

## Data Mining

### Lab 3: Clasificarea datelor

---

Sumar:

- Problema clasificării
- Clasificatori bazati pe instanțe (k- NearestNeighbor)
- Clasificatori bazati pe arbori de decizie (rpart)
- Clasificatori bazati pe reguli de decizie (OneR, JRip)
- Clasificatori probabiliști (Naïve Bayes)
- Clasificatori bazați pe rețele neuronale (perceptron multinivel)
- Clasificatori bazați pe vectori suport

#### 1. Problema clasificării

*Scopul unui model de clasificare:* stabilirea clasei căreia îi aparțin datele de intrare

*Construirea unui clasificator* = extragerea unui model de clasificare pornind de la date de antrenare; procesul de construire este denumit învățare sau antrenare.

*Validarea unui clasificator:* tehnica validării încrucișate

*Evaluarea performanței:* matrice de confuzie, acuratețe, sensibilitate, specificitate, regăsire (recall), precizie (precision), F-measure, curba receiver-operating characteristic (ROC) și aria de sub curbă (AUC=area under curve).

**Accuracy** =  $(TP+TN)/(TP+TN+FP+FN)$  = nr date clasificate corect/ nr total de date

**Sensitivity** =  $TP/(TP+FN)$  (TP rate sau recall = rata de regăsire)

**Specificity** =  $TN/(TN+FP)$  (TN rate), 1-specificitate =  $FP/(TN+FP)$  = FP rate

**Precision** =  $TP/(TP+FP)$  (nr cazuri real pozitive/ nr cazuri clasificate ca fiind pozitive)

**F-measure** =  $2 * precision * recall / (precision + recall)$

**Kappa** =  $(Accuracy - ExpectedAcc)/(1-ExpectedAcc)$  - ajustare a măsurii de acuratețe care ține cont de proporția cazurilor în care clasificatorul generează răspuns corect datorită întâmplării - este util în particular în cazul seturilor nebalansate de date.

$$ExpectedAcc = (TP+FP)(TP+FN)/N^2 + (TN+FN)(TN+FP)/N^2 \quad (N = \text{nr total de date})$$

**Curba ROC:** TP rate (sensitivitate) vs. FP rate (1-specificitate). Se calculează perechi de valori (FP rate, TP rate) pentru fiecare felie din setul de antrenare (în contextual validării încrucișate cu n felii) și se unesc punctele (0,0),  $P_1, \dots, P_n$ , (1,1) (unde  $P_i$  este punctul având a i-a valoare în ordine crescătoare a valorii FP rate). Ideal ar fi ca toate punctele să aibă coordonatele (0,1), iar aria de sub curbă să coincidă cu aria pătratului definit de (0,0) și (1,1), adică 1.

Pentru seturile de date nebalansate (semnificativ mai multe exemple într-o clasă în raport cu celelalte clase) se folosește coeficientul de corelație Matthew (MCC):

$$MCC = (TP * TN - FP * FN) / \sqrt{((TP + FP) * (TP + FN) * (TN + FP) * (TN + FN))}$$

**Exercițiul 1.** Familiarizare cu pachetul “caret” – Classification And Regression Training (interfață unitară pentru peste 200 de variante de algoritmi de clasificare și regresie implementați în diverse pachete R). Facilități principale:

- Partiționare seturi de date în subset de antrenare și subset de testare (function [createDataPartition](#))
- Antrenarea unui model (funcția [train](#)) și definirea unei proceduri de control a procesului de antrenare și determinare a valorilor parametrilor de control (hiperparametri) (funcția [trainControl](#)); permite specificarea unei strategii de validare: selecție aleatoare cu repetare - *bootstrapping* (implicit – cu 25 de repetări), *cross-validation* - validare încrucișată etc.
- Includerea unui pas de pre-procesare în faza de antrenare (e.g. parametrul [preProc](#) în funcția [train](#)); exemple de preprocesare: centrare (“[center](#)”), scalare cu valoarea abaterii standard (“[scale](#)”), transformare PCA (“[pca](#)”), imputarea valorilor absente (“[knnImpute](#)”)

Punct de pornire: <https://cran.r-project.org/web/packages/caret/caret.pdf> , [templateUsingCaret.r](#)

## 2. Clasificatori bazați pe arbori de decizie

Un arbore de decizie conține:

- *Noduri interne* (fiecare nod intern are asociat un atribut); fiecare nod intern are asociate condiții de ramificare, fiecare condiție corespunzând unei ramuri în arbore)
- *Noduri frunză* (fiecare nod frunză are asociată o clasă)

Mod de construire:

- pentru nodul rădăcină se identifică atributul și condițiile de ramificare caracterizate prin cel mai mare câștig informațional (information gain – IG) în ceea ce privește clasificarea datelor din setul de date
- pentru fiecare dintre ramuri și setul de date corespunzător ramurii se aplică în mod recursiv aceeași strategie până este îndeplinită o condiție de oprire (s-a ajuns la un set care conține date din aceeași clasă sau la un set mic de date)

*Observatie:* un aspect important este cel referitor la complexitatea arborelui (sunt de preferat arbori cât mai simpli); simplificarea unui arbore (*tree pruning*) se poate realiza:

- în etapa de construire (stopând procesul de ramificare dacă numărul de elemente din setul curent este mai mic decât un prag)
- după construirea arborelui, prin înlocuirea unor subarbori cu noduri frunză (în cazul în care performanța clasificatorului nu este alterată în mod semnificativ)

Mod de utilizare: pentru o anumită dată se identifică ramura din arborele de decizie care se potrivește cu valorile atributelor iar eticheta nodului frunză la care se ajunge reprezintă clasa asociată datei.

*Pachete R:* [rpart](#), [party](#)

*Implementare în Weka:* Id3, J48

**Exercițiul 2/ R:** Construire, vizualizare și evaluarea performanțelor unor arbori de clasificare pentru: (a) iris dataset; (b) Titanic dataset; (c) Breast Cancer dataset; (d) [bioinfo] DNA (din pachetul mlbench). Utilizare pachete: party, rpart și caret

*Punct de pornire:* DecisionTrees\_PartyRpart.R, DecisionTrees\_caret.R

**Exercițiul 2/Weka:** Compararea algoritmilor de construire a arborilor de decizie

- a) Deschideți în Weka fișierul “weather\_nominal.arff”. Comparați performanțele următorilor clasificatori: Id3, J48 (variantele “pruned” respectiv “unpruned” – se setează din lista de parametri de la J48)
- b) Deschideți în Weka fișierul “weather\_numeric.arff” și încercați să aplicați aceleași prelucrări
- c) Deschideți în Weka fișierul “glass.arff” și analizați impactul asupra performanței clasificatorului J48 a numărului minim de date corespunzător unui nod frunză (se specifică ca valoare a parametrului minNumObj)

### 3. Clasificatori bazați pe reguli de decizie

Regulile de decizie sunt de forma IF <antecedent> THEN <eticheta clasa>, unde partea de antecedent conține condiții referitoare la valorile atributelor combinate prin operator de conjuncție sau disjuncție.

*Mod de construire:*

- Pornind de la un arbore de decizie, pentru fiecare ramură se poate construi o regulă de clasificare (condițiile din antecedent sunt combinate prin conjuncții)
- Direct pornind de la date prin algoritmi de acoperire (“covering algorithms”)

*Pachete R:* caret include clasificatori bazați pe reguli din RWeka (JRip și OneR)

*Implementare în Weka:*

- Pornind de la arbori de decizie: M5rules, PART
- Covering algorithms: OneR, JRip

**Exercițiul 3/ R:** Construire, vizualizare și evaluarea unor clasificatori bazați pe reguli pentru: (a) iris dataset; (b) Breast Cancer dataset; (c) Titanic dataset; (d) [bioinfo] DNA (din pachetul mlbench). Utilizare pachete: caret și RWeka

*Punct de pornire:* ClassificationRules.R

**Exercițiul 3/Weka:** analiza algoritmilor de extragere a regulilor de decizie

- a) Deschideți fișierul “weather.nominal.arff” și aplicați următoarele modele de clasificare bazate pe reguli: OneR, JRip, PART
- b) Identificați cel mai bun clasificator bazat pe reguli pentru “weather.nominal.arff”

#### 4. Clasificatori bazati pe instanțe (Instance Based Learning Classifiers)

Cei mai simpli clasificatori bazat pe instanțe sunt cei din categoria “cel mai apropiat vecin” (nearest neighbor)

*Mod de construire:*

- modelul de clasificare constă chiar din setul de date (nu există etapă propriu-zisă de antrenare)

*Mod de utilizare:*

- in procesul de clasificare, pentru o nouă instanță se identifică cele mai similare instanțe din setul de antrenare și clasa dominantă va fi considerată ca fiind cea corespunzătoare noii date.

*Pachete R:* pachet [caret](#)/ metoda [knn](#), pachet [class](#) / metode [knn](#), [knn1](#)

*Implementare Weka:* [IB1](#) (se utilizează un singur vecin), [IBk](#) (se utilizează k vecini)

**Exercițiul 4/ R:** Construirea și evaluarea performanțelor unui clasificator de tip kNN pentru: (a) Breast Cancer dataset; (b) Titanic dataset. (c) [\[bioinfo\]](#) DNA (din pachetul [mlbench](#)).

- (i) Analizați tipului atributelor și a preprocesărilor (e.g. standardization) asupra performanței în clasificare și asupra valorii determinate a parametrului de control ( $k$  = numărul de vecini utilizați în faza de predicție)
- (ii) Visualizați curba curba ROC (in cazul problemelor de clasificare binară) și analizați valoarea ariei AUC.

*Punct de pornire:* [kNN.r](#)

**Exercițiul 4/ Weka:** Alegerea numărului de “vecini” la k-Nearest Neighbour

- a)** Deschideți in Weka fișierul “[breast-w.arff](#)”. Pentru [IBk](#) identificați valoarea lui  $k$  (din  $\{1,2,3,4,5,6,7,8,9,10\}$ ) care conduce la cel mai bun rezultat (Indicație: modificați valoarea parametrului  $k$  de la [IBk](#)).
- b)** Deschideți in Weka fișierul “[glass.arff](#)” și efectuați aceeași prelucrare

#### 5. Modelele probabiliste

Permit estimarea probabilității ca o dată să aparțină unei clase ( $P(C_k|(a_1,a_2,\dots,a_n))$ ). Clasa corespunzătoare datei este cea pentru care probabilitatea este maximă. Calculele se bazează pe formula lui Bayes:  $P(C_k|a_1,a_2,\dots,a_n)=P(a_1,a_2,\dots,a_n|C_k)P(C_k)/P(a_1,a_2,\dots,a_n)$ .

*Variante:*

- Model Bayesian simplu ([Naïve Bayes](#)): presupune că atributele nu sunt corelate (probabilitatea observării unei date este produsul probabilităților de a observa fiecare dintre valorile atributelor).
- Rețea cauzală ([Bayesian Net](#) sau [Belief network](#)): permite descrierea unui model sub forma unui graf orientat în care sunt specificate relații de cauzalitate.

Pachete R: pt Naïve Bayes: [caret](#) / metoda [bn](#); pt rețele Bayesiene: pachet [bnlearn](#)

**Exercițiu 5/ R:** Construiți și evaluați performanța unui clasificator de tip Naïve Bayes pt: (a) Breast Cancer dataset; (b) Titanic dataset; (d) [\[bioinfo\]](#) DNA (din pachetul [mlbench](#)).

Punct de pornire: [NaiveBayes.R](#)

### Exercițiul 5/ Weka:

- a) Deschideți fișierul “[weather.nominal.arff](#)”. Aplicați clasificatorii [Naïve Bayes](#) și [Bayes Net](#)
- b) Utilizați [Bayes Network Editor](#) (de la Weka [GUI Choser](#) -> [Tools](#)) pentru a construi o rețea bayesiană. Etape:
  - Selectare set date: [Tools->Set Data](#)
  - Plasare noduri (fiecare atribut, inclusiv cel de clasă va avea asociat un nod) folosind [Edit->Add node](#)
  - Specificare relații folosind [Edit->Add arc](#) (de exemplu: se plasează arc de la nodul corespunzător clasei la fiecare dintre nodurile asociate atributelor)
  - Antrenare rețea ([Tools->Learn network \(învățarea structurii\)](#), [Tools->Learn CPT](#) (determinarea tabelor de probabilitati))
  - Utilizare: se selectează valorile corespunzătoare instanței ce urmează să fie clasificată (folosind click dreapta pe fiecare nod asociat unui atribut și [Set evidence](#)); decizia se bazează pe valorile probabilităților corespunzătoare nodului asociat clasei

Obs: detalii privind utilizarea rețelelor bayesiene se găsesc la [http://www.cs.waikato.ac.nz/~remco/weka\\_bn/](http://www.cs.waikato.ac.nz/~remco/weka_bn/)

## 6. Rețele neuronale

Etape în proiectarea unei rețele neuronale pentru clasificare:

- *Stabilirea arhitecturii.* In cazul rețelelor multi-nivel (multi-layer perceptrons) se fixează:
  - Număr de unitati de intrare = numărul atributelor
  - Număr de nivele/unități ascunse – depinde de complexitatea problemei
  - Număr de unități de ieșire:
    - Clasificare binară: o unitate (rezultatul se interpretează folosind o valoare prag) sau două unități (unitatea care produce cea mai mare valoare va indica clasa)
    - Clasificare multiplă: numărul de unități coincide cu numărul de clase iar clasa dominantă este cea considerată clasa răspuns
- *Stabilirea funcțiilor de activare.* Pentru rețelele antrenate folosind algoritmul Backpropagation:
  - Unitățile ascunse au funcții de activare de tip sigmoidal (funcția logistică sau tanh)
  - Unitățile de ieșire au funcții de activare de tip sigmoidal sau funcții liniare
- *Alegerea algoritmului de antrenare și a parametrilor acestuia.* In cazul algoritmului Backpropagation parametrii uzuali de control sunt:
  - Numărul de epoci de antrenare
  - Rata de învățare
  - Coeficientului termenului de tip „moment”

*Implementare R:* pachet [nnet](#) (inclus in [caret](#)) – rețea neuronală cu un nivel ascuns(cu diferite funcții de activare la nivelul de ieșire – logistică (implicit), liniară, softmax) antrenată folosind un algoritm de tip Backpropagation ce minimizează MSE (mean squared error – implicit) sau entropia încrucișată (când e utilizată softmax ca funcție de activare la nivelul de ieșire). Inclusă în Rattle.

*Implementare Weka:* MultilayerPerceptron = rețea feedforward antrenată cu Backpropagation (varianta cu moment); arhitectura rețelei este implicit cu un nivel de unități ascunse; numărul de unități ascunse se poate specifica prin opțiunea [hiddenLayers](#)

**Exercițiul 6/ R:** Construiți și evaluați performanța unei rețele neuronale cu un nivel ascuns pt: (a) Breast Cancer dataset; (b) Titanic dataset; (d) [\[bioinfo\]](#) DNA (din pachetul [mlbench](#)).

Punct de pornire: [NeuralNetworks.R](#)

**Exercițiul 6/Weka :** Analiza influenței arhitecturii rețelei asupra performanțelor clasificatorului

- a) Deschideți fișierul “[breast-w.arff](#)”
- b) Antrenați o rețea neuronală folosind setările standard (un nivel ascuns cu număr de unități ascunse cu  $K=(nr\ attribute+nr\ clase)/2$  unități)
- c) Comparați performanțele rețelei pentru următoarele valori ale parametrului [Hidden Layers \(H\)](#):
  - a. ‘a’ ( $K=(nr\ attribute+nr\ clase)/2$ )
  - b. ‘i’ ( $K=nr\ attribute$ )
  - c. ‘o’ ( $K=nr\ clase$ )
  - d. ‘t’ ( $K = nr\ attribute+nr\ clase$ )
  - e. 4,2 (două nivele ascunse cu 4, respectiv 2 unități)
- d) Analizați influența ratei de învățare și a coeficientului corespunzător termenului “moment”.
- e) Comparați rezultatul obținut folosind [MultilayerPerceptron](#) cu cel obținut folosind o rețea de tip RBF – radial basis function ([functions->RBF Network](#))

**Obs.** Prin activarea opțiunii GUI de la Multilayer Perceptron poate fi vizualizată arhitectura rețelei.

## 7. Clasificare folosind vectori suport (Support Vector Machines)

**Exercițiul 7/ R:** Construiți și evaluați performanța unui clasificator de tip SVM pt: (a) Breast Cancer dataset; (b) Titanic dataset; (d) [\[bioinfo\]](#) DNA (din pachetul [mlbench](#)). Analizați influența nucleului (metode [svmRadial](#), [svmLinear](#), [svmPoly](#)).

Punct de pornire: [SVM.r](#)

Varianta de clasificatori bazați pe vectori suport implementată în Weka folosește un algoritm rapid pentru rezolvarea problemei de optimizare (Sequential Minimal Optimization). In cazul clasificării in M clase problema se transformă în mai multe probleme de clasificare binară (corespunzătoare perechilor de clase)

### Exercițiul 7/Weka:

- c) Deschideți în Weka fișierul “breast-w.arff” și utilizați SVM (functions->SMO) ; analizați influența funcției nucleu asupra performanțelor clasificatorului (variante: PolyKernel, RBFKernel)
- d) Deschideți în Weka fișierul “arrhythmia.arff” și efectuați aceeași prelucrare; încercați să rezolvați aceeași problemă utilizând o rețea neuronală.

Exemple adiționale (scikit-learn)

- [https://scikit-learn.org/stable/auto\\_examples/index.html#classification](https://scikit-learn.org/stable/auto_examples/index.html#classification)

### Temă:

1. Utilizați Rattle sau R (la alegere) pentru a construi și compara performanțele unor clasificatori bazați pe (i) arbori de decizie; (ii) vectori suport; (iii) rețele neuronale.

Seturi de date (la alegere):

- Glass (from mlbench);
- LetterRecognition (from mlbench);
- German credit data (GermanCredit in caret)  
[<http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29>]
- [bioinfo] Molecular Biology (Promoter Gene Sequences) Data Set  
[<http://archive.ics.uci.edu/ml/datasets/Molecular+Biology+%28Promoter+Gene+Sequences%29>]