DATA ANALYTICS

Elisabetta Fersini

a.a. 2024/2025

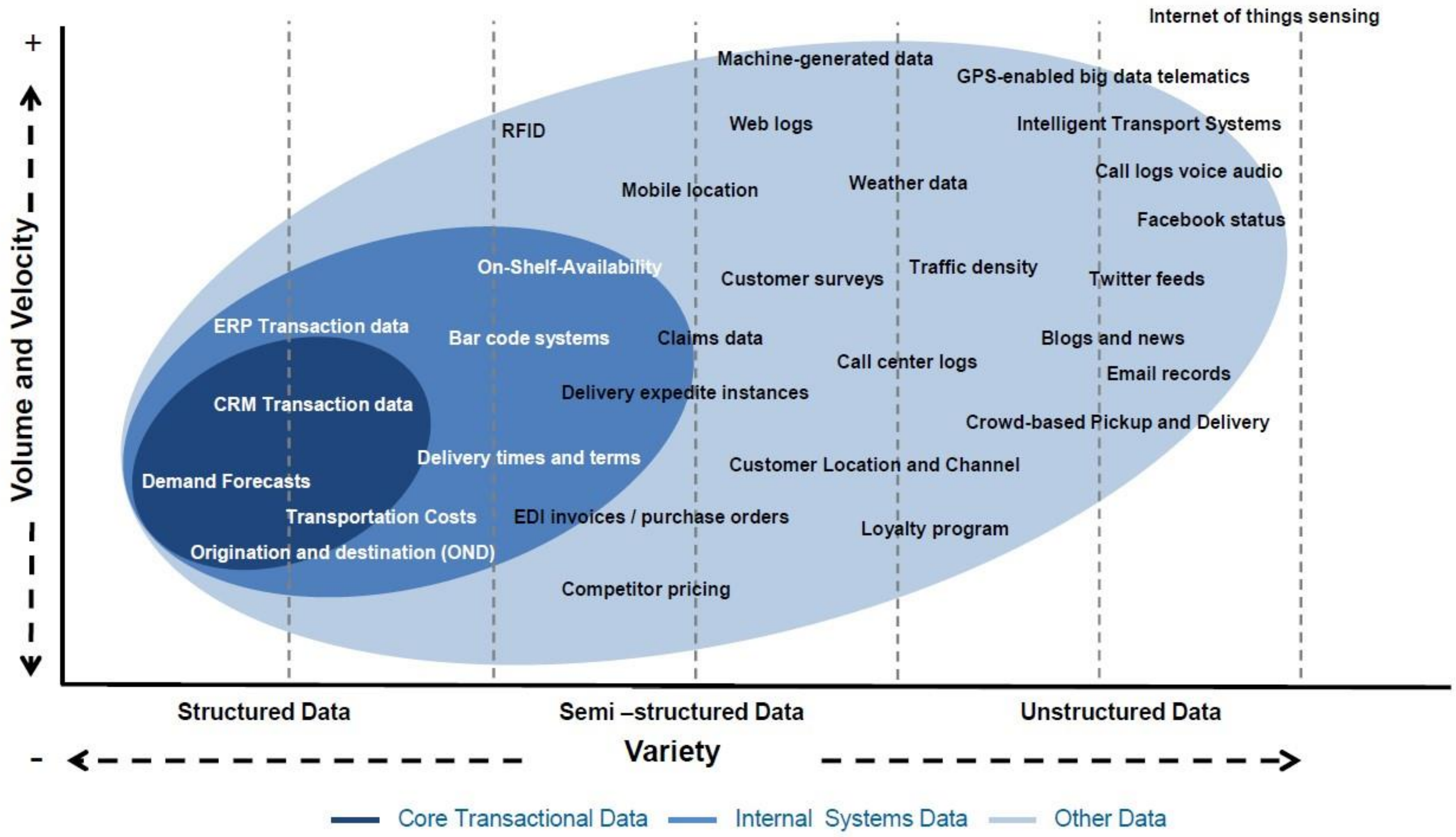
OUTLINE

- Types of Data:
  - Structured, Semi-structured, Unstructured
- Types of Analytics
  - Descriptive, Predictive, Prescriptive
  - Recall of ML models
    - Propositional vs Relational
    - Discriminative, Generative, Clustering, Associative, Forecast

TRENDS LEADING TO DATA FLOOD

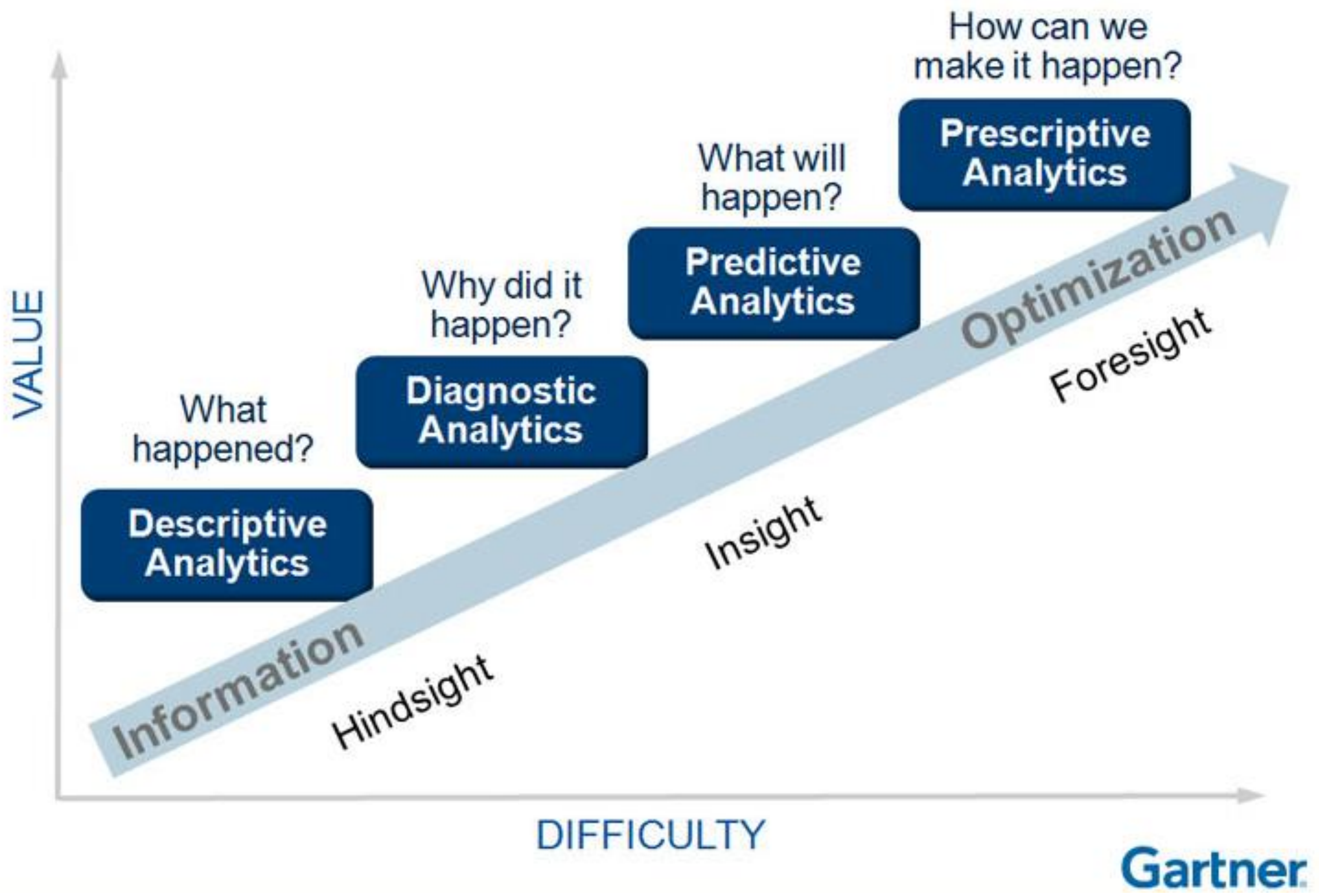
- More data is *<u>generated</u>*:
  - Web, text, images …
  - Business transactions, calls, ...
  - Scientific data: astronomy, biology, etc
- More data is *<u>captured</u>*:
  - Storage technology faster and cheaper
  - DBMS can handle bigger DB

DATA GROWTH



Internet of things sensing
Machine-generated data
GPS-enabled big data telematics
RFID
Web logs
Intelligent Transport Systems
Mobile location
Weather data
Call logs voice audio
Facebook status
On-Shelf-Availability
Customer surveys
Traffic density
Twitter feeds
ERP Transaction data
Bar code systems
Claims data
Blogs and news
Call center logs
Email records
CRM Transaction data
Delivery expedite instances
Crowd-based Pickup and Delivery
Delivery times and terms
Customer Location and Channel
Demand Forecasts
Transportation Costs
EDI invoices / purchase orders
Loyalty program
Origination and destination (OND)
Competitor pricing
+
Volume and Velocity
Structured Data
Semi –structured Data
Unstructured Data
-
Variety
Core Transactional Data
Internal Systems Data
Other Data

TYPES OF ANALYTICS



DATA PRE-PROCESSING

Missing data…feature reduction…unbalanced dataset

WHY DATA PREPROCESSING?

- Data in the real world is dirty
  - incomplete: lacking *attribute values*, lacking certain *attributes of interest*, or containing only aggregate data
  - noisy: containing errors or outliers
  - inconsistent: containing discrepancies in codes or names
- No quality data, no quality mining results!
  - Quality decisions must be based on quality data
  - Data warehouse needs consistent integration of quality data
  - Required for both OLAP and Data Analytics!

MAJOR TASKS

outliers=exceptions!

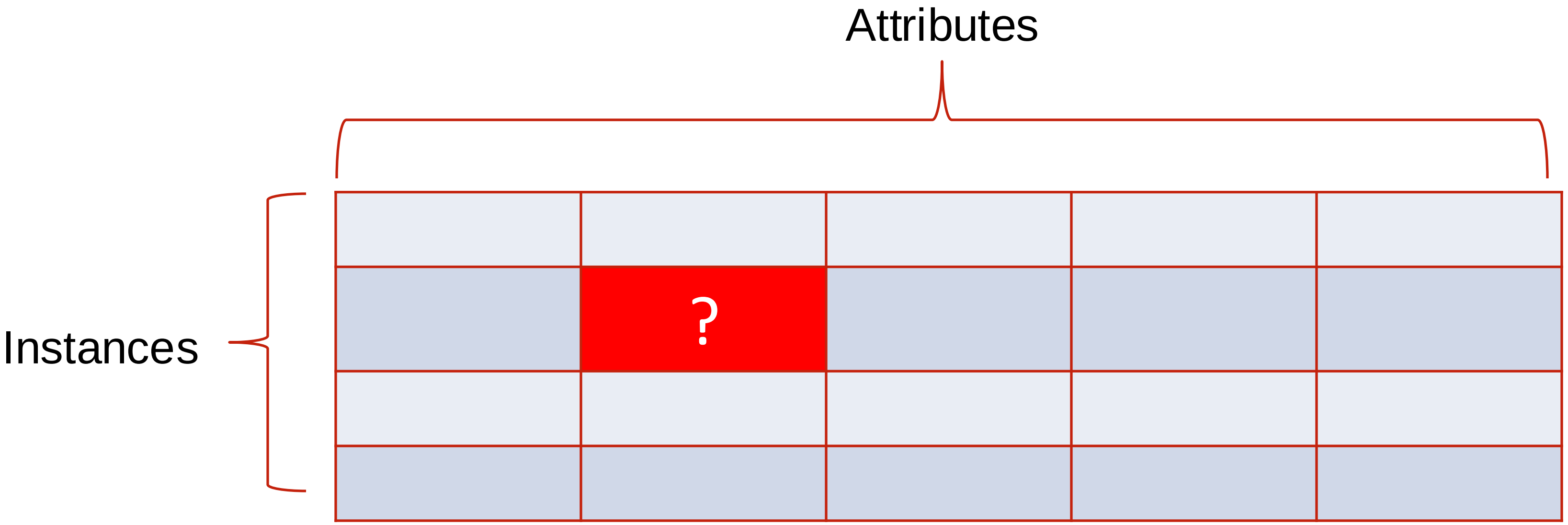
- **<u>Data cleaning</u>**
  - <u>Fill in missing values</u>, <u>smooth noisy data</u>, identify or remove outliers, and resolve inconsistencies
- Data integration
  - Integration of multiple databases or files
- Data transformation
  - Normalization and aggregation
- **<u>Data reduction</u>**
  - <u>Feature selection</u>, feature extraction

DATA CLEANING

- Data cleaning tasks
  - Fill in missing values
  - Smooth out noisy data

WHAT IS A MISSING VALUES?

- A *missing value* is an empty cell in the table that represents a dataset



WHY CAN DATA BE INCOMPLETE?

- Attributes of interest are not available (e.g., customer information for sales transaction data)
- Data were not considered important at the time of transactions, so they were not recorded!
- Data not recorder because of misunderstanding or malfunctions
- Data may have been recorded and later deleted!
- Missing/unknown values for some data

WHY MISSING VALUES ARE IMPORTANT?

- Three reasons:
  - Loss of efficacy: less patterns extracted from data or conclusions statistically less strong
  - Complications in handling and analyzing the data. Methods are in general not prepared to handle them
  - Bias resulting from differences between missing and complete data.

WHY CAN DATA BE NOISY?

- Faulty instruments for data collection
- Human or computer errors
- Errors in data transmission
- Technology limitations (e.g., sensor data come at a faster rate than they can be processed)
- Inconsistencies in naming conventions or data codes (e.g., 2/5/2002 could be 2 May 2002 or 5 Feb 2002)
- Duplicate tuples, which were received twice should also be removed

CHARACTERIZATION OF MISSING VALUES

- Three categories of missing values
  - **M**issing **C**ompletely **A**t **R**andom (**MCAR**), when the distribution of an example having a missing value for an attribute does not depend on either the observed data or the missing data
  - **M**issing **A**t **R**andom (**MAR**), when the distribution of an example having a missing value for an attribute depends on the observed data, but does not depend on the missing data
  - **N**ot **M**issing **A**t **R**andom (**NMAR**), when the distribution of an example having a missing value for an attribute depends on the missing values.
- Depending on the type of missing value, some of the handling methods will be suitable or not

STRATEGIES FOR MISSING VALUES HANDLING

- **Ignore** instances/attributes with missing values
  - Simplest approach. Allows the use of unmodified data mining methods
  - Only practical if there are few examples with missing values. Otherwise, it can introduce bias
- **Convert** the missing values into a new value
  - Use a special value for it => ‘missing’, ‘?’, ‘NA’
- **Imputation** methods
  - Assign a value to the missing one, based on the rest of the dataset
  - Use the unmodified data mining methods

IMPUTATION METHODS

- As they extract a model from the dataset to perform the imputation, they are suitable for MCAR and MAR types of missing values

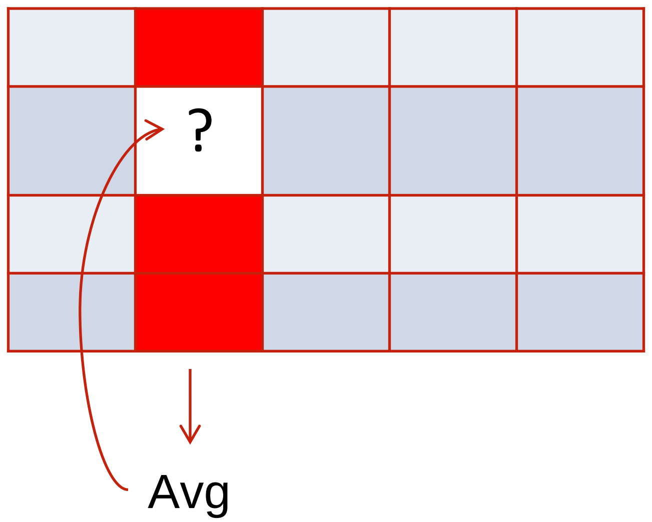
- Not suitable for NMAR type of missing data
  - It would be necessary in this case to go back to the source of the data to obtain more information

MOST COMMON (MC) VALUE

- If the missing value is continuous
  - Replace it with the mean value of the attribute for the dataset

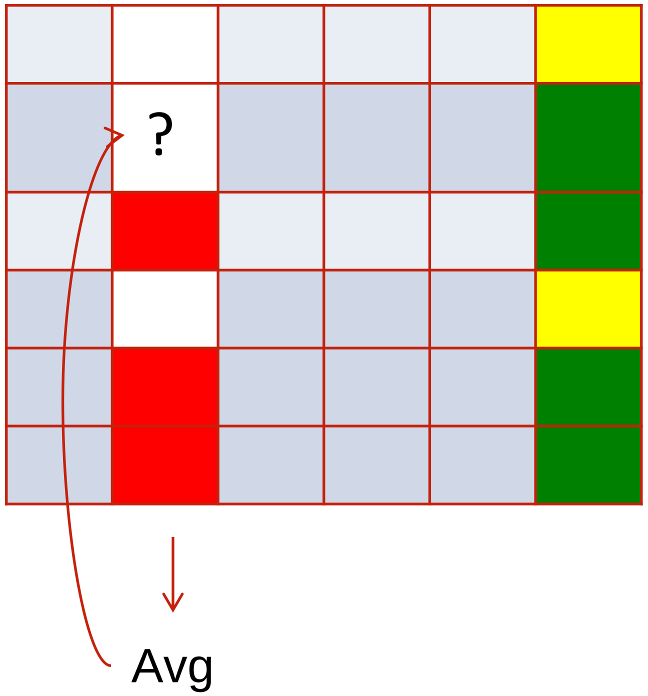
- If the missing value is discrete
  - Replace it with the most frequent value of the attribute for the dataset

- Simple and fast to compute
- Assumes that each attribute presents a normal distribution



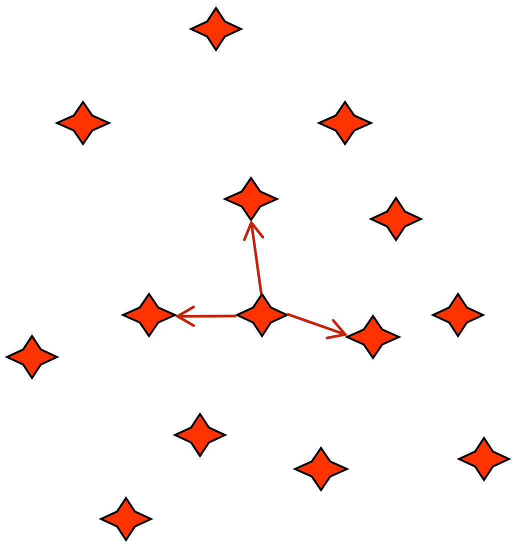
CONCEPT MOST COMMON (CMC) VALUE

- Refinement of the MC policy
- The MV is replaced with the *mean*/most frequent value computed from the instances *belonging to the same class*
- Assumes that the distribution for an attribute of all instances from the same class is normal



IMPUTATION WITH K-NEAREST NEIGHBOUR

- k-NN machine learning algorithm
  - Given an unlabeled new instance
  - Select the *k* instances from the training set most similar to the new instance
  - Predict the majority class from these *k* instances



- k-NN for MV imputation
  - Select the *k* nearest neighbours
  - Replace the MV with the most frequence/mean value from these *k* instances

HOW TO HANDLE NOISY DATA? SMOOTHING TECHNIQUES

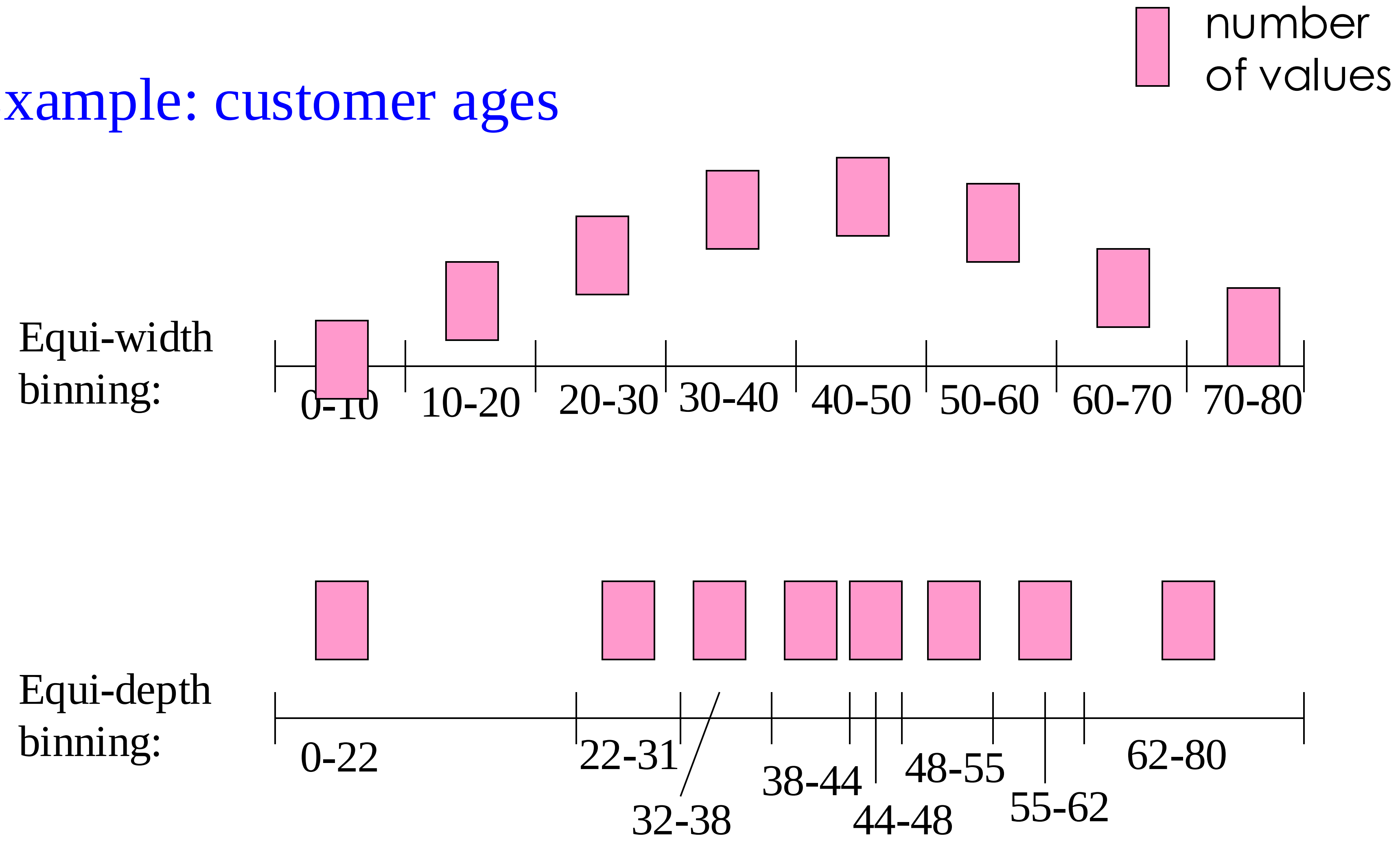
- Binning method:
  - first sort data and partition into bins
  - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.

SIMPLE DISCRETIZATION METHODS: BINNING

- Equal-width (distance) partitioning:
  - It divides the range into *N* intervals of equal size: uniform grid
  - if *A* and *B* are the lowest and highest values of the attribute, the width of intervals will be: $W = (B-A)/N$.
  - The most straightforward
  - But outliers may dominate presentation
  - Skewed data is not handled well.
- Equal-depth (frequency) partitioning:
  - It divides the range into *N* intervals, each containing approximately same number of samples
  - Good data scaling – good handing of skewed data

SIMPLE DISCRETIZATION METHODS: BINNING

Example: customer ages



SMOOTHING USING BINNING METHODS

- Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34

* Partition into (equi-depth) bins:

- Bin 1: 4, 8, 9, 15

- Bin 2: 21, 21, 24, 25

- Bin 3: 26, 28, 29, 34

* Smoothing by bin means:

- Bin 1: 9, 9, 9, 9

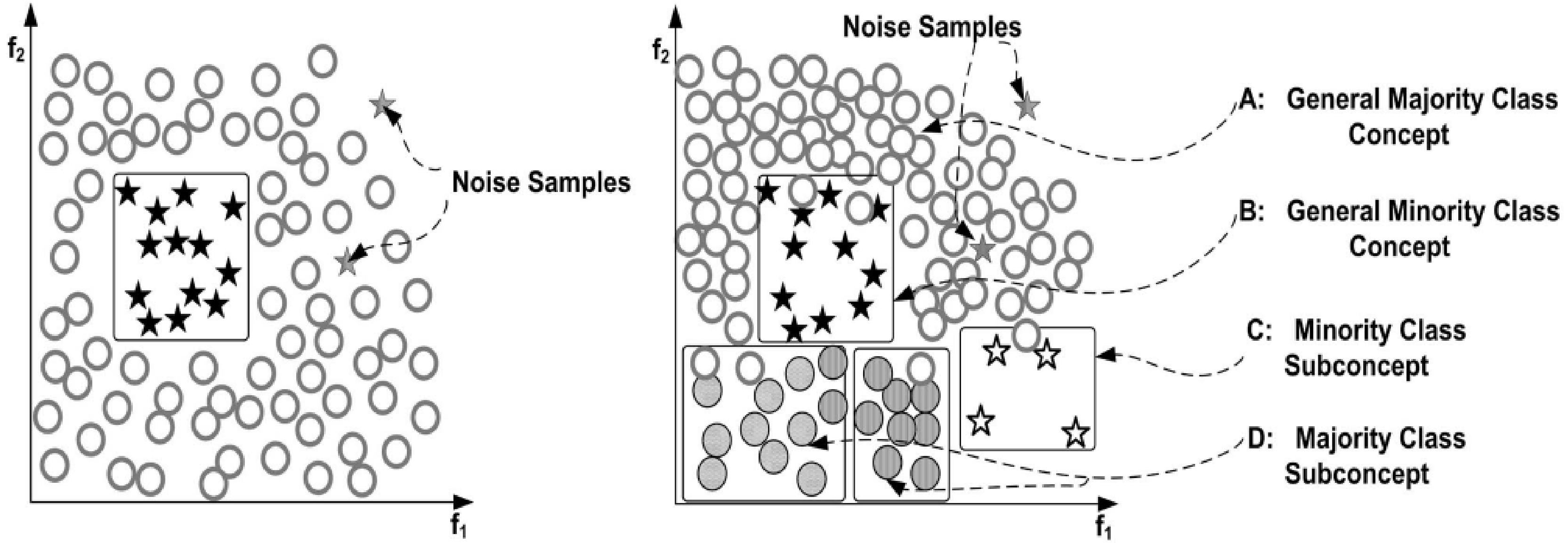
- Bin 2: 23, 23, 23, 23

- Bin 3: 29, 29, 29, 29

UNBALANCED DATA

- Sometimes, classes have very *unequal frequency*
  - Attrition prediction: 97% stay, 3% attrite (in a month)
  - medical diagnosis: 90% healthy, 10% disease
  - eCommerce: 99% don’t buy, 1% buy
  - Security: >99.99% of Americans are not terrorists
- Similar situation with multiple classes
- Majority class classifier can be 97% correct, but useless

UNBALANCED DATA



HANDLING UNBALANCED DATA

If we have two classes that are very unbalanced, then how can we evaluate our classifier?

BALANCING UNBALANCED DATA

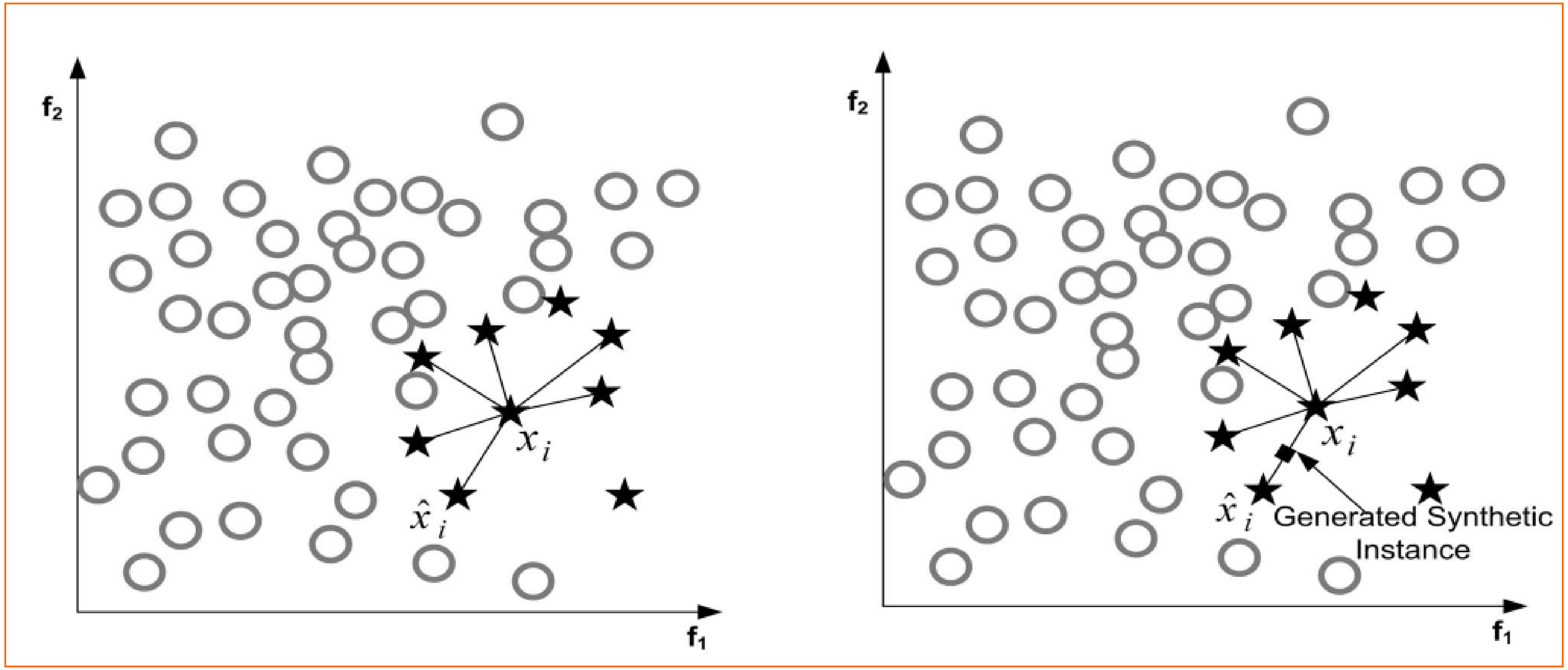
- With two classes, a good approach is to build **BALANCED** train and test sets, and train model on a balanced set
  - Oversampling: select desired number of minority class instances and add equal number of randomly selected majority class
  - Undersampling: select desired number of majority class instances and remove equal number of randomly selected minority class

BALANCING UNBALANCED DATA

- Baseline Methods
  - Random over-sampling
  - Random under-sampling
- Over-sampling Methods
  - Smote
- Under-sampling Methods
  - Tomek links
  - Condensed Nearest Neighbor Rule
  - One-sided selection
  - CNN + Tomek links
  - Neighborhood Cleaning Rule
- Combination of Over-sampling method with Under-sampling method
  - Smote + Tomek links
  - Smote + ENN

OVERSAMPLING

- **S**ynthetic **M**inority **O**versampling **Te**chnique (SMOTE)



OVERSAMPLING

- **S**ynthetic **M**inority **O**versampling **Te**chnique (SMOTE)

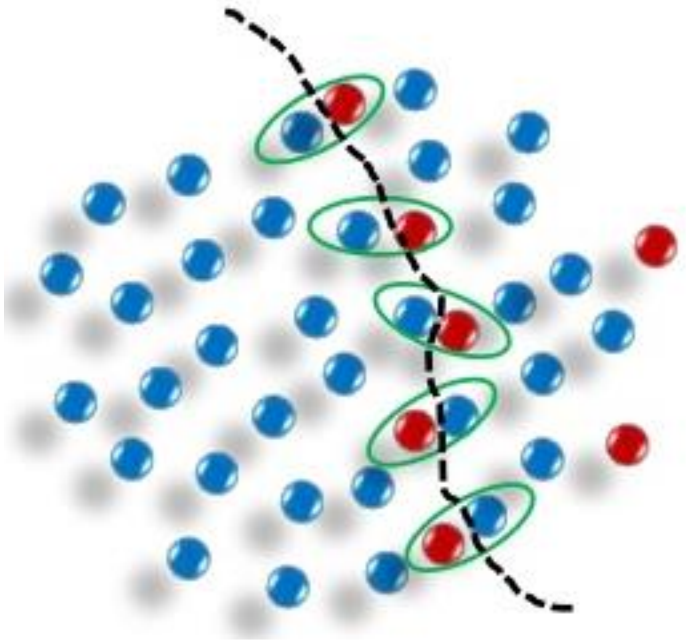
```
Algorithm: SMOTE ( d is Dataset, N is integer, k is integer ) return (dOut is Dataset)
Data: N is the proportion of over-sampling
      k is the number of neighbors considered to create new instances
Result: dOut is the new re-sampled data set
var
    numNewNeigh, numMin, i is integer
    currEx, newEx, selectedNeigh is Example
    att is Attribute
    dOut, neighbors is Dataset
end
numMin := number of instances of the minority class in dOut
i := 0
dout := d
while i < numMin do
    currEx := get the ith example of the minority class
    neighbors := get the k nearest neighbors of the minority class closer to currEx
    numNewNeigh := N
    while numNewNeigh > 0 do
        selectedNeigh := randomly get a neighbor from neighbors
        /* create a new example of the minority class
        forall attribute att do
            newEx[att] := curEx[att] + (currEx[att] - selectedNeigh[att])· rand(0,1)
        end
        numNewNeigh := numNewNeigh - 1
        dout := addExample(dout, newEx)
    end
    i := i + 1
end
return dout
```

UNDERSAMPLING

- **Tomek Link Method**
  - A **Tomek Link** is a pair of instances <E_i,E_j> of *different* class from the dataset for which there is no other example E_k in the dataset that is closer to any of them
  - The collection of Tomek Links in the dataset define the *class frontiers*
  - This undersampling method removes all examples from the majority class that do not belong to Tomek Links

UNDERSAMPLING

- **Tomek Link Method**



UNDERSAMPLING

```
Algorithm: TomekLinks ( d is Dataset )
Data: d is the training data set
Result: Collection of Tomek links represented as pairs of examples
var
    setTomek is PairsExamples
    exMin, exMaj, ex is TrainingExample
    dst is double
end
forall  example of the majority class  exMaj  in  d do
    forall  example of the minority class  exMin  in  d do
        dst = dist(exMin, exMaj)
        if ¬ ∃ex ∈ d|dist(ex, exMin) < dst ∨ dist(ex, exMaj) < dist then
            cjtTomek := addLink (cjtTomek,<exMin,exMaj>)
        end
    end
end
```

WHY FEATURE REDUCTION?

Why even think about Feature Reduction?

- The information about the target class is **inherent in the variables**!

- Naive theoretical view:
  More features
  => More information
  => More discrimination power.

- In practice:
  many reasons why this is not the case!

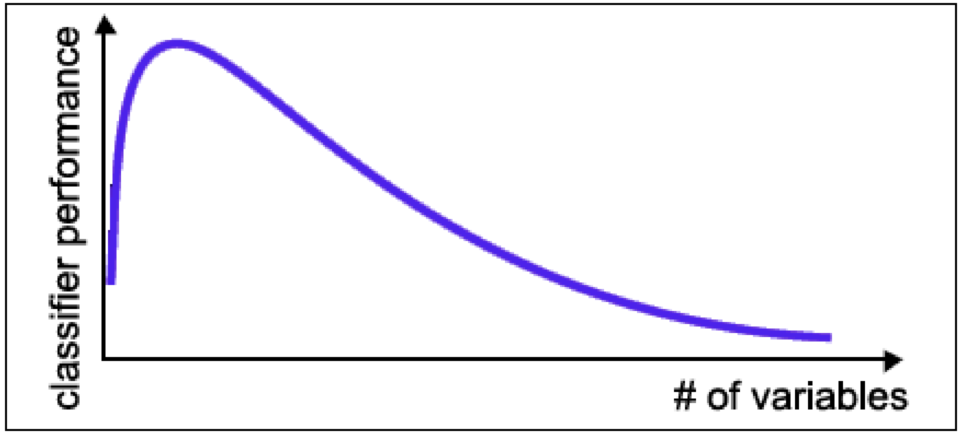
WHY FEATURE REDUCTION?

- Many explored domains have hundreds to tens of thousands of variables/features with many irrelevant and redundant ones!
- In domains with many features the underlying probability distribution can be very complex and very hard to estimate (e.g. dependencies between variables) !
- Irrelevant and redundant features can “confuse” learners!
- Limited training data!
- Limited computational resources!
- **Curse of dimensionality**!

CURSE OF DIMENSIONALITY

- The required number of samples (to achieve the same accuracy) grows exponentionally with the number of variables!
- In practice: number of training examples is fixed!

  => the classifier's performance usually will degrade for a large number of features!



In many cases the information that is lost by discarding variables is made up for by a more accurate training in the lower-dimensional space!

FEATURE SELECTION VS FEATURE EXTRACTION

- **Two general approaches for dimensionality reduction**
  - Feature extraction: Transforming the existing features into a lower dimensional space
  - Feature selection: Selecting a subset of the existing features without a transformation

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \\ x_N \end{bmatrix} \xrightarrow{\text{feature selection}} \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ \\ x_{i_M} \end{bmatrix} \qquad \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \\ x_N \end{bmatrix} \xrightarrow{\text{feature extraction}} \begin{bmatrix} y_1 \\ y_2 \\ \\ y_M \end{bmatrix} = f\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \\ x_N \end{bmatrix}\right)$$

FEATURE SELECTION

- Given a feature set $\boldsymbol{x=\{x_i \mid i=1...N\}}$ find a mapping $\boldsymbol{y= f(x):R^N\text{->}R^M}$ with M<N, such that the transformed feature vector $\boldsymbol{y_i}$ preserves (most of) the information or structure in R^N.
- An *optimal mapping* $\boldsymbol{y= f(x)}$ will be one that results in no increase in the minimum probability of error
  - There is no systematic way to generate non- linear transforms
  - The selection of a particular subset of transforms is problem dependent
  - Feature extraction is commonly limited to linear transforms: y=Wx
  - Other techniques: Principal Component Analysis, Kernel Methods

FEATURE SELECTION

- Given a feature set $\boldsymbol{x=\{x_i}$ | $i=1...N\}$ find a subset $\boldsymbol{x_M}$ $=\{x_{i1},x_{i2},...,x_{iM}\}$, with M<N, that optimizes an objective function J(Y), e.g. P(correct classification)
- **Feature Selection is necessary in a number of situations**
  - Features may be expensive to obtain
  - Want to extract meaningful rules from your classifier
  - When you transform or project, measurement units (length, weight, etc.) are lost
  - Features may not be numeric (e.g. strings)
- **Methods:**
  - <u>Filters</u>
  - <u>Wrappers</u>

FILTERS

- Filters requires:
  - A ***search strategy*** to select candidate subset
  - An objective function to evaluate the candidates

- Search strategy
  - Exhaustive search combinations $\binom{n}{m}$
    - Exhaustive evaluation of 10 out of 20 features involves 184756 feature subsets => unfeasible!
  - Heuristics
    - Sub-optimal feature space, but efficient!

FILTERS: VARIABLE RANKING

- Given a set of features F

  Variable Ranking is the process of ordering the features by the value of some scoring function (which usually measures feature-relevance)

- Resulting set:
  - a permutation of F: $F' = \{ f_{i_1}, \dots, f_{i_j}, \dots f_{i_n} \}$ with $S : F \to$ ¡
  - the score $S(f_i)$ is computed from the training data, estimating some measure of f_i.

- By convention **a high score** is indicative for **a relevant feature**.

$$S(f_{i_j}) \geq S(f_{i_{j+1}}); \quad j = 1, \dots, n-1;$$

FILTERS: RANKING CRITERIA

- Information Gain:
  - The number of "bits of information" gained by knowing the features
  - A **good feature** A can contribute, indipendently on each other feature, to reduce the uncertainty of class C given the value of the feature A.

$$\boxed{IG(C,A) = H(C) - H(C|A)}$$

$$H(C) = \sum_{k=1}^{K} P(c_k) \log P\left(\frac{1}{c_k}\right) \qquad H(C|A) = \sum_{k=1}^{K}\sum_{t=1}^{T} P(c_k, a_t) \log \frac{P(a_t)}{P(c_k, a_t)}$$

(probabilities are estimated from frequency counts)

FILTERS: SUBSET EVALUATION

- Correlation-based Feature Selection (CFS):
  - A **good feature subset** S contains ***g*** features highly correlated with a specific class value, yet uncorrelated with the others classes

how predictive of the class the set of features S is

$$M_S = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}$$

redundancy among the features

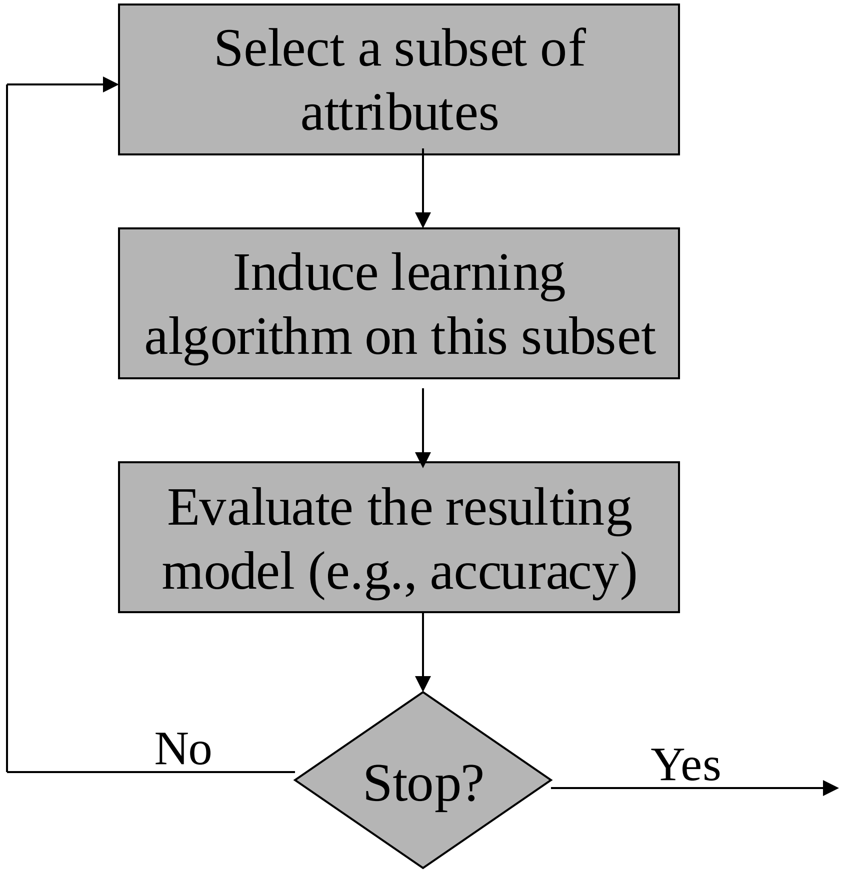
  - k denotes the number of features in the subset S
  - r_{cf} is the mean feature-classcorrelation
  - r_{ff} is the means feature-feature intercorrelation.

WRAPPERS

- Learner is considered a black-box
- Interface of the black-box is used to score subsets of variables according to the predictive power of the learner when using the subsets.
- Results vary for different learners
- One needs to define:
  - how to search the space of all possible variable subsets ?
  - how to assess the prediction performance of a learner ?

WRAPPERS

- “Wrap around” the learning algorithm
- Must therefore always evaluate subsets
- Return the best subset of attributes
- Apply for each learning algorithm
- Use same search methods as before



WRAPPERS

- The problem of finding the optimal subset is NP-hard!
- A wide range of heuristic search strategies can be used.
- Two different classes:
  - **Forward selection:**
    - start with empty feature set and add features at each step
  - **Backward elimination:**
    - start with full feature set and discard features at each step
- Predictive power is usually measured on a validation set or by cross-validation
- By using the learner as a black box wrappers are universal and simple!
- Criticism: a large amount of computation is required.