**Data Mining**

**Lab 2: Data pre-processing**

---

Summary: Preparing the data for the data mining tasks:

1. Cleaning (e.g. deal with errors and missing values)
2. Transformation
    a. Conversion between types (e.g. discretization, binarization)
    b. Scaling
    c. Standardization
3. Reduction
    a. Feature selection
    b. Instance selection

## 1. Data cleaning

The data might contain erroneous or missing values caused by measuring devices malfunctioning, human errors, not provided info (e.g. privacy issues). In some cases these situations can be treated in an automatic way:

a) Values which are outside normal ranges (e.g. negative age, very large or very small body temperature etc). In such cases there are several approaches:
   - Remove the attributes which contain such values
   - Remove the instances which contain values which are not valid
   - Consider the value as a missing one

b) Missing values can be treated in different ways:
   - Consider the missing value as a specific case which has a particular value assigned (e.g. like "NA" in R datasets or like "?" in arff Weka files) – in the case of nominal attributes the value assigned to a missing field is interpreted as an usual value and it can be involved in the classification models (e.g. in the classification rules).
   - Remove the instances which contain missing values
   - Replace the missing fields with values estimated based on the values corresponding to other instances in the dataset (e.g. the average value of the corresponding attribute over the entire dataset or over the most similar instances – similarity is measured with respect to the other attributes). Such an approach is called imputation.

**Exercise 1:**

a) Load (in R or Rattle) the file "autos.arff" and remove all instances having values of the attribute 25 (price) larger than 10000.

**Hint:**

**R:** read an arff file:

```
> library(foreign)
> setwd(" … ") # set the working directory

> autos <- read.arff("datasets/autos.arff")
> autos10000 <- autos[autos$price<10000,]
```

b) Load the file with the Titanic dataset ("trainTitanic.csv"). Identify which attributes contain missing values and remove the instances which are characterized by missing values.

Hint:
**R/Rattle**:

```
> titanic <- read.csv("datasets/trainTitanic.csv")
> # remove instances with missing Age
> titanicAge <- titanic[!is.na(titanic$Age),]
```

or `Transform -> Cleanup` (in Rattle)

Question: how is specified the missing cabin number? How can be removed the instances with missing cabin number?

c) Impute the missing values for the attribute Age in the Titanic dataset using the mean/ median/ mode

Hint:
**Rattle:** `Transform->Impute->Mean/Median/Mode,` select the Age attribute and click on `Execute`

Question: Which variant is more appropriate? Does the imputation step has an impact on the performance of a classification model based on a decision tree? (check this by using Model->Tree and then Evaluate in Rattle).

## 2. Transformations

### a) Conversion between types

- Discretization: transformation of the attributes taking values in a continuous range (e.g. real) in attributes which take discrete values (e.g. integer, ordinal). The simplest approach is when the interval of attribute values [min, max] is divided in r subintervals (e.g [min, min+h), [min+h, min+2h), …[min+(r-1)h,max], where h=(max-min)/r) and each interval will be associated to a discrete value (e.g. 1, 2, 3,….r).
- Binarization: used to transform a nominal attribute in a set of binary or logical attributes or to transform a set of transaction items into binary attributes. For instance if an attribute A has the values {v1,v2,v3} then it will be replaced with 3 binary attributes A1, A2 and A3.

### b) Scaling

- useful when different attributes have values in highly different ranges (e.g. one attribute takes values in [-0.1, 0.2] and another one takes values in [10000, 20000]) and the prediction model is sensitive in differences in the attributes' scales (e.g. nearest neighbor based approaches).
- the simplest scaling method is the linear scaling which uses a linear function to transform all values v of an attribute A in [0,1]:

$s(v) = \frac{v - \min_A}{\max_A - \min_A}$ , where min and max correspond to the minimal and maximal values of the attribute

**c) Standardization**

- useful when we are interested in measuring how much are the data concentrated / spread around their average
- by standardization, a value v is transformed as follows:
- $st(v) = \frac{v - avg(A)}{stdev(A)}$, where avg(A) denotes the average of the values of attribute A and stdev(A) is the standard deviation of the values of attribute A

**Exercise 2:** Transform the attribute Age in the Titanic dataset (**after imputation**) by discretization (using 8 bins). Analyze the impact of the discretization on the performance of a classification based on decision trees.

Hint:
**Rattle:** `Transform -> Recode -> EqualWidth -> Number=8`

**Exercise 3:** Load (in Rattle or Weka) the file "car.arff".

a) Transform all nominal attributes through binarization and save the transformed data in file carBinary.arff

Hint:
**Rattle:** `Transform -> Recode -> Indicator variable`

b) Analyze the impact of binarization on the classification performance

**Rattle:** use `Model -> Tree` and then `Evaluate` for the initial dataset and for the transformed dataset

**Case study 1.** Predicting Algae Bloom (source: L.Torgo, Data Mining with R. Learning with Case Studies, 2011)

The dataset consists of 200 instances containing characteristics of water samples collected from different rivers. Each instance contains, besides the characteristics of the river – size and speed, mean values of concentrations of various chemical substances obtained based on measurements over a period of 3 months, during the same season of the year. Each instance contains 18 attributes. Three of these variables are nominal and describe the season of the year when the water samples to be aggregated were collected, as well as the size and speed of the river in question. The next eight remaining attributes are values of different chemical parameters measured in the water samples forming the aggregation, namely:

- Maximum pH value
- Minimum value of O2 (oxygen)
- Mean value of Cl (chloride)
- Mean value of NO3- (nitrates)
- Mean value of NH4+ (ammonium)
- Mean of PO34- (orthophosphate)
- Mean of total PO4 (phosphate)
- Mean of chlorophyll

The final 7 attributes are target values and correspond to frequency of different harmful algae found in the respective water samples. No information is given regarding the names of the algae that were identified.

The problem is to predict the presence of various types of algae based on the values of the concentrations of the chemical substances

**Step 1:** analyze of the data characteristics and decide if some pre-processing tasks should be applied

**Starting point:** CaseStudy_Algae.R

**Data preprocessing using Scikit-learn.** (sklearn.preprocessing package – details at https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing)

- scaling: MinMaxScaler (rmk: the parameters of an object from this class are adapted using the fit method and the transformation is applied to data using the transform method)

Ex:
```
import numpy as np
from sklearn import preprocessing
X = np.array([[ 1., -1.,  2.], [ 2.,  0.,  0.], [ 0.,  1., -1.]])
min_max_scaler = preprocessing.MinMaxScaler()
X_minmax = min_max_scaler.fit(X).transform(X)
#X_minmax = min_max_scaler.fit_transform(X)  # alta varianta
```

- standardization: scale

```
import numpy as np
from sklearn import preprocessing
X = np.array([[ 1., -1.,  2.], [ 2.,  0.,  0.], [ 0.,  1., -1.]])
X_standardizat = preprocessing.scale(X)
```

- normalization: normalize

Ex:
```
X = np.array([[ 1., -1.,  2.], [ 2.,  0.,  0.], [ 0.,  1., -1.]])
X_normalized = preprocessing.normalize(X, norm='l2')
```

- discretization: KBinsDiscretizer

```
Ex:
import numpy as np
from sklearn import preprocessing
X = np.array([[ -3., 5., 15 ],[  0., 6., 14 ],[  6., 3., 11 ]])
discret = preprocessing.KBinsDiscretizer(n_bins=[3, 2, 2],
encode='ordinal')
discret.fit(X)
discret.transform(X)
```

- binarization of categorical (nominal) attributes: OneHotEncoder

Ex:
```
import numpy as np
from sklearn import preprocessing
enc = preprocessing.OneHotEncoder()
```

```
X = [['male', 'from US', 'uses Safari'], ['female', 'from Europe',
'uses Firefox']]
enc.fit(X)
enc.transform([['female', 'from US', 'uses Safari'],
               ['male', 'from Europe', 'uses Safari']]).toarray()
```

- Missing values imputation: SimpleImputer

Ex. 1.

```
from sklearn.impute import SimpleImputer
imp = SimpleImputer(missing_values=np.nan, strategy='mean')
imp.fit([[1, 2], [np.nan, 3], [7, 6]])
X = [[np.nan, 2], [6, np.nan], [7, 6]]
print(imp.transform(X))
```

```
Ex.2.
import pandas as pd
df = pd.DataFrame([["a", "x"],
                   [np.nan, "y"],
                   ["a", np.nan],
                   ["b", "y"]], dtype="category")
imp = SimpleImputer(strategy="most_frequent")
print(imp.fit_transform(df))
```

Remark. Other imputation variants which use information from several attributes are **IterativeImputer** and **KNNImputer** (the missing values are estimated using the values corresponding to the k nearest neighbours)

```
Ex.
import numpy as np
from sklearn.impute import KNNImputer
nan = np.nan
X = [[1, 2, nan], [3, 4, 3], [nan, 6, 5], [8, 8, 7]]
imputer = KNNImputer(n_neighbors=2, weights="uniform")
imputer.fit_transform(X)
```

## 3. Data reduction

*Motivation:* some attributes or instances could be irrelevant or redundant. For instance an attribute having the same value over the entire dataset is not relevant and should be ignored. Two highly correlated attributes (e.g. A1=2*A2) are redundant, as it is enough to use only one of them.

- Reducing the number of attributes can reduce the computational cost but also improve the performance of a data mining model (e.g. classifier);
- Reducing the number of instances in the training set can reduce the computational cost of the training process or of the classification process (in the case of lazy classifiers) but also improve the quality of a classifier especially in the case of unbalanced data sets (the number of examples in a class is significantly larger than the number of data in another class);

Attribute reduction can be done through:

- Attribute selection:
  - Filter-based selection (the selection is done by analyzing only the properties of data and it is independent of a data mining task)
  - Wrapper-based selection (the selection is done such that the selected attributes are the most relevant ones in the context of a given data mining task)

- Data projection on a space of a smaller size (e.g. Principal Component Analysis)

Instance reduction can be done through:

- Instance removal (based on some specific rules)
- Instance sampling (using a random sampler with or without replacement)

### 3.1. Filter-based attribute selection

Attribute selection can be done in at least two ways:

- By searching the space of attribute subsets (in the case of n attributes the search space is of cardinality $2^n$-2 – we ignore the empty set and the full set)
- By ranking the attributes according to their relevance and take the most relevant ones (in the case of n ranked attributes A1,A2,….An, there are n-1 variants to analyze: {A1}, $\{A_1,A_2\}$, $\{A_1,A_2,A_3\}$….{ $A_1,A_2,\ldots,A_{n-1}$}

In each case it should be specified:

- A measure of relevance of an attribute or a subset of attributes
  - *Unsupervised case:* the class information is not used to evaluate the attribute relevance (the analysis is mainly based on the correlation or the similarities between attributes)
  - *Supervised case:* the class information is not used to evaluate the attribute relevance (the analysis is based on the correlation between attribute values and classes)
- A search technique:
  - Exhaustive search (all subsets of attributes are analyzed)
  - Greedy search (forward/backward): add/remove at each step the best/worst attribute
  - Random search
  - Rank-based search (the attributes are selected in the order specified by a ranking)

**Exercise 4.**

**R:** Implement a greedy strategy for selecting the attributes based on the Gini index (increasing order of values) and the Information Gain (decreasing order of values). Dataset: weatherNominal.csv

**Starting point:** see GiniIndex.r and InformationalGain.r

### 3.2. Data projection – Principal Component Analysis

Aim: transform the data (by changing the system of coordinates) such that the correlation between attributes is removed and then keep only the attributes corresponding to the coordinate axes which capture the most of data variability. This transformation can be done by using Principal Component Analysis which consists of the following steps:

- Compute the **covariance matrix** C of the data set (if there are n attributes then the matrix will have the size nxn); in practice before computing the covariance matrix the data are **centered** (the average of the dataset is subtracted from each data instance)
- Compute the **eigenvalues** and the **eigenvectors** of C
- Sort the eigenvalues decreasingly
- Select m<n eigenvectors corresponding to the largest eigenvalues; m is usually chosen such that the transformed data preserve most of the original data variability (e.g. 95%)
- Project the data on the space defined by the m eigenvectors by multiplying the data matrix D with the matrix consisting of the first m eigenvectors

Another approach is based of the **singular value decomposition** of the data matrix: $D=U\ S\ V^T$, where

- the data matrix D has N rows and n columns,
- U is a NxN orthogonal matrix (i.e. $U\ U^T=I$),
- S is a diagonal matrix with N rows and n columns containing the **singular values** (the singular values are related to the square roots of the eigenvalues)
- V is a nxn matrix which contains on its columns the eigenvectors of the matrix $D^TD$
- Project the data by multiplying the data matrix D (Nxn) with first m columns of V (nxm).

**Exercise 5.** Apply the PCA transformation on the `iris` dataset

Hint:

**R:** starting point for the implementation of the main steps: pca.r

**Rattle:** use `Explore -> Principal Components` (`eig` corresponds to the variant based on eigendecomposition, svd corresponds to the variant based on Singular Value Decomposition)

**Case Study 2.** Analysis of a synthetic dataset (distribution of the distances between data and computation of the Fisher score for numerical attribute selection)

Starting point: AttributeAnalysisSyntheticData.r

**Homework.**

1. Analyze the data pre-processing functions from the caret R package - see https://machinelearningmastery.com/pre-process-your-dataset-in-r/

2. Download the *Arrythmia* data set from the *UCI Machine Learning Repository*.
   a. Transform all records such that they have a mean of 0 and a standard deviation of 1.
   b. Discretize each numerical attribute into (i) 10 equi-width ranges and (ii) 10 equi-depth ranges.

3. Download the *Musk* data set from the *UCI Machine Learning Repository*. Apply *PCA* to the data set, and report the eigenvectors and eigenvalues.

4. Analyze the usage of PCA in the context of extracting features from face images (eigenfaces). Such an example (used in the context of face recognition) is available at https://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html#sphx-glr-download-auto-examples-applications-plot-face-recognition-py