postlearn Documentation

Release 0+untagged.30.g5f036bf.dirty

Tom Augspurger

November 17, 2016

Contents

A collection of useful post-estimation diagnostics for scikit-learn models.

Contents:

CHAPTER 1

Postlearn

Common post-estimation tasks for scikit-learn.

CHAPTER 2

Usage

There are two interfaces to the library. Anything that actually does something useful, like plotting the grid scores or a learning curve, is available as a top-level function.

There are convenience classes built on top of these functions, These classes take a fit estimator, and the training data (optionally test data). The classes provide caching of predicted values and a few other conveniences. Post-estimation reporting methods.

class postlearn.reporter.**ClassificationResults**(*model*, *X_train*, *y_train*, *X_test=None*, *y_test=None*, *labels=None*)

A convinience class, wrapping all the reporting methods and caching intermediate calculations.

plot_roc_curve(*, *ax=None*, *y_true=None*, *y_score=None*)

Plot the ROC.

proba_test

Predicted probabilities for the test set

proba_train

Predicted probabilities for the training set

y_pred_test

Predicted values for the test set

y_pred_train

Predicted values for the training set

y_score_test

Predicted positive score (column 1) for the test set

y_score_train

Predicted positive score (column 1) for the training set

class postlearn.reporter.**GridSearchMixin**

Helper methods appropriate for estimators fit with a GridSearch.

postlearn.reporter.**confusion_matrix**(*y_true=None*, *y_pred=None*, *labels=None*)

Dataframe of confusion matrix. Rows are actual, and columns are predicted.

Parameters **y_true** : array

y_pred : array

labels : list-like

Returns **confusion_matrix** : DataFrame

postlearn.reporter.**default_args**(***attrs*)

Pull the defaults for a method from *self*.

Parameters **attrs** : dict

mapping parameter name to attribute name Attributes with the same name need not be included.

Returns deco: new function, injecting the *attrs* into *kwargs*

Notes

Only usable with keyword-only arguments.

Examples

@default_args({'y': 'y_train'}) def printer(self, *, y=None, y_pred=None):

print('y: ', y) print('y_pred: ', y_pred)

postlearn.reporter.**extract_grid_scores**(*model*)

Extract grid scores from a model or pipeline.

Parameters **model** : Estimator or Pipeline

must end in sklearn.grid_search.GridSearchCV

Returns **scores** : list

See also:

unpack_grid_scores

postlearn.reporter.**plot_feature_importance**(*model*, *labels*, *n=10*, *orient='h'*)

Bar plot of feature importance.

Parameters **model** : Pipeline or Estimator

labels : list-like

n : int

number of features to include

orient : {'h', 'v'}

horizontal or vertical barplot

Returns **ax** : matplotlib.axes

Notes

Works with Regression, **coefs_**, or ensembes with **feature_importances_**

postlearn.reporter.**plot_grid_scores**(*model*, *x*, *y=None*, *hue=None*, *row=None*, *col=None*, *col_wrap=None*, ***kwargs*)

Wrapper around seaborn.factorplot.

Parameters **model** : Pipeline or Estimator

x, hue, row, col : str

parameters grid searched over

y : str

the target of interest, default *'mean_'*

Returns **g** : seaborn.FacetGrid

postlearn.reporter.**plot_learning_curve**(*estimator*, *X*, *y*, *train_sizes=array([0.1*, *0.325*, *0.55*, *0.775*, *1.])*, *cv=None*, *n_jobs=1*, *ax=None*)

Plot the learning curve for *estimator*.

Parameters **estimator** : sklearn.Estimator

X : array-like

y : array-like

train_sizes : array-like

list of floats between 0 and 1

cv : int

n_jobs : int

ax : matplotlib.axes

postlearn.reporter.**plot_regularization_path**(*model*)

Plot the regularization path of coefficients from e.g. a Lasso

postlearn.reporter.**plot_roc_curve**(*y_true*, *y_score*, *ax=None*)

Plot the Receiving Operator Characteristic curved, including the Area under the Curve (AUC) score.

Parameters **y_true** : array

y_score : array

ax : matplotlib.axes, defaults to new axes

Returns **ax** : matplotlib.axes

postlearn.reporter.**unpack_grid_scores**(*model=None*)

Unpack mean grid scores into a DataFrame

Parameters **model** : Estimator or Pipeline

must end in sklearn.grid_search.GridSearchCV

Returns **scores** : DataFrame

See also:

plot_grid_scores

Examples

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> from sklearn import datasets
>>> from sklearn.grid_search import GridSearchCV
>>> from sklearn.preprocessing import StandardScaler
>>> X, y =datasets.make_classification()
>>> model = GridSearchCV(RandomForestClassifier(),
...                      param_grid={
...                          'n_estimators': [10, 20, 30],
...                          'max_features': [.1, .5, 1]
```

```
...                         })
>>> model.fit(X, y)
>>> unpack_grid_scores(model)
   mean_      std_  max_features  n_estimators
0   0.88  0.062416           0.1            10
1   0.88  0.046536           0.1            20
2   0.85  0.095309           0.1            30
3   0.88  0.062686           0.5            10
4   0.91  0.072044           0.5            20
5   0.90  0.073366           0.5            30
6   0.78  0.032929           1.0            10
7   0.86  0.048224           1.0            20
8   0.85  0.072174           1.0            30
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

p

C

D

E

G

P

U

Y