**Feature Scaling**

We discussed previously that the scale of the features is an important consideration when building machine learning models. Briefly:

**Feature magnitude matters because:**

* The **regression coefficients of linear** models are **directly influenced by the scale of the variable.**
* Variables with **bigger magnitude / larger value** range **dominate** over those with smaller magnitude / value range
* **Gradient descent converges faster** when features are on similar scales
* Feature scaling helps **decrease the time to find support vectors for SVMs**
* **Euclidean distances** are sensitive to feature magnitude.
* Some algorithms, like **PCA require the features to be centered at 0.**

**The machine learning models affected by the feature scale are:**

* Linear and Logistic Regression
* Neural Networks
* Support Vector Machines
* KNN
* K-means clustering
* Linear Discriminant Analysis (LDA)
* Principal Component Analysis (PCA)

**Feature Scaling**

**Feature scaling** refers to the methods or techniques used to **normalize the range of independent variables in our data**, or in other words, the methods to set the feature value range within **a similar scale.** Feature scaling is generally the last step in the data preprocessing pipeline, performed **just before training the machine learning algorithms**.

There are several Feature Scaling techniques, which we will discuss throughout this section:

* Standardisation
* Mean normalisation
* Scaling to minimum and maximum values - MinMaxScaling
* Scaling to maximum value - MaxAbsScaling
* Scaling to quantiles and median - RobustScaling
* **Normalization to vector unit length**

In this notebook, we will discuss **Standardisation**.

=================================================================

**Standardisation**

Standardisation **involves centering** the variable **at zero, and standardising the variance to 1**. The procedure involves subtracting the mean of each observation and then dividing by the standard deviation:

**z = (x - x\_mean) / std**

The result of the above transformation is **z**, which is **called the z-score**, and **represents how many standard deviations a given observation deviates from the mean**.

A z-score specifies the location of the observation within a distribution (in numbers of standard deviations respect to the mean of the distribution). The sign of the **z-score (+ or - )** indicates whether the observation is above (+) or below ( - ) the mean.

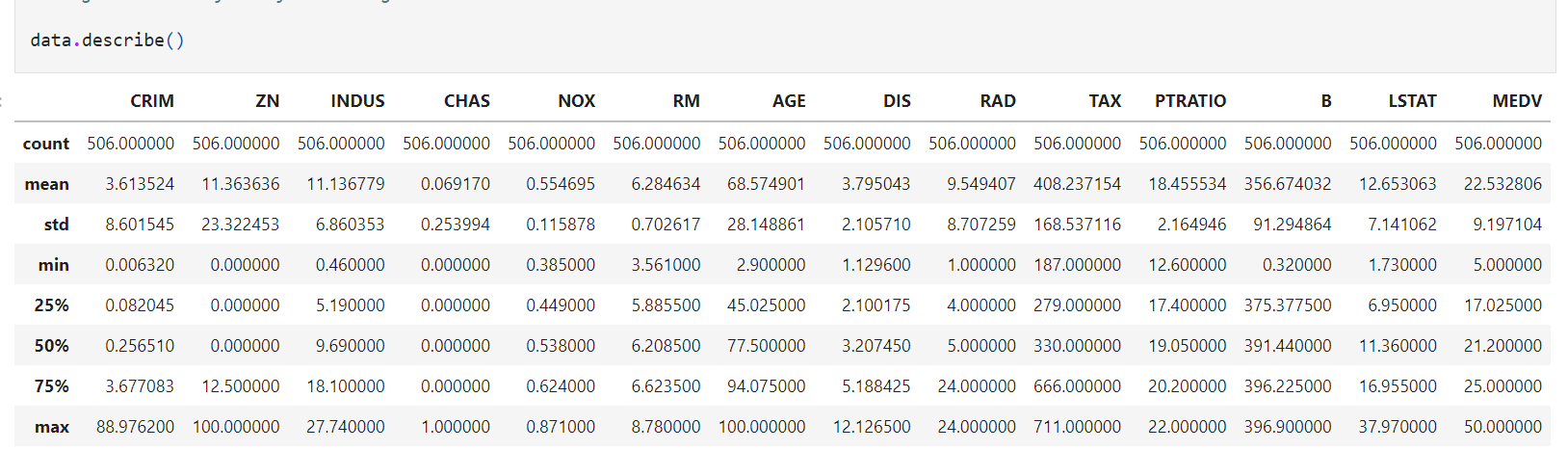
The shape of a **standardised (or z-scored normalised) distribution** will be identical to the **original distribution of the variable**. If the original distribution is normal, then the **standardised distribution will be normal.** But, if the **original distribution is skewed**, then the **standardised distribution of the variable will also be skewed**. In other words, **standardising a variable does not normalize the distribution of the data** and if this is the desired outcome, we should implement any of the techniques discussed in section 7 of the course.

In a nutshell, **standardisation:**

* centers the mean at 0
* scales the variance at 1
* preserves the **shape of the original distribution**
* the **minimum and maximum values of the different variables may vary**
* **preserves outliers**

Good for algorithms that require **features centered at zero.**

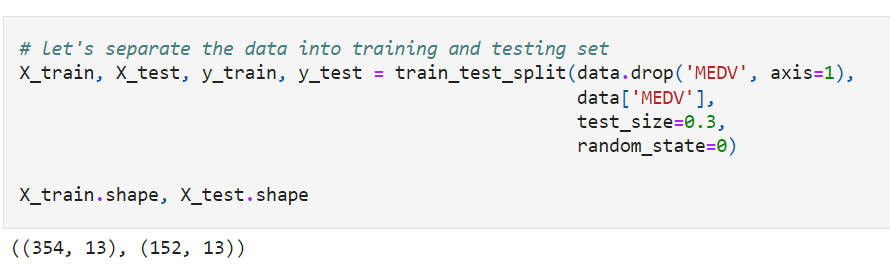




The different variables present different value ranges, mean, max, min, standard deviations, etc.

In other words, they show different magnitudes or scales. Note for this demo, how **the mean values are not centered at zero, and the standard deviations are not scaled to 1**.

When standardising the data set, **we need to first identify the mean and standard deviation** of the variables. These **parameters need to be learned from the train set**, stored, and then used to scale test and future data. Thus, we will first divide the data set into train and test, as we have done throughout the course.

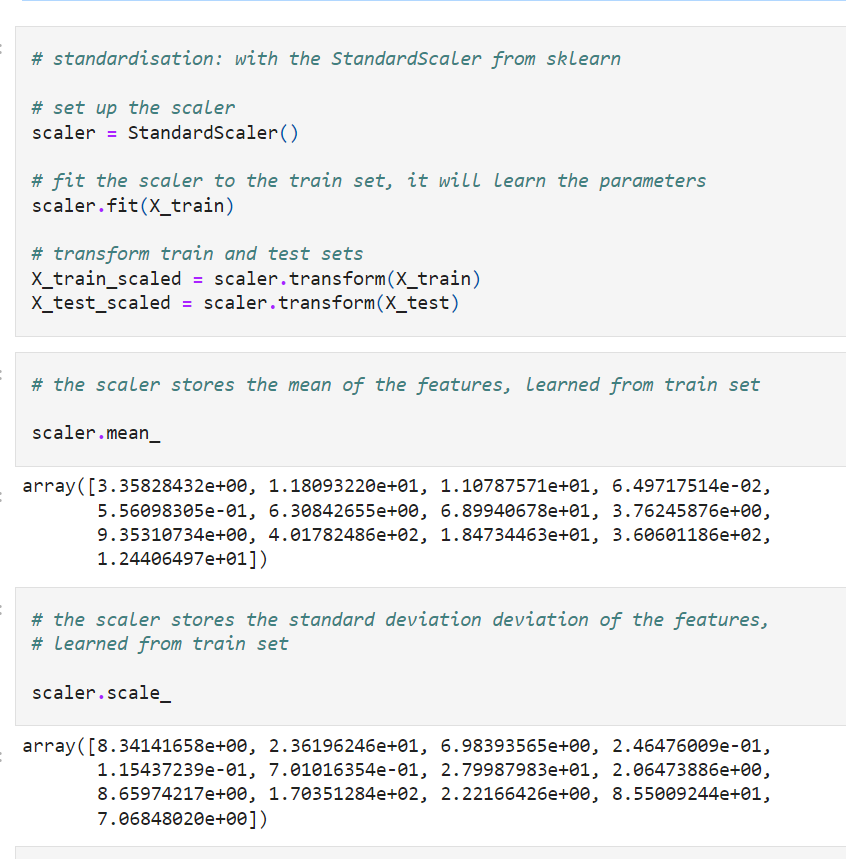


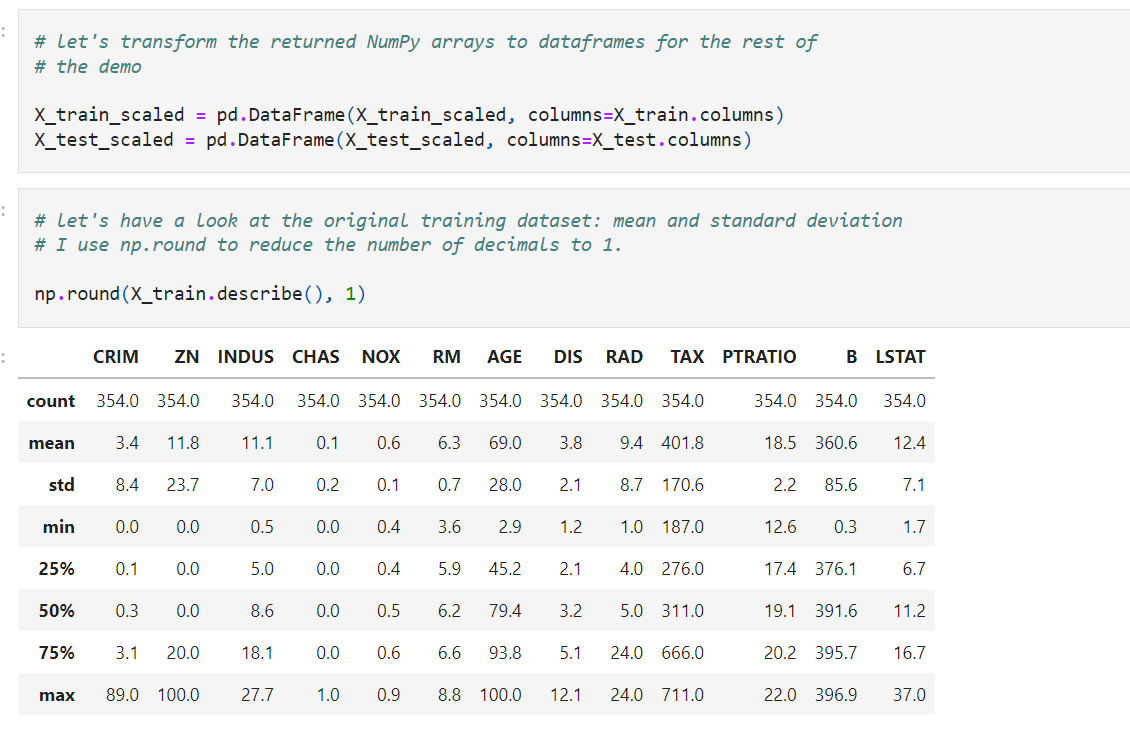
**Standardisation**

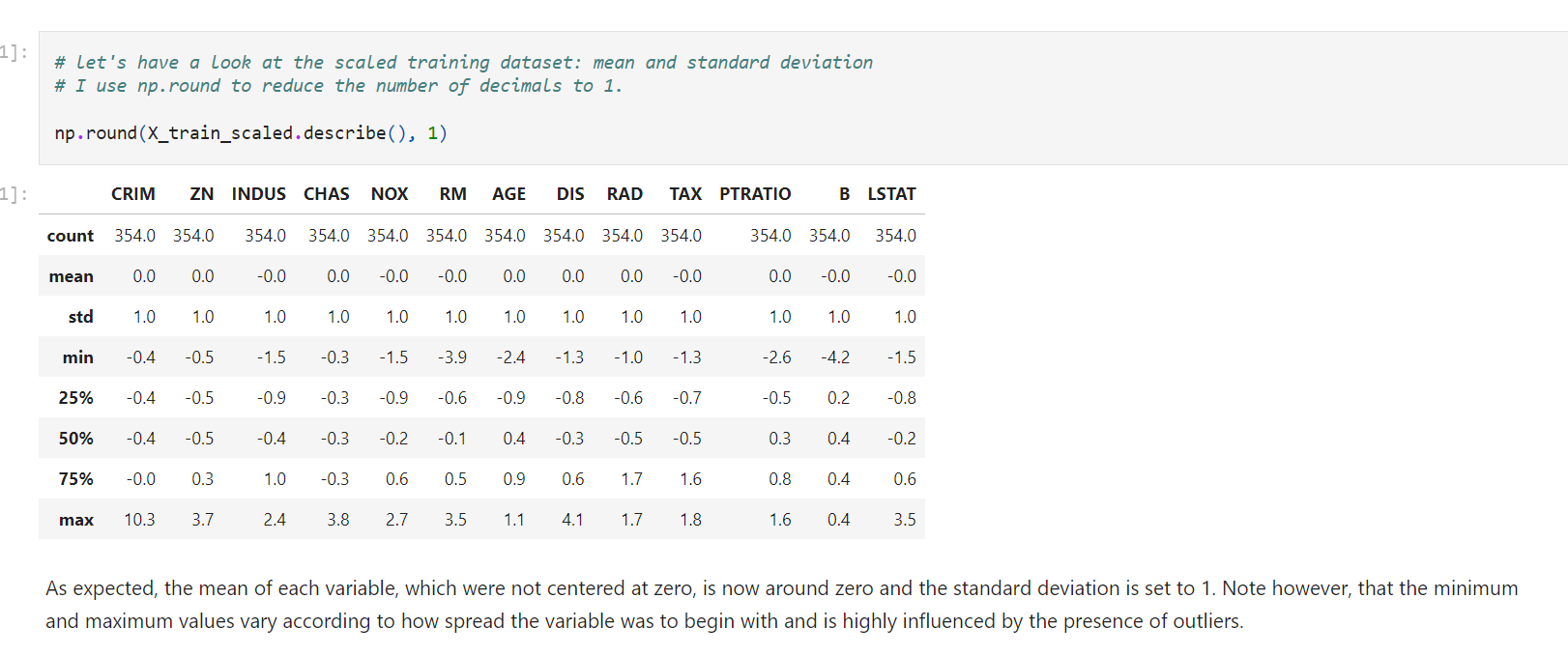
The StandardScaler from scikit-learn removes the **mean and scales the data to unit variance.**

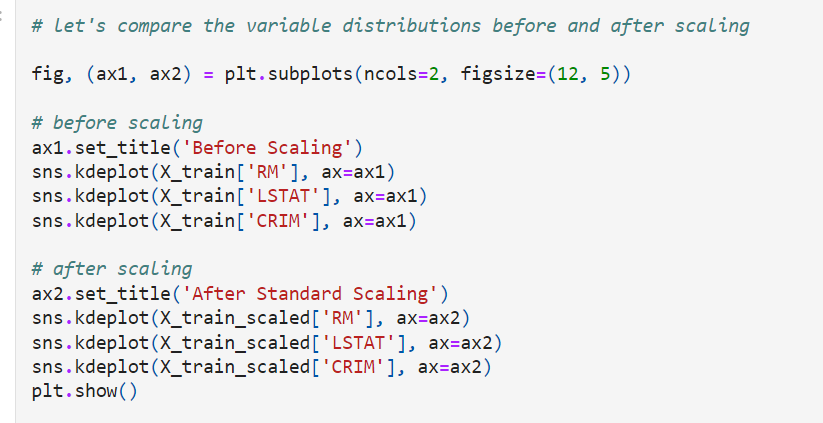
Plus, it learns and stores the parameters needed for scaling. Thus, it is top choice for this feature scaling technique.

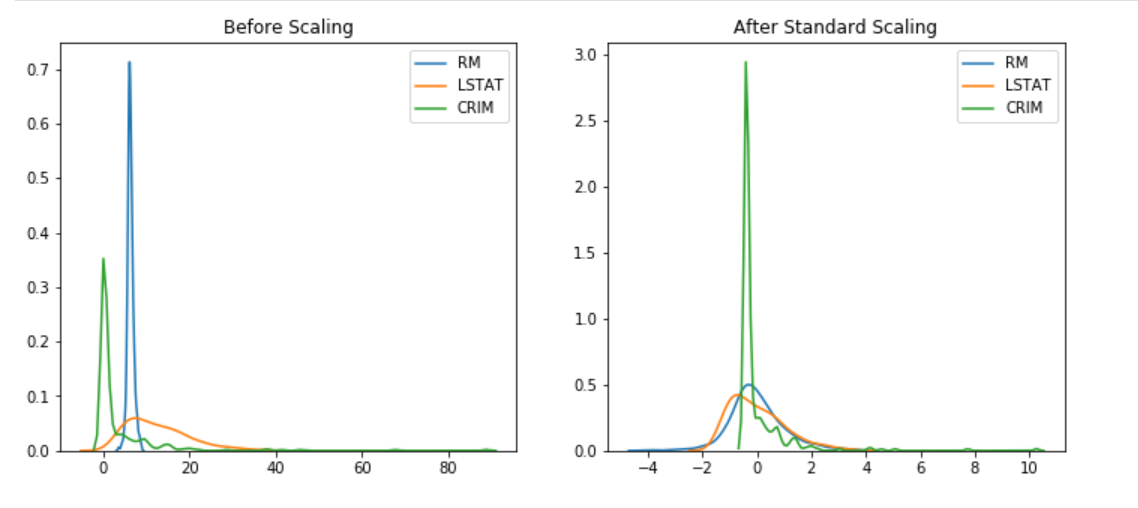
On the downside, you can't select which variables to scale directly, it will scale the entire data set, and it returns a NumPy array, without the variable values.





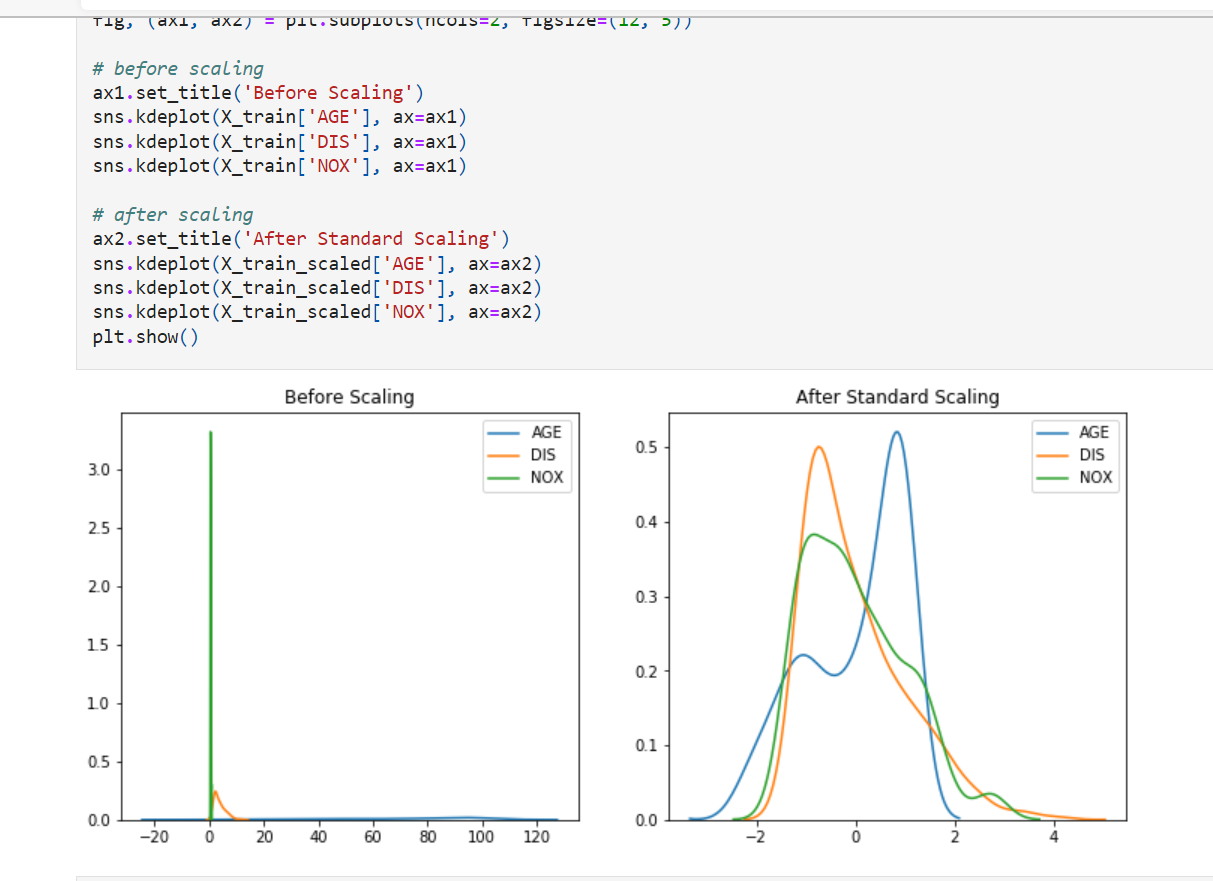


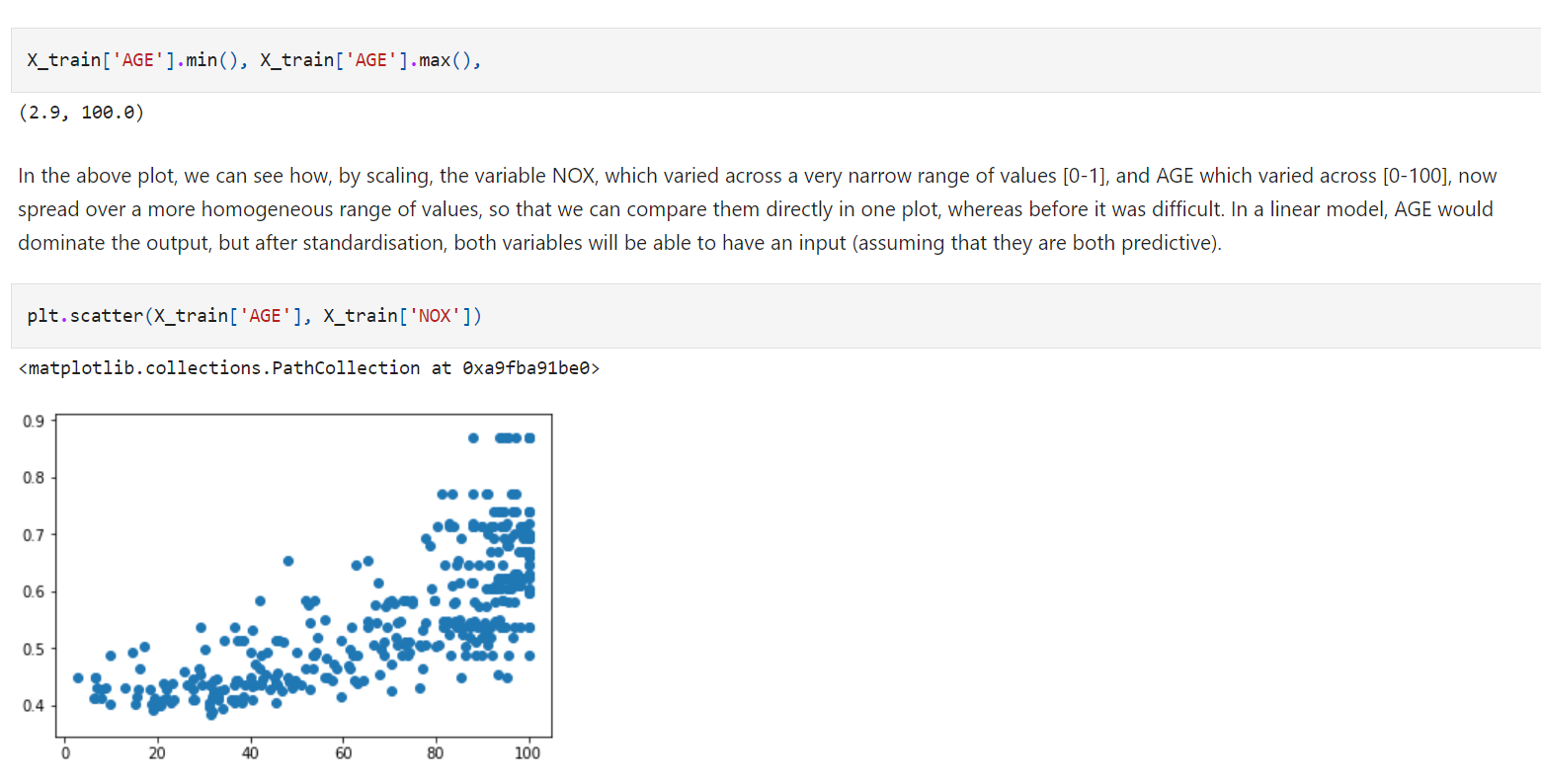


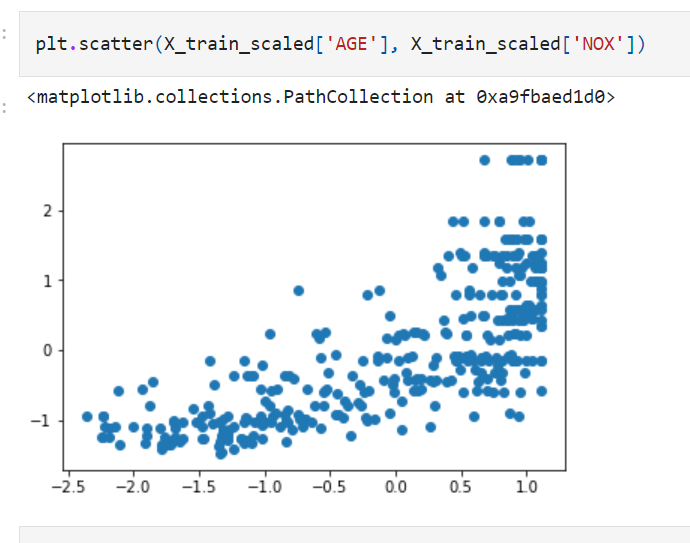


Note from the above plots how **standardisation centered all the distributions at zero**, but it **preserved their original distribution**. The value range is not identical, but it looks more homogeneous across the variables.

Note something interesting in the following plot:







In this notebook, we will discuss **Mean Normalisation**.

=================================================================

**Mean Normalisation**

Mean normalisation involves centering the variable at zero, and re-scaling to the value range. The procedure involves subtracting the mean of each observation and then dividing by difference between the minimum and maximum value:

**x\_scaled = (x - x\_mean) / ( x\_max - x\_min)**

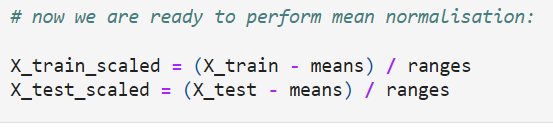
The result of the above transformation is a distribution that is centered at 0, and its minimum and maximum values are within the **range of -1 to 1.** The shape of a mean **normalised distribution will be very similar to the original distribution of the variable**, but the **variance may change**, so not identical.

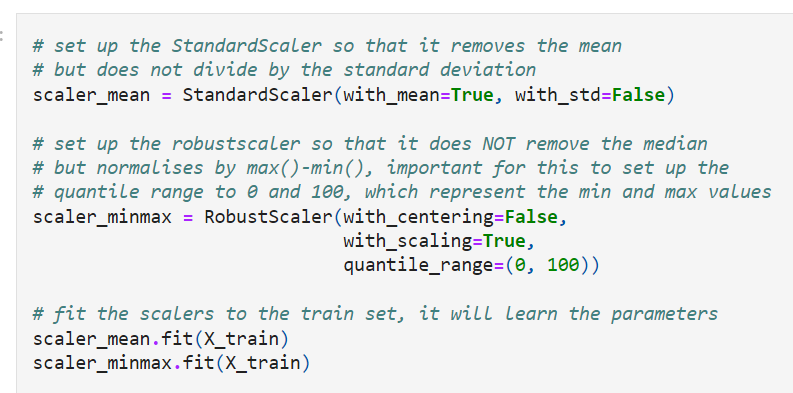
Again, this technique will not **normalize the distribution of the data** thus if this is the desired outcome, we should implement any of the techniques discussed in section 7 of the course.

In a nutshell, mean normalisation:

* centers the mean at 0
* variance will be different
* may alter the shape of the original distribution
* the minimum and maximum values squeezed between -1 and 1
* preserves outliers

Good for algorithms that require features centered at zero.





**Scaling to Minimum and Maximum values - MinMaxScaling**

Minimum and maximum scaling squeezes the values between 0 and 1. It subtracts the minimum value from all the observations, and then divides it by the value range:

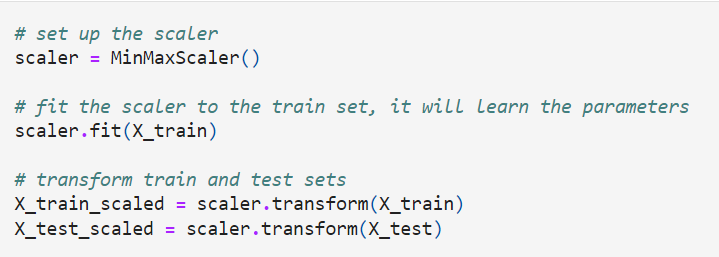
**X\_scaled = (X - X.min / (X.max - X.min)**

The result of the above transformation is a distribution which values vary within the range of 0 to 1. But the mean is not centered at zero and the standard deviation varies across variables. The shape of a min-max scaled distribution will be similar to the original variable, but the variance may change, so not identical. This scaling technique is also sensitive to outliers.

This technique will not **normalize the distribution of the data** thus if this is the desired outcome, we should implement any of the techniques discussed in section 7 of the course.

In a nutshell, MinMaxScaling:

* does not center the mean at 0
* variance varies across variables
* **may not preserve the shape of the original distribution**
* the **minimum and maximum values are 0 and 1.**



**Scaling to maximum value - MaxAbsScaling**

Maximum absolute scaling scales the data to its absolute maximum value:

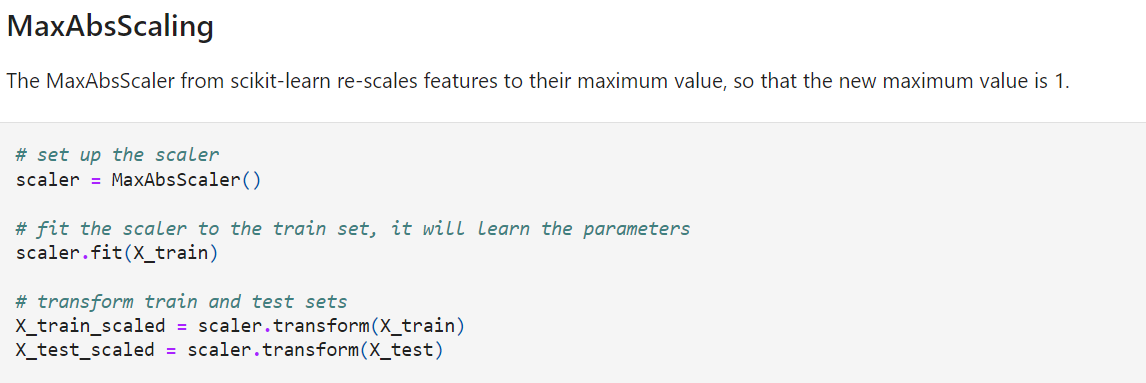
**X\_scaled = X / abs(X.max)**

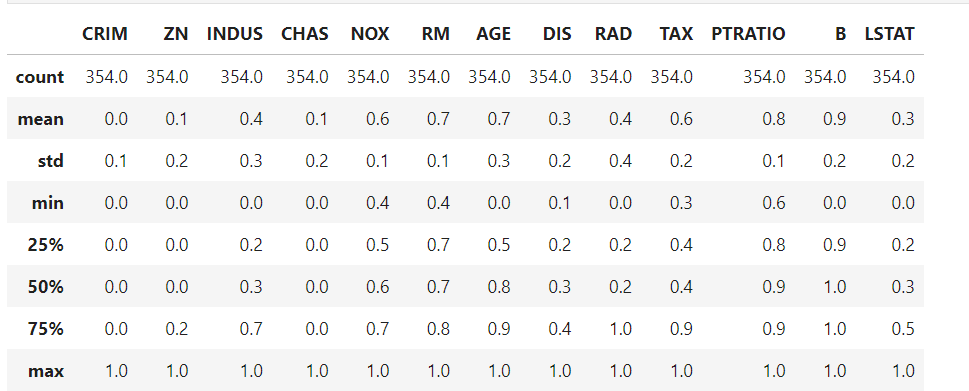
The result of the above transformation is a distribution which values vary within the range of -1 to 1. But the mean is not centered at zero and the standard deviation varies across variables.

Scikit-learn suggests that this transformer is meant for data that is centered at zero, and for sparse data.

In a nutshell, MaxAbsScaling:

* **does not center the mean at 0** (but it might be a good idea t center it with another method)
* **variance varies across variables**
* may **not preserve the shape of the original distribution**
* **sensitive outliers**





Compare the effect of different scalers on data with outliers

[Compare the effect of different scalers on data with outliers — scikit-learn 1.1.2 documentation](https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#sphx-glr-auto-examples-preprocessing-plot-all-scaling-py)

**Additional reading resources**

**Additional reading resources**

Sklearn docs

* [Compare the effect of different scalers on data with outliers](http://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#sphx-glr-auto-examples-preprocessing-plot-all-scaling-py)
* [Sklearn documentation on preprocessing and scaling](https://scikit-learn.org/stable/modules/preprocessing.html)

Other blogs and articles

* [About Feature Scaling and Normalization](https://sebastianraschka.com/Articles/2014_about_feature_scaling.html)
* [Normalization vs Standardization — Quantitative analysis](https://towardsdatascience.com/normalization-vs-standardization-quantitative-analysis-a91e8a79cebf)
* [Z scores: Use & misuse](https://influentialpoints.com/Training/z_scores_use_and_misuse.htm)
* [L0 Norm, L1 Norm, L2 Norm & L-Infinity Norm](https://medium.com/@montjoile/l0-norm-l1-norm-l2-norm-l-infinity-norm-7a7d18a4f40c)