Getting Python To Learn From Only Parts Of Your Data

Ami Tavory
Final, Israel

PyConTW 2013

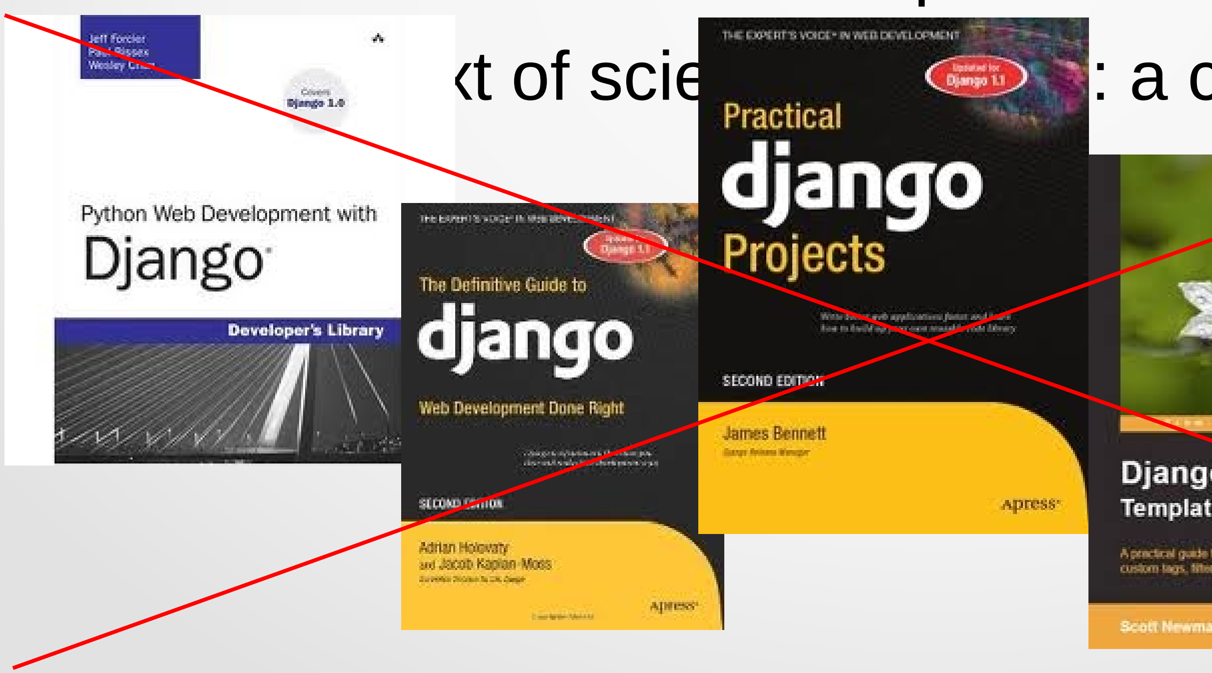
Outline

- Introduction
- Model Selection And Cross Validation
- Model Assessment And The Bootstrap
- Conclusions

Prevalence Of Computing & Python - Two Implications

- General computing:
  - Tasks & calculations unlimited by human limitations
- Python (& high-level languages):
  - Low bar for new-area experimenta
  - xt of scie : a c

As time goes by...



Low Bar For Experimentation & Model Selection

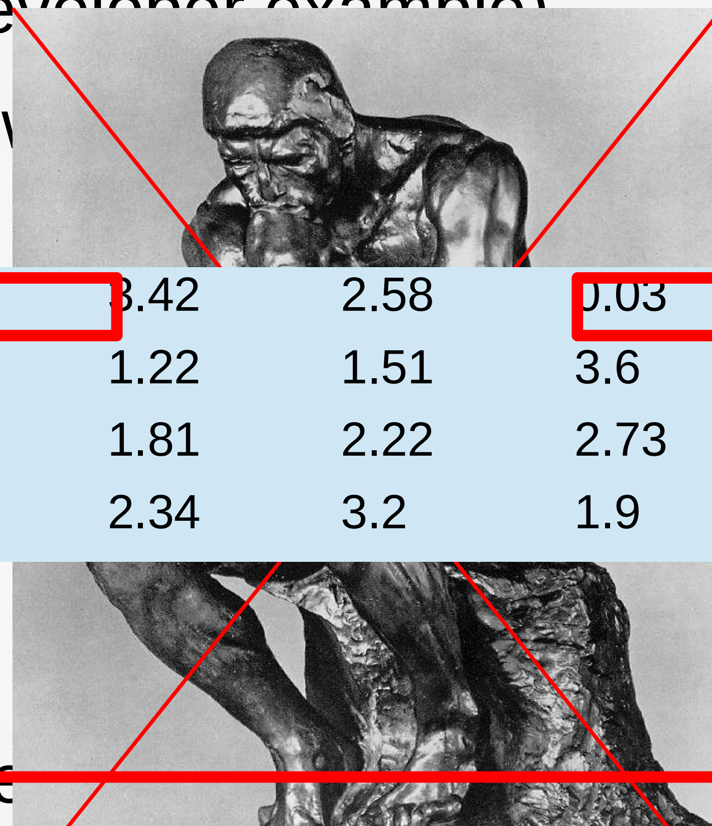
oper exa

3		
2	0.1	500
10	2	2

- Revenues seem inverse to clicks-to-purchase & page load time.
- Can we quantify using SciPy?

Computing Power & Model Assessment

- (Another web developer example)
- “The XYZ framew ts in under 2.5 msec”



1.71	2.55	12.23	3.42	2.58	0.03	2.46	0.01
2.56	1.17	1.46	1.22	1.51	3.6	1.9	23.99
1.12	2.73	2.21	1.81	2.22	2.73	2.63	8.13
2.23	0.00	2.0	2.34	3.2	1.9	3.4	

- Without compute tatistics:
  - With n 1.89 and 4.63

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N}[x_i], \quad \rho = \sqrt{\frac{1}{N}\sum_{i=1}^{N}[x_i - \bar{x}]}$$

- No nee
  - We can, e.g., assess the median and its confidence.

This Talk

- Two problems, common solution:

  Getting Python To Learn From

  Only Parts Of Your Data

- Talk will use:

  nPy / Sc

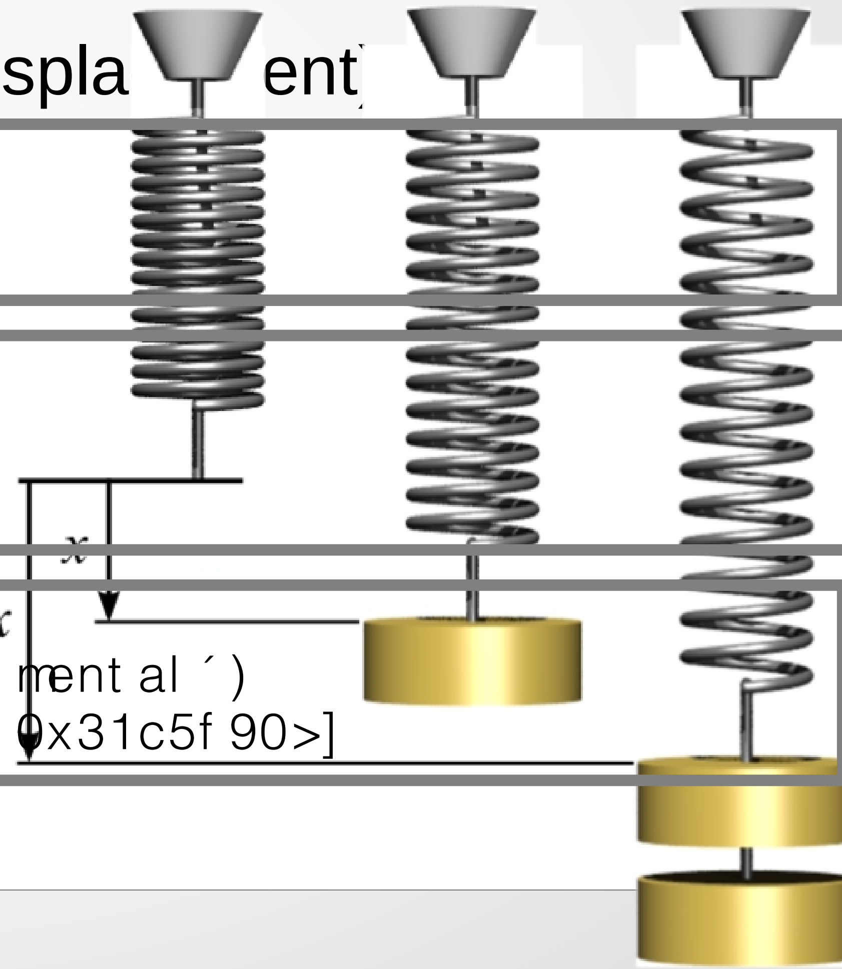
Outline

- Introduction
- Model Selection And Cross Validation
- Model Assessment And The Bootstrap
- Conclusions

Example Domain – Hooke's Law

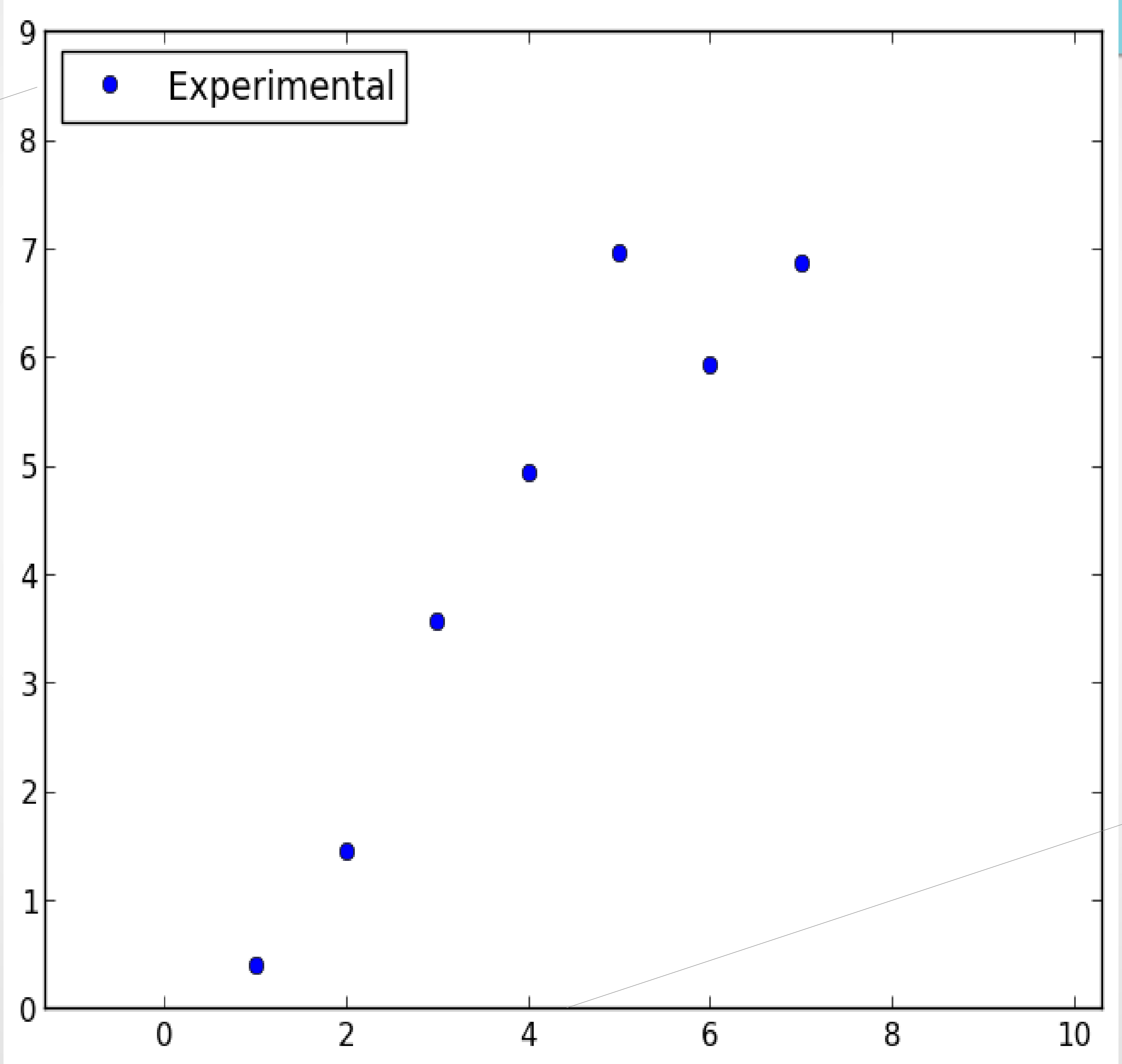
- Basic experiment with springs.
- Hooke's Law (Force proportional to displacement)
  - $F \sim x$ (F = force, x = displacement)

```
>>> from numpy import *
>>> from scipy import *
>>> from matplotlib.pyplot import *
>>>
>>> # measure 8 displacements
>>> x = arange(1, 8)
>>> # note measurement errors
>>> F = x + 3 * random.rand(8)
>>>
>>> plot(x, F, 'bo', label = 'Experimental')
[<matplotlib.lines.Line2D object at 0x31c5f90>]
>>> show()
```



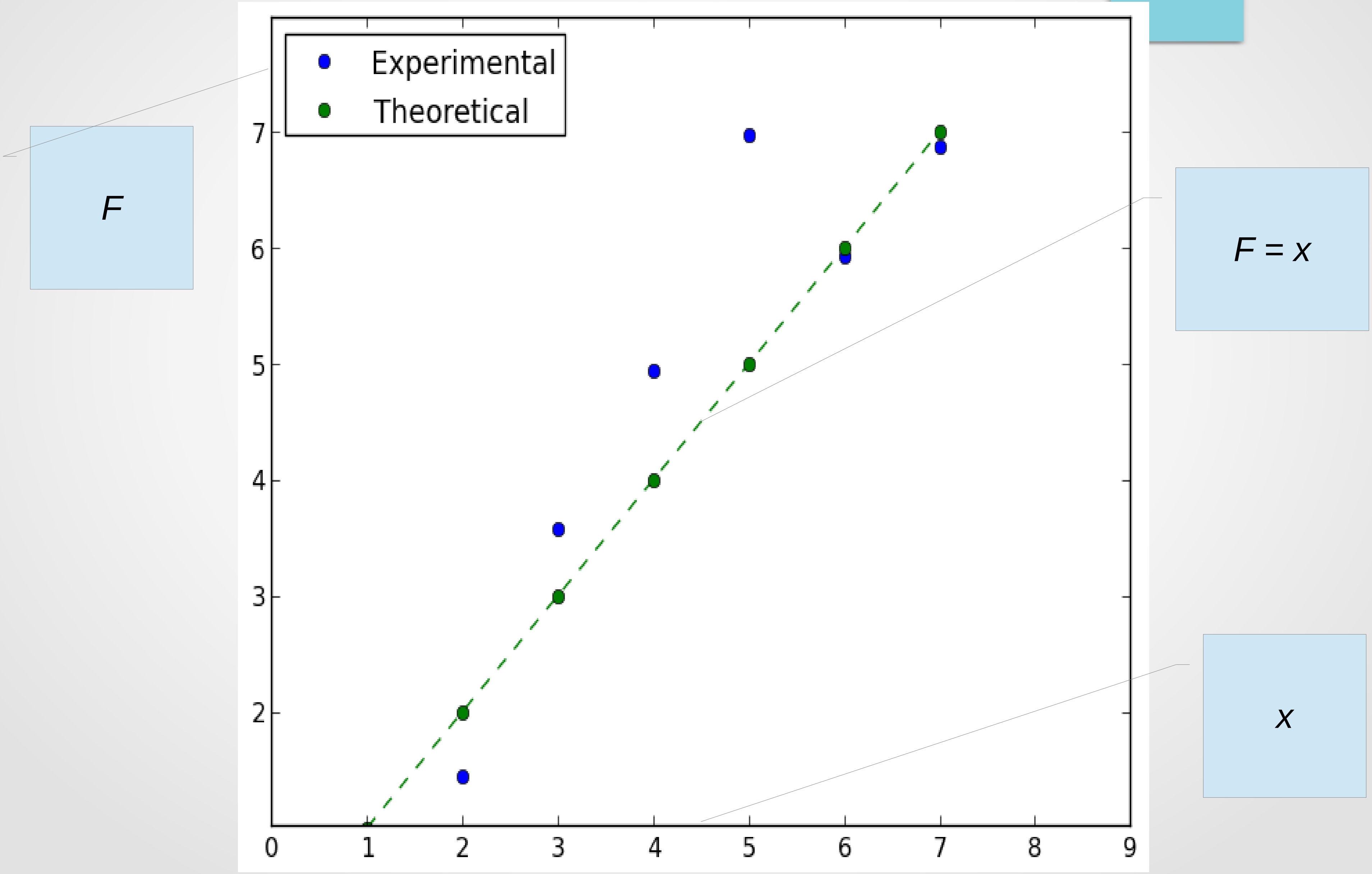
Example Domain – Hooke's Law – Experiment Results

F



x

Example Domain – Hooke's Law – Experiment & Theoretical Results



Example Predictor – Polynomial Fitting

- Polynomial fit (scipy.polyfit)

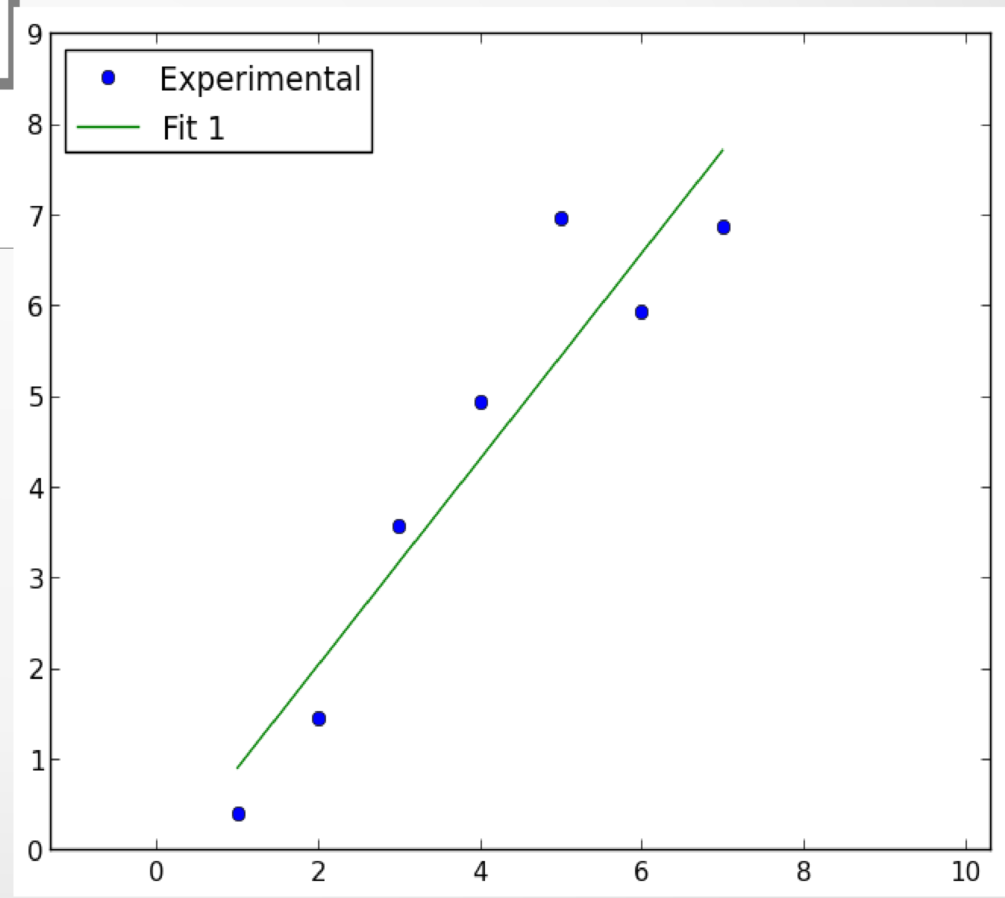
```
>>> help(polyfit)
polyfit(x, y, deg)
    Least squares polynomial fit.

    Fit a polynomial ``p(x) = p[0] * x**deg + ... + p[deg]``
of degree `deg`
    to points `(x, y)`. Returns a vector of coefficients `p`
that minimises
    the squared error.
```

Example Predictor – Polynomial Fitting Applied

- Polynomial fit (scipy.polyfit)

```
>>> x = arange(1, 8)
>>> F = x + 3 * random.randn(8)
>>> # Fit the data to a straight line.
>>> print polyfit(x, F, 1)
>>> [ 1.13 -0.23]
>>> # F ~ 1.13 * x - 0.23
```



The Problem – Which Model?

- Polynomial fit (scipy.polyfit)

```
>>> x = arange(1, 8)
>>> F = x + 3 * random.randn(8)
>>> # Fit the data to a
straight line
>>> print polyfit(x, F, 1)
>>> [ 1.13   -0.23]
>>> # F ~ 1.13 * x - 0.23
```

Cheating!

<u>What Types Of Problems Are We Considering?</u>

Find model relating # clicks + load time / revenue

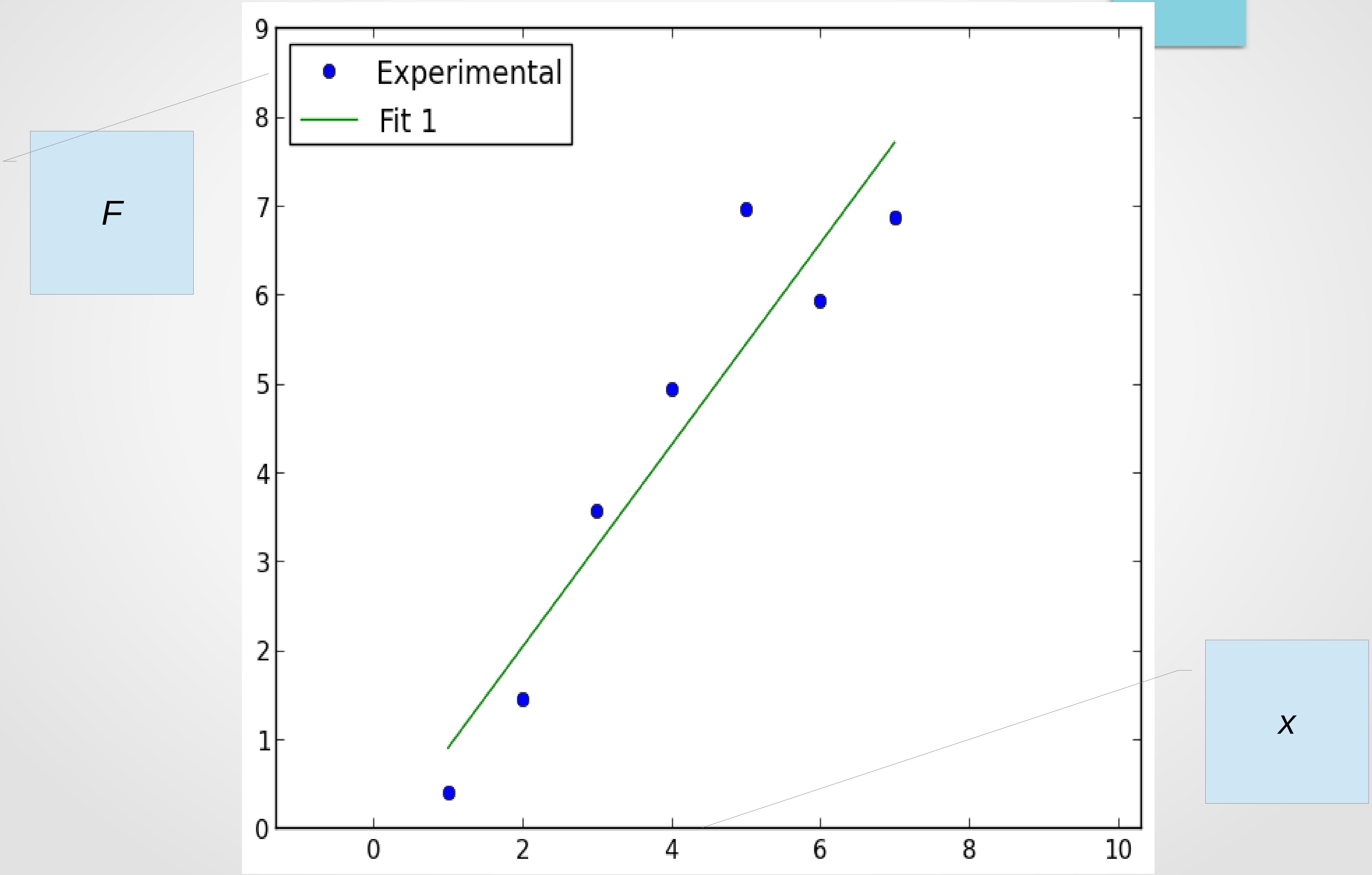
The Problem – Alternatives?

- Polynomial fit (scipy.polyfit)

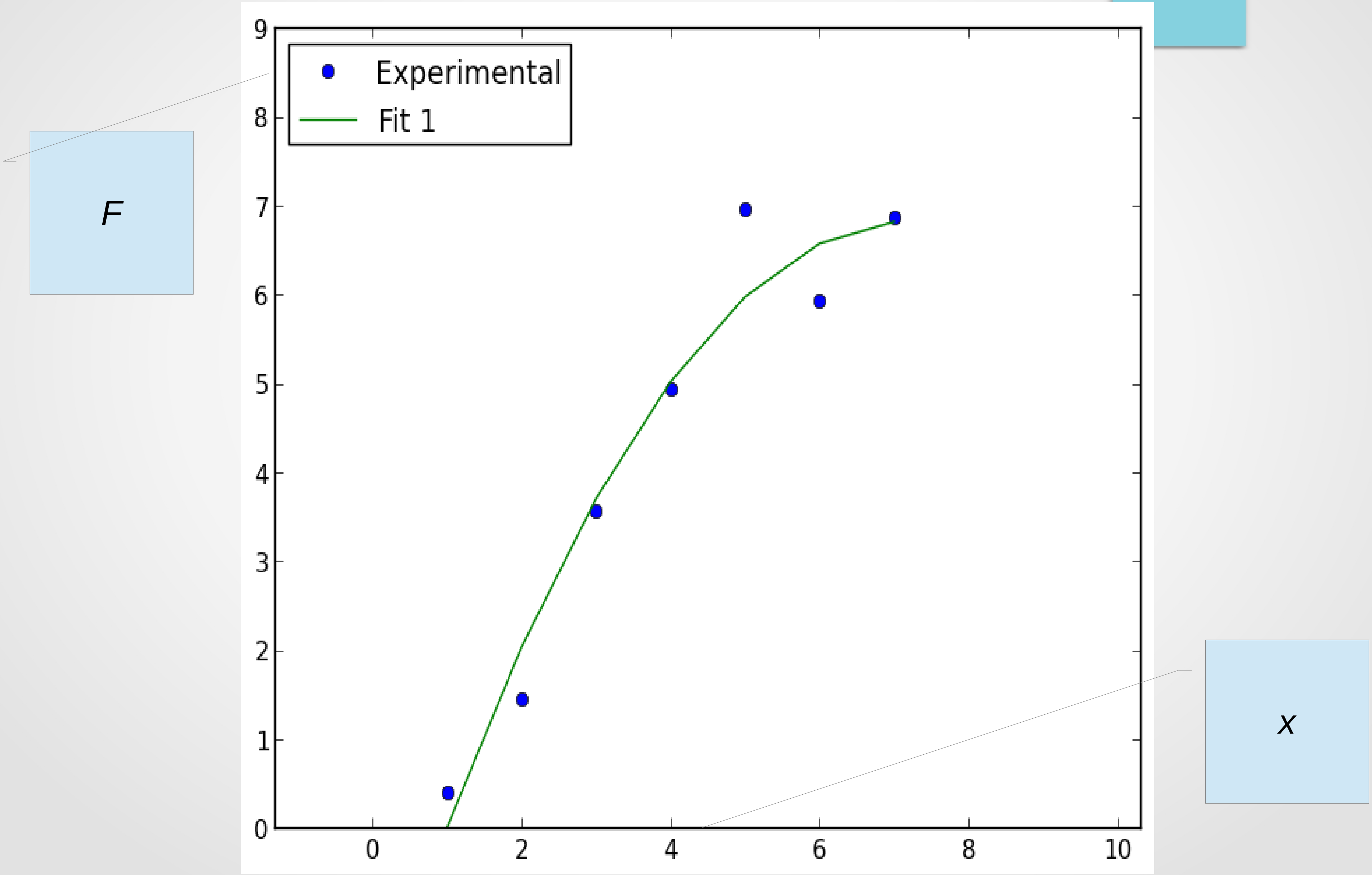
```
>>> x = arange(1, 8)
>>> F = x + 3 * random.randn(8)
>>> # Fit the data to a
straight line
>>> print polyfit(x, F, 1)
>>> [ 1.13  -0.23]
>>> # F ~ 1.13 * x - 0.23
```

```
>>> x = arange(1, 8)
>>> F = x + 3 * random.randn(8)
>>> # Fit the data to a parabola.
>>> print polyfit(x, F, 2)
>>> [-0.18  2.56 -2.38]
>>> # F ~ -0.18 * x^2 + 2.56 * x -
2.38 * x
```

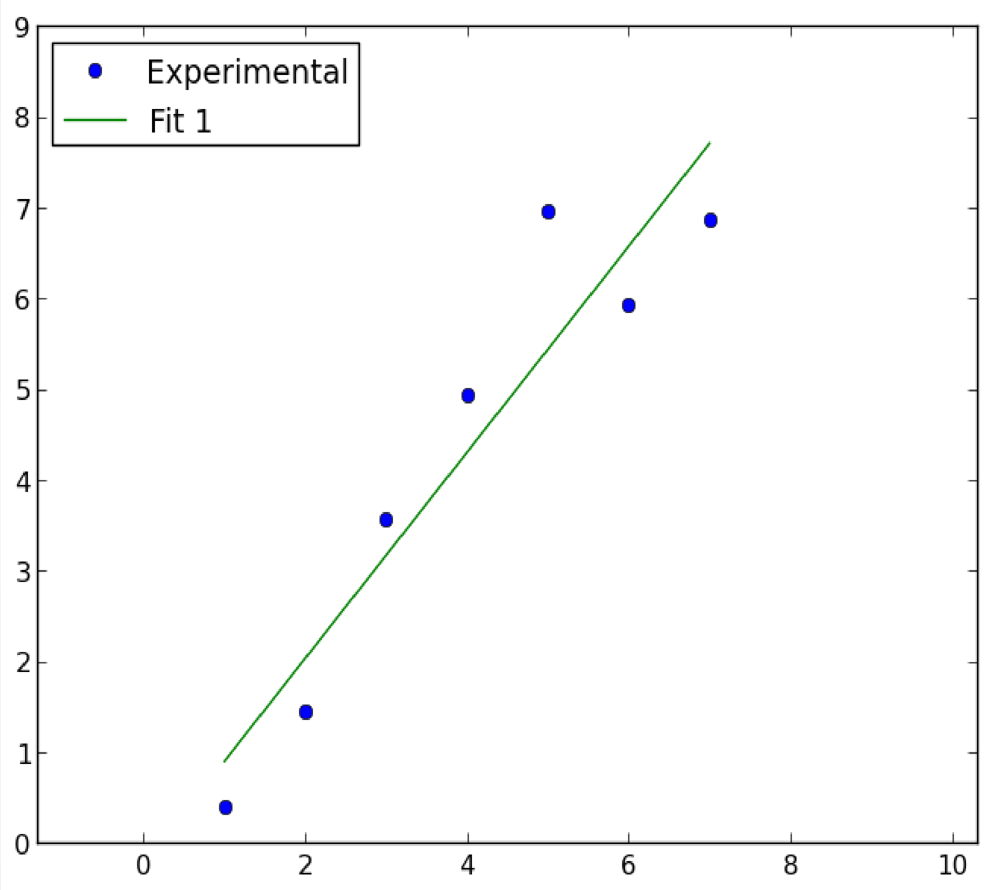
The Problem – Alternatives In Depth – Straight Line

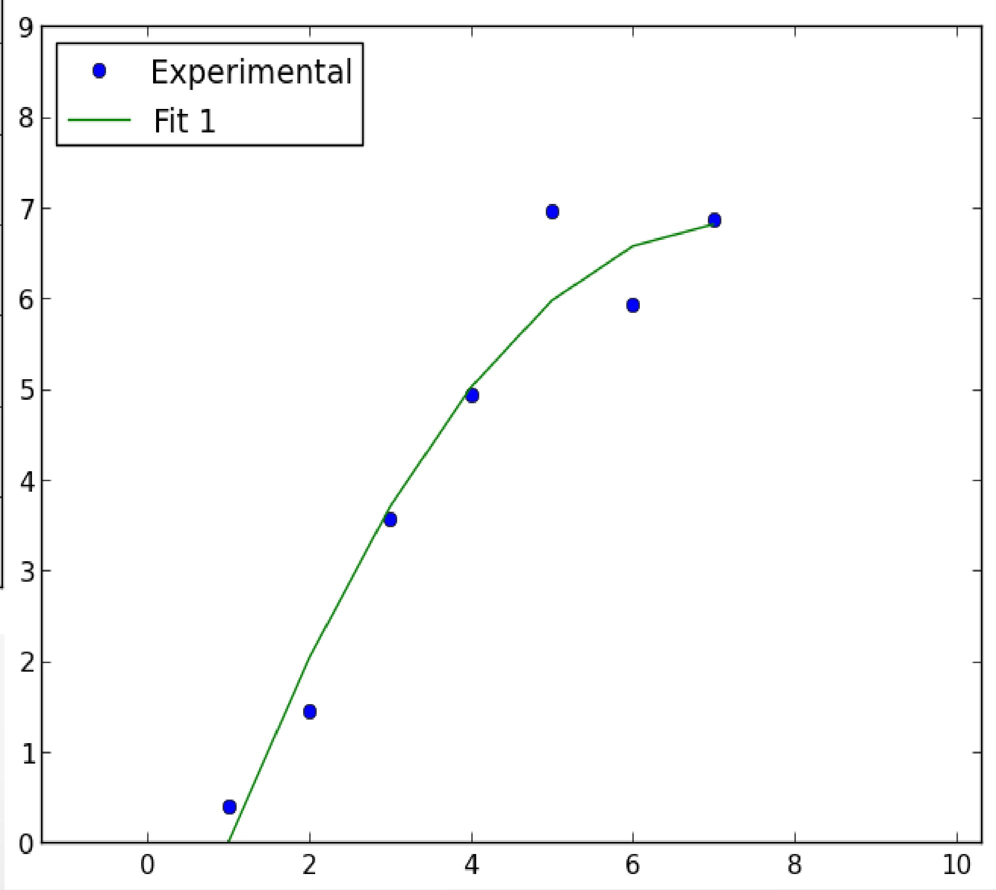


The Problem – Alternatives In Depth – Parabola

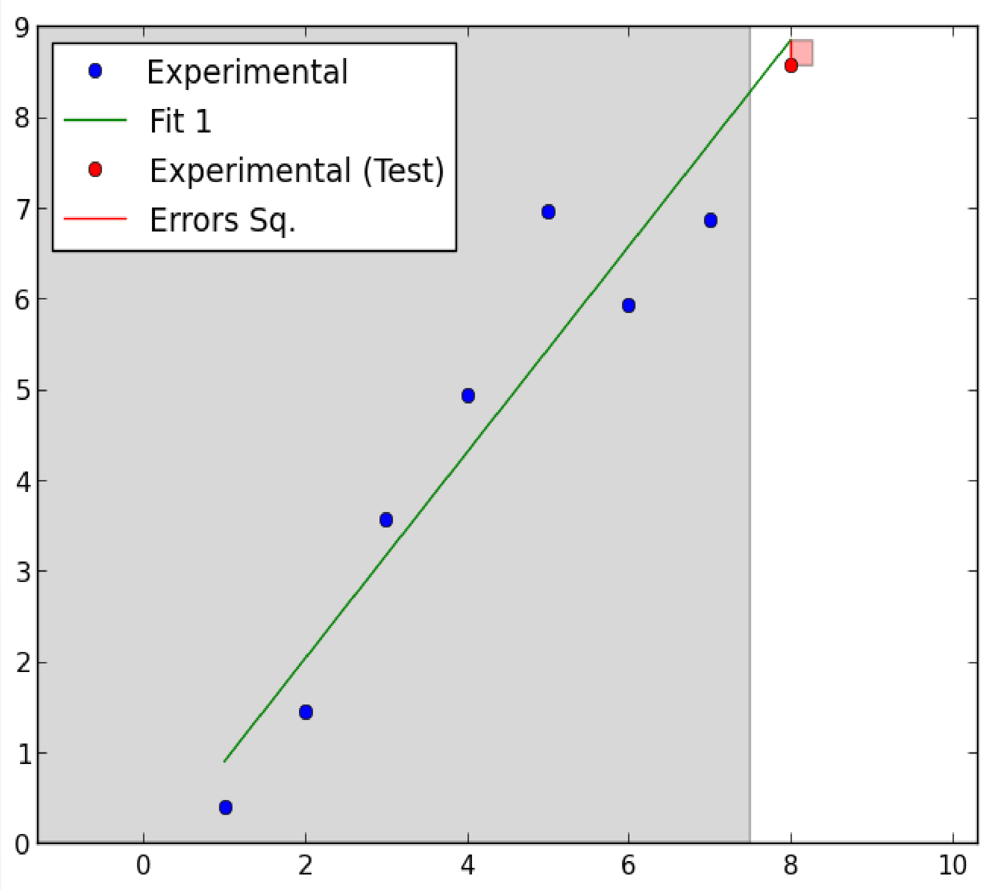


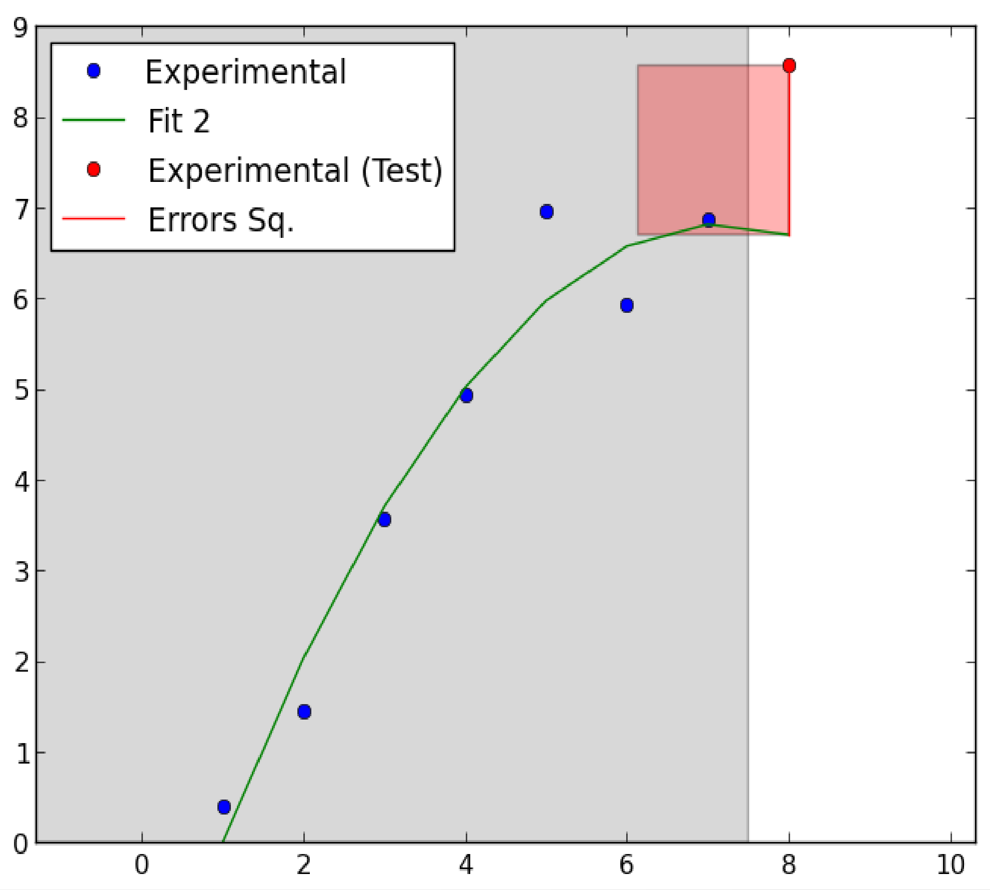
The Problem – Alternatives Comparison



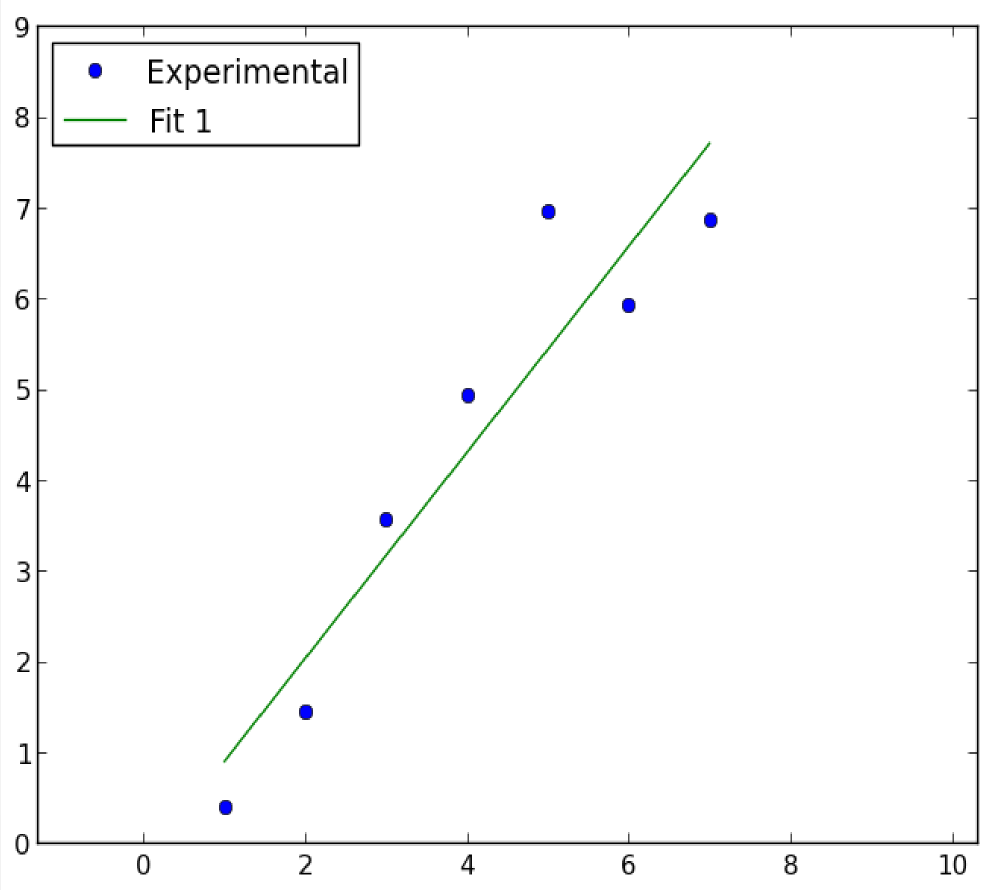


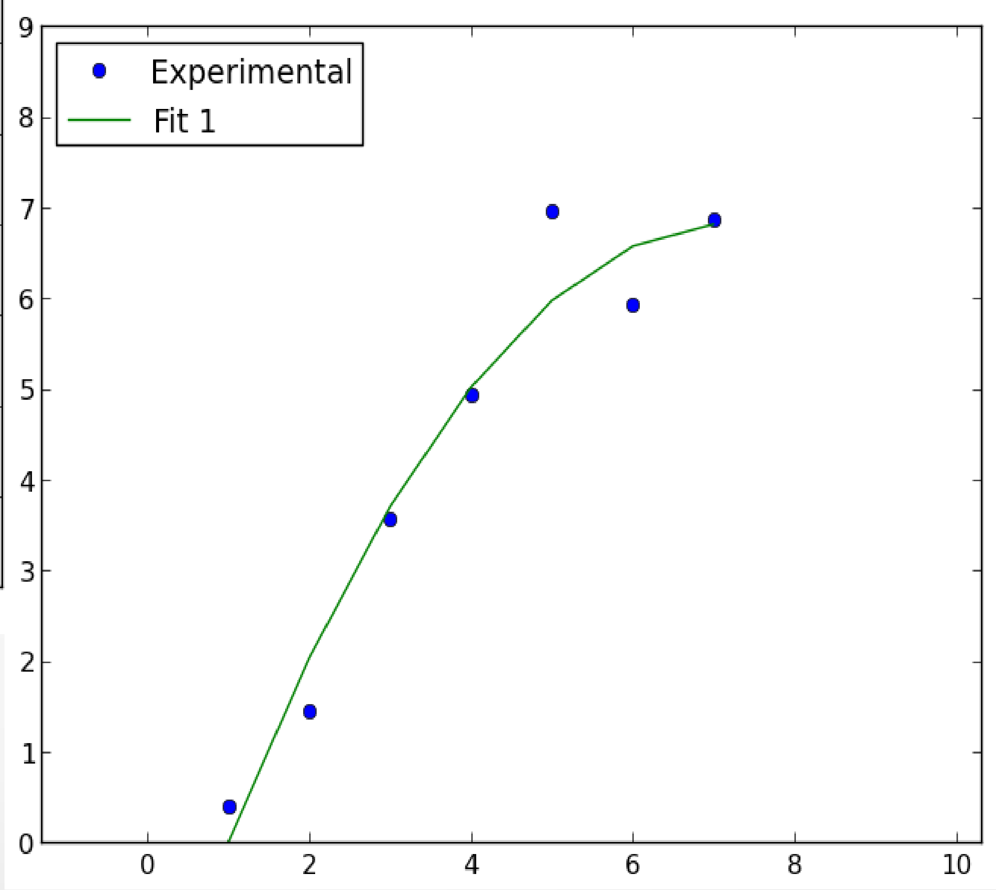
The Problem – Alternatives Comparison



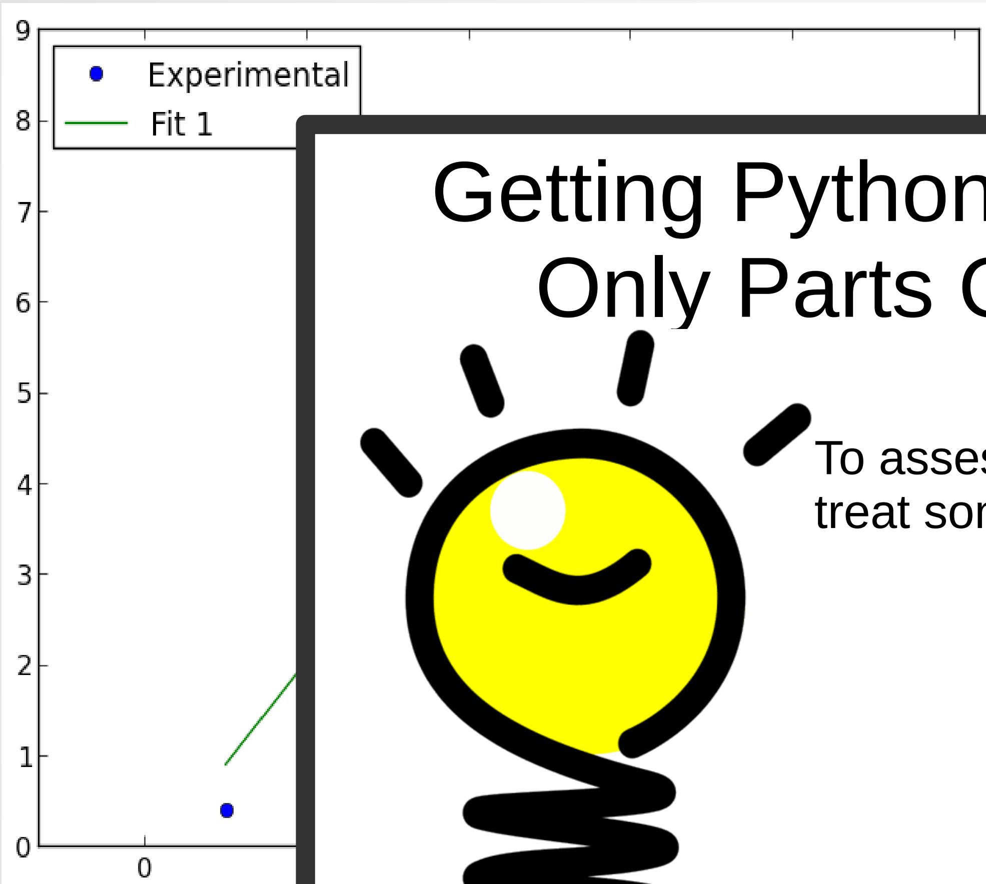


The Problem – Alternatives Comparison



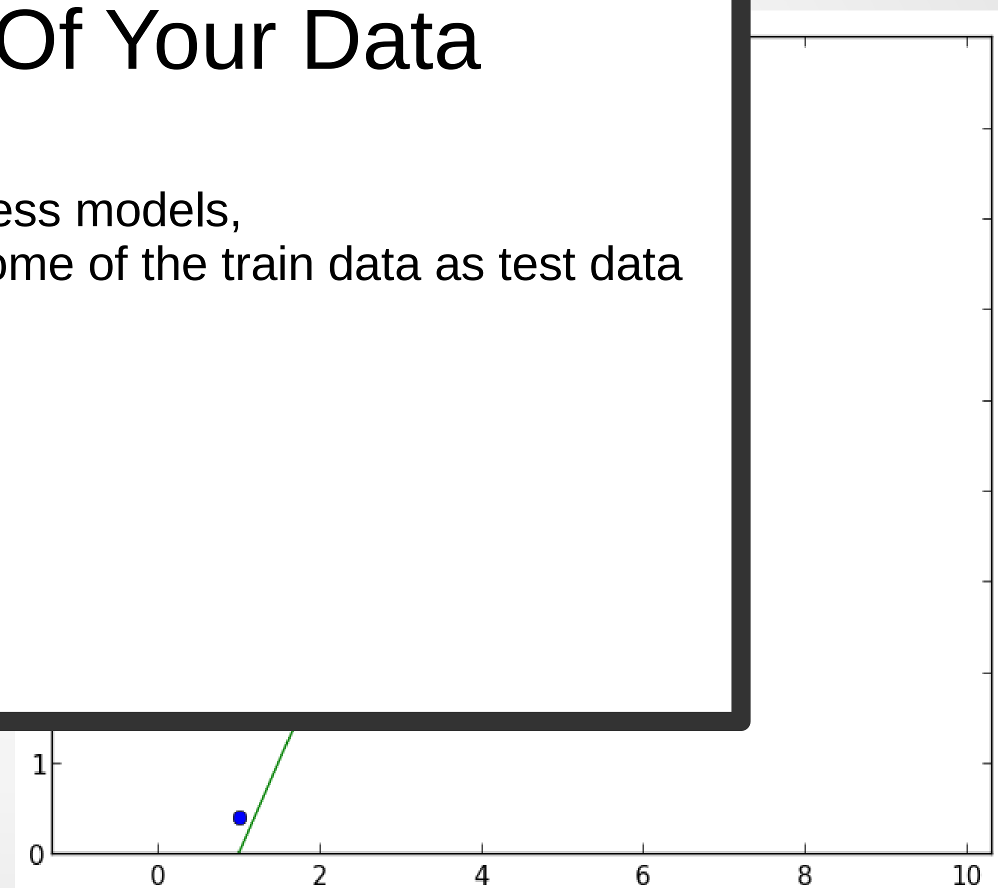


The Solution – Observation

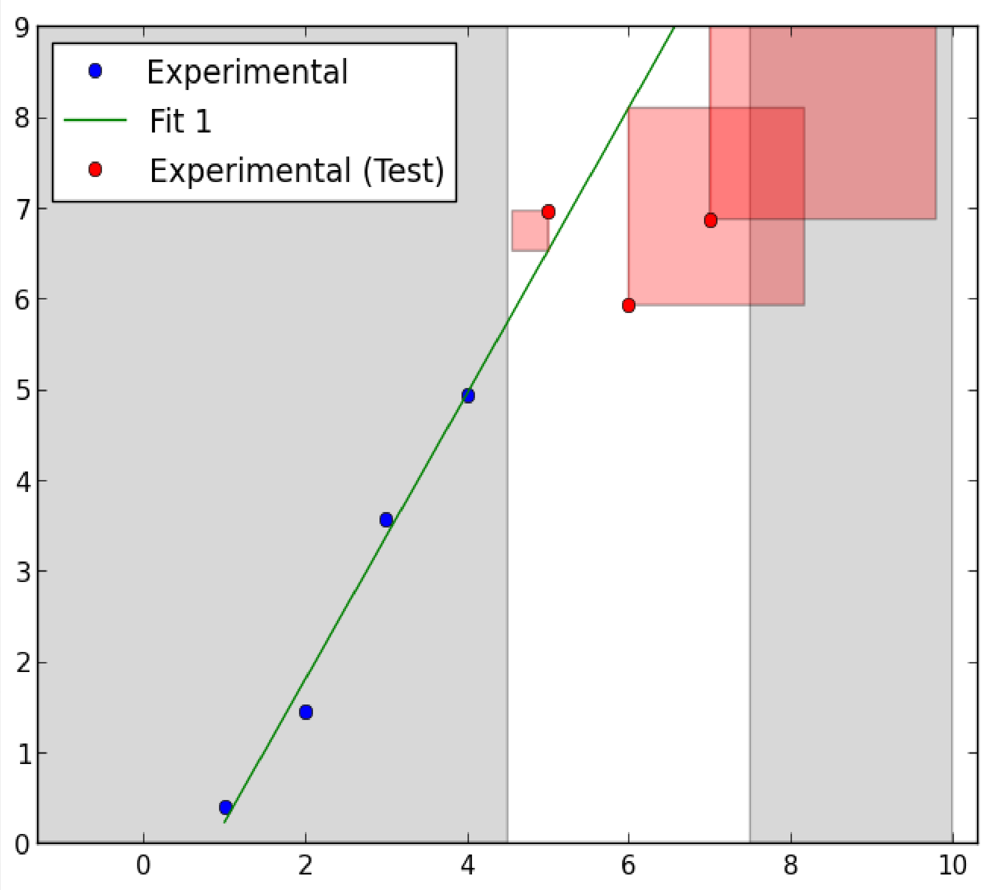


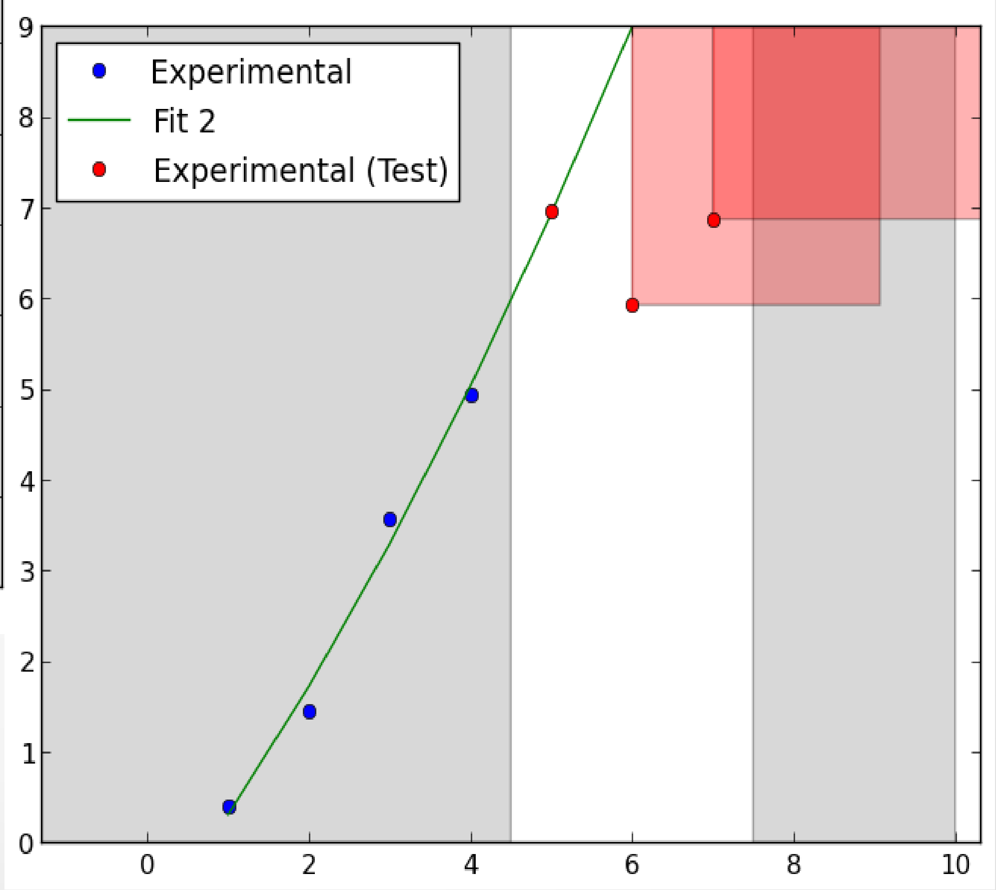
Getting Python To Learn Fron Only Parts Of Your Data

To assess models,
treat some of the train data as test data

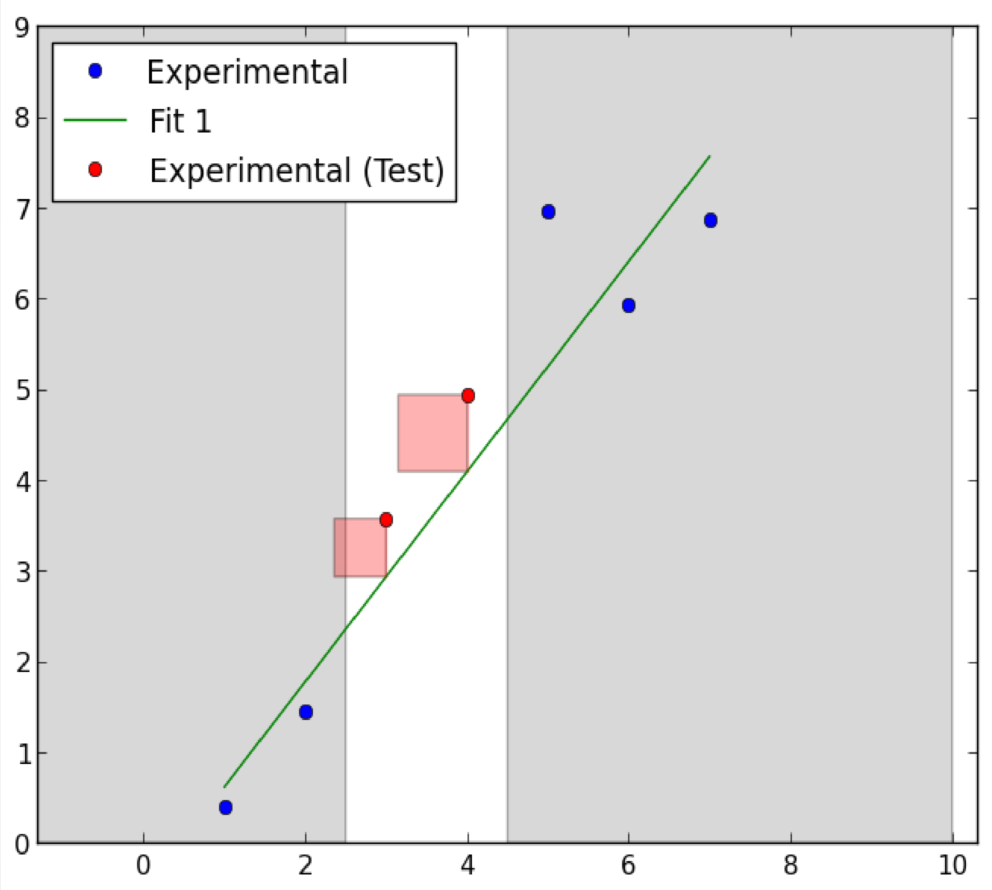


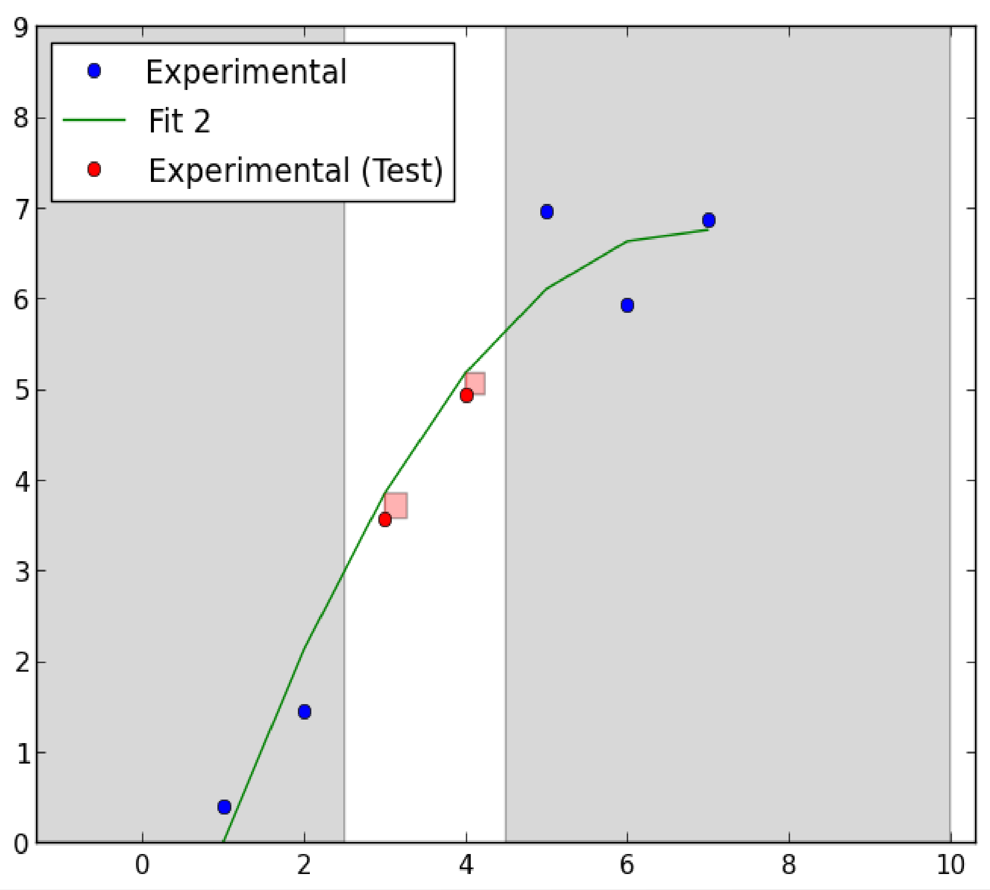
The Solution – Cross Validation



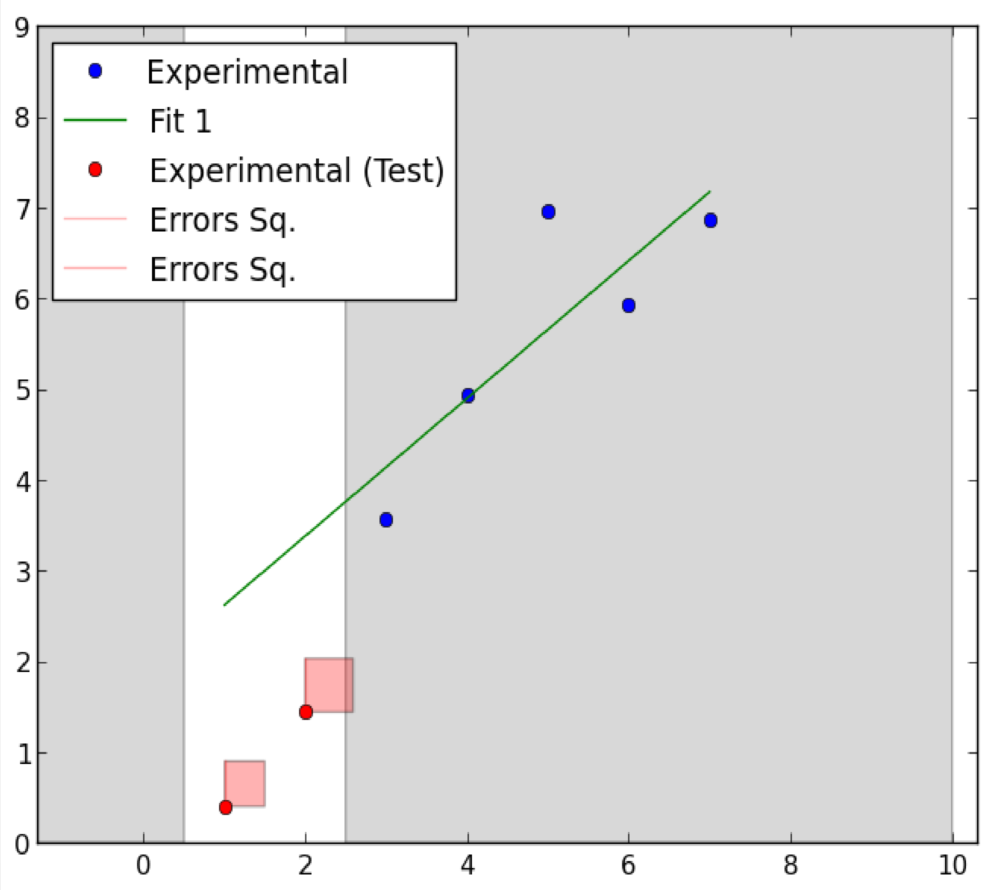


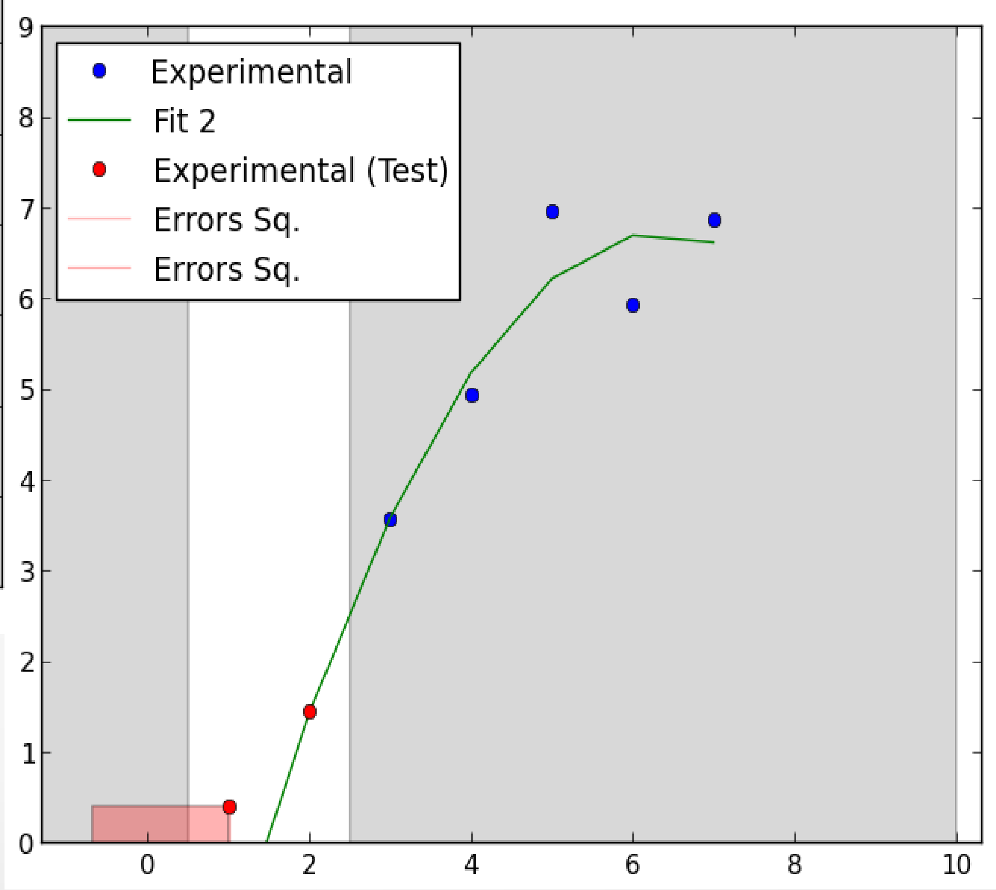
The Solution – Cross Validation



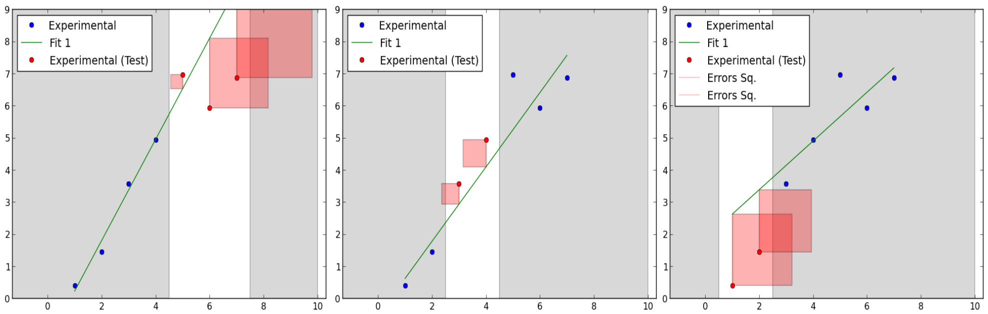


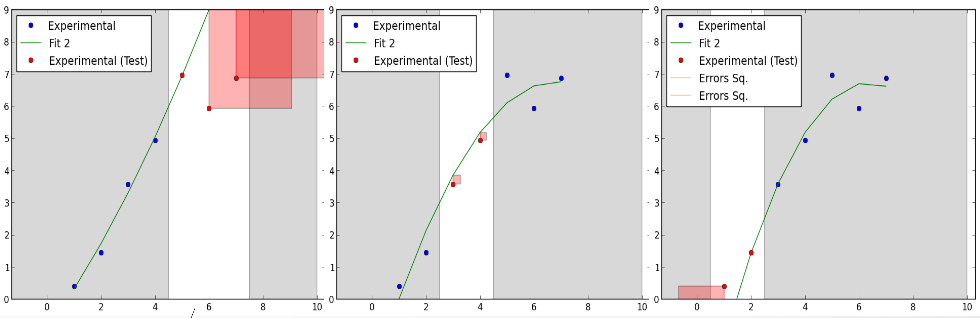
The Solution – Cross Validation





The Solution – Cross Validation

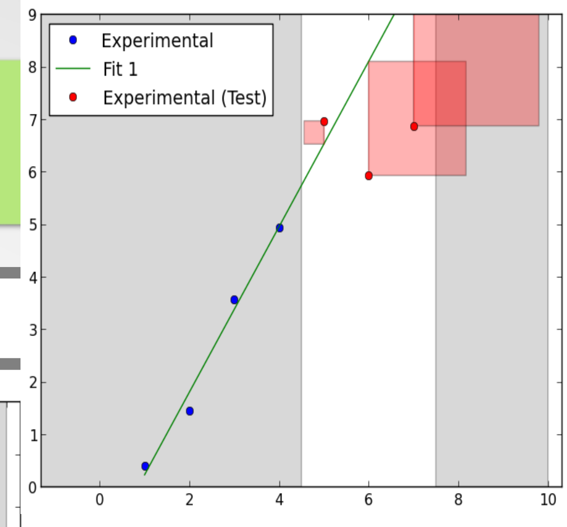


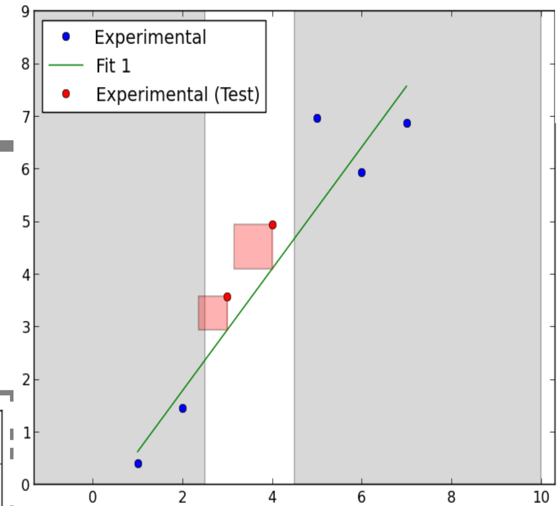


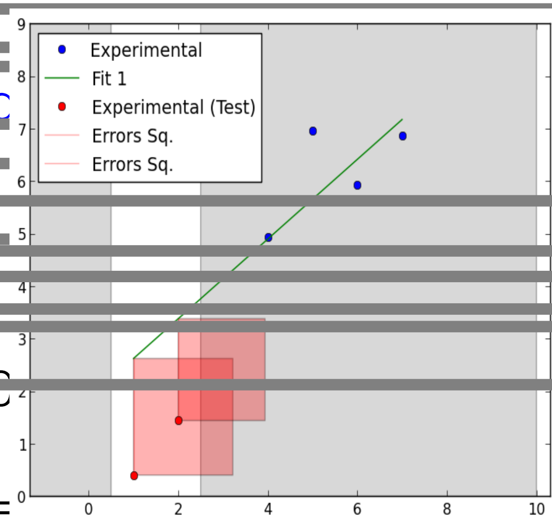
On average,
the linear model has lower error than the parabolic model

Let's See It In Code...

```
>>> from sklearn.cross_validation import *
>>>
>>> for tr, te in KFold(8):
...     print tr, te
...
[0 1 2 3 4] [5 6 7]
[0 1 4 5 6 7] [2 3]
[2 3 4 5 6 7] [0 1]
>>>
>>> Assume:
>>> # buil                                  del from train x.
train_F
>>> # find                  test F) evaluates model on test x.
test_F
>>>
>>> err = 0
>>> for tr,
...     m =                                , F[tr])
...     err += find_err(m, x[te], F[te])
```







Outline

- Introduction
- Model Selection And Cross Validation
- Model Assessment And The Bootstrap
- Conclusions

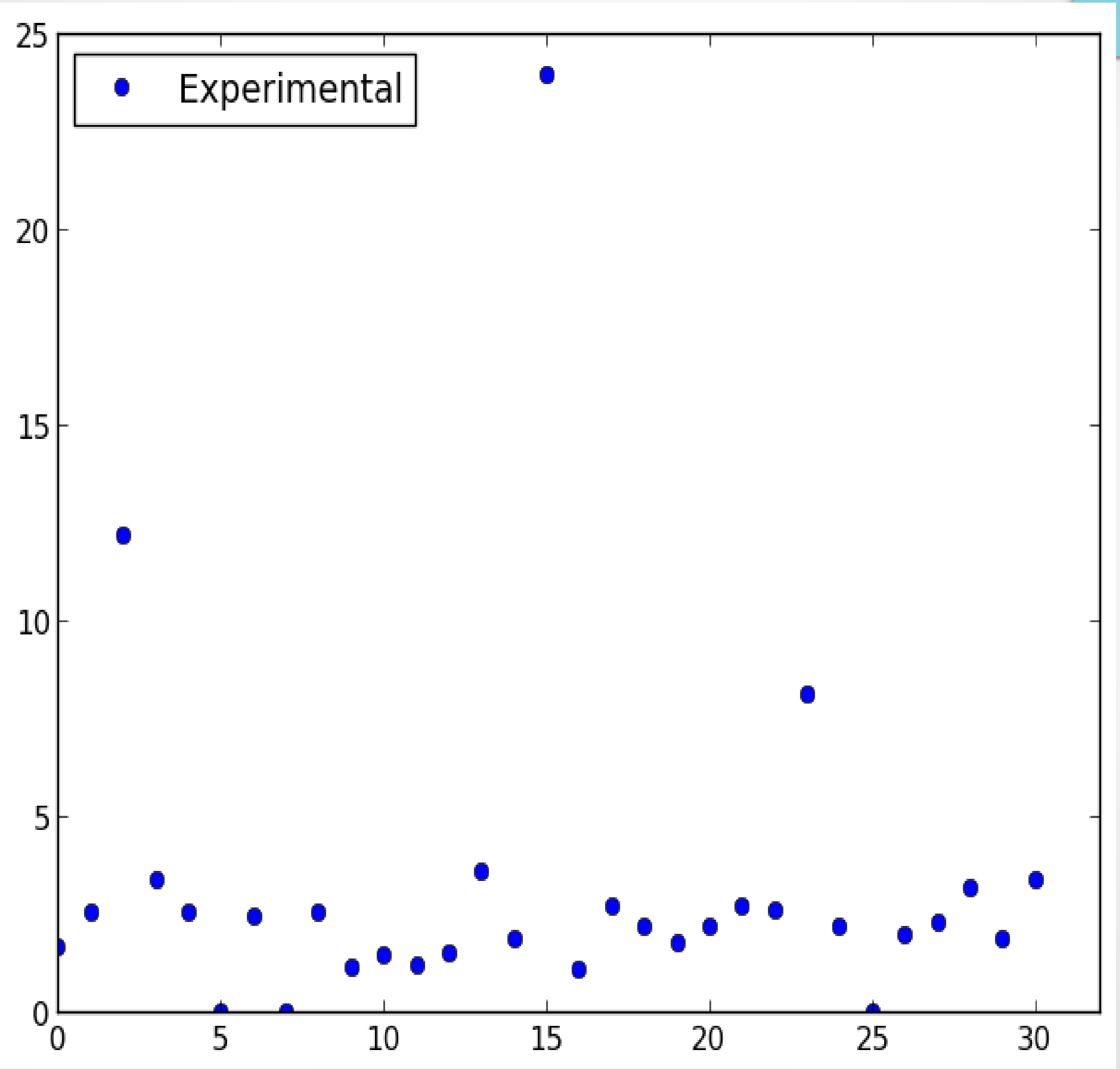
Example Domain – Processing Time

- Transaction processing time (msecs.) of XYZ:

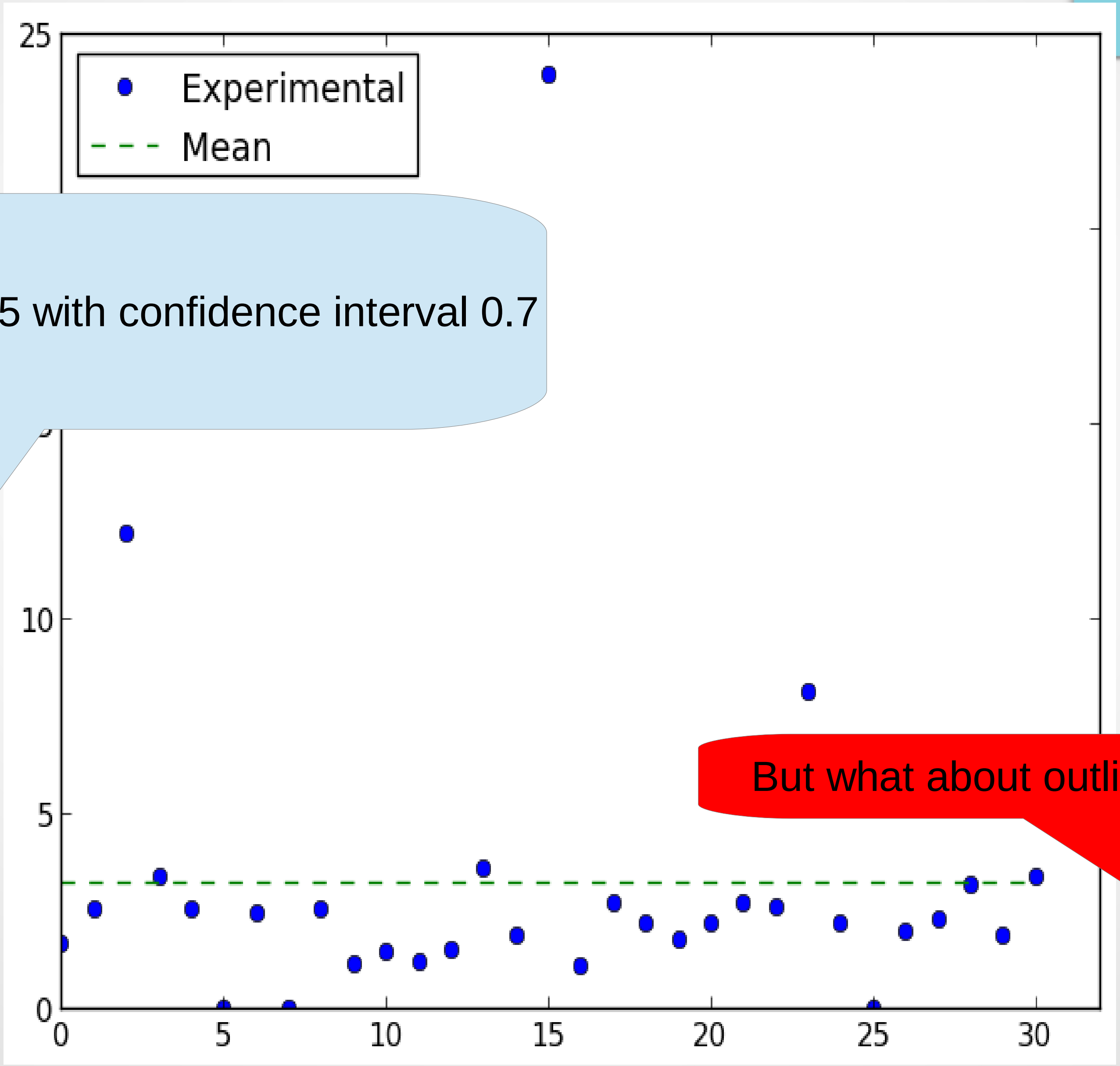
```
>>> t = [1.71, 2.55, 12.23, 3.42, 2.58, 0.03, 2.46, 0.01, 2.56,
1.17, 1.46, 1.22, 1.51, 3.6, 1.9, 23.99, 1.12, 2.73, 2.21,
1.81, 2.22, 2.73, 2.63, 8.13, 2.23, 0.00, 2.0, 2.34, 3.2, 1.9,
3.4]
```

- What is the “typical” processing time?

Example Domain – Processing Time



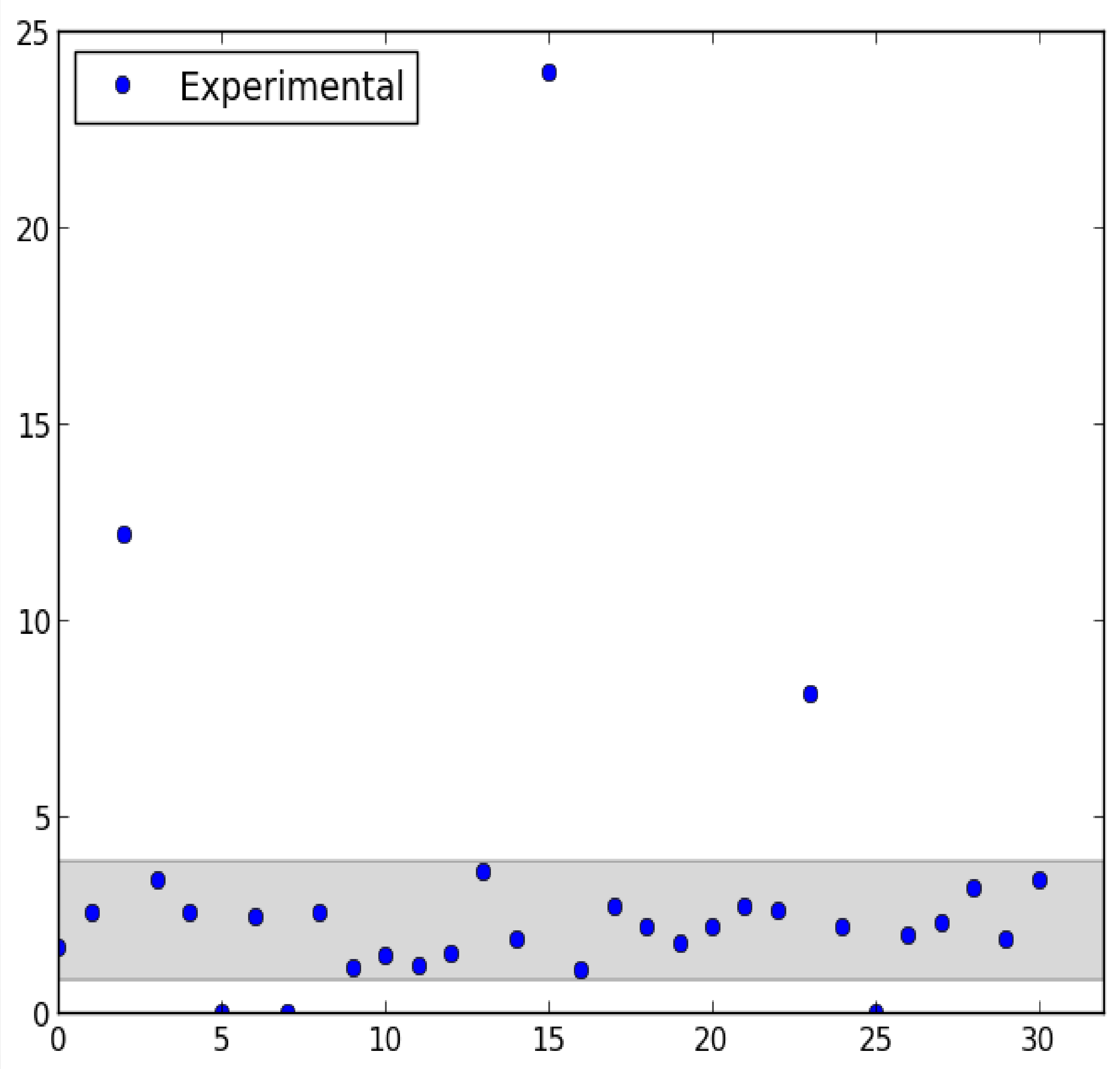
Example Domain And Model - Mean



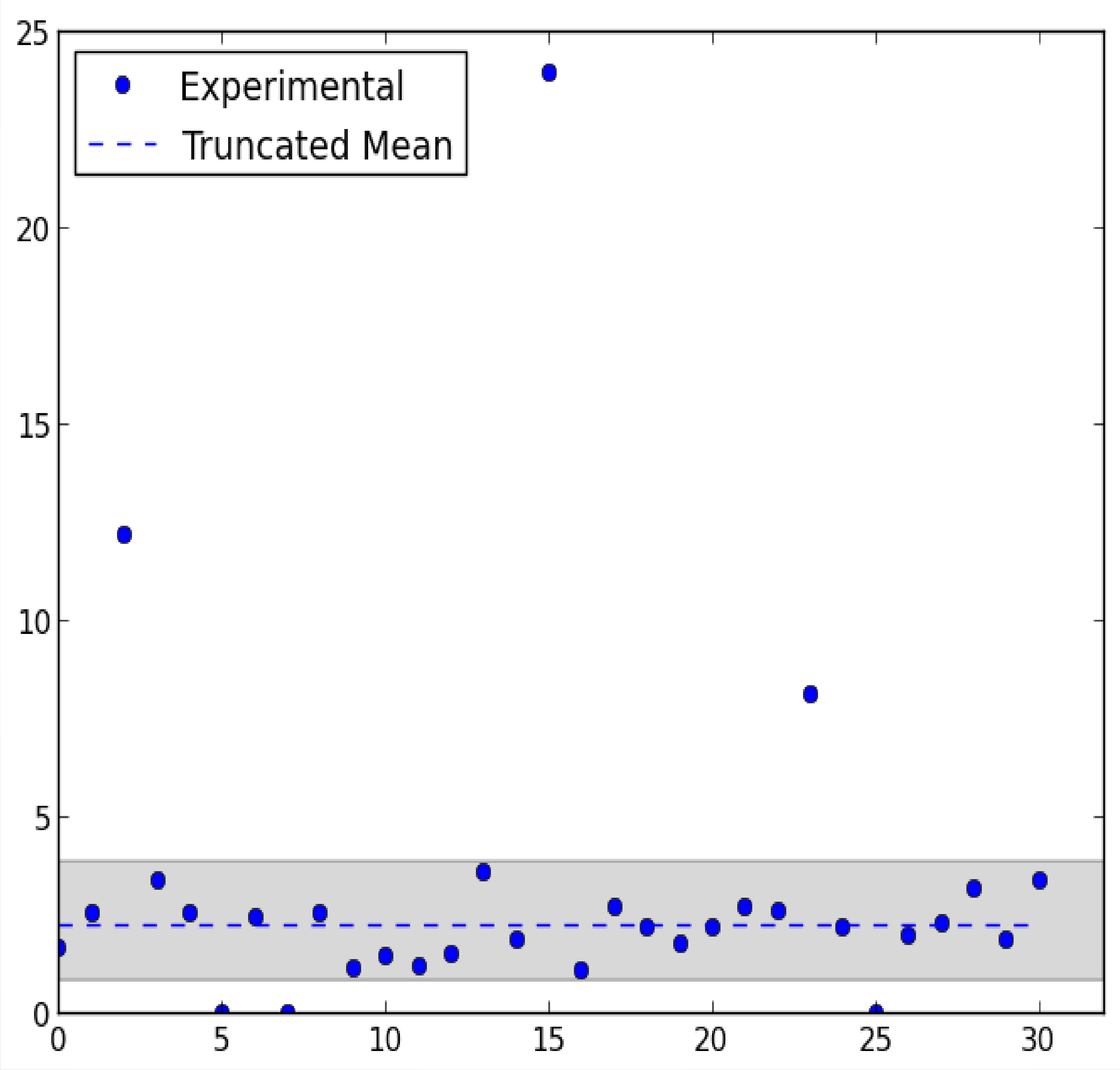
The mean is 3.25 with confidence interval 0.7

But what about outliers?

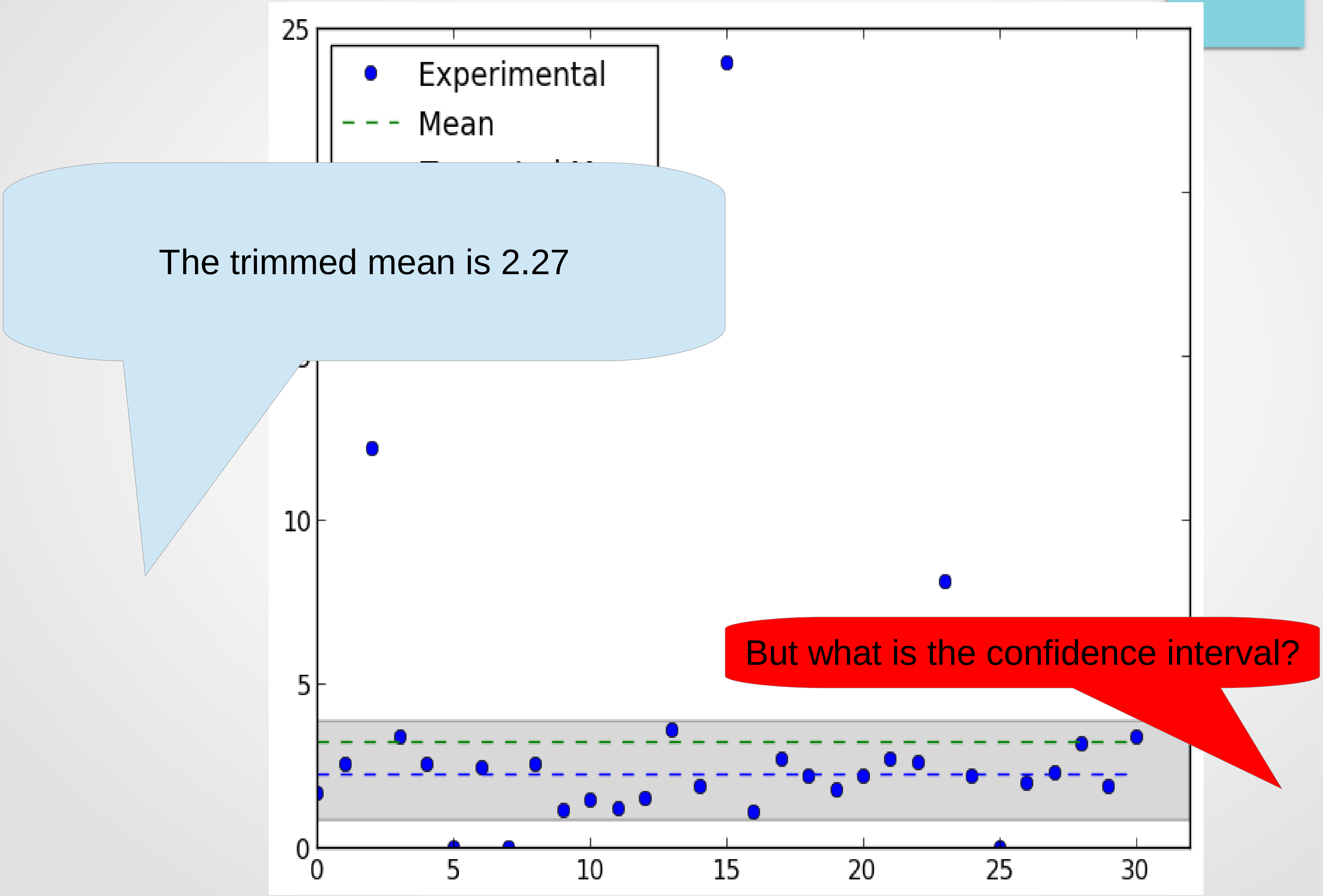
Example Domain And Model – Trimmed Mean



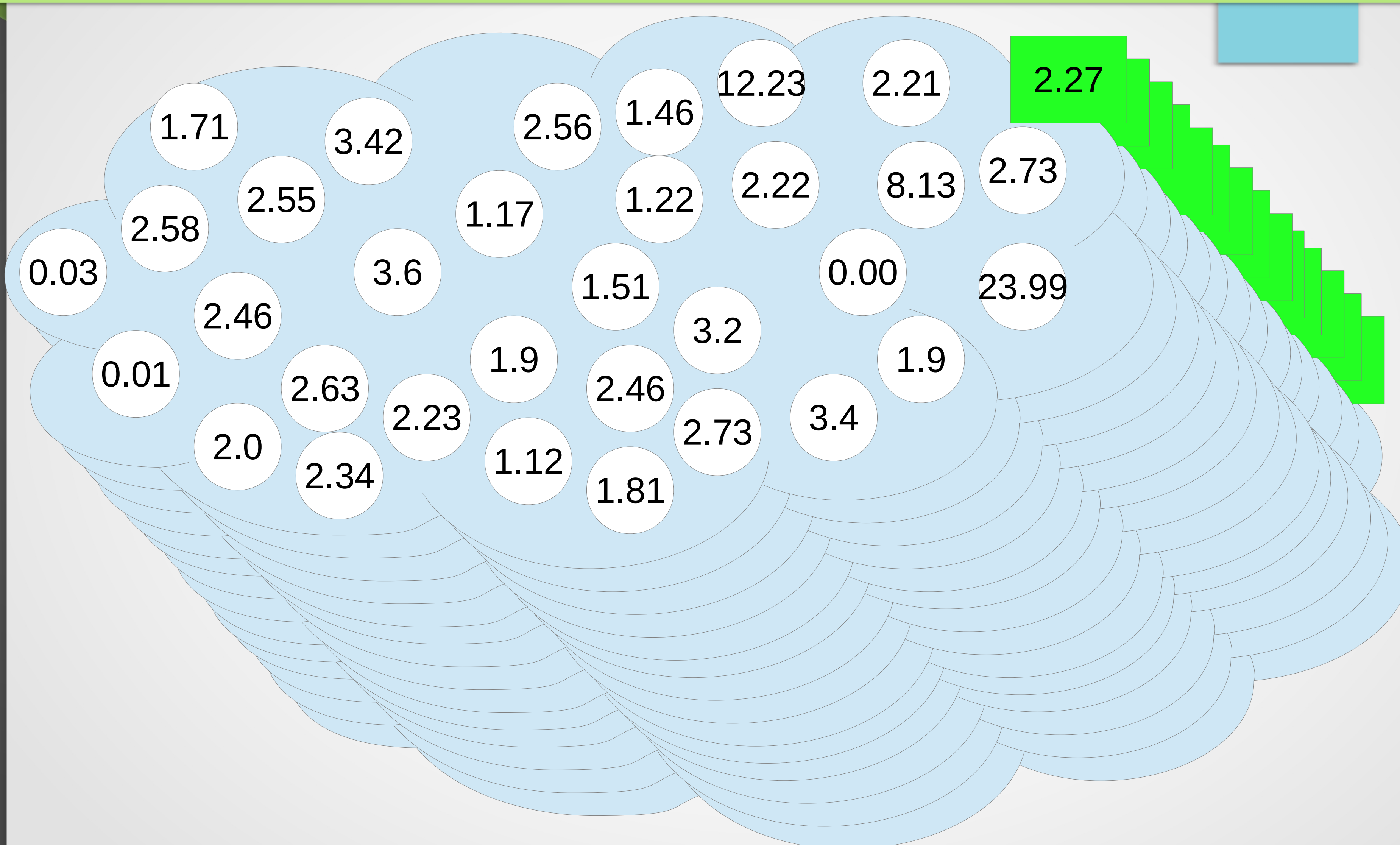
Example Domain And Model – Trimmed Mean



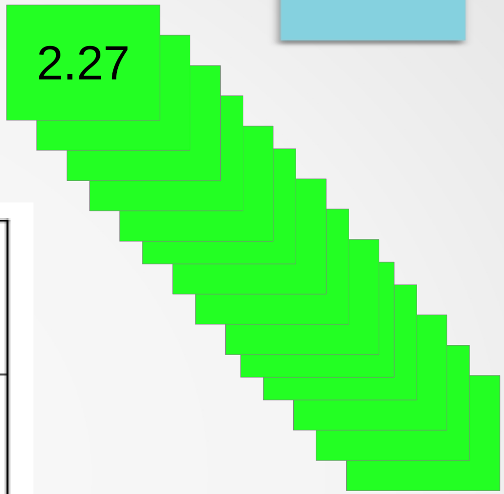
The Problem – Model Assessment

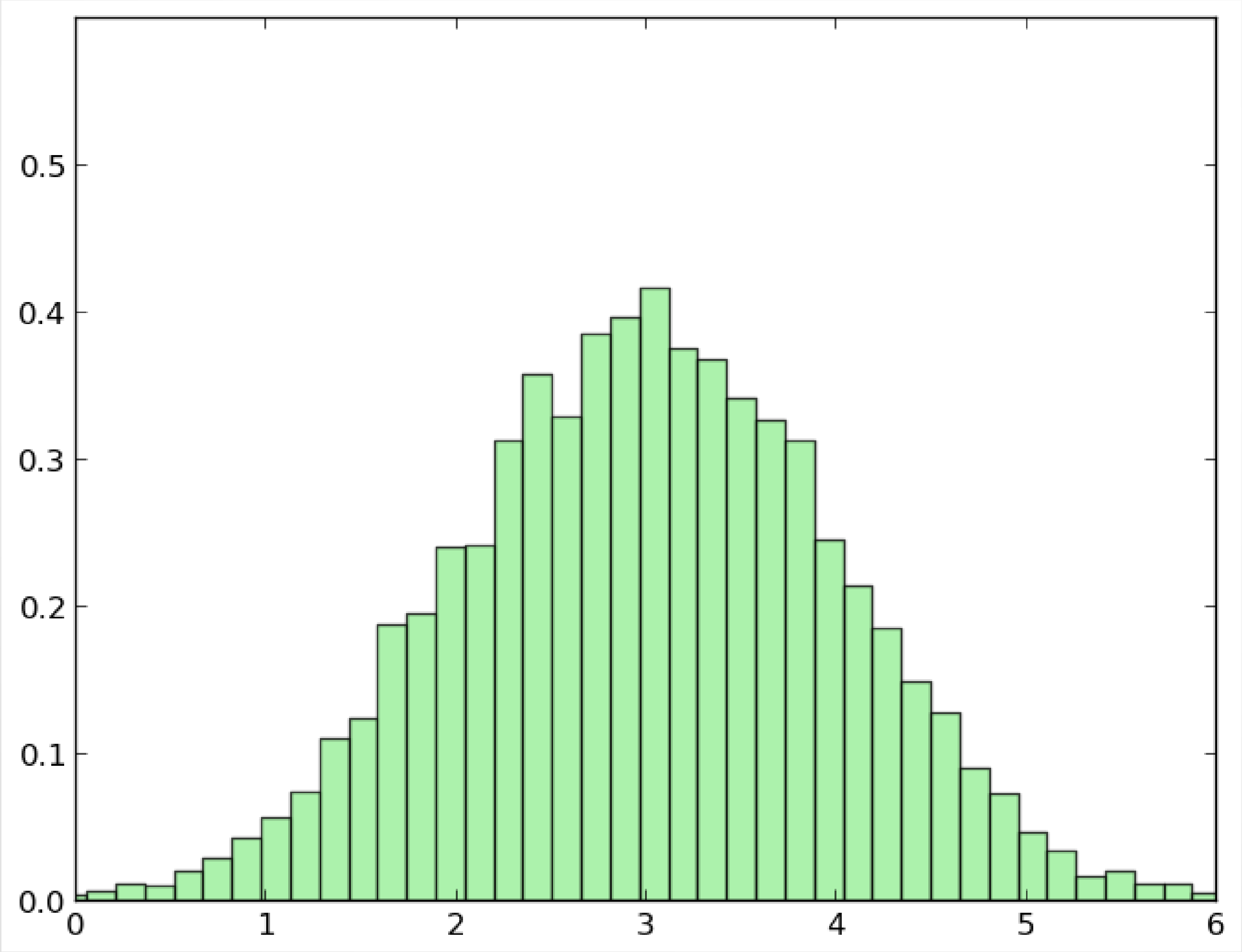


The Problem – Model Assessment



The Problem – Model Assessment



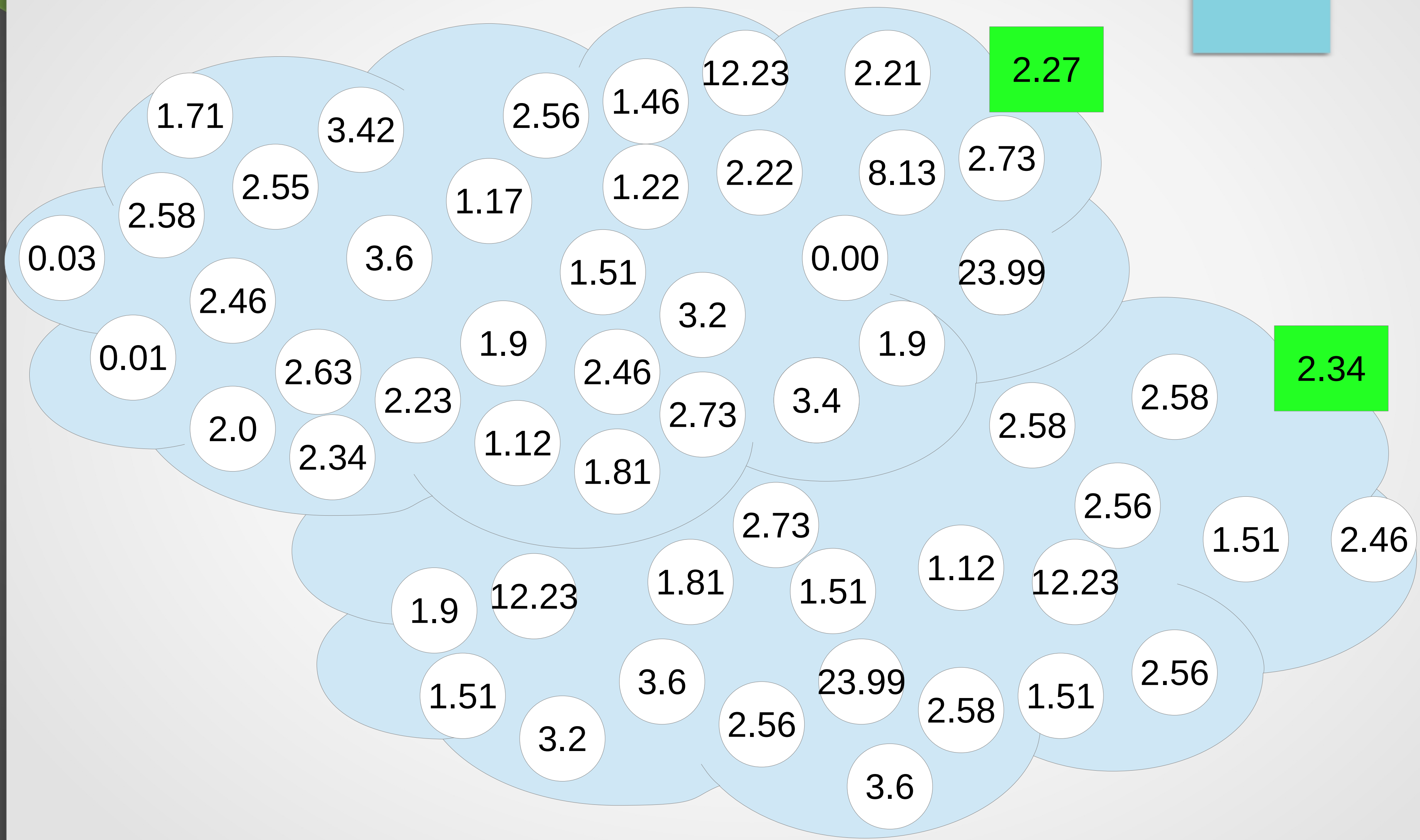


The Solution – Observation

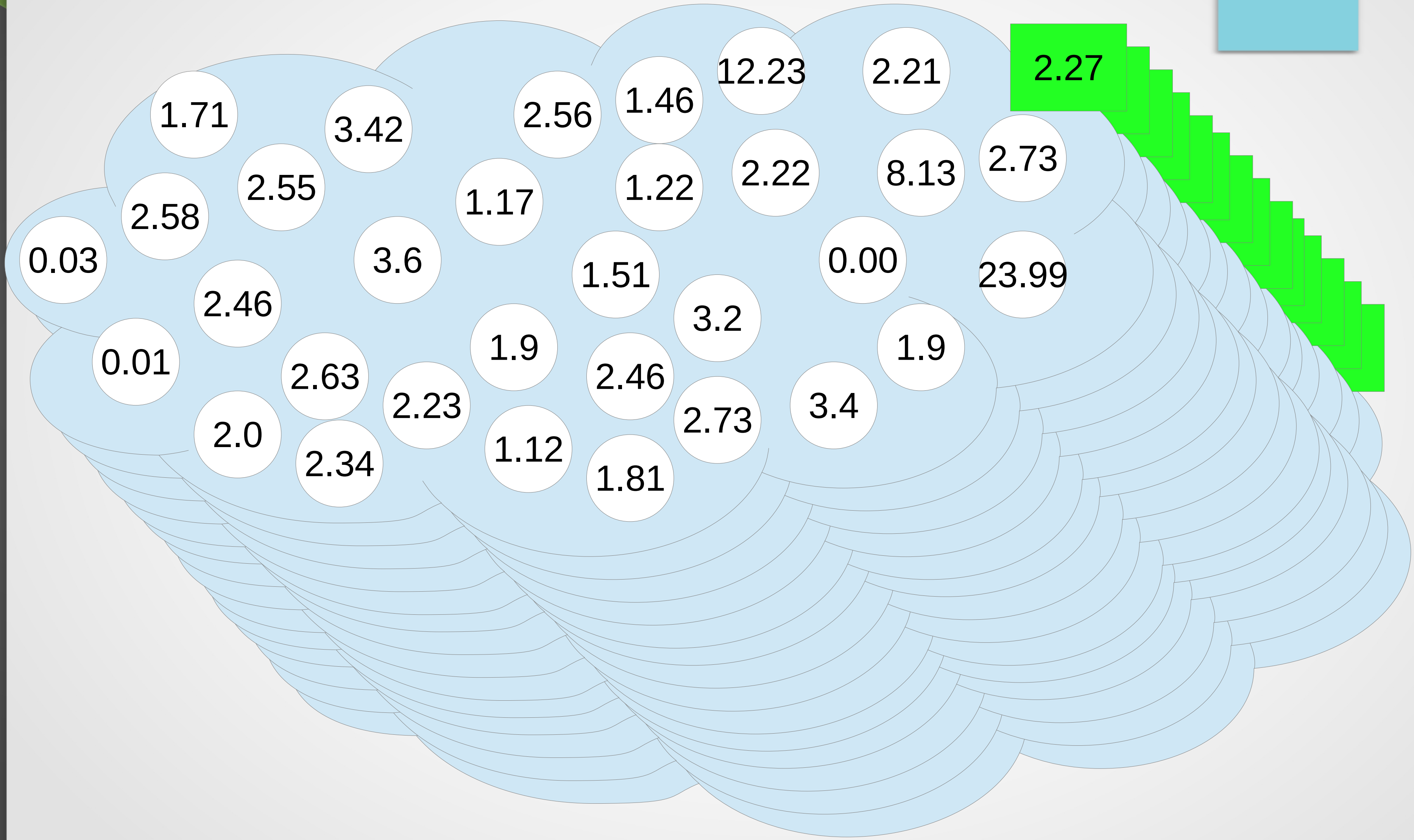
Getting Python To Learn Fron Only Parts Of Your Data

Create new datasets by random Sampling (with replacement)

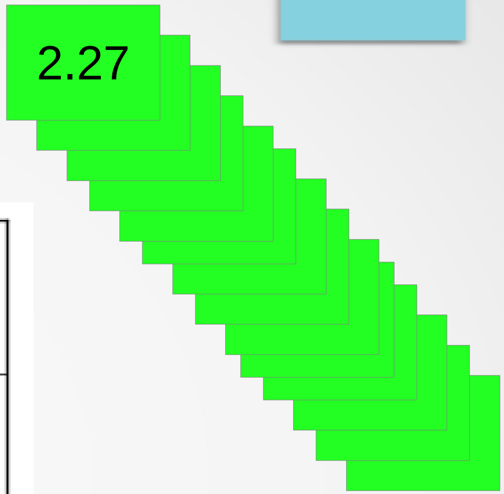
The Solution – Bootstrapping

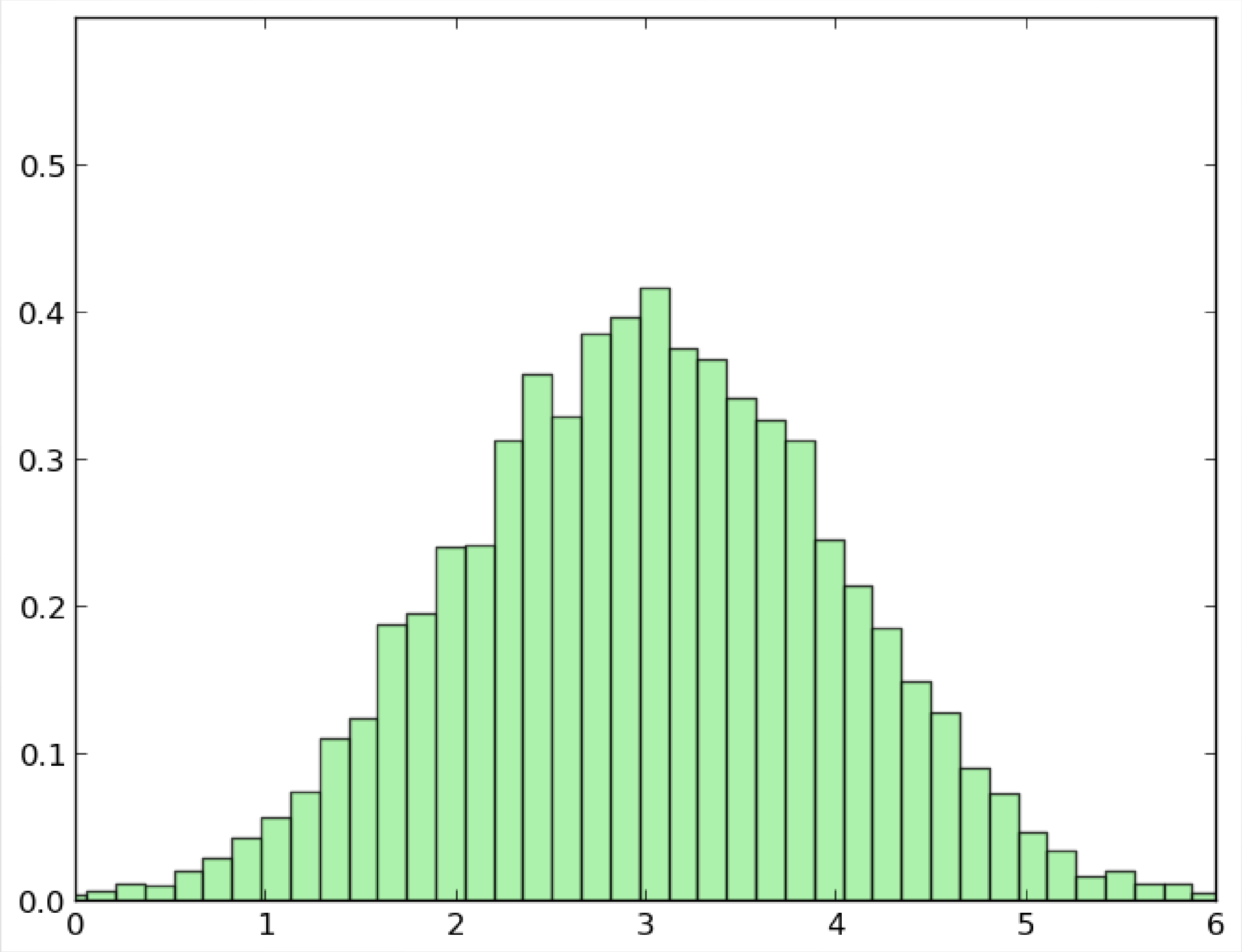


The Solution – Bootstrapping



The Solution – Bootstrapping





Let's See It In Code...

```
>>> from scipy.stats.mstats import *
>>> from scikits.bootstrap.bootstrap import *
>>>
>>> t = [1.71, 2.55, 12.23, 3.42, 2.58, 0.03, 2.46, 0.01, 2.56,
1.17, 1.46, 1.22, 1.51, 3.6, 1.9, 23.99, 1.12, 2.73, 2.21, 1.81,
2.22, 2.73, 2.63, 8.13, 2.23, 0.00, 2.0, 2.34, 3.2, 1.9, 3.4]
>>>
>>> print mean(t), trimmed_mean(t)
3.25967741935 2.2664
>>>
>>> print ci(t, trimmed_mean)
[ 1.7604  3.2816]
```

Outline

- Introduction
- Model Selection And Cross Validation
- Model Assessment And The Bootstrap
- Conclusions

Conclusions

- Computers & Python availability implies that:
  - We are faced with varied, configurable prediction techniques
  - We are unconstrained by pen-and-paper-only statistical methods
- We can use Python's scientific libraries to ad
  - Cross validation
  - strapp

Thanks!

- Thank you for your time!