CSE 519: Data Science

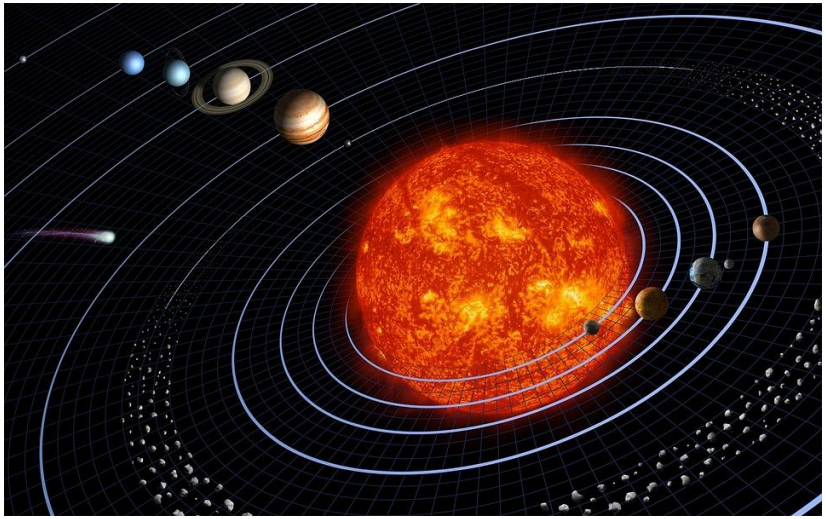
Steven Skiena
Stony Brook University

Lecture 4: Python for Data Science II

Lecture Goals

- Introduce “Learning from data” paradigm
- Focus on one learning model in depth
- Not a course on Machine Learning.
- Machine Learning in Python

On the orbits of planets



What are the laws governing the motions of planets?

Planetary Observations

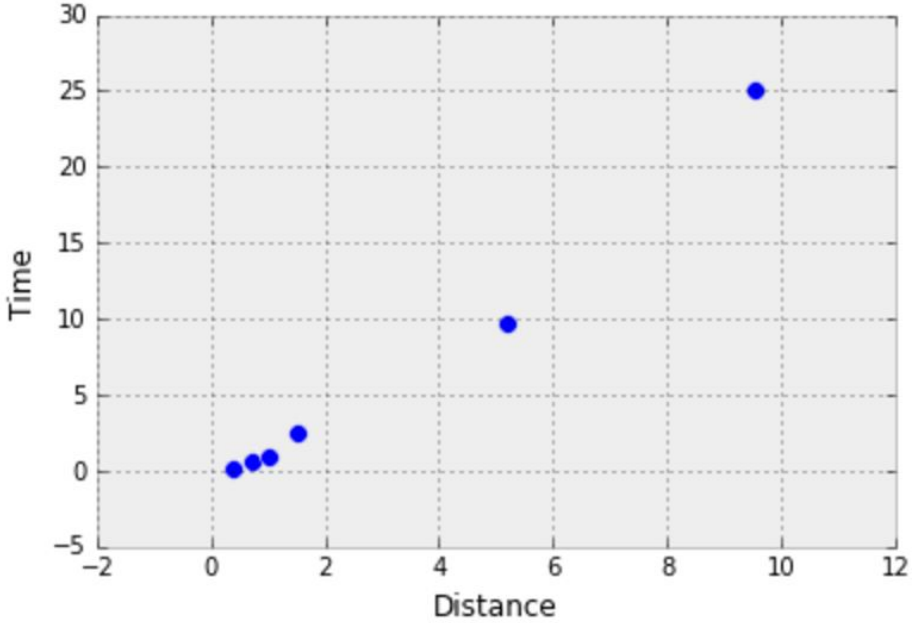
- It is the year 1609, and Newton has not even been born
- Astronomers through several observations have measured the distance of planets from the Sun and the planet’s period.

Observations

Planet	Distance (R) (AU)	Period (T) (Earth years)
Mercury	0.39	0.161
Venus	0.72	0.537
Earth	1.00	1.00
Mars	1.52	2.459
Jupiter	5.20	9.727
Saturn	9.54	25.04

Can this be used to derive a law?

Visualizing the data



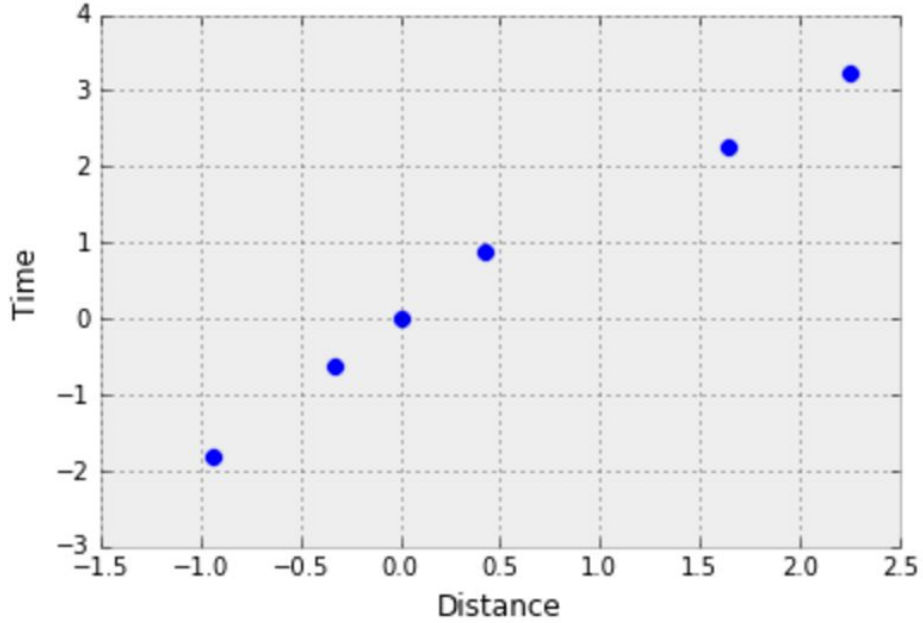
Does not look like a straight line!

Napiers Logarithms

- Napiers had just invented logarithms (around 1590). A manuscript was published in 1614

- What if we plot log(Time) versus log(Distance)?

Visualizing data - Log Log plot

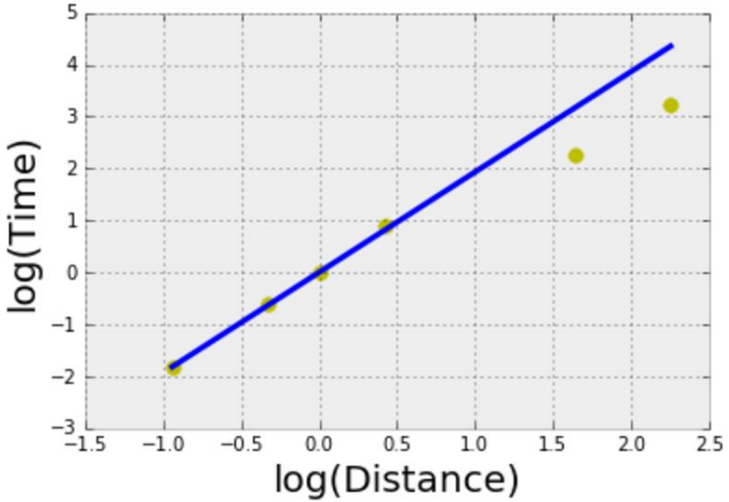


Looks much more like a line! Note observations may have experimental errors !

Challenge

- A straight line is defined by exactly 2 points
- Grand Challenge:
  - But we have more points than required! (Big Data!)
  - How to draw a line now?
  - Choosing any 2 points and drawing a line is not going to explain other planetary observations properly.

Using only 2 points



Does not explain Jupiter and Saturn's observations!

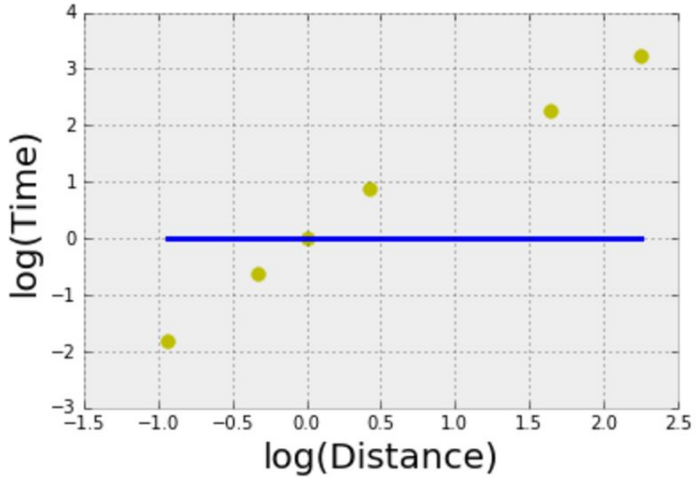
Least Squares Problem- Gauss

- Instead of having to choose 2 points to draw a line what if
  - Minimize total error over all observations
  - Let line be defined by slope ‘a’ and intercept ‘b’
  - Minimize $\boldsymbol{\Sigma}\ (y_i-(ax_i+b))^2$
- Not trivial to solve!
- Can solve using calculus!

Illustration

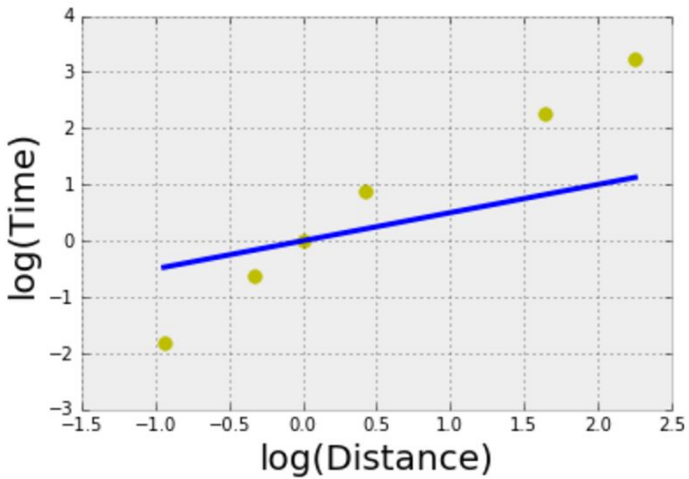
- A = 0 , B = 0

Slope = 0.0, Bias = 0.0
Residual sum of squares: 3.34

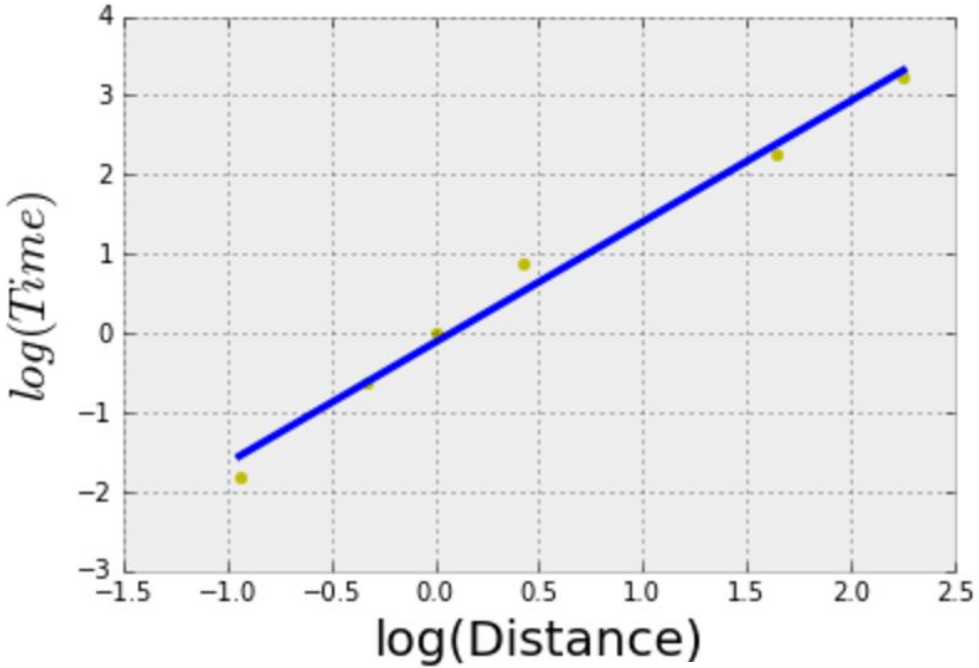


A=0.5, B = 0

Slope = 0.5, Bias = 0.0
Residual sum of squares: 1.50



Using Least Squares Solution



- A=1.5,B =-0.1

```
Residual sum of squares: 0.04
```

Sample code: Least Squares fit

Linear Model

In [101]:
```
from sklearn.linear_model import LinearRegression
```

- bias term is called also `intercept`.
- If the features vary significantly in range, it pays off to `normalize` the data as a preprocessing step.

In [107]:
```
model = LinearRegression()
model = model.fit(X1.reshape(-1,1), y1)
predictions = model.predict(X1.reshape(-1,1))
print("Model is trained with the following params: {}".format(model.get_params()))
```

```
Model is trained with the following params: {'copy_X': True, 'normalize': False, 'n_jobs': 1, 'fit_intercept': True}
```

In [110]:
```
print("Slope = {}, Bias = {}".format(model.coef_[0], model.intercept_))
# The mean square error
print("Residual sum of squares: %.2f"
      % np.mean((predictions - y1) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % model.score(X1.reshape(-1,1), y1))

# Plot outputs
plt.scatter(X1, y1,  color='y')
plt.plot(X1, predictions, color='blue', linewidth=3)
plt.xlabel("log(Distance)", fontsize=20); plt.ylabel("$log(Time)$", fontsize=20)
#_ = plt.title("$y\,\vs\,\X_2$", fontsize=20)
```

```
Slope = 1.52334323195, Bias = -0.117734147463
Residual sum of squares: 0.04
Variance score: 0.99
```

Analyzing the solution

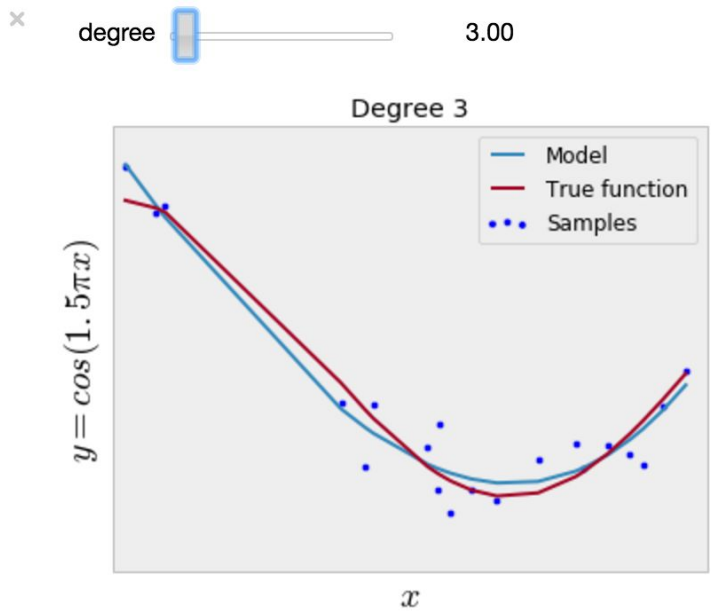
- Our least squares model suggests that
  - $\log T \propto (1.5) \log R$
  - $2 \log T \propto 3 \log R$
- This implies: $T^2 \propto R^3$

★ We have just derived Kepler's Third Law!
★ We just learned Linear Regression!

Machine Learning

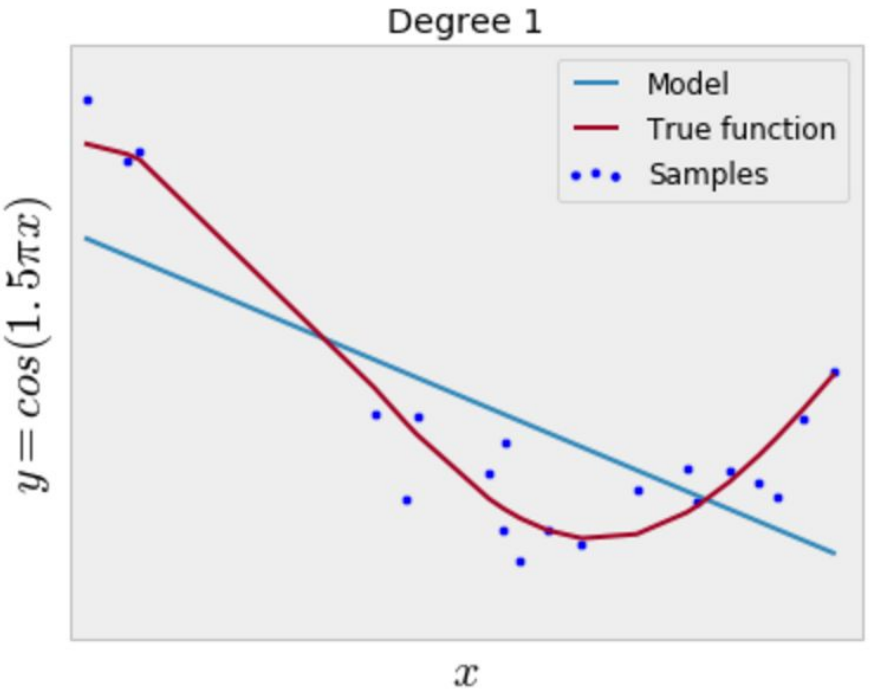
- Instead of restricting ourselves to linear boundaries, we generalize to (a) more complex functions, and (b) different types of target variables
  - Polynomial
  - Non-linear functions (Exponential)
  - Discrete target variables
- Machine learning is just Function Approximation!

Example

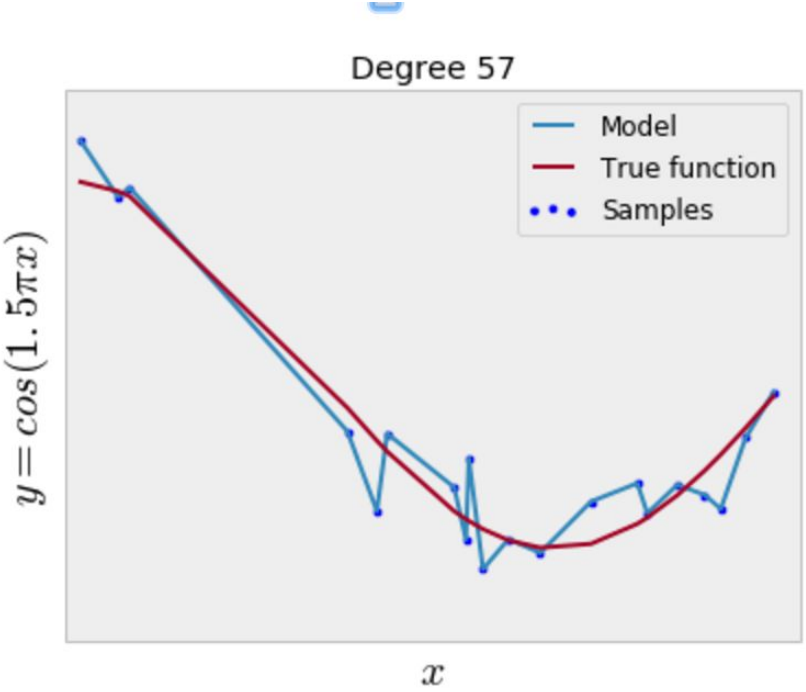


- Approximate a function using a cubic polynomial!
- Be aware of overfitting, underfitting

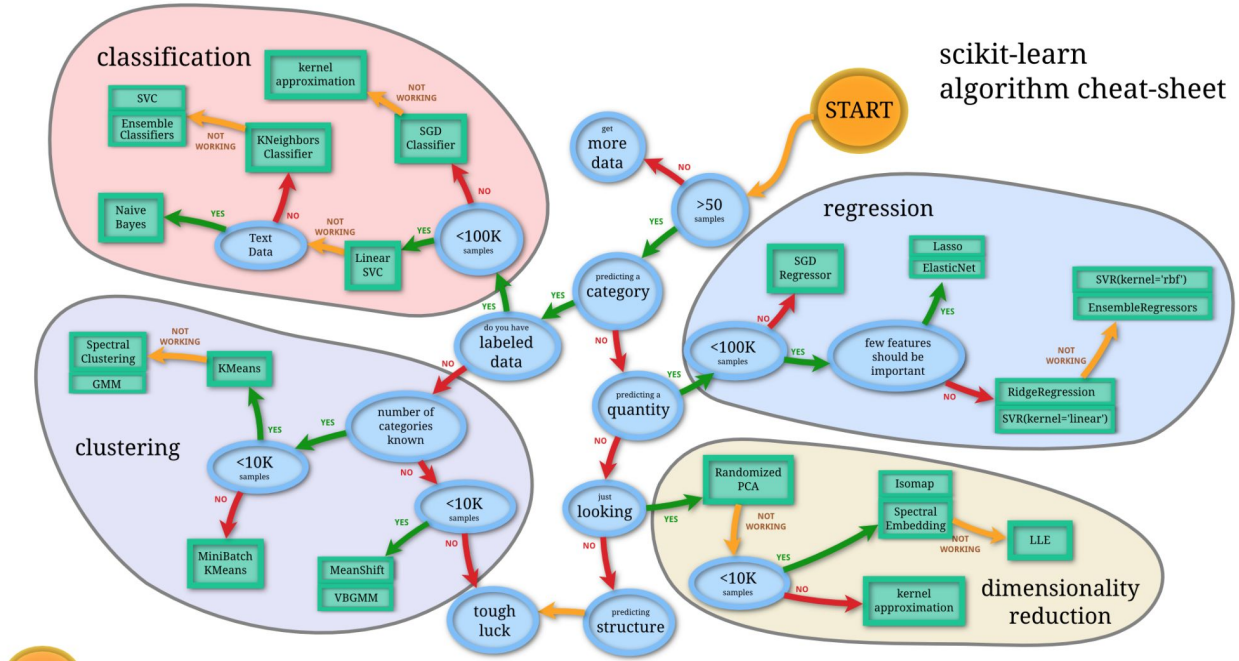
Example: Underfit model



Example: Overfit model



ML Cheat Sheet using Scikit Learn



Machine Learning in Python

- Scikit-learn is the definitive toolkit for machine learning in Python
- Supports regression, classification, clustering and dimensionality reduction
- Many models: SVM, Linear Regression, Logistic Regression
- Catch: Understand how these algorithms work before you apply them

Notebooks

All the code is available here:

https://github.com/vivekSCK/data_science_course