This page is to collate issues related to Pipelines and other meta-estimator API design. In general, a meta-estimator `M` with (primary) sub-estimator `S` should be more-or-less usable in place of `S`. Deficiencies in the current models mean this is not always the case; which of these deficiencies should be fixed and how? Other issues related to meta-estimator support (e.g. nested parameter setting) may also be relevant.

General meta-estimator issues

Duck-typing and methods (#1805, #2019)

FIXED

`hasattr` may be used to check an estimator supports a particular functionality (e.g. `fit_transform`, `predict_proba`). In meta-estimators this is conditioned on the presence of that method on a sub-estimator. This behaviour can be ensured using magic methods (`__getattr__` or `__getattribute__`) or using descriptors (e.g. `property`): when these raise `AttributeError`, `hasattr` returns false.

PR #2019 supports common methods using `property`, sacrificing some readability; it also tests for their conditional availability. The question of which common methods need to be supported is a further issue.

A further concern is that in traditional estimators, `hasattr` will work before or after fitting. If something like `GridSearchCV` delegates `hasattr` to its `best_estimator_`, this will only have effect *after* fitting.

Accessing fitted attributes (cf. #2561, #2568, #2630 wrt `Pipeline`)

It can be cumbersome to access a fitted attribute of an estimator (e.g. in a `Pipeline` within `GridSearchCV`, this may involve `gs.best_estimator_.steps[-1][1].coef_`). To be interpreted with respect to the input space, this may require further transformation (e.g. `Pipeline(gs.best_estimator_.steps[:-1]).inverse_transform(gs.best_estimator_.steps[-1][1].coef_)`).

Moreover, some fitted attributes are used by meta-estimators; `AdaBoostClassifier` assumes its sub-estimator has a `classes_` attribute after fitting, which means that presently `Pipeline` cannot be used as the sub-estimator of `AdaBoostClassifier`. Either meta-estimators such as `AdaBoostClassifier` need to be configurable in how they access this attribute, or meta-estimators such as `Pipeline` need to make some fitted attributes of sub-estimators accessible.

Pipeline / FeatureUnion issues

Passing parameters such as `sample_weight` to methods (cf. #2630)

It should be possible to pass more than `X` and `y` to a pipeline's `fit`, `score`, etc. methods, and the most common use-case is `sample_weight`. `Pipeline.fit` presently forwards kwargs to individual sub-estimators based on a prefixing system (e.g. `fit(X, y, clf__sample_weight=...)`). This is not usable within `AdaBoost*` or grid search (pending support) which expect to provide the `sample_weight` kwarg.

`sample_weight` cannot be passed to all estimators of a pipeline; some will not have support. Implicitly detecting support is not future-proof (i.e. a transformer may introduce support for `sample_weight`, affecting earlier code). @jcrudy has suggested specifying the forwarding paths for kwargs in the `Pipeline` constructor:

```
Pipeline([('t1', t1), ('t2', t2), ('c', c)], forward_kwargs={'sample_weight': ['t1', 'c']})
```

Note that this would be presumed to apply for any `Pipeline` method to which `sample_weight` is passed (e.g. `fit`, `score`).

`Pipeline.get_feature_names()` (#2007)

Feature extractors provide `get_feature_names()` to identify the extraced features and their ordering. `FeatureUnion.get_feature_names` concatenates the output of this function on each of its transformers. It would be similarly to get the feature names *produced* by a transformation `Pipeline`. This faces the following problems:

- not all transformers support `get_feature_names()`, and the main extractor may not be the first Pipeline step.
- many transformers alter the set of features, and `get_feature_names()` must apply corresponding feature selection masks (this is just `selector.transform(names)`).
- some transformers produce features that cannot be named (e.g. PCA, random projection)
- where the features are already extracted to array form (i.e. the pipeline does not contain a feature extractor), it would also be useful to get the transformed feature names given a set of input feature names

@kmike's suggestion is to extend the `get_feature_names()` API to take an optional list of input feature names, allowing them to be transformed and output.

[Aside: It's possible that scikit-learn's handling of feature names needs reconsideration. For example, building a feature selector by name is best implemented as a feature-extracting meta-estimator, which seems awkward:

```
class SelectByName(FeatureSelectorMixin, BaseEstimator):
    def __init__(self, extractor, selected_names): ...
    def fit(self, X, y):
        names = {name: i for i, name in enuemrate(self.extractor.fit(X,
y).get_feature_names())}
        self.indices_ = [names[name] for name in self.selected_names]
    def transform(self, X):
        return self.extractor.transform(X, y)[:, self.indices_]
```

]

Inconsistency between `get_params` and `set_params` treatment of sub-estimators (#1769, #1800)

FIXED

In order to support the `__` meta-estimator parameter notation, `get_params` needs to return a mapping to sub-estimators from their parameter prefixes (e.g. 'clf' in 'clf__C'). In `Pipeline` and `FeatureUnion`, unlike other meta-estimators, these prefixes do not correspond to attributes. Hence using `set_params` with one of these prefixes is broken (and may overwrite an existing attribute!) but raises no error.

Solutions include:

- Make it possible to set steps using `set_params` (or, indeed, as attributes).
- Do not use `get_params` to map parameter name prefixes, and introduce `get_sub_estimators` to return this mapping.

Efficiently reusing partial models/transformations during grid search (#2086)

FIXED

Pipelines and FeatureUnions in grid search may perform a lot of redundant work in fitting and transforming the same data for sub-pipelines. Caching these partial results may provide great efficiency gains; the simplest solution is to allow individual estimators to be cached, without special handling at the `Pipeline` level.

Doesn't clone

Minor functionality and syntax issues:

- constructor verbosity due to naming (#2589) FIXED
- alternating or disabling components through `set_params()` (#1769) FIXED
- retrieving a final model in input feature space (#2561, #2568)
- heterogeneous input in `FeatureUnion` (#2034)
- partitioning the `FeatureUnion` output space by transformer (#1952)