Lab #3: Linear Regression and Model Selection

CS 109A, STAT 121A, AC 209A: Data Science

Fall 2016

Harvard

Today's lab: Problem 1

a) Multiple linear regression from scratch
   - Fit regression model
   - Score regression model

b) Confidence intervals on model parameters
   - Analyze histograms of model parameters
   - Compute confidence intervals

Today’s lab: Problem 1

a) Multiple linear regression from scratch
   - Fit regression model
   - Score regression model

b) Confidence intervals on model parameters
   - Analyze histograms of model parameters
   - Compute confidence intervals

Review:
numpy basics

Data and models as matrices

X_{1}`	…	X_d	Y
X_{11}	…	X_{1d}	Y_1
	⋮		⋮
X_{n1}	…	X_{nd}	Y_n

$$\longrightarrow \quad \begin{array}{cc} X & Y \\ \begin{bmatrix} X_{11} & \ldots & X_{1d} \\ \vdots & \ddots & \vdots \\ X_{n1} & \ldots & X_{nd} \end{bmatrix} & \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} \end{array}$$

- *X* is two dimensional array with shape (n, d)
- *Y* is two dimensional array with shape (n, 1)

Data and models as matrices

- Model: Predictions as matrix multiplication

$$\begin{bmatrix} \widehat{Y}_1 \\ \vdots \\ \widehat{Y}_n \end{bmatrix} = \begin{bmatrix} w_1 X_{11} + \ldots + w_d X_{1d} + c \\ \vdots \\ w_1 X_{n1} + \ldots + w_d X_{nd} + c \end{bmatrix}$$

Data and models as matrices

- Model: Predictions as matrix multiplication

$$\begin{bmatrix} \widehat{Y}_1 \\ \vdots \\ \widehat{Y}_n \end{bmatrix} = \begin{bmatrix} X_{11} & \dots & X_{1d} \\ \vdots & \ddots & \vdots \\ X_{n1} & \dots & X_{nd} \end{bmatrix} \times \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix} + \begin{bmatrix} c \\ \vdots \\ c \end{bmatrix}$$

Data and models as matrices

- Model: Predictions as matrix multiplication

$$\begin{bmatrix} \widehat{Y}_1 \\ \vdots \\ \widehat{Y}_n \end{bmatrix} = \begin{bmatrix} X_{11} & \dots & X_{1d} \\ \vdots & \ddots & \vdots \\ X_{n1} & \dots & X_{nd} \end{bmatrix} \times \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix} + \begin{bmatrix} c \\ \vdots \\ c \end{bmatrix}$$

$$\widehat{Y} = X \times w + c$$

Data and models as matrices

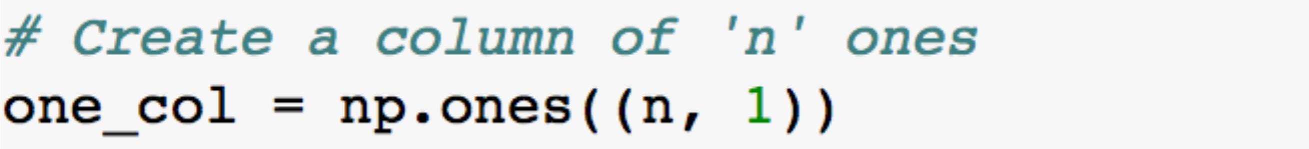
- Model: Predictions as matrix multiplication

$$\begin{bmatrix}\widehat{Y}_1\\ \vdots \\ \widehat{Y}_n\end{bmatrix} = \begin{bmatrix} X_{11} & \dots & X_{1d} & 1\\ \vdots & \ddots & \vdots & \vdots \\ X_{n1} & \dots & X_{nd} & 1\end{bmatrix} \times \begin{bmatrix} w_1 \\ \vdots \\ w_d \\ c\end{bmatrix}$$

Combine coefficients & intercepts into a single array by appending a column of ones to the data matrix

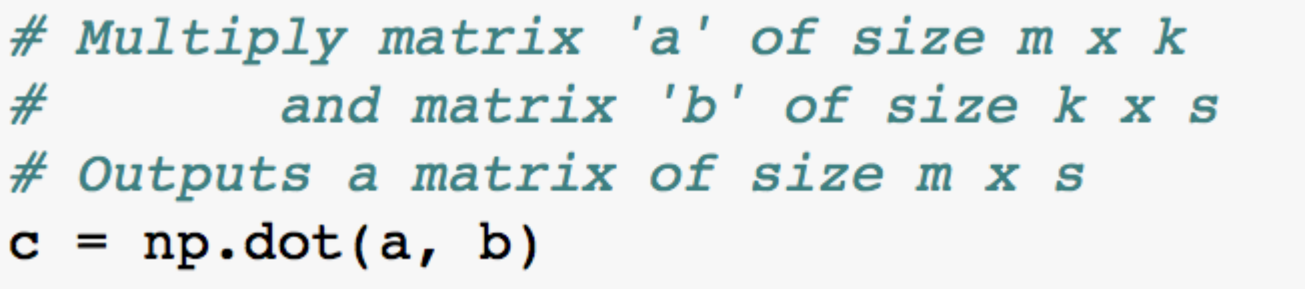
numpy: basic matrix operations

- Create column matrix of ones



```
# Create a column of 'n' ones
one_col = np.ones((n, 1))
```

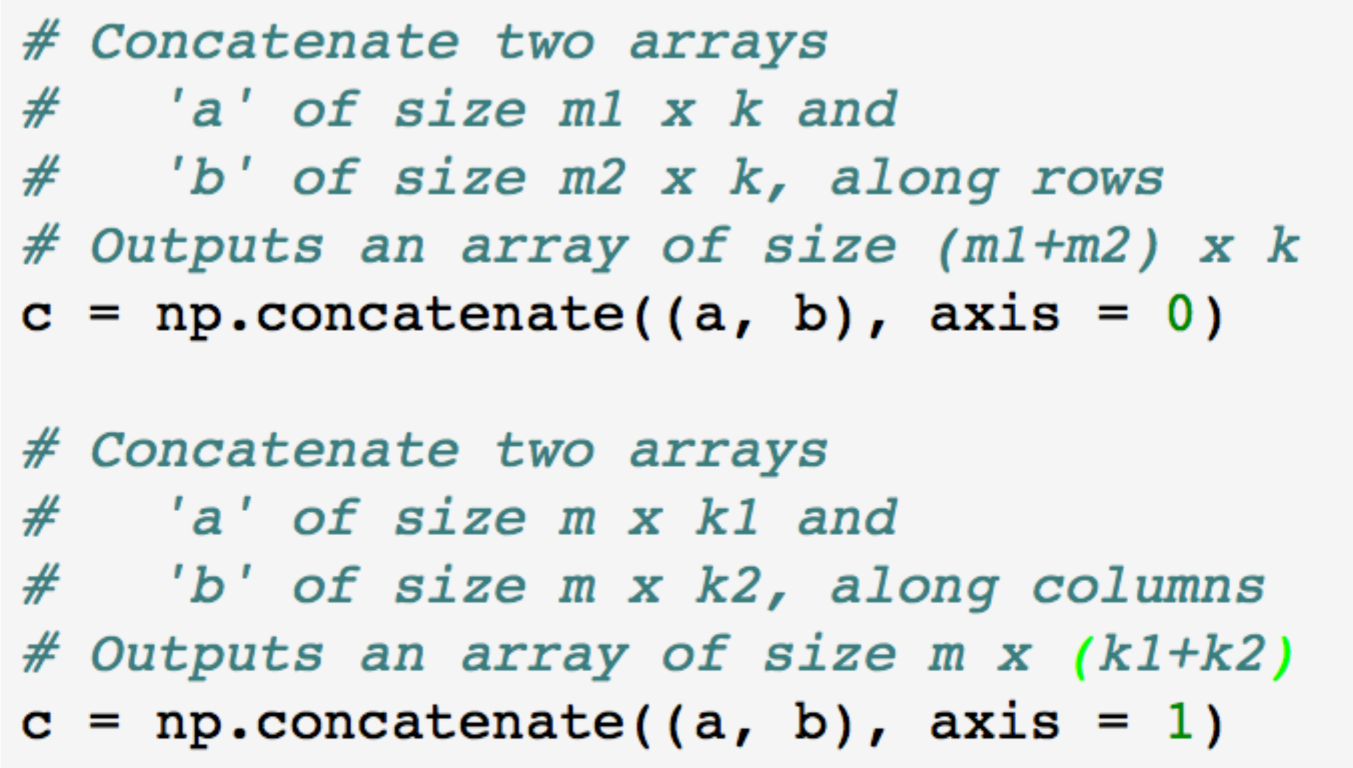
- Matrix multiplication



```
# Multiply matrix 'a' of size m x k
#       and matrix 'b' of size k x s
# Outputs a matrix of size m x s
c = np.dot(a, b)
```

numpy: basic matrix operations

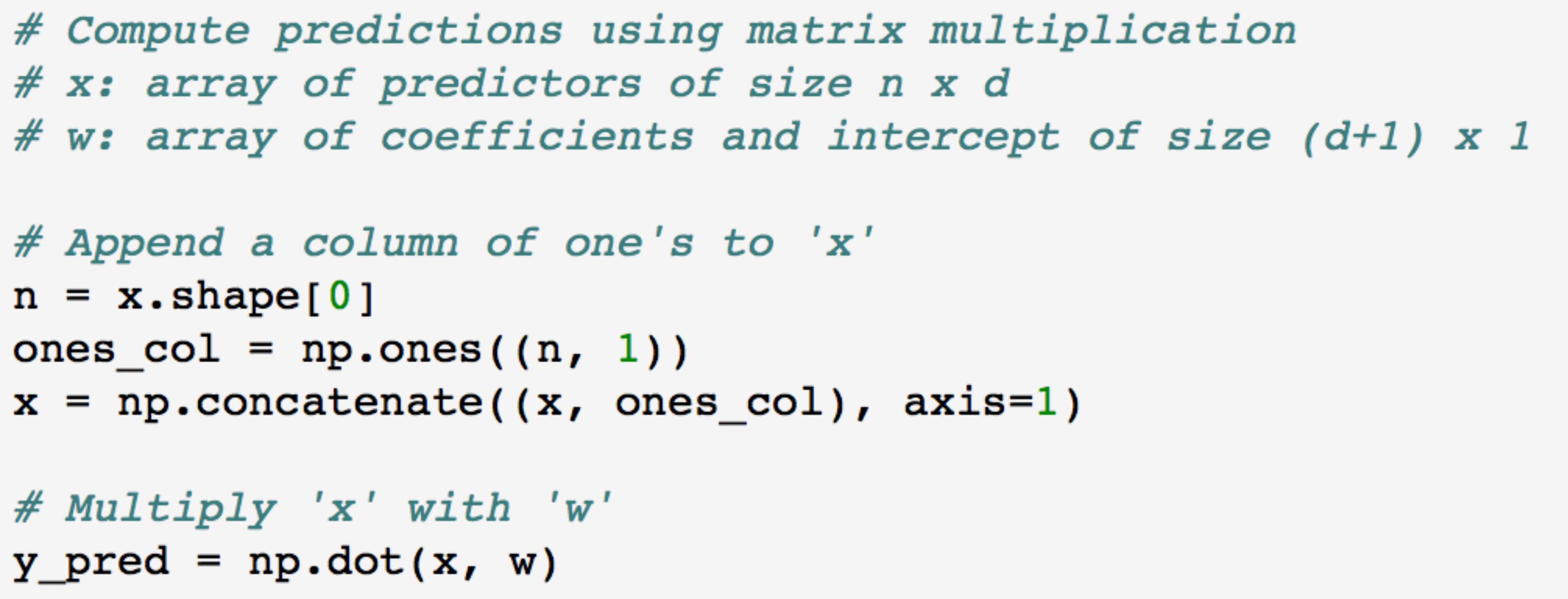
- Concatenating arrays



```
# Concatenate two arrays
#    'a' of size m1 x k and
#    'b' of size m2 x k, along rows
# Outputs an array of size (m1+m2) x k
c = np.concatenate((a, b), axis = 0)

# Concatenate two arrays
#    'a' of size m x k1 and
#    'b' of size m x k2, along columns
# Outputs an array of size m x (k1+k2)
c = np.concatenate((a, b), axis = 1)
```

Computing predictions in numpy

- Append column of ones and multiply

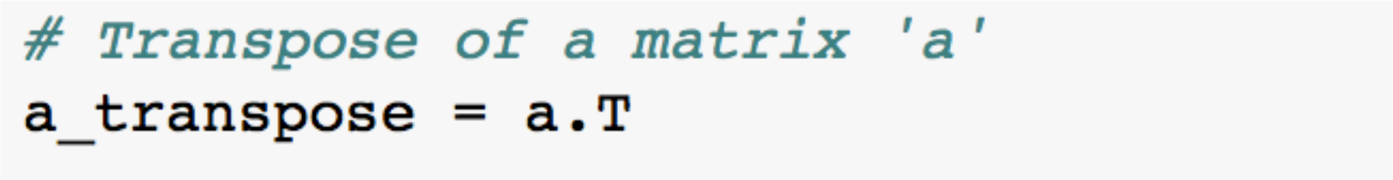


```
# Compute predictions using matrix multiplication
# x: array of predictors of size n x d
# w: array of coefficients and intercept of size (d+1) x 1

# Append a column of one's to 'x'
n = x.shape[0]
ones_col = np.ones((n, 1))
x = np.concatenate((x, ones_col), axis=1)

# Multiply 'x' with 'w'
y_pred = np.dot(x, w)
```

w contains
the intercept

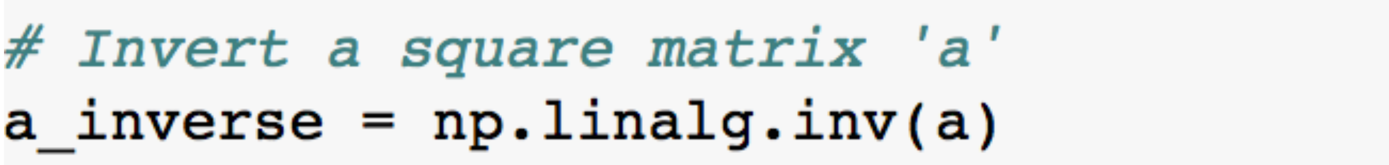
numpy: other useful operations

- Matrix transpose



```
# Transpose of a matrix 'a'
a_transpose = a.T
```

- Matrix inverse



```
# Invert a square matrix 'a'
a_inverse = np.linalg.inv(a)
```

Review:
Multiple Linear Regression

Least-squares Solution

- Training data:

$$X = \begin{bmatrix} X_{11} & \dots & X_{1d} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ X_{n1} & \dots & X_{nd} & 1 \end{bmatrix} \qquad Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix}$$

- Model parameters:

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_d \\ c \end{bmatrix}$$

c = w[-1]

Least-squares Solution

- Predictions:

$$\widehat{Y}_i = \sum_{j=1}^{d} w_j X_{ij}$$

- Minimize Least-squares Loss:

$$\begin{aligned} L(w) &= \sum_{i=1}^{n} (\widehat{Y}_i - Y_i)^2 \\ &= \sum_{i=1}^{n} \Big(\sum_{j=1}^{d} w_j X_{ij} - Y_i \Big)^2 \end{aligned}$$

Least-squares Solution

- Set derivatives to zero:

$$\frac{\partial L(w)}{\partial w_1} = 0 \quad \ldots \quad \frac{\partial L(w)}{\partial w_d} = 0$$

- Solve system of linear equations

$$2\sum_{i=1}^{n}\left(\sum_{j=1}^{d} w_j X_{ij} - Y_i\right) X_{i1} = 0$$

$$\vdots$$

$$2\sum_{i=1}^{n}\left(\sum_{j=1}^{d} w_j X_{ij} - Y_i\right) X_{id} = 0$$

Least-squares Solution

- Formulating in matrix form:

$$X^\top (Xw - Y) = 0$$

or

$$X^\top X w = X^\top Y$$

- Solution:

$$\boxed{w = (X^\top X)^{-1} X^\top Y}$$

Least-squares Solution

- Formulating in matrix form:

$$X^\top (Xw - Y) = 0$$

or

$$X^\top X w = X^\top Y$$

- Solution:

What can go wrong?

$$\boxed{w = (X^\top X)^{-1} X^\top Y}$$

Evaluating regression model

- R^2 score:

$$\mathrm{R}^2 = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}}$$

  - Residual Sum of Squares (RSS)

$$\mathrm{RSS} = \sum_{i=1}^{n} (\widehat{Y}_i - Y_i)^2$$

  - Total Sum of Squares (TSS)

$$\mathrm{TSS} = \sum_{i=1}^{n} (\overline{Y} - Y_i)^2$$

where $\overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$

How better is the model over the best constant predictor

Tasks

- Fit regression model to training set
  - multiple_linear_regression_fit
    - input: x_train, y_train
    - returns: w, c
- Evaluate model on test set
  - multiple_linear_regression_score
    - input: w, c, x_test, y_test
    - returns: r_squared, y_pred

Today's lab: Problem 1

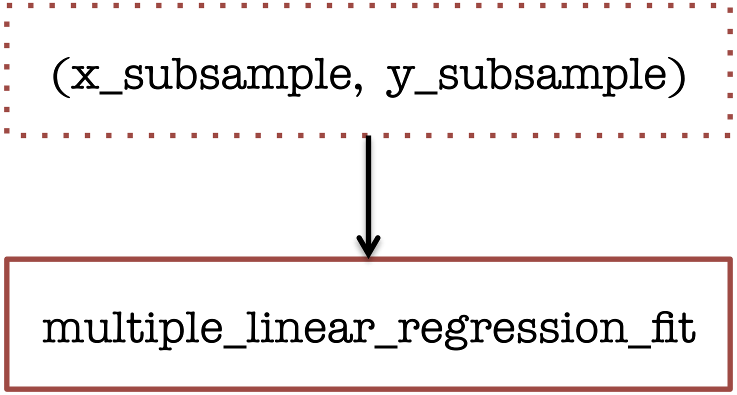
a) Multiple linear regression from scratch
   - Fit regression model
   - Score regression model

b) Confidence intervals on model parameters
   - Analyze histograms of model parameters
   - Compute confidence intervals

Subsampling

- Repeat 200 times: Random subsamples of size 100

X_1	X_2	Y
3	4	5
2	1	0
5	2	3
1	0	1
	⋮	
9	3	6
7	0	1



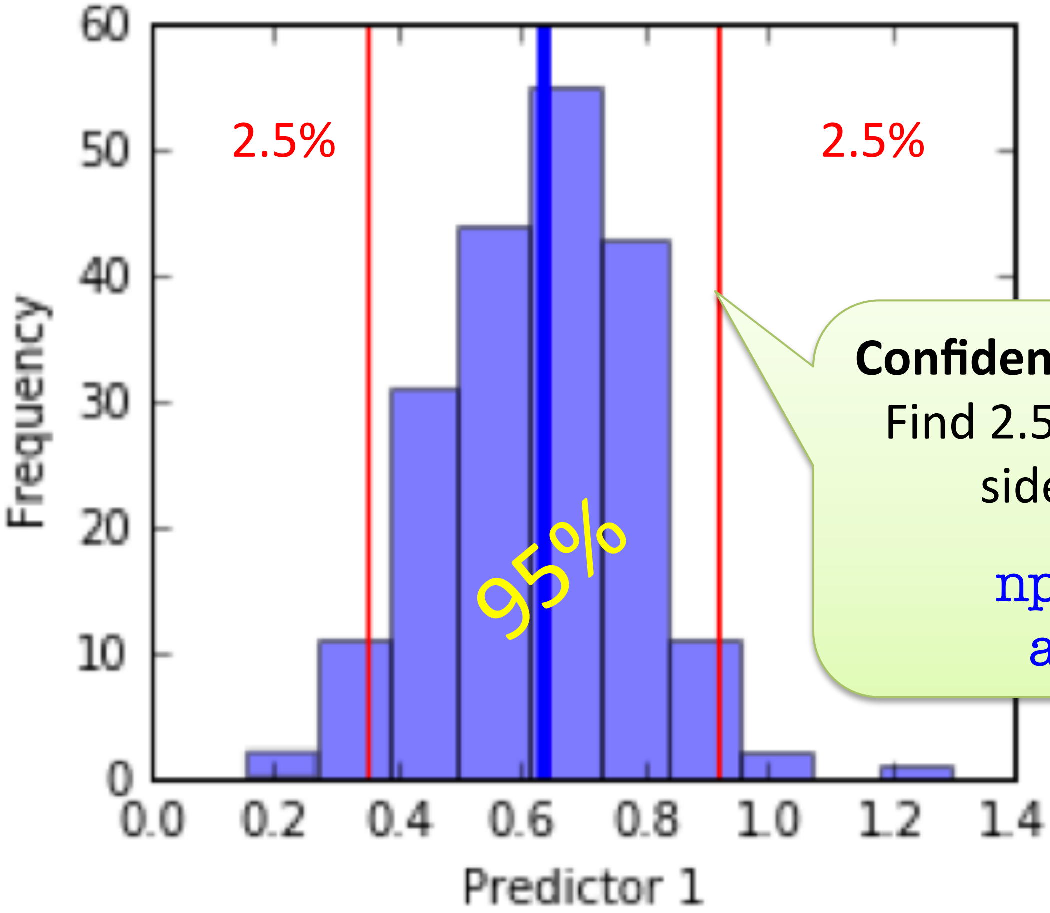
(x_subsample, y_subsample)

multiple_linear_regression_fit

Hint:
Use np.random.permutation
to permute the data set,
and pick the top 100 entries

Confidence Intervals: Subsampling

- Plot histogram of coefficients: ax.hist(...)



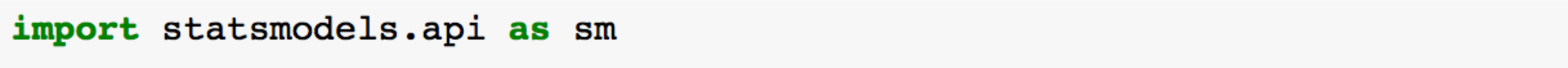
Confidence interval at 95% level

Find 2.5% percentile on either side of the histogram

np.percentile(...)
ax.axvline(...)

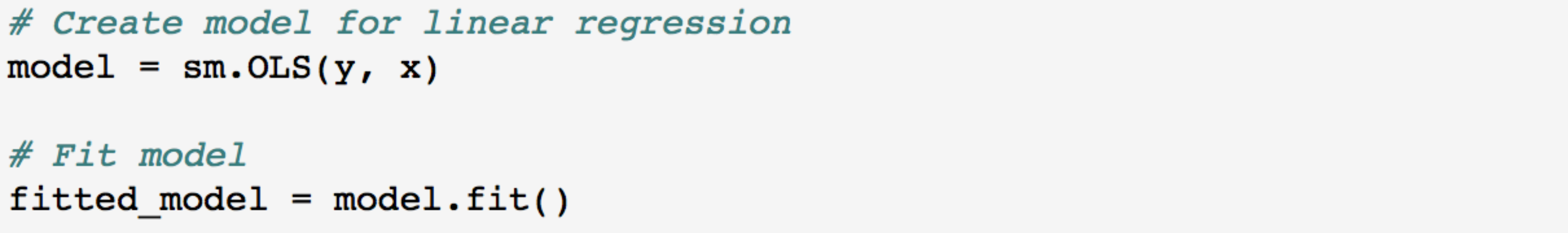
Confidence Intervals: Statsmodels

- Built-in python module



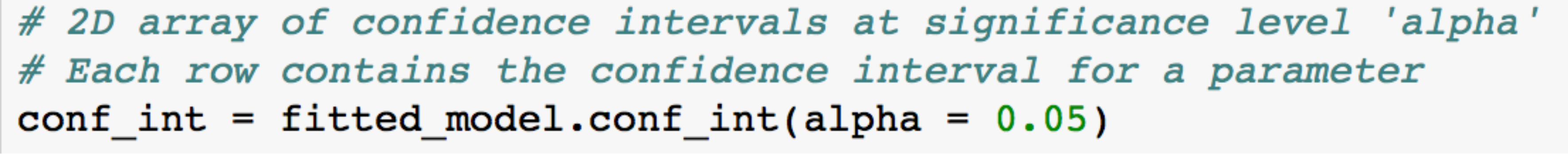
```
import statsmodels.api as sm
```

- Ordinary least squares (OLS) regression



```
# Create model for linear regression
model = sm.OLS(y, x)

# Fit model
fitted_model = model.fit()
```

- Confidence intervals for fitted model

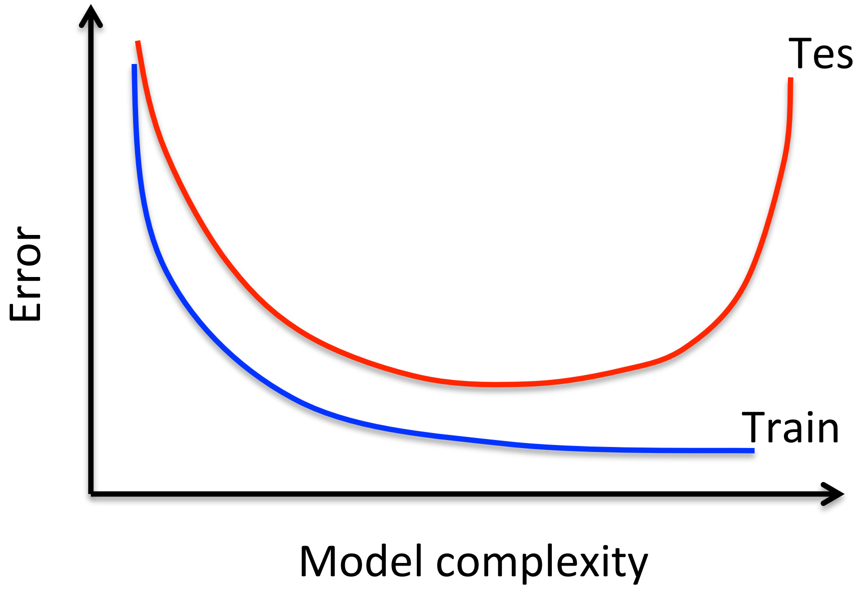


```
# 2D array of confidence intervals at significance level 'alpha'
# Each row contains the confidence interval for a parameter
conf_int = fitted_model.conf_int(alpha = 0.05)
```

Review:
Model Selection

Training vs. Test errors

- Polynomial regression
  - Model complexity: Degree of polynomial
  - Is larger always better?



Model Selection Criterion

- How does once choose the ‘best’ polynomial degree using only the training set?
- Use a *model selection criterion* as a proxy for the test error:

-2 x Log-likehood + penalty term

Model Selection Criterion

- Akaike Information Criterion
  - AIC = -2 x Log-likehood + 2 x *K*
  - For least-squares regression:

K: degree of polynomial

n: training sample size

$$\text{AIC} = n \log\left(\frac{\text{RSS}}{n}\right) + 2K$$

- Bayesian Information Criterion (BIC)
  - BIC = -2 x Log-likehood + 2 x log(*K*)
  - For least-squares regression:

$$\text{BIC} = n \log\left(\frac{\text{RSS}}{n}\right) + \log(n)K$$

Note: The AIC and BIC definitions are slightly different from the text book, and correspond to the case where the residual error variance σ^2 is unknown.