Python Data Products
Course 2: Design thinking and predictive pipelines

Lecture: Feature transformations
In this lecture we will...

- Demonstrate the use of **transformations** to incorporate non-linear functions into linear models
We’ve previously seen simple examples of regression models such as Weight vs. Height:
Motivating example

We’ve previously seen simple examples of regression models such as Weight vs. Height:

• A linear relationship is probably *okay* for modeling this data, and in practice we’d often get away with using this type of model
• But, it certainly makes some assumptions that aren’t totally justified
• How can we fit more suitable, or more general functions
Motivating example

How *should* the right model look for weight vs. height?

- A linear function?
- A quadratic or polynomial function?
- An asymptotic function?
- etc.
Fitting complex functions

Let’s start with a polynomial function (e.g. a cubic function):

\[
\text{weight} = \theta_0 + \theta_1 \times \text{height} + \theta_2 \times \text{height}^2 + \theta_3 \times \text{height}^3
\]

- Note that this is perfectly straightforward: the model still takes the form

\[
\text{weight} = \theta \cdot x
\]

- We just need to use the feature vector

\[
x = [1, \text{height}, \text{height}^2, \text{height}^3]
\]
Fitting complex functions

Note that we can use the same approach to fit arbitrary functions of the features! E.g.:

\[
\text{weight} = \theta_0 + \theta_1 \times \text{height} + \theta_2 \times \text{height}^2 + \theta_3 \exp(\text{height}) + \theta_4 \sin(\text{height})
\]

- We can perform arbitrary combinations of the **features** and the model will still be linear in the **parameters** (theta):

\[
\text{weight} = \theta \cdot x
\]
Fitting complex functions

The same approach would **not** work if we wanted to transform the parameters:

\[
\text{weight} = \theta_0 + \theta_1 \times \text{height} + \theta_2^2 \times \text{height} + \sigma(\theta_3) \times \text{height}
\]

- The **linear** models we’ve seen so far do not support these types of transformations (i.e., they need to be linear in their parameters)
- There **are** alternative models that support non-linear transformations of parameters, e.g. neural networks
Summary of concepts

- Showed how to apply arbitrary transformations to features in a linear model
- Further explained the restrictions and assumptions of **linear models**

On your own...

- Extend our previous code (on pm2.5 levels vs. air temperature) to handle simple polynomial functions