Minería de Datos

U3 Preprocesamiento de Datos

Héctor Maravillo

Section 1

Data Normalization

Data Normalization

The data collected in a data set may not be useful enough for a DM algorithm.

Sometimes the attributes selected are **raw attributes** that have a meaning in the original domain from where they were obtained, but they are not good enough to obtain accurate predictive models.

Therefore, it is common to perform a series of manipulation steps **to transform** the original attributes or **to generate** new attributes with better properties that will help the predictive power of the model.

The new attributes are usually named **modeling variables** or **analytic variables**.

Data Normalization

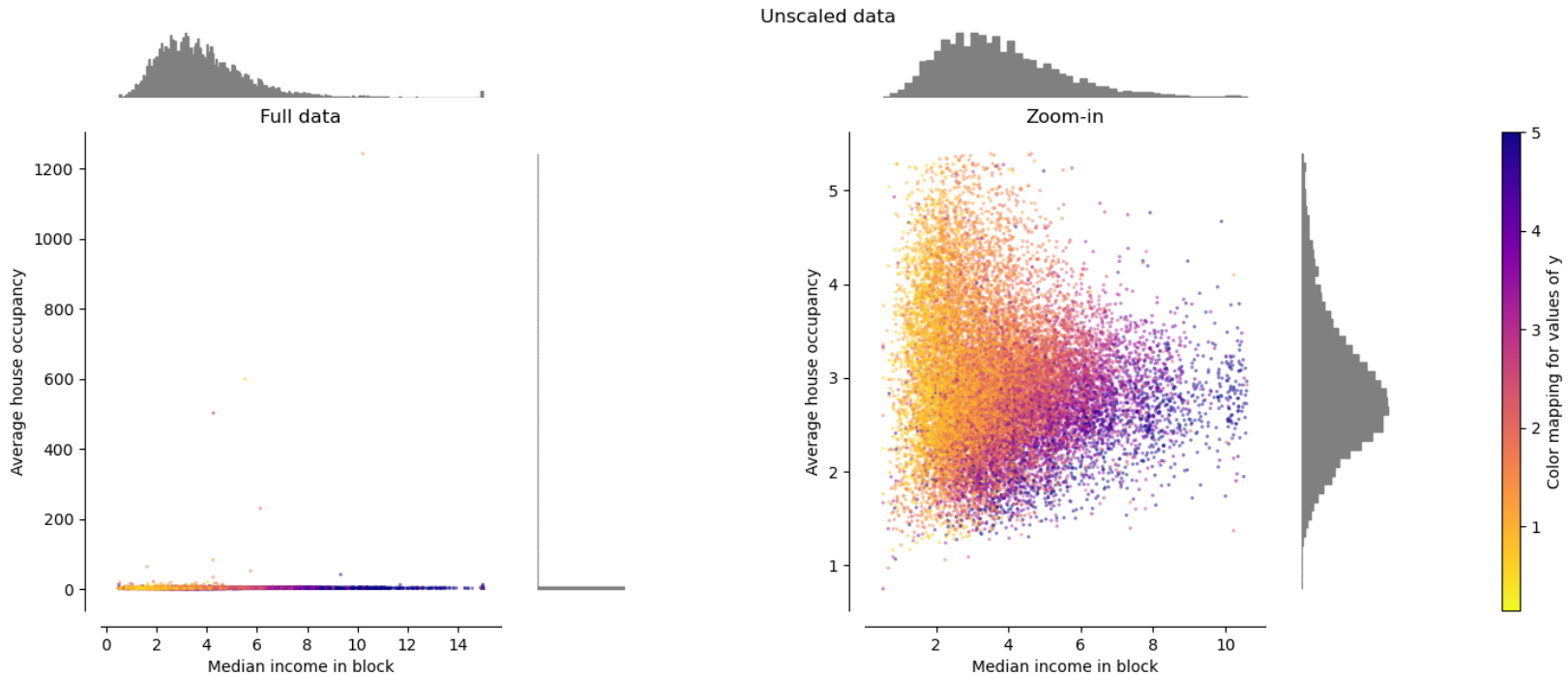

Figure: Original data

Min-Max scaling

The **min-max scaling** aims to **scale** all the numerical values x of a numerical attribute A to a specified range denoted by $[a, b]$.

Thus a **transformed value** is obtained by applying the following expression to x in order to obtain the new value $x^{'}$:

$$x^{'} = \frac{x - \min_A}{\max_A - \min_A} (max_{new} - min_{new}) + min_{new}$$

[0,1]

where $\max_A$ and $\min_A$ are the original maximum and minimum attribute values respectively.

The **new range** of the data is $[min_{new}, max_{new}]$.

Min-Max scaling

Min-Max scaling **doesn't reduce the effect of outliers**, but it **linearly scales** them down into a **fixed range**, where the largest occurring data point corresponds to the maximum value and the smallest one corresponds to the minimum value.

In the literature **scaling** usually refers to a particular case of the min-max scaling in which the final interval is $[0, 1]$. In this case, the transformation becomes

$$x^{'} = \frac{x - \min_A}{\max_A - \min_A}$$

The interval $[-1, 1]$ is also typical when scaling the data.

Min-Max scaling

This type of normalization is very common in those data sets being prepared to be used with **learning methods based on distances**.

Using a scaling to **rescale** all the data to the **same range** of values will avoid those attributes with a large $\max_A - \min_A$ difference **dominating** over the other ones in the distance calculation, misleading the learning process by giving more importance to the former attributes.

This scaling is also known for speeding up the learning process in ANNs, helping the weights to converge faster.

Min-Max scaling

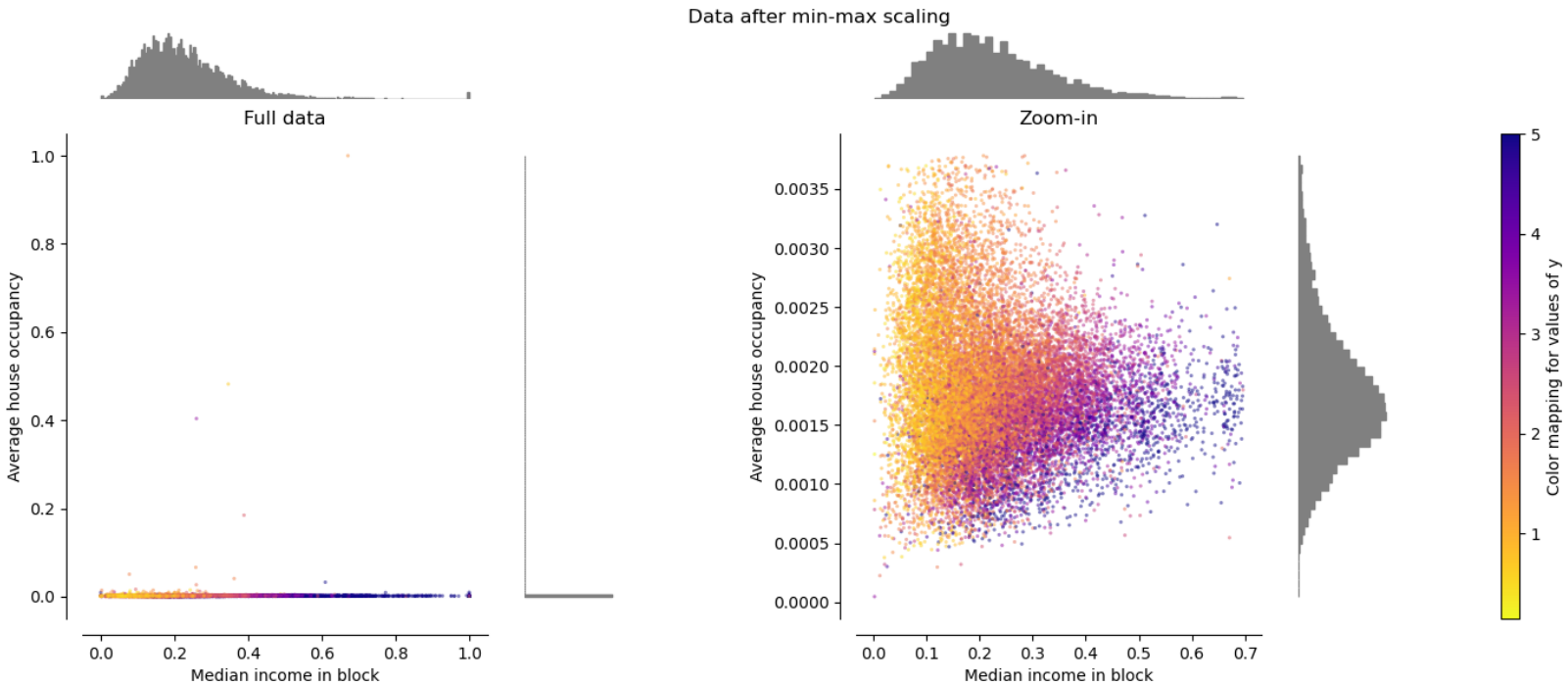

Figure: Min-Max scaling

Max-abs scaling

Max-Abs scaling works in a very similar fashion, but scales in a way that the training data lies within the range $[-1, 1]$ by dividing through the **largest maximum** value in each feature.

$$x^{'} = \frac{x}{\max_{|A|}}$$

It is meant for data that is already **centered at zero** or **sparse data**.

Max-abs scaling

Max-abs scaling is similar to MinMaxScaler except that the values are mapped across several ranges depending on whether negative or positive values are present.

- If only positive values are present, the range is [0, 1].
- If only negative values are present, the range is [-1, 0].
- If both negative and positive values are present, the range is [-1, 1].

Z-score standardization

In some cases, the **min-max scaling** is not useful or cannot be applied:

- When the minimum or maximum values of attribute A **are not known**, the min-max scaling is infeasible.
- The presence of **outliers** can bias the min-max scaling by grouping the values and limiting the digital precision available to represent the values.

Z-score standardization

If $\bar{A}$ is the mean of the values of attribute A and σ_A is the standard deviation of A, original value x of A is **standardized** to $x^{'}$ using

$$x^{'} = \frac{x - \bar{A}}{\sigma_A}$$

By applying this transformation, the attribute values now present a **mean** equal to 0 and a **standard deviation** of 1.

Z-score standardization

If the mean and standard deviation associated to the probability distribution are not available, it is usual to use instead the **sample mean**

$$\bar{A} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

and **sample standard deviation**

$$\sigma_A = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{A})^2}$$

Z-score standardization

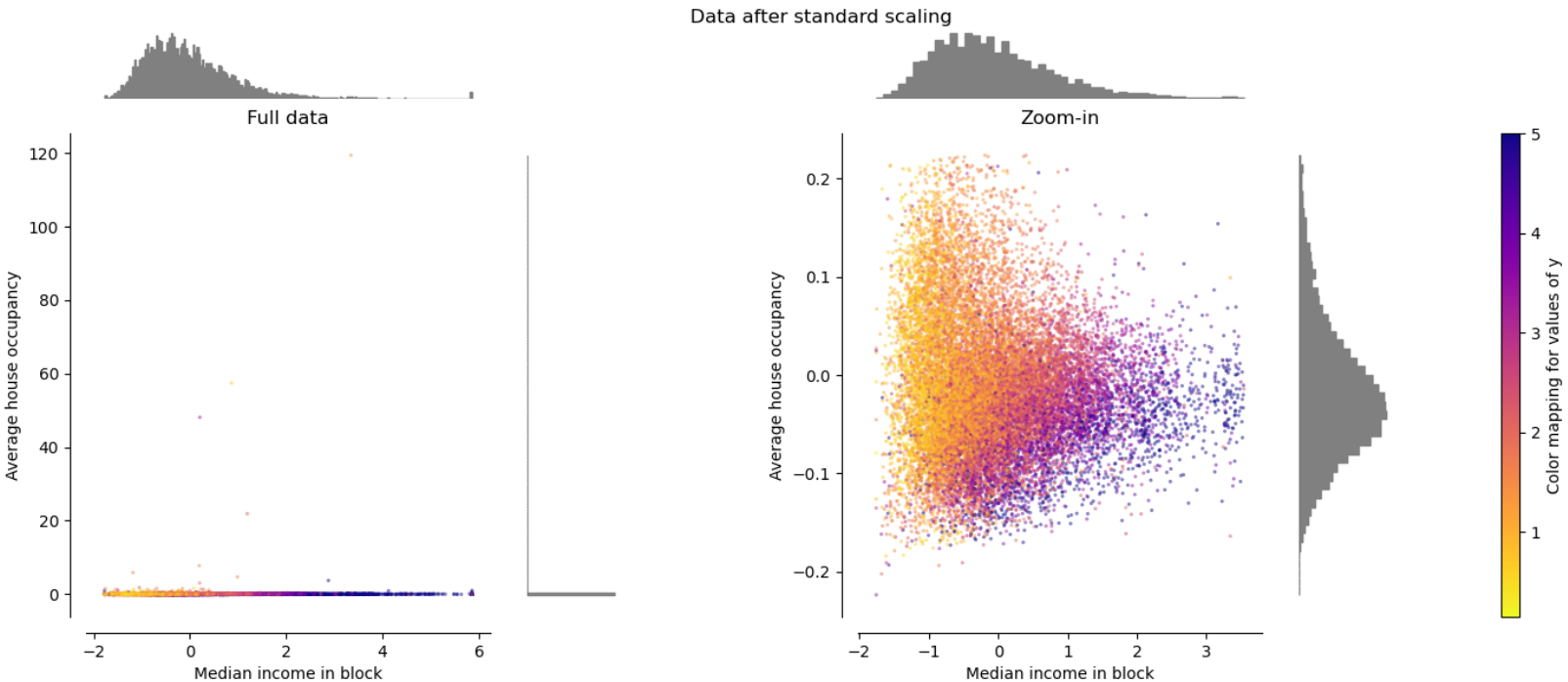

Figure: Z-score standardization

Z-score standardization

A variation of the z-**score standardization**, uses the **mean absolute deviation** s_A of A instead of the **standard deviation**.

It is computed as

$$s_A = \frac{1}{n} \sum_{i=1}^{n} |x_i - \bar{A}|$$

As a result, the **modified** z-**score** now becomes:

$$x^{'} = \frac{x - \bar{A}}{s_A}$$

An advantage of the s_A mean absolute deviation is that it is more **robust to outliers** than the standard deviation σ_A as the deviations from the mean calculated by $|x_i - A|$ are not squared.

Robust scaling

The **robust scaler** removes the **median** and scales the data according to a **quantile range**. Usually the **Interquartile Range (IQR)**, which is the range between the 1st quartile (25th percentile) and the 3rd quartile (75th percentile).

The robust scaler $x^{'}$ is

$$x^{'} = \frac{x - A_{med}}{Q_3(A) - Q_1(A)} = \frac{x - A_{med}}{IQR(A)}$$

where $Q_1(A)$, A_{med} and $Q_3(A)$ are the 25th, 50th and the 75th percentiles of the attribute A.

Robust scaling

If the data contains **many outliers**, scaling using the mean and variance of the data is likely to not work very well. In these cases, **robust scaling** can be used as a replacement. It uses more **robust estimates** for the **center** and **range** of the data.

The centering and scaling statistics of robust scaler are based on percentiles and are therefore not influenced by a small number of very large marginal outliers.

Robust scaling

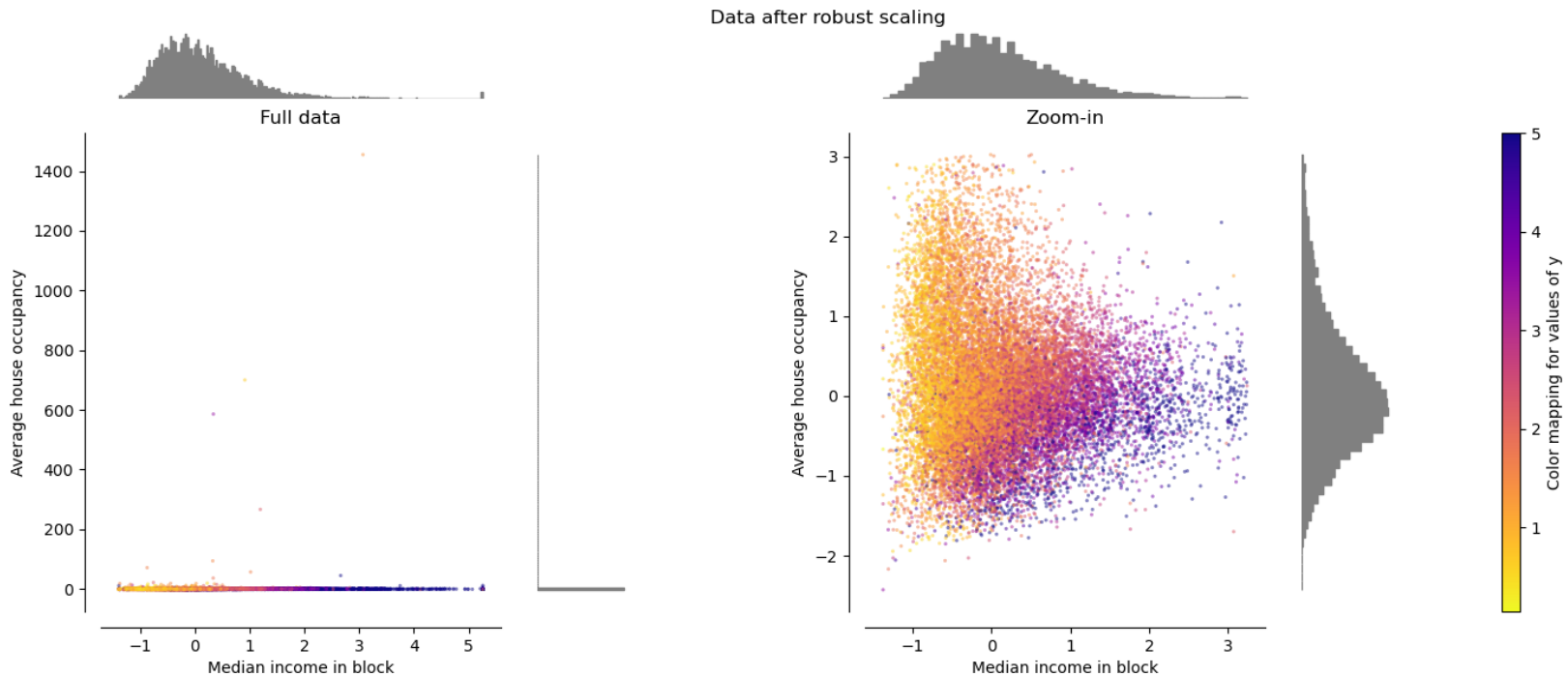

Figure: Robust scaling

Normalization

Normalization is the process of **scaling individual samples** to have **unit norm**. This process can be useful if you plan to use a quadratic form such as the dot-product or any other kernel to quantify the similarity of any pair of samples.

Let X be the **attribute vector** of an individual sample, the then the normalization formula is

$$X' = \frac{X}{||X||}$$

Where $|| \cdot ||$ is a norm, usually the Euclidean or L_2 norm.

Normalization

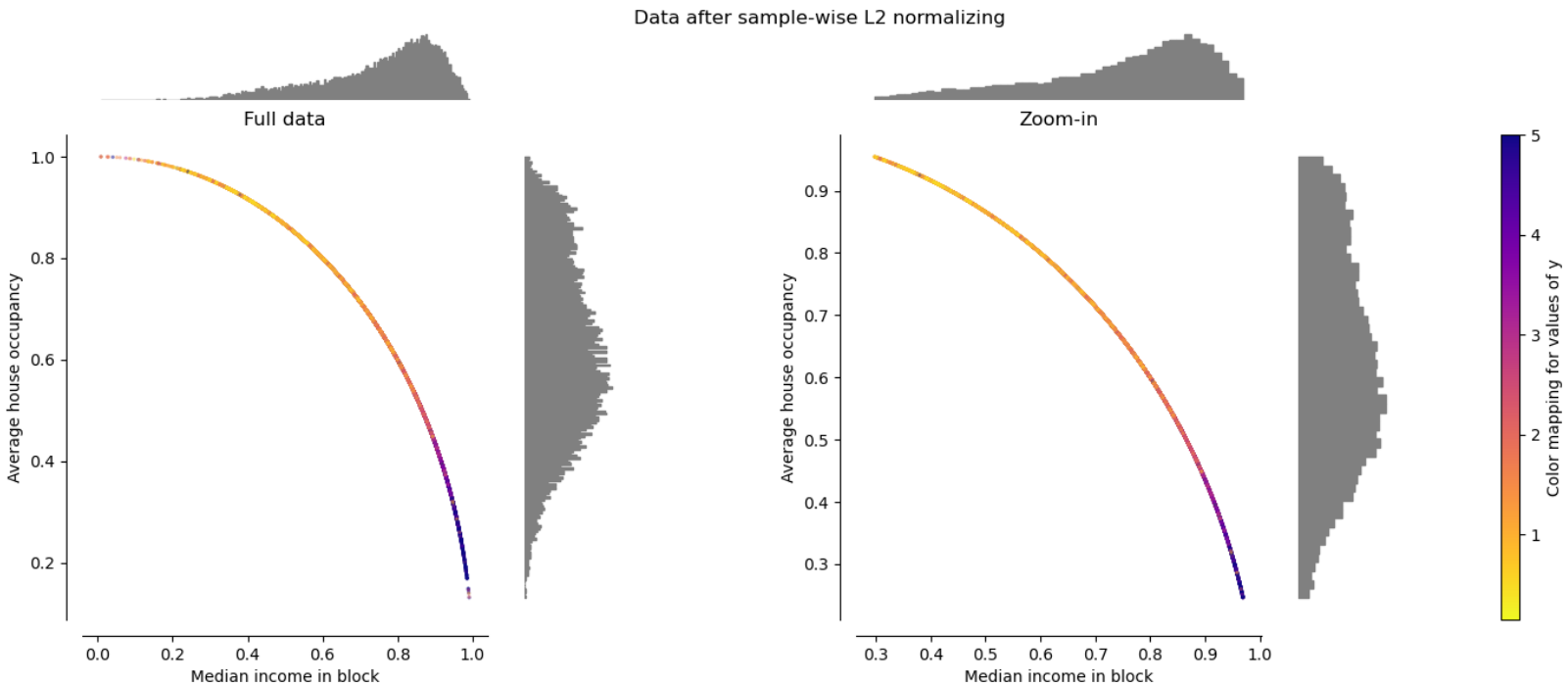

Figure: Normalization

Example

ID	A1303	edad	a0108
1	3	66	9
2	3	67	15
3	6	28	3
4	4	38	2
5	15	58	2
6	2	36	5
7	3	41	20
8	10	46	5
9	5	45	12
10	2	80	10

Section 2

Data Transformation

Data Transformation

The process to create new attributes is called **data transformation** or **attribute set**.

Data transformation usually combines the original raw attributes using different mathematical formulas originated in business models or pure mathematical formulas.

Linear Transformations

In certain areas of science, such as machine control, normalizations may not be enough to adapt the data to improve the generated model.

In these cases, **aggregating** the information contained in various attributes might be beneficial.

A family of simple methods that can be used to this purpose are **linear transformations**. They are based on simple algebraic transformations such as sums, averages, rotations, translations and so on.

Linear Transformations

Let $A = A_1, A_2, \ldots, A_n$ be a set of attributes, let $B = B_1, B_2, \ldots B_m$ be a subset of the complete set of attributes A.

If the following expression is applied

$$Z = r_1 B_1 + r_2 B_2 + \ldots + r_m B_m = \sum_{i=1}^{m} r_i B_i = r^t B$$

a new derived attribute is constructed by taking a **linear combination** of attributes in B.

A special case arises when the m values are set to $r_1 = r_2 = \ldots = r_m = \frac{1}{m}$ that situation averages the considered attributes in B:

$$Z = (B_1 + B_2 + \ldots + B_m)/m$$

Quadratic Transformations

In **quadratic transformations** a new attribute is built as follows:

$$\begin{aligned} Z &= r_{1,1}B_1^2 + r_{1,2}B_1B_2 + \ldots + r_{m-1,m}B_{m-1}B_m + r_{m,m}B_m^2 \\ &= B^T RB \end{aligned}$$

where $r_{i,j}$ is a real number, $B = (B_1, B_2, \ldots, B_m)$ and $R = [r_{i,j}]$ is a $m \times m$ matrix.

These kinds of transformations have been studied extensively and can help to transform data to make it separable.

Example

Let us consider a set of conic sections described by the coefficients of the algebraic expression that follows:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

From this equation there is no obvious interpretation that can be used to label the tuples to any type of conic section.

A	B	C	D	E	F	Conic
1	0	42	2	34	-33	
-5	0	0	-64	29	-68	
88	-6	-17	-79	97	-62	

Example

Computing the quadratic transformation known as the discriminant given by

$$Z = (B^2 - 4AC)$$

the sign of Z provides enough information to correctly label the tuples.

A	B	C	D	E	F	Conic	Sign
1	0	42	2	34	-33	Hyperbola	-
-5	0	0	-64	29	-68	Parabola	0
88	-6	-17	-79	97	-62	Ellipse	+

Although this example indicates that transformations are necessary for knowledge discovery in certain scenarios, we usually do not have any clue on how such transformation can be found and when should they be applied.

Log-Transformation

Log-Transformation refers to a data transformation process where each variable x is replaced with its logarithm

$$x^{'} = \log(x)$$

such as base 10, base 2, or natural log ln.

This process is useful for **compressing** the y-axis when plotting histograms, making visualization clearer.

Log transformation also **de-emphasizes outliers** and allows us to potentially obtain a bell-shaped distribution. The idea is that taking the log of the data can restore **symmetry** to the data.

Nominal to Binary Transformation

The presence of **nominal attributes** in the data set can be problematic, especially if the DM algorithm used cannot correctly handle them. This is the case of SVMs and ANNs

The first option is to transform the nominal variable to a numeric one, in which **each nominal value** is **encoded** by an integer, typically starting from 0 or 1 onwards.

Ordinal Encoding

Original Data

ID	Color
1	Red
2	Blue
3	Green
4	Red
5	Green
6	Blue

Ordinal Encoding

ID	Encoded Color
1	2
2	0
3	1
4	2
5	1
6	0

Nominal to Binary Transformation

Although simple, this approach has two big drawbacks that discourage it:

- With this transformation we assume an **ordering** of the attribute values, as the integer values are ranked. However, the original nominal values did not present any ranking among them.
- With this nominal to integer transformation we are establishing **unequal differences** between pairs of nominal values, which is not correct.

One-hot encoding

In order to avoid the afore mentioned problems, a very typical transformation used for DM methods is **to map each nominal attribute** to a set of **newly generated attributes**.

If N is the number of **different values** the nominal attribute has, we will substitute the nominal variable with a new set of **binary attributes**, each one representing one of the N possible values.

This transformation is also referred in the literature as 1-**to**-N **transformation** or **one-hot encoding.**

One-Hot Encoding

Original Data

ID	Color
1	Red
2	Blue
3	Green
4	Red
5	Green
6	Blue

One-Hot Encoding

ID	Red	Blue	Green
1	1	0	0
2	0	1	0
3	0	0	1
4	1	0	0
5	0	0	1
6	0	1	0

Data reduction

The new set of attributes are **linearly dependent**. That means that one of the attributes can be dismissed without loss of information as we can infer the value of one of the new attributes by knowing the values of the rest of them.

A problem with this kind of transformation appears when the original nominal attribute has a **large cardinality**.:

- In this case, the number of attributes generated will be large as well, resulting in a very **sparse data** set which will lead to numerical and performance problems.

Data reduction

When the data set is very large, performing complex analysis and DM can take a long computing time.

Data reduction techniques are applied in these domains to reduce the size of the data set while trying to maintain the integrity and the information of the original data set as much as possible.

References

- Scikit-Learn, *Preprocessing data*. `https://scikit-learn.org/stable/modules/preprocessing.html#references`
- García, S., Luengo, J. & Herrera, F. *Data Preprocessing in Data Mining*, Springer, 2015.