Feature Scaling

Feature Scaling is a data preprocessing technique. By preprocessing, we mean the transformations that are applied to the data before it is fed into some algorithm for some processing.

Feature Scaling is a technique where we standardize the range of all independent features of a data-set. It is also called Normalization.

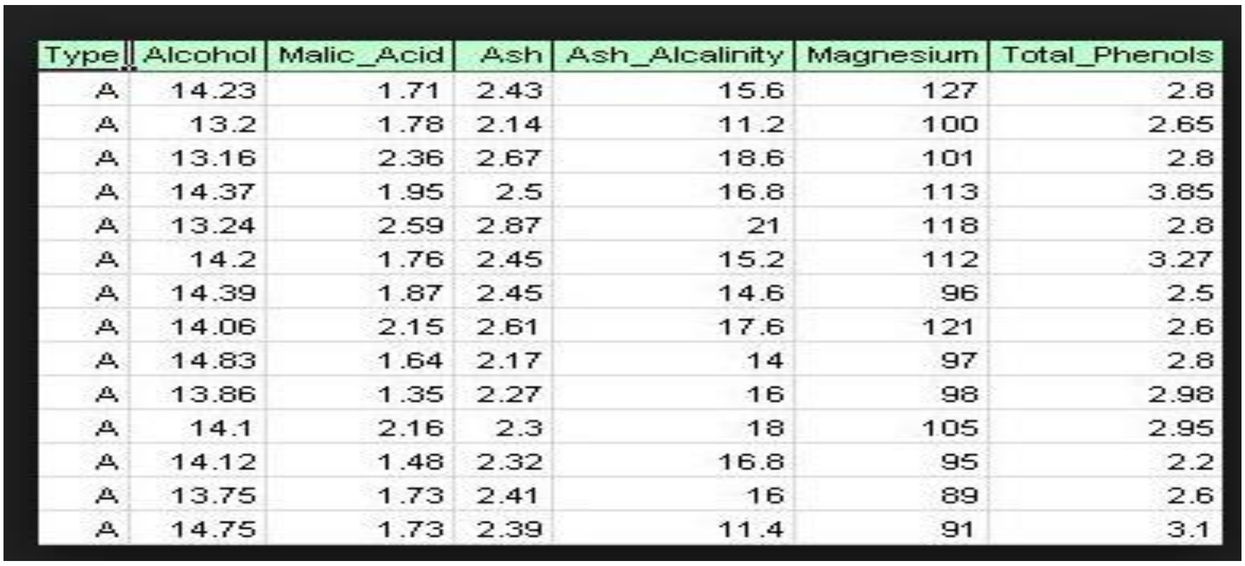
Generally, when we get raw data, all the features values varies on different scales. It is important to bring all the feature values on the scale so that value of one feature should not dominate over the others and hinder the performance of the learning algorithm. This process of bringing all the features values on the same scale is called feature scaling. Ensuring standardised feature values implicitly weights all features equally in their representation.

Feature scaling or re-scaling of the features is performed such that they have the properties of normal distribution (most of the time) where values have standard deviation = 1, and mean = 0.

Why Feature Scaling is required?

Most of the real world applied machine learning algorithms are classification algorithms. Many of the classification algorithms works by calculating the distance between data points in space. If one feature has a wide range of values, then this feature is likely to dominate the distance measure between the data points over other features. Above this, if this feature proves to be insignificant in the end then, it will be hindering the algorithm results to a large extent. This will result in decrease in accuracy of the algorithm.

For example : Let us look at the subset of wine dataset:



Type	Alcohol	Malic_Acid	Ash	Ash_Alcalinity	Magnesium	Total_Phenols
A	14.23	1.71	2.43	15.6	127	2.8
A	13.2	1.78	2.14	11.2	100	2.65
A	13.16	2.36	2.67	18.6	101	2.8
A	14.37	1.95	2.5	16.8	113	3.85
A	13.24	2.59	2.87	21	118	2.8
A	14.2	1.76	2.45	15.2	112	3.27
A	14.39	1.87	2.45	14.6	96	2.5
A	14.06	2.15	2.61	17.6	121	2.6
A	14.83	1.64	2.17	14	97	2.8
A	13.86	1.35	2.27	16	98	2.98
A	14.1	2.16	2.3	18	105	2.95
A	14.12	1.48	2.32	16.8	95	2.2
A	13.75	1.73	2.41	16	89	2.6
A	14.75	1.73	2.39	11.4	91	3.1

The range of value for “Magnesium” feature is 80 - 100 or above, while range of values of “Malic_Acid” feature is ranging 1.something.
Now if we apply distance formula(as we do in case of KNN) on two data points here, Magnesium will dominate the distance value to a greater extent than any of the other feature, thus resulting in wrong predictions later.

Note - Distance Formula is given as d = ((x1 - x2)^2 + (y1-y2)^2.. .)^½

Let’s understand this better with the help of another example –
Suppose you have a company employees’ data. Now the age of employees in a company may be between 21-70 years, the size of the house they live is 500-5000 Sq feet and their salaries may range from 30000−80000. In this situation if you use a simple Euclidean metric, the age feature will not play any role because it is several order smaller than other features. However, it may contain some important information that may be useful for the task. Here, you may want to normalize the features independently to the same scale, say [0,1], so they contribute equally while computing the distance.”

How Feature Scaling is applied in sklearn?

There are many ways by which we can apply feature scaling on the dataset:

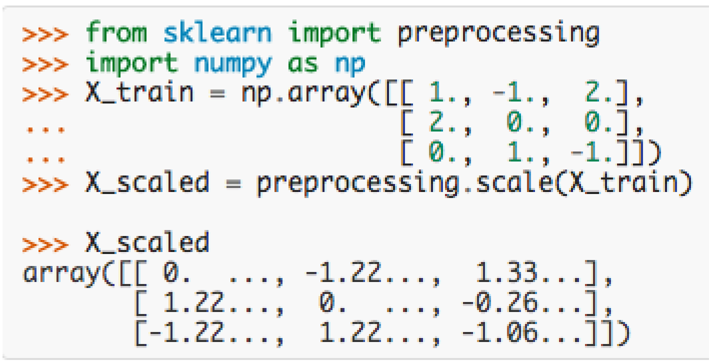
1. The easiest way of scaling is to use - preprocessing.scale() function

   A numpy array of values is given as input and output is numpy array with scaled

   values. This will scale the values in such a way that mean of the values will be 0 and

   standard

   deviation will be 1.



```
>>> from sklearn import preprocessing
>>> import numpy as np
>>> X_train = np.array([[ 1., -1.,  2.],
...                     [ 2.,  0.,  0.],
...                     [ 0.,  1., -1.]])
>>> X_scaled = preprocessing.scale(X_train)

>>> X_scaled
array([[ 0.  ..., -1.22...,  1.33...],
       [ 1.22...,  0.  ..., -0.26...],
       [-1.22...,  1.22..., -1.06...]])
```

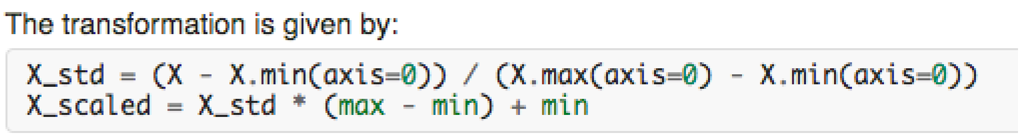
Another method is to scale the features between given minimum and maximum values, generally between 0 and 1.

Function -preprocessing.MinMaxScaler(feature_range=(0, 1), copy=True)

Feature_range is given in the form of tuple - (min , max)

Copy - True (default), set it to False if you want inplace transformation

Formula used -



The transformation is given by:

```
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max - min) + min
```