FEATURE SCALING

Feature scaling is used to scale down large values into small values.

Why feature scaling?

What is Normalization(**Min-Max scaling)**?

**Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.**



Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.

## What is Standardization?

**Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.**



* Normalization is good to use when you know that the distribution of your data does not follow a Gaussian distribution. This can be useful in algorithms that do not assume any distribution of the data like K-Nearest Neighbors and Neural Networks.
* Standardization, on the other hand, can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true. Also, unlike normalization, standardization does not have a bounding range. So, even if you have outliers in your data, they will not be affected by standardization.

### **Normalization using sklearn**

To normalize your data, you need to import the MinMaxScalar from the [sklearn](https://courses.analyticsvidhya.com/courses/get-started-with-scikit-learn-sklearn?utm_source=blog&utm_medium=feature-scaling-machine-learning-normalization-standardization" \t "_blank) library and apply it to our dataset. So, let’s do that!

|  |  |
| --- | --- |
|  | # data normalization with sklearn |
|  | from sklearn.preprocessing import MinMaxScaler |
|  |  |
|  | # fit scaler on training data |
|  | norm = MinMaxScaler().fit(X\_train) |
|  |  |
|  | # transform training data |
|  | X\_train\_norm = norm.transform(X\_train) |
|  |  |
|  | # transform testing dataabs |
|  | X\_test\_norm = norm.transform(X\_test) |

### **Standardization using sklearn**

To standardize your data, you need to import the StandardScalar from the sklearn library and apply it to our dataset. Here’s how you can do it:

|  |  |
| --- | --- |
|  | # data standardization with sklearn |
|  | from sklearn.preprocessing import StandardScaler |
|  |  |
|  | # copy of datasets |
|  | X\_train\_stand = X\_train.copy() |
|  | X\_test\_stand = X\_test.copy() |
|  |  |
|  | # numerical features |
|  | num\_cols = ['Item\_Weight','Item\_Visibility','Item\_MRP','Outlet\_Establishment\_Year'] |
|  |  |
|  | # apply standardization on numerical features |
|  | for i in num\_cols: |
|  |  |
|  | # fit on training data column |
|  | scale = StandardScaler().fit(X\_train\_stand[[i]]) |
|  |  |
|  | # transform the training data column |
|  | X\_train\_stand[i] = scale.transform(X\_train\_stand[[i]]) |
|  |  |
|  | # transform the testing data column |
|  | X\_test\_stand[i] = scale.transform(X\_test\_stand[[i]]) |

### **Comparing unscaled, normalized and standardized data**

It is always great to visualize your data to understand the distribution present. We can see the comparison between our unscaled and scaled data using boxplots.

