Transformations

In the notebook on non-Homogeneous data (Becoming_a_successful_Data_Scientist.ipynb) we hopefully have motivated

- the need to process your features (and sometimes target)
- from "raw" to "synthesized" form

This process is called **feature engineering**.

Knowing how to perform feature engineering is a key skill of a Data Scientist.

We explore this topic in more depth in this module.

We will illustrate with transformations on the datasets from the "non homogeneous" examples.

Transformation: basics

Before diving into the many different types of transformations

- we want to establish some basics

Let's review the Mechanics of transformations (Transformations_Mechanics.ipynb)

For a refresher on implementation in `sklearn`

- Revisit our previous module Coding transformations in `sklearn` (Transformations_Pipelines.ipynb))

The why's of transformations

We illustrate several useful types of transformations below.

This is organized more as a "case study" than a taxonomy.

- Many transformations have multiple uses, making a hard taxonomy difficult

Missing features

Sometimes, your examples have all the "information" you need, but in the wrong form.

Creating new "synthetic" features from raw features is one way of making this information available to the model.

But sometimes, the problem is the *absence* of a relevant feature.

Missing numeric features

Let's visit the notebook section on adding a missing numeric feature (Transformations_Missing_Features.ipynb#Transformation-to-add-a-%22missing%22-numeric-feature)

Missing categorical features

In the "curvy data" example: the missing feature was numeric.

Sometimes, the missing feature is categorical.

This is the case of our non-homogeneous data resulting from sampling at different points in time

Let's see how adding a categorical feature (Transformations_Missing_Features.ipynb#Transformation-to-add-a-%22missing%22-categorical-feature) that distinguishes between groups solves this issue.

Cross features

We illustrated how to create *binary categorical indicator* features to identify groups

- OHE creates a binary indicator feature for each class within a Category

But sometimes, group membership may involves a *compound* condition.

Let's examine how to create features to express compound conditions (Transformations_Missing_Features.ipynb#Cross-features)

Scaling

There is a class of transformations that alter the *scale* (magnitude) of features or targets.

In the Recipe for ML, Scaling is treated as separate from the other transformations.

Perhaps this is because scaling is sometimes performed

- *Not* strictly because of the relationship between target and features
- But because of the mathematics of the *loss function*

Let's visit the notebook Transformations: scaling (Transformations_Scaling.ipynb)

Normalization: heterogeneous data

Our introduction to Transformations focused on several tasks whose data appeared to come from different distributions.

Sometimes, we can make the examples appear to come from a common distribution by scaling transformations.

We will refer to such scaling transformations as a type of *Normalization* of examples.

We devote a separate notebook to Normalization Transformation (Transformations_Normalization.ipynb).

Other transformations

For other transformations see the notebook Other Transformations (Transformations_Other.ipynb)

In [4]:
```
print("Done")
```

```
Done
```