Scale Features

When your data has different values, and even different measurement units, it can be difficult to compare them. What is kilograms compared to meters? Or altitude compared to time?

The answer to this problem is scaling. We can scale data into new values that are easier to compare.

Take a look at the table below, it is the same data set that we used in the [multiple regression chapter](https://www.w3schools.com/python/python_ml_multiple_regression.asp), but this time the **volume** column contains values in*liters* instead of *cm3* (1.0 instead of 1000).

The file is meant for testing purposes only, you can download it here: [cars2.csv](https://www.w3schools.com/python/cars2.csv)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Car | Model | Volume | Weight | CO2 |
| Toyota | Aygo | 1000 | 790 | 99 |
| Mitsubishi | Space Star | 1200 | 1160 | 95 |
| Skoda | Citigo | 1000 | 929 | 95 |
| Fiat | 500 | 900 | 865 | 90 |
| Mini | Cooper | 1500 | 1140 | 105 |
| VW | Up! | 1000 | 929 | 105 |
| Skoda | Fabia | 1400 | 1109 | 90 |
| Mercedes | A-Class | 1500 | 1365 | 92 |
| Ford | Fiesta | 1500 | 1112 | 98 |
| Audi | A1 | 1600 | 1150 | 99 |
| Hyundai | I20 | 1100 | 980 | 99 |
| Suzuki | Swift | 1300 | 990 | 101 |
| Ford | Fiesta | 1000 | 1112 | 99 |
| Honda | Civic | 1600 | 1252 | 94 |
| Hundai | I30 | 1600 | 1326 | 97 |
| Opel | Astra | 1600 | 1330 | 97 |
| BMW | 1 | 1600 | 1365 | 99 |
| Mazda | 3 | 2200 | 1280 | 104 |
| Skoda | Rapid | 1600 | 1119 | 104 |
| Ford | Focus | 2000 | 1328 | 105 |
| Ford | Mondeo | 1600 | 1584 | 94 |
| Opel | Insignia | 2000 | 1428 | 99 |
| Mercedes | C-Class | 2100 | 1365 | 99 |
| Skoda | Octavia | 1600 | 1415 | 99 |
| Volvo | S60 | 2000 | 1415 | 99 |
| Mercedes | CLA | 1500 | 1465 | 102 |
| Audi | A4 | 2000 | 1490 | 104 |
| Audi | A6 | 2000 | 1725 | 114 |
| Volvo | V70 | 1600 | 1523 | 109 |
| BMW | 5 | 2000 | 1705 | 114 |
| Mercedes | E-Class | 2100 | 1605 | 115 |
| Volvo | XC70 | 2000 | 1746 | 117 |
| Ford | B-Max | 1600 | 1235 | 104 |
| BMW | 2 | 1600 | 1390 | 108 |
| Opel | Zafira | 1600 | 1405 | 109 |
| Mercedes | SLK | 2500 | 1395 | 120 |

It can be difficult to compare the volume 1.0 with the weight 790, but if we scale them both into comparable values, we can easily see how much one value is compared to the other.

There are different methods for scaling data, in this tutorial we will use a method called standardization.

The standardization method uses this formula:

z = (x - u) / s

Where z is the new value, x is the original value, u is the mean and s is the standard deviation.

If you take the **weight** column from the data set above, the first value is 790, and the scaled value will be:

(790 - [1292.23](https://www.w3schools.com/python/showpython.asp?filename=demo_ml_scale_mean1)) / [238.74](https://www.w3schools.com/python/showpython.asp?filename=demo_ml_scale_std1) = -2.1

If you take the **volume** column from the data set above, the first value is 1.0, and the scaled value will be:

(1.0 - [1.61](https://www.w3schools.com/python/showpython.asp?filename=demo_ml_scale_mean2)) / [0.38](https://www.w3schools.com/python/showpython.asp?filename=demo_ml_scale_std2) = -1.59

Now you can compare -2.1 with -1.59 instead of comparing 790 with 1.0.

Scale all values in the Weight and Volume columns:

import pandas  
from sklearn import linear\_model

from sklearn.preprocessing import StandardScaler  
scale = StandardScaler()  
  
df = pandas.read\_csv("cars2.csv")  
  
X = df[['Weight', 'Volume']]  
  
scaledX = scale.fit\_transform(X)  
  
print(scaledX)

### **Result:**

Note that the first two values are -2.1 and -1.59, which corresponds to our calculations:

[[-2.10389253 -1.59336644]

[-0.55407235 -1.07190106]

[-1.52166278 -1.59336644]

[-1.78973979 -1.85409913]

[-0.63784641 -0.28970299]

[-1.52166278 -1.59336644]

[-0.76769621 -0.55043568]

[ 0.3046118 -0.28970299]

[-0.7551301 -0.28970299]

[-0.59595938 -0.0289703 ]

[-1.30803892 -1.33263375]

[-1.26615189 -0.81116837]

[-0.7551301 -1.59336644]

[-0.16871166 -0.0289703 ]

[ 0.14125238 -0.0289703 ]

[ 0.15800719 -0.0289703 ]

[ 0.3046118 -0.0289703 ]

[-0.05142797 1.53542584]

[-0.72580918 -0.0289703 ]

[ 0.14962979 1.01396046]

[ 1.2219378 -0.0289703 ]

[ 0.5685001 1.01396046]

[ 0.3046118 1.27469315]

[ 0.51404696 -0.0289703 ]

[ 0.51404696 1.01396046]

[ 0.72348212 -0.28970299]

[ 0.8281997 1.01396046]

[ 1.81254495 1.01396046]

[ 0.96642691 -0.0289703 ]

[ 1.72877089 1.01396046]

[ 1.30990057 1.27469315]

[ 1.90050772 1.01396046]

[-0.23991961 -0.0289703 ]

[ 0.40932938 -0.0289703 ]

[ 0.47215993 -0.0289703 ]

[ 0.4302729 2.31762392]]

## Predict CO2 Values

The task in the [Multiple Regression chapter](https://www.w3schools.com/python/python_ml_multiple_regression.asp) was to predict the CO2 emission from a car when you only knew its weight and volume.

When the data set is scaled, you will have to use the scale when you predict values:

### **Example**

Predict the CO2 emission from a 1.3 liter car that weighs 2300 kilograms:

import pandas  
from sklearn import linear\_model

from sklearn.preprocessing import StandardScaler  
scale = StandardScaler()  
  
df = pandas.read\_csv("cars2.csv")  
  
X = df[['Weight', 'Volume']]  
y = df['CO2']  
  
scaledX = scale.fit\_transform(X)  
  
regr = linear\_model.LinearRegression()  
regr.fit(scaledX, y)  
  
scaled = scale.transform([[2300, 1.3]])  
  
predictedCO2 = regr.predict([scaled[0]])  
print(predictedCO2)

### **Result:**

[107.2087328]