

# 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 class.
19     """
20
21     @validate_call(config={"arbitrary_types_allowed": True})
22     def __init__(self,
23                  alpha: NonNegativeFloat,
24                  cv: BaseCrossValidator,
25                  n_trials: NonNegativeInt,
26                  timeout: NonNegativeInt,
27                  config: RegressionCVConfig,
28                  backend: str = "xgboost",
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             backend: Backend to use for the model. Either "xgboost" or "catboost".
40
41         """
42         super().__init__("quantile", cv, n_trials, timeout, config,
43                         backend=backend)
44         self.alpha_ = alpha
45
46     @property
47     def scorer(self) -> tp.Callable[..., float]:
48         """
49         Returns correct scorer to use when scoring with QuantileCV.
50         """
51
52         # For quantile regression, we always use the quantile metric with alpha
53         # parameter
54         return make_scorer(
55             update_wrapper(
56                 partial(
57                     regression_metrics["quantile"],
58                     alpha=self.alpha_,
59                 ),
60                 regression_metrics["quantile"],
61             ),
62             greater_is_better=False,
63         )
```

**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, backend=backend)
43     self.alpha_ = alpha
```