Scaling

This lesson explains how scaling helps cleaning the data.

Introduction to scaling

The scale of your features matter for many machine learning algorithms. Having income values that range from 100 to 100,000 and ages that range from 0 to 100 can cause issues because of the large difference in scale of these two data columns. To deal with this, it is standard to rescale the data. There are many ways to do this, but the two most common ones are:

- Standard scaling
- Min/Max scaling.

Types of scaling

Standard scaling

Standard scaling subtracts the mean and divides by the standard deviation. This centers the feature on zero with unit variance.

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import numpy as np

# Create a matrix of data
data = [[-1, 2],
        [-0.5, 6],
        [0, 10],
        [1, 18]]
        [1, 18]]

print("Before Standard scaling")
print(np.mean(data, 0))
print(np.std(data, 0))

# Initalize a StandardScaler
standard = StandardScaler()
# Fit and transform the data with the StandardScaler
standard_data = standard.fit_transform(data)

print("After Standard scaling")
print(np.mean(standard_data, 0))
print(np.std(standard_data, 0))
```

In the example above, we created a NumPy array of shape (4,2). We then use the `StandardScaler()` from `sklearn` which will automatically subtract the mean and divide by the standard deviation of each of our columns. This is done with the `fit_transform()` call.

We check that it worked by printing the mean and standard deviation. We can see that both columns now have a mean of 0 and a standard deviation of 1.

Min/Max scaling

Let’s look at the same example but instead use the `MinMaxScaler()` from `sklearn`.

```
from sklearn.preprocessing import MinMaxScaler
import numpy as np

# Create matrix of data
data = [[-1, 2],
        [-0.5, 6],
        [0, 10],
        [1, 18]]

# Initalize MinMaxScaler
min_max = MinMaxScaler()
# Fit and transform the data
min_max_data = min_max.fit_transform(data)

print(np.min(min_max_data, 0))
print(np.max(min_max_data, 0))
print(np.mean(min_max_data, 0))
print(np.std(min_max_data, 0))
```

By default, the `MinMaxScaler()` scales each column to have a min value of **0** and a max value of **1**.

We confirm that our code did this by printing out the min and max values. We also printed out the mean and std of the columns to see that they are no longer 0 and 1 respectively. You can change the min, max range to what you want by passing the following parameters:

```
feature_range=(min, max)
```

That’s it about scaling the data. In the next lesson, you’ll learn how to handle the data if data is categorical.