Last Lecture

① Final Exam

(i) Friday -> 9:00am - ....

(ii) Two sections

a) Zybooks programs (3-4)

b) Data Science Problems (2)

-> Send to me via email.!!

② Basic Plotting - Full Example.

③ Linear Regression - Complex Data Example

Colab with numpy & scikit-learn

(a) [illegible]

(b) with pandas & statsmodels

Linear Regression

Complex Data :

x1	x2	x3	x4	y

Example : y → housing prices

x1 → sq. footage

x2 → zip code

x3 → # of bedrooms

→ # of bathrooms

x 4 →

We would like, in the end, to determine an equation like:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4$$

i.e. an equation which is linear in the possible factors.

N.B. To visualize this, we would need to make a FIVE dimensional plot !!!

→ obviously, impossible !!

This is why this is a hard problem to solve and understand.

Key Idea: We need to

determine

$$\begin{aligned} \beta_0 &\pm \delta\beta_0 \\ \beta_1 &\pm \delta\beta_1 \\ &\vdots \\ \beta_n &\pm \delta\beta_n \end{aligned}$$

If some $\beta_k \pm \delta\beta_k$ is consistent with ZERO, then y does not depend on this factor.

Example: maybe housing prices do not depend on the number of bedrooms.

Key Idea 2:

The "Explore" stage of this analysis is important!! We should first plot:

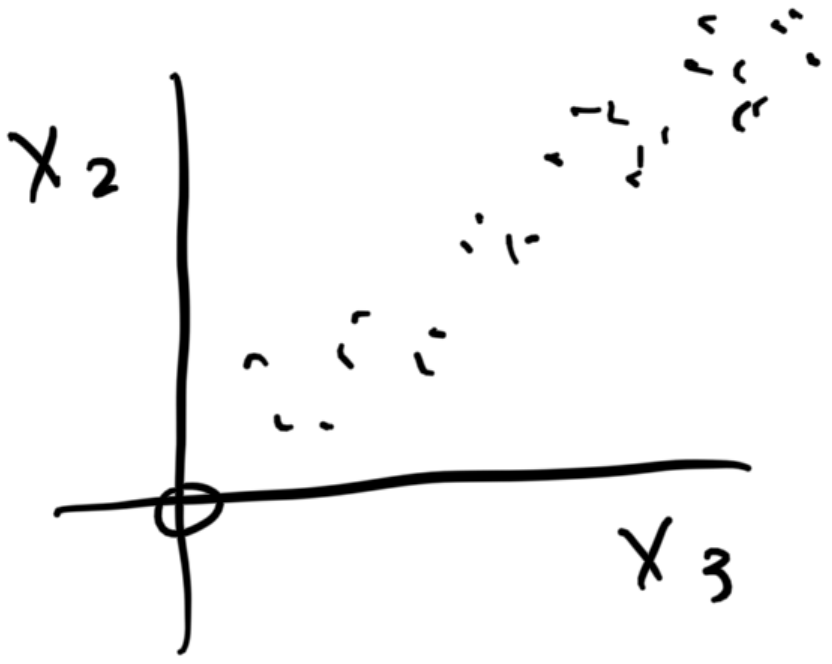
$$y \text{ vs. } x_1, \quad y \text{ vs. } x_2, \quad \cdots$$

and some "basic"

to understand the behavior of the data.

Key Idea # 3 :

We are ignoring correlations.



This can cause misleading results.

N.B. We need to be cognizant of Correlations vs. Causation

Let's start with a simpler example, so that we can understand the engine of ... in Python.

Linear regression

Step 1 : Get the data into appropriate data structures.

TWO CHOICES :

(1) numpy arrays

(2) Pandas data frames.

(i) numpy arrays.

Aside on Linear Algebra.

Suppose we have :

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

x_1	x_2	y

N data points.

We can write:

$$y_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{21}$$
$$y_2 = \beta_0 + \beta_1 x_{12} + \beta_2 x_{22}$$
$$y_3 = \beta_0 + \beta_1 x_{13} + \beta_2 x_{23}$$
$$\vdots$$
$$y_N = \beta_0 + \beta_1 x_{1N} + \beta_2 x_{2N}$$

Transform this to a matrix equation:

$$\begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \beta_0 \\ \vdots \\ \beta_0 \end{pmatrix} + \begin{pmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \\ x_{13} & x_{23} \\ \vdots & \vdots \\ x_{1N} & x_{2N} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

easy to determine from average of y data

$$\begin{pmatrix} y_1 - \beta_0 \\ y_2 - \beta_0 \\ \vdots \\ y_N - \beta_0 \end{pmatrix} = \begin{pmatrix} x_{11} & x_{21} \\ 1 & 1 \\ 1 & 1 \\ \vdots & \vdots \\ x_{1N} & x_{2N} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

$$\therefore \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \underbrace{\begin{pmatrix} x_{11} & x_{21} \\ 1 & 1 \\ 1 & 1 \\ \vdots & \vdots \\ x_{1N} & x_{2N} \end{pmatrix}^{-1}}_{\text{inverse!!}} \begin{pmatrix} y_1 - \beta_0 \\ \vdots \\ y_N - \beta_0 \end{pmatrix}$$

This is, in a broad sense, how linear regression algorithms work!!

For our purposes, the main point is that the:

> X data should be a 2-D Matrix

Y data should be a 1-D Matrix

Example :

xi = [1, 2, 3, 4, 5]

yi = [1.1, 2.1, 2.9, 3.9, 5.1]

X = numpy.array(xi).reshape((-1,1))

y = numpy.array(yi)

print(X)

[[1] [2] [3] [4] [5]]

[[1 [1 [1 [1 [1

print(y)

[1.1 2.1 2.9 3.9 5.1]

N.B. numpy arrays → printed with no commas!!
(as opposed to Python lists)

Step 2: Clean the data

→ plot y vs. x

→ look for bad/missing data.

→ think about appropriate relationships.