A recap on Scikit-learn's estimator interface

Scikit-learn strives to have a uniform interface across all methods. Given a scikit-learn *estimator* object named `model`, the following methods are available (not all for each model):

- Available in **all Estimators**
  - `model.fit()` : fit training data. For supervised learning applications, this accepts two arguments: the data `X` and the labels `y` (e.g. `model.fit(X, y)`). For unsupervised learning applications, `fit` takes only a single argument, the data `X` (e.g. `model.fit(X)`).
- Available in **supervised estimators**
  - `model.predict()` : given a trained model, predict the label of a new set of data. This method accepts one argument, the new data `X_new` (e.g. `model.predict(X_new)`), and returns the learned label for each object in the array.
  - `model.predict_proba()` : For classification problems, some estimators also provide this method, which returns the probability that a new observation has each categorical label. In this case, the label with the highest probability is returned by `model.predict()`.
  - `model.decision_function()` : For classification problems, some estimators provide an uncertainty estimate that is not a probability. For binary classification, a decision_function >= 0 means the positive class will be predicted, while < 0 means the negative class.
  - `model.score()` : for classification or regression problems, most (all?) estimators implement a score method. Scores are between 0 and 1, with a larger score indicating a better fit. For classifiers, the `score` method computes the prediction accuracy. For regressors, `score` computes the coefficient of determination (R^2) of the prediction.
  - `model.transform()` : For feature selection algorithms, this will reduce the dataset to the selected features. For some classification and regression models such as some linear models and random forests, this method reduces the dataset to the most informative features. These classification and regression models can therefore also be used as feature selection methods.
- Available in **unsupervised estimators**
  - `model.transform()` : given an unsupervised model, transform new data into the new basis. This also accepts one argument `X_new`, and returns the new representation of the data based on the unsupervised model.
  - `model.fit_transform()` : some estimators implement this method, which more efficiently performs a fit and a transform on the same input data.
  - `model.predict()` : for clustering algorithms, the predict method will produce cluster labels for new data points. Not all clustering methods have this functionality.
  - `model.predict_proba()` : Gaussian mixture models (GMMs) provide the probability for each point to be generated by a given mixture component.
  - `model.score()` : Density models like KDE and GMMs provide the likelihood of the data under the model.

Apart from `fit`, the two most important functions are arguably `predict` to produce a target variable (a `y`) `transform`, which produces a new representation of the data (an `X`). The following table shows for which class of models which function applies:

``model.predict``	**``model.transform``**
Classification	Preprocessing
Regression	Dimensionality Reduction
Clustering	Feature Extraction
	Feature Selection