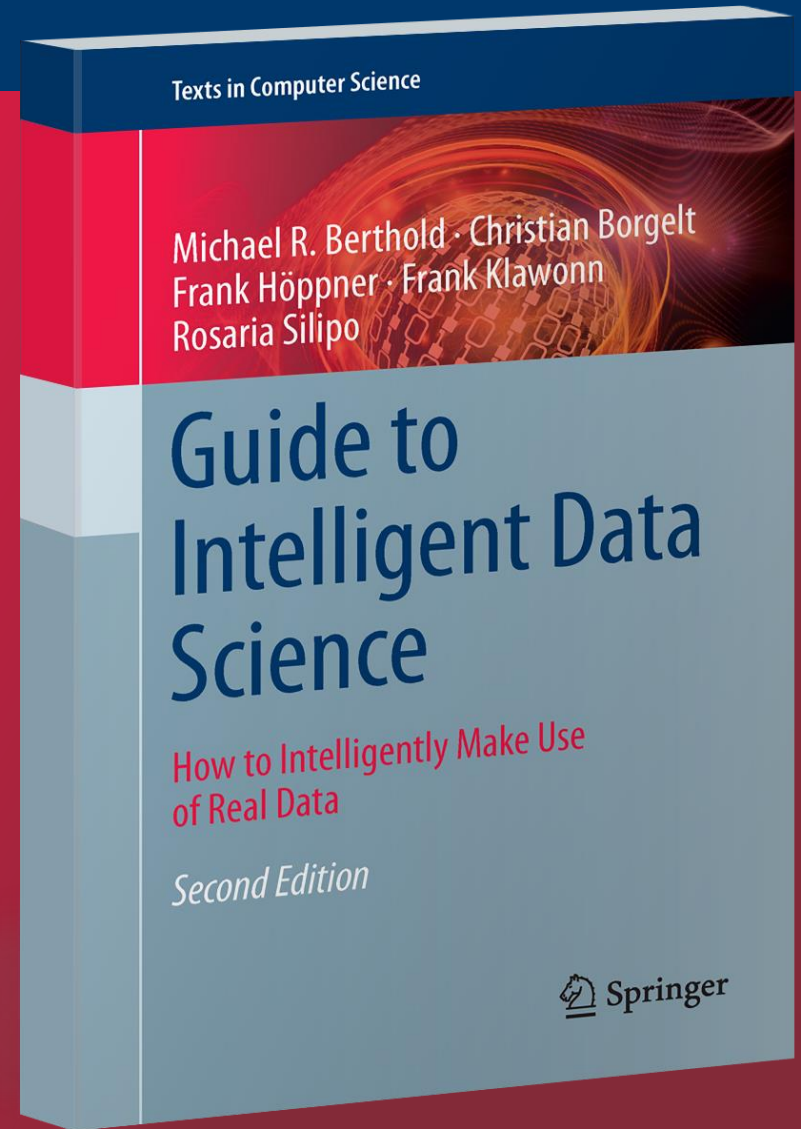


Data Preparation

Caption



*„Success is where preparation and opportunity meets“
-Bobby Unser*

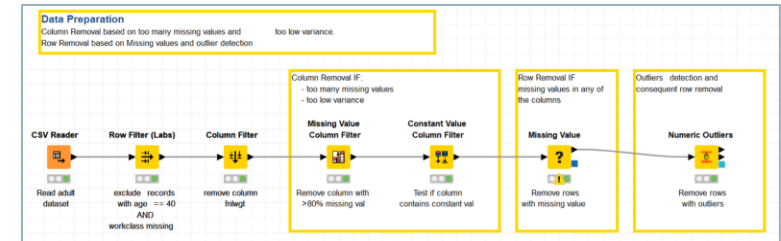
How can we prepare our data for analysis?

**This lesson refers to chapter 6 of the GIDS book*

- Select data
- Clean data
- Construct data
- Data Integration
- Practical example

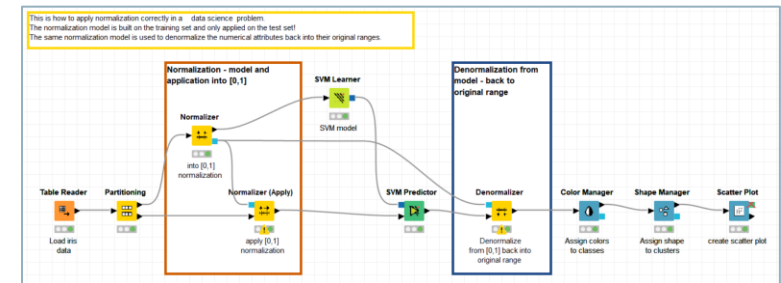
- Datasets used: adult dataset and iris dataset
- Example Workflows:
 - „Dimensionality Reduction & Outlier Detection“ <https://kni.me/w/3eRugmiCNg6uSmOz>

- Manual filter
- Missing and constant value column filter
- Missing value handling
- Outliers handling



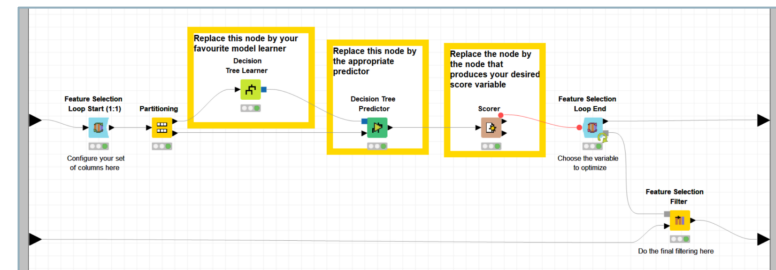
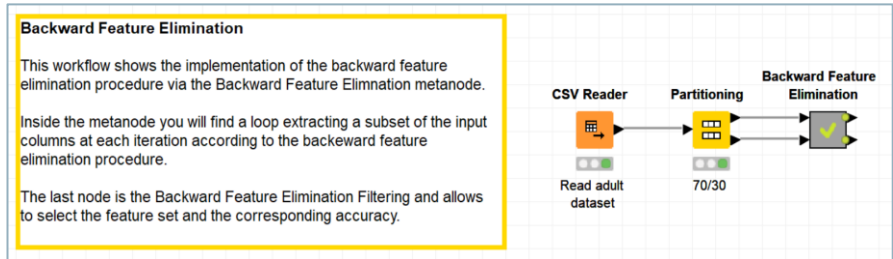
- “How to use normalization in a data science solution” <https://kni.me/w/l9AcqvFQbXp3AfNS>

- Build normalization model
- Apply normalization model
- Denormalization



Datasets and workflows

- Datasets used: adult dataset and iris dataset
- Example Workflows:
 - „Backward Feature Elimination“ <https://kni.me/w/Cim28Kbg7Rt2EsE8>



Data Preparation

Data understanding

- Provides general information about the data, such as
 - the existence and partly also about the character of missing values,
 - outliers,
 - the characteristics of attributes
 - dependencies between attributes

Data preparation

- Uses information from the data understanding process to
 - select attributes,
 - reduce the dimensionality of the data set,
 - select records,
 - (try to) handle missing values,
 - (try to) handle outliers,
 - integrate, unify and transform data
 - improve data quality.

- **Real world data is „dirty“**
 - Contains missing values, noises, outliers, inconsistencies
- **Comes from different information sources**
 - Different attribute names, values expressed differently, related tuples
- **Different value ranges and hierarchies**
 - One attribute range may overpower another
- **Huge amount of data**
 - Makes analysis difficult and time consuming

- **Data cleaning**
 - Fill in missing values, smooth noisy data, identify or remove outliers, resolve inconsistencies
- **Data integration**
 - Integration of multiple databases, data cubes, or files
- **Data transformation**
 - Normalization and aggregation
- **Data reduction**
 - Reduce number of attributes

- Feature Selection
- Dimensionality Reduction
- Sampling
- Outlier Detection
- Missing Value Imputation
- Data Cleaning & Standardization (domain dependent)
- Aggregations (often domain dependent)
- Normalization
- Feature Engineering
- Integration of multiple Data Sources

Feature Selection

- **Feature Selection:** choose a subset of the features (attributes) that is as small as possible and sufficient for the data analysis (= still informative!)
- Feature Selection includes:
 - Removing (more or less) **irrelevant features** and
 - Removing **redundant features**
- Evaluation function to compare sets of attributes
- Strategy (heuristic) to select the possible feature subsets to be compared against each other with this measure

- **Selecting the top-ranked features.**
- Choose the features with the best evaluation when single features are evaluated.
- **Selecting the top-ranked subset.**
- Choose the subset of features with the best performance. This requires exhaustive search and is impossible for larger numbers of features. (For 20 features there are already more than one million possible subsets.)
- **Forward selection.**
- Start with the empty set of features and add features one by one. In each step, add the feature that yields the best improvement of the performance.
- **Backward elimination.**
- Start with the full set of features and remove features one by one. In each step, remove the feature that results in the smallest decrease in performance.

- **Selecting the k Top-Ranked features**
- Rank only the single feature rather than the full subsets
- Pearson's χ^2 -test (categorical values) or information gain
- Straightforward approach using Relief family algorithms

Algorithm Relief($\mathcal{D}, \mathcal{A}, C$) $\rightarrow w[\cdot]$

input: data set \mathcal{D} , $|\mathcal{D}| = n$,
attribute set \mathcal{A} , target variable $C \in \mathcal{A}$
output: attribute weights $w[A]$, $A \in \mathcal{A}$

```
1  set all weights  $w[A] = 0$ 
2  for all records  $\mathbf{x} \in \mathcal{D}$ 
3      find nearest hit  $\mathbf{h}$  (same class label  $\mathbf{x}_C = \mathbf{h}_C$ )
4      find nearest miss  $\mathbf{m}$  (different class label)
5      for all  $A \in \mathcal{A}$ :
6           $w[A] = w[A] - \frac{\text{diff}(A, \mathbf{x}, \mathbf{h})}{n} + \frac{\text{diff}(A, \mathbf{x}, \mathbf{m})}{n}$ 
```

where

$$\text{diff}(A, \mathbf{x}, \mathbf{y}) = \begin{cases} 0 & \text{if } \mathbf{x}_A = \mathbf{y}_A, \\ 1 & \text{otherwise} \end{cases}$$

for categorical attributes A and

$$\text{diff}(A, \mathbf{x}, \mathbf{y}) = \frac{|\mathbf{x}_A - \mathbf{y}_A|}{\max(A) - \min(A)}$$

for numerical attributes A .

- **Selecting the Top-Ranked subsets**
- Recognizes the power of complementing attributes rather than individuals contribution only
- Computationally more expensive
- How can we evaluate the attribute subset?

- **Cross-product** of all attributes in the subset
- **Wrapper**: build and evaluate a model for each subset
- **Filter based on correlation**: remove redundant attributes using heuristic:
 - \bar{r}_{ci} the average correlation of the target attribute with all other attributes in the set
 - \bar{r}_{ii} the average attribute-attribute correlation between different attributes (except the target)
 - k the number of attributes
- **Filter based on missing values**: remove attributes with missing value ratio higher than a given threshold
- **Filter based on variance**: remove attributes with variance lower than a given threshold

$$\frac{k\bar{r}_{ci}}{\sqrt{k + k(k - 1)\bar{r}_{ii}}}$$

- Standard strategies to select the minimal optimal subset of features
- **Forward selection** (greedy bottom-up)
 - Start with empty set of features and add one by one the feature that yield the best improvements
- **Backward elimination** (greedy top-down)
 - Start with full set of features and remove one by one the feature that result in the smallest decrease in performances

1. First train one model on n input features
2. Then train n separate models each on $n - 1$ input features and remove the feature whose removal produced the least disturbance
3. Then train $n - 1$ separate models each on $n - 2$ input features and remove the feature whose removal produced the least disturbance
4. And so on. Continue until desired maximum error rate on *training* data is reached.

1. First, train n separate models on one single input feature and keep the feature that produces the best accuracy.
2. Then, train $n - 1$ separate models on 2 input features, the selected one and one more. At the end keep the additional feature that produces the best accuracy.
3. And so on ... Continue until an acceptable error rate is reached.

Dimensionality Reduction

“Too much data”:

- Consumes storage space
- Eats up processing time
- Is difficult to visualize
- Inhibits ML algorithm performance
- Beware of the model: Garbage in → Garbage out

- Both methods are used for reducing the number of features in a dataset. However:
 - Feature selection is simply selecting and excluding given features **without changing** them.
 - Dimensionality reduction **might transform** the features into a lower dimension.
 - Feature selection is often a somewhat more aggressive and more computationally expensive process.
 - Backward Feature Elimination
 - Forward Feature Construction

- Measure based
 - Ratio of missing values
 - Low variance
 - High Correlation
- Transformation based
 - Principal Component Analysis (PCA)
 - Linear Discriminant Analysis (LDA)
 - t-SNE
- Machine Learning based
 - Random Forest of shallow trees
 - Neural auto-encoder

Missing Values Ratio

▲ First partition (as defined in dialog) - 0:337:0:276 - Partitioning (80% vs. 20%)


File Hilite Navigation View

Table "default" - Rows: 40000 Spec - Columns: 231 Properties Flow Variables

Row ID	D Var16	I Var17	I Var18	I Var19	S Var20	I Var21	I Var22	I Var23	I Var24	I Var25	I Var26	I Var27	D Var28
Row0	?	?	?	?	?	464	580	?	14	128	?	?	166.56
Row1	?	?	?	?	?	168	210	?	2	24	?	?	353.52
Row2	?	?	?	?	?	1212	1515	?	26	816	?	?	220.08
Row4	?	?	?	?	?	64	80	?	4	64	?	?	200
Row7	?	?	?	?	?	32	40	?	2	16	?	?	230.56
Row8	?	?	?	?	?	200	250	?	2	64	?	?	300.32
Row10	?	?	?	?	?	92	115	?	6	112	?	?	133.12
Row11	?	?	?	?	?	236	295	?	8	40	?	?	133.12
Row12	?	?	?	?	?	0	0	?	?	0	?	?	240.56
Row13	?	?	?	?	?	480	600	?	10	216	?	?	176.56
Row14	?	?	?	?	?	148	185	?	0	8	?	?	236.08
Row16	?	?	?	?	?	584	730	?	6	320	?	?	220.08
Row17	?	?	?	?	?	168	210	?	2	32	?	?	166.56
Row18	?	?	?	?	?	12	15	?	2	0	?	?	253.52
Row20	?	?	?	?	?	168	210	?	2	56	?	?	272.08
Row21	?	?	?	?	?	20	25	?	2	0	?	?	86.96
Row22	?	?	?	?	?	192	240	?	2	80	?	?	166.56
Row23	?	?	?	?	?	52	65	?	0	56	?	?	198.88
Row24	?	?	?	?	?	216	270	?	8	128	?	?	200
Row25	?	?	?	?	?	152	190	?	4	16	?	?	20.08
Row26	?	0	0	0	?	?	?	?	?	?	?	?	?
Row28	?	?	?	?	?	0	0	?	?	0	?	?	257.28
Row29	?	?	?	?	?	312	390	?	0	120	?	?	200
Row30	?	?	?	?	?	112	140	?	4	56	?	?	166.56
Row31	?	?	?	?	?	28	35	?	0	16	?	?	288.2
Row33	?	?	?	?	?	160	200	?	4	40	?	?	Missing Value
Row36	?	?	?	?	?	612	765	?	14	360	?	?	200
Row37	?	?	?	?	?	380	475	?	4	208	?	?	336.56
Row38	?	?	?	?	?	76	95	?	0	16	?	?	213.36
Row40	?	?	?	?	?	228	285	?	22	56	?	?	200
Row41	?	?	?	?	?	120	150	?	10	80	?	?	133.12
Row42	?	5	0	0	?	?	?	?	?	?	?	?	?
Row43	?	?	?	?	?	72	90	?	0	40	?	?	191.36
Row44	?	?	?	?	?	0	0	?	?	0	?	?	120.4
Row47	?	?	?	?	?	0	0	?	?	0	?	?	186.64
Row48	?	?	?	?	?	172	215	?	4	200	?	?	137.68
Row49	?	?	?	?	?	0	0	?	?	0	?	?	274.16

IF (% missing value > threshold) THEN remove column

Low Variance


 Output table - 0:347:0:337 - Missing Value (Numeric: 0)

File

Hilite

Navigation

View

Table "default" - Rows: 40000

Spec - Columns: 231

Properties

Flow Variables

Row ID	20	Var21	Var22	Var23	Var24	Var25	Var26	Var27	Var28
Row51	336	420	0	8	72	0	0	133.12	
Row52	120	150	0	0	16	0	0	286.96	
Row54	124	155	0	0	0	0	0	234.72	
Row55	184	230	0	4	64	0	0	642.64	
Row56	268	335	0	4	88	0	0	133.12	
Row57	128	160	0	0	96	0	0	198.88	
Row59	132	165	0	0	112	0	0	253.52	
Row60	44	55	0	0	24	0	0	186.64	
Row61	104	130	0	4	72	0	0	166.56	
Row62	212	265	0	6	136	0	0	379.6	
Row63	20	25	0	0	0	0	0	166.56	
Row65	492	615	0	18	256	0	0	133.12	
Row66	148	185	0	2	8	0	0	186.64	
Row68	140	175	0	2	40	0	0	176.56	
Row69	0	0	0	0	0	0	0	166.56	
Row71	0	0	0	0	0	0	0	392.08	
Row72	124	155	0	6	88	0	0	153.2	
Row73	152	190	0	0	32	0	0	253.52	
Row74	324	405	0	8	104	0	0	186.64	
Row75	0	0	0	0	0	0	0	0	
Row76	60	75	0	6	0	0	0	200	
Row77	180	225	0	4	88	0	0	166.56	
Row78	232	290	0	4	144	0	0	200	
Row79	16	20	0	0	16	0	0	313.68	
Row81	152	190	0	0	48	0	0	220.08	
Row82	108	135	0	4	88	0	0	166.56	

<

>

Note: requires min-max-normalization, and only works for numeric columns

- If column has **constant** value (variance = 0), it contains no useful information
- In general: IF (variance < threshold) THEN remove column

High Correlation

- Two **highly correlated** input variables probably carry similar information
- IF (**corr(var1, var2) > threshold**) => remove var1

Note: requires min-max-normalization of numeric columns



Principal Component Analysis (PCA)

- PCA is a statistical procedure that **orthogonally** transforms the original n coordinates of a data set into a new set of n coordinates, called principal components.

$$(PC_1, PC_2, \dots PC_n) = PCA(X_1, X_2, \dots X_n)$$

- The first principal component PC_1 follows the direction (eigenvector) of the **largest possible variance** (largest eigenvalue of the covariance matrix) in the data.
- Each succeeding component PC_k follows the direction of the **next largest possible variance** under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding components $(PC_1, PC_2, \dots PC_{k-1})$.

If you're still curious, there's LOTS of different ways to think about PCA:

<https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues>

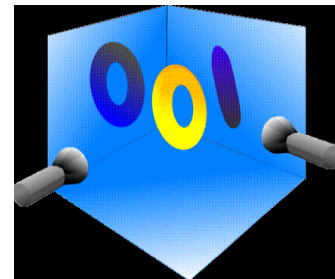
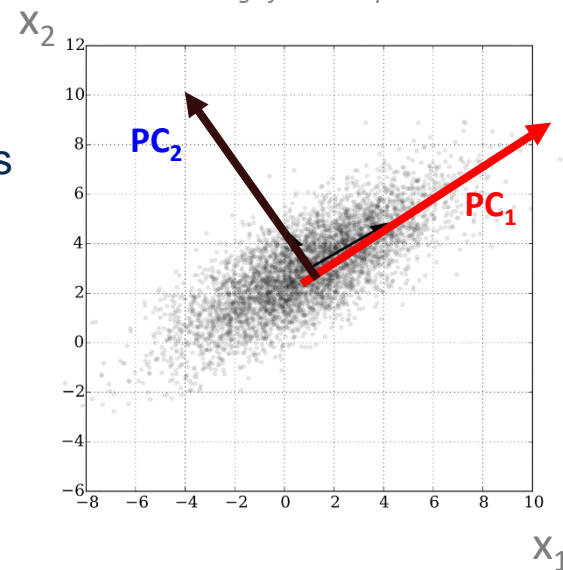


Image from Wikipedia



- PC_1 describes most of the variability in the data, PC_2 adds the next big contribution, and so on. In the end, the last PCs do not bring much more information to describe the data.
- Thus, to describe the data we could use only the top $m < n$ (i.e., PC_1, PC_2, \dots, PC_m) components with little - if any - loss of information
- Caveats:
 - Results of PCA are quite difficult to interpret
 - Normalization required
 - Only effective on numeric columns

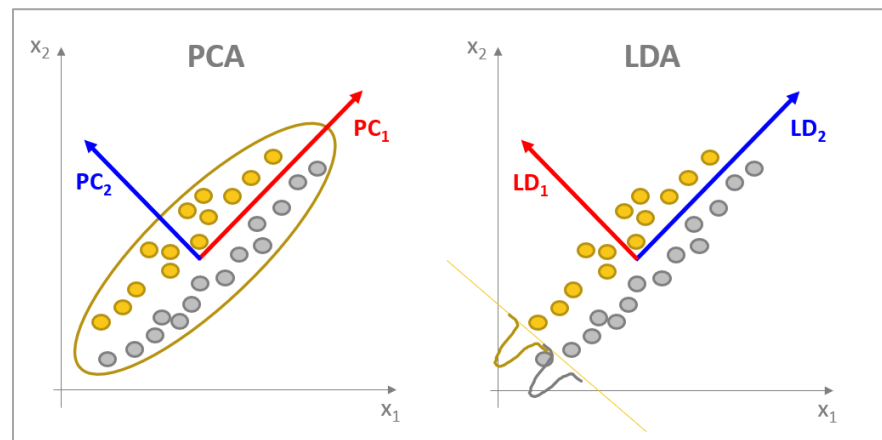
Dimensionality Reduction

- LDA is a statistical procedure that **orthogonally** transforms the original n coordinates of a data set into a new set of n coordinates, called linear discriminants.

$$(LD_1, LD_2, \dots, LD_n) = LDA(X_1, X_2, \dots, X_n)$$

- Here, however, discriminants (components) **maximize the separation between classes**

- PCA : unsupervised
- LDA : supervised

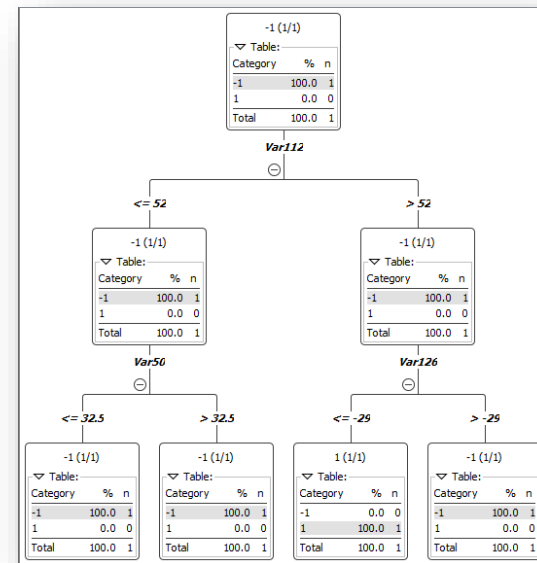


- LD_1 describes best the class separation in the data, LD_2 adds the next big contribution, and so on. In the end, the last LDs do not bring much more information to separate the classes.
- Thus, for our classification problem we could use only the top $m < n$ (i.e., LD_1, LD_2, \dots, LD_m) discriminants with little - if any - loss of information
- Caveats:
 - Results of LDA are quite difficult to interpret
 - Normalization required
 - Only effective on numeric columns

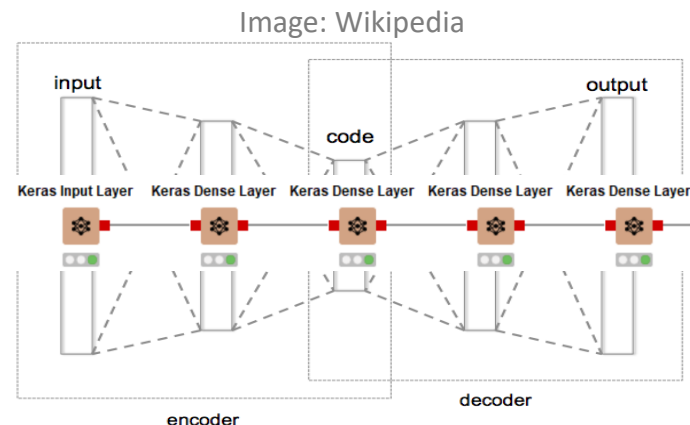
Dimensionality Reduction

Ensembles of Shallow Decision Trees

- Often used for classification, but can be used for feature selection too
- Generate a large number (we used 2000) of trees that are very shallow (2 levels, 3 sampled features)
- Calculate the statistics of candidates and selected features. The more often a feature is selected in such trees, the more likely it contains predictive information
- Compare the same statistics with a forest of trees trained on a random dataset.



- Feed-Forward Neural Network architecture with encoder / decoder structure. The network is trained to reproduce the input vector onto the output layer.



- That is, it compresses the input vector (dimension n) into a smaller vector space on layer “code” (dimension $m < n$) and then it reconstructs the original vector onto the output layer.
- If the network was trained well, the reconstruction operation happens with minimal loss of information.

Record Selection and Sampling

- **Record Selection:** Select a subsample of the data that is still representative of the whole population
- **Reasons for using subsamples:**
 - **Faster Computation**
 - **Cross-Validation** with training and test set
 - **Timeliness:** use more recent data to predict future events
 - **Representativeness:** Is the given sample matching the whole population? If not and we do have information about the true distribution, select a representative subsample. Choose under-represented cases more frequently.
 - **Rare events:** as above, rare events can be increased artificially by sampling with higher probability (over-sampling)

- Size of the subsample: Hoeffding inequality

$$P(|\bar{x} - E[x]| \geq \varepsilon) \leq 2 \exp\left(-\frac{2n\varepsilon^2}{(b-a)^2}\right)$$

- **Instance selection:** reduce data set to those cases that are really important for the modeling technique
- **Support Vector Machines:** data close to boundaries are more relevant than internal ones
- **Clustering extension:** attribute *weight* indicating how many similar records can be replaced

Missing Value Imputation

Data is not always available

- E.g., many tuples have no recorded value for several attributes, such as weight in a people database

Missing data may be due to:

- Equipment malfunctioning (broken sensors)
- Inconsistency with other recorded data and thus deleted
- Data not entered (manually)
- Data not considered important at the time of collection
- Data format / contents of database changes
- Refusal to answer a question
- Irrelevant attribute for the corresponding object (pregnant (yes/no) for men)
- Missing value might not necessarily be indicated as missing (instead: zero or default values).

Types of missing values:

Example: Suppose you are modeling weight Y as a function of sex X

- **Missing Completely At Random (MCAR)**: the probability that a value for X is missing does neither depend on the value of X nor on other variables.
There may be no particular reason why some people told you their weights and others didn't.
- **Missing At Random (MAR)**: the probability that Y is missing depends only on the value of X .
One sex X may be less likely to disclose its weight Y .
- **Not Missing At Random (NMAR)**: the probability that Y is missing depends on the unobserved value of Y itself.
Heavy (or light) people may be less likely to disclose their weight.

- Some methods cannot deal with empty fields
- **Ignorance/Deletion**
 - Can distort the data
- **Imputation**
 - Estimate with regression, classification or domain knowledge
 - Mean, variance, variability of the data will be affected
- **Explicit Value or Variable**
 - Introduce a special global constant (e.g. *MISSING*)

How to handle missing values?

- Ignore/delete the record
- Fill in (impute) missing value as:
 - Fixed value: e.g., “unknown”, -9999, -1 when only positive numbers in the domain, etc.
 - Attribute mean / median / mode
 - Attribute most frequent value
 - Next / previous / avg interpolation / moving avg value (in time series)
 - A predicted value based on the other attributes (inference-based such as Bayesian, Decision Tree, ...)

Outlier Detection

- **What are outliers?**
- Errors in measurements or exceptional conditions that don't describe the common functioning of the underlying system
- Outliers are supposed to be rare: removing them affects the system less than a wrong value
- Completely drop the case
- Only remove/impute the suspicious values

- An **outlier** is a value or data object that is far away or very different from all or most of the other data.
- Causes for outliers:
 - Data **quality** problems (erroneous data coming from wrong measurements or typing mistakes)
 - Exceptional or unusual **situations**/data objects.
- Outlier handling :
 - Outliers coming from erroneous data should be **excluded** from the analysis.
 - Even if the outliers are correct (exceptional data), it is sometimes useful to exclude them from the analysis.
 - For example, a single extremely large outlier can lead to completely misleading values for the mean value.

Categorical Attributes

- An outlier is a value that occurs with a **frequency extremely lower** than the frequency of all other values.

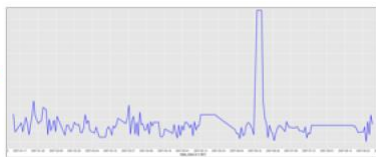
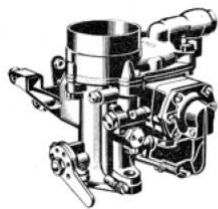
Example: Automatic quality control system

- **Goal:** Train a classifier, classifying the parts as correct or with failures based on measurements of the produced parts.
- The frequency of the correct parts will be so high that the parts with failure might be considered as outliers.

Numerical Attributes

- Outliers in boxplots.
- Statistical tests, for example Grubb's test

- An outlier could be, for example, a rare behavior, a system defect, a measurement error, or a reaction to an unexpected event



FTSE 100 Index

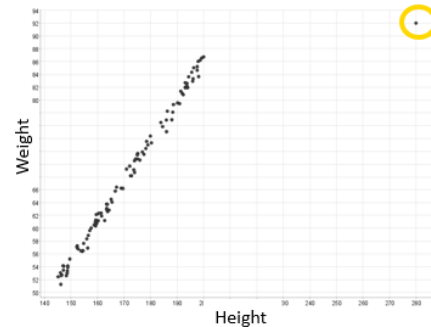
At market close 06/24/2016



Image:

<https://www.nytimes.com/2016/06/25/business/international/brexit-financial-economic-impact-leave.html>

Brexit
referendum



- Why finding outliers is important?
 - Summarize data by statistics that represent the majority of the data
 - Train a model that generalizes to new data
 - Finding the outliers can also be the focus of the analysis and not only data cleaning

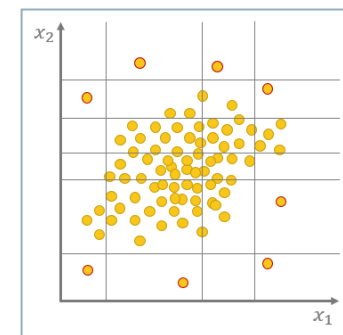
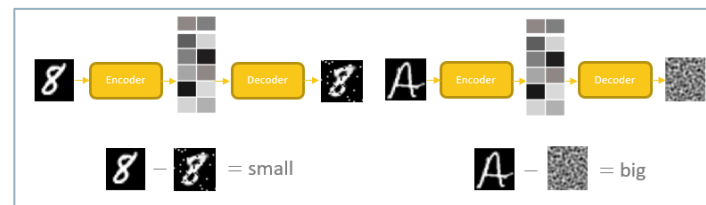
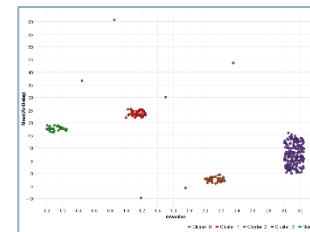
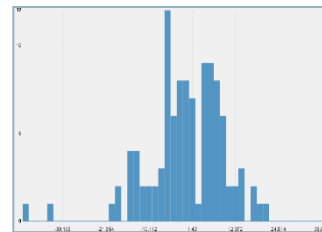
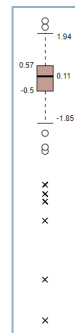
Outlier Detection Techniques

— Knowledge-based

- We know that a 200 year old person must be a mistake
- We know that “A” in a number corpus is an outlier

— Statistics-based

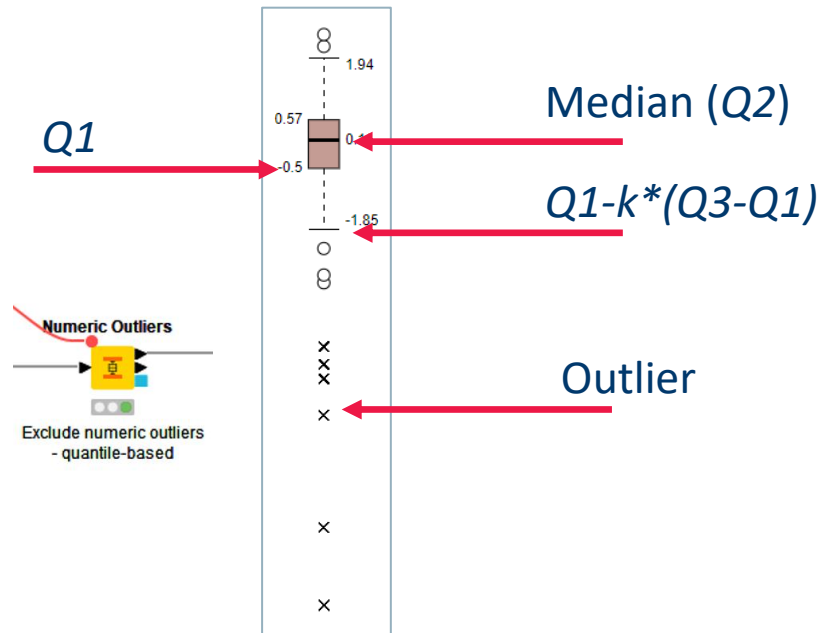
- Distance from the median
- Position in the distribution tails
- Distance to the closest cluster center
- Error produced by an autoencoder
- Number of random splits to isolate a data point from other data



- Quantile-based: Box plot
- Distribution-based: Z-Score
- Cluster-based: DBSCAN
- Neural Autoencoder
- Isolation Forest
- ...

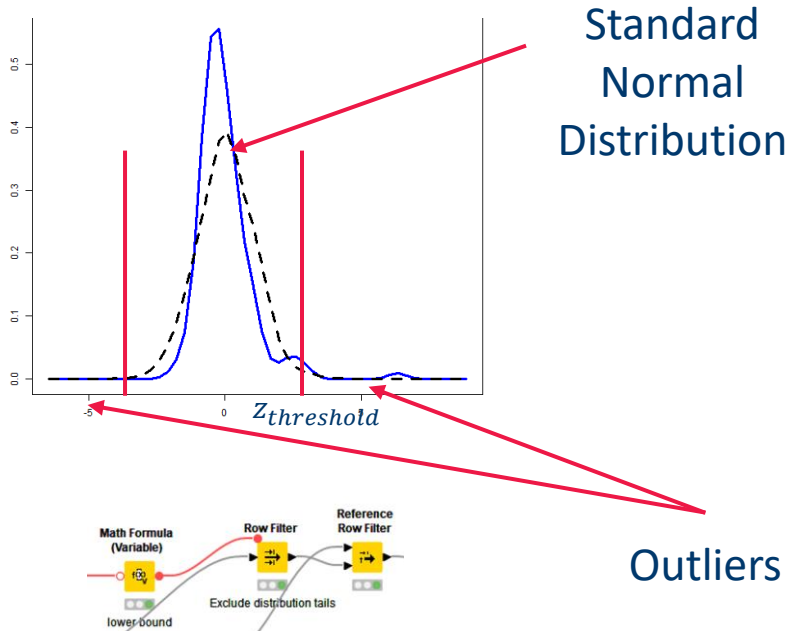
Quartile based (Box Plots)

- Flag data points outside the upper and lower whiskers of a box plot as outliers



- Challenges:
 - Outliers in the data expand the quantiles
 - Skewed data might require different k to detect upper and lower outliers
 - One-dimensional

- Flag data points in the distribution tails as outliers

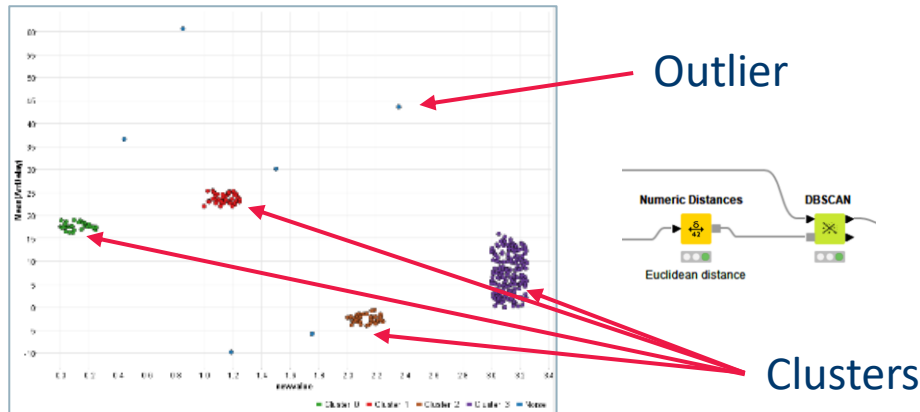


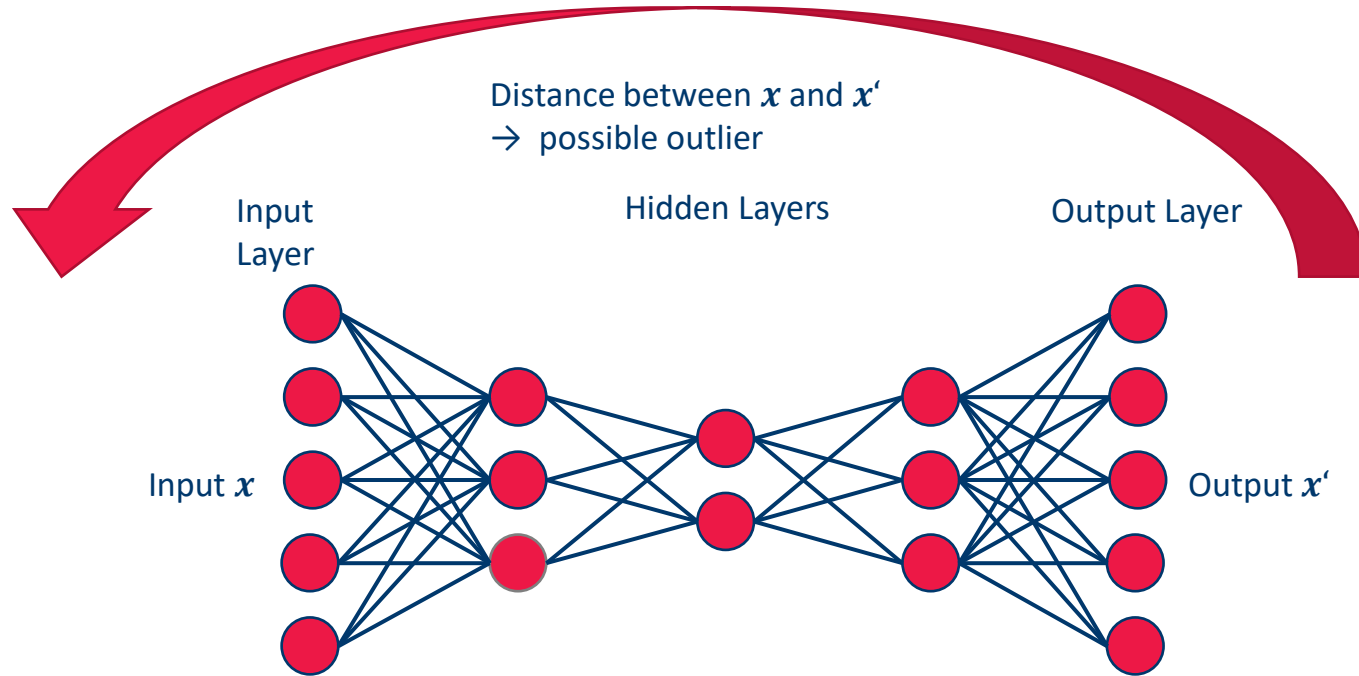
- Challenges:
 - Normality assumption
 - The parameters of the distribution are sensitive to outliers
 - Doesn't work for data with a trend and seasonality

Clustering-based methods: DBSCAN

- Flag noise points outside the clusters as outliers
- Density based clustering method that defines all data points as
 - Core points: at least $MinPts$ neighbors within ϵ
 - Border points: within ϵ from a core point
 - Noise points: all other data points

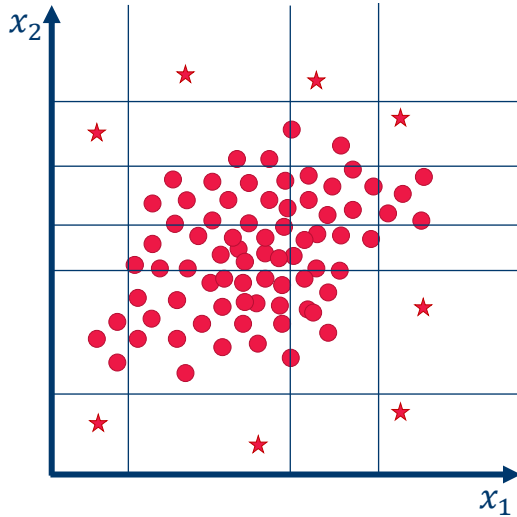
- Challenges:
 - Quadratic runtime
 - Sensitive to the parameter values $MinPts$ and ϵ
 - Clusters with varying densities are difficult to detect



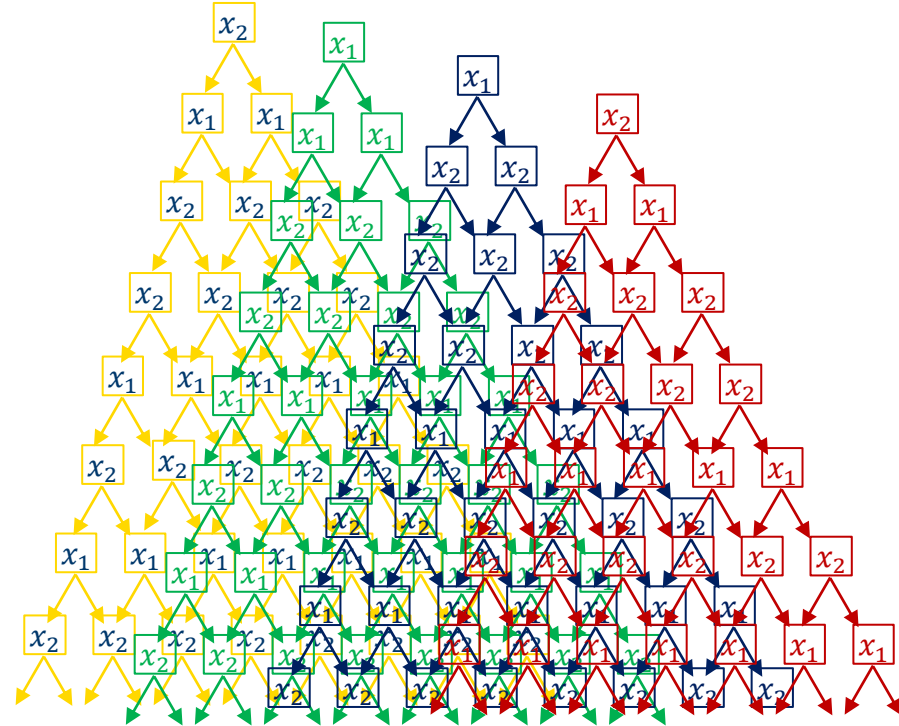


- Challenges:
 - Low interpretability
 - Requires large amounts of data

Idea: Outlier can be isolated with less random splits



- Challenges:
 - Low interpretability



Data Cleaning and Standardization

– **Single record investigation**

- Correct values in order to recognise differently spelled but semantically identical values
- Converting capital letters into lower case
- Removing spaces and void character
- Fixing the mormat of numbers, date and time
- Splitting fields containing mixed information (*field overloading*)
- Using spell-checker or stemming
- Replacing abbreviations with their long form
- Normalizing the writing of addresses and names
- Converting numerical values into standard units
- Using dictionaries to check if all values map to domain knowledge

– **Multiple records investigation**

- Reduce redundance by merging records that are variant of the same entity
- **Record linkage** (or entity resolution): merge records identified to be similar

From Categorical to Numerical

- Binary attribute: numerical attribute with the values 0 and 1.
- Ordinal attribute (“sortable”): enumerate in the correct order $1, \dots, k$
- Categorical attribute (not ordinal) with more than two values, say a_1, \dots, a_k , should **not be converted into a single numerical attribute** **instead:** convert to k attributes A_1, \dots, A_k with values 0 and 1.
- a_i is represented by $a_i = 1$ and $a_j = 0$ for $i \neq j$ (1-of- n encoding).

From Numerical to Categorical

- Splitting a numerical range into a number of bins
- **Equi-width discretization.** Splits the range into intervals (bins) of the same length.
- **Equi-frequency discretization.** Splits the range into intervals such that each interval (bin) contains (roughly) the same number of records.
- **V-optimal discretization.** Minimizes $\sum_i n_i V_i$ where n_i is the number of data objects in the i th interval and V_i is the sample variance of the data in this interval.
- **Minimal entropy discretization.** Minimizes the entropy. (Only applicable in the case of classification problems.)

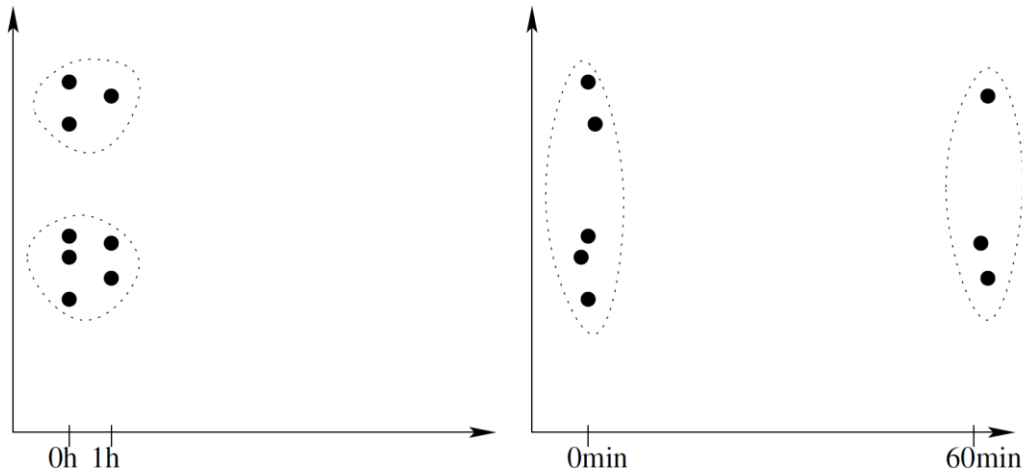
Normalization

- For some data analysis techniques (e.g. PCA, MDS; cluster analysis) the influence of an attribute depends on the scale or measurement unit.
- To guarantee impartiality, some kind of **standardisation** or **normalisation** should be applied.

Example:

- Lengths in cm (100 – 200) and weights in kilogram (30 – 150) fall both in approximately the same scale
- What about lengths in m (1-2) and weights also in gram (30000 – 150000)?
 - The weight values in mg dominate over the length values for the similarity of records!

- 0h vs 1h can be expressed as 0min vs 60min



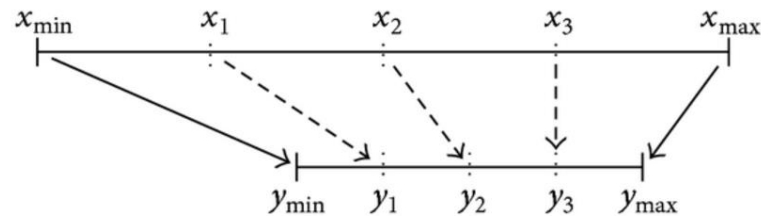
Goal of normalization:

- Transformation of attributes to make record ranges comparable

– min-max normalization

$$n: \text{dom}(X) \rightarrow [0,1]$$

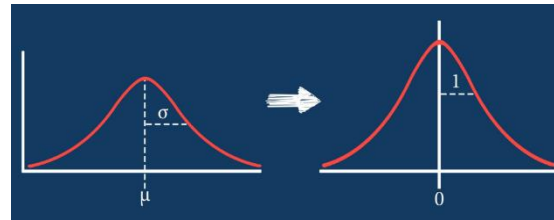
$$y = \frac{x - x_{\min}}{x_{\max} - x_{\min}} (y_{\max} - y_{\min}) + y_{\min}$$



– z-score normalization

$$s: \text{dom}(X) \rightarrow \mathbb{R}$$

$$y = \frac{x - \hat{\mu}(X)}{\hat{\sigma}(X)}$$



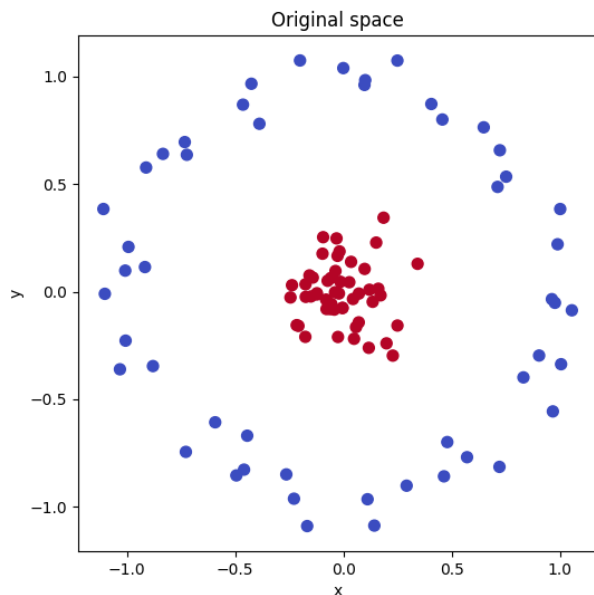
– normalization by decimal scaling

$$d: \text{dom}(X) \rightarrow [0,1]$$

$$y = \frac{x}{10^j} \quad \text{where } j \text{ is the smallest integer value larger than } \log_{10}(\max(X))$$

Feature Engineering

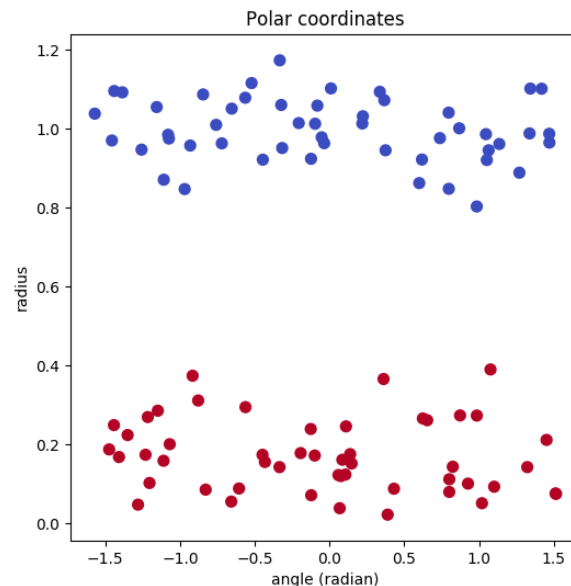
Sometimes transforming the original data allows for better discrimination by ML algorithms.



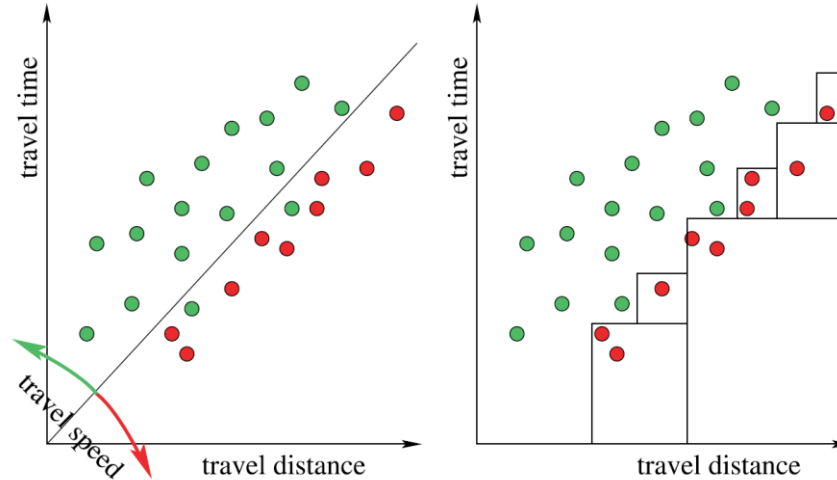
Euclidean to
polar coordinates

$$\text{Radius } r = \sqrt{x^2 + y^2}$$

$$\text{Angle } \theta = \text{Tan}^{-1}(y/x)$$



- Idea: Offering new features that were difficult to represent by the original model



Simple example for the usefulness of derived features: for a number of journeys, the travel time and distance are shown; the color indicates whether the driver was ticketed or not. Discriminating both classes with axis-parallel rectangles is laborious, but easy with a new attribute for travel speed

- Transformation of existing and construction of new attributes that may replace the original attributes
- **Provide operability**
 - Data transformation is required by the model
- **Assure impartiality**
 - Every attribute affects the model equally
- **Maximize efficiency**
 - Exploit domain knowledge to improve the model results

- **Scale Conversion**

- Categorical → Numerical: map categorical and ordinal values to a set of binary values
- Numerical → Categorical: **Discretization** (equal-width, equal-depth, V-optimal)

- **Dynamic Domains**

- Map original values to more abstract or general values
- Newly occurring entries can be as well mapped to the generalized terms

- **Problem Reformulation**

- Reformulate a problem so that the range of applicable techniques is enlarged

- **Coordinate Transformations**

Remember PCA and LDA?

Polar coordinates , ...

- Distances to cluster centres, after data clustering

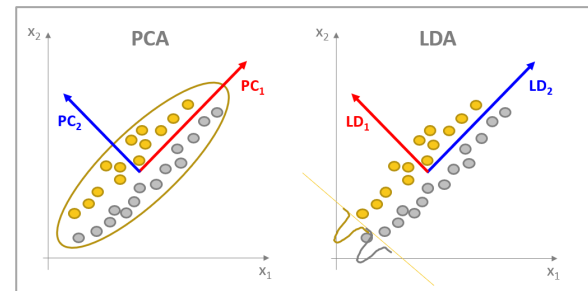
- Simple math transformations on single columns

$(e^x, x^2, x^3, \tanh(x), \log(x), \dots)$

- Combining together multiple columns in math functions

$(f(x_1, x_2, \dots, x_n), x_2 - x_1, \dots)$

- The whole process is domain dependent



- Some measures (e.g. Information gain) tend to prefer attributes with a large number of values. This can lead to overfitting
- Reduce granularity of the data (e.g. ZIP code reduced to regional info)
- Typical assumption: some variables obey a certain distribution (e.g. Gaussian)
- Transform the data to better approximate the distribution using the **power transform**

$$y \mapsto \begin{cases} \frac{y^\lambda - 1}{\lambda \bar{y}^{(\lambda-1)}} & \text{if } \lambda \neq 0 \\ \bar{y} \log y & \text{if } \lambda = 0 \end{cases}$$

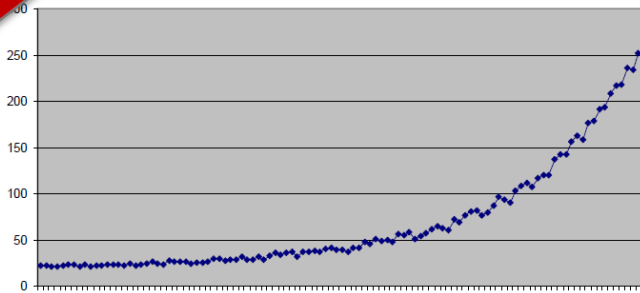
- **Derived features** (e.g. Average speed from travel time and travel distance)
- **Change the baseline** (e.g. Define income „per household“ instead of „income per head“)
- **Grouping**: add the attribute „*group label*“ after clustering analysis
- **Hyperplanes** (e.g. Binary attribute that becomes true if n out m preselected features hold)
- **Level of granularity**: test for dependencies at various level of detail
- **Nonlinearities**: transform beforehand non linearly dependent variables in order to build a more complex linear model
- **Add metadata** (e.g. indicating when data were altered)

Feature Engineering in Time Series Analysis

- Second order differences: $y = x(t) - x(t - 1)$ & $y'(t) = y(t) - y(t - 1)$
- Logarithm: $\log(y'(t))$

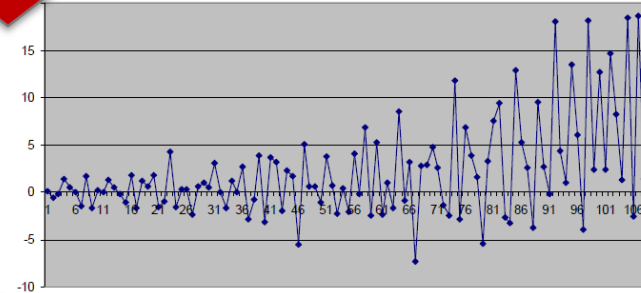
1

1 - Original Time Series: non-stationary (mean and variance)



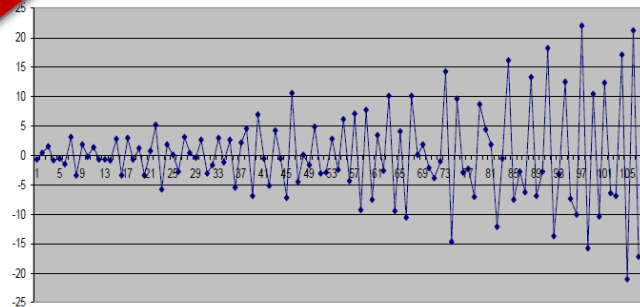
2

2 - First Order Differencing: non-stationary (mean and variance)



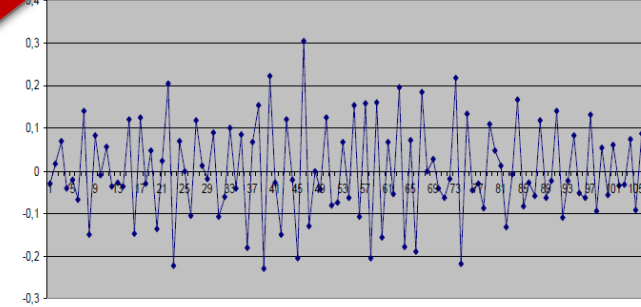
3

3 - Second Order Diff.: stationary in mean, but not in variance



4

4 - Double Differencing applied to Log(Series): stationary series



- Feature Construction refers to generating new features from the existing ones
- Example: **Find the best workers in a company.**
- Attributes :
 - the tasks, a worker has finished within each month,
 - the number of hours he has worked each month,
 - the number of hours that are normally needed to finish each task.
- These attributes do *contain* information about the efficiency of the worker.
- But instead of using these three “raw” attributes, it might be more useful to define a new attribute ***efficiency***.

$$efficiency = \frac{\text{hours actually spent to finish the tasks}}{\text{hours usually spent to finish the tasks}}$$

- Texts
- Graphs
- Images
- Molecules
- Other Objects

- Especially for complex data types, feature extraction is required
- **Text data analysis.** Frequency of keyword, . . .
- **Time series data analysis.** Fourier or wavelet coefficients, . . .
- **Image data analysis.** Fourier or wavelet coefficients, . . .
- **Graph data analysis.** Number of vertices, number of edges, . . .

Data Integration

– Vertical Data Integration (Concatenation)

- Unify database structures
- Remove duplicates

id	Last name	First name	Gender
p2	Mayer	Susan	F
p5	Smith	Walter	M
p7	Brown	Jane	F
...

 +

Shopper id	Item id	Price
p2	i254	12.50
p5	i4245	1.99
p5	i32123	1.29
p5	i254	12.50
p5	i21435	5.99
p7	i254	12.50
...

– Horizontal Data Integration (Join)

- Overrepresentation of items
- Data explosion

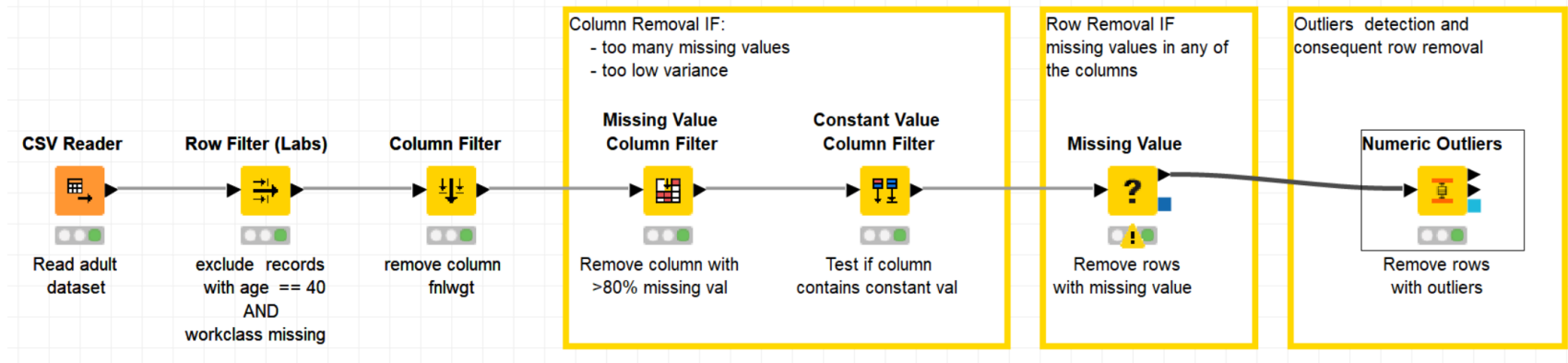
Item id	Price	Last name	First name	gEnder
i254	12.50	Mayer	Susan	F
i4245	1.99	Smith	Walter	M
i32123	1.29	Smith	Walter	M
i254	12.50	Smith	Walter	M
i21435	5.59	Smith	Walter	M
i254	12.50	Brown	Jane	F
...

The two data sets on top contain information about customers and product purchases. The joint data set at the bottom combines these two tables. Note how we loose information about individual customers and how a lot of duplicate information is introduced. In reality this effect is, of course, far more dramatic

Practical Examples

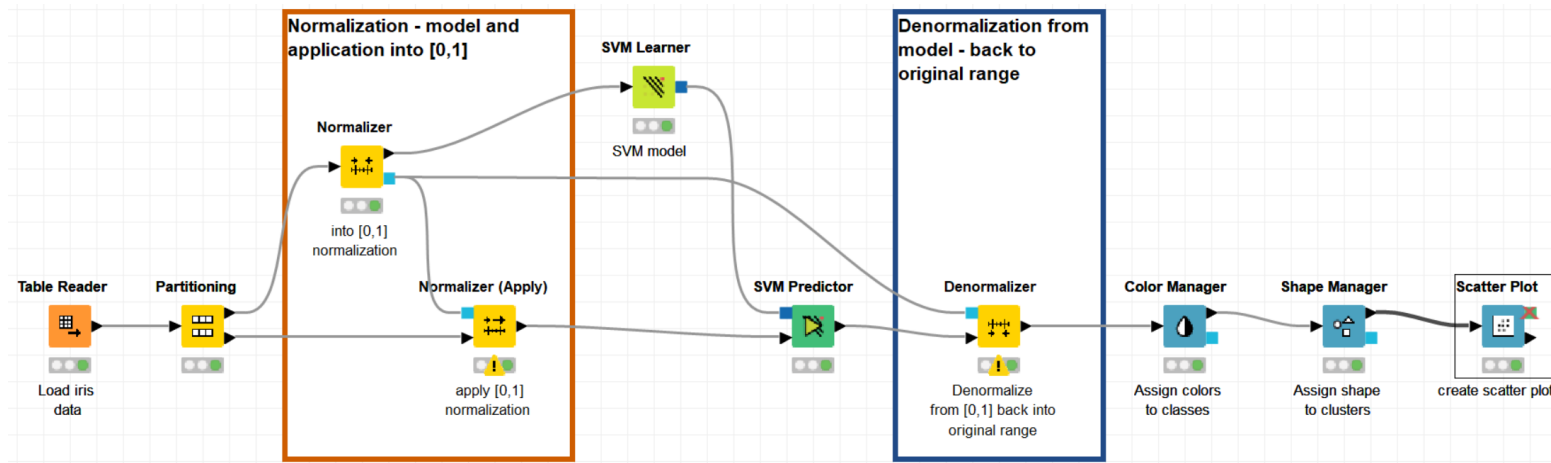
- **Visual data-flow-oriented tool advantages**
- Intuitive Data Preparation
- Easy repeatability and reproducibility
- **Steps:**
- Remove (almost) empty attributes and records
- Normalization and Denormalization
- Backward Feature Elimination

– Remove (almost) empty attributes and records



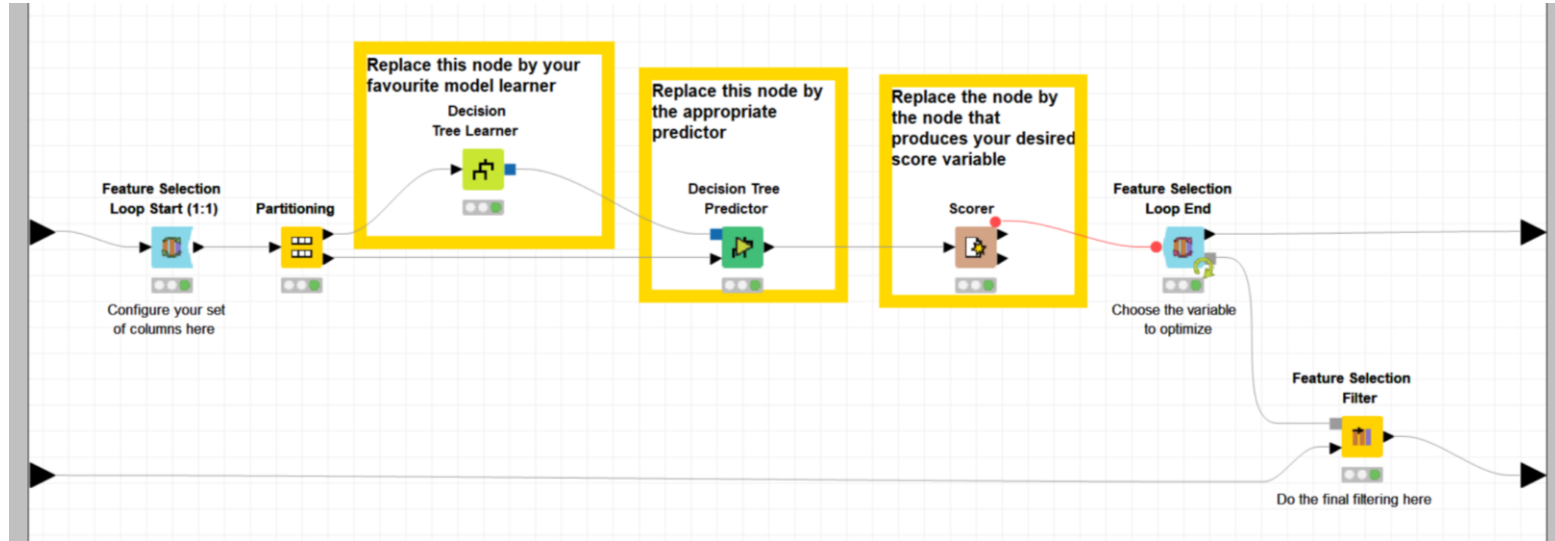
Filtering columns and rows by manually inserted rules; then columns by missing value percentage and by constant value check; rows by outlier detection or missing values in any column

– Normalization and Denormalization



This is how to apply normalization correctly in a data science problem. The normalization model is built on the training set and only applied on the test set! The same normalization model is used to denormalize the numerical attributes back into their original ranges

– Backward Feature Elimination



Content of the Backward Feature Elimination metanode implementing the backward feature elimination procedure

For any questions please contact: email@email.com