

**Current**

Algorithms CHANGE ALGORITHM PRESETS ▾

- Lasso Regression  OFF
- LightGBM  OFF
- XGBoost  OFF
- Decision Tree  OFF
- Support Vector Machine  OFF
- Stochastic Gradient Descent  OFF
- KNN  OFF
- Extra Random Trees  OFF
- Neural Network  OFF
- Lasso Path  OFF
- LightGBM Regression  OFF
- Custom Python model  ON

+ ADD CUSTOM PYTHON MODEL

**Suggested**

Algorithms CHANGE ALGORITHM PRESETS ▾

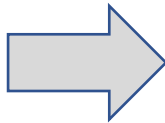
- Lasso Regression  OFF
- LightGBM  OFF
- XGBoost  OFF
- Decision Tree  OFF
- Support Vector Machine  OFF
- Stochastic Gradient Descent  OFF
- KNN  OFF
- Extra Random Trees  OFF
- Neural Network  OFF
- Lasso Path  OFF
- LightGBM Regression  OFF
- Custom Python model  OFF

+ ADD CUSTOM PYTHON MODEL

Auto-generated Recipe  ON

+ ADD AUTO-GENERATED RECIPE

User selects



**Auto-generated Recipe**  ON

For an Auto-generated Recipe, you indicate the package and algorithm. Packages must be loaded into the correct environment (e.g., `decision trees`).

[Show more...](#)

base\_estimator

n\_estimators

learning\_rate

loss

random\_state

DSS auto-populates fields and defaults



**sklearn.ensemble.AdaBoostRegressor**

```
class sklearn.ensemble.AdaBoostRegressor(base_estimator=None, *, n_estimators=50, learning_rate=1.0, loss='linear', random_state=None) [source]
```

**Parameters::**

- base\_estimator** : *object, default=None*  
The base estimator from which the boosted ensemble is built. If `None`, then the base estimator is `DecisionTreeRegressor` initialized with `max_depth=3`.
- n\_estimators** : *int, default=50*  
The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early. Values must be in the range `[1, inf)`.
- learning\_rate** : *float, default=1.0*  
Weight applied to each regressor at each boosting iteration. A higher learning rate increases the contribution of each regressor. There is a trade-off between the `learning_rate` and `n_estimators` parameters. Values must be in the range `(0.0, inf)`.
- loss** : *{'linear', 'square', 'exponential'}, default='linear'*  
The loss function to use when updating the weights after each boosting iteration.
- random\_state** : *int, RandomState instance or None, default=None*  
Controls the random seed given at each `base_estimator` at each boosting iteration. Thus, it is only used when `base_estimator` exposes a `random_state`. In addition, it controls the bootstrap of the weights used to train the `base_estimator` at each boosting iteration. Pass an `int` for reproducible output across multiple function calls. See Glossary.