

▷ Source

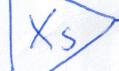
code

○ Node :

□ Machine

upstream → downstream

$X_s = \text{source}(X)$



$Y_s = \text{Source}(y)$



```
julia> W  
Node{Machine{Standardizer,...}}  
args:  
1: Source @627  
formula:  
transform  
Machine{Standardizer,...},  
Source @627)
```

```
julia> z  
Node{Machine{UnivariateBoxCoxTransformer,...}}  
args:  
1: Source @198  
formula:  
transform  
Machine{UnivariateBoxCoxTransformer,...},  
Source @198)
```

Node: "A function" that uses trained machine

Reference:
<https://juliaai.github.io/DataScienceTutorials.jl/getting-started/learning-networks/>
<https://juliaai.github.io/DataScienceTutorials.jl/getting-started/learning-networks-2/>

fit!(W, rows=train)

W.args = Xs-like input

W = transform(stand, Xs)

Machine{Standardizer,...}
stand = machine(Standardizer(), Xs)

z.args = W-like input

$\hat{z} = \text{predict}(\text{ridge}, W)$

Machine{RidgeRegressor,...}

ridge = machine(Ridge(lambda=0.1), W, z)

Node{Machine{}}

z = transform(box, ys)

Machine{UnivariateBoxCoxTransformer,...}
box = machine(, ys)

$\hat{y} = \text{inverse_transform}(box, \hat{z})$

```
julia> box  
Machine{UnivariateBoxCoxTransformer,...} trained 1 time; caches data  
model: UnivariateBoxCoxTransformer  
args:  
1: Source @198 ↗ `AbstractVector{Continuous}`
```

```
julia> ridge  
Machine{RidgeRegressor,...} trained 1 time; caches data  
model: MLJMultivariateStatsInterface.RidgeRegressor  
args:  
1: Node{Machine{Standardizer,...}}  
2: Node{Machine{UnivariateBoxCoxTransformer,...}}
```

→ rms(
 $y[\text{test}]$,
 $\hat{y}[\text{rows}=\text{test}]$)

```
julia> z  
Node{Machine{RidgeRegressor,...}}  
args:  
1: Node{Machine{Standardizer,...}}  
formula:  
predict  
Machine{RidgeRegressor,...},  
transform  
Machine{Standardizer,...},  
Source @627))
```

```
julia> z.machine  
Machine{RidgeRegressor,...} trained 0 times; caches data  
model: MLJMultivariateStatsInterface.RidgeRegressor  
args:  
1: Node{Machine{Standardizer,...}}  
2: Node{Machine{UnivariateBoxCoxTransformer,...}}
```

```
julia> y  
Node{Machine{UnivariateBoxCoxTransformer,...}}  
args:  
1: Node{Machine{RidgeRegressor,...}}  
formula:  
inverse_transform  
Machine{UnivariateBoxCoxTransformer,...},  
predict  
Machine{RidgeRegressor,...},  
transform  
Machine{Standardizer,...},  
Source @627)))
```