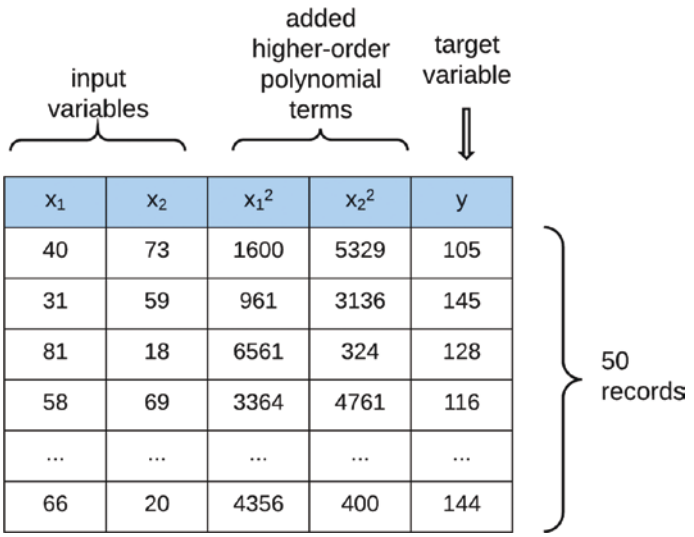
In the preceding code, using the *train_test_split()* function, the dataset is split into training and testing sets. The linear regression algorithm is applied to the training dataset to find the optimal values that parameterize the weights of the model. The model is evaluated by calling the *.predict()* function on the test set.

The error of the model is evaluated using the RMSE error metric (discussed in Chapter 14).

Adapting to Non-linearity

Although linear regression has the premise that the underlying structure of the dataset features is linear, this is, however, not the case for most datasets. It is nevertheless possible to adapt linear regression to fit or build a model for non-linear datasets. This process of adding non-linearity to linear models is called ***polynomial regression***.

Polynomial regression fits a non-linear relationship to the data by adding higher-order polynomial terms of existing data features as new features in the dataset. More of this is visualized in Figure 19-5.



x_1	x_2	x_1^2	x_2^2	y
40	73	1600	5329	105
31	59	961	3136	145
81	18	6561	324	128
58	69	3364	4761	116
...	...	...	...	...
66	20	4356	400	144

Figure 19-5. *Adding polynomial features to the dataset*

It is important to note that from a statistical point of view, when approximating the optimal values of the weights to minimize the model, the underlying assumption of the interactions of the parameters is linear. Non-linear regression models may tend to overfit the data, but this can be mitigated by adding regularization to the model. Here is a formal example of the polynomial regression model.

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_2 + \theta_4 x_2^2 + \ldots + \theta_n x_n + \theta_n x_n^2$$

An illustration of polynomial regression is shown in Figure 19-6.

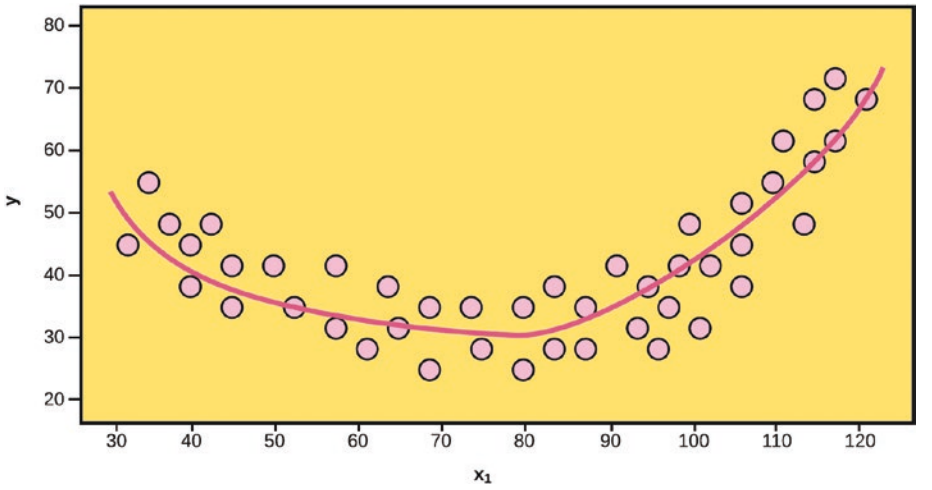


Figure 19-6. *Fitting a non-linear model with polynomial regression*

Higher-Order Linear Regression with Scikit-learn

In this example, we will create higher-order polynomials from the dataset features in hope of fitting a more flexible model that may better capture the variance in the dataset. As seen in Chapter 18, we will use the PolynomialFeatures method to create these higher-order polynomial and interaction features. The following code example is similar to the previous code example except where it extends the feature matrix with higher-order features.

```
# import packages
from sklearn.linear_model import LinearRegression
from sklearn import datasets
```