CAPTER – 6

Feature Scaling

*Learning Topics*



* What is feature scaling
* Min-Max scaling
* Z-score normalization
* log transformation

**GitHub link:** [*Feature\_Scaling*](https://github.com/ramasureshvijjana/Data_Science/tree/master/06_Feature_Scaling)

***Def:*** Feature scaling is a technique used in machine learning to standardize (*converting large range values into small range values*) the range of features or variables of a dataset.

* Usually, the scaling range between 0 and 1 by using normal distribution, or a standard distribution with a mean of 0 and standard deviation of 1.
* The primary objective of feature scaling is to normalize the data, to remove any biases that may arise the ranges of the input features.
* Some machine learning algorithms, such as k-nearest neighbours (KNN) and support vector machines (SVM), are sensitive to the scale of input features, and the performance of the algorithm may be improved by scaling the features.
* Feature scaling required especially when the features have different scales or units in the dataset.

|  |
| --- |
| OwlThe choice of scaling technique depends on the nature of the data and the requirements of the machine learning algorithm being used. |

Some of the problems that can be reduced by feature scaling include:

* *Gradient Descent Convergence*: In gradient descent algorithms, feature scaling can help the algorithm converge faster by avoiding oscillation or overshooting during the weight update steps.
* *Distance based algorithms*: Distance-based algorithms such as K-Nearest Neighbours, Support Vector Machines (SVMs), and Principal Component Analysis (PCA) are highly sensitive to the scale of the input features. Feature scaling can help to ensure that these algorithms work correctly by bringing all the features to the same scale.
* *Regularization*: Regularization techniques such as L1 and L2 regularization assume that all features have equal importance. In case of unequal feature scales, regularization techniques may be more biased towards features with larger scales. Feature scaling can help prevent this bias.
* *Neural Networks*: In deep learning, feature scaling can help the network learn faster and more effectively by reducing the chance of vanishing gradients or exploding gradients.

**Advantages:**

* *Helps with feature selection:* The MinMaxScaler can help with feature selection by reducing the impact of features with large values. This can be important when using techniques such as linear regression or logistic regression, where large values can have a disproportionate impact on the model.

**Disadvantages:**

* *May not be appropriate for all algorithms:* The *Feature Scaling* may not be appropriate for all machine learning algorithms. For example, some algorithms, such as tree-based models, are not sensitive to the scale of the data and may not benefit from scaling.
* *Can result in loss of information:* The *Feature Scaling* can result in a loss of information, especially when the scaling range is small. This can be important when the original values of the data are important for interpretation or analysis.

**1. Min-Max Scaler:**

* The *MinMaxScaler* is a data normalization technique, that works by transforming each feature (column) values between 0 and 1, or any other specified range.
* This is achieved by subtracting the minimum value of the feature and dividing by the range (i.e., the difference between the maximum and minimum values). The resulting values will fall between 0 and 1.

The formula for the *MinMaxScaler* is:

where = original value of the feature,

= minimum value of the feature,

= maximum value of the feature,

= scaled value of the feature.

By using *MinMaxScaler*, we can arrange all features are on the same scale, which can be important for many machine learning algorithms. It can also help to avoid the issues that the domination of a feature with a large scale over other features with smaller scales.

**1.1. Working:**

* ***Identify the range of the data:*** The first step is to determine the minimum and maximum values of scaling feature in the dataset.
* ***Scale the data:*** Once the minimum and maximum values are identified, then scale the data to the desired range by using *MinMaxScaler*.
* The formula calculates the scaled value by subtracting the minimum value from the original value, and then dividing the result by the range (i.e., the difference between the maximum and minimum values).

**1.2. Mathematical Intuition Behind MinMaxScaler:**

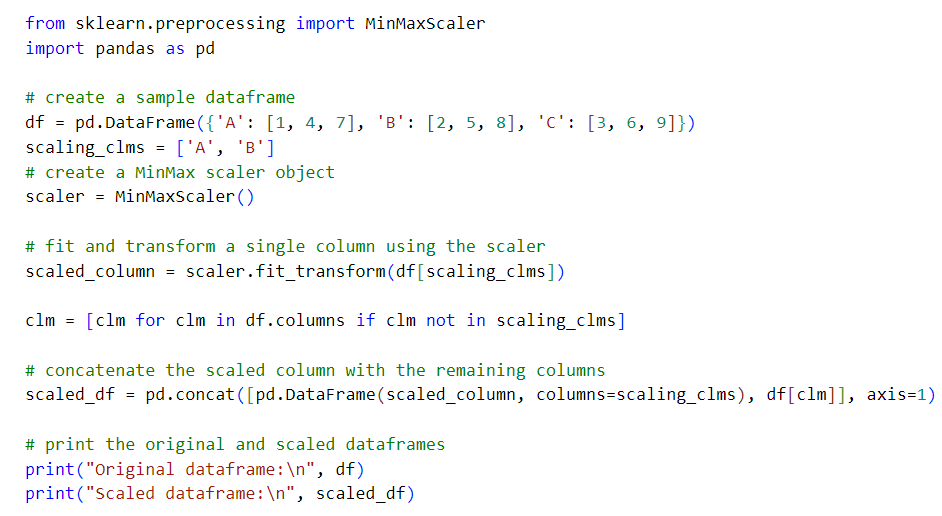
* The mathematical intuition behind MinMaxScaler is based on the concept of linear transformation.
* Linear transformation is a type of transformation that preserves the structure (i.e. relationships between variables) of the data, while scaling the data to a different range.
* In the case of MinMaxScaler, the linear transformation is used to scale the data to a specified range, typically between 0 and 1.

The formula for MinMaxScaler is:

The formula works by first subtracting the minimum value () from the original value () to obtain the "distance" of the value from the minimum (). This "distance" is then divided by the range, which is the difference between the maximum () and minimum () values of the feature. This gives a scaled value that is proportional to the original value, but scaled to the range between 0 and 1.

|  |
| --- |
| If then  If then |

**1.3. Code:**

****

**1.4. Advantages:**

* *Simple and easy to use:* The MinMaxScaler is a simple and easy-to-use technique for scaling data. It is easy to understand and implement, even for beginners.
* *Preserves the shape of the distribution:* The MinMaxScaler preserves the shape of the distribution of the data. It scales the data to a common range, but does not change the shape of the distribution. This can be important for some applications, such as when the data has a non-normal distribution.
* *Can improve model accuracy:* The *MinMaxScaler* can improve the accuracy of machine learning models, especially when the data has a clear minimum and maximum value. It can help to reduce the impact of outliers and ensure that all features are on a similar scale.

**1.5. Disadvantages:**

* *Sensitive to outliers:* The MinMaxScaler is sensitive to outliers, which can affect the scaling of the data. Outliers misleading the minimum and maximum values which means if column has outlier at max value the it will take that outlier as max value, which can reduce the effectiveness of the scaler.
* *May not work well with some distributions:* The MinMaxScaler may not work well with some distributions, such as those with a very narrow range or those with a lot of zero values. In these cases, other scaling techniques may be more appropriate.

**2. Standard Scaler**

* Standard Scaler is a pre-processing technique used to *standardize* the features of a dataset.
* It is widely used method in data pre-processing and machine learning, particularly for algorithms that assume normal distribution of the input variables.
* Standard Scaler transforms the data with mean of each feature is zero and the standard deviation is 1. This is the main mathematical logic behind the Standard Scaler.

**2.1. Working:**

The StandardScaler works by subtracting the mean from each feature and then dividing by the standard deviation.

The formula for standardizing a feature is:

where = standardized value

= original value of the feature

= mean of the feature

= standard deviation of the feature.

* Compute the mean and standard deviation of scaling feature in the dataset.
* Subtract the mean from each value in the column. This centres the data around zero.
* Divide the centre values by the standard deviation. This scales the data so that it has a standard deviation of 1.

The resulting dataset has a mean of zero and a standard deviation of one.

**2.2. Mathematical Intuition Behind StandardScaler:**

The mathematical intuition behind the StandardScaler is based on the concept of the z-score, which measures the distance of a data point from the mean in units of the standard deviation.

When we standardize a feature using the StandardScaler, we calculate the z-score for each value in the feature by subtracting the mean of the feature from the value and then dividing the result by the standard deviation of the feature. The resulting z-score represents the number of standard deviations that the value is away from the mean.

The mean and standard deviation formulas are:

The formula for standardizing a feature is:

**2.3. Code:**

|  |
| --- |
| 1. from sklearn.preprocessing import StandardScaler 2. std\_sclr = StandardScaler() 3. data['calories'] = std\_sclr.fit\_transform(data.loc[:, ['calories']]) 4. display(data) |

***Output:***

|  |  |
| --- | --- |
| ***Before standard scalar:*** | ***After standard encoder:*** |

**2.4. Advantages:**

* StandardScaler makes it easier to compare features that are measured in different units or have different scales.
* StandardScaler helps machine learning algorithms to converge more quickly, particularly gradient descent-based algorithms, by making the cost surface smoother.
* StandardScaler can help to reduce the impact of outliers, by scaling the values to be within a more reasonable range.
* StandardScaler is a simple and fast technique that can be easily applied to any dataset.

**2.5. Disadvantages:**

* StandardScaler assumes that the data is normally distributed, which may not be true in all cases. In such cases, other scaling techniques may be more appropriate.
* StandardScaler can be sensitive to outliers if the number of outliers is very large or if the outliers are very extreme. In such cases, robust scaling techniques may be more appropriate.
* StandardScaler can increase the risk of overfitting in some cases, particularly if the number of features is very large. In such cases, feature selection or regularization may be more appropriate.
* StandardScaler can lead to loss of interpretability, as the transformed features no longer have the same units as the original features.

**3. Max Absolute Scaler:**

* It rescales the feature values to the range [-1, 1] by dividing through the largest maximum absolute value in each feature.
* It is similar to MinMaxScaler, but instead of scaling the data to a fixed range [0 -1], MaxAbsScaler scales based on the absolute maximum value of the feature.
* This scaler is useful when the data contains large outliers or when the data is not normally distributed.
* MaxAbsScaler can be applied to both dense and sparse input data. It is commonly used in feature scaling for linear models, clustering algorithms, and neural networks.
* MaxAbsScaler transforms the data by making feature's absolute maximum value is 1.
* It does not shift/center the data and thus does not destroy any sparsity.

**3.1. Working of MaxAbsScaler**

1. First identify the maximum absolute value of the feature.
2. Divide each value of the feature with maximum absolute value. The resultant is the scaled value.

**3.2. Mathematical Intuition**

This scaling is important because many machine learning algorithms are sensitive to the scale of the input features, and can perform poorly if the input features are not scaled correctly.

It is a simple linear transformation that divides each value in a feature by its absolute maximum value.

The largest absolute value in the feature becomes 1 when . In this case both numerator and denominator are same in above formula.

**3.3. Code:**

|  |
| --- |
| 1. from sklearn.preprocessing import MaxAbsScaler 2. # Create a MaxAbsScaler object 3. scaler = MaxAbsScaler() 4. # Scale the data 5. data['calories'] = scaler.fit\_transform(data.loc[:,['calories']]) 6. # Print the scaled data 7. print("Scaled data:\n\n", data.head()) |

***Output:***

|  |  |
| --- | --- |
| ***Before MaxAbsScaler:*** | ***After MaxAbsScaler:*** |

**3.4. Advantages:**

* *Maintains the sign of the data:* Unlike MinMaxScaler, which shifts all the data to a positive range, MaxAbsScaler preserves the sign of the data. This is useful when the data contains negative values, as it maintains the relative distances between different values within a feature.
* *Suitable for sparse data:* MaxAbsScaler can be applied to sparse data, which is useful when working with large datasets. It preserves the sparsity of the data, and ensures that the scaling is applied only to the non-zero elements of the data.
* *Easy to interpret:* MaxAbsScaler is a simple linear transformation that is easy to understand and interpret. It is based on the absolute maximum value of each feature, which is an intuitive scaling parameter.

**3.5. Disadvantages:**

* *Sensitive to outliers:* Like all scaling methods, MaxAbsScaler is sensitive to outliers. Large outliers in the data can affect the scaling and distort the relative distances between the other values.
* *Limited scaling range:* MaxAbsScaler scales the data to a range of -1 to 1, which may not be suitable for all datasets. If the data has a wider range of values, the scaling may not be sufficient to bring all the features to the same scale.
* *Not suitable for non-linear data:* MaxAbsScaler is a linear transformation, and may not be suitable for datasets with non-linear relationships between the input features and the target variable. In such cases, non-linear scaling methods such as logarithmic or exponential scaling may be more appropriate.

**4. Robust Scaler**

* The Robust scaler is similar to Standard scaler, but Standard scaler uses ***mean and standard deviation***. Robust scaler uses ***median and interquartile range (IQR)***.
* The median is a measure of central tendency that is less sensitive to outliers than the mean, while the IQR is a measure of the spread of the data that is also less sensitive to outliers than the standard deviation.
* This formula centers the data around the median and scales it by the IQR. The resulting dataset has a similar scale to the standardization (mean = 0, standard deviation = 1) but is more robust to outliers.
* The meaning of Robust scaling is “***distance of a data point from the median in units of the IQR***”. Which means ***How many IQRs required to get the distance from a data point to median***?.
* This scaler working internally by setting the median as 0 and IQR as 1.
* The Robust Scaler doesn't explicitly set predefined minimum and maximum values like some other scaling methods such as Min-Max Scaling. Instead, it focuses on centering the data by subtracting the median and scaling it by the interquartile range (IQR).

**4.1. Working:**

1. First calculate Median of scaling feature.
2. Calculate the interquartile range of the feature.
3. Substitute median and IQR in above formula for each and every value of the feature then the resultant is the robust scaled value.

**4.2. Mathematical Intuition:**

Internally, the Robust Scaler achieves robustness by centering the data based on the median and scaling it based on the interquartile range (IQR). Here's how it works:

**1. Centering using Median (location shift):**

* The median is a measure of central tendency that represents the middle value of a dataset when it is sorted in ascending order.
* The Robust Scaler subtracts the median of the feature from each data point in that feature.
* This operation effectively centres the data, making the median of the feature become 0 after scaling.

**2. Scaling using IQR (scale shift):**

* The interquartile range (IQR) is a measure of statistical dispersion, representing the range between the first quartile (25th percentile) and the third quartile (75th percentile) of the data.
* The Robust Scaler then divides each data point in the feature by the IQR.
* This operation scales the data by the spread of the middle 50% of the data, making the IQR of the feature equal to 1 after scaling.

The formula for the Robust Scaler transformation for a given feature X is:

**3. Outlier Effect:**

The robust scalar is not sensitive to outliers but standard scalar is more sensitive to outliers. To understand this better by understand the mean and median calculations as follow:

Consider the list of input numbers below.

X = {1,2,3,4, 5, 500} Here 500 is the outlier.

***Mean:***

The mean of X is:

Now the X mean is , this mean very far from first 5 values and it is affected by outlier 500.

***Median:***

* To find the median, we first sort the list. Median is the middle value that splits the list in half.
* The list above is already sorted. Thus its median is 3.5

The outlier doesn’t affect the median. That’s because the median doesn’t depend on every value in the list. The last value could have been 500 or even 10000. And it wouldn’t change the median at all.

**4.3. Code:**

|  |
| --- |
| 1. from sklearn.preprocessing import RobustScaler 2. # create a RobustScaler object 3. scaler = RobustScaler() 4. # fit the scaler to the data and transform it 5. data['calories'] = scaler.fit\_transform(data.loc[:,['calories']]) 6. # print the original and scaled data 7. print("Scaled data:\n\n", data.head()) |

***Output:***

|  |  |
| --- | --- |
| ***Before RobustScaler:*** | ***After RobustScaler:*** |

**4.3. Advantages:**

* *Robustness to outliers:* The Robust Scaler is less sensitive to outliers than other scaling methods, such as standardization, which makes it useful in cases where there are extreme values in the data.
* *Preserves the distribution shape:* The Robust Scaler preserves the distribution shape of the original data, unlike methods like standardization that can change the distribution shape.
* *Not affected by the range of values:* The Robust Scaler is not affected by the range of values in the data, making it suitable for use with datasets that have a wide range of values.

**4.4. Disadvantages:**

* *Data transformation:* The Robust Scaler transforms the data, which can make it difficult to interpret the original values after scaling. This can be a disadvantage in some applications where the original values need to be preserved.
* *May not work well with small datasets:* The Robust Scaler uses the median and interquartile range, which may not be as reliable with small datasets. In such cases, the standardization method may be more suitable.
* *Sensitivity to the shape of the distribution:* The Robust Scaler is sensitive to the shape of the distribution of the data, and may not work as well if the distribution is not symmetric. In such cases, other scaling methods such as normalization may be more suitable.

**5. Log transformation** **scale**

* Log transformation is a mathematical transformation that is used to reduce the magnitude of values and to make highly skewed data more suitable for analysis or modelling.
* The log transformation involves taking the logarithm of each value in a dataset, which compresses the range of the data and can make patterns more visible.
* There are two common types of log transformation: natural logarithm (log base e) and base-10 logarithm. The natural logarithm is commonly used because it has certain mathematical properties that make it convenient for analysis.

**5.1. Working**

The log transformation is a mathematical transformation that involves taking the logarithm of each value in a dataset. In the case of the natural logarithm (log base e), the formula for the transformation is:

or

1. Substitute each value of a feature into above formula and get the scaled value “y”.

The log transformation is useful in situations where the data is highly skewed, such as in financial or economic data, where the range of values can vary widely and outliers can skew the data. By taking the logarithm of each value, the range of values is compressed, and the distribution of the data can be brought closer to a normal distribution.

**5.2. Mathematical Intuition**

The mathematical intuition behind the log transformation is that it compresses the range of values in a dataset by applying a logarithmic function to each value.

The natural logarithm (log base e) is commonly used in the log transformation because it has certain mathematical properties that make it useful for analysis. For example, the natural logarithm is a continuous, increasing function that maps positive values to negative values, and it is able to handle both large and small values.

One important property of the log transformation is that it has the effect of reducing the impact of extreme values, such as outliers.

Another important property of the log transformation is that it can be used to linearize relationships between variables. For example, if there is a non-linear relationship between two variables in a regression model, applying a log transformation to one or both of the variables can help to linearize the relationship and make it more suitable for analysis.

**5.3. Code:**

|  |
| --- |
| 1. import numpy as np 2. data['calories'] = np.log(data.loc[:,['calories']]) 3. data.head() |

***Output:***

|  |  |
| --- | --- |
| ***Before RobustScaler:*** | ***After RobustScaler:*** |

**5.3. Advantages:**

* Normalization: Log transformation can help to normalize the data by reducing the skewness in the data distribution, which is especially useful when dealing with data with long tails and extreme values.
* Data Compression: Log transformation can compress the data, making it easier to interpret and visualize trends and patterns.
* Multiplicative Effects: Log transformation can help to handle multiplicative effects in data. For example, in finance, a log transformation can be used to convert percentage changes to additive changes.
* Homogeneity of Variance: Log transformation can help to stabilize the variance in the data, making it more homogeneous across different groups.

**5.4. Disadvantages:**

* Interpretation: It can be challenging to interpret the results of data that has been transformed logarithmically, as the units of measurement are no longer easily interpretable.
* Data Loss: Depending on the nature of the data, some information can be lost during log transformation. For example, negative values cannot be transformed logarithmically.
* Outliers: Log transformation can be highly affected by outliers, which can distort the data and its interpretation.
* Zero Values: Zero values can be problematic when using a logarithmic transformation, as it is impossible to take the logarithm of zero.
* Negative Values: if the data negative values, the log transformation may not be applicable.