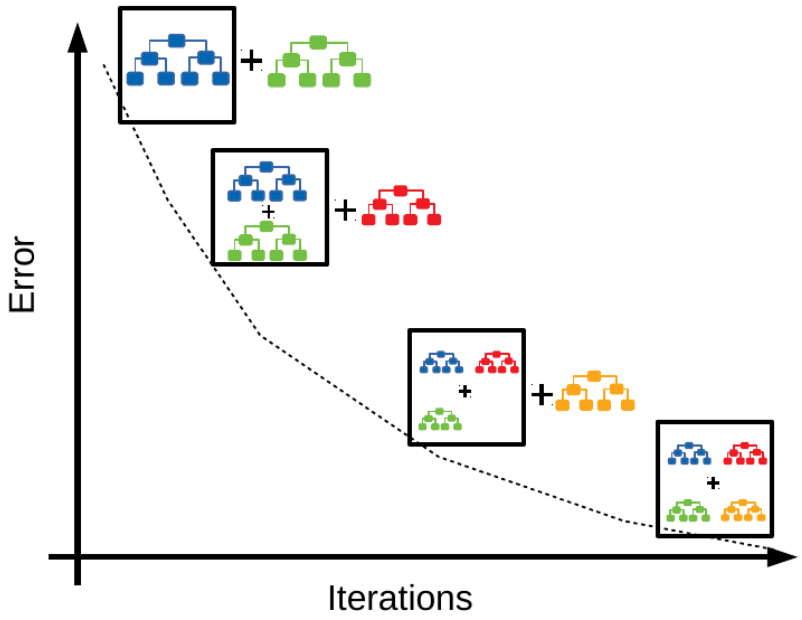
Xgboost is an algorithm that has dominating applied machine learning and Kaggle competitions before the development of deep neural network but that still work very well for structured or tabular data (Chen T, Guestrin C. XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD'16. August 13-17, 2016: 785–94. https://doi.org/10.1145/2939672.2939785).

The principle of boosting is to add sequentially small trees to correct the residuals of the prior sequence of tree until no improvement in an error function is observed. The error is minimized using the gradient descent algorithm when adding new models. The principle is schematized in the figure below (https://medium.com/swlh/gradient-boosting-trees-for-classification-a-beginners-guide-596b594a14ea)



As Tianqi Chen, the authors of Xgboost library said about Xgboost in comparison to classical boosting algorithms: "Xgboost used a more regularized model formalization to control over-fitting, which gives it better performance in comparison to classical gradient boosting approaches. The name xgboost, though, actually refers to the engineering goal to push the limit of computations resources for boosted tree algorithms. Which is the reason why many people use xgboost. For model, it might be more suitable to be called as regularized gradient boosting".

They are a few hyperparameters that have been tune in the present work with this algorithm (https://parsnip.tidymodels.org/reference/boost_tree.html):

- mtry corresponds to the number of predictors that will be randomly sampled at each split when creating the tree models (mtry, between 1 and the number of predictors)
- min_n corresponds to the minimum number of data points in a node required for the node to be split further. (min_n, between 1 and 40)
- tree_depth corresponds to the maximum depth of the tree (tree_depth, between 1 and 15)
- learn_rate correposnds to the rate at which the boosting algorithm adapts from iteration to iteration (learn_rate, between 0 and 0.08)

A semi-random grid of 30 combinations of the candidate parameters described above was created (https://tune.tidymodels.org/reference/tune_grid.html). This number allows a good compromise between time of run and finding of a good combination of parameters and has been previously successfully used (Woillard et al CPT 2021, PMID=33253425).