# 6 Must-Know Parameters for Machine Learning Algorithms

## Boost the performance of your models

Soner Yıldırım Mar 7 · 6 min read ★


Photo by Thomas Thompson on Unsplash

There are several machine learning algorithms for both supervised and unsupervised tasks. Thanks to the rich selection of open source libraries, we can implement these models with a few lines of code.

Although these algorithms are ready-to-use, we usually need to tune them through model parameters. The model parameters that we can customize

or adjust are known as hyperparameters. What tuning hyperparameters achieve is that it customizes an algorithm to fit our data or task better.

In order to make the most of an algorithm, the use of hyperparameters is of crucial importance. Thus, we need to have a comprehensive understanding of the model hyperparameters and their effects.

In this article, we will cover 6 critical hyperparameters of commonly used supervised machine learning algorithms. The article is based on Scikit-learn so if you are using a different library, the name of the parameters might be slightly different. However, what they mean and do are the same.

## 1. C parameter for Support Vector Machines

Support Vector Machine (SVM) is a widely-used supervised machine learning algorithm. It is mostly used in classification tasks but suitable for regression as well. SVM creates a decision boundary that separates different classes.

When determining the decision boundary, a SVM model tries to solve an optimization problem with the following goals:

- Increase the distance of decision boundary to classes (or support vectors)
- Maximize the number of points that are correctly classified in the training set

There is obviously a trade-off between these two goals. Decision boundary might have to be very close to a particular class to maximize the number of correctly labeled data points. The downside of this approach is having a model that is sensitive to small changes and noise in the independent variables. This situation is also known as overfitting.

On the other hand, a decision boundary might be placed as far apart as possible to each class with the expense of some misclassified exceptions.

The trade-off between these two approaches is controlled by the **c parameter.** It adds a penalty for each misclassified data point. If c is small, the penalty for misclassified points is low so a decision boundary with a

large margin is chosen at the expense of a greater number of misclassifications.

If c is large, SVM tries to minimize the number of misclassified examples due to high penalty which results in a decision boundary with a smaller margin. Penalty is not same for all misclassified examples. It is directly proportional to the distance to decision boundary.

## 2. Gamma parameter for Support Vector Machines

Before introducing the **gamma** parameter, we need to talk about kernel tricks. In some cases, data points that are not linearly separable are transformed using a kernel function so that they become linearly separable.

Kernel function is kind of a similarity measure. The inputs are original features and the output is a similarity measure in the new feature space. Similarity here means a degree of closeness.

One of the commonly used kernel functions is radial basis function (RBF). Gamma parameter of RBF controls the distance of influence for data points.

Low values of gamma indicates a large similarity radius which results in more points being grouped together. For high values of gamma, the points need to be very close to each other in order to be considered in the same group (or class). Therefore, models with very large gamma values tend to overfit. As the gamma decreases, the regions separating different classes get more generalized.

### 3. Max_depth for Decision Trees

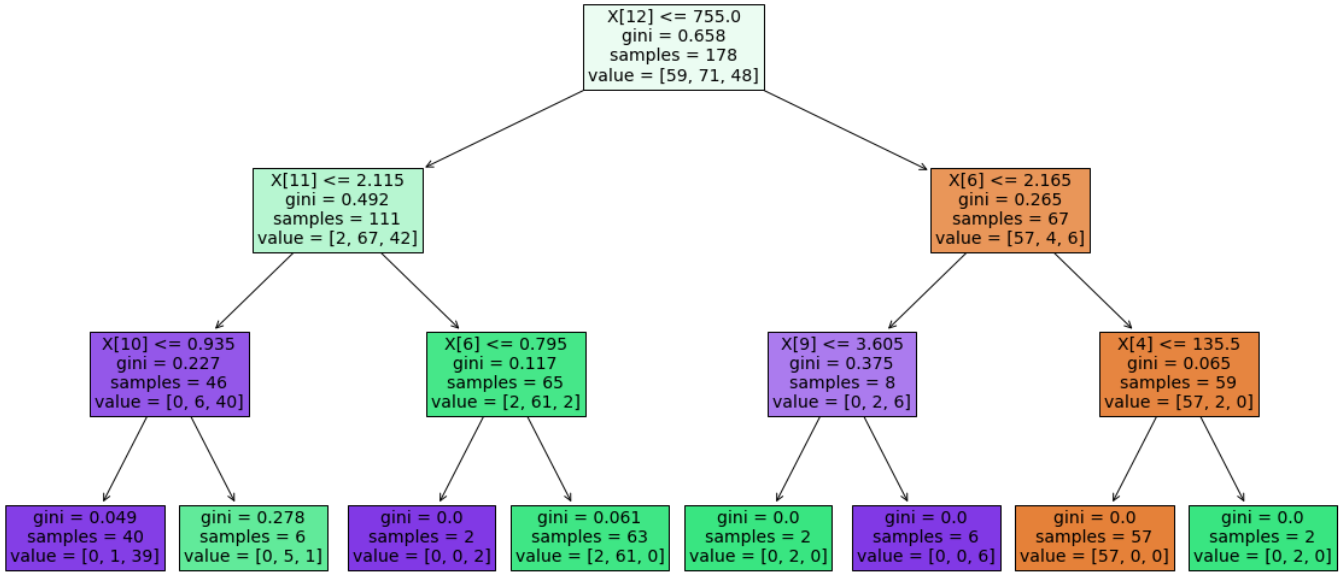
Decision tree is a widely-used supervised learning algorithm which is suitable for both classification and regression tasks. Decision trees serve as building blocks for some prominent ensemble learning algorithms such as random forests, GBDT, and XGBOOST.

A decision tree builds upon iteratively asking questions to split data points. The max_depth parameter is used to control the depth of a tree. The model stops splitting when max_depth is reached.

There is no standard or optimal value for the max_depth parameter. It highly depends on the data. In general, as the trees get deeper, the chance

of overfitting increases.

If we use a very small value for tree depth, the model may not be able to capture enough details or information about the dataset. Thus, it performs poorly on both training and test sets. This situation is also known as underfitting.



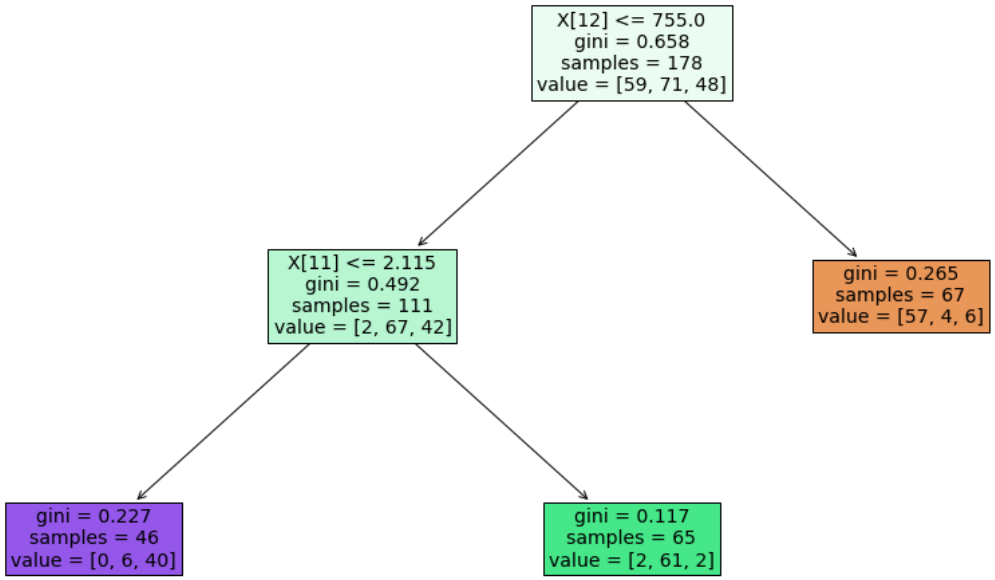

A decision tree with max_depth of 3 (image by author)

## 4. Min_impurity_decrease for Decision Trees

When the algorithm performs a split, the main goal is to decrease impurity as much as possible. If you'd like to learn more about impurity, information gain, and how decision trees use these concepts, here is a detailed article that I wrote a while back.

The more the impurity decreases, the more informative power that split gains. As the tree gets deeper, the amount of impurity decrease becomes lower. We can use this to prevent the tree from doing further splits. The hyperparameter for this task is **min_impurity_decrease**.

Scikit-learn provides two functions to measure the quality of a split which are gini for gini impurity and entropy for information gain. Consider the following visualization of a decision tree with a min_impurity_decrease value of 0.2. Each split achieves a decrease of 0.2 in the impurity.




(image by author)

## 5. Number of trees for Random Forest and GBDT

Random forests and gradient boosted decision trees (GBDT) are ensemble learning methods which means they combine many learners to build a more robust and accurate model. They are used to solve supervised

learning tasks. What random forests and GBDTs have in common is the base algorithm they use which is a decision tree.

The main difference between them is the way they combine decision trees. Random forests are built using a method called bagging in which each decision trees are used as parallel estimators.

GBDTs use boosting method to combine individual decision trees. Boosting means combining a learning algorithm in series to achieve a strong learner from many sequentially connected weak learners.

The n_estimators parameter is used to determine the number of trees used in the ensemble model. The effect of the n_estimators parameter is different on random forests and GBDT.

Increasing the number of trees in random forests does not cause overfitting. After some point, the accuracy of the model does not increase by adding more trees but it is also not negatively affected by adding excessive trees. You still do not want to add an unnecessary amount of trees due to computational reasons but there is no risk of overfitting associated with the number of trees in random forests.

However, the number of trees in gradient boosting decision trees is very critical in terms of overfitting. Adding too many trees will cause overfitting so it is important to stop adding trees at some point.

## 6. N_neighbors for K-Nearest Neighbors

K-nearest neighbors (kNN) is a supervised machine learning algorithm that can be used to solve both classification and regression tasks.

The kNN classifier determines the class of a data point by majority voting principle. If k is set to 5, the classes of 5 closest points are checked. Prediction is done according to the majority class. Similarly, kNN regression takes the mean value of the 5 closest points.

The n_neighbors parameter determines the k value. Just like the max_depth for decision trees, there is no standard or optimal k value for all problems.

If k is too small, the model becomes too specific. It also tends to be sensitive to noise and small changes in the features. The model accomplishes a high

accuracy on train set but will be a poor predictor on new, previously unseen data points. Therefore, we are likely to end up with an overfit model.

If we select a too large value for k, the model becomes too generalized and thus is not a good predictor on both train and test sets. It fails to capture the structure within the data. The model is likely to be underfit.

## Conclusion

We have covered 6 hyperparameters that are of crucial importance for the related algorithms. There are many more hyperparameters in the scope of machine learning algorithms but it would make this article very lengthy to cover them all.

In most cases, it requires many trials to find the optimal values for these hyperparameters. There are tools to make the process easier such as GridSearchCV function of Scikit-learn library.

Thank you for reading. Please let me know if you have any feedback.