Hyperparameter Tuning for Classification Models

Janani Ravi
CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Understanding hyperparameter tuning

Grid search for hyperparameter tuning

Tuning hyperparameters for decision trees and logistic regression models

Hyperparameters in Classification Models

Hyperparameters

Model configuration properties that define a model, and remain constant during the training of the model

Hyperparameters

Part of the model design

Understanding Hyperparameters

Model Inputs

Model Parameters

Model Hyperparameters

Understanding Hyperparameters

Model Inputs

Training data from which the model learns

Model Parameters

Model Hyperparameters

Understanding Hyperparameters

Model Inputs

Training data from which the model learns

Model Parameters

Model coefficient and intercept

Model Hyperparameters

Understanding Hyperparameters

Model Inputs

Training data from which the model learns

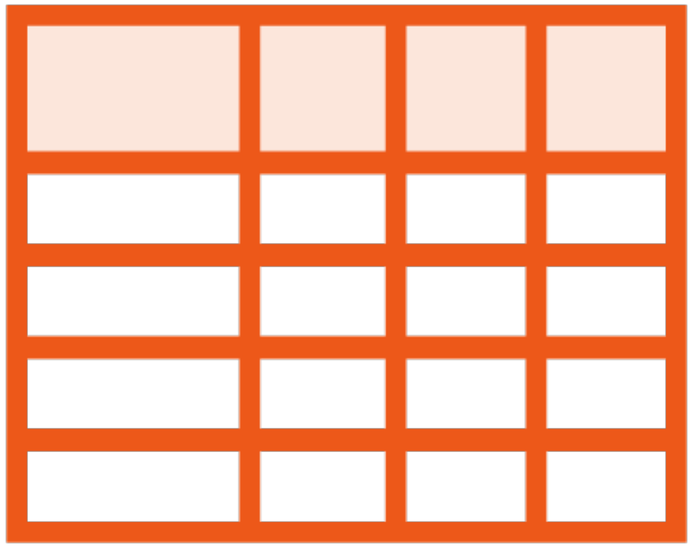
Model Parameters

Model coefficient and intercept

Model Hyperparameters

Depth of the decision tree, number of neighbors

Grid Search

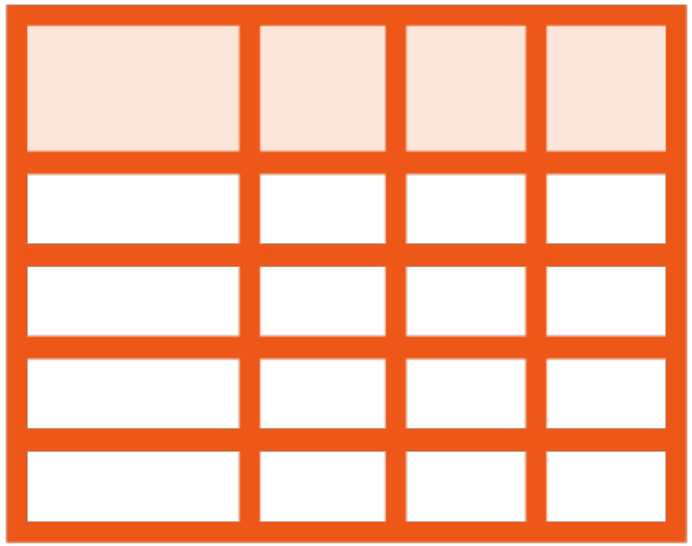


Specify possible values for each hyperparameter

These values form a grid where each cell is a candidate model

gridsearchcv in scikit-learn to evaluate each candidate model

Grid Search



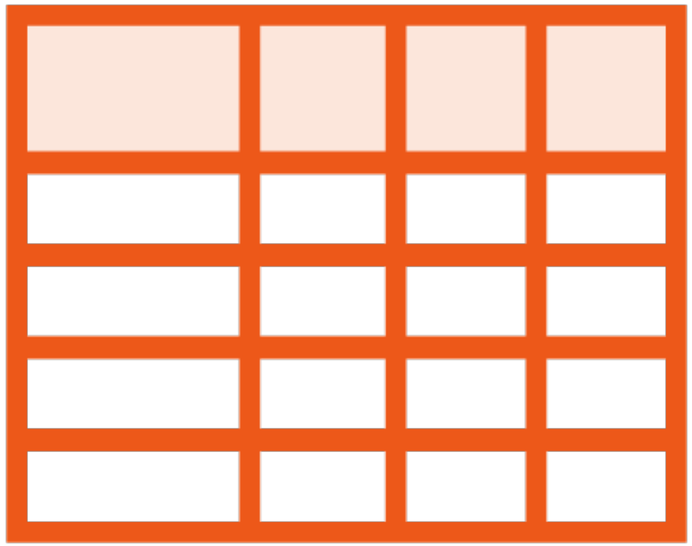
Scikit-learn ensures proper evaluation and cross-validation

Very computationally expensive

2 hyperparameters, 3 values each

9 models to train and evaluate

Grid Search



Cost and complexity of Grid Search can grow very quickly

Cloud-based evaluation can become very expensive

Also does not differentiate between important and trivial hyperparameters

Random search of hyperparameter space is an alternative

Hyperparameters in Decision Trees

Splitting strategy

Max depth

Min samples split

Min samples leaf

Min weight fraction

Max features

splitter : ***string, optional (default="best")***

The strategy used to choose the split at each node. Supported strategies are “best” to choose the best split and “random” to choose the best random split.

max_depth : ***int or None, optional (default=None)***

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

Titlecase

min_samples_split : ***int, float, optional (default=2)***

The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

Changed in version 0.18: Added float values for fractions.

min_samples_leaf : ***int, float, optional (default=1)***

The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least `min_samples_leaf` training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

- If int, then consider `min_samples_leaf` as the minimum number.
- If float, then `min_samples_leaf` is a fraction and `ceil(min_samples_leaf * n_samples)` are the minimum number of samples for each node.

Changed in version 0.18: Added float values for fractions.

min_weight_fraction_leaf : *float, optional (default=0.)*

The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.

max_features : *int, float, string or None, optional (default=None)*

The number of features to consider when looking for the best split:

- If int, then consider `max_features` features at each split.
- If float, then `max_features` is a fraction and `int(max_features * n_features)` features are considered at each split.
- If “auto”, then `max_features=sqrt(n_features)`.
- If “sqrt”, then `max_features=sqrt(n_features)`.
- If “log2”, then `max_features=log2(n_features)`.
- If None, then `max_features=n_features`.

Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than `max_features` features.

random_state : *int, RandomState instance or None, optional (default=None)*

If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by `np.random`.

max_leaf_nodes : *int or None, optional (default=None)*

Grow a tree with `max_leaf_nodes` in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

min_impurity_decrease : *float, optional (default=0.)*

A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

The weighted impurity decrease equation is the following:

```
N_t / N * (impurity - N_t_R / N_t * right_impurity
                    - N_t_L / N_t * left_impurity)
```

where `N` is the total number of samples, `N_t` is the number of samples at the current node, `N_t_L` is the number of samples in the left child, and `N_t_R` is the number of samples in the right child.

`N`, `N_t`, `N_t_R` and `N_t_L` all refer to the weighted sum, if `sample_weight` is passed.

New in version 0.19.

min_impurity_split : *float, (default=1e-7)*

Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf.

Deprecated since version 0.19: `min_impurity_split` has been deprecated in favor of `min_impurity_decrease` in 0.19. The default value of `min_impurity_split` will change from 1e-7 to 0 in 0.23 and it will be removed in 0.25. Use `min_impurity_decrease` instead.

Demo

Hyperparameter tuning of a Decision Tree classifier

Demo

Hyperparameter tuning a logistic regression classifier

Summary

Understanding hyperparameter tuning

Grid search for hyperparameter tuning

Tuning hyperparameters for decision trees and logistic regression models