

# 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     ) -> 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
38             model.
39             timeout: Timeout in seconds to stop the optimization.
40             config: Configuration to use when fitting the model.
41         """
42         super().__init__("quantile", cv, n_trials, timeout, config)
43         self.alpha_ = alpha
44
45     @property
46     def scorer(self) -> tp.Callable[... , float]:
47         """
48         Returns correct scorer to use when scoring with QuantileCV.
49         """
50         # For quantile regression, we always use the quantile metric with
51         alpha parameter
52         return make_scorer(
53             update_wrapper(
54                 partial(
55                     regression_metrics["quantile"],
56                     alpha=self.alpha_,
57                 ),
58                 regression_metrics["quantile"],
59             ),
60             greater_is_better=False,
61         )
```

**attr** **scorer** property

```
scorer
```

Returns correct scorer to use when scoring with QuantileCV.

**meth** `__init__`

```
__init__(alpha, cv, n_trials, timeout, config)
```

Constructor for QuantileCV.

**Parameters:**

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

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 ) -> None:
30     """
31     Constructor for QuantileCV.
32
33     Args:
34         alpha: The quantile to estimate, which must be between 0 and 1.
35         cv: Splitter object to use when estimating the model.
36         n_trials: Number of optimization trials to use when finding a model.
37         timeout: Timeout in seconds to stop the optimization.
38         config: Configuration to use when fitting the model.
39     """
40     super().__init__("quantile", cv, n_trials, timeout, config)
41     self.alpha_ = alpha
```