Title: Data Transformations

Course: Data Mining

Instructor: Claudio Sartori

Date: 13/04/2023

Master: Data Science and Business Analytics

Academic Year: 2022/2023

BOLOGNA BUSINESS SCHOOL

Alma Mater Studiorum Università di Bologna

Why do we need type conversion?

- Many algorithms require numeric features
  - categorical features must be transformed into numeric
  - ordinal features must be transformed into numeric, and the order must be preserved
- Classification requires a target with nominal values
  - a numerical target can be discretised
- Discovery of association rules require boolean features
  - a numerical feature can be discretised and transformed int a series of boolean features

Nominal to numeric

- One–Hot–Encoding
  - a feature with V unique values is substituted by V binary features each one corresponding to one of the unique values
  - if object x has value $v$in feature d then the binary feature corresponding to v has `True` for x, all the other binary features have value `False`
- `sklearn.preprocessing.OneHotEncoder`

link to manual page

Ordinal to numeric

- The ordered sequence is transformed into consecutive integers
  - by default the lexicographic order is assumed
  - The user can specify the proper order of the sequence
- `sklearn.preprocessing.OrdinalEncoder`

link to manual page

Numeric to binary with threshold

- Not greater than the threshold becomes zero
- Greater than the threshold becomes one
- `sklearn.preprocessing.Binarizer`

link to manual page

Numeric to k values

- The numbers are discretised into a sequence of integers 0 to $k-1$
- Several strategies are available
  - {'uniform','quantile','means'}
- `sklearn.preprocessing.KBinsDiscretizer`

link to manual page

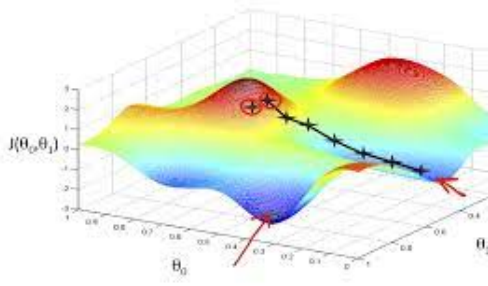
Why Data Transformation

- the features may have different scales
  - this can alterate the results of many learning techniques
  - some machine learning algorithms are sensitive to feature scaling while others are virtually invariant to it
- there can be outliers

Gradient descent

Machine learning algorithms that use *gradient descent* as an optimization technique require data to be scaled

- e.g. linear regression, logistic regression, neural network, etc.
- The presence of feature value X in the formula will affect the step size of the gradient descent
- The difference in ranges of features will cause different step sizes for each feature.
- Similar ranges of the various features ensure that the gradient descent moves smoothly towards the minima and that the steps for gradient descent are updated at the same rate for all the features



Distance–based algorithms

- KNN, K–Means, SVM, ...
- distances between points are used to determine their similarity

Example

Original data		
Student	**CGPA**	**Salary**
A	3	60
B	3	40
C	4	40
D	4.5	50
E	4.2	52

Scaled data (`StandardScaler`)		
Student	**CGPA**	**Salary**
A	-1.18431	1.520013
B	-1.18431	-1.100699
C	0.41612	-1.100699
D	1.21635	0.209657
E	0.736212	0.471728

Distances before and after scaling

$$
\begin{aligned}
distance(A, B) &= \sqrt{(40-60)^2 + (3-3)^2} = 20 \\
distance(B, C) &= \sqrt{(40-40)^2 + (4-3)^2} = 1 \\
distance(A_s, B_s) &= \sqrt{(1.1+1.5)^2 + (1.18-1.18)^2} = 2.6 \\
distance(B_s, C_s) &= \sqrt{(1.1-1.1)^2 + (0.41+1.18)^2} = 1.59
\end{aligned}
$$

Before the scaling the two distances seemed to be very different, due to the a big numeric difference in the `Salary` attribute, now they are comparable

Range–based scaling and standardization

operate on single features

- Range–based scaling stretches/shrinks and translates the range, according to the range of the feature (there are some variants)
  - good when we know that the data are not gaussian, or we do not make any assumption on the distribution
  - the base variant, the `MinMaxScaler`, remaps to $0, 1$
- Standardization subtracts the mean and divides by the standard deviation
  - the resulting distribution has mean *zero* and *unitary* standard deviation
  - ideal when the distribution is gaussian
  - `StandardScaler`

Range–based scalers in Scikit-Learn

affine transformations: linear transformation plus translation

- `MinMaxScaler` – remaps the feature to $[0, 1]$
- `RobustScaler` – centering and scaling statistics is based on percentiles
  - not influenced by a few number of very large marginal outliers
  - the resulting range of the transformed feature values is larger than the one given by `MinMaxScaler` and `StandardScaler`

Normalization

- Normalization is mentioned sometimes with different meanings
  - frequently it refers to `MinMaxScaler`
- in `Scikit-learn` the `Normalizer` normalizes each data row to unit norm

Workflow

1. transform the features as required both for the train and test data
2. fit and optimize the model(s)
3. test
4. possibly, use the original data to plot relevant views (e.g. to plot cluster assignments)

Imbalanced data in classification

- The performance minority class (classes) has little impact on standard performance measures
- The optimised model could be less effective on minority class (classes)
- Some estimators allow to weight classes
- Some performance measures allow to take into account the contribution of minority class (classes)

Cost Sensitive learning

Already introduced in *Machine-Learning-03-classification*

- several classifiers have the parameter `class_weight`
- it changes the cost function to take into account the imbalancing of classes
- in practice it is equivalent to oversampling the minority class, (repeating random examples) in order to produce a balanced training set

Undersampling

- Obtains a balanced training set by randomly reducing the number of examples of the majority class
- Obviously part of the knowledge embedded in the training set is dropped out

Oversampling with SMOTE

Synthetic Minority Oversampling Technique

- synthesize new examples from the minority class
- a type of data augmentation
- a random example from the minority class is first chosen
- k of the nearest neighbors are found
- a randomly selected neighbor is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space

Workflow for *undersampling/oversampling*

- resample the training set
- fit and optimise the estimator
- test the fitted estimator on the test set (untouched)