

# Peak-Engines Documentation

## Contents

<b>Module peak_engines</b>	<b>1</b>
Sub-modules	1
Classes	1
Class WarpedLinearRegressionModel	1
Parameters	1
Examples	1
Instance variables	1
Methods	2
Class Warper	2
Methods	2
<b>Module peak_engines.peak_engines_impl</b>	<b>3</b>
Functions	3
Function WarpedLinearRegressionModel	3

## Module peak\_engines

### Sub-modules

- [peak\\_engines.peak\\_engines\\_impl](#)

### Classes

#### Class WarpedLinearRegressionModel

```
class WarpedLinearRegressionModel(init0=None, fit_intercept=True, normalize=True,
    num_steps=1, tolerance=0.0001)
```

Warped linear regression model fit so as to maximize likelihood.

#### Parameters

**init0 : object, default=None** Functor that can be used to change the starting parameters of the optimizer.  
**fit\_intercept : bool, default=True** Whether to center the target values and feature matrix columns.  
**normalize : bool, default=True** Whether to rescale the target vector and feature matrix columns.  
**num\_steps : int, default=1** The number of components to use in the warping function. More components allows for the model to fit more complex warping functions but increases the chance of overfitting.  
**tolerance : float, default=0.0001** The tolerance for the optimizer to use when deciding to stop the objective. With a lower value, the optimizer will be more stringent when deciding whether to stop searching.

### Examples

#### Instance variables

#### Variable noise\_stddev

Return the fitted noise standard deviation.

**Variable noise\_variance**

Return the fitted noise variance.

**Variable regressors**

Return the regressors of the latent linear regression model.

**Variable warper**

Return the warper associated with the model.

**Variable within\_tolerance**

Return True if the optimizer found parameters within the provided tolerance.

**Methods****Method fit**

```
def fit(self, X, y)
```

Fit the warped linear regression model.

**Method get\_params**

```
def get_params(self, deep=True)
```

Get parameters for this estimator.

**Method predict**

```
def predict(self, X_test)
```

Predict target values.

**Method predict\_latent\_with\_stddev**

```
def predict_latent_with_stddev(self, X_test)
```

Predict latent values along with the standard deviation of the error distribution.

**Method predict\_logpdf**

```
def predict_logpdf(self, X_test)
```

Predict target values with a functor that returns the log-likelihood of given target values under the model's error distribution.

**Method set\_params**

```
def set_params(self, **parameters)
```

Set parameters for this estimator.

**Class Warper**

```
class Warper(impl)
```

Warping functor for a dataset's target space.

**Methods**

**Method** `compute_latent`

```
def compute_latent(self, y)
```

Compute the warped latent values for a given target vector.

**Method** `compute_latent_with_derivative`

```
def compute_latent_with_derivative(self, y)
```

Compute the warped latent values and derivatives for a given target vector.

**Method** `invert`

```
def invert(self, z)
```

Invert the warping transformation.

**Module** `peak_engines.peak_engines_impl`

Ridge Regression Module

**Functions****Function** `WarpedLinearRegressionModel`

```
def WarpedLinearRegressionModel(...)
```

Constructs a warped linear regression model

---

Generated by *pdoc* 0.8.1 (<https://pdoc3.github.io>).