# Optuna

1. Links

* Website: <https://optuna.org/> (Code examples)
* Docs: <https://optuna.readthedocs.io/en/stable/>
* Github: <https://github.com/optuna/optuna>
* Article: see slides

1. Search algorithms

* Grid search
* Random search
* TPE
* CMA-ES
* Multi-objective sampler using the NSGA-II algorithm
* Multi-objective sampler using the MOTPE algorithm

1. Objective function

* Objective function created by the user
* Define the hyperparam space within the objective function

1. Distributions

* Samples Reals, Integers, and Categories
* Uniform and log-uniform distributions
* Define-by-run API -> Nested spaces

1. Acquisition function

* Uses function described in original work that introduced each algorithm
* Can’t choose which function to use
* EI
* Expected hypervolume improvement (EHVI)

1. Search analysis

* Can store the search in a SQL like DB
* Study object returns a dataframe with search data
* Built-in functions for plotting

1. Parallelization – SQLite

* Allows search in parallel storing in SQL like database

# Main functions

1. Main setup

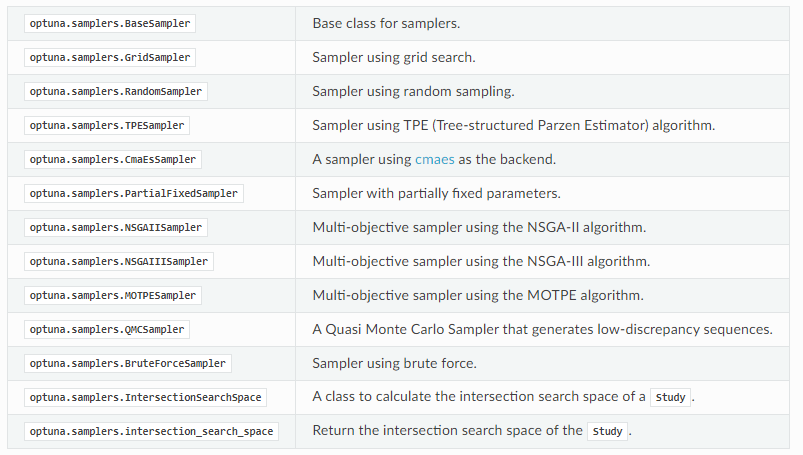


1. Optuna – create\_study

* optuna.create\_study() – Parameters:
* storage: url to a database
* sampler: the hyperparameter search algorithm (defo: tpe)
* pruner: the algorithme to prune unsuccessful trails (defo: MedianPruner)
* direction: ‘minimize’ or ‘maximize’
* study\_name: name for the study, in case saved and retrieved later

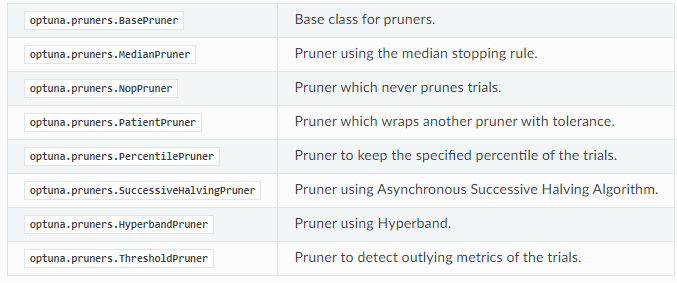
1. Optuna – samplers

* The samplers module defines a base class for parameter sampling as described extensively in BaseSampler. The remaining classes in this module represent child classes, deriving from BaseSampler, which implement different sampling strategies.



1. Optuna – pruner

* The pruners module defines a BasePruner class characterized by an abstract prune() method, which, for a given trial and its associated study, returns a Boolean value representing whether the trial should be pruned.
* This determination is made based on stored intermediate values of the objective function, as previously reported for the trial using optuna.trial.Trial.report()
* The remaining classes in this module represent child classes, inheriting from BasePruner, which implement different pruning strategies



1. Optuna – trials

* optuna.trial.Trials()
* Used under the hood by Optuna.study.Study.optimize()
* We only need the below for the objective function

