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).

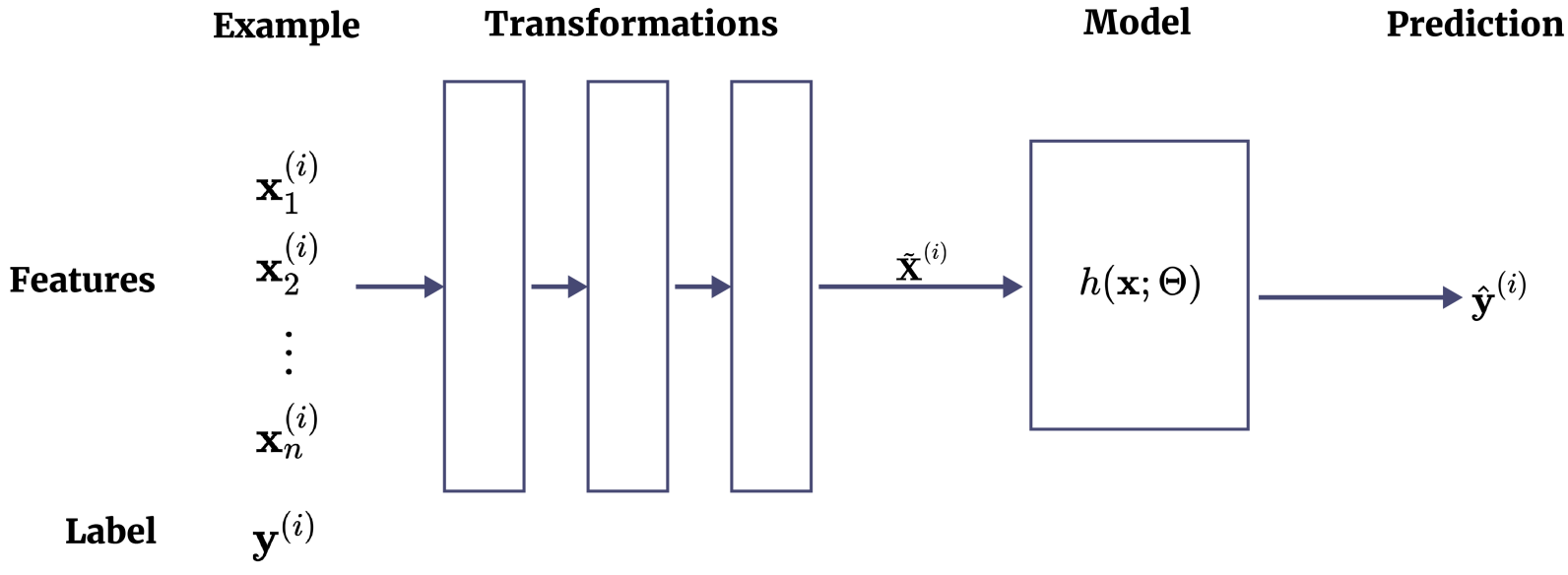
We recap the key points:

Fitting transformations

Feature engineering, or transformations

- takes an example: vector $\mathbf{x^{(i)}}$ 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 have parameters $\Theta_{\text{transform}}$ distinct from the model's parameters Θ.

- Example: Missing data imputation for a feature substitutes the mean/median feature value
- $\Theta_{\text{transform}}$ stores this value

Our prediction is thus

$$\begin{aligned} \hat{\mathbf{y}} &= h_{\Theta}(\tilde{\mathbf{x}}) \\ &= h_{\Theta}\left(T_{\Theta_{\text{transform}}}(\mathbf{x}) \right) \end{aligned}$$

Transformations can be applied to the target as well. For example

- One Hot Encoding a categorical target for a Classification task
- Scaling the target (e.g., pixel intensities from a range $[0 \ldots 255]$ to a range $[-1 \ldots +1]$)

If we transform the target $\mathbf{y}$ into new units, the predicted $\hat{\mathbf{y}}$ will also be in the new units

- If we want to report our prediction in original units
- We must be able to invert the transformation

For example:

- Logistic Regression transforms the target into Log Odds
- We want to report our prediction in terms of one class of the Categorical variable

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

Transformers in `sklearn`

A transformer in `sklearn` provides the following methods

- `fit`: using training examples, compute $\Theta_{\text{transform}}$
- `transform`: map an example $[\mathbf{x}^{(\mathbf{i})}, \mathbf{y}^{(\mathbf{i})}]$ into transformed example $[\tilde{\mathbf{x}}^{(\mathbf{i})}, \tilde{\mathbf{y}}^{(\mathbf{i})}]$
- `inverse_transform`: map a transformed example $[\tilde{\mathbf{x}}^{(\mathbf{i})}, \tilde{\mathbf{y}}^{(\mathbf{i})}]$ back to its source example $[\mathbf{x}^{(\mathbf{i})}, \mathbf{y}^{(\mathbf{i})}]$

In [3]:
```
print("Done")
```

```
Done
```