Reference guide: Random forest tuning

Previously, you learned about random forest models and studied how to build and tune them. This reading is a quick-reference guide to help you when you're building models of your own. It includes:

- Import statements
- Hyperparameters

Import statements

The following are some of the most commonly used import statements for random forest models in sklearn.

Models

For classification tasks:

```
from sklearn.ensemble import RandomForestClassifier
```

For regression tasks:

```
from sklearn.ensemble import RandomForestRegressor
```

Evaluation metrics

For classification tasks:

```
from sklearn.metrics import
```

accuracy_score(y_true, y_pred, *[, ...])	Accuracy classification score.
average_precision_score(y_true, ...)	Compute average precision (AP) from prediction scores.
confusion_matrix(y_true, y_pred, *)	Compute confusion matrix to evaluate the performance of the training of a model .
f1_score(y_true, y_pred, *[, ...])	Compute the F1 score, also known as balanced F-score or F-measure.
fbeta_score(y_true, y_pred, *, beta)	Compute the F-beta score.
metrics.log_loss(y_true, y_pred, *[, eps, ...])	Log loss, aka logistic loss or cross-entropy loss.
multilabel_confusion_matrix(y_true, ...)	Compute a confusion matrix for each class or sample.
precision_recall_curve(y_true, ...)	Compute precision-recall pairs for different probability thresholds.
precision_score(y_true, y_pred, *[, ...])	Compute the precision.
recall_score(y_true, y_pred, *[, ...])	Compute the recall.
roc_auc_score(y_true, y_score, *[, ...])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

For regression tasks:
from sklearn.metrics import

mean_absolute_error(y_true, y_pred, *)	Mean absolute error regression loss.
mean_squared_error(y_true, y_pred, *)	Mean squared error regression loss.
mean_squared_log_error(y_true, y_pred, *)	Mean squared logarithmic error regression loss.
median_absolute_error(y_true, y_pred, *)	Median absolute error regression loss.
mean_absolute_percentage_error(...)	Mean absolute percentage error (MAPE) regression loss.
r2_score(y_true, y_pred, *[, ...])	R^2 (coefficient of determination) regression score function.

Hyperparameters
The following are some of the most important hyperparameters for random forest classification models in scikit-learn.

Hyperparameter	**What it does**	**Input type**	**Default Value**	**Considerations**
n_estimators	Specifies the number of trees your model will build in its ensemble	int	100	A typical range is 50–500. Consider how much data you have, how deep the trees are allowed to grow and how many samples are bootstrapped to grow each tree (you generally need more trees if they're shallow, and more trees if your bootstrap sample size is smaller). Also consider if your use case has latency requirements.

max_depth	Specifies how many levels your tree can have. If None, trees grow until leaves are pure or until all leaves contain less than min_samples_split samples.	int	None	Random forest models often use base learners that are fully grown, but restricting tree depth can reduce train/latency times and prevent overfitting. If not None, consider values 3–8.
min_samples_split	Controls threshold below which nodes become leaves If float, then it represents a percentage (0–1] of max_samples.	int or float	2	Consider (a) how many samples are in your dataset, and (b) how much of that data you're allowing each base learner to use (i.e., the value of the max_samples hyperparameter). The fewer samples available, the lesser the number of samples may need to be allowed in each leaf node (otherwise the tree would be very shallow).

min_samples_leaf	A split can only occur if it guarantees a minimum of this number of observations in each resulting node. If float, then it represents a percentage (0–1] of max_samples.	int or float	1	Consider (a) how many samples are in your dataset, and (b) how much of that data you're allowing each base learner to use (i.e., the value of the max_samples hyperparameter). The fewer samples available, the lesser the number of samples may need to be allowed in each leaf node (otherwise the tree would be very shallow).

max_features	Specifies the number of features that each tree randomly selects during training If int, then consider max_features features at each split. If float, then max_features is a fraction and round(max_features * n_features) features are considered at each split. If “sqrt”, then max_features=sqrt(n_features). If “log2”, then max_features=log2(n_features). If None, then max_features=n_features.	{“sqrt”, “log2”, None}, int or float,	“sqrt”	Consider how many features the dataset has and how many trees will be grown. Fewer features sampled during each bootstrap means more base learners would be needed. Small max_features values on datasets with many features mean more unpredictive trees in the ensemble.

max_samples*	Specifies the number of samples bootstrapped from the dataset to train each base model If float, then it represents a percentage (0–1] of the dataset. If None, then draw X.shape[0] samples.	int or float	None	* Consider the size of your dataset. When working with large datasets, it can be beneficial to limit the number of samples in each tree, because doing so can greatly reduce training time and yet still result in a robust model. For example, 20% of 1 billion may be enough to capture patterns in the data, but if you have 1,000 samples then you'll probably need to use them all.

* Note that max_samples was not used in the tether video, but is included here so you can use this hyperparameter in your own work. Remember that using fractions of the data to train each base learner can possibly improve model predictions and certainly speed up execution times.

Key takeaways

When building machine learning models, it's essential to have the right tools and to understand how to use them. While there are numerous other hyperparameters to explore, the ones in this reference guide are among the most important. Be inquisitive and try different approaches. Discovering ways to improve your model is a lot of fun!

Resources for more information

More detailed information about random forest tuning can be found here:

- scikit-learn documentation:
    - Model metrics
    - Random forest classifier: documentation for model used for classification tasks
    - Random forest regressor: documentation for model used for regression tasks