
expandas Documentation

Release 0.0.1

sinhrks

March 06, 2015

1	Data Handling	3
1.1	Data Preparation	3
1.2	Data Manipulation	4
2	Use scikit-learn	7
2.1	Basics	7
2.2	Pipeline	10
2.3	Cross Validation	11
2.4	Grid Search	12
3	expandas.core package	13
3.1	Submodules	13
3.2	expandas.core.accessor module	13
3.3	expandas.core.frame module	13
3.4	expandas.core.series module	16
3.5	Module contents	16
4	expandas.skaccessors package	17
4.1	Subpackages	18
4.2	Submodules	18
4.3	expandas.skaccessors.cluster module	18
4.4	expandas.skaccessors.covariance module	19
4.5	expandas.skaccessors.cross_validation module	19
4.6	expandas.skaccessors.decomposition module	19
4.7	expandas.skaccessors.ensemble module	20
4.8	expandas.skaccessors.feature_extraction module	20
4.9	expandas.skaccessors.feature_selection module	20
4.10	expandas.skaccessors.gaussian_process module	21
4.11	expandas.skaccessors.grid_search module	21
4.12	expandas.skaccessors.isotonic module	21
4.13	expandas.skaccessors.linear_model module	21
4.14	expandas.skaccessors.manifold module	22
4.15	expandas.skaccessors.metrics module	22
4.16	expandas.skaccessors.multiclass module	22
4.17	expandas.skaccessors.neighbors module	23
4.18	expandas.skaccessors.pipeline module	23
4.19	expandas.skaccessors.preprocessing module	23
4.20	expandas.skaccessors.svm module	23

4.21	Module contents	24
	Python Module Index	25

Contents:

Data Handling

1.1 Data Preparation

This section describes how to prepare basic data format named `ModelFrame`. `ModelFrame` defines a metadata to specify target (response variable) and data (explanatory variable / features). Using these metadata, `ModelFrame` can call other statistics/ML functions in more simple way.

You can create `ModelFrame` as the same manner as `pandas.DataFrame`. The below example shows how to create basic `ModelFrame`, which DOESN'T have target values.

```
>>> import expandas as expd

>>> df = expd.ModelFrame({'A': [1, 2, 3], 'B': [2, 3, 4],
...                       'C': [3, 4, 5]}, index=['A', 'B', 'C'])
>>> df
   A  B  C
A  1  2  3
B  2  3  4
C  3  4  5

>>> type(df)
<class 'expandas.core.frame.ModelFrame'>
```

You can check whether the created `ModelFrame` has target values using `ModelFrame.has_target()` function.

```
>>> df.has_target()
False
```

Target values can be specifyied via `target` keyword. You can simply pass a column name to be handled as target. Target column name can be confirmed via `target_name` property.

```
>>> df2 = expd.ModelFrame({'A': [1, 2, 3], 'B': [2, 3, 4],
...                       'C': [3, 4, 5]}, target='A')
>>> df2
   A  B  C
0  1  2  3
1  2  3  4
2  3  4  5

>>> df2.has_target()
True

>>> df2.target_name
'A'
```

Also, you can pass any list-likes to be handled as a target. In this case, target column will be named as ".target".

```
>>> df3 = expd.ModelFrame({'A': [1, 2, 3], 'B': [2, 3, 4],
...                       'C': [3, 4, 5]}, target=[4, 5, 6])
>>> df3
   .target  A  B  C
0         4  1  2  3
1         5  2  3  4
2         6  3  4  5

>>> df3.has_target()
True

>>> df3.target_name
'.target'
```

Also, you can pass `pandas.DataFrame` and `pandas.Series` as data and target.

```
>>> import pandas as pd
df4 = expd.ModelFrame({'A': [1, 2, 3], 'B': [2, 3, 4],
...                   'C': [3, 4, 5]}, target=pd.Series([4, 5, 6]))
>>> df4
   .target  A  B  C
0         4  1  2  3
1         5  2  3  4
2         6  3  4  5

>>> df4.has_target()
True

>>> df4.target_name
'.target'
```

Note: Target values are mandatory to perform operations which require response variable, such as regression and supervised learning.

1.2 Data Manipulation

You can access to each property as the same as `pandas.DataFrame`. Sliced results will be `ModelSeries` (simple wrapper for `pandas.Series` to support some data manipulation) or `ModelFrame`

```
>>> df
   A  B  C
A  1  2  3
B  2  3  4
C  3  4  5

>>> sliced = df['A']
>>> sliced
A    1
B    2
C    3
Name: A, dtype: int64

>>> type(sliced)
<class 'expandas.core.series.ModelSeries'>
```



```
>>> subset = df[['A', 'B']]
>>> subset
   A  B
A  1  2
B  2  3
C  3  4

>>> type(subset)
<class 'expandas.core.frame.ModelFrame'>
```

ModelFrame has a special properties `data` to access data (features) and `target` to access target.

```
>>> df2
   A  B  C
0  1  2  3
1  2  3  4
2  3  4  5

>>> df2.target_name
'A'

>>> df2.data
   B  C
0  2  3
1  3  4
2  4  5

>>> df2.target
0    1
1    2
2    3
Name: A, dtype: int64
```

You can update data and target via properties, in addition to standard `pandas.DataFrame` ways.

```
>>> df2.target = [9, 9, 9]
>>> df2
   A  B  C
0  9  2  3
1  9  3  4
2  9  4  5

>>> df2.data = pd.DataFrame({'X': [1, 2, 3], 'Y': [4, 5, 6]})
>>> df2
   A  X  Y
0  9  1  4
1  9  2  5
2  9  3  6

>>> df2['X'] = [0, 0, 0]
>>> df2
   A  X  Y
0  9  0  4
1  9  0  5
2  9  0  6
```

You can change target column specifying `target_name` property. Specifying a column which doesn't exist in ModelFrame results in target column to be data column.

```
>>> df2.target_name
'A'

>>> df2.target_name = 'X'
>>> df2.target_name
'X'

>>> df2.target_name = 'XXXX'
>>> df2.has_target()
False

>>> df2.data
   A  X  Y
0  9  0  4
1  9  0  5
2  9  0  6
```

Use scikit-learn

This section describes how to use `scikit-learn` functionalities via `expandas`.

2.1 Basics

You can create `ModelFrame` instance from `scikit-learn` datasets directly.

```
>>> import expandas as expd
>>> import sklearn.datasets as datasets

>>> df = expd.ModelFrame(datasets.load_iris())
>>> df.head()
   .target  sepal length (cm)  sepal width (cm)  petal length (cm)  \
0         0                5.1                3.5                1.4
1         0                4.9                3.0                1.4
2         0                4.7                3.2                1.3
3         0                4.6                3.1                1.5
4         0                5.0                3.6                1.4

      petal width (cm)
0                   0.2
1                   0.2
2                   0.2
3                   0.2
4                   0.2

# make columns be readable
>>> df.columns = ['.target', 'sepal length', 'sepal width', 'petal length', 'petal width']

ModelFrame has accessor methods which makes easier access to scikit-learn namespace.

>>> df.cluster.KMeans
<class 'sklearn.cluster.k_means_.KMeans'>
```

Following table shows `scikit-learn` module and corresponding `ModelFrame` module.

scikit-learn	ModelFrame accessor
sklearn.cluster	ModelFrame.cluster
sklearn.covariance	ModelFrame.covariance
sklearn.cross_validation	ModelFrame.cross_validation
sklearn.decomposition	ModelFrame.decomposition
sklearn.datasets	(not accesible from accessor)
sklearn.dummy	ModelFrame.dummy
sklearn.ensemble	ModelFrame.ensemble
sklearn.feature_extraction	ModelFrame.feature_extraction
sklearn.gaussian_process	ModelFrame.gaussian_process (WIP)
sklearn.grid_search	ModelFrame.grid_search
sklearn.isotonic	ModelFrame.isotonic (WIP)
sklearn.kernel_approximation	ModelFrame.kernel_approximation
sklearn.lda	ModelFrame.lda
sklearn.linear_model	ModelFrame.linear_model
sklearn.manifold	ModelFrame.manifold
sklearn.metrics	ModelFrame.metrics
sklearn.mixture	ModelFrame.mixture
sklearn.multiclass	ModelFrame.multiclass
sklearn.naive_bayes	ModelFrame.naive_bayes
sklearn.neighbors	ModelFrame.neighbors
sklearn.cross_decomposition	ModelFrame.cross_decomposition (WIP)
sklearn.pipeline	ModelFrame.pipeline
sklearn.preprocessing	ModelFrame.preprocessing
sklearn.qda	ModelFrame.qda
sklearn.semi_supervised	ModelFrame.semi_supervised
sklearn.svm	ModelFrame.svm
sklearn.tree	ModelFrame.tree
sklearn.utils	(not accesible from accessor)

Thus, you can instantiate each estimator via ModelFrame accessors. Once create an estimator, you can pass it to ModelFrame.fit then predict. ModelFrame automatically uses its data and target properties for each operations.

```
>>> estimator = df.cluster.KMeans(n_clusters=3)
>>> df.fit(estimator)
```

```
>>> predicted = df.predict(estimator)
>>> predicted
0      1
1      1
2      1
...
147    2
148    2
149    0
Length: 150, dtype: int32
```

ModelFrame has following methods corresponding to various scikit-learn estimators.

- ModelFrame.fit
- ModelFrame.transform
- ModelFrame.fit_transform
- ModelFrame.inverse_transform
- ModelFrame.predict

- `ModelFrame.fit_predict`
- `ModelFrame.score`
- `ModelFrame.predict_proba`
- `ModelFrame.predict_log_proba`
- `ModelFrame.decision_function`

Following example shows to perform PCA, then revert principal components back to original space.

```
>>> estimator = df.decomposition.PCA()
>>> df.fit(estimator)

>>> transformed = df.transform(estimator)
>>> transformed.head()
   .target  0      1      2      3
0      0 -2.684207 -0.326607  0.021512  0.001006
1      0 -2.715391  0.169557  0.203521  0.099602
2      0 -2.889820  0.137346 -0.024709  0.019305
3      0 -2.746437  0.311124 -0.037672 -0.075955
4      0 -2.728593 -0.333925 -0.096230 -0.063129

>>> type(transformed)
<class 'expandas.core.frame.ModelFrame'>

>>> transformed.inverse_transform(estimator)
   .target  0      1      2      3
0      0  5.1  3.5  1.4  0.2
1      0  4.9  3.0  1.4  0.2
2      0  4.7  3.2  1.3  0.2
3      0  4.6  3.1  1.5  0.2
4      0  5.0  3.6  1.4  0.2
..      ...
145     2  6.7  3.0  5.2  2.3
146     2  6.3  2.5  5.0  1.9
147     2  6.5  3.0  5.2  2.0
148     2  6.2  3.4  5.4  2.3
149     2  5.9  3.0  5.1  1.8

[150 rows x 5 columns]
```

Note: columns information will be lost once transformed to principal components.

`ModelFrame` preserves last predicted result in `predicted` attribute. If `ModelFrame` both has `target` and `predicted` values, the model evaluation can be performed using functions available in `ModelFrame.metrics`.

```
>>> estimator = df.svm.SVC()
>>> df.fit(estimator)

>>> df.predict(estimator)
0      0
1      0
2      0
...
147     2
148     2
149     2
Length: 150, dtype: int64
```

```
>>> df.predicted
0      0
1      0
2      0
...
147     2
148     2
149     2
Length: 150, dtype: int64

>>> df.metrics.confusion_matrix()
Predicted    0    1    2
Target
0           50    0    0
1            0   48    2
2            0    0   50
```

2.2 Pipeline

ModelFrame can handle pipeline as the same as normal estimators.

```
>>> estimators = [('reduce_dim', df.decomposition.PCA()),
...               ('svm', df.svm.SVC())]
>>> pipe = df.pipeline.Pipeline(estimators)
>>> df.fit(pipe)

>>> df.predict(pipe)
0      0
1      0
2      0
...
147     2
148     2
149     2
Length: 150, dtype: int64
```

Above expression is the same as below:

```
>>> df2 = df.copy()
>>> df2 = df2.fit_transform(df2.decomposition.PCA())
>>> svm = df2.svm.SVC()
>>> df2.fit(svm)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0, degree=3, gamma=0.0,
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
>>> df2.predict(svm)
0      0
1      0
2      0
...
147     2
148     2
149     2
Length: 150, dtype: int64
```

2.3 Cross Validation

scikit-learn has some classes for cross validation. `cross_validation.train_test_split` splits data to training and test set. You can access to the function via `cross_validation` accessor.

```
>>> train_df, test_df = df.cross_validation.train_test_split()
>>> train_df
```

	.target	sepal length	sepal width	petal length	petal width
0	0	4.8	3.4	1.9	0.2
1	1	6.3	3.3	4.7	1.6
2	0	4.8	3.4	1.6	0.2
3	2	7.7	2.6	6.9	2.3
4	0	5.4	3.4	1.7	0.2
..
107	0	5.1	3.7	1.5	0.4
108	1	6.7	3.1	4.7	1.5
109	0	4.7	3.2	1.3	0.2
110	0	5.8	4.0	1.2	0.2
111	0	5.1	3.5	1.4	0.2

[112 rows x 5 columns]

```
>>> test_df
```

	.target	sepal length	sepal width	petal length	petal width
0	2	6.3	2.7	4.9	1.8
1	0	4.5	2.3	1.3	0.3
2	2	5.8	2.8	5.1	2.4
3	0	4.3	3.0	1.1	0.1
4	0	5.0	3.0	1.6	0.2
..
33	1	6.7	3.1	4.4	1.4
34	0	4.6	3.6	1.0	0.2
35	1	5.7	3.0	4.2	1.2
36	1	5.9	3.0	4.2	1.5
37	2	6.4	2.8	5.6	2.1

[38 rows x 5 columns]

Also, there are some iterative classes which returns indexes for training sets and test sets. You can slice `ModelFrame` using these indexes.

```
>>> kf = df.cross_validation.KFold(n=150, n_folds=3)
>>> for train_index, test_index in kf:
...     print('training set shape: ', df.iloc[train_index, :].shape,
...           'test set shape: ', df.iloc[test_index, :].shape)
('training set shape: ', (100, 5), 'test set shape: ', (50, 5))
('training set shape: ', (100, 5), 'test set shape: ', (50, 5))
('training set shape: ', (100, 5), 'test set shape: ', (50, 5))
```

For further simplification, `ModelFrame.cross_validation.iterate` can accept such iterators and returns `ModelFrame` corresponding to training and test data.

```
>>> kf = df.cross_validation.KFold(n=150, n_folds=3)
>>> for train_df, test_df in df.cross_validation.iterate(kf):
...     print('training set shape: ', train_df.shape,
...           'test set shape: ', test_df.shape)
('training set shape: ', (100, 5), 'test set shape: ', (50, 5))
('training set shape: ', (100, 5), 'test set shape: ', (50, 5))
```

```
('training set shape: ', (100, 5), 'test set shape: ', (50, 5))
```

2.4 Grid Search

You can perform grid search using `ModelFrame.fit`.

```
>>> tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
...                       'C': [1, 10, 100]},
...                      {'kernel': ['linear'], 'C': [1, 10, 100]}]

>>> df = expd.ModelFrame(datasets.load_digits())
>>> cv = df.grid_search.GridSearchCV(df.svm.SVC(C=1), tuned_parameters,
...                                  cv=5, scoring='precision')

>>> df.fit(cv)

>>> cv.best_estimator_
SVC(C=10, cache_size=200, class_weight=None, coef0=0.0, degree=3, gamma=0.001,
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

In addition, `ModelFrame.grid_search` has a `describe` function to organize each grid search result as `pd.DataFrame` accepting estimator.

```
>>> df.grid_search.describe(cv)
   mean      std    C  gamma kernel
0  0.974108  0.013139    1  0.0010   rbf
1  0.951416  0.020010    1  0.0001   rbf
2  0.975372  0.011280   10  0.0010   rbf
3  0.962534  0.020218   10  0.0001   rbf
4  0.975372  0.011280  100  0.0010   rbf
5  0.964695  0.016686  100  0.0001   rbf
6  0.951811  0.018410    1     NaN  linear
7  0.951811  0.018410   10     NaN  linear
8  0.951811  0.018410  100     NaN  linear
```

API:

expandas.core package

3.1 Submodules

3.2 expandas.core.accessor module

class `expandas.core.accessor.AccessorMethods` (*df, module_name=None, attrs=None*)

Bases: `object`

Accessor to related functionalities.

3.3 expandas.core.frame module

class `expandas.core.frame.ModelFrame` (*data, target=None, *args, **kwargs*)

Bases: `pandas.core.frame.DataFrame`

Data structure subclassing `pandas.DataFrame` to define a metadata to specify target (response variable) and data (explanatory variable / features).

data : same as `pandas.DataFrame` *target* : str or array-like

Column name or values to be used as target

args : arguments passed to `pandas.DataFrame` *kwargs* : keyword arguments passed to `pandas.DataFrame`

cluster = None

covariance = None

cross_decomposition = None

cross_validation = None

data

decision

Return current estimator's decision function

probabilities : `ModelFrame`

decision_function (*estimator, *args, **kwargs*)

Call estimator's `decision_function` method.

args : arguments passed to decision_function method
 kwargs : keyword arguments passed to decision_function method

returned : decisions

decomposition = None

dummy = None

ensemble = None

estimator
 Return most recently used estimator

estimator : estimator

feature_extraction = None

feature_selection = None

fit (*estimator*, *args, **kwargs)
 Call estimator's fit method.

args : arguments passed to fit method
 kwargs : keyword arguments passed to fit method

returned : None or fitted estimator

fit_predict (*estimator*, *args, **kwargs)
 Call estimator's fit_predict method.

args : arguments passed to fit_predict method
 kwargs : keyword arguments passed to fit_predict method

returned : predicted result

fit_transform (*estimator*, *args, **kwargs)
 Call estimator's fit_transform method.

args : arguments passed to fit_transform method
 kwargs : keyword arguments passed to fit_transform method

returned : transformed result

gaussian_process = None

grid_search = None

has_data ()
 Return whether `ModelFrame` has data

has_data : bool

has_target ()
 Return whether `ModelFrame` has target

has_target : bool

inverse_transform (*estimator*, *args, **kwargs)
 Call estimator's inverse_transform method.

args : arguments passed to inverse_transform method
 kwargs : keyword arguments passed to inverse_transform method

returned : transformed result

isotonic = None

kernel_approximation = None

lda = None

linear_model = None

log_proba

Return current estimator's log probabilities

probabilities : `ModelFrame`

manifold = None

metrics = None

mixture = None

multiclass = None

naive_bayes = None

neighbors = None

pipeline = None

predict (*estimator*, **args*, ***kwargs*)

Call estimator's predict method.

args : arguments passed to predict method *kwargs* : keyword arguments passed to predict method

returned : predicted result

predict_log_proba (*estimator*, **args*, ***kwargs*)

Call estimator's predict_log_proba method.

args : arguments passed to predict_log_proba method *kwargs* : keyword arguments passed to predict_log_proba method

returned : probabilities

predict_proba (*estimator*, **args*, ***kwargs*)

Call estimator's predict_proba method.

args : arguments passed to predict_proba method *kwargs* : keyword arguments passed to predict_proba method

returned : probabilities

predicted

Return current estimator's predicted results

predicted : `ModelSeries`

preprocessing = None

proba

Return current estimator's probabilities

probabilities : `ModelFrame`

qda = None

score (*estimator*, **args*, ***kwargs*)

Call estimator's score method.

args : arguments passed to score method *kwargs* : keyword arguments passed to score method

returned : score

semi_supervised = None

```

svm = None
target
target_name
transform (estimator, *args, **kwargs)
    Call estimator's transform method.
    args : arguments passed to transform method
    kwargs : keyword arguments passed to transform method
    returned : transformed result
tree = None

```

3.4 expandas.core.series module

```

class expandas.core.series.ModelSeries (data=None, index=None, dtype=None, name=None,
                                         copy=False, fastpath=False)
    Bases: pandas.core.series.Series
    Wrapper for pandas.Series to support sklearn.preprocessing
    preprocessing = None
    to_frame (name=None)
        Convert Series to DataFrame
    name [object, default None] The passed name should substitute for the series name (if it has one).
    data_frame : DataFrame

```

3.5 Module contents

expandas.skaccessors package

4.1 Subpackages

4.1.1 expandas.skaccessors.test package

Submodules

expandas.skaccessors.test.test_cluster module

expandas.skaccessors.test.test_covariance module

expandas.skaccessors.test.test_cross_decomposition module

expandas.skaccessors.test.test_cross_validation module

expandas.skaccessors.test.test_decomposition module

expandas.skaccessors.test.test_dummy module

expandas.skaccessors.test.test_ensemble module

expandas.skaccessors.test.test_feature_extraction module

expandas.skaccessors.test.test_feature_selection module

expandas.skaccessors.test.test_gaussian_process module

expandas.skaccessors.test.test_grid_search module

expandas.skaccessors.test.test_isotonic module

expandas.skaccessors.test.test_kernel_approximation module

expandas.skaccessors.test.test_lda module

expandas.skaccessors.test.test_linear_model module

expandas.skaccessors.test.test_manifold module

expandas.skaccessors.test.test_metrics module

expandas.skaccessors.test.test_mixture module

expandas.skaccessors.test.test_multiclass module

Accessor to `sklearn.cluster`.

```
affinity_propagation (*args, **kwargs)
bicluster = None
dbscan (*args, **kwargs)
k_means (n_clusters, *args, **kwargs)
mean_shift (*args, **kwargs)
spectral_clustering (*args, **kwargs)
```

4.4 expandas.skaccessors.covariance module

```
class expandas.skaccessors.covariance.CovarianceMethods (df, module_name=None,
                                                         attrs=None)
    Bases: expandas.core.accessor.AccessorMethods
    Accessor to sklearn.covariance.
    empirical_covariance (*args, **kwargs)
    ledoit_wolf (*args, **kwargs)
    oas (*args, **kwargs)
```

4.5 expandas.skaccessors.cross_validation module

```
class expandas.skaccessors.cross_validation.CrossValidationMethods (df,
                                                                      module_name=None,
                                                                      attrs=None)
    Bases: expandas.core.accessor.AccessorMethods
    Accessor to sklearn.cross_validation.
    StratifiedShuffleSplit (*args, **kwargs)
    check_cv (cv, *args, **kwargs)
    cross_val_score (estimator, *args, **kwargs)
    iterate (cv)
    permutation_test_score (estimator, *args, **kwargs)
    train_test_split (*args, **kwargs)
```

4.6 expandas.skaccessors.decomposition module

```
class expandas.skaccessors.decomposition.DecompositionMethods (df,
                                                                module_name=None,
                                                                attrs=None)
    Bases: expandas.core.accessor.AccessorMethods
    Accessor to sklearn.decomposition.
    dict_learning (n_components, alpha, *args, **kwargs)
```

```
dict_learning_online(*args, **kwargs)
fastica(*args, **kwargs)
sparse_encode(dictionary, *args, **kwargs)
```

4.7 expandas.skaccessors.ensemble module

```
class expandas.skaccessors.ensemble.EnsembleMethods(df, module_name=None,
                                                    attrs=None)
    Bases: expandas.core.accessor.AccessorMethods
    Accessor to sklearn.ensemble.
    partial_dependence

class expandas.skaccessors.ensemble.PartialDependenceMethods(df,
                                                            module_name=None,
                                                            attrs=None)
    Bases: expandas.core.accessor.AccessorMethods
    partial_dependence(gbrt, target_variables, **kwargs)
    plot_partial_dependence(gbrt, features, **kwargs)
```

4.8 expandas.skaccessors.feature_extraction module

```
class expandas.skaccessors.feature_extraction.FeatureExtractionMethods(df,
                                                                        module_name=None,
                                                                        attrs=None)
    Bases: expandas.core.accessor.AccessorMethods
    Accessor to sklearn.feature_extraction.
    image = None
    text = None
```

4.9 expandas.skaccessors.feature_selection module

```
class expandas.skaccessors.feature_selection.FeatureSelectionMethods(df,
                                                                      module_name=None,
                                                                      attrs=None)
    Bases: expandas.core.accessor.AccessorMethods
    Accessor to sklearn.feature_selection.
```


4.10 expandas.skaccessors.gaussian_process module

```
class expandas.skaccessors.gaussian_process.GaussianProcessMethods(df, module_name=None,
                                                                    attrs=None)

    Bases: expandas.core.accessor.AccessorMethods
    Accessor to sklearn.gaussian_process.

    correlation_models
    regression_models

class expandas.skaccessors.gaussian_process.RegressionModelsMethods(df, module_name=None,
                                                                    attrs=None)

    Bases: expandas.core.accessor.AccessorMethods
```

4.11 expandas.skaccessors.grid_search module

```
class expandas.skaccessors.grid_search.GridSearchMethods(df, module_name=None,
                                                         attrs=None)

    Bases: expandas.core.accessor.AccessorMethods
    Accessor to sklearn.grid_search.

    describe(estimator)
        Return cross validation results as pd.DataFrame.
```

4.12 expandas.skaccessors.isotonic module

```
class expandas.skaccessors.isotonic.IsotonicMethods(df, module_name=None,
                                                    attrs=None)

    Bases: expandas.core.accessor.AccessorMethods
    Accessor to sklearn.isotonic.

    IsotonicRegression

    check_increasing(*args, **kwargs)
    isotonic_regression(*args, **kwargs)
```

4.13 expandas.skaccessors.linear_model module

```
class expandas.skaccessors.linear_model.LinearModelMethods(df, module_name=None,
                                                            attrs=None)

    Bases: expandas.core.accessor.AccessorMethods
    Accessor to sklearn.linear_model.

    lars_path(*args, **kwargs)
    lasso_path(*args, **kwargs)
    lasso_stability_path(*args, **kwargs)
```

```
orthogonal_mp_gram(*args, **kwargs)
```

4.14 expandas.skaccessors.manifold module

```
class expandas.skaccessors.manifold.ManifoldMethods(df, module_name=None,
                                                    attrs=None)
    Bases: expandas.core.accessor.AccessorMethods
    Accessor to sklearn.manifold.

    locally_linear_embedding(n_neighbors, n_components, *args, **kwargs)
    spectral_embedding(*args, **kwargs)
```

4.15 expandas.skaccessors.metrics module

```
class expandas.skaccessors.metrics.MetricsMethods(df, module_name=None, attrs=None)
    Bases: expandas.core.accessor.AccessorMethods
    Accessor to sklearn.metrics.

    auc(kind='roc', reorder=False, **kwargs)
    average_precision_score(*args, **kwargs)
    confusion_matrix(*args, **kwargs)
    consensus_score(*args, **kwargs)
    f1_score(*args, **kwargs)
    fbeta_score(beta, *args, **kwargs)
    hinge_loss(*args, **kwargs)
    log_loss(*args, **kwargs)
    pairwise
    precision_recall_curve(*args, **kwargs)
    precision_recall_fscore_support(*args, **kwargs)
    precision_score(*args, **kwargs)
    recall_score(*args, **kwargs)
    roc_auc_score(*args, **kwargs)
    roc_curve(*args, **kwargs)
    silhouette_samples(*args, **kwargs)
    silhouette_score(*args, **kwargs)
```

4.16 expandas.skaccessors.multiclass module

```
class expandas.skaccessors.multiclass.MultiClassMethods(df, module_name=None,
                                                         attrs=None)
    Bases: expandas.core.accessor.AccessorMethods
```

Accessor to `sklearn.multiclass`.

OneVsOneClassifier

OneVsRestClassifier

OutputCodeClassifier

fit_ecoc (*args, **kwargs)

fit_ovo (*args, **kwargs)

fit_ovr (*args, **kwargs)

predict_ecoc (*args, **kwargs)

predict_ovo (*args, **kwargs)

predict_ovr (*args, **kwargs)

4.17 expandas.skaccessors.neighbors module

```
class expandas.skaccessors.neighbors.NeighborsMethods(df, module_name=None,
                                                    attrs=None)
    Bases: expandas.core.accessor.AccessorMethods
    Accessor to sklearn.neighbors.
```

4.18 expandas.skaccessors.pipeline module

```
class expandas.skaccessors.pipeline.PipelineMethods(df, module_name=None,
                                                    attrs=None)
    Bases: expandas.core.accessor.AccessorMethods
    Accessor to sklearn.pipeline.
    make_pipeline
    make_union
```

4.19 expandas.skaccessors.preprocessing module

```
class expandas.skaccessors.preprocessing.PreprocessingMethods(df,
                                                            module_name=None,
                                                            attrs=None)
    Bases: expandas.core.accessor.AccessorMethods
    Accessor to sklearn.preprocessing.
    add_dummy_feature (value=1.0)
```

4.20 expandas.skaccessors.svm module

```
class expandas.skaccessors.svm.SVMMethods(df, module_name=None, attrs=None)
    Bases: expandas.core.accessor.AccessorMethods
```

Accessor to `sklearn.svm`.

`ll_min_c(*args, **kwargs)`

`liblinear`

`libsvm`

`libsvm_sparse`

4.21 Module contents

e

- [expandas.core](#), 16
- [expandas.core.accessor](#), 13
- [expandas.core.frame](#), 13
- [expandas.core.series](#), 16
- [expandas.skaccessors](#), 24
 - [expandas.skaccessors.cluster](#), 18
 - [expandas.skaccessors.covariance](#), 19
 - [expandas.skaccessors.cross_validation](#), 19
 - [expandas.skaccessors.decomposition](#), 19
 - [expandas.skaccessors.ensemble](#), 20
 - [expandas.skaccessors.feature_extraction](#), 20
 - [expandas.skaccessors.feature_selection](#), 20
 - [expandas.skaccessors.gaussian_process](#), 21
 - [expandas.skaccessors.grid_search](#), 21
 - [expandas.skaccessors.isotonic](#), 21
 - [expandas.skaccessors.linear_model](#), 21
 - [expandas.skaccessors.manifold](#), 22
 - [expandas.skaccessors.metrics](#), 22
 - [expandas.skaccessors.multiclass](#), 22
 - [expandas.skaccessors.neighbors](#), 23
 - [expandas.skaccessors.pipeline](#), 23
 - [expandas.skaccessors.preprocessing](#), 23
 - [expandas.skaccessors.svm](#), 23
 - [expandas.skaccessors.test](#), 18

A

AccessorMethods (class in `expandas.core.accessor`), 13

`add_dummy_feature()` (ex-
pandas.skaccessors.preprocessing.PreprocessingMethods
method), 23

`affinity_propagation()` (ex-
pandas.skaccessors.cluster.ClusterMethods
method), 19

`auc()` (`expandas.skaccessors.metrics.MetricsMethods`
method), 22

`average_precision_score()` (ex-
pandas.skaccessors.metrics.MetricsMethods
method), 22

B

`bicluster` (`expandas.skaccessors.cluster.ClusterMethods`
attribute), 19

C

`check_cv()` (`expandas.skaccessors.cross_validation.CrossValidationMethods`
method), 19

`check_increasing()` (ex-
pandas.skaccessors.isotonic.IsotonicMethods
method), 21

`cluster` (`expandas.core.frame.ModelFrame` attribute), 13

`ClusterMethods` (class in `expandas.skaccessors.cluster`),
18

`confusion_matrix()` (ex-
pandas.skaccessors.metrics.MetricsMethods
method), 22

`consensus_score()` (`expandas.skaccessors.metrics.MetricsMethods`
method), 22

`correlation_models` (ex-
pandas.skaccessors.gaussian_process.GaussianProcessMethods
attribute), 21

`covariance` (`expandas.core.frame.ModelFrame` attribute),
13

`CovarianceMethods` (class in ex-
pandas.skaccessors.covariance), 19

`cross_decomposition` (`expandas.core.frame.ModelFrame`
attribute), 13

`cross_val_score()` (`expandas.skaccessors.cross_validation.CrossValidationMethods`
method), 19

`cross_validation` (`expandas.core.frame.ModelFrame` at-
tribute), 13

`CrossValidationMethods` (class in ex-
pandas.skaccessors.cross_validation), 19

D

`data` (`expandas.core.frame.ModelFrame` attribute), 13

`dbscan()` (`expandas.skaccessors.cluster.ClusterMethods`
method), 19

`decision` (`expandas.core.frame.ModelFrame` attribute), 13

`decision_function()` (`expandas.core.frame.ModelFrame`
method), 13

`decomposition` (`expandas.core.frame.ModelFrame`
attribute), 14

`DecompositionMethods` (class in ex-
pandas.skaccessors.decomposition), 19

`describe()` (`expandas.skaccessors.grid_search.GridSearchMethods`
method), 21

`dict_learning()` (`expandas.skaccessors.decomposition.DecompositionMethods`
method), 19

`dict_learning_online()` (ex-
pandas.skaccessors.decomposition.DecompositionMethods
method), 19

`dummy` (`expandas.core.frame.ModelFrame` attribute), 14

E

`empirical_covariance()` (ex-
pandas.skaccessors.covariance.CovarianceMethods
method), 19

`ensemble` (`expandas.core.frame.ModelFrame` attribute),
14

`EnsembleMethods` (class in ex-
pandas.skaccessors.ensemble), 20

`estimator` (`expandas.core.frame.ModelFrame` attribute),
14

`expandas.core` (module), 16

[expandas.core.accessor \(module\)](#), 13
[expandas.core.frame \(module\)](#), 13
[expandas.core.series \(module\)](#), 16
[expandas.skaccessors \(module\)](#), 24
[expandas.skaccessors.cluster \(module\)](#), 18
[expandas.skaccessors.covariance \(module\)](#), 19
[expandas.skaccessors.cross_validation \(module\)](#), 19
[expandas.skaccessors.decomposition \(module\)](#), 19
[expandas.skaccessors.ensemble \(module\)](#), 20
[expandas.skaccessors.feature_extraction \(module\)](#), 20
[expandas.skaccessors.feature_selection \(module\)](#), 20
[expandas.skaccessors.gaussian_process \(module\)](#), 21
[expandas.skaccessors.grid_search \(module\)](#), 21
[expandas.skaccessors.isotonic \(module\)](#), 21
[expandas.skaccessors.linear_model \(module\)](#), 21
[expandas.skaccessors.manifold \(module\)](#), 22
[expandas.skaccessors.metrics \(module\)](#), 22
[expandas.skaccessors.multiclass \(module\)](#), 22
[expandas.skaccessors.neighbors \(module\)](#), 23
[expandas.skaccessors.pipeline \(module\)](#), 23
[expandas.skaccessors.preprocessing \(module\)](#), 23
[expandas.skaccessors.svm \(module\)](#), 23
[expandas.skaccessors.test \(module\)](#), 18

F

[f1_score\(\) \(expandas.skaccessors.metrics.MetricsMethods method\)](#), 22
[fastica\(\) \(expandas.skaccessors.decomposition.DecompositionMethods method\)](#), 20
[fbeta_score\(\) \(expandas.skaccessors.metrics.MetricsMethods method\)](#), 22
[feature_extraction \(expandas.core.frame.ModelFrame attribute\)](#), 14
[feature_selection \(expandas.core.frame.ModelFrame attribute\)](#), 14
[FeatureExtractionMethods \(class in expandas.skaccessors.feature_extraction\)](#), 20
[FeatureSelectionMethods \(class in expandas.skaccessors.feature_selection\)](#), 20
[fit\(\) \(expandas.core.frame.ModelFrame method\)](#), 14
[fit_ecoc\(\) \(expandas.skaccessors.multiclass.MultiClassMethods method\)](#), 23
[fit_ovo\(\) \(expandas.skaccessors.multiclass.MultiClassMethods method\)](#), 23
[fit_ovr\(\) \(expandas.skaccessors.multiclass.MultiClassMethods method\)](#), 23
[fit_predict\(\) \(expandas.core.frame.ModelFrame method\)](#), 14
[fit_transform\(\) \(expandas.core.frame.ModelFrame method\)](#), 14

G

[gaussian_process \(expandas.core.frame.ModelFrame attribute\)](#), 14

[GaussianProcessMethods \(class in expandas.skaccessors.gaussian_process\)](#), 21
[grid_search \(expandas.core.frame.ModelFrame attribute\)](#), 14
[GridSearchMethods \(class in expandas.skaccessors.grid_search\)](#), 21

H

[has_data\(\) \(expandas.core.frame.ModelFrame method\)](#), 14
[has_target\(\) \(expandas.core.frame.ModelFrame method\)](#), 14
[hinge_loss\(\) \(expandas.skaccessors.metrics.MetricsMethods method\)](#), 22

I

[image \(expandas.skaccessors.feature_extraction.FeatureExtractionMethods attribute\)](#), 20
[inverse_transform\(\) \(expandas.core.frame.ModelFrame method\)](#), 14
[isotonic \(expandas.core.frame.ModelFrame attribute\)](#), 14
[isotonic_regression\(\) \(expandas.skaccessors.isotonic.IsotonicMethods method\)](#), 21
[IsotonicMethods \(class in expandas.skaccessors.isotonic\)](#), 21
[IsotonicRegression \(expandas.skaccessors.isotonic.IsotonicMethods attribute\)](#), 21
[iterate\(\) \(expandas.skaccessors.cross_validation.CrossValidationMethods method\)](#), 19

K

[k_means\(\) \(expandas.skaccessors.cluster.ClusterMethods method\)](#), 19
[kernel_approximation \(expandas.core.frame.ModelFrame attribute\)](#), 14

L

[l1_min_c\(\) \(expandas.skaccessors.svm.SVMMethods method\)](#), 24
[lars_path\(\) \(expandas.skaccessors.linear_model.LinearModelMethods method\)](#), 21
[lasso_path\(\) \(expandas.skaccessors.linear_model.LinearModelMethods method\)](#), 21
[lasso_stability_path\(\) \(expandas.skaccessors.linear_model.LinearModelMethods method\)](#), 21
[lda \(expandas.core.frame.ModelFrame attribute\)](#), 14
[ledoit_wolf\(\) \(expandas.skaccessors.covariance.CovarianceMethods method\)](#), 19
[liblinear \(expandas.skaccessors.svm.SVMMethods attribute\)](#), 24

- libsvm (expandas.skaccessors.svm.SVMMethods attribute), 24
- libsvm_sparse (expandas.skaccessors.svm.SVMMethods attribute), 24
- linear_model (expandas.core.frame.ModelFrame attribute), 15
- LinearModelMethods (class in expandas.skaccessors.linear_model), 21
- locally_linear_embedding() (expandas.skaccessors.manifold.ManifoldMethods method), 22
- log_loss() (expandas.skaccessors.metrics.MetricsMethods method), 22
- log_proba (expandas.core.frame.ModelFrame attribute), 15
- ## M
- make_pipeline (expandas.skaccessors.pipeline.PipelineMethods attribute), 23
- make_union (expandas.skaccessors.pipeline.PipelineMethods attribute), 23
- manifold (expandas.core.frame.ModelFrame attribute), 15
- ManifoldMethods (class in expandas.skaccessors.manifold), 22
- mean_shift() (expandas.skaccessors.cluster.ClusterMethods method), 19
- metrics (expandas.core.frame.ModelFrame attribute), 15
- MetricsMethods (class in expandas.skaccessors.metrics), 22
- mixture (expandas.core.frame.ModelFrame attribute), 15
- ModelFrame (class in expandas.core.frame), 13
- ModelSeries (class in expandas.core.series), 16
- multiclass (expandas.core.frame.ModelFrame attribute), 15
- MultiClassMethods (class in expandas.skaccessors.multiclass), 22
- ## N
- naive_bayes (expandas.core.frame.ModelFrame attribute), 15
- neighbors (expandas.core.frame.ModelFrame attribute), 15
- NeighborsMethods (class in expandas.skaccessors.neighbors), 23
- ## O
- oas() (expandas.skaccessors.covariance.CovarianceMethods method), 19
- OneVsOneClassifier (expandas.skaccessors.multiclass.MultiClassMethods attribute), 23
- OneVsRestClassifier (expandas.skaccessors.multiclass.MultiClassMethods attribute), 23
- orthogonal_mp_gram() (expandas.skaccessors.linear_model.LinearModelMethods method), 22
- OutputCodeClassifier (expandas.skaccessors.multiclass.MultiClassMethods attribute), 23
- ## P
- pairwise (expandas.skaccessors.metrics.MetricsMethods attribute), 22
- partial_dependence (expandas.skaccessors.ensemble.EnsembleMethods attribute), 20
- partial_dependence() (expandas.skaccessors.ensemble.PartialDependenceMethods method), 20
- PartialDependenceMethods (class in expandas.skaccessors.ensemble), 20
- permutation_test_score() (expandas.skaccessors.cross_validation.CrossValidationMethods method), 19
- pipeline (expandas.core.frame.ModelFrame attribute), 15
- PipelineMethods (class in expandas.skaccessors.pipeline), 23
- plot_partial_dependence() (expandas.skaccessors.ensemble.PartialDependenceMethods method), 20
- precision_recall_curve() (expandas.skaccessors.metrics.MetricsMethods method), 22
- precision_recall_fscore_support() (expandas.skaccessors.metrics.MetricsMethods method), 22
- precision_score() (expandas.skaccessors.metrics.MetricsMethods method), 22
- predict() (expandas.core.frame.ModelFrame method), 15
- predict_ecoc() (expandas.skaccessors.multiclass.MultiClassMethods method), 23
- predict_log_proba() (expandas.core.frame.ModelFrame method), 15
- predict_ovo() (expandas.skaccessors.multiclass.MultiClassMethods method), 23
- predict_ovr() (expandas.skaccessors.multiclass.MultiClassMethods method), 23
- predict_proba() (expandas.core.frame.ModelFrame method), 15
- predicted (expandas.core.frame.ModelFrame attribute), 15
- preprocessing (expandas.core.frame.ModelFrame attribute), 15

preprocessing (expandas.core.series.ModelSeries attribute), 16
 PreprocessingMethods (class in expandas.skaccessors.preprocessing), 23
 proba (expandas.core.frame.ModelFrame attribute), 15
 train_test_split() (expandas.skaccessors.cross_validation.CrossValidationMethods method), 19
 transform() (expandas.core.frame.ModelFrame method), 16
 tree (expandas.core.frame.ModelFrame attribute), 16

Q

qda (expandas.core.frame.ModelFrame attribute), 15

R

recall_score() (expandas.skaccessors.metrics.MetricsMethods method), 22
 regression_models (expandas.skaccessors.gaussian_process.GaussianProcessMethods attribute), 21
 RegressionModelsMethods (class in expandas.skaccessors.gaussian_process), 21
 roc_auc_score() (expandas.skaccessors.metrics.MetricsMethods method), 22
 roc_curve() (expandas.skaccessors.metrics.MetricsMethods method), 22

S

score() (expandas.core.frame.ModelFrame method), 15
 semi_supervised (expandas.core.frame.ModelFrame attribute), 15
 silhouette_samples() (expandas.skaccessors.metrics.MetricsMethods method), 22
 silhouette_score() (expandas.skaccessors.metrics.MetricsMethods method), 22
 sparse_encode() (expandas.skaccessors.decomposition.DecompositionMethods method), 20
 spectral_clustering() (expandas.skaccessors.cluster.ClusterMethods method), 19
 spectral_embedding() (expandas.skaccessors.manifold.ManifoldMethods method), 22
 StratifiedShuffleSplit() (expandas.skaccessors.cross_validation.CrossValidationMethods method), 19
 svm (expandas.core.frame.ModelFrame attribute), 15
 SVMMethods (class in expandas.skaccessors.svm), 23

T

target (expandas.core.frame.ModelFrame attribute), 16
 target_name (expandas.core.frame.ModelFrame attribute), 16
 text (expandas.skaccessors.feature_extraction.FeatureExtractionMethods attribute), 20
 to_frame() (expandas.core.series.ModelSeries method), 16