CSCD01 Keycap Guardians

Assignment #1: Architecture and Design Patterns

Su Tong Kong
Ivan Shao
Jeremy Tanuan
Mengqi Zhao
Haolong Zhang
Ho Ki Yuen
Kelvin Tan

x *- external*

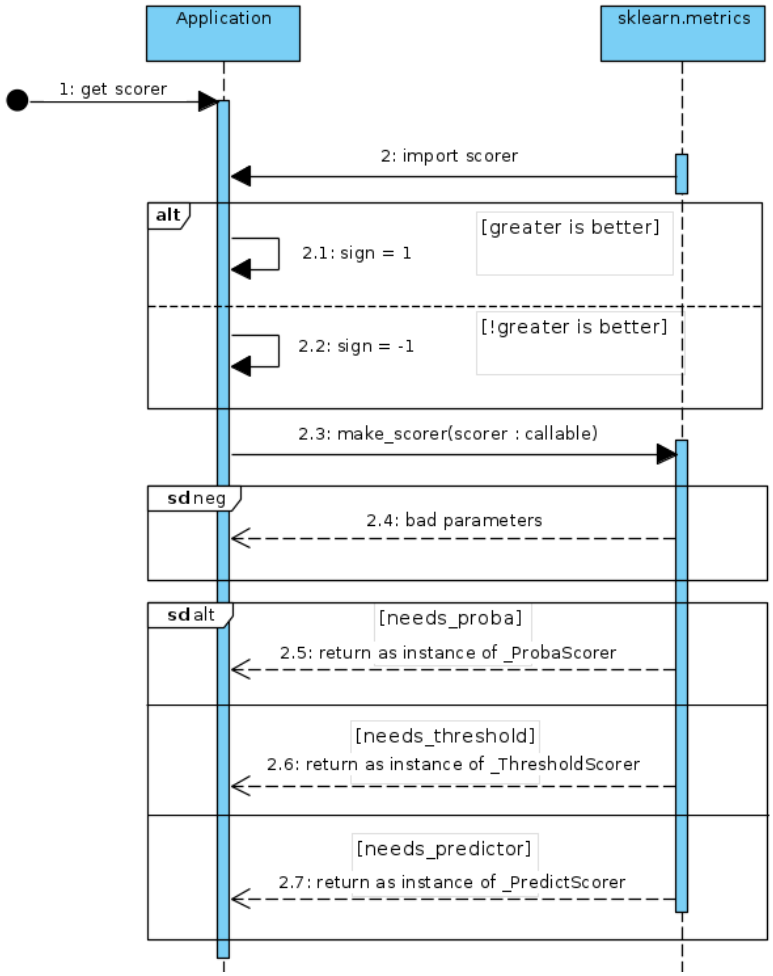
Design pattern 1: Factory Method Pattern

When we want to use a scorer object in our application, we do so by calling the make_scorer() function. It decides which kind of scorer object to create (either probabilityScorer, thresholdScorer, or predictionScorer) based on which algorithm is used for the scorer (without exposing the exact class of the object that will be created), then it returns the callable scorer object to our application. Thus the scorer classes in _scorer.py form a **factory method pattern**.

Class Diagram

*Diagram attached separately since it doesn't fit.

Sequence diagram



References to Code

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

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

Design pattern 2: Builder Pattern

The Builder Pattern is located in “sklearn/tree/_tree.pxd” and “sklearn/tree/_tree.pxy”. These files are written in Cython and therefore the syntax follows a combination of C and Python. Cython uses .pxd files to declare global variables, functions, and classes, similar to a .h header file in C, while .pxy files implement the functions and classes defined in the .pxd file. The class Tree is a binary tree used for predictions and feature importances. It is a generalization of Depth-First Tree and Best-First Tree which have different criterias to split and classify the data. TreeBuilder is an interface that defines the variables and methods that must exist in concrete builders. Then, DepthFirstTreeBuilder and BestFirstTreeBuilder extends TreeBuilder and implement those functions. DepthFirstTreeBuilder and BestFirstTreeBuilder construct a Tree differently with the function “build()”. Notice build() does not return a Tree, instead, it takes in a pointer to the tree that it will build upon. After calling build(), the caller will obtain a Depth-First Tree or Best-First Tree at the memory address of the pointer that was passed into build(). We can observe BestFirstTreeBuilder has the variable min_impurity_decrase and the function _add_split_node which do not exist in DepthFirstTreeBuilder.

Class Diagram

*Diagram attached separately since it doesn’t fit.

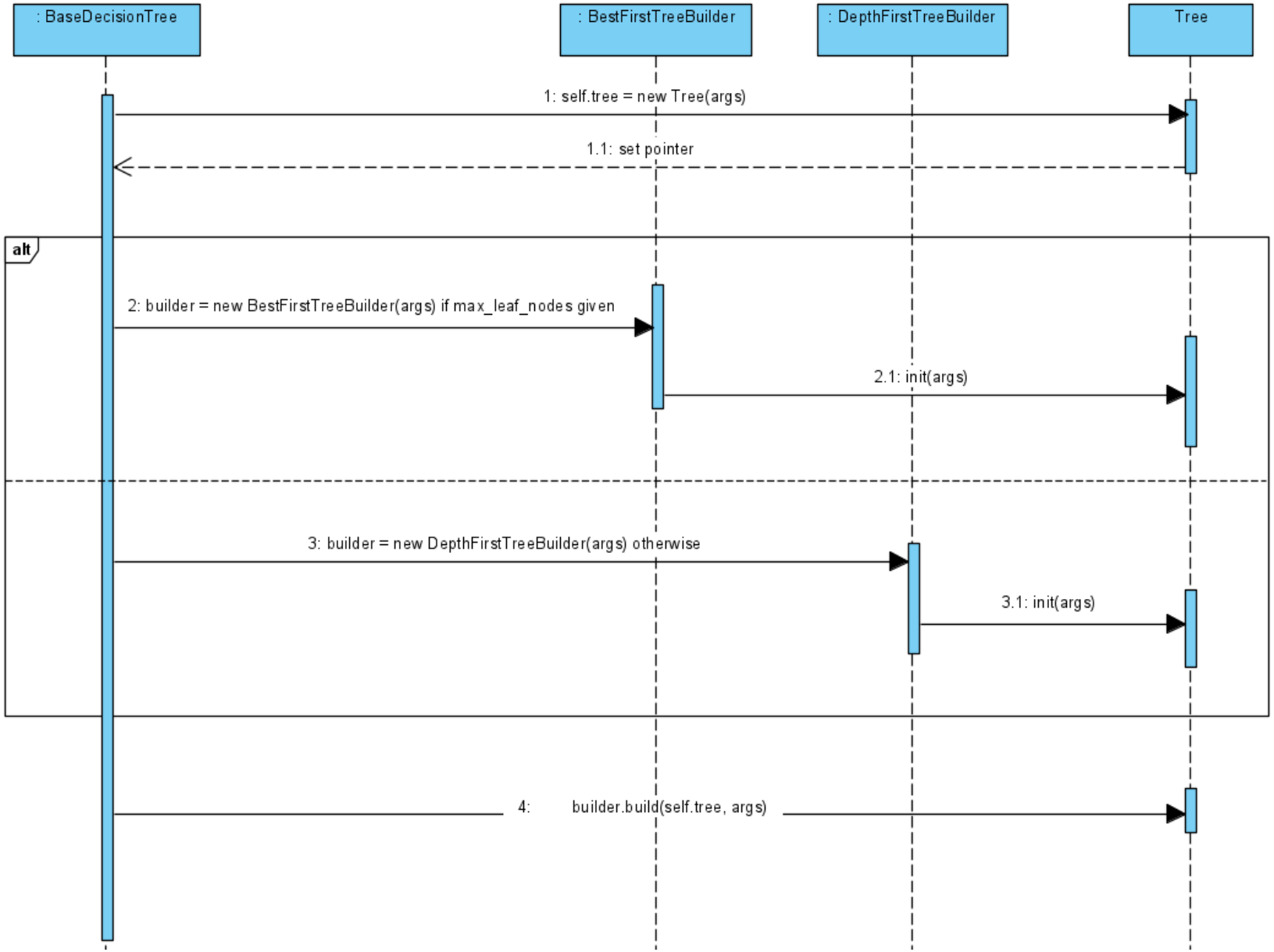
References to Code

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

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

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

Builder Pattern Sequence diagram



Design pattern 3: Template Method Pattern

We found a template method pattern in _univariate_selection.py. If a user wants to select a single variable of a dataset, the user can construct a transformer by calling the GenericUnivariateSelect() class while specifying the score function, the feature selection mode and a multiplier based on the mode. Then, by passing in the dataset as a variable to the fit_transform() method of the transformer, the univariate feature is selected. During the whole process, the GenericUnivariateSelect() class was inherited from its parent class which is _Basefilter and invokes the initialization of it. After that, two abstract methods are overridden, namely _get_support_mask() and _check_params(). And it also implemented a new hook method, which is _make_selector().

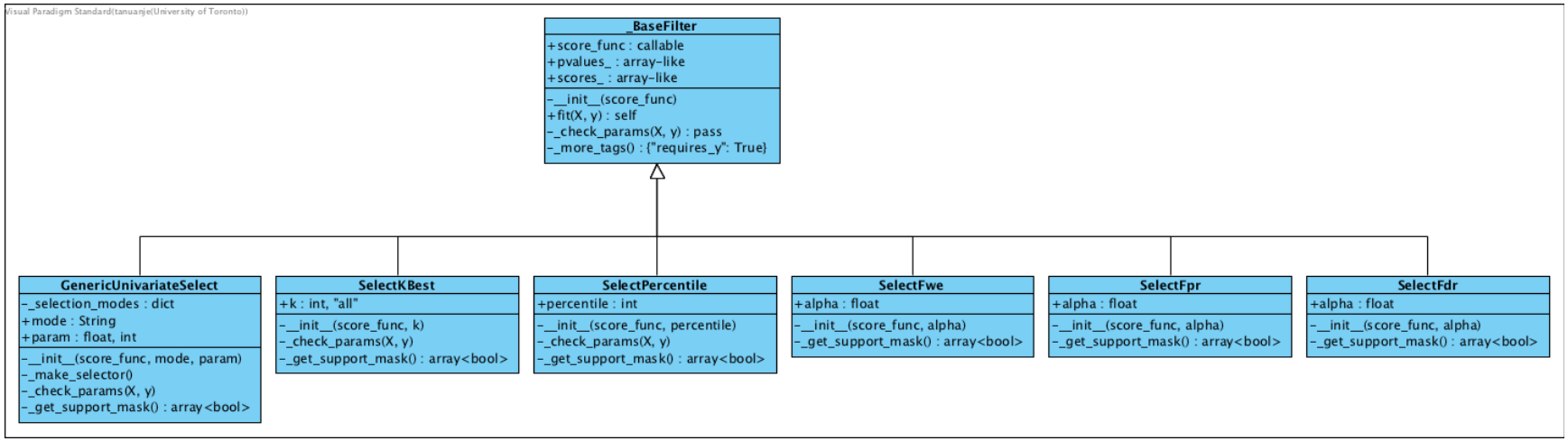
References to code

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

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

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

Template Method Class Diagram



Template Method Sequence Diagram