Regularization

METIS

OLS Regression Results

Dep. Variable:	DomesticTotalGross	**R-squared:**	0.286
Model:	OLS	**Adj. R-squared:**	0.278
Method:	Least Squares	**F-statistic:**	34.82
Date:	Sun, 14 Sep 2014	**Prob (F-statistic):**	6.80e-08
Time:	21:59:46	**Log-Likelihood:**	-1738.1
No. Observations:	89	**AIC:**	3480.
Df Residuals:	87	**BIC:**	3485.
Df Model:	1		

	coef	**std err**	**t**	**P>|t|**	**[95.0% Conf. Int.]**
Budget	0.7846	0.133	5.901	0.000	0.520 1.049
Ones	4.44e+07	1.27e+07	3.504	0.001	1.92e+07 6.96e+07

Omnibus:	39.749	**Durbin-Watson:**	0.674
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	99.441
Skew:	1.587	**Prob(JB):**	2.55e-22
Kurtosis:	7.091	**Cond. No.**	1.54e+08

$$AIC = 2k - 2\ln(L)$$

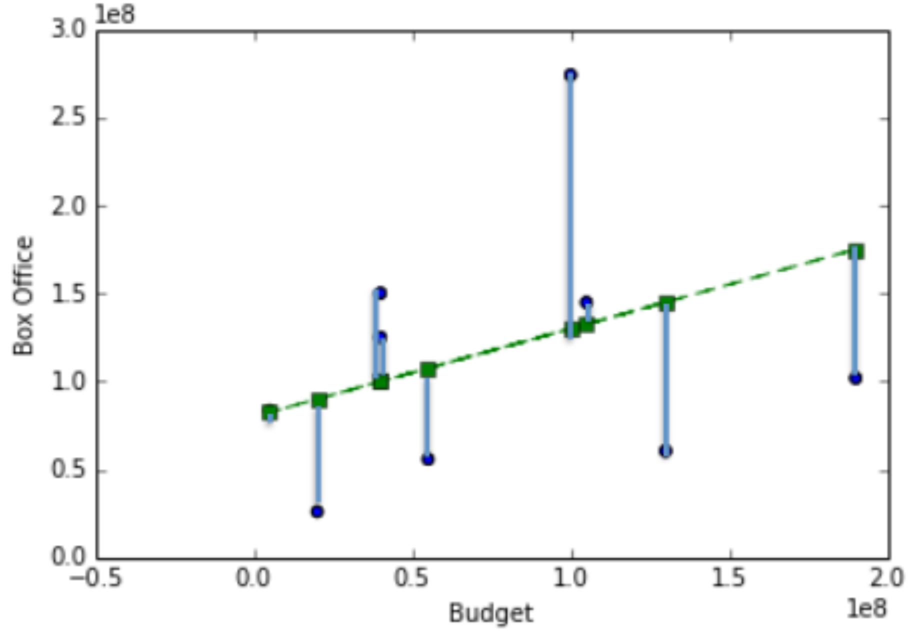
parameters

Log likelihood

While awarding goodness of fit, penalize model complexity

While awarding goodness of fit, penalize model complexity

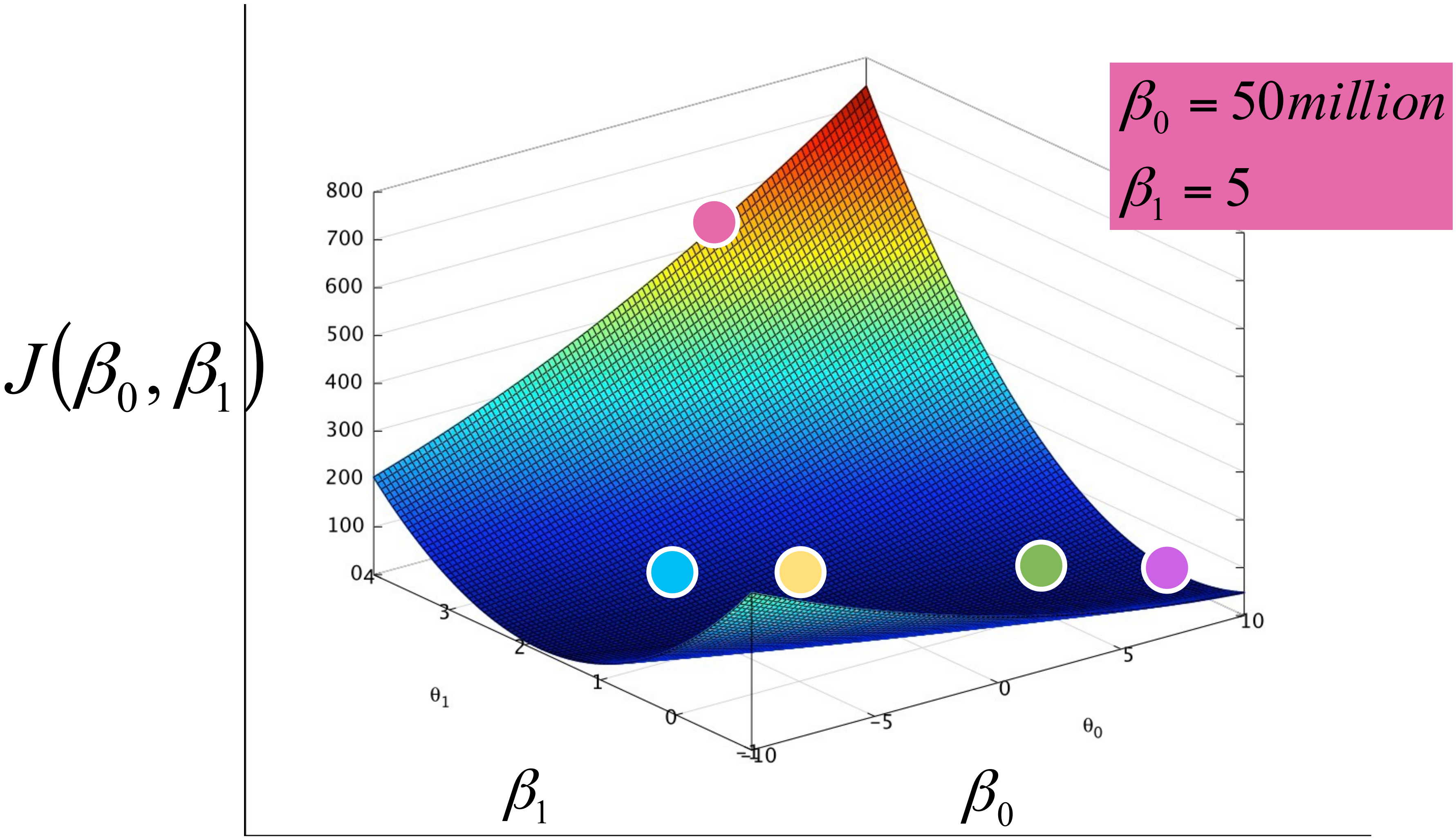
Why not do that while we are fitting?



Cost function

Takes a model (specific parameter values), returns a score

$$J(\beta_0, \beta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left((\beta_0 + \beta_1 x_{obs}^{(i)}) - y_{obs}^{(i)} \right)^2$$



$\beta_0 = 80million$
$\beta_1 = 0.5$

$\beta_0 = 0$
$\beta_1 = 1.5$

$\beta_0 = 120million$
$\beta_1 = 0.1$

$\beta_0 = 30million$
$\beta_1 = 2$

Cost function

$$J(\beta_0, \beta_1) = \frac{1}{2m} \underbrace{\sum_{i=1}^{m} \left((\beta_0 + \beta_1 x_{obs}^{(i)}) - y_{obs}^{(i)} \right)^2}_{\text{Lower for better fits}}$$

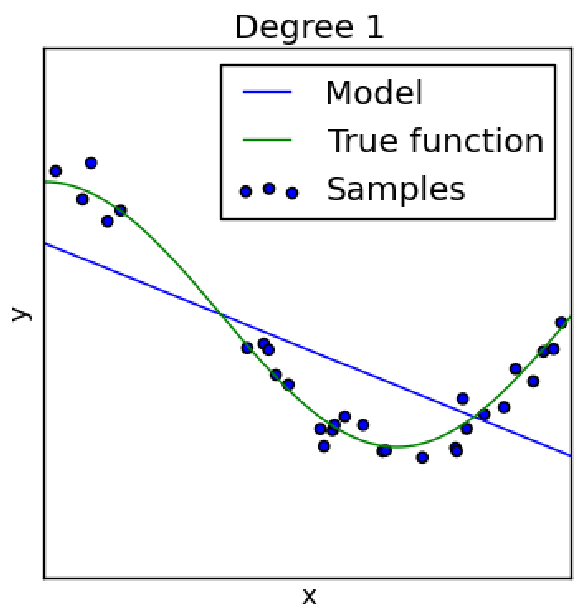
Cost function

Add a penalty for the size of each parameter!

$$J(\beta_0, \beta_1) = \underbrace{\frac{1}{2m}\sum_{i=1}^{m}\left(y_\beta(x_{obs}^{(i)}) - y_{obs}^{(i)}\right)^2}_{\substack{\text{Low: good fit} \\ \text{High: bad fit}}} + \underbrace{\lambda\sum_{j=1}^{k}\beta_j^2}_{\substack{\text{Low: simple model} \\ \text{High: complex model}}}$$

Diagnostics to detect under/overfitting

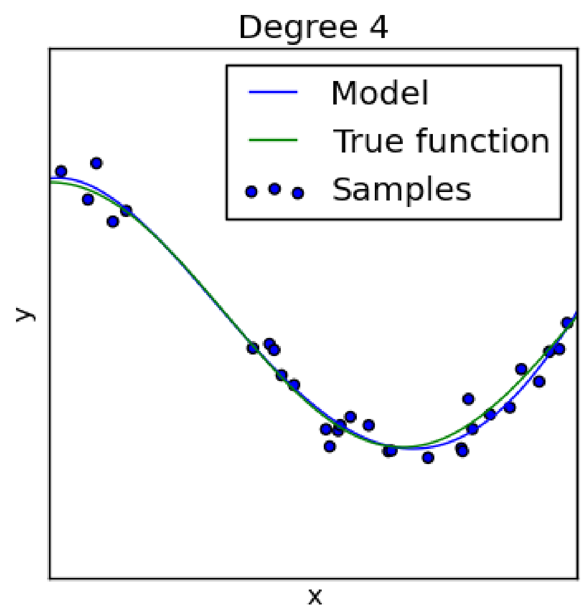
Underfitting



$$J(\beta_0,\beta_1)=\frac{1}{2m}\sum_{i=1}^{m}\left(y_\beta(x_{obs}^{(i)})-y_{obs}^{(i)}\right)^2+\lambda\sum_{j=1}^{k}\beta_j^2$$

J = V. High + Low

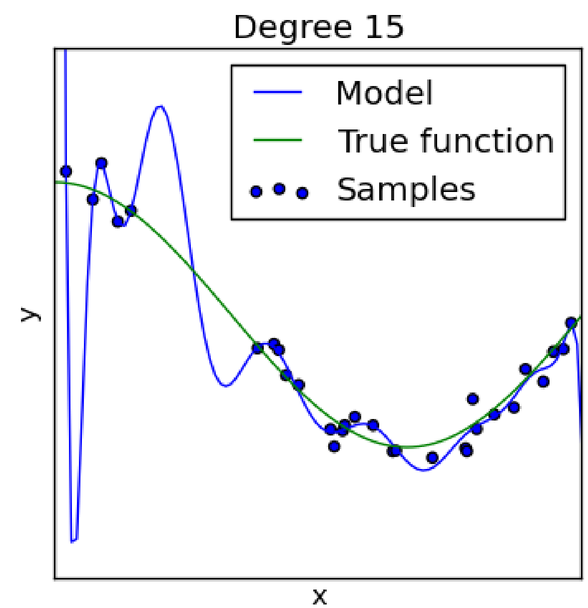
Just Right



$$J(\beta_0,\beta_1)=\frac{1}{2m}\sum_{i=1}^{m}\left(y_\beta(x_{obs}^{(i)})-y_{obs}^{(i)}\right)^2+\lambda\sum_{j=1}^{k}\beta_j^2$$

J = Low + Low

Overfitting



$$J(\beta_0,\beta_1)=\frac{1}{2m}\sum_{i=1}^{m}\left(y_\beta(x_{obs}^{(i)})-y_{obs}^{(i)}\right)^2+\lambda\sum_{j=1}^{k}\beta_j^2$$

J = Low + V. High

Ridge Regression

$$J\left(\beta_0, \beta_1\right) = \frac{1}{2m} \sum_{i=1}^{m} \left(y_\beta (x_{obs}^{(i)}) - y_{obs}^{(i)} \right)^2 + \lambda \sum_{j=1}^{k} \beta_j^2$$

Underfitting

J = V. High + Low

Just Right

J = Low + Low

Overfitting

J = Low + V. High

$\lambda = 1$

$$J(\beta_0, \beta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(y_\beta(x_{obs}^{(i)}) - y_{obs}^{(i)} \right)^2 + \lambda \sum_{j=1}^{k} \beta_j^2$$

$$y_\beta(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \varepsilon$$

Underfitting

J = V. High + Low

Just Right

J = Low + Low

Overfitting

J = Low + V. High

$$\lambda = 1$$

$$J(\beta_0, \beta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(y_\beta(x_{obs}^{(i)}) - y_{obs}^{(i)} \right)^2 + \lambda \sum_{j=1}^{k} \beta_j^2$$

$$\approx 0 \qquad \approx 0$$

$$y_\beta(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \varepsilon$$

Underfitting

J = V. High + Medium

Just Right

J = Low + V High

Overfitting

J = Low + VVVHigh

VERY LARGE

underfit

$$J(\beta_0, \beta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(y_\beta(x_{obs}^{(i)}) - y_{obs}^{(i)} \right)^2 + \lambda \sum_{j=1}^{k} \beta_j^2$$

≈0 ≈0 ≈0 ≈0

$$y_\beta(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \varepsilon$$

Underfitting

J = V. High + Tiny

Just Right

J = Low + Tiny

Overfitting

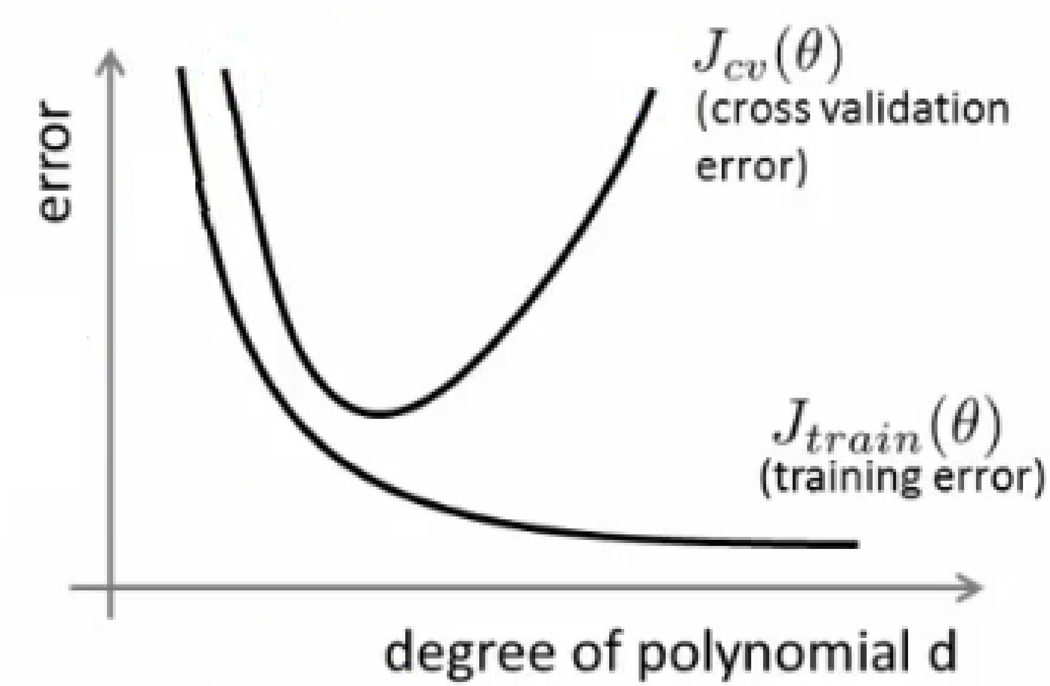
J = Low + Tiny

very small possible overfit

$$J(\beta_0, \beta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(y_\beta(x_{obs}^{(i)}) - y_{obs}^{(i)} \right)^2 + \lambda \sum_{j=1}^{k} \beta_j^2$$

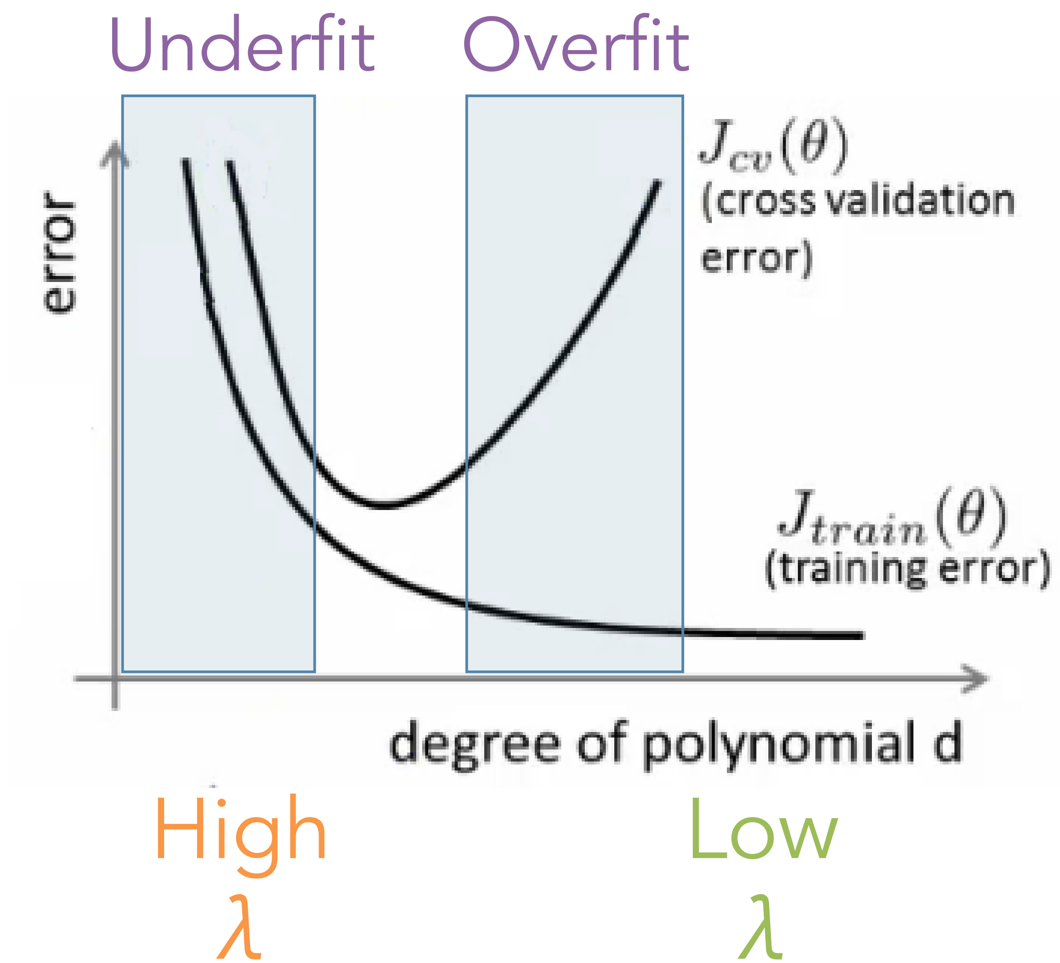
$$y_\beta(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \varepsilon$$

Error vs. regularization λ



High λ

Low λ


Underfit
Overfit
error
$J_{cv}(\theta)$
(cross validation error)
$J_{train}(\theta)$
(training error)
degree of polynomial d
High
λ
Low
λ

Ridge Regularization (L2)

$$J\left(\beta_0,\beta_1\right) = \frac{1}{2m}\sum_{i=1}^{m}\left(y_\beta(x_{obs}^{(i)}) - y_{obs}^{(i)}\right)^2 + \lambda\sum_{j=1}^{k}\beta_j^2$$

Ridge Regularization (L2)

$$J\left(\beta_0,\beta_1\right)=\frac{1}{2m}\sum_{i=1}^{m}\left(y_\beta(x_{obs}^{(i)})-y_{obs}^{(i)}\right)^2+\lambda\sum_{j=1}^{k}\beta_j^2$$

Lasso Regularization (L1)

$$J\left(\beta_0,\beta_1\right)=\frac{1}{2m}\sum_{i=1}^{m}\left(y_\beta(x_{obs}^{(i)})-y_{obs}^{(i)}\right)^2+\lambda\sum_{j=1}^{k}\left|\beta_j\right|$$

Ridge Regularization (L2)

$$J(\beta_0,\beta_1)=\frac{1}{2m}\sum_{i=1}^{m}\left(y_\beta(x_{obs}^{(i)})-y_{obs}^{(i)}\right)^2+\lambda\sum_{j=1}^{k}\beta_j^2$$

Lasso Regularization (L1)

$$J(\beta_0,\beta_1)=\frac{1}{2m}\sum_{i=1}^{m}\left(y_\beta(x_{obs}^{(i)})-y_{obs}^{(i)}\right)^2+\lambda\sum_{j=1}^{k}\left|\beta_j\right|$$

Elastic Net (L1 + L2)

$$J(\beta_0,\beta_1)=\frac{1}{2m}\sum_{i=1}^{m}\left(y_\beta(x_{obs}^{(i)})-y_{obs}^{(i)}\right)^2+\lambda_1\sum_{j=1}^{k}\left|\beta_j\right|+\lambda_2\sum_{j=1}^{k}\beta_j^2$$

How can I use this?

We were doing:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X,Y)
```

How can I use this?

We were doing:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X,Y)
```

To use Lasso Regularization:

```
from sklearn.linear_model import Lasso
model = Lasso(1.0)
model.fit(X,Y)
```

λ (sklearn Calls It alpha)

How can I use this?

We were doing:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X,Y)
```

To use Ridge Regularization:

```
from sklearn.linear_model import Ridge
model = Ridge(1.0)
model.fit(X,Y)
```

λ (sklearn Calls It alpha)

How can I use this?

We were doing:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X,Y)
```

To use Elastic Net:

```
from sklearn.linear_model import ElasticNet
model = ElasticNet(1.0, l1_ratio = 0.5)
model.fit(X,Y)
```

total weight for the full penalty term

ratio of l1/l2 penalty

How can I use this?

We were doing:

```
import statsmodels.formula.api as sm
model = sm.OLS.fit(Y, X)
```

How can I use this?

We were doing:

```
import statsmodels.formula.api as sm
model = sm.OLS.fit(Y, X)
```

To use Lasso Regularization:

```
import statsmodels.formula.api as sm
model = sm.OLS.fit_regularized(Y, X, alpha=1.0)
```

How can I use this?

We were doing:

```
import statsmodels.formula.api as sm
model = sm.OLS.fit(Y, X)
```

To use Ridge Regularization:

```
import statsmodels.formula.api as sm
model = sm.OLS.fit_regularized(Y, X,
                               alpha=1.0,
                               L1_wt = 0)
```

How can I use this?

We were doing:

```
import statsmodels.formula.api as sm
model = sm.OLS.fit(Y, X)
```

To use Elastic Net:

```
import statsmodels.formula.api as sm
model = sm.OLS.fit_regularized(Y, X,
                               alpha=1.0,
                               L1_wt = 0.5)
```

My model is not
awesome
enough.

What do I do?

Try these and check test error again:

Use a smaller set of features
Try adding polynomials
Check functional forms for each feature
Try including other features
Use more data (bigger training set)
Regularization

Try these and check test error again:

Use a smaller set of features
Try adding polynomials
Check functional forms for each feature
Try including other features
Use more data (bigger training set)
Regularization: Increase/decrease λ