# Discovery Systems

PETRA POVALEJ, MATEJA VERLIC, GREGOR STIGLIC
Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor, Slovenia

## Article Outline



## Glossary

**Accuracy (rate)** Used for evaluating quality of induced model.

**Average class accuracy** One of the simplest metrics for estimating the quality of a model. Classification accuracy is calculated for each class of the target variable and then the average of all accuracies per class is calculated.

**Aggregation** Process of combining two or more objects into single one. Typical statistical aggregation functions for quantitative attributes are sum and average.

**Attribute** A property or characteristic of data object, which may vary in time and also from object to object. Attributes have usually assigned values or symbols for the purpose of analysis. Other frequently used names for an attribute are variable and feature.

**Binarization** Transformation of continuous or discrete attributes into binary attributes. Binary attributes have only two possible values.

**Classification** Classification of data objects is a process of assigning classes or class labels to data objects. It is a type of predictive modeling and it is used for predicting discrete target variable.

**Classification accuracy** See description under *Accuracy (rate)*.

**Classifier** A model based on data used for classification.

**Confusion matrix** A matrix of results from testing model versus predicted class values. It is very useful visual tool for understanding results of testing a classification model.

**Data cleaning** Step of KDDM usually involving detection and correction of data quality problems, removal of noise, defining outliers, and dealing with missing values.

**Data mining** A technology combining traditional data analysis methods and sophisticated algorithms for automatically processing large volumes of data and finding and extracting novel, useful and usually hidden patterns.

**Data object** A record of attribute values about an object or person. Other common names for data object are data record, case, point, sample, observation or entity.

**Data preprocessing/preparation** A phase of KDDM related to the preparation and transformation of data for data mining. It comprises of several techniques for selecting relevant data and attributes and creating or changing the attributes.

**Data set** A collection of similar or related data objects. Data objects are usually collected for a particular study.

**Dimensionality reduction** Reduction in the number of attributes. It is used for eliminating irrelevant features and noise by creating new attributes as a combination of the old attributes. Feature subset selection or feature selection is other type of dimensionality reduction, where dimensionality is reduced by selecting and using only a subset of old attributes.

**Discretization** A transformation of continuous numerical attributes into categorical or discrete attributes.

**Ensemble** Also known as committee or multiple classifier system is a group of classifiers. Ensemble approaches exploit the classification abilities of multiple classifiers. The integration of classifiers usually enhances the performance of final classification.

**Feature** A feature (variable) is a synonym for attribute. It is frequently used in data-mining domain. See *Attribute*.

**Feature extraction** A process of creating new features from the original raw data. It is highly domain-specific.

**Knowledge discovery and data mining (KDDM)** The "umbrella" term for the overall process of knowledge discovery.

**Knowledge discovery (KD)** Nontrivial process of mapping low-level data into other more meaningful forms that are easier to understand, like patterns, rules, summaries or even graphs.

**Noise** Result of erroneous measurements. It can involve distortion of values or addition of unauthentic data objects. Unlike outlier, noise is not legitimate data.

**Outlier** An anomalous object or atypical value of an attribute. Outliers can be legitimate data objects or val-

ues. Detecting outliers is especially important in fraud detection or network intrusion detection.

**Pattern** In KDDM defined as a high-level description of a subset of data and can be in many forms, e. g. statistical or predictive models of data, relationships among parts of data sets, association rules, clusters, graphs, summaries, or classification rules, tree structures, linear equations etc.

**Precision** Fraction of positive samples correctly classified as positive among all samples classified as positive.

**Recall** See *Sensitivity*.

**Regression** Regression is a type of predictive modeling used for predicting continuous target variable.

**Sampling** A process of selecting a subset of data or sample, for the data analysis. Basic sampling techniques are simple random sampling with or without replacement, stratified sampling and adaptive or progressive sampling.

**Sensitivity (recall)** Proportion of samples correctly classified as positive (true positives) of all positive samples tested. If sensitivity is 1, all positive samples have been identified as positive.

**Specificity** Proportion of samples classified as negative of all negative samples tested.

## Definition of the Subject

By definition, to *discover* is to see, get knowledge of, learn of, find or find out; gain sight or knowledge of something previously unseen or unknown [18], therefore a discovery system can be defined as a system that supports the process of finding new knowledge. Results of a simple query for *discovery system* on the World Wide Web returns different types of discovery systems: from knowledge discovery systems in databases, internet-based knowledge discovery, service discovery systems and resource discovery systems to more specific, like for example drug discovery systems [10], gene discovery systems [43], discovery system for personality profiling [48], and developmental discovery systems [17] among others. As illustrated variety of discovery systems can be found in many different research areas, but we will focus on knowledge discovery and knowledge discovery systems from the computer science perspective. Inconsistent definitions of terms knowledge discovery (abbreviated, KD), knowledge discovery in databases (abbreviated, KDD) and data mining (abbreviated, DM) frequently confuse potential a novice in the field of knowledge discovery.

DM is a technology combining traditional data analysis methods and sophisticated algorithms for automatically processing large volumes of data and finding and extracting novel, useful and usually hidden patterns. DM is also associated with other names, such as knowledge extraction, information harvesting, data archaeology, data pattern processing, and knowledge discovery in databases. DM and KDD are often used as synonyms, although DM is an integral part of KDD.

KD is a process of mapping low-level data into other more meaningful forms that are easier to understand [21] and knowledge discovery systems are systems that implement knowledge discovery process. Also, KD can be defined as a process that seeks new knowledge about specific problem or application domain.

KDD is a KD process applied to databases [30]. In the most popular definition by Fayyad et al. [21] KDD is defined as a non-trivial process of identifying valid, novel, potentially useful, and understandable patterns in data. Note that although databases are considered as a primary source of data in KDD, the definition itself is not limited to one type of data sources only.

The umbrella terms *knowledge discovery* and *data mining* (abbreviated, KDDM) was proposed as the most appropriate name for the overall process of KD [30] instead of *knowledge discovery process* (abbreviated, KDP) and it will be used from now on to describe the overall process of knowledge discovery from different sources of data including methods for storing and accessing data, scalability of algorithms for large data sets, interpretation and visualization of results, and modeling of human-machine interaction [21].

The importance of KD systems lies in discovering the knowledge that may otherwise remain hidden and to integrate extracted knowledge into decision support systems used by scientist and businesses. The need for automated or at least semi-automated data analysis arose when computers revolutionized our way of living and making business. The digital information era brought advancements and benefits but also a very serious pitfall - data overload or data flood. Waste amounts of data, currently measured in terabytes, are collected daily, much more than human analysts can examine in foreseeable time. Still, the importance of human involvement in KD is not to be neglected. KD can only assist people with finding hidden patterns and relationships in data; it cannot tell whether discovered patterns are valuable for the organization.

## Introduction

Knowledge discovery is not a new approach to analyzing data. People soon discovered that data need to be analyzed to make sense. The term *knowledge discovery in databases* was introduced in 1989, when first workshop on KDD was

organized [35]. Soon after that, in 1991, Frawley [23] published a definition of KDD, which was later revised by Fayyad et al. [21]. The revised definition of KDD became very popular in KD community.

Classical approach to data analysis involves one or more analysts who need to be closely acquainted with the data. The analysts act as an interface between the data and the end-users of gained knowledge. Although this approach is perhaps appropriate for small amounts of data or with data with small number of variables or attributes, it has many drawbacks if larger amount of data or large number of variables are involved. Unfortunately, these drawbacks cannot be afforded in the modern high-paced business world. First, analyzing large amounts of data manually is too slow to effectively apply the results of the analysis. Second, this approach is too expensive because experts need to be very familiar with the data (their expertise is valuable) and third, the level of subjectivity of results is too high. Hence, the urge for at least semi-automated data analysis became even stronger.

Several computational approaches to data analysis have been proposed as an alternative to the classical approach. They are based on the idea that while on one hand computers contribute to data overload, they can on the other hand assist humans in extracting useful knowledge by finding meaningful patterns from growing volumes of data. For example, statistical, artificial intelligence and machine learning and even cognitive approaches can all deal in the same problem domain but from different perspectives and in different ways using different models, methods, and techniques.

In contrast to single-disciplinary approaches, knowledge discovery in databases is an interdisciplinary approach. It evolved from several research fields and it incorporates many techniques and findings from statistics, artificial intelligence, machine learning, pattern recognition, databases, and data visualization among others. All those disciplines are combined in KDD with one common goal: extracting high-level knowledge from low-level data.

Before the beginning of the 21st century Piatetsky-Shapiro [36] had already identified three generations in KD systems. First-generation KD systems (early-1990s) were intended for expert users and provided only single data mining technique for data analysis per system. They had very weak support for the overall KDDM process and had only limited commercial success. The main research was focused on the development of new data mining algorithms. Second-generation KD systems (mid-1990s), called suites, were collections of multiple data analysis methods with the support for data cleaning, preprocessing and visualization. They were certainly a step forward, but the idea of KDDM process model was still not implemented. Third-generation KD systems (late-1990s) introduced different approach by addressing specific business problems like fraud detection. They provided interface to hide complexity of underlying data mining techniques and introduced first KDDM process models. Now, when we are approaching the second decade of the 21st century, it is possible to identify new generation of KD systems. This new generation of KD systems aims at integration and interoperability of modern KDDM models through use of popular industrial standards like eXtensible Markup Language (abbreviated, XML) and Predictive Model Markup Language (abbreviated, PMML) or some other approaches [30].

After the first workshop on KDD the idea of KDDM model evolved simultaneously with the development of KD systems. In 1996 Fayyad et al. [22] laid the foundation for KDDM process model by proposing the basic structure of the model. After the publication of their book the development of KDDM models followed two different streams: data-centric and human-centric. Data-centric (or data-driven) models focused on the iterative and interactive nature of the data analysis task, while the human-centric models focused on a series of knowledge-intensive tasks with complex interactions between human and the database [30]. Furthermore, the models can be broadly divided into academic, usually not considering industrial aspects of KDDM projects, and industrial, which address specific industrial issues. All process models consist of multiple sequential steps with loops and interactions; they differ in the number and the scope of proposed specific steps. Also, all models emphasize the iterative nature of the model.

Although many models have been developed, only five of them left significant impact and were applied in at least several real KDDM projects [30]. The nine step academic data-centric KDDM model by Fayyad et al. [22] in 1996 was the first reported model. In the same year the CRISP-DM (Cross-Industry Standard Process for DM) model was proposed by a consortium of four companies: SPSS, NCR, Daimler Chrysler and OHRA, but its mature version was released in 2000. The second model, this time from industrial area, was suggested in 1998 by Cabena et al. [9] and it consists of five steps. Third model was an eight-step academic model by Anand and Buchner [1], developed at about the same time as the second model. In 2000 the fourth model, already mentioned CRISP-DM model consisting of six steps, was officially released [44]. This model is still strongly supported by the industry, CRISP-DM Special Interest Group and the European Commission. The last, fifth model consisting of six steps, was proposed by

Cios et al. [11]. This model adopted CRISP-DM model to the needs of academic research community [30]. The study by Kurgan and Musilek [30] offers detailed description and an exhaustive comparison of the major five existing KDDM models. They also introduced a generic model, which tries to capture main steps and tasks of the mentioned KDDM models. In following sections and subsections this generic model was used in order to avoid favoring any particular KDDM model. Using KDDM models for knowledge discovery is essential to ensure that useful knowledge is discovered; blind application of data mining methods, notoriously known as data dredging or data fishing can easily lead to the discovery of potentially interesting but meaningless patterns.

Other pioneers, who were not explicitly mentioned above but also contributed significantly to the development of contemporary KDDM, are P. Smyth [21,26] and H. Manilla [26,32].

In the following sections knowledge discovery process will be described in more details. Each step of the process will be briefly explained to show how a particular step contributes to the results of the entire KDDM process. Detailed description of the KDDM process will be followed by a section on KDDM frameworks, which will include a review of contemporary frameworks. In next section several applications of KDDM will be mentioned so that we can recognize the real value of applying KD to the real world problems. In the last section some guidelines for the future development of KDDM will be suggested.

## Knowledge Discovery and Data Mining Process

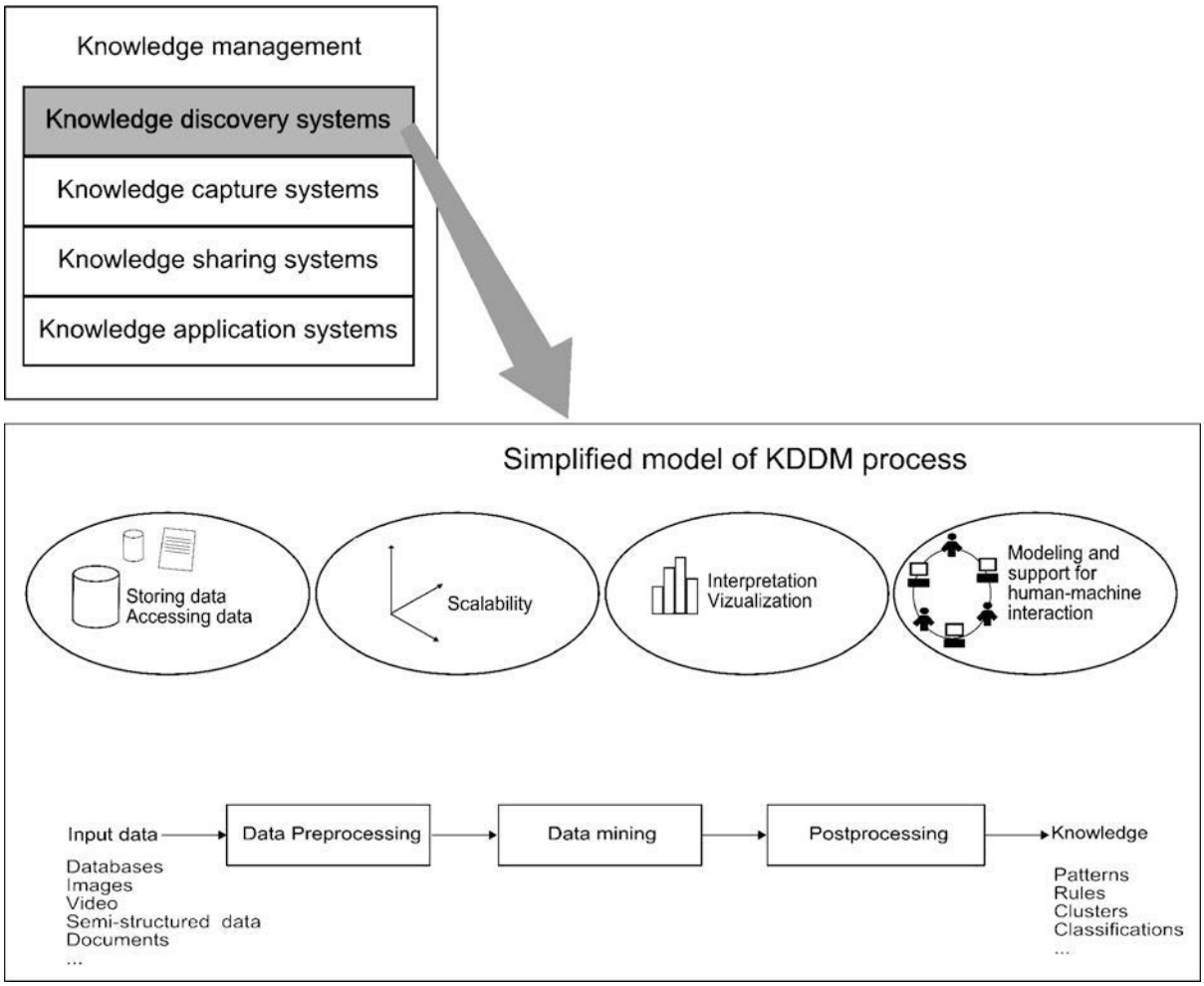
The simplest definition of KDDM is an overall process of transforming raw data into knowledge. As already mentioned in the introductory section, KDDM process includes, among others, methods for this transformation, which is the central part of KDDM process. Fayyad et al. [21] defined this transformation as KDD and described it as 'the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data'. Before we describe KDDM in more details, it is important to understand the meaning of basic terms in this definition. **Data** is a set of facts and a **pattern** is an expression (high-level description) that describes a subset of the data or a model applicable to this subset. Patterns, as mentioned in definition, can take many forms; for example, they can be statistical or predictive models of data, relationships among parts of data sets, classification rules, association rules, summaries, linear equations, clusters, graphs, tree structures, or recurrent patterns in time series. In this context a pattern is considered as **knowledge** if it exceeds a certain level of **interestingness**, usually set by user or even data itself. The notion of interestingness was introduced by Fayyad et al. [21] and it represents an overall measure of pattern's value, novelty, usefulness, and understandability. Because functions and thresholds for measuring interestingness are chosen by user, this definition of knowledge is purely user oriented and domain specific. As we may intuitively know, knowledge is much more than just patterns or relationships in data. For instance, researchers from the field of knowledge management define knowledge as a mix of experience, values and insights that provides some sort of framework for evaluating new experiences and information. It is embedded not only in documents but also in organizational routines, processes, practices and norms [14]. This broader definition of knowledge is certainly better than the definition of knowledge as specific patterns, but unfortunately it is not very useful in the context of KDDM.

Figure 1 shows the relationship between knowledge management, knowledge discovery systems, which implement KDDM process, and data mining. KDDM process is essential part of an even broader process named knowledge management. While knowledge management deals with aspects of gathering, organizing, sharing and analyzing intellectual capital – knowledge [4], KDDM focuses on the overall process of knowledge discovery from data, including methods for storing and accessing data, scalability of algorithms for large data sets, interpretation and visualization of results, and modeling of human-machine interaction. Simplified model of KDDM process clearly illustrates the central role of DM in KDDM process.

KDDM is a process of several steps which transforms inputs in form of raw data into outputs – applicable knowledge. Different KDDM models define steps involved in this process, but the number and the scope of particular step vary from model to model. We decided to describe the process of KDDM using the generalized model of KDDM process proposed by Kurgan and Musilek [30]. Figure 2 shows the steps of generalized model around the schematic diagram of general sequential structure of KDDM process [12] in the center. The sequential structure includes many feedback loops between steps, because the process of revision can show that some of the steps need to be repeated.

The generalized KDDM model has six steps:

(1) Application domain understanding,
(2) Data understanding,
(3) Data preparation and identification of DM technology,
(4) Data mining,




**Discovery Systems, Figure 1**
**Relation between knowledge management, knowledge discovery systems, and data mining**

(5) Evaluation and
(6) Knowledge consolidation and deployment.

Each step will be described in more details in the following sections.

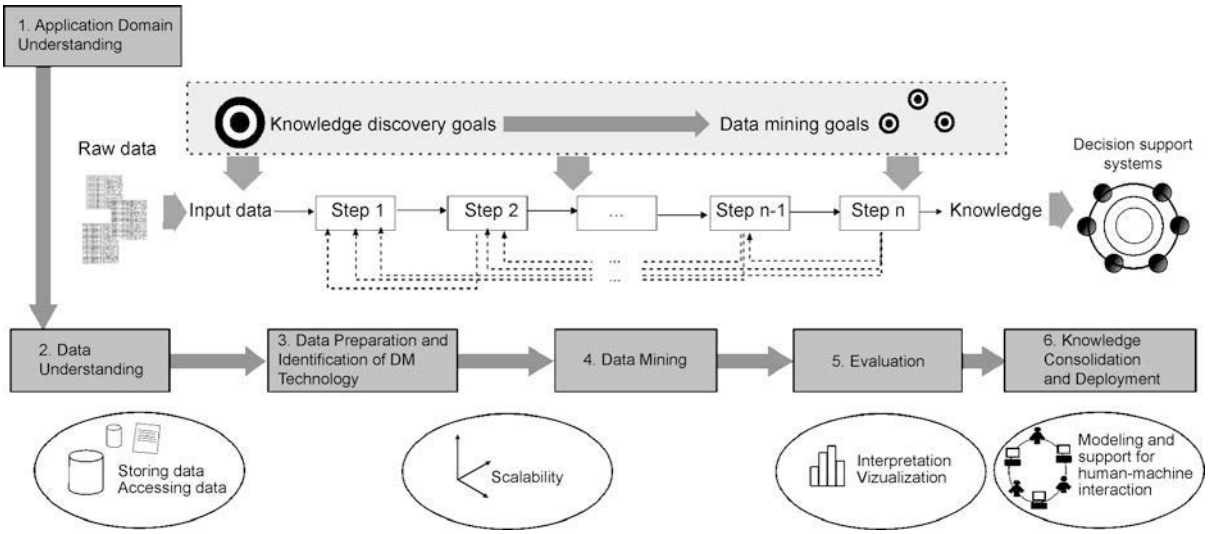
### Application Domain Understanding

Before analyzing data it is recommended to define the goals or objectives of the analysis. First, we have to understand the domain objectives that are important for the end-user, for example a customer. We have to carefully consider constraints, assumptions and any other factors that may affect our search for new knowledge. Second, some project management efforts are needed to identify human and technical resources, decompose KDDM project into tasks, estimate the duration of each task, define inputs, outputs, and dependencies to successfully apply data mining techniques. Each task should also be associated with particular KD goal. Third, at this point business goals or KD goals need to be translated into more specific DM goals, so that later one the most appropriate DM techniques can be chosen and applied.

Furthermore, understanding relevant prior knowledge can also help to effectively execute further steps of the process.

### Data Understanding

When knowledge discovery goals and more specific data mining goals are known, we can start to collect data included in the resources we included in the plan. Resources may include transactions data, customer histories, demographic information, protein sequences, microarray expression data or weather data. In this phase fusion of data




**Discovery Systems, Figure 2**
**Generalized model of KDDM process**

from multiple sources has to be performed and we have to familiarize with data, so we can more easily identify potential data issues. Diversity of resources contributes to diversity of formats in which data can be stored, for example we can find data in flat files, relational tables, spreadsheets or even semi-structured documents at either centralized or distributed locations.

A **data set** can be considered as a set of **data objects** (also known as records, cases, points, samples, observations, entities) and data objects are described by **attributes** (also known as variables, characteristics, features, dimensions). For example, in files with a record data set data objects are written as records in rows and each field or column corresponds to an attribute (Table 1).

Attribute is a property or characteristic of a data object that may vary from object to object and even from one time to another. We assign numbers or symbols to attributes so we can analyze characteristics of object. As it can be seen from the example, different types of values can be assigned to attributes. Which type of value, either symbolic or numerical, will be associated with attribute is defined by a rule or function called a **measurement scale**.

Note that properties of an attribute and the properties of the values for measuring it do not have to be the same. For example, patient's identification number (ID) and age can be both represented as numeric values, usually integers. While it is quite reasonable to calculate average age of the patients, it is senseless to calculate average ID, because the purpose of using IDs is to distinguish data objects. Also, it can happen that an attribute is measured in such a way that measurements do not capture all the properties of the attribute. Thus, knowing the type of an attribute is important. It tells us which properties of the measured values are consistent with the properties of the attribute, so we can avoid senseless actions.

According to the properties of numbers that correspond to the properties of attribute we can define four types of attributes: nominal, ordinal, interval, and ratio. **Nominal** attributes are described with symbols or numbers which distinguish one object from another. Nomi-

**Discovery Systems, Table 1**
**Sample data set containing information of patients. Each row in a table is a record and attributes are person identification number, name, age, gender, height, weight, level of tiredness, blood sugar, and body temperature**

| Person ID | Name | Age | Gender | Height (cm) | Weight (kg) | Level of tiredness | Blood sugar | Body temp. (°C) |
|---|---|---|---|---|---|---|---|---|
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **13678** | John Doe | 37 | M | 183.5 | 84.5 | Low | Normal | 35.7 |
| **13679** | Jane Smith | 28 | F | 158.2 | 75.3 | Medium | High | 37.1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |

nal attributes from the example in Table 1 are identification number, gender and name. **Ordinal** attributes provide information to order the objects. Examples of ordinal attributes are level of tiredness and blood sugar. For **interval** attributes, the difference between values are important and unit of measurement exists. An example of interval attribute is body temperature in the Celsius scale. For **ratio** attributes, both differences and rations are important. Examples of ratio attributes are age, height and weight.

Nominal and ordinal attributes are **categorical** or **qualitative** attributes, while interval and ratio attributes are **quantitative** or **numeric** attributes. Qualitative attributes take on values that are names or labels and do not have properties of numbers. Even if their values are numeric, they should be treated more like symbols. Quantitative attributes are represented by numbers and have most of the properties of numbers.

If an attribute can have any value between two specified values, then it is **continuous**; but if it can take only a limited or finite number of values it is **discrete**. Example of continuous attributes is weight, while gender is an example of discrete attribute.

Discrete attributes are often represented using integer values. **Binary attributes** are special case of discrete attributes. They can have only two values like yes/no, true/false, male/female or 1/0 and are usually represented with Boolean variables or as integer variables. Gender, level of tiredness and blood sugar from Table 1 are examples of discrete attributes. Continuous attributes have values which are real numbers and are typically represented as floating point variables. Height, weight and body temperature from Table 1 are examples of continuous attributes.

Different types of data sets exist. Most common types are data sets with record data (e. g. transaction data, data matrix, and sparse data matrix), graph-based data (e. g. data with relationships, data as graphs) and ordered data (e. g. sequential data, time series data, and spatial data). Data sets may even display some special characteristics. For example, when we are working with time series data, we should consider temporal autocorrelation; or when we are dealing with spatial data, we should not ignore spatial autocorrelation. Furthermore, some data sets might even include explicit relationships within data (e. g. linked documents in the WWW). Many data sets share three general characteristics that significantly affect data mining techniques: dimensionality, sparsity, and resolution.

The **dimensionality** of a data set is the number of attributes that are used to describe data. Sometimes it is possible to come across the term **curse of dimensionality**. This term is used for difficulties that usually occur during analysis of high-dimensional data and therefore dimensionality reduction is an important part of data pre-processing phase.

In some data sets only few attributes of an object have non-zero values. In this case we can talk about **sparsity**. Attributes, for which only non-zero values are important, are called **asymmetric attributes**. Sparsity in data set can be considered as an advantage because we don't have to store, manipulate or analyze non-important (zero) values. Some data mining algorithms work well only for sparse data sets.

Data is often obtained at different levels of resolution. The properties of data at different resolution levels are different. For example, we measure weight of babies in grams, while the weight of adults is measured in kilograms. If the resolution is too fine, a pattern might not be visible or too much noise is present; on the other hand, if resolution is too coarse, the pattern may disappear.

In this phase we also have to deal with several data-related issues: the types of data in our selected data sets, the quality of data, how to improve data quality or how to modify data for the needs of data mining methods. Data is never perfect: values may be missing, duplicate objects may exist, inconsistencies may occur due to human error, hardware and software limitations, or flaws in the data collection process. Detection and correction of data quality problems is often called **data cleaning**. When dealing with data quality we have to consider measurement and data collection issues and also issues related to applications. We will briefly describe some of most frequently occurring data errors: noise, outliers and missing values.

We can encounter a variety of problems related to **measurement error**, when recorded value differs from 'true' value to a certain extent, and **data collection error**, which refers to omitting data objects or attribute values or including data object in inappropriate way. Both errors can be systematic or random. Good news is that for some types of data errors well-developed techniques for detecting and correcting these errors exist.

**Noise** is an example of measurement error. Often it occurs randomly and it may involve distortion of values or addition of unauthentic objects. Usually, the term noise is used in connection with temporal or spatial data, but is not uncommon in other types of data as well. The elimination of noise is difficult, therefore **robust** (insensitivity for noise) data mining algorithms are desired. Precision, bias and accuracy are also important factors for measuring the quality of data.

**Outliers** are data objects that are different from most of the other data objects or values of an attribute that are atypical for this attribute. Outliers are considered

anomalous objects or values. It is important to distinguish between noise and outliers. Unlike noise, outliers can be legitimate data objects or values and may sometimes be interesting, especially in fraud detection or network intrusion detection.

**Missing values** are not uncommon in data sets. A data object can have one or more missing attribute values; sometimes the values were not collected or attributes are not applicable to all objects or attributes are optional. Usually, for the sake of simplicity, all attributes (fields) are stored, but during the data analysis all missing values should be included.

The problem of **duplicate data** should also be taken into an account in data cleaning. Data duplication can occur due to inconsistent values and these values must be resolved. However, duplication can also occur due to accidentally combining data objects, that are similar but not the same. For example, two or even more persons can have identical names.

### Data Preparation and Identification of DM Technology

Data pre-processing phase is related to preparation and transformation of data into more suitable form for data mining. This particular step is by far the most time-consuming part of the KDDM process [12].

Data pre-processing strategies and techniques can be roughly divided into two groups – strategies and techniques for:

(1) Selecting relevant data object and attributes, and
(2) Creating or changing the attributes.

We will familiarize with terms aggregation, sampling, dimensionality reduction, feature subset selection, feature creation, discretization, binarization, and variable transformation. In data pre-processing the term feature or variable is frequently used as a synonym for attribute.

The idea behind **aggregation** is to combine two or more objects into a single object. Typically, quantitative attributes are aggregated by simple statistic functions like a sum or an average. Qualitative attributes are more difficult to deal with. They can be either omitted or summarized in some way. If aggregation is used, then resulting smaller data sets require less computer resources (memory and processing time) and thus more expensive data mining techniques can be used. Aggregation can also serve as a high-level view of the data. However, aggregation can also lead to potential loss of interesting details.

**Sampling** is an approach for selecting a subset of data objects for the data analysis. It has been long used in statistics and it can be also very useful for data mining. In statistics sampling is used because obtaining the entire set of data of interest is too expensive or even impossible; in data mining sampling is used because processing of all the data would be too expensive or time consuming. Sampling can, similar to aggregation, reduce the size of data set and thus more expensive data mining algorithms can be used. Sampling is effective only if the sample is **representative**; in other words, a sample should have very similar or even identical characteristics as the original (entire) set of data or **population**. Different sampling approaches have been developed. One of the basic sampling techniques is **simple random sampling**, which has two variations:

(1) Sampling without replacement and
(2) Sampling with replacement.

In random sampling all items in the population have equal opportunity to be selected. In sampling without replacement an item that was selected is removed from the population. In sampling with replacement selected item is not removed from the population, so the same item can be selected more than once. Because simple random sampling is not suitable for less frequent types of objects (rare cases), different sampling method is needed for such data sets. **Stratified sampling** takes into account differing frequencies of objects types. It starts with predefined groups of objects. In one version equal number of objects is drawn from each group, regardless of potentially different sizes of groups. In another version, the number of objects drawn from each group is proportional to the size of corresponding group. After selecting sampling technique, we have to decide on the **sample size**. Sample size is very important, because if we choose large sample sizes the probability of having a representative sample is higher, but we lose advantage of sampling. On the other hand, if we use a smaller sample size we may miss some patterns or even detect erroneous patterns. Because defining proper sample size can be difficult, we can sometimes use **adaptive** or **progressive sampling**. In this kind of approach, sample size is progressively increased until sufficient size is obtained, but we need a way to determine if the sample size is large enough.

Data sets can have a large number of attributes or features, even tens of thousands attributes. Because many data mining algorithms are more suitable for data sets with smaller number of attributes, reduction in the number of attributes or **dimensionality reduction** is desired. Dimensionality reduction has many benefits, a key one is that data mining algorithms work better with lower dimensionality of data. This is partly due to elimination of irrelevant features and noise reduction. Dimensionality reduction can also lead to more understandable model with

fewer attributes, data can be more easily visualized and data mining algorithms require less computer resources. Dimensionality reduction techniques usually reduce the dimensionality by creating new attributes as a combination of the old attributes. Dimensionality of data sets can be also reduced by selecting and using only a subset of old attributes. This reduction is known as **feature subset selection** or **feature selection**. Redundant features duplicate much or all information contained in one or more features while **irrelevant features** contain little or no information that could be used in data mining. Redundant and irrelevant features can even reduce classification accuracy and the quality of clustering and eliminating them can only be beneficial. Some of features can be eliminated immediately, but otherwise selecting the best subset of features requires a systematic approach. Three standard approaches to feature selection exist: embedded (feature selection occurs naturally as a part of data mining), filter (features are selected before data mining), and wrapper (learning algorithm itself is used as part of the evaluation function for selecting features). Sometimes we want to create new set of attributes from original ones, especially if new features capture important information more effectively. Creating new features from the original raw data is known as **feature extraction**, but unfortunately this approach is highly domain-specific. For some cases data need to be mapped to a new space to reveal interesting and important features, for example, using a Fourier transformation in time series analysis. In situation, when the features in the original data set have the necessary information but they are not in the form suitable for data mining, constructing one or more features from the original ones can be more useful. The most common approach to **feature construction** is using domain expertise.

Several data mining algorithms, especially in classification or association analysis, require data in the form of categorical attributes of binary attributes. Normally attributes in data sets are not only categorical or binary and transformation of attributes is needed. Transforming continuous attributes into categorical attributes is **discretization** and transformation of continuous or discrete attributes into binary attributes is **binarization**. There are many different approaches to discretization and binarization, but they all try to produce the best result for data mining algorithm we intend to use for the data analysis. Another type of transformation is **variable transformation** that is applied to all values of an attribute or variable. Two important types of variable transformations are simple functional transformations and normalization. For **simple functional transformations** a simple mathematical function is applied individually to each value. For example, if $x$ is a variable such simple functions are $x^k$, $\log x$, $e^x$, $\sqrt{x}$, $1/x$, $\sin x$, or $|x|$. Note that variable transformations should be applied with caution, because they change the nature of data (e. g. changing magnitudes, order, or negative values). Standardization or normalization of a variable is concerned with transforming entire set of values in such a way, that they share particular property. This transformation is often necessary if are combining different attributes (variables) in some way and we want to omit the dominance of large values.

Some researchers recommend choosing DM technology before data preparation, others say that methods should be selected we have prepared, but in practice these two things are intermingled. Data need to be prepared at least in such a way that we can decide which DM methodology is most appropriate for selected DM task. Among core DM tasks are predictive modeling, anomaly detection, association analysis and cluster analysis.

With **predictive modeling** we build a model for the target, or dependent variable as a function of other independent variables. Well-known types of predictive modeling are **classification**, which is used for predicting discrete target variable, and **regression**, which is used for predicting continuous target variables. The goal of predictive modeling is to minimize the error between predicted and true values of the target variable [47]. **Anomaly detection** is identification of data samples, known as **anomalies** or **outliers** that are significantly different from the rest of data. The main goal of this task is to identify the real outliers not the noise. **Association analysis** is used to discover strongly associated features and patterns that occur in data because of these associations. Discovered patterns are usually in the form of implication rules or feature subsets. The main goal of this analysis is efficient extraction of the most interesting patterns, for example, which items are frequently bought together by customers. **Cluster analysis** is a task of finding groups or **clusters** of data samples. Data samples in a cluster are more similar to each other than samples of other clusters.

After identifying DM tasks, we can choose one or more of appropriate algorithms for selected DM task.

## Applying Data Mining

When applying DM process we can generally pursue two different types of objectives: either to verify some user predefined hypotheses (**verification**) or find new knowledge upon which a user can base decisions using (**knowledge discovery**). Both verification and knowledge discovery are done by building a model of a real world based on collected data in a specific application domain. The

result of model building is a description of patterns and relationships in data, which can be used for verification of predefined knowledge (hypothesis), discovery of new knowledge for either **prediction**, where the system autonomously finds patterns for predicting future behavior of some entities (class attributes), or **description**, where the system finds patterns for a presentation in a human-understandable form. The boundary between prediction and description is blurred – some prediction models can also be understandable to some degree.

After selecting specific goals of DM process and the type of prediction, which is the most important for mining the data from the data set, a DM model type can be chosen. Several traditional statistical models exist, for example discriminant analysis, general linear models and logistic regression, but there are also some models from the fields of machine learning and pattern recognition, such as neural net to perform regression, a decision tree or a decision rules for the classification.

Furthermore, several algorithms are available for building selected model. For example, we may build a neural net using backpropagation or radial basis functions; for inducing a decision tree we can choose among C4.5 [39], CART (classification and regression trees) [7], CHAID (CHi-squared automatic interaction detector) [29], QUEST (quick, unbiased, efficient, statistical tree) [31] and many other algorithms. Algorithms tend to differ primarily in the goodness-of-fit criterion used to evaluate model fit or to find the good fit in a process of searching.

In predictive models, the values (classes) we are predicting with a model are called **response/dependent/target variables** and the values used as an input to build a model are called **prediction/independent variables**. The process of building predictive models requires a well-defined training and validation protocol in order to ensure the most accurate and robust predictions. This kind of protocol is called **supervised learning**. The idea of supervised learning is to *train* a model on a portion of data from the data set (**a training set**), then test and validate it on remainder of the data (**test set**). For each observation in a data set values predicted by the model are compared with actual (known) values of the response variable. For contrast, some of the descriptive techniques like clustering can build a model without known values of the response variable. These types of techniques are sometimes referred to as **unsupervised learning**.

It is important to remember that no model or algorithm should be used exclusively. Model building is an iterative process. The nature of the problem and the data itself affect the choice of models and algorithms. There is no best model or algorithm in general. Selection of the most appropriate model type and algorithm can be crucial for the success of DM process. Therefore for a given problem different models should be explored to find the one producing the most useful solution.

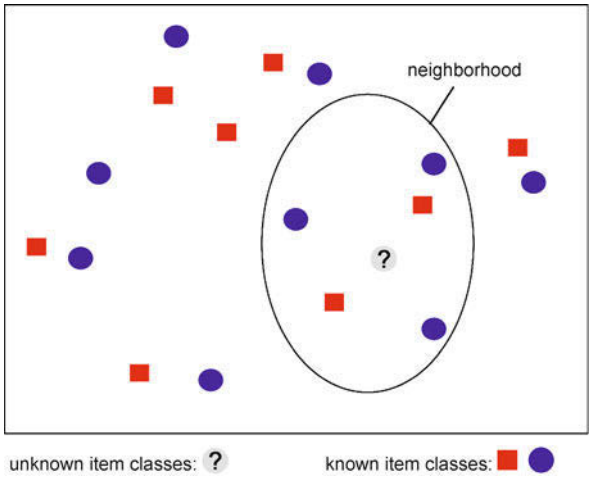
In the following sections, some of the types of models and algorithms used to mine data will be briefly examined. In general, most of the model types can be viewed as an extension or a hybrid of some primary types of models. Advanced descriptions of models can be found in other sections of this chapter.

### K-Nearest Neighbor (k-NN)

K-nearest neighbor (k-NN) is a classification technique that defines how to classify an observed item on the basis of the majority class of $k$ items in the neighborhood (Fig. 3).

The idea of k-NN is based on measuring a distance between attributes in the database. If the attributes are numerical the calculation of distances is easy, but categorical attributes require special handling (e. g. what is the distance between blue and gray?). The next task is a selection of the neighborhood of pre-classified cases to be used for classification of a new case and possibly the influence of neighbors considering the distance from the new case. For example, nearest neighbors can have a higher influence (weight) on classification of a new case than the neighbors that are father away.

Since k-NN models are computationally demanding they are mostly useful when only a few prediction variables




Discovery Systems, Figure 3
**k-Nearest neighbor. ? are known classes of the neighbors. In most simple algorithm the class of unknown item ? and 2 items have class**

are included. Different data types of variables can be included into the model under only one basic requirement: the metric for measuring distance has to be defined. The advantage of k-NN models is their understandability.

### Memory–Based Reasoning (MBR)

MBR is one of the memory based technologies that belongs to the family of nearest-neighbor-like models. It is used on databases with large number of cases where the data is kept in a system's memory in order to speed up computations. As in k-NN model, MBR algorithms search for the most similar problem from the database of pre-classified cases and apply its class to a new case. There is no learning phase and no adaptive or learning parameters to manipulate.

A basic advantage of MBR algorithms is their effectiveness (on a proper database) that is comparable (and sometimes even better) to neural networks. The disadvantages of MBR algorithms are: a lack of generalization (a large number of good examples are needed), no data compression, large computational demands and consequently large memory requirements that all make them unsuitable for on-line learning.

It has been often used for object recognition and classification of free text samples taken from very large databases.

### Rule Induction

Rule induction algorithms derive a set of independent rules from the database of pre-defined cases to classify a new case. Decision rules can be in a simple form, like:

$$\textbf{IF} \quad condition_j \quad \textbf{THEN} \quad \text{CLASS} = class_i \ .$$

Example of a simple rule is: *IF body temperature is higher than 38 degrees THEN patient is ill*. More complex rules can include conjunction (logical AND) and/or disjunction (logical OR), like:

$$\textbf{IF} \quad condition_a \quad \textbf{AND} \quad condition_b \quad \textbf{OR} \quad condition_C \quad \textbf{THEN} \quad \text{CLASS} = class_i \ .$$

An example of a more complex rule is: *IF body temperature is higher than 37.5 degrees AND diarrhea is true THEN patient is ill*. More complex dependent rules can form a decision tree.

Rules can also include ELSE statement, like:

$$\textbf{IF} \quad condition_j \quad \textbf{THEN} \quad \text{CLASS} = class_i \quad \textbf{ELSE} \quad \text{CLASS} = class_k .$$

An example of IF-THAN-ELSE decision rule is: *IF body temperature is higher than 37.5 degrees AND diarrhea is true THEN patient is ill* ELSE *patient is healthy*.

Disadvantages of rule induction that have to be considered are:

(1) A set of induced rules does not necessarily cover all possible situations (it is possible that the system will not be able to classify a new case),
(2) Two or more induced rules my conflict in their predictions. A usual solution to the second disadvantage is to assign a confidence to each rule and for classification of a new case use the most confident one or alternatively to use weighted voting of rules according to their confidence.

An important advantage of decision rule induction is certainly their simplicity and understandability.

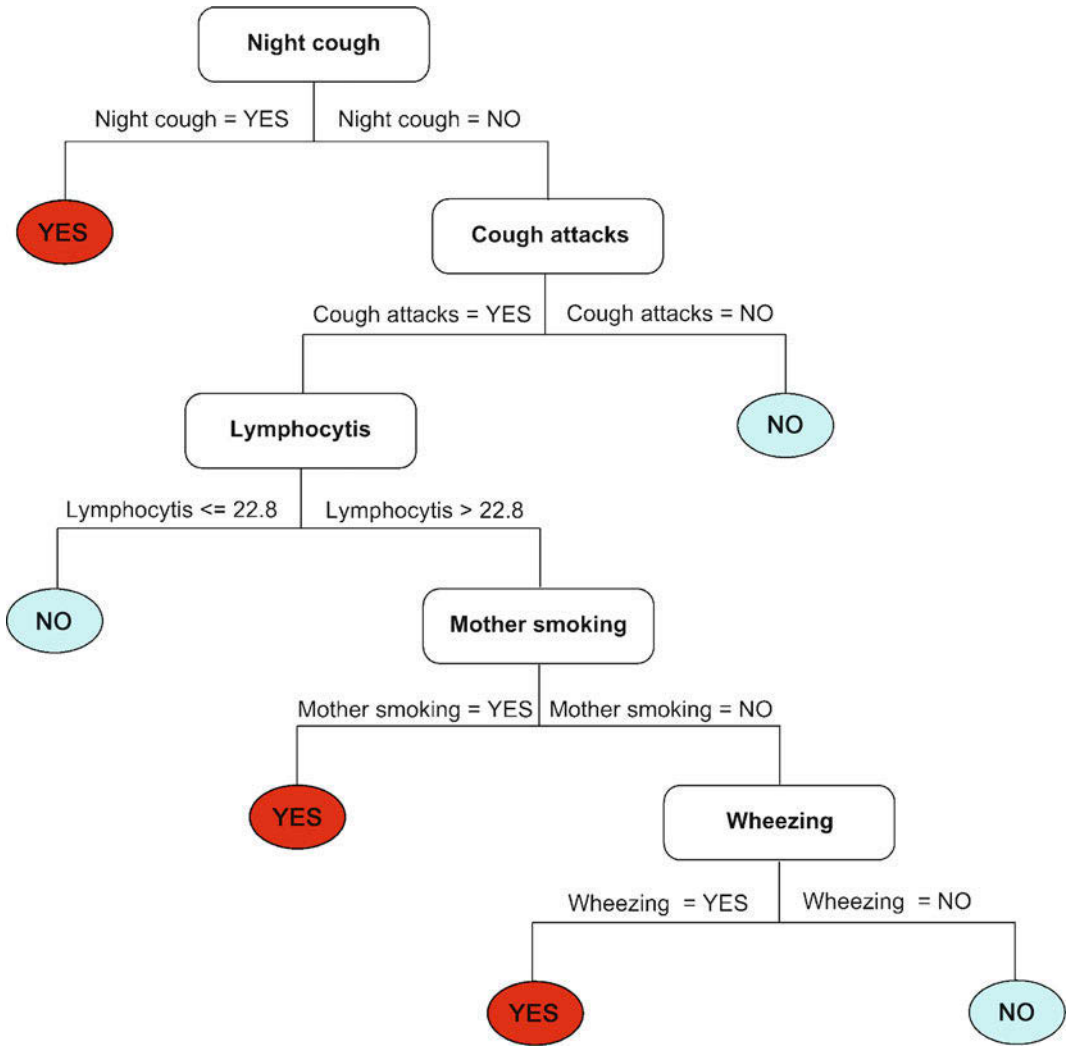
### Decision Trees

Decision trees are a way of representing a series of rules that lead to classification of known classes. For example, we might wish to diagnose young children with asthma on the basis of gathered data about family history, blood examinations and other indicators of asthma in adults. Figure 4 shows a simple decision tree that solves this problem while illustrating basic components of a decision tree: internal nodes, branches and external nodes (also called leaves).

Decision trees are induced on the basis of iterative splitting of data from the training set into discrete groups, where the goal is to minimize the 'distance' among groups at each split. Therefore one of the distinctions among decision tree algorithms is the splitting criteria. The most commonly used splitting criteria are: information gain and information gain ratio used by Quinlan [38] in ID3, C4.5 and C5.0 algorithms; chi square goodness of fit [46] used in CHAID (SPSS answer tree) [29]; gini index used by Breiman in CART algorithm [7]; j-measure or cross-entropy introduced by Smyth and Goodman [45], etc.

Decision trees models are extensively used in data mining for prediction purposes such as decision support and knowledge extraction in various fields. Decision trees provide very important advantage compared to many of other data mining models - the possibility of explaining the decisions in a way understandable by humans. A main disadvantage of classic decision tree induction is sensitivity to noise (missing or corrupted data).

Decision trees are able to process both numerical and categorical variables. However, on the basis of the type of prediction variable we distinguish between **classification trees** used to predict categorical variables, and **regression trees** for continuous prediction variable.

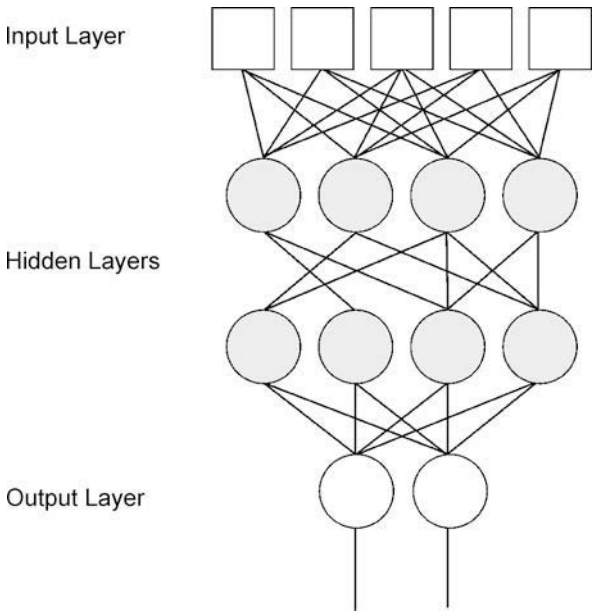



Discovery Systems, Figure 4
**A simple decision tree**

## Neural Networks

A neural network is a powerful model modeled after the human brain. As the human brain consist of millions of neurons interconnected by synapses, neural networks are formed from large number of artificial neurons (nodes) connected to each other to form a network. It can be used to capture and represent complex relationships between prediction variables (input) and target variables (output) or to find patterns in data. Neural networks may be applied to classification problems (where the output is categorical variable) as well to regression (where the output variable is continuous).

A neural network (Fig. 5) starts with an **input layer**, where each node (neuron) corresponds to a predictor variable. Each input node is connected to every node in a **hidden layer**. The nodes in a hidden layer can be connected to nodes in another hidden layer or to an **output layer**. The output layer consists of one or more response variables. Each connection has a **weight** assigned which reflects strength of a connection between the connected neurons. Like in human brain the strength of a connection can change in response to a presented stimulation or an obtained output, which enables the network to learn. After the input layer, each node (in the first hidden layer) gets a set of inputs which are then multiplied with connection weights and summed together to form a linear combination of values of nodes from the previous layer. An activation function is applied to the linear combination of values and the output is passed to the node(s) in the next layer.




Discovery Systems, Figure 5
**A neural network with two hidden layers**

In the beginning of model building (learning a network) the connection weights are unknown parameters which have to be learned on the basis of training data using one of the training algorithms. One of the most known training algorithms is the backpropagation algorithm [42,51], however newer methods include genetic algorithms [41], quasi-Newton [15], conjugate gradient [49], and many others.

The mayor advantages of neural networks are: good results on large databases, low sensitivity to noise and data distribution (on large databases or when proper mechanisms for preventing overfitting are used), applicability to multivariate non-linear problems, easy to implement to run on parallel computers, applicability to several problem domains.

Neural networks also have some disadvantages that have to be mentioned. The complexity of neural network model makes it hard to interpret the results. They belong to a group of so called black box models which means that the relationships among prediction and target variables cannot be represented in humanly understandable (symbolic) form. Therefore they are often not applicable to problems that demand an explanation of induced models.

Second disadvantage of neural networks is their proneness to overfitting due to large number of adjustable parameters which can adopt to every data if the size of a neural network is large enough. In order to reduce overfitting some mechanisms such as weight decay and/or cross validation have to be used.

The next disadvantage is an extensive amount of computational time needed for training a network (especially on large databases). However, as stated before they can be run on massively parallel computers with each node simultaneously doing its own calculations.

Neural networks have been successfully applied to a broad spectrum of data mining applications, such as: character recognition for optical character recognition (OCR) applications, target recognition used in military applications, machine diagnosis, medical diagnosis for image data (MRIs, X-rays), voice recognition, intelligent searching for internet search engines, fraud detection, quality control, etc.

### Ensemble Approaches

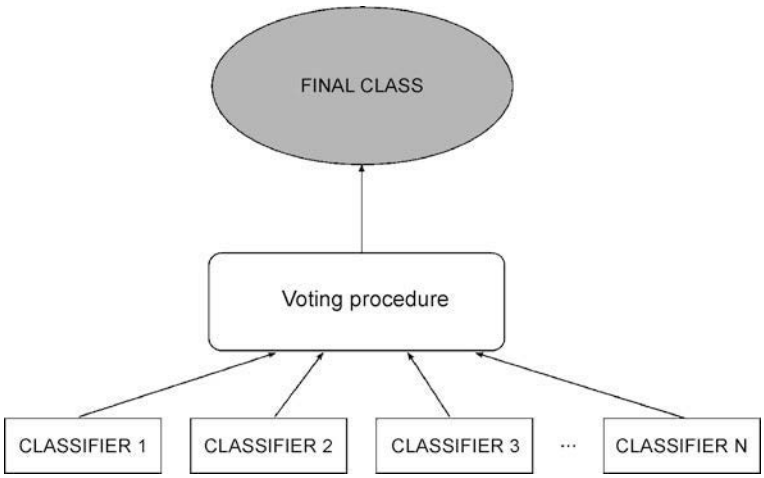
In recent years there has been a growing interest in the area of combining classifiers into ensembles also known as **committees** or **multiple classifier systems** (Fig. 6). The intuitive concept of ensemble approaches is that no single classifier can claim to be uniformly superior to any other, and that the integration of several single approaches will enhance the performance of final classification [53]. Hence, using the classification capabilities of multiple classifiers, where each classifier may make different and perhaps complementary errors, tend to yield an improved performance over single classifiers. Some ensemble approaches are actively trying to perturb some aspects of the training set, such as training samples, attributes or classes, in order to ensure classifier diversity. One of the most popular perturbation approaches are bootstrap aggregation (bagging) and boosting.

Bagging was first introduced by Breimen [6] in 1996 manipulates the training samples and forms replicate training sets. The final classification is based on a majority vote. Boosting, introduced by Freund and Schapire in 1996 [24] combines classifiers with weighted voting and is more complex since the distribution of training samples in training set is adaptively changed according to the performance of sequentially constructed classifiers.

In general ensemble approaches can be divided into three groups:

(1) Ensemble approaches that combine different independent classifiers (such as: Bayesian voting, majority voting, etc.),

(2) Ensemble learning approaches which construct a set of classifiers on the basis of one base classifier with perturbation of a training set (such as: bagging, boosting, windowing, etc.) and




Discovery Systems, Figure 6
**Combining classifiers into an ensemble**

(3) A combination of (1) and (2).

A detailed empirical study is presented in [19].

### Hybrid Approaches

The hybrid approaches rest on the assumption that only in the synergetic combination of single models can unleash their full power [28]. Each of the single model types has its advantages, but also inherent limitations and disadvantages, which must be taken into account when using the particular model type. Therefore the logical step is to combine different models (classic approaches) to overcome the disadvantages and limitations of a model.

According to the way of combining different models into a hybrid, four basic types of hybrids can be differentiated:

**Sequential hybrid** the inputs in the hybrid are fed as inputs in model 1. The outputs from model 1 represent inputs for model 2 and the outputs from model 2 are also outputs of the sequential hybrid (Fig. 7a). Both models in the hybrid are autonomous and can be also used independently.

**External hybrid** model 1 represents the main method which forwards the data to model 2. Model 2 processes the data and sends the results back to model 1 (Fig. 7b). Model 2 totally depends on model 1 and its outputs cannot be used as hybrid outputs.

**Embedded hybrid** has the most powerful connection between both models which are so strong interlaced, that none of them can function independently (Fig. 7c).

**Parallel hybrid** models in the parallel hybrid are totally independent. They both operate on the same inputs, but produce separate outputs. The outputs of the hybrid are determined within the arbitrary mechanism, embedded in the hybrid (Fig. 7d).

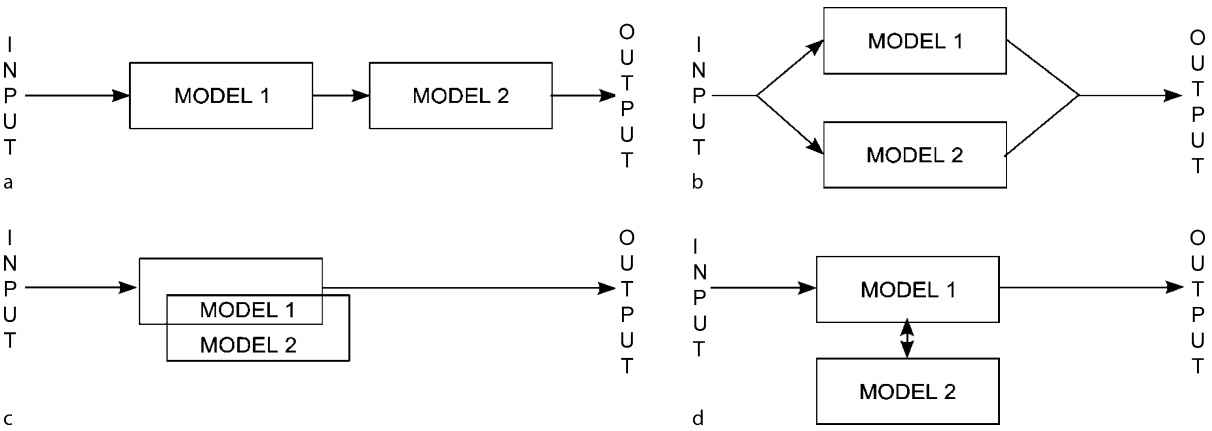
These four types of hybrids differ in type and strength of connection between the embedded methods. Such hybrids can also be building blocks for making more complex hybrids.

For example, in 1999 Boz [8] presented a method for converting a trained backpropagation neural network into a decision tree. In this approach the neural network was used to determine the list of attributes that influence the outcomes of the network the most. That list of attributes was then used in the process of the decision tree induction instead of the splitting criteria (Fig. 8).

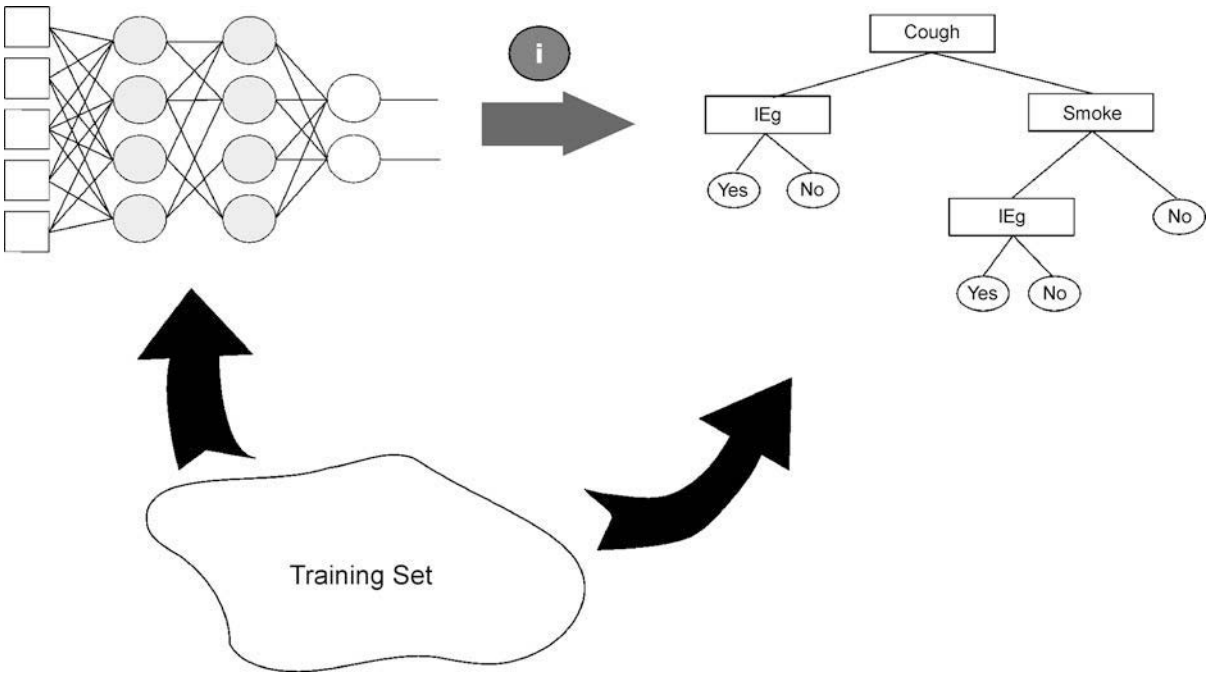
### Interpretation and Evaluation of Results

Evaluation of an induced model is a very important step of KDDM process. Testing of an induced model is crucial for evaluation. For that purpose, an initial data set is usually randomly split into two data sets: a training set (70% of randomly chosen samples) and a test set (30% of samples). When a training set is large it can be divided into two sets: a training set and a validation set. A validation set is used to assess how well the model fits the data, to adjust the model, or to select the best model among those that have been validated.

Beside random selection of training and testing samples from the database, the distribution of classes in training set is also very important for effective learning. In most cases it is best to preserve the natural distribution of classes (as it appears in initial data set) in the training set. However, in real world problems (especially from the medical




Discovery Systems, Figure 7
**Basic types of hybrids: a sequential hybrid, b external hybrid, c embedded hybrid and d parallel hybrid**




Discovery Systems, Figure 8
**External hybrid of a decision tree and neural network**

field) we often come across the data sets with very unequal distribution of classes (for example: less than 10% of samples belong to a specific class). In cases like that the use of natural distribution is not appropriate simply because there are not enough training samples to effectively learn a model of basic concepts. Equalization of class distribution is therefore a better solution [50].

The quality of the induced model is most commonly evaluated with the **accuracy rate**. The accuracy rate is calculated separately on the training set and the test set. Training accuracy indicates the extent of concepts learned, whereas the accuracy of correctly classified samples from test set provides us more information about the predictive abilities of a model and its applicability to practice. Accuracy is calculated as follows:

$$\text{accuracy} = \frac{\text{num of correctly classified samples}}{\text{num of all samples}} .$$

Unfortunately, the accuracy by itself is not necessarily the right metric for selecting the best model. As mentioned before, the distribution of target variable classes in real world data sets is often very biased. In those cases we can obtain a model with high accuracy; however the accuracy of classifying the samples of non-dominant class can be catastrophically low. Therefore use of other metrics that expose more information of the types of errors and the costs associated with them are a necessity.

**Average class accuracy** is one of the simple metrics that gives more information about model's quality. The classification accuracy is calculated separately for each class of a target variable and the average across all classes is calculated:

$$\text{average class accuracy} = \frac{\sum_c \text{accuracy}_c}{\text{num of classes}},$$

where $c$ is one of the possible classes and $\text{accuracy}_c$ is:

$$\text{accuracy}_c = \frac{\text{num of correctly classified objects in class } c}{\text{num of all objects in class } c}.$$

When accuracy is equal or similar to average class accuracy, one can speculate that the model does not have problems with classification of samples belonging to a specific class. Knowing the accuracies for each class of target variable gives us additional information about problems in the database or model.

For classification problems a **confusion matrix** [37] is a very useful tool for understanding the results of testing a model. A confusion matrix shows the results of actual versus predicted class values. Table 2 presents a sample of a confusion matrix where the columns show actual classes and the rows show predicted classes. Therefore the diagonal shows all correctly classified samples (98 samples out of 125). Class accuracies can also be calculated. For example, 30 samples out of 45 belonging to Class 1 were correctly classified (class accuracy for Class 1 is 67%); 10 samples were misclassified as Class 2 and 5 samples were misclassified as Class 3).

The confusion matrix for binary classification can be interpreted with two commonly used matrices: **specificity** and **sensitivity**. They are most often used to evaluate a performance of medical test. Sensitivity is the proportion of samples that were classified as positive of all the positive samples tested. In the case of classifying people with illness it can be seen as *the probability that the test is positive given that the patient is sick*. Consequently, the higher the sensitivity, the fewer sick people are undetected.

Specificity is the proportion of samples that were classified as negative of all the negative samples tested. In the case of classifying people with illness it can be seen as *the probability that the test is negative given that the patient is not sick*. Consequently, the higher the specificity, the fewer healthy people are classified as sick.

From the definitions above we can conclude the following: *Sensitivity and specificity are inversely related (higher sensitivity leads to lower specificity)*.

Sensitivity and specificity are calculated as follows:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{Specificity} = \frac{\text{TN}}{\text{FP} + \text{TN}}$$

where the notations are explained in Table 3.

The relationship between specificity and sensitivity can be visualized with the **ROC graph** [5].

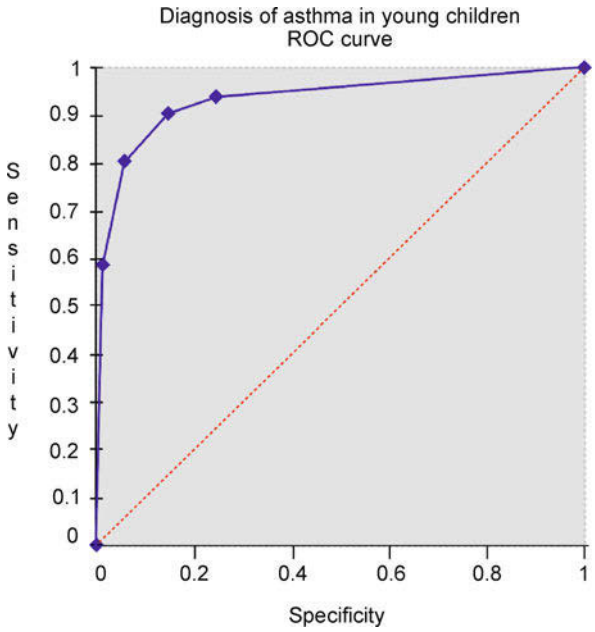
ROC (receiver operating characteristics) graph [5] is a two dimensional plot with the false positive rate (1-specificity) on the $x$-axis and the true positive rate (sensitivity) on $y$-axis (Fig. 9). The best possible prediction model would yield a point (0,1) in the upper left corner of ROC graph, which represents 100% sensitivity (all true positive samples correctly classified) and 100% specificity (no false positive samples classified). On the contrary, the point (1,0) in the lower right corner of ROC graph represents the worst possible prediction model with 0% sensitivity and 0% specificity. A model with completely random classification would give a point along diagonal line from the left bottom to the top right corner ($y = x$). The diagonal line divides the ROC space into two areas: points above diagonal line represent good models and points below diagonal line represent bad models. A more precise way of characterizing this is simply to calculate the **area under the ROC curve** [3]. The closer the area is to 0.5, the worse is the model, and the closer it is to 1.0, the better is the model.

Discovery Systems, Table 2
**A sample confusion matrix for three classes**

| *Predicted* | *Actual* | | |
|---|---|---|---|
| | Class 1 | Class 2 | Class 3 |
| **Class 1** | 30 | 2 | 0 |
| **Class 2** | 10 | 25 | 3 |
| **Class 3** | 5 | 7 | 43 |

Discovery Systems, Table 3
**A binary confusion matrix**

| *Predicted* | *Actual* | |
|---|---|---|
| | Positive | Negative |
| **Positive** | **TP** (*True Positive*) | **FP** (*False Positive*) |
| **Negative** | **FN** (*False Negative*) | **TN** (*True Negative*) |




Discovery Systems, Figure 9
**ROC graph**

All presented evaluation techniques express the quality of induced model in dependence of data gathered in the data set. However, no matter how well the model performs on data sets used for testing, there is no 100% guarantee that the model will perform well in the real world. One of the reasons can be found in the quality of data used to build the model (a lot of noise, missing data, implicit assumptions about choosing prediction variables, etc.). When the training data fails to match the real world, the model can learn incorrect concepts and will therefore perform very poor in the practice. For that purpose it is very important to test the model in the real world. For example: if the model is used for diagnosing asthma in young children, it should be tested on a small set of symptomatic children.

An evaluation of a model by a **domain expert** (usually also an end-user) is also very important before presenting the model to the real world. In that case symbolic models (such as: decision rules, decision trees, etc.) have essential advantage over the so called black-box models (such as: neural networks, support vector machines, etc.) – the possibility to interpret the relationships among prediction and target variables.

Data mining extracts already known and expected information from the database, but on the other hand, useful relationships between variables that are non-intuitive are the jewels that data mining hopes to find. In order to enable a domain expert to evaluate the model and possibly to extract some new knowledge, it is very important to choose a proper visualization of a model. This requires understanding end-user's needs. Visualizing a model allows a user to discuss and explain the logic behind the model with colleagues, customers, and other users. It also builds the user's trust in the results. Symbolic models are relatively easy to visualize, however black-box models can pose more problems but novel solutions are starting to appear.

## Utilization of Results

Once a data mining model is built and evaluated it can be used in one of the following ways: (1) analyzing a model and extracting knowledge or (2) applying a model in practice for a decision support.

In (1) the analyst (usually a domain expert) studies the model and the results in order to extract some new knowledge about the relationships among variables (for example: comparison of different clusters, interpretation of induced rules, etc).

In (2) the model is applied into practice and used on new data in order to perform some classification or regression tasks. Prediction models can be used for decision support independently. For instance, a model for diagnosing asthma in young children can support the physician in making a diagnosis. However, usually models are used in some business process and therefore they are most commonly incorporated into business applications. For instance, a model for predicting a risky loan applicant can be integrated into the loan application.

Some prediction models turn out to be very time and recourse consuming in practical use (for example: monitoring credit card transactions, fraud detections, etc). In those cases parallel systems can provide an effective solution.

Since all practical knowledge and experiences of domain experts are often not captured in prediction models built in data mining process, they may be used in combination with a model and as such applied into practice. Usually a combination of expert knowledge and knowledge discovered in data mining process can be very effective.

Since we are all aware that over time all systems evolve (all things are changing – nothing is static) it is very important to continuously monitor the performance of a model used in practice. When a significant reduction in performance of a model is detected, the model has to be retrained or sometimes even completely rebuilt.

## Knowledge Discovery Frameworks and Tools

As already stated, knowledge discovery is a process of searching through large amounts of data and picking out

relevant information. It is very often used in science but has become a very important process in the business and other areas of everyday life, too. Due to a huge demand for knowledge discovery and data mining tools, there are a lot of different tools and frameworks that are becoming available over the Internet. Most of them are coming from academic circles and can be obtained on open source or general public license basis. In this chapter we mostly focus on knowledge discovery and data mining frameworks and tools that are freely available and are used by a wide range of users from different fields.

The following tools that cover the biggest share of KD tools users will be presented:

- Weka/Pentaho
- YALE/Rapid-i
- Orange
- Tanagra.

### Weka/Pentaho

Originally created in 1993, the Weka project [52] was established by the University of Waikato as a platform for research and testing of advanced machine learning techniques. Since then, Weka has developed a large and loyal following in both academic and industry circles. It is so popular because of easy tailoring and extending of powerful analytical techniques with advanced visualization and industry-specific algorithms. In 2006 it was acquired by Pentaho [20], which is an even larger suite of tools that provides a full spectrum of open source business intelligence (BI) capabilities including reporting, analysis, dashboards, data mining, data integration, and a BI platform that have made it the world's most popular open source BI suite. Pentaho is also the primary sponsor and owner of popular open source projects including Mondrian, JFreeReport, Kettle, and Weka. By this acquisition, Weka got even wider range of users and established itself in the BI knowledge discovery circles.

### YALE/RapidMiner

Similar to Weka, YALE [33] was also transformed into a more commercial, but still freely available set of tools for machine learning based tasks that can be used in knowledge discovery process. Both have their roots in academic circles and are based on most recent research papers. YALE was renamed into RapidMiner after the acquisition of YALE by Rapid-I in May 2007.

YALE/RapidMiner is a rapid prototyping system for knowledge discovery and data mining. With more than 400 data mining operators it is one of the most comprehensive open-source data mining tools. It is widely used by a large number of organizations covering a wide range of different branches.

As the main focus of the YALE authors has been flexibility, it is reasonable to expect most advantage for YALE in this area. But flexibility in creating experiments, reusing previous work and preprocessing data to work on, were main advantages only in the early versions of YALE compared to Weka. However, visualization of results is still better supported in YALE.

### Orange

Orange [16] is component-based data mining software. It includes a range of preprocessing, modeling and data exploration techniques. It is based on C++ components, that are accessed either directly, through Python scripts, or through GUI objects called Orange Widgets.

Orange core objects and Python modules support various data mining tasks that span from data preprocessing to modeling and evaluation. Among other it supports techniques for:

- Data input, providing the support for various popular data formats,
- Data manipulation and preprocessing, like sampling, filtering, scaling, discretization, construction of new attributes, and alike,
- Methods for development of classification models, including classification trees, naïve Bayesian classifier, instance-based approaches, logistic regression and support vector machines,
- Regression methods, including linear regression, regression trees, and instance-based approaches,
- Various wrappers, like those for calibration of probability predictions of classification models,
- Ensemble approaches, like boosting and bagging,
- Various state-of-the-art constructive induction methods, including function decomposition,
- Association rules and data clustering methods,
- Evaluation methods, different hold-out schemes and range of scoring methods for prediction models including classification accuracy, AUC, Brier score, and alike. Various hypothesis testing approaches are also supported,
- Methods to export predictive models to PMML.

Orange is based on C++, which could be the main advantage when fast knowledge discovery tools are needed in comparison to before mentioned Java based frameworks.

### Tanagra

Tanagra [40] is a free data mining and knowledge discovery software for academic and research purposes. It

proposes several data mining methods from exploratory data analysis, statistical learning, machine learning and databases area. In comparison to other presented tools, Tanagra contains more statistical functions than other frameworks, but is aimed mainly to academic circles. The main purpose of Tanagra project is to give researchers and students an easy-to-use data mining software, conforming to the present standards of the software development in this domain (especially in the design of its graphical user interface and the way in which it is used), and allowing one to analyze either real or synthetic data.

Another approach to data analysis is R [13], which is heavily used for statistical analysis of data and also features a number of machine learning algorithms. Being very efficient with handling data, it lacks a graphical user interface for building projects and uses a rather nonintuitive syntax one has to get used to. Using the extensions that are constantly becoming available one can expect that R will feature most of the knowledge discovery functionality in the future.

## Conclusions and Future Directions

The aggressive rate of data that is stored on modern computer systems has far outpaced our ability to process and analyze this data. This way data is often deposited to merely rest in peace, never to be accessed again. Novel applications of knowledge discovery systems in supporting scientific discoveries, business exploitation or complex decision making, have awaken the growing commercial interest in knowledge discovery and data mining techniques. That has stimulated new interest in automatic knowledge extraction from examples stored in large databases.

Knowledge discovery can be broadly defined as automated discovery of novel and useful information from databases. Unfortunately there is no method or formal model that would be able to define the optimal knowledge discovery solution that should be applied for a certain problem. This problem will become even more important in the future as a growing amount of collected data will not allow time costs that are caused by empirical evaluation of all possible knowledge discovery techniques to find the most appropriate one. The future directions and possible solutions for this problem were defined in [34] where different meta-learning techniques along with a knowledge-driven framework for a KD system that contains a limited number of data mining techniques are presented.

Another interesting direction for future discoveries is application of KD techniques in the field of structured knowledge or textual documents that can be transformed into formally structured form. This field of research is heading toward advanced techniques from recent text mining research like sentiment analysis and tries to combine them with classical data mining techniques. This will only be possible when we will be able to transform unstructured text into machine readable form that would formalize parts of text and extract the underlying messages in the text.

Despite the rapid growth, KD systems are still evolving and will represent an important part of the future intelligent systems in various fields. It is obvious that KD systems are also becoming an important part of the large business systems that are fortunately available through different forms of open source initiatives that can help in their sustainable development for the future.

## Bibliography

### Primary Literature


1. Anand S, Buchner A (1998) Decision support using data mining. Financial Time Management, London
2. Baeck T (1996) Evolutionary algorithms in theory and practice. Oxford University Press, New York
3. Barley AP (1997) The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognit 30(7):1145–1159
4. Becerra-Fernandez I, Gonzalez A, Sabherwal R (2004) Knowledge management: Challenges, solutions, and technologies. Prentice Hall, Upper Saddle River
5. Beck JR, Shultz E (1986) The use of relative operating characteristic (ROC) curves in test performance evaluation. Arch Pathol Lab Med 110:13–20
6. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140
7. Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and regression trees. Wadsworth International Group, Belmont
8. Boz O (2000) Converting a trained neural network to a decision tree dectext – decision tree extractor. Ph D thesis, Computer Science and Engineering, Lehigh University. http://citeseer.ist.psu.edu/boz00converting.html. Accessed 12 Nov 2007
9. Cabena P, Hadjinian P, Stadler R, Verhees J, Zanasi A (1998) Discovering data mining: From concepts to implementation. Prentice Hall, Upper Saddle River
10. Caspase Drug Discovery Systems. drug discovery system. http://www.biomol.com/Online_Catalog/Online_Catalog/Products/36/?categoryId=420. Accessed 6 Nov 2007
11. Cios K, Teresinska A, Konieczna S, Potocka J, Sharma S (2000) Diagnosing myocardial perfusion from PECT bull's-eye maps – a knowledge discovery approach. IEEE Eng Med Biol Mag, Special Issue Med Data Mining Knowl Discov 19(4):17–25
12. Cios KJ, Pedrycz W, Swiniarski RW, Kurgan LA (2007) Data mining. A knowledge discovery approach. Springer, New York
13. Dalgaard P (2002) Introductory statistics with R. Springer, New York
14. Davenport TH, Prusak L (1997) Information ecology: Mastering the information and knowledge environment. Oxford University Press, New York

15. Dennis JE Jr, Schnabel RB (1989) A view of unconstrained optimization. In: Nemhauser GL, Runnooy Kan AHG, Todd MJ (eds) Handbook in operations research and management science, vol 1 Optimization. Elsevier, Amsterdam
16. Demsar J, Zupan B (2004) Orange: From experimental machine learning to interactive data mining. White Paper. Faculty of Computer and Information Science, University of Ljubljana. http://www.ailab.si/orange
17. Developmental Discovery System (TM). Developmental discovery system. http://www.gotofocus.com/. Accessed 6 Nov 2007
18. Dictionary.com Unabridged (v 1.1). discover. http://dictionary.reference.com/browse/discover. Accessed 5 Nov 2007
19. Dietterich TG (2000) Ensemble methods in machine learning. In: First International Workshop on Multiple Classifier Systems. Lecture Notes in Computer Science. Springer, New York, pp 1–15
20. Dixon J (2005) Pentaho Open Source Business Intelligence Platform Technical White Paper. Pentaho Corporation, Orlando. http://sourceforge.net/project/showfiles.php?group_id=140317
21. Fayyad U, Piatetsky-Shapiro G, Smyth P (1996) From data mining to knowledge discovery in databases (a survey). AI Mag 17(3):37–54
22. Fayyad U, Piatesky-Shapiro G, Smyth P, Uthurusamy R (eds) (1996) Advances in knowledge discovery and data mining. AAAI Press, Menlo Park
23. Frawley W, Piatesky-Shapiro G, Matheus C (1991) Knowledge discovery in databases: An overview. In: Piatesky-Shapiro G, Frowley W (eds) Knowledge Discovery in Databases. AAAI/MIT Press, pp 1–27, Menlo Park
24. Freund Y, Schapire RE (1996) Experiments with a new boosting algorithm. In: Proceedings Thirteenth International Conference on Machine Learning. Morgan Kaufman, San Francisco, pp 148–156
25. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison, Reading
26. Hand D, Mannila H, Smyth P (eds) (2001) Principles of data mining. MIT Press, Cambridge
27. Holland JH (1975) Adaptation in natural and artificial systems. MIT Press, Cambridge
28. Iglesias CJ (1996) The role of hybrid systems in intelligent data management: The case of fuzzy/neural hybrids. Control Eng Pract 4(6):839–845
29. Kass GV (1980) An exploratory technique for investigating large quantities of categorical data. Appl Stat 29:119–127
30. Kurgan L, Musilek P (2006) A survey of Knowledge Discovery and Data Mining process models. Knowl Eng Rev 21(1):1–24
31. Loh W, Shih Y (1997) Split selection methods for classification trees. Stat Sinica 7:815–840
32. Mannila H (2000) Theoretical frameworks of data mining. SIGKDD Explor 1:30–32
33. Mierswa I, Wurst M, Klinkenberg R, Scholz M, Euler T (2006) YALE: Rapid Prototyping for Complex Data Mining Tasks. In: Proc of the 12th ACMSIGKDD. International Conference on Knowledge Discovery and Data Mining, Philadelphia, pp 1–6
34. Pechenizkiy M, Tsymbal A, Puuronen S (2005) Meta-knowledge management in multistrategy process-oriented knowledge discovery systems. Technical Report, Dublin, Trinity College Dublin, Department of Computer Science, TCD-CS-2005–30, p 12
35. Piatetsky-Shapiro G (1991) Knowledge discovery in real databases: A report on the IJCAI-89 Workshop. AI Mag 11(5): 68–70
36. Piatetsky-Shapiro G (1999) The data mining industry coming to age. IEEE Intel Syst 14(6):32–33
37. Provost F, Fawcett T, Kohavi R (1998) The case against accuracy estimation for comparing classifiers. In: Proceedings of the Fifteenth International Conference on Machine Learning, (ICML-98), San Francisco
38. Quinlan JR (1986) Induction of decision trees. In: Machine Learning, vol 1. Kluwer, Hingham
39. Quinlan R (1993) C4.5: Programs for machine learning. Morgan Kaufmann, San Francisco
40. Rakotomalala R (2005) TANAGRA: Un logiciel gratuit pour l'enseignement et la recherche. In: Proc of the 5th Journees d'Extraction et Gestion des Connaissances 2:697–702
41. Reeves CR (ed) (1993) Modern heuristic techniques for combinatorial problems. Wiley, New York
42. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323:533–536
43. Sano M, Katoa Y, Taira K (2005) Functional gene-discovery systems based on libraries of hammerhead and hairpin ribozymes and short hairpin RNAs. Mol Biosyst 1:27–35
44. Shearer C (2000) The CRISP-DM model: the new blueprint for data mining. J Data Wareh I5(4):13–19
45. Smyth P, Goodman RM (1991) Rule induction using information theory. In: Piatetsky-Schapiro G, Frawley WJ (eds) Knowledge Discovery in Databases. AAAI Press, pp 159–176, Menlo Park
46. Snedecor GW, Cochran WG (1989) Statistical methods, 8th edn. Iowa State University Press, Ames
47. Tan P, Steinbach M, Kumar V (2005) Introduction to data mining. Addison, Boston
48. The Discovery System. discovery system for personality profiling. http://www.insights.com/core/English/TheDiscoverySystem/default.shtm. Accessed 6 Nov 2007
49. Towsey M, Alpsan D, Sztriha L (1995) Training a neural network with conjugate gradient methods. IEEE Proc Neural Netw 1:373–378
50. Weiss GM, Provost F (2001) The effect of class distribution on classifier learning. Technical Report ML-TR 43, Department of Computer Science, Rutgers University
51. Werbos PJ (1994) The roots of backpropagation. Wiley, New York
52. Witten IH, Frank E (2005) Data mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco
53. Wolpert D, Macready W (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

## Books and Reviews

Berthold M, Hand DJ (2003) Intelligent data analysis: An introduction, 2nd edn. Springer, New York

Lin TY, Ohsuga S, Liau CJ, Hu X, Tsumoto S (eds) (2005) Foundations of data mining and knowledge discovery. Studies in Computational Intelligence, vol 6. Springer, New York