Data Mining

Lab 2: Pre-procesarea datelor

Sumar: Pregătirea datelor pentru aplicarea tehnicilor de analiză:

1. Curățare (e.g. tratarea erorilor și a valorilor absente)
2. Transformare
   a. Conversii intre diferite tipuri de atribute (e.g. discretizare, binarizare)
   b. Scalare
   c. Standardizare
3. Reducerea dimensiunii
   a. Selecția atributelor
   b. Selectia instanțelor
   c. Proiecţia datelor (analiza componentelor principale)

1. Curățarea datelor

Datele pot conține valori eronate sau absente cauzate fie de disfuncționalități ale dispozitivelor de măsurare/înregistrare, erori umane, refuzul de a completa anumite informații (în cazul informațiilor colectate pe bază de chestionare). In unele situații erorile pot fi detectate și corectate în mod automat:

a) Valori care sunt in afara domeniilor de valori valide (de exemplu, vârstă negativă, valori ale temperaturii corporale, tensiunii arteriale etc in afara plajei de valori plauzibile). In astfel de situații există mai multe variante:

   - Eliminarea atributelor ce conțin astfel de valori
   - Eliminarea instanțelor ce conțin valori care nu sunt valide
   - Eliminarea valorii eronate (și interpretarea ei ca valoare absentă)

b) Valorile absente pot fi tratate in diferite moduri:

   - Se consideră valoarea absentă ca un caz specific – în acest caz simbolul utilizat pentru a marca o valoare absentă poate fi tratat ca o valoare distinctă și poate sa apară în modelele construite pe baza datelor (de exemplu în regulile de clasificare).
   - Eliminarea instanțelor ce conțin valori absente
   - Completarea valorilor absente cu valori estimate pe baza celor existente în setul de date (de exemplu cu media tuturor valorilor existente pentru atributul corespunzător sau cu media valorilor din instanțele similar celei care conține valoarea absent – similaritatea se măsoară folosind valorile asociate celorlalte attribute). Această abordare este cunoscută sub numele de tehnica imputării.

Exercițiul 1:

a) Deschideţi (în R, Rattle) fişierul "autos.arff" şi excludeţi toate instanţele pentru care valoarea atributului 25 (price) este mai mare decât 10000.

(Indicaţie:

R: citirea unui fişier de tip arff :

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

> autos <- read.arff("datasets/autos.arff")
```

```
> autos10000 <- autos[autos$price<10000,]
```

b) Deschideţi setul de date cu informaţii despre pasagerii de pe Titanic ("trainTitanic.csv"). Identificaţi care dintre instanţe au valori absente şi eliminaţi aceste instanţe.

Indicaţie:
R/Rattle:

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

sau `Transform -> Cleanup` (în Rattle)

Intrebare: cum sunt specificate valorile absente în cazul atributului corespunzător numărului cabinei? Cum pot fi eliminate instanţele pentru care nu e completat numărul cabinei?

c) Completaţi valorile absente ale atributului Age în setul de date Titanic utilizând media/ mediana/ moda

Indicaţie:
R/Rattle: `Transform->Impute->Mean/Median/Mode,` selectaţi atributul Age şi click pe `Execute`

Intrebare: Care variantă este mai potrivită? Influenţează completarea valorilor absente performanţa unui clasificator bazat pe arbori de decizie?

2. Transformări

a) Conversii între tipuri

- Discretizare: transformarea atributelor care iau valori într-un domeniu continuu (atribute de tip real) în atribute care iau valori într-o mulțime discretă (de tip întreg, nominal sau ordinal). Cea mai simplă abordare este de a diviza intervalul de valori [min, max] in r subintervale de aceeași lungime (de exemplu [min, min+h), [min+h, min+2h), …[min+(r-1)h,max], unde h=(max-min)/r) și fiecare interval va fi asociat cu o valoare discretă (de exemplu 1, 2, 3,….r).
- Binarizare: permite transformarea unui atribut nominal într-un set de atribute binare (sau logice) sau transformarea unui subset de itemi (corespunzători unei tranzacţii) într-un vector binar. De exemplu, dacă un atribut A poate lua valori din mulțimea {v1,v2,v3} atunci el va fi înlocuit cu 3 atribute binare A1, A2 și A3.

b) Scalare

- Este utilă când atribute diferite iau valori în domenii care diferă semnificativ (de exemplu un atribut ia valori in [-0.1, 0.2] și altul ia valori in [10000, 20000])
- Cea mai simplă variantă de scalare este cea liniară prin care un atribut A având valoarea v din $[min_A, max_A]$ este transformat într-un atribut care ia valori in [0,1]:

$s(v) = \frac{v - \min_A}{\max_A - \min_A}$, unde min și max corespund valorii minime respectiv celei maxime

c) Standardizare

- E utilă când interesează măsurarea abaterii valorilor față de medie în unități proporționale cu valoarea abaterii standard a datelor
- Prin standardizare, o valoare v este transformată după cum urmează:
- $st(v) = \frac{v - avg(A)}{stdev(A)}$, unde avg(A) reprezintă media valorilor atributului A iar stdev(A) este abaterea standard a valorilor atributului A

Exercițiu 2: Transformaţi atributul Age din setul Titanic (după completarea valorilor lipsă) prin discretizare (utilizând 8 sub-intervale). Analizaţi impactul discretizării asupra performanţei unui clasificator bazat pe arbori de decizie.

Indicaţie:
Rattle: `Transform -> Recode -> EqualWidth -> Number=8`

Exercițiu 3: Deschideţi (folosind Rattle) fişierul "car.arff".

a) Transformaţi toate atributele nominale prin binarizare şi salvaţi rezultatul în fişierul carBinary.arff

Indicaţie:
Rattle: `Transform -> Recode -> Indicator variable`

b) Analizaţi impactul clasificării asupra preformanţei unui clasificator

Rattle: utilizaţi `Model -> Tree` şi `Evaluate` pentru setul initial şi pentru cel transformat

Studiu de caz 1. Predicţia prezenţei unor tipuri de alge în apă (Algae Bloom - source: L.Torgo, Data Mining with R. Learning with Case Studies, 2011)

Setul de date constă din 200 de instanţe ce conţin caracteristici (în principal de natură chimică) corespunzătoare unor eşantioane de apă colectate din diferite râuri. Fiecare instanţă conţine, pe lângă caracteristicile râului (dimensiune şi viteză a apei) valori medii ale concentraţiei unor substanţe chimice bazate pe măsurători efectuate pe parcursul a 3 luni (separate pentru fiecare anotimp). Fiecare instanţă conţine 18 atribute. Trei dintre acestea sunt nominale şi specifică: anotimpul, dimensiunea râului şi viteza apei. Următoarele 8 atribute corespund concentraţiilor unor substanţe chimice:

- valoare maximă pH
- valoare minimă O2 (oxigen)
- valoare medie Cl (clor)
- valoare medie NO3- (nitraţi)
- valoare medie NH4+ (amoniac)
- valoare medie PO34- (ortofosfat)
- valoare medie PO4 (fosfat)
- valoare medie clorofilă

Ultimele 7 atribute sunt valorile ţintă şi corespund frecvenţei de apariţie ale unor alge în eşantioanele de apă analizate. Scopul urmărit este de a prezice prezenţa unor tipuri de alge în funcţie de caracteristicile râului şi concentraţiile de substanţe chimice.

Pas 1: analizaţi caracteristicile datelor şi decideţi ce tipuri de pre-procesări sunt utile

Punct de start: CaseStudy_Algae.R

Preprocesarea datelor folosind Scikit-learn. Pachet sklearn.preprocessing (detalii la https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing)

- scalare: MinMaxScaler

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
```

- standardizare: 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)
```

- normalizare: normalize

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

- discretizare: 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)
```

- binarizare atribute nominale: OneHotEncoder (sklearn.preprocessing.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()
```

- Tratarea valorilor absente: 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))
```

Obs. O varianta de imputare care ia in considerare mai multe attribute este implementata in **IterativeImputer** iar cea care se bazează pe principiul celui mai apropiat vecin este implementata in **KNNImputer**

```
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. Reducerea dimensiunii datelor

Motivație: anumite atribute sau instanţe pot fi irelevante sau redundante. De exemplu un atribut care are aceeaşi valoare pe întregul set de date ar trebui ignorat. Două atribute puternic corelate (e.g. A1=2*A2) sunt redundante și ar fi suficient să se rețină doar unul dintre ele.

- Reducerea numărului de atribute poate reduce costul de calcul dar poate să și conducă la o imbunătățire a performanței modelului construit pe baza datelor (e.g. clasificator)
- Reducerea numărului de instanțe din setul de antrenare poate reduce timpul necesar antrenării (sau clasificării, in cazul clasificatorilor "leneși") dar poate să și îmbunătățească performanțele modelului (în special în cazul seturilor de date nebalansate)

Reducerea numărului de atribute poate fi realizată prin:
- Selecția atributelor:
  - Selector de tip "filtru": selecția se bazează doar pe analiza proprietăților datelor fiind independentă de prelucrarea care va fi ulterior aplicată datelor.
  - Selector de tip "wrapper": selecția se realizează în contextul utilizării datelor intr-un proces specific de analiză (calitatea subsetului selectat este evaluată prin prisma performanței unui model de analiză extras din date)

- Proiecţia datelor pe un spațiu de dimensiune mai mică (e.g. Analiza componentelor principale - Principal Component Analysis)

Reducerea numărului de instanțe poate fi realizată prin:
- Eliminarea unor instanțe (pe baza unor reguli specifice)
- Selecția instanțelor (utilizând selectie aleatoare cu sau fără revenire)

3.1. Selector de tip filtru

Selectia atributelor poate fi realizată in două moduri:
- Căutând în spațiul submulțimilor de atribute (în cazul a n atribute spațiul de căutare are cardinalul 2^n-2 – se ignoră mulțimea vidă și întreg setul de atribute)
- Prin ierarhizarea atributelor în concordanță cu relevanța lor și selectarea celor mai relevante (în cazul a n atribute ierarhizate $A_1,A_2,\ldots A_n$, sunt n-1 variante de analizat: $\{A_1\}$, $\{A_1,A_2\}$, $\{A_1,A_2,A_3\}\ldots\{A_1,A_2,\ldots,A_{n-1}\}$

Pentru fiecare dintre variante trebuie specificate:
- O măsură a relevanței unui atribut sau subset de atribute
  - *Cazul nesupervizat:* informația privind clasa atributului nu este folosită pentru a evalua relevanța (analiza se bazează in principal pe corelațiile sau similaritățile dintre atribute)
  - *Cazul supervizat*: informația privind clasa este utilizată (analiza se bazează pe corelația dintre valorile atributelor si eticheta clasei)

- O tehnică de căutare:
  - Căutare exhaustivă (toate subseturile de atribute sunt analizate)
  - Căutare greedy (forward/backward): se adaugă/elimină cel mai bun/slab atribut identificat la momentul curent
  - Căutare aleatoare
  - Căutare bazată pe o ierarhizare (atributele sunt selectate în ordinea dictate de o ierarhie stabilită anterior)

Exercițiu 4.

R: Implementaţi o strategie greedy pentru selecţia incrementală a atributelor în ordinea crescătoare a indexului Gini respectiv în ordinea descrescătoare a câştigului informațional. Set de date: weatherNominal.csv

Punct de start: GiniIndex.r şi InformationalGain.r

3.2. Proiecția datelor – analiza componentelor principale

Scop: se transformă datele (schimbând sistemul de axe de coordonate) astfel încât să fie eliminată corelația dintre atribute și să se păstreze doar acele atribute care conservă cât mai mult din variabilitatea datelor. Această transformare poate fi realizată prin analiza în componente principale (Principal Component Analysis) parcurgând următoarele etape:

- Se calculează matricea de covarianță C a setului de date (daca datele au n atribute atunci matricea va avea dimesniunea nxn); in practică se obișnuiește să se centreze datele

inainte de a construi matricea (se scade din fiecare instanță media setului de date) – în felul acesta se ajunge să se calculeze matricea de corelație
- Se calculează valorile și vectorii proprii ai matricii C
- Se ordonează descrescător valorile proprii
- Se selecteaza m<n vectori proprii (cei corespunzători celor mai mari valori proprii); m se alege astfel incât datele transformate să conserve cât mai mult din variabilitatea datelor inițiale (e.g. 95%)
- Se proiectează datele pe spațiul definit de cei m vectori proprii

Exerciţiu 5. Aplicaţi analiza componentelor principale asupra setului de date `iris`

Indicaţie:

R: punct de start pentru implementarea principalilor paşi ai metodei: pca.r

Rattle: utilizaţi `Explore -> Principal Components` (`eig` corespunde variantei bazate pe descompunerea matricii de covarianţă pe baza vectorilor proprii).

Scikit learn: se poate folosi clasa `sklearn.decomposition.PCA`

Studiu de caz 2. Analiza transformărilor pentru un set de date sintetic. Analiza distribuţiei distanţelor dintre instanţe şi calculul scorului Fisher (vezi curs 2).

Punct de start: CaseStudy_SyntheticData.r

Temă.

1. Analizaţi funcţiile din pachetul R dprep (pt pre-procesare).

2. Descărcaţi setul de date *Arrythmia* de la *UCI Machine Learning Repository*.
   a. Transformaţi toate instanţele astfel încât să aibă media 0 şi abaterea standard 1.
   b. Discretizaţi fiecare atribut numeric în (i) 10 sub-intervale de aceeaşi dimensiune (ii) 10 sub-intervale ce conţin acelaşi număr de valori.

3. Descărcaţi setul de date *Musk* de la *UCI Machine Learning Repository*. Determinaţi vectorii şi valorile proprii asociate setului de date. Ce informaţii oferă?

4. Analizati utilizarea descompunerii dupa componente principale in contextual clasificarii imaginilor (eigenfaces)
   Exemplu implementare folosind Sciki-learn:
   https://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html#sphx-glr-download-auto-examples-applications-plot-face-recognition-py