

# QuantileCV

**mod** quantile\_cv

Definition for RegressionCV.

**class** QuantileCV

Bases: [RegressionCV](#)

Defines an auto quantile tree, based on the bayesian optimization base class.

Source code in `src/tree_machine/quantile_cv.py`

```
16 class QuantileCV(RegressionCV):
17     """
18     Defines an auto quantile tree, based on the bayesian optimization base
19     class.
20     """
21
22     @validate_call(config={"arbitrary_types_allowed": True})
23     def __init__(
24         self,
25         alpha: NonNegativeFloat,
26         cv: BaseCrossValidator,
27         n_trials: NonNegativeInt,
28         timeout: NonNegativeInt,
29         config: RegressionCVConfig,
30         backend: str = "xgboost",
31     ) -> None:
32         """
33         Constructor for QuantileCV.
34
35         Args:
36             alpha: The quantile to estimate, which must be between 0 and 1.
37             cv: Splitter object to use when estimating the model.
38             n_trials: Number of optimization trials to use when finding a
39             model.
40             timeout: Timeout in seconds to stop the optimization.
41             config: Configuration to use when fitting the model.
42             backend: Backend to use for the model. Either "xgboost" or
43             "catboost".
44         """
45         super().__init__("quantile", cv, n_trials, timeout, config,
46             backend=backend)
47         self.alpha_ = alpha
48
49     @property
50     def scorer(self) -> tp.Callable[..., float]:
51         """
52         Returns correct scorer to use when scoring with QuantileCV.
53         """
54         # For quantile regression, we always use the quantile metric with
55         alpha parameter
56         return make_scorer(
57             update_wrapper(
58                 partial(
59                     regression_metrics["quantile"],
60                     alpha=self.alpha_,
61                 ),
62                 regression_metrics["quantile"],
63             ),
64             greater_is_better=False,
65         )
```

**attr** **scorer** property

scorer

Returns correct scorer to use when scoring with QuantileCV.

**meth** **\_\_init\_\_**

```
__init__(alpha, cv, n_trials, timeout, config, backend='xgboost')
```

Constructor for QuantileCV.

**Parameters:**

Name	Type	Description	Default
alpha	NonNegativeFloat	The quantile to estimate, which must be between 0 and 1.	<i>required</i>
cv	BaseCrossValidator	Splitter object to use when estimating the model.	<i>required</i>
n_trials	NonNegativeInt	Number of optimization trials to use when finding a model.	<i>required</i>
timeout	NonNegativeInt	Timeout in seconds to stop the optimization.	<i>required</i>
config	RegressionCVConfig	Configuration to use when fitting the model.	<i>required</i>
backend	str	Backend to use for the model. Either "xgboost" or "catboost".	'xgboost'

Source code in `src/tree_machine/quantile_cv.py`

```
21 @validate_call(config={"arbitrary_types_allowed": True})
22 def __init__(
23     self,
24     alpha: NonNegativeFloat,
25     cv: BaseCrossValidator,
26     n_trials: NonNegativeInt,
27     timeout: NonNegativeInt,
28     config: RegressionCVConfig,
29     backend: str = "xgboost",
30 ) -> None:
31     """
32     Constructor for QuantileCV.
33
34     Args:
35         alpha: The quantile to estimate, which must be between 0 and 1.
36         cv: Splitter object to use when estimating the model.
37         n_trials: Number of optimization trials to use when finding a model.
38         timeout: Timeout in seconds to stop the optimization.
39         config: Configuration to use when fitting the model.
40         backend: Backend to use for the model. Either "xgboost" or "catboost".
41     """
42     super().__init__("quantile", cv, n_trials, timeout, config,
43                     backend=backend)
44     self.alpha_ = alpha
```