Standardisation data- the necessity?

Data used in machine learning can be varied including integers, float among others . This often may lead to more computing while running the learning algorithm. It may not be efficient to use the data without creating a common format. So standardization often comes to rescue in this case by standardizing the whole data. S**tandardization** rescales data to have a mean (μ ) of 0 and standard deviation (σ ) of 1 (unit variance)

One can make use of the library for the Python i.e scikit-learn for s**tandardization**

class sklearn.preprocessing.StandardScaler(copy=True, with\_mean=True, with\_std=True)

Standardize features by removing the mean and scaling to unit variance

The standard score of a sample x is calculated as:

z = (x - u) / s

where u is the mean of the training samples or zero if with\_mean=False, and s is the standard deviation of the training samples or one if with\_std=False.

Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using the transform method.

The following code implements the standardization in colab notebook.

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits import mplot3d

import matplotlib.colors

import pandas as pd

from sklearn.preprocessing import StandardScaler, MinMaxScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, mean\_squared\_error

from tqdm import tqdm\_notebook

R = np.random.random([100, 1])

plt.plot(R)

plt.show()

plt.plot(R)

plt.show()

np.mean(R)

np.std(R)

scaler = StandardScaler()

scaler.fit(R)

scaler.mean\_

RT = scaler.transform(R)

np.mean(RT)

np.std(RT)

plt.plot(RT)

plt.show()

In case of train and test data, you might wonder whether to standardization before or after splitting? It would be feasible to fit the data on the train data and could be used on test data for s**tandardization**