# https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35

# <https://www.geeksforgeeks.org/ml-feature-scaling-part-2/>

# <https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/>?(Best One)

# https://www.atoti.io/when-to-perform-a-feature-scaling/(Best One)

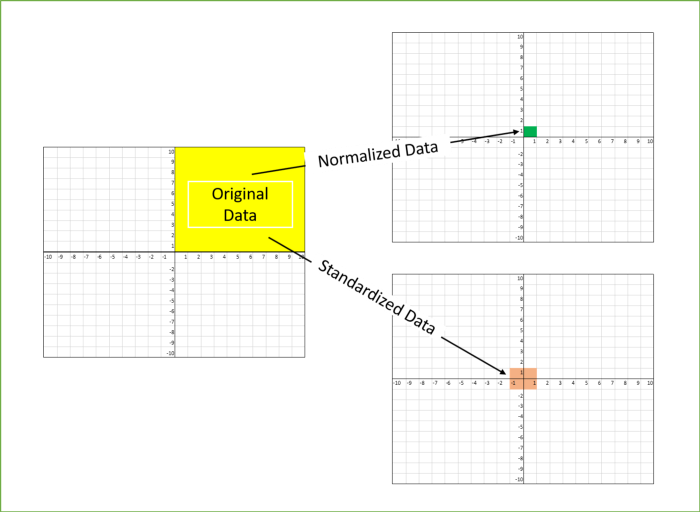
# Feature Scalling :- Feature scaling in machine learning is one of the most critical steps during the pre-processing of data before creating a machine learning model. Scaling can make a difference between a weak machine learning model and a better one.

# Similarly, in many machine learning algorithms, to bring all features in the same standing, we need to do scaling so that one significant number doesn’t impact the model just because of their large magnitude.

The most common techniques of feature scaling are Normalization and Standardization.

Normalization is used when we want to bound our values between two numbers, typically, between [0,1] or [-1,1].

While Standardization transforms the data to have zero mean and a variance of 1, they make our data unitless. Refer to the below diagram, which shows how data looks after scaling in the X-Y plane.



### **The million-dollar question: Normalization or Standardization**

If you have ever built a machine learning pipeline, you must have always faced this question of whether to Normalize or to Standardize. While there is no obvious answer to this question, it really depends on the application, there are still a few[generalizations that can be drawn](https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/).

Normalization is good to use when the distribution of data does not follow a Gaussian distribution. It can be useful in algorithms that do not assume any distribution of the data like K-Nearest Neighbors.

In Neural Networks algorithm that require data on a 0–1 scale, normalization is an essential pre-processing step. Another popular example of data normalization is image processing, where pixel intensities have to be normalized to fit within a certain range (i.e., 0 to 255 for the RGB color range).

Standardization can be helpful in cases where the data follows a Gaussian distribution. Though this does not have to be necessarily true. Since standardization does not have a bounding range, so, even if there are outliers in the data, they will not be affected by standardization.

In clustering analyses, standardization comes in handy to compare similarities between features based on certain distance measures. Another prominent example is the Principal Component Analysis, where we usually prefer standardization over Min-Max scaling since we are interested in the components that maximize the variance.

There are some points which can be considered while deciding whether we need Standardization or Normalization

* Standardization may be used when data represent Gaussian Distribution, while Normalization is great with Non-Gaussian Distribution
* Impact of Outliers is very high in Normalization

To conclude, you can always start by fitting your model to raw, normalized, and standardized data and compare the performance for the best results.

# **Why do we need scaling?**

Machine learning algorithm just sees number — if there is a vast difference in the range say few ranging in thousands and few ranging in the tens, and it makes the underlying assumption that higher ranging numbers have superiority of some sort. So these more significant number starts playing a more decisive role while training the model.

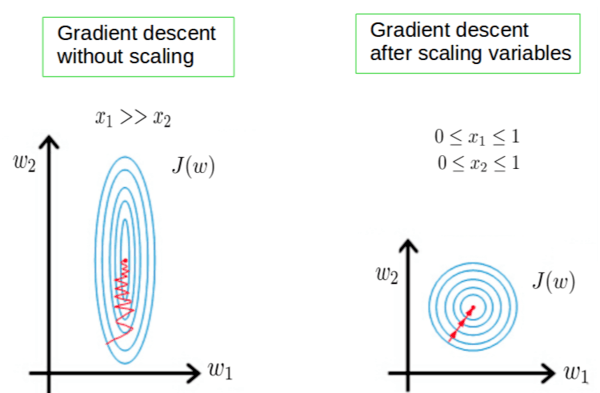
The machine learning algorithm works on numbers and does not know what that number represents. A weight of 10 grams and a price of 10 dollars represents completely two different things — which is a no brainer for humans, but for a model as a feature, it treats both as same.

Suppose we have two features of weight and price, as in the below table. The “Weight” cannot have a meaningful comparison with the “Price.” So the assumption algorithm makes that since “Weight” > “Price,” thus “Weight,” is more important than “Price.”

# 

So these more significant number starts playing a more decisive role while training the model. Thus feature scaling is needed to bring every feature in the same footing without any upfront importance. Interestingly, if we convert the weight to “Kg,” then “Price” becomes dominant.

Another reason why feature scaling is applied is that few algorithms like Neural network gradient descent converge much faster with feature scaling than without it.



One more reason **is saturation**, like in the case of sigmoid activation in Neural Network, scaling would help not to saturate too fast.

# **When to do scaling?**

Feature scaling is essential for machine learning algorithms that calculate **distances between data.** If not scale, the feature with a higher value range starts dominating when calculating distances, as explained intuitively in the “why?” section.

The ML algorithm is sensitive to the **“relative scales of features,”** which usually happens when it uses the numeric values of the features rather than say their rank.

In many algorithms, when we desire**faster convergence,** scaling is a MUST like in Neural Network.

Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions do not work correctly without normalization. For example, the majority of classifiers calculate the distance between two points by the distance. If one of the features has a broad range of values, the distance governs this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance.

Even when the conditions, as mentioned above, are not satisfied, you may still need to rescale your features if the ML algorithm expects some scale or a saturation phenomenon can happen. Again, a neural network with saturating activation functions (e.g., sigmoid) is a good example.

Rule of thumb we may follow here is an algorithm that computes distance or assumes normality, **scales your features.**

Some examples of algorithms where feature scaling matters are:

* K-nearest neighbors (KNN) with a Euclidean distance measure is sensitive to magnitudes and hence should be scaled for all features to weigh in equally.
* K-Means uses the Euclidean distance measure here feature scaling matters.
* Scaling is critical while performing Principal Component Analysis(PCA). PCA tries to get the features with maximum variance, and the variance is high for high magnitude features and skews the PCA towards high magnitude features.
* We can speed up gradient descent by scaling because θ descends quickly on small ranges and slowly on large ranges, and oscillates inefficiently down to the optimum when the variables are very uneven.

 rely on rules CART, Random Forests, Gradient Boosted Decision Trees do not require normalization

Algorithms like Linear Discriminant Analysis(LDA), Naive Bayes is by design equipped to handle this and give weights to the features accordingly. Performing features scaling in these algorithms may not have much effect.

Few key points to note :

* Mean centering does not affect the covariance matrix
* Scaling of variables does affect the covariance matrix
* Standardizing affects the covariance

# How to perform feature scaling?

Below are the few ways we can do feature scaling.

1) Min Max Scaler  
2) Standard Scaler  
3) Max Abs Scaler  
4) Robust Scaler  
5) Quantile Transformer Scaler  
6) Power Transformer Scaler  
7) Unit Vector Scaler

For the explanation, we will use the table shown in the top and form the data frame to show different scaling methods.

# For the explanation, we will use the table shown in the top and form the data frame to show different scaling methods.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

df = pd.DataFrame({'WEIGHT': [15, 18, 12,10],

'PRICE': [1,3,2,5]},

index = ['Orange','Apple','Banana','Grape'])

print(df)WEIGHT PRICE

Orange 15 1

Apple 18 3

Banana 12 2

Grape 10 5

## **1)Min-Max scaler**

Transform features by scaling each feature to a given range. This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g., between zero and one. This Scaler shrinks the data within the range of -1 to 1 if there are negative values. We can set the range like [0,1] or [0,5] or [-1,1].

This Scaler responds well if the standard deviation is small and when a distribution is **not Gaussian**. This Scaler is **sensitive to outliers**.

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()df1 = pd.DataFrame(scaler.fit\_transform(df),

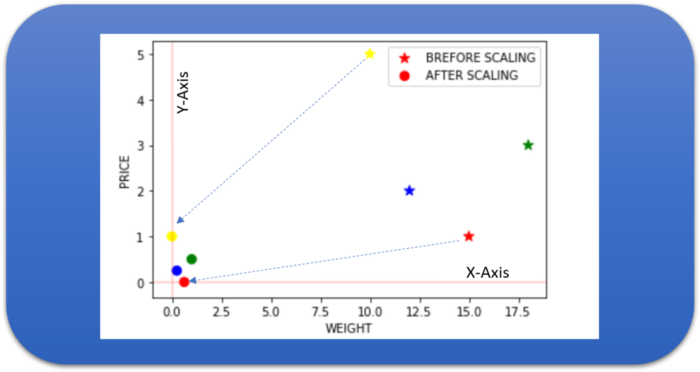
columns=['WEIGHT','PRICE'],

index = ['Orange','Apple','Banana','Grape'])ax = df.plot.scatter(x='WEIGHT', y='PRICE',color=['red','green','blue','yellow'],

marker = '\*',s=80, label='BREFORE SCALING');df1.plot.scatter(x='WEIGHT', y='PRICE', color=['red','green','blue','yellow'],

marker = 'o',s=60,label='AFTER SCALING', ax = ax);plt.axhline(0, color='red',alpha=0.2)

plt.axvline(0, color='red',alpha=0.2);



## **2) Standard Scaler**

The Standard Scaler assumes data is normally distributed within each feature and scales them such that the distribution centered around 0, with a standard deviation of 1.

Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. If data is not normally distributed, this is not the best Scaler to use.

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

df2 = pd.DataFrame(scaler.fit\_transform(df),

columns=['WEIGHT','PRICE'],

index = ['Orange','Apple','Banana','Grape'])

ax = df.plot.scatter(x='WEIGHT', y='PRICE',color=['red','green','blue','yellow'],

marker = '\*',s=80, label='BREFORE SCALING');

df2.plot.scatter(x='WEIGHT', y='PRICE', color=['red','green','blue','yellow'],

marker = 'o',s=60,label='AFTER SCALING', ax = ax)

plt.axhline(0, color='red',alpha=0.2)

plt.axvline(0, color='red',alpha=0.2);

# https://www.youtube.com/watch?v=nmBqnKSSKfM&list=PLZoTAELRMXVPwYGE2PXD3x0bfKnR0cJjN&index=4