```
# 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.

Applying the technique of standardization to the datasets transforms the features into a standard Gaussian (or normal) distribution with a mean of 0 and standard deviation of 1. Scikit-learn implements data standardization in the **StandardScaler** module. Let's look at an example.

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

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

# print first 5 rows of X before standardization
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]])

# standardize X
scaler = StandardScaler().fit(X)
standardize_X = scaler.transform(X)

# print first 5 rows of X after standardization
standardize_X[0:5,:]
'Output':
array([[-0.90068117,  1.03205722, -1.3412724 , -1.31297673],
       [-1.14301691, -0.1249576 , -1.3412724 , -1.31297673],
       [-1.38535265,  0.33784833, -1.39813811, -1.31297673],
       [-1.50652052,  0.10644536, -1.2844067 , -1.31297673],
       [-1.02184904,  1.26346019, -1.3412724 , -1.31297673]])
```

Normalization

Data normalization involves transforming the observations in the dataset so that it has a unit norm or has magnitude or length of 1. The length of a vector is the square root of the sum of squares of the vector elements. A unit vector (or unit norm) is obtained by dividing the vector by its length. Normalizing the dataset is particularly useful in scenarios where the dataset is sparse (i.e., a large number of observations are zeros) and also has differing scales. Normalization in Scikit-learn is implemented in the **Normalizer** module.

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

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

# print first 5 rows of X before normalization
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]])

# normalize X
scaler = Normalizer().fit(X)
normalize_X = scaler.transform(X)

# print first 5 rows of X after normalization
normalize_X[0:5,:]
'Output':
```