`sklearn.feature_extraction`.DictVectorizer

class `sklearn.feature_extraction.`**`DictVectorizer`**(**, dtype=<class 'numpy.float64'>, separator='=', sparse=True, sort=True*) [source]

Transforms lists of feature-value mappings to vectors.

This transformer turns lists of mappings (dict-like objects) of feature names to feature values into Numpy arrays or scipy.sparse matrices for use with scikit-learn estimators.

When feature values are strings, this transformer will do a binary one-hot (aka one-of-K) coding: one boolean-valued feature is constructed for each of the possible string values that the feature can take on. For instance, a feature “f” that can take on the values “ham” and “spam” will become two features in the output, one signifying “f=ham”, the other “f=spam”.

If a feature value is a sequence or set of strings, this transformer will iterate over the values and will count the occurrences of each string value.

However, note that this transformer will only do a binary one-hot encoding when feature values are of type string. If categorical features are represented as numeric values such as int or iterables of strings, the DictVectorizer can be followed by `OneHotEncoder` to complete binary one-hot encoding.

Features that do not occur in a sample (mapping) will have a zero value in the resulting array/matrix.

For an efficiency comparison of the different feature extractors, see FeatureHasher and DictVectorizer Comparison.

Read more in the User Guide.

Parameters:

dtype : *dtype, default=np.float64*
The type of feature values. Passed to Numpy array/scipy.sparse matrix constructors as the dtype argument.

separator : *str, default=”=”*
Separator string used when constructing new features for one-hot coding.

sparse : *bool, default=True*
Whether transform should produce scipy.sparse matrices.

sort : *bool, default=True*
Whether `feature_names_` and `vocabulary_` should be sorted when fitting.

Attributes:

vocabulary_ : *dict*
A dictionary mapping feature names to feature indices.

feature_names_ : *list*
A list of length n_features containing the feature names (e.g., “f=ham” and “f=spam”).

See also:

`FeatureHasher`
Performs vectorization using only a hash function.

`sklearn.preprocessing.OrdinalEncoder`
Handles nominal/categorical features encoded as columns of arbitrary data types.

Examples

Toggle Menu

```
>>> from sklearn.feature_extraction import DictVectorizer
>>> v = DictVectorizer(sparse=False)
>>> D = [{'foo': 1, 'bar': 2}, {'foo': 3, 'baz': 1}]
>>> X = v.fit_transform(D)
>>> X
array([[2., 0., 1.],
       [0., 1., 3.]])
>>> v.inverse_transform(X) == [{'bar': 2.0, 'foo': 1.0},
...                            {'baz': 1.0, 'foo': 3.0}]
True
>>> v.transform({'foo': 4, 'unseen_feature': 3})
array([[0., 0., 4.]])
```

Methods

`fit`(X[, y])	Learn a list of feature name -> indices mappings.
`fit_transform`(X[, y])	Learn a list of feature name -> indices mappings and transform X.
`get_feature_names_out`([input_features])	Get output feature names for transformation.
`get_metadata_routing`()	Get metadata routing of this object.
`get_params`([deep])	Get parameters for this estimator.
`inverse_transform`(X[, dict_type])	Transform array or sparse matrix X back to feature mappings.
`restrict`(support[, indices])	Restrict the features to those in support using feature selection.
`set_inverse_transform_request`(*[, dict_type])	Request metadata passed to the `inverse_transform` method.
`set_output`(*[, transform])	Set output container.
`set_params`(**params)	Set the parameters of this estimator.
`transform`(X)	Transform feature->value dicts to array or sparse matrix.

fit(*X*, *y=None*) [source]

Learn a list of feature name -> indices mappings.

Parameters:

X : *Mapping or iterable over Mappings*

Dict(s) or Mapping(s) from feature names (arbitrary Python objects) to feature values (strings or convertible to dtype).

Changed in version 0.24: Accepts multiple string values for one categorical feature.

y : *(ignored)*

Ignored parameter.

Returns:

self : *object*

DictVectorizer class instance.

fit_transform(*X*, *y=None*) [source]

Learn a list of feature name -> indices mappings and transform X.

Like fit(X) followed by transform(X), but does not require materializing X in memory.

Parameters:

X : *Mapping or iterable over Mappings*

Dict(s) or Mapping(s) from feature names (arbitrary Python objects) to feature values (strings or convertible to dtype).

Changed in version 0.24: Accepts multiple string values for one categorical feature.

y : *(ignored)*

Ignored parameter.

Returns:

Xa : *{array, sparse matrix}*

Feature vectors; always 2-d.

Toggle Menu

`get_feature_names_out`(*input_features=None*) [source]

Get output feature names for transformation.

Parameters:

input_features : *array-like of str or None, default=None*
Not used, present here for API consistency by convention.

Returns:

feature_names_out : *ndarray of str objects*
Transformed feature names.

`get_metadata_routing`() [source]

Get metadata routing of this object.

Please check User Guide on how the routing mechanism works.

Returns:

routing : *MetadataRequest*
A `MetadataRequest` encapsulating routing information.

`get_params`(*deep=True*) [source]

Get parameters for this estimator.

Parameters:

deep : *bool, default=True*
If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns:

params : *dict*
Parameter names mapped to their values.

`inverse_transform`(*X, dict_type=<class 'dict'>*) [source]

Transform array or sparse matrix X back to feature mappings.

X must have been produced by this DictVectorizer's transform or fit_transform method; it may only have passed through transformers that preserve the number of features and their order.

In the case of one-hot/one-of-K coding, the constructed feature names and values are returned rather than the original ones.

Parameters:

X : *{array-like, sparse matrix} of shape (n_samples, n_features)*
Sample matrix.

dict_type : *type, default=dict*
Constructor for feature mappings. Must conform to the collections.Mapping API.

Returns:

D : *list of dict_type objects of shape (n_samples,)*
Feature mappings for the samples in X.

`restrict`(*support, indices=False*) [source]

Toggle Menu eatures to those in support using feature selection.

This function modifies the estimator in-place.

Parameters:

support : *array-like*
Boolean mask or list of indices (as returned by the get_support member of feature selectors).

indices : *bool, default=False*
Whether support is a list of indices.

Returns:

self : *object*
DictVectorizer class instance.

Examples

```
>>> from sklearn.feature_extraction import DictVectorizer
>>> from sklearn.feature_selection import SelectKBest, chi2
>>> v = DictVectorizer()
>>> D = [{'foo': 1, 'bar': 2}, {'foo': 3, 'baz': 1}]
>>> X = v.fit_transform(D)
>>> support = SelectKBest(chi2, k=2).fit(X, [0, 1])
>>> v.get_feature_names_out()
array(['bar', 'baz', 'foo'], ...)
>>> v.restrict(support.get_support())
DictVectorizer()
>>> v.get_feature_names_out()
array(['bar', 'foo'], ...)
```

`set_inverse_transform_request`(*, *dict_type: bool | None | str = '$UNCHANGED$'*) → DictVectorizer [source]

Request metadata passed to the `inverse_transform` method.

Note that this method is only relevant if `enable_metadata_routing=True` (see **`sklearn.set_config`**). Please see User Guide on how the routing mechanism works.

The options for each parameter are:

- `True`: metadata is requested, and passed to `inverse_transform` if provided. The request is ignored if metadata is not provided.
- `False`: metadata is not requested and the meta-estimator will not pass it to `inverse_transform`.
- `None`: metadata is not requested, and the meta-estimator will raise an error if the user provides it.
- `str`: metadata should be passed to the meta-estimator with this given alias instead of the original name.

The default (`sklearn.utils.metadata_routing.UNCHANGED`) retains the existing request. This allows you to change the request for some parameters and not others.

New in version 1.3.

Note: This method is only relevant if this estimator is used as a sub-estimator of a meta-estimator, e.g. used inside a **`Pipeline`**. Otherwise it has no effect.

Parameters:

dict_type : *str, True, False, or None, default=sklearn.utils.metadata_routing.UNCHANGED*
Metadata routing for `dict_type` parameter in `inverse_transform`.

Returns:

self : *object*
The updated object.

`set_output`(*, *transform=None*) [source]

Set output container.

See Introducing the set_output API for an example on how to use the API.

Toggle Menu

Parameters:

transform : *{"default", "pandas"}, default=None*

Configure output of `transform` and `fit_transform`.

- `"default"`: Default output format of a transformer
- `"pandas"`: DataFrame output
- `"polars"`: Polars output
- `None`: Transform configuration is unchanged

New in version 1.4: `"polars"` option was added.

Returns:

self : *estimator instance*

Estimator instance.

`set_params`(***params*) [source]

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as `Pipeline`). The latter have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

Parameters:

****params : *dict***

Estimator parameters.

Returns:

self : *estimator instance*

Estimator instance.

`transform`(*X*) [source]

Transform feature->value dicts to array or sparse matrix.

Named features not encountered during fit or fit_transform will be silently ignored.

Parameters:

X : *Mapping or iterable over Mappings of shape (n_samples,)*

Dict(s) or Mapping(s) from feature names (arbitrary Python objects) to feature values (strings or convertible to dtype).

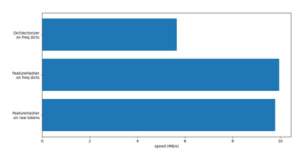
Returns:

Xa : *{array, sparse matrix}*

Feature vectors; always 2-d.

Examples using `sklearn.feature_extraction.DictVectorizer`

Column Transformer with Heterogeneous Data Sources


FeatureHasher and DictVectorizer Comparison

Toggle Menu

Toggle Menu