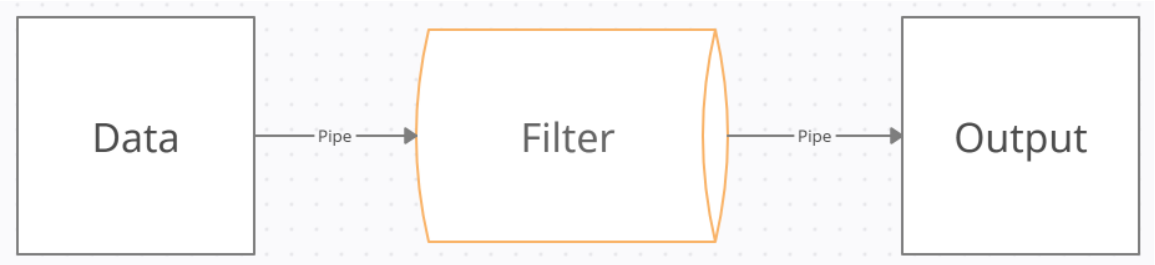
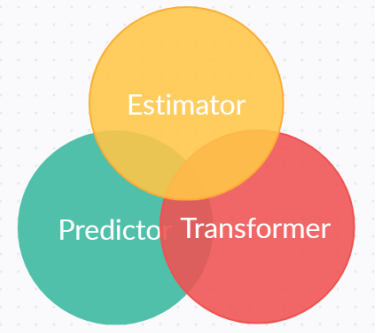
Overall Architecture of Scikit-learn

Scikit-learn is an open-source machine learning library for python. The main goal of Scikit-learn is to "provide efficient ...machine learning tools that are accessible to non-machine learning experts". This means that the library is intended for an audience that may not know much about machine learning, and so must rely on heavy abstraction of concepts and algorithms, implying the existence of an API. In addition, Scikit-learn is designed to work in conjunction with numeric and scientific oriented packages NumPy and SciPy to boost performance in which Scikit-learn can process data from directly, meaning there is some dependency between the corresponding packages. Scikit-learn must be able to do some hidden processing to transform data and train ML models.

Furthermore, Scikit-learn can apply multiple ML algorithms to the same data to create more accurate and complex models. Scikit-learn is able to process information in many steps. Familiarizing of the codebase reveals that Scikit's architecture can be thought of a Pipe and Filter design pattern.



[overall architecture of the system]



[Interfaces]

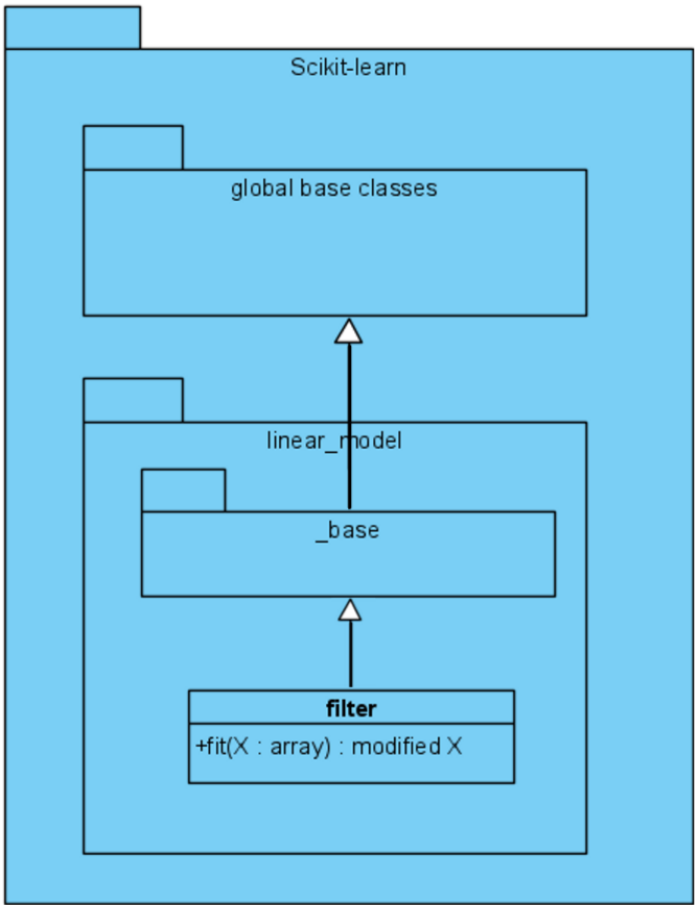
Scikit-learn initially takes input data and applies a "filter" to process that data. The initial "filter"can be an estimator or transformer. The predictor interface extends the estimators interface and can only be applied after an estimator has been initially applied. All of the implementation details for choosing any specific filter (or algorithm) are abstracted away behind an API.

In this way, Scikit-learn maintains highly expandable code. Individual "filter" modules can be added and constantly improved upon without affecting any other part of the application. In addition, users can combine multiple "filters" to suit their

specific ML needs in any application. As Scikit-learn is open sourced, it is an open architecture software.

Scikit-learn API

Scikit's API allows the library to effectively encapsulate all of their implementation related code through the strategy behavior pattern.



[filter interface]

All "filters" (represented by classes) extend the sklearn.base interface and therefore implement its function fit(). Different machine learning algorithms aka "filters" are interchangeable at runtime. Further interfaces narrow down requirements for specific algorithms such as in the sklearn.linear_model family where an additional sklear.linear_model._base.py includes a revised interface that extends sklearn.base. In addition, the __init__ file unique to each module allows Scikit-learn to effectively tailor requirements of each specific algorithm while maintaining high cohesion. Finally, Scikit-learn keeps low coupling between "filters" aka. ML algorithms, as each algorithm is self contained and doesn't rely on files outside of its package.

Links to related files:
base interface in linear_model

- https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/base.py

_base interface in linear_model

- https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/linear_model/_base.py

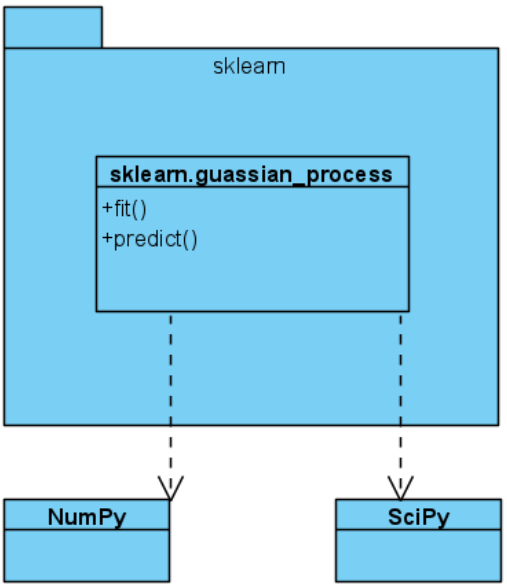
__init__

- https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/linear_model/__init__.py

“Filters" are each ML learning algorithm

- https://github.com/scikit-learn/scikit-learn/tree/main/sklearn

Coupling



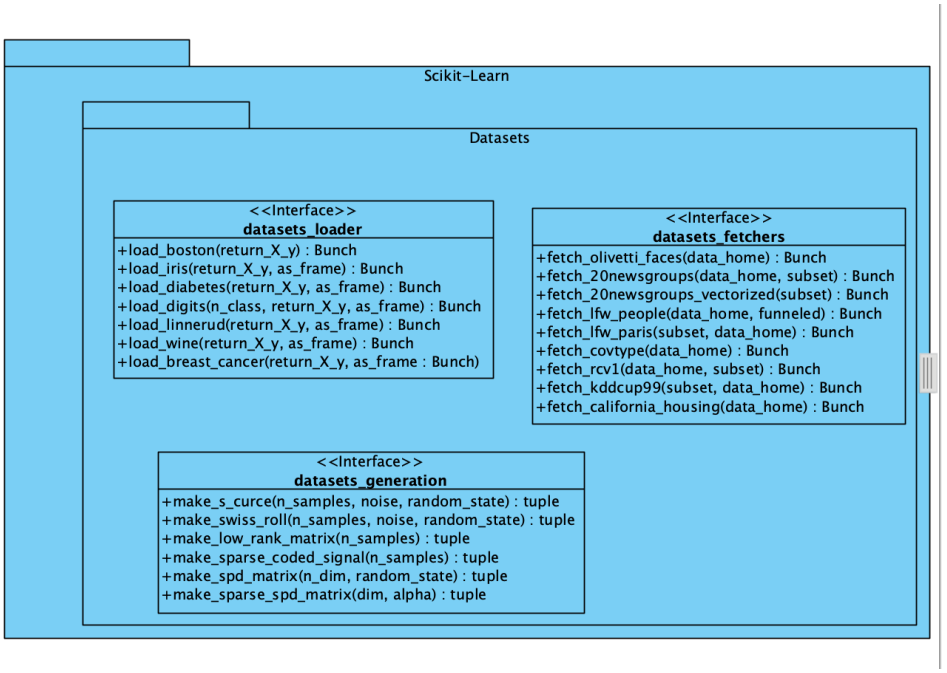
[scikit uses numpy and scipy]

Scikit-learn relies heavily on NumPy and SciPy formats to the point that a user could import data directly from a NumPy array into SciKit’s API calls, in this sense, Scikit-learn is externally coupled. Built upon NumPy and SciPy allows for high-performance linear algebra and array operation. Scikit-learn is also integrated with many other Python libraries: Matplotlib/Plotly for plotting, pandas for data frame to name a few.

Links to files importing NumPy and SciPy as an example:
Gaussian Process as an example, imports numpy as np

- https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/gaussian_process/_gpr.py

Data



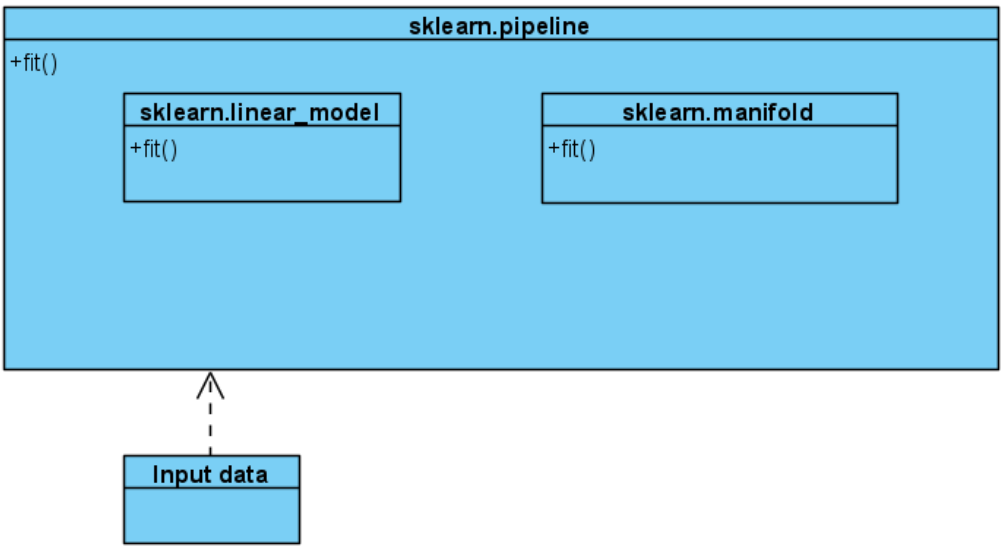
[dataset interfaces]

Scikit-learn offers some data management functionality in addition to its API. The sklearn.dataset embeds some small datasets which are dictionary-like objects that contain data and metadata. There are three main kinds of dataset interfaces: the dataset loaders, the dataset fetchers and the dataset generation functions. These three interfaces are used to get datasets depending on the desired type of dataset. The dataset is helpful for quickstart but bloats the codebase, we believe instead a tutorial video with data publicly sourced elsewhere would be better.

Links to dataset and related files:

- https://github.com/scikit-learn/scikit-learn/tree/main/sklearn/datasets

Pipeline and OOP



[pipeline object]

Scikit-learn includes a pipeline feature used to chain Estimators together to form more complex models of data. Using a pipeline allows users to fit, predict and transform over multiple Estimators in the pipeline all at once. The object-oriented design of scikit-learn achieves this by representing each Estimator algorithm as a class, and placing those objects inside a sklearn.pipeline object at runtime. The pipeline then works as any other Estimator would and applies fit() to each Estimator inside, a process only possibly with OOP.

Link to pipeline file:

- https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/pipeline.py