**Hyper parameters and how to Tune them**

**Parameters:**

Parameters are components of the model and are necessary to make predictions. The parameters are decided by the “math” of the algorithm.

**Hyper parameters:**

They are the “settings” of the machine learning algorithm, that let you “tune” your machine learning algorithm, like a car. Each type of model has its own hyper parameters.

Tuning is the LAST thing you do, after you finished building the model. It can add 3-6% accuracy on top of what you end up building.

We can use two pieces of python code to tell us what tuning is best:

1. **GridSearchCV**: tries all combination of recommended feature settings, tells you which one is best. Takes exponentially longer for each feature you are trying to optimize, in accordance with the amount of rows in your data
2. **RandomizedSearchCV**: tries X random “areas” in the combination of settings (matrix), and informs on which “areas” contain the best parameters. It informs us where to best conduct a grid search, to reduce the amount of time we need to GridSearch. If you have a big data set with a lot of features (and a crap PC), you can do a RandomizedSearchCV to reduce the amount of time the GridSearch is running

**Using GridSearchCV:**

1. Define a model (estimator)
2. Specify a dictionary containing the hyperparameters and their possible values
3. Choose a cross-validation strategy and evaluation metric
4. Fit the GridSearchCV object to your data

**Using RandomizedSearchCV:**

1. Define a model
2. Specify a dictionary containing the hyper parameters and their possible values
3. Specify the amount of iterations you want to run RandomizedSearchCV

|  |  |  |
| --- | --- | --- |
|  | Advantages | Disadvantages |
| GridSearchCV | Guaranteed to find the best hyperparameter combination | It can be computationally expensive (the larger the search space is) |
|  |  | It does not scale well with hyperparameter models |
| RandomizedSearchCV | More efficient than GridSearchCV, especially for large search spaces (or computationally expensive models) | It is not guaranteed to find the best combination in the specified search space, as it samples only a subset of the possible combinations |
|  | Allows specifying a number of iterations |  |

**K fold Cross Validation techniques:**

* When we do train test split it becomes 80%, 20% split.
* But what if by chance 20% of the data contains all the target positives?
* We can cut up the total data into “folds” and then train the model independently on the “folds” of data

HOWEVER!

* Runs into the same issue, where data could be siloed in a specific fold
  + splitting each fold 80-20 and evaluating the model is called “stratified” K fold cross validation
* Arguably more important than hypertuning and should be done before tuning, as your data split could be a big source of error