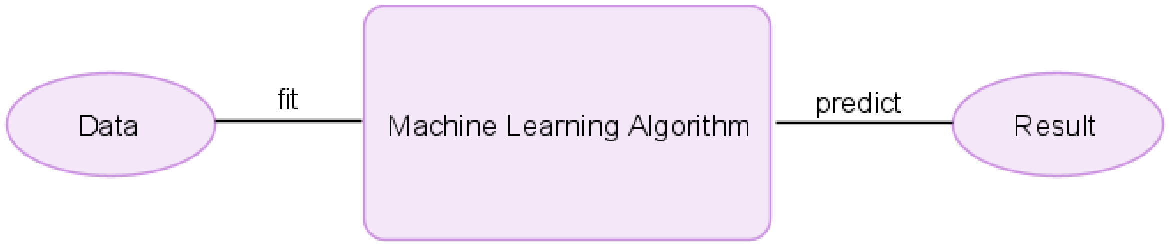
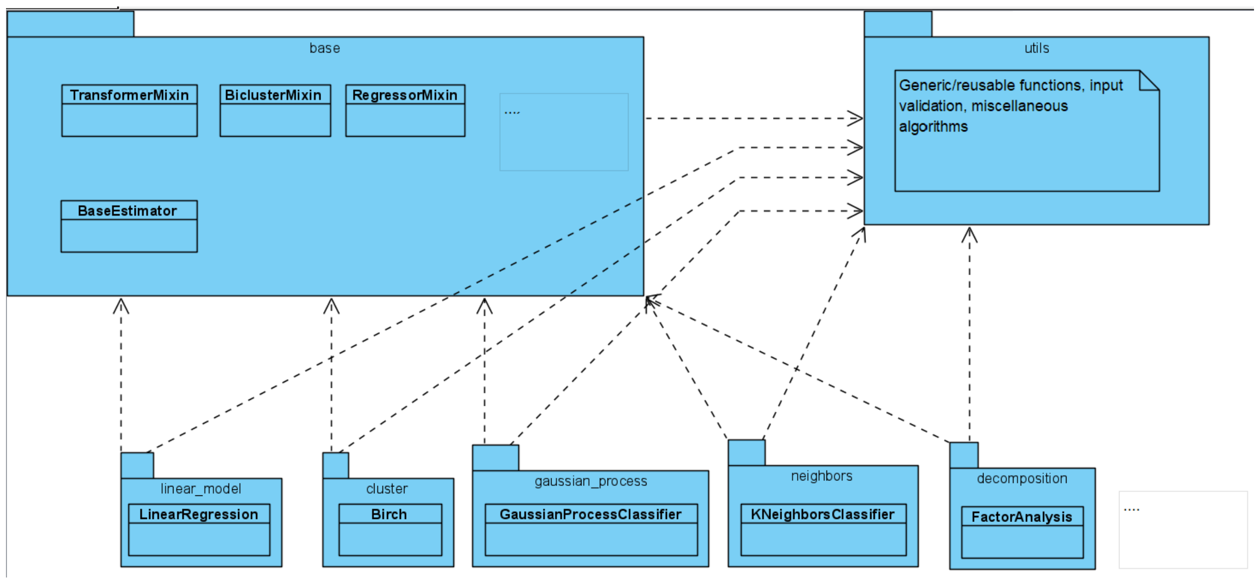
Software Architecture of Scikit Learn

Basic background:

Scikit learn is a machine learning library for Python users. The main use case is to train models using a dataset to predict results on new data. There are many different algorithms (some very complex and some simple) that scikit-learn provides. The usage of Scikit learn is wide in the statistical realm but the main focus is on Machine Learning, some other functionalities of Scikit Learn include Statistical Analysis and Data Modeling.



High level overview



Each of the bottom packages (linear_model, cluster, gaussian_process, etc) exposes an API of algorithms/classes for the library user. For example, a user who wants to use the birch clustering algorithm will import from the cluster package and use the exposed Birch class. All of these packages heavily rely on the base package (base.py in the source) and the utils package. The utils package holds all the generic/reusable functions that almost all other packages depend on. Some examples include input validation, generic graph search algorithms, randomization functions, and math/matrix operations. The base package contains fundamental interfaces and base classes in which many of the packages derive and implement (see below for more details). In this system, there is low coupling. Most of the coupling comes from the bottom packages with base.py and the utils package (which intuitively makes sense). There is also some light coupling in between the bottom packages, for example some might make use of the metrics package or the preprocessing package. However, in general coupling is low.

References to code

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

Various classes in Utils including class_weight.py, graph.py, optimize.py, and _testing.py
https://github.com/scikit-learn/scikit-learn/tree/main/sklearn/utils

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

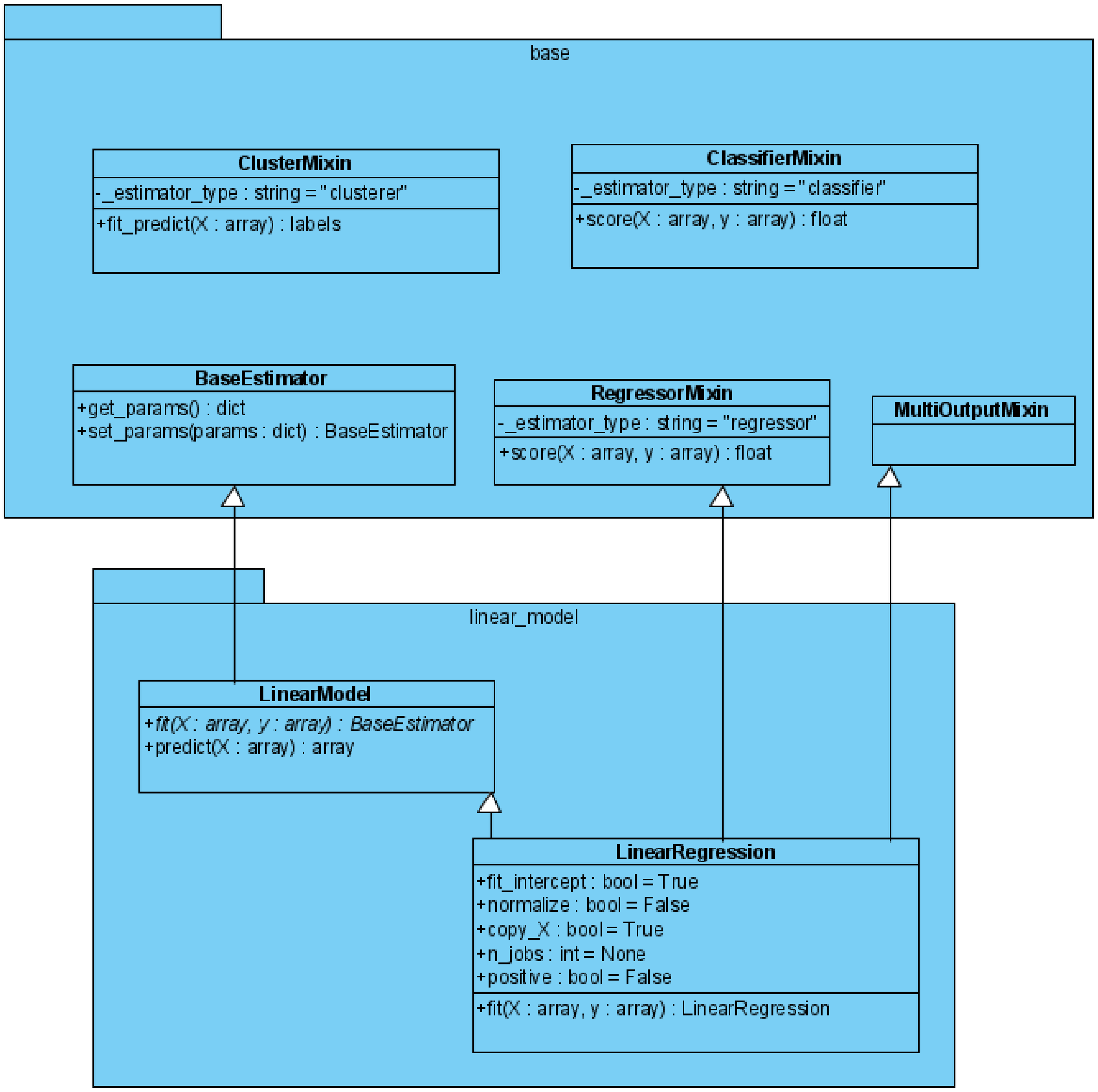
Cluster _birch.py
https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/cluster/_birch.py

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

Neighbors _base.py and _classification.py
https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/neighbors/_base.py
https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/neighbors/_classification.py

Decomposition _factor_analysis.py
https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/decomposition/_factor_analysis.py

Detailed overview

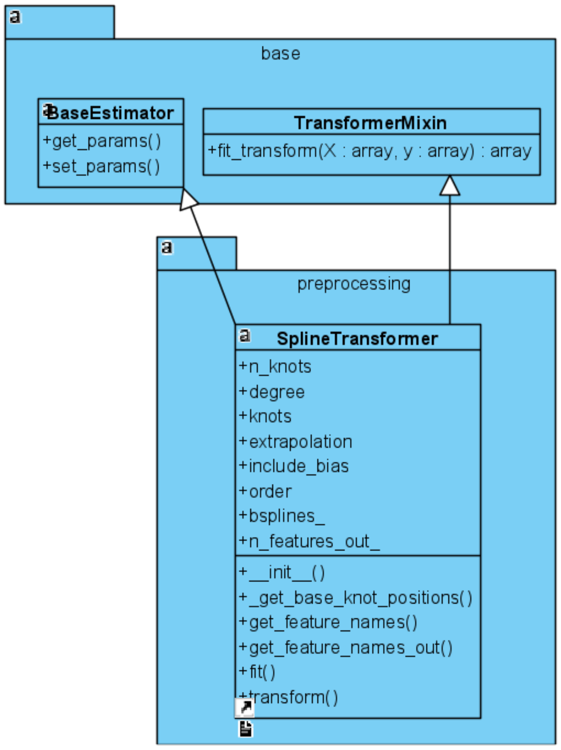

base
ClusterMixin
-_estimator_type : string = "clusterer"
+fit_predict(X : array) : labels
ClassifierMixin
-_estimator_type : string = "classifier"
+score(X : array, y : array) : float
BaseEstimator
+get_params() : dict
+set_params(params : dict) : BaseEstimator
RegressorMixin
-_estimator_type : string = "regressor"
+score(X : array, y : array) : float
MultiOutputMixin
linear_model
LinearModel
+fit(X : array, y : array) : BaseEstimator
+predict(X : array) : array
LinearRegression
+fit_intercept : bool = True
+normalize : bool = False
+copy_X : bool = True
+n_jobs : int = None
+positive : bool = False
+fit(X : array, y : array) : LinearRegression

In scikit-learn there are four main objects: estimators, predictors, transformers, and models. The LinearRegression class is an estimator, predictor, and model (an example of a transformer shown below). The LinearRegression class uses internal algorithms based on the parameters given to the constructor to perform a linear regression on the input data. This is similar to how other exposed classes work: they give the user access to the fit(...), predict(...), and score(...) methods so they don't need to worry about the inner algorithmic details.

A few key classes include: BaseEstimator, ClusterMixin, ClassifierMixin, and RegressorMixin. There are many more mixin classes depending on the specific algorithm used. The mixin classes will be inherited by subclasses that need it, kind of like an interface (but not really because Python uses multiple inheritance). Almost all user-exposed classes from any package will be a descendant of BaseEstimator and will probably inherit from multiple mixins.

The cohesion is high in this system because every package works well within itself. For example, in the linear_model package the LinearModel class is reused multiple times within the package (LinearRegression, ARDRegression, BayesianRidge, HuberRegressor, etc).

Below is an example of a transformer.



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

Linear model _base.py, _bayes.py
https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/linear_model/_base.py
https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/linear_model/_bayes.py
Spline transformer
https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/preprocessing/_polynomial.py