In your algorithms, when you use Gradient Boosting, do you prefer RandomSearchCV or GridSearchCV in order to optimize your hyperparameters ?

ANSWER:

It depends on the size of your multi dimensional grid. If it is small, then you can afford to be exhaustive and do a grid search. But if it is very large, and your computation time for a grid search extends too much, then definitely go to a random search. In fact, with random search one can explore larger regions than with grid search, and that is an advantage.

In any case, for hyperparameters search there are two keys:

1. Monitor it while it is running, so that you can stop it when you are happy with the results. This is specially important for random search.
2. Split your data in three, train, cross validation and test. Evaluate the hyperparameter search in the cv set. Once finished, rank them by their performance there, and then take the best point and re-evaluate in test. It may happen that you do what is called "overfitting to cv set", in which case the performance in test set will deteriorate significantly vs the performance in cv set. If that happens, try with the second best, third best etc until the performance in test set is good. But take into account that the more points you take from the ranking, the higher the chance of "overfitting to test set".

Once you find the best regions, you can do either perform a more constrained there with either grid search or random search again.

Another option, which tends to work very well, is **bayesian optimisation**.

Here the library that you use is important. In Python, after trying several which gave different problems, the best I found was skopt: <https://scikit-optimize.github.io/>