

There are three key API parts to scikit learning design interface: estimators, transformers and predictors. SciKit-Learn is a library and not a framework that is easy to invoke by importing packages and then instantiating objects from classes.

With estimators, by taking the dataset as a parameter or two for supervised learning algorithms, estimators use the fit() method for data sets. In the case of supervised learning, through the fit method that takes the input feature vectors and labels, the estimator trains on novel data for predictions. The instantiation of an Estimator is separate from the learning process where one fits the model with training data. In constructing an estimator, hyper-parameters are passed in and training data is passed to the fit method. Imputer's strategy is a hyperparameter that is set as a constructor parameter. The Estimator is key to estimation and “decoding” of a fit model, using a random state of numpy.random to become a deterministic function(Brennan). The creators of the library could have added a hierarchical class but chose not to for accessibility reasons. It is represented by the “partial function application” as in some arguments are frozen to one function and some arguments in one function is passed to another higher order function. Functools.partial supports this pattern.

Transforms mutate a dataset through “preprocessing feature selection feature extractoion and dimensionality reduction algorithms are all provided” (Buitinck et al.). In the API the transform() method uses the parameter of the data set to return a transformed dataset using learned parameters such as the imputer, TransformerMixin class and StandardScaler transformer which is also an Estimator. After fitting the transform method can be used, and the fit method returns an estimator called upon. Conveniently, there is also a method of fit() and transform() as well as there is a fit_transform() that does both.

With predictors that can be summoned within as little as three lines of code, there are also predictors that make educated guesses on the estimated dataset and are separate from Estimators; they extend the interface of Estimators. Given that test features utilized by the model and the predict method through the Predict interface; the predict method is derived from the estimator via inheritance. The predict() method inputs a dataset and returns a dataset of parallel predictions for the inputted feature vector, also returning prediction scores and probabilities. The score() method derived from the “mixin” class indicates the quality of given predictions with regards to the test set and labels in a supervised learning algorithm. The “mixin” classes package and separate functionality with respect to simple inheritance.

Inspection where public instance variables like imputer strategy access hyperparameters and the learned parameters of the estimator. There is nonproliferation of classes where familiar and highly accessible NumPy, optimized by C, and SciPy represent the matrices, and regular Python strings and numbers that make up hyperparameters. Through transparency, reusing the building blocks are accessible via simplified public interface; the example of the pipeline estimator builds a reusable final estimator through chaining a pipeline of sequential and multiple transformative methods especially when fit_transform() is called. Fit_transform() returns the transformed data through parsimonious code. There are sensible defaults in the system that give reasonable default values with respect to most parameters, creating an easy baseline system. There is also allowance of extensions of the library through Duck-Typing and avoiding inheritance and allowing for extension of the library through writing a custom transformer; this allows objects to used interchangeably with other elements in the same interface.

Reference:

Brennan, M. A. (2019, September 25). *Scikit-Learn Design Principles*. Medium.
<https://towardsdatascience.com/scikit-learn-design-principles-d1371958059b>.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., ... & Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*.

Géron, A. (2020). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems*. O'Reilly.