numpy as np
import pandas as pd
from sklearn import datasets
import feature_grouper

iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
print(iris_df.head())
"""
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
"""

threshold = 0.5 # correlation coefficient threshold for clustering
fg = feature_grouper.FeatureGrouper(threshold)
iris_trans = fg.fit_transform(iris_df)

loadings = pd.DataFrame(fg.components_, columns=iris.feature_names)
print(loadings)
"""
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0           0.471405               0.0           0.471405          0.471405
1           0.000000               1.0           0.000000          0.000000
"""

# column 0 is now a linear combination of correlated features
```

(continues on next page)

(continued from previous page)

```
# sepal length, petal length, and petal width.
print(iris_trans.head())
"""
          0    1
0  3.158410  3.5
1  3.064129  3.0
2  2.922708  3.2
3  2.969848  3.1
4  3.111270  3.6
"""
```

CHAPTER 2

`feature_grouper` API reference

A set of functions and an sklearn transformer class for finding clusters of correlated features and grouping them together into feature groups.

class FeatureGrouper(*threshold=0.5*, *copy=True*)

Hierarchical clustering-based dimensionality reduction.

Calculates correlation matrix of all features in X, applies hierarchical clustering to create flat clusters of highly correlated features, then generates and applies a loading matrix that evenly weights the input features within each cluster.

Input features should be normalized (i.e. z-scores).

Parameters

- **threshold** – float The minimum correlation similarity threshold to group descendants of a cluster node into the same flat cluster.
- **copy** – bool If False, data passed to transform are overwritten.

Variables

- **components_** – array, shape (n_components, n_features) The loading matrix obtained from clustering and weighting correlated features.
- **n_components_** – int The number of components that were estimated from the data.

fit(*X*, *y=None*)

Fit the model with X.

Parameters **X** – array-like, shape (n_samples, n_features) New data, where n_samples is the number of samples and n_features is the number of features.

inverse_transform(*X*)

Transform data back to its original space. In other words, return an input X_original whose transform would be X.

Parameters **X** – array-like, shape (n_samples, n_components) New data, where n_samples is the number of samples and n_components is the number of components.

transform(*X*)
Apply dimensionality reduction on X.

Parameters X – array-like, shape (n_samples, n_features) New data, where n_samples is the number of samples and n_features is the number of features.

cluster(*X*, *threshold=0.5*)
Find clusters of correlated features from a correlation matrix using hierarchical clustering.

Parameters

- **X** – array-like, shape (n_samples, n_features) New data, where n_samples is the number of samples and n_features is the number of features.
- **threshold** – float The minimum correlation similarity threshold to group descendants of a cluster node into the same flat cluster.

make_loadings(*labels*, *threshold=0.5*)
Generate a loading matrix from the feature cluster labels, given a minimum correlation similarity threshold.

Apply the loading matrix to the original data with `np.matmul` or the `@` operator.

Example

```
>>> import numpy as np
>>> import feature_grouper
>>> threshold = 0.5
>>> clusters = feature_grouper.cluster(X, threshold)
>>> loading_matrix = feature_grouper.make_loading_matrix(clusters, threshold)
>>> X_transformed = X @ loading_matrix
```

Parameters

- **labels** – array-like, shape (n,) A numpy 1d array containing the cluster number label for each column in the original dataset.
- **threshold** – float The minimum correlation similarity threshold that was used to cluster the features.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

Index