The mechanics of transformations

We briefly introduced transformations in the overview of the Prepare the data step of the Recipe for ML (Prepare_data_Overview.ipynb).

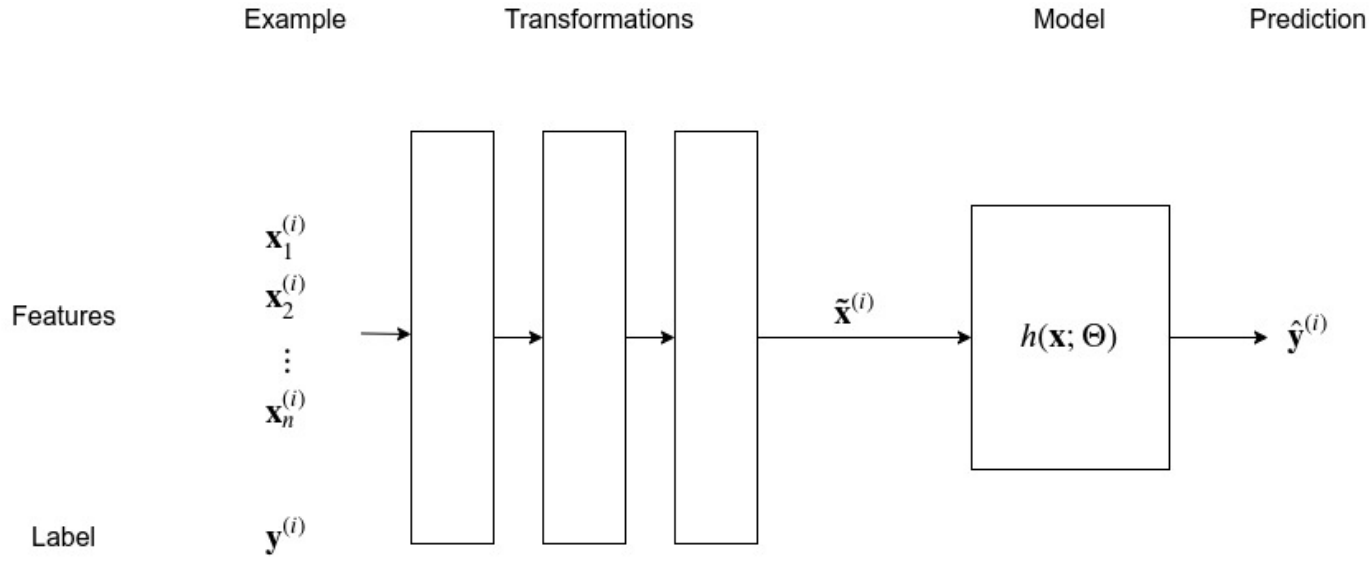
We recap the key points:

Fitting transformations

To review: transformations (feature engineering)

- takes an example: vector $\mathbf{x}^{(\mathbf{i})}$ with n features
- produces a new vector $\tilde{\mathbf{x}}^{(\mathbf{i})}$, with n' features

We ultimately fit the model with the transformed *training* examples.



- MIssing data inputation
- Standardization
- Discretization
- Categorical variable encoding

Transformations often have their own parameters $\Theta_{\text{transform}}$ that is separate from the Θ parameters of the model.

For example: a "missing data transformation" -That substitutes the mean/median (over the training examples) of a feature j for a missing value.

- Is "fit" (or "trained") by giving it all the training examples
- The median for each feature is recorded in $\Theta_{\text{transform}}$

Transformations are applied to both training and test examples

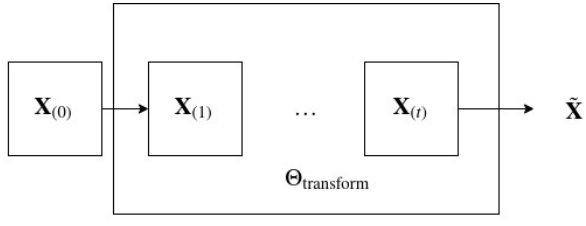
The domain of the prediction model h

- is the domain of the transformed $\tilde{\mathbf{x}}$
- **not** the domain of the original examples $\mathbf{x}$

So before any example, such as a test example, is fed through the model, it must be transformed.

- Transformed using the $\Theta_{\text{transform}}$ obtained by fitting training examples
- We **do not** refit $\Theta_{\text{transform}}$ on test examples !

Feature engineering: fit, then transform



Train

$\text{fit} : \mathbf{X}_{(0)} \mapsto \Theta_{\text{transform}}$

$\text{transform}(\mathbf{x}^{(\mathbf{i})}; \Theta_{\text{transform}}) \mapsto \tilde{\mathbf{x}}^{(i)}$

Test

$\text{transform}(\underline{\mathbf{x}}; \Theta_{\text{transform}}) \mapsto \underline{\mathbf{x}}'$

NO fitting at test time, re-use

$\Theta_{\text{transform}}$

- standardization: mean($\mathbf{X}$), std-dev($\mathbf{X}$)
- scaling: min($\mathbf{X}$), max($\mathbf{X}$)
- imputation: median($\mathbf{X}$)

Targets and features can be transformed

Although we have framed transformations as something performed on features $\mathbf{x}$, we sometimes transform the target $\mathbf{y}$

- Logistic Regression transformed the target p to log odds: $\log \frac{p}{1-p}$
    - The log odds is amenable to a linear model; the raw target is not

Inverting transformations

If we transform the target, then domain of the values predicted by the model are in the same units as the transformed targets

- Example: log odds rather than probability
- Example: you might convert a price level to a percent change
    - Your predictions are then predictions of percent change, not price

We probably want to report our predictions to our clients in the original domain of the targets.

You may need to *invert the transformation* to convert prediction $\hat{y}$ back into the same units as the original targets

Transformations in `sklearn`

`sklearn` provides an easy API with the following methods

- `fit`: set parameters of transformation; fit to training data
- `transform`: apply transformations. Do this for both train and test data
- `inverse_transform` to convert from transformed data units back to original

In [1]:
```
print("Done")
```

```
Done
```