Depending on the use case, the **fit()** method can be used to learn the parameters of the dataset, while the **transform()** method applies the data transform based on the learned parameters to the same dataset and also to the test or validation datasets before modeling. Also, the **fit_transform()** method can be used to learn and apply the transformation to the same dataset in a one-off fashion. Data transformation packages are found in the **sklearn.preprocessing** package.

This section will cover some critical transformation for numeric and categorical variables. They include

- Data rescaling
- Standardization
- Normalization
- Binarization
- Encoding categorical variables
- Input missing data
- Generating higher-order polynomial features

Data Rescaling

It is often the case that the features of the dataset contain data with different scales. In other words, the data in column A can be in the range of 1–5, while the data in column B is in the range of 1000–9000. This different scale for units of observations in the same dataset can have an adverse effect for certain machine learning models, especially when minimizing the cost function of the algorithm because it shrinks the function space and makes it difficult for an optimization algorithm like gradient descent to find the global minimum.

When performing data rescaling, usually the attributes are rescaled with the range of 0 and 1. Data rescaling is implemented in Scikit-learn using the **MinMaxScaler** module. Let's see an example.

```
# import packages
from sklearn import datasets
from sklearn.preprocessing import MinMaxScaler
```

```
# load dataset
data = datasets.load_iris()
# separate features and target
X = data.data
y = data.target

# print first 5 rows of X before rescaling
X[0:5,:]
'Output':
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])

# rescale X
scaler = MinMaxScaler(feature_range=(0, 1))
rescaled_X = scaler.fit_transform(X)

# print first 5 rows of X after rescaling
rescaled_X[0:5,:]
'Output':
array([[0.22222222, 0.625     , 0.06779661, 0.04166667],
       [0.16666667, 0.41666667, 0.06779661, 0.04166667],
       [0.11111111, 0.5       , 0.05084746, 0.04166667],
       [0.08333333, 0.45833333, 0.08474576, 0.04166667],
       [0.19444444, 0.66666667, 0.06779661, 0.04166667]])
```

Standardization

Linear machine learning algorithms such as linear regression and logistic regression make an assumption that the observations of the dataset are normally distributed with a mean of 0 and standard deviation of 1. However, this is often not the case with real-world datasets as features are often skewed with differing means and standard deviations.