

# Inteligență Artificială: Tema 2 - ML aplicat

## Seria CC

Alexandru Sorici, Alexandru Baci, Florin Dumitrescu, Cătălin Rîpanu, Ana Simion

## 1. Descriere generală

În practica de zi cu zi a unui inginer sau cercetător în domeniul inteligenței artificiale, și al învățării automate în particular, intră frecvent următoarele trei aspecte:

- Vizualizarea și “explorarea” datelor unei probleme (eng. Exploratory Data Analysis - EDA)
- Încercarea de a extrage attribute ale datelor problemei pentru a fi utilizate în obiectivul de analiză ales (e.g. clasificare, regresie, detectie de anomalii)
- Evaluarea mai multor modele pentru găsirea soluției celei mai bune pentru problema dată

Sarcinile de lucru solicită utilizarea de biblioteci de **vizualizare a datelor (crearea de diagrame)**, **extragerea de attribute (feature extraction)** pentru folosirea algoritmilor de clasificare discutați la curs (folosind implementările voastre de laborator, precum și cele dintr-o bibliotecă existentă, [scikit-learn](https://scikit-learn.org/)).

## 2. Descrierea Seturilor de Date

Se propun următoarele două seturi de date pentru o **problemă de clasificare**.

Scopul este să antrenați modele de învățare care să reușească să prezică valoarea variabilelor țintă cu performanțe cât mai bune. Găsiți seturile de date pe Moodle la secțiunea aferentă acestei teme.

Fiecare set de date este furnizat sub forma unui fișier CSV, având pe prima linie numele coloanelor.

Fiecare set de date are o împărțire din oficiu a datelor în:

- Set de date pentru **antrenare** (sufixul **\_train**)
- Set de date pentru **testare** (sufixul **\_test**)
- Set de date complet, antrenare și test laolaltă (sufixul **\_full**)

### Pirvision

---

Setul de date PIRvision conține date pentru detectarea prezenței, colectate de la un nod de senzori pasivi în infraroșu (PIR) cu consum redus de energie și sincronizat electronic, utilizat în medii rezidențiale și de birou. Fiecare observație reprezintă 4 secunde de activitate umană înregistrată în câmpul vizual (FoV) al senzorului. Clasificarea este de tip multi-class.

### Poluarea aerului

---

Acest set de date, care acoperă 170 de țări și peste 300 de orașe, oferă o perspectivă cuprinzătoare asupra dinamicii calității aerului la nivel global. Concentrarea este pusă pe poluanți esențiali precum monoxidul de carbon (CO), ozonul (O<sub>3</sub>), dioxidul de azot (NO<sub>2</sub>), particulele fine (PM2.5), etc. Setul de

date reprezintă o resursă valoroasă pentru oamenii de știință din domeniul mediului, factorii de decizie politică și cercetători, oferind informații utile pentru analizarea tendințelor privind calitatea aerului, elaborarea de politici eficiente și contribuția la un mediu mai sănătos. Setul de date include coloane esențiale precum numele țării, numele orașului, valoarea generală a indicelui calității aerului (AQI) și concentrațiile unor poluanți specifici. Aceste informații permit realizarea unor analize detaliate și studii de corelație. Cercetătorii pot identifica modele și tendințe în calitatea aerului prin explorarea relațiilor dintre diverși poluanți și valoarea generală a AQI. Prin amploarea și detalierea sa, acest set de date este un instrument indispensabil pentru cei interesați de înțelegerea calității aerului și e implicarea activă în eforturile colective pentru un mediu mai curat și mai sănătos. Obiectivul pentru acest set de date este de a realiza o clasificare pe mai multe clase (Good, Moderate, Unhealthy for Sensitive Groups, Unhealthy, Very Unhealthy și Hazardous) a nivelului calității aerului, în funcție de datele colectate din diferite zone geografice.

Tabelele care conțin atributele, alături de tipul acestora (numeric sau categoric) și descrierea corespunzătoare, sunt disponibile în Anexă-Descrierea atributelor.

## 3. Cerințe

### 3.1. Explorarea Datelor (Exploratory Data Analysis) [2p]

Primul pas recomandat în rezolvarea unei probleme de clasificare este obținerea unor informații asupra caracteristicilor principale ale problemei. De regulă, foarte folositoare în această etapă este aplicarea unor metode de **vizualizare a datelor** și de **raportare a distribuțiilor de valori** pe fiecare variabila folosită în predicție.

Este, de asemenea, esențial a se identifica dacă:

- Există atribute cu **valori lipsă**
- Există atribute cu **valori extreme** (eng. outlier identification)
- Există atribute **redundante / neinformative**

### Analize cerute

#### 1. Analiza tipului de atribute și a plajei de valori a acestora

Înainte de utilizarea unui model de ML pentru un set de date este importantă identificarea tipurilor de atribute (features) din setul de date și a valorilor acestora.

Este relevantă distincția dintre:

- Atribute cu **valori numerice continue**
- Atribute cu **valori discrete** (e.g. zilele unei săptămâni, tipuri de boli)
- Atribute **ordinale** - unde valorile indică o relație de ordine (e.g. număr de steluțe la recenzia unui produs). Atributele ordinale au de regulă o valoare numerică, dar valoarea este subiectivă (e.g. Între un produs *cu rating* 3 și unul cu rating 5, diferența nu este doar valoarea numerică 2).

Pentru fiecare tip de atribut este relevantă extragerea unor statistici la nivelul exemplurilor din setul de date.

**Pentru atribute numerice continue extrageți și prezentați într-un tabel**

- numărul de exemple din setul de date care **nu** au valori lipsă
- valoarea medie
- deviația standard a valorilor
- valoarea minimă
- valoarea percentilei de 25% (valoarea sub care se găsesc 25% din exemple)
- valoarea percentilei de 50% (valoarea sub care se găsesc 50% din exemple) - aceasta este valoarea mediană
- valoarea percentilei de 75% (valoarea sub care se găsesc 75% din exemple)
- valoarea maximă

**Afișați grafice** de tip [Boxplot](#) (a se vedea indicațiile din [Anexă: Explorarea datelor](#)) prin care să observați vizual plajele de valori ale atributelor numerice continue.

**Pentru attribute discrete sau ordinale** extrageți și prezentați într-un tabel:

- Numărul de exemple din setul de date care **nu** au valori lipsă
- Numărul de **valori unice**

**Prezentați grafice** de tip histogramă (a se vedea indicațiile din [Anexă: Explorarea datelor](#)) pentru a observa vizual care este distribuția valorilor pentru fiecare atribut categoric / ordinal peste exemplele din setul de date.

## 2. Analiza echilibrului de clase

**Realizați un grafic** al frecvenței de apariție a fiecărei etichete (clase) în setul de date de antrenare / test, folosind **bar plot** / **count plot** (a se vedea indicațiile din [Anexă: Explorarea Datelor](#)).

Pentru realizarea unor astfel de bar plots puteți folosi mai multe biblioteci:

- Folosind biblioteca seaborn pentru [barplot](#) sau [countplot](#)
- Direct dintr-un DataFrame Pandas folosind [pandas.DataFrame.plot.bar](#)

Dacă într-un set de date numărul de exemple aferent unui subset de clase este semnificativ mai mare decât exemplele aferente celorlalte clase, atunci clasificarea va fi mai dificilă, riscându-se predicții eronate pentru clasele cu *suport* (i.e. număr de exemple etichetate cu clasa respectivă) scăzut.

În cazul unui set de date cu suport al clase dezechilibrat se va insista pe raportarea metricilor de performanță de tip **precizie, recall** și **F1**.

## 3. Analiza corelației între attribute

1. **Realizați analize** de [corelație](#) (a se vedea indicațiile din [Anexă: Explorarea Datelor](#)) între **attributele numerice continue**, pentru a identifica dacă există attribute redundante.
2. **Realizați analize** de **corelație** între **attributele categorice**, pentru a identifica dacă există attribute redundante.

Două attribute care sunt puternic corelate (valori ale indicelui de corelație apropiate de -1 sau 1) vor avea, în mare măsură, aceeași putere de predicție, astfel încât utilizarea amândurora într-un algoritm de predicție nu aduce performanțe suplimentare, ci doar adaugă la complexitatea de calcul.

Ca atare o analiză a corelației pe perechi de atribute ne poate indica dacă putem renunța la unele care sunt puternic corelate între ele.

## 3.2. Preprocesarea datelor [2p]

Folosind înțelegerea asupra datelor câștigată la pasul 3.1, putem aplica o serie de pași de preprocesare a datelor care să abordeze următoarea serie de posibile probleme.

### 1. Date lipsă pentru un atribut într-un eșantion

Pentru atribute cu valori lipsă se utilizează o **procedură de imputare**. Acestea pot fi univariate (să implice doar atributul pentru care lipsește o valoare) sau multivariate (valoarea atributului lipsă se obține în funcție de valorile celorlalte atribute din eșantion).

Exemple de imputare univariată sunt: valoarea medie, valoarea mediană, valoarea cea mai frecventă. Imputările multivariate implică, de regulă, diverse forme de regresie peste valorile celorlalte atribute.

**Sarcina este** de a determina care atribute au date lipsă și să folosiți o metodă de imputare potrivită pentru acel atribut (a se vedea indicațiile din [Anexă: Preprocesare Datelor](#))

### 2. Valori extreme pentru un atribut într-un eșantion

Atributele numerice pot avea valori extreme (e.g. fie prea mici, fie prea mari în raport cu media sau mediana valorilor pentru acel atribut). Valorile extreme pun probleme în cazul unei clasificări, astfel că dorim să le identificăm în avans și să înlocuim valoarea lor printr-o procedură de imputare (le tratăm ca pe o valoare lipsă)

**Sarcina este** de:

- A determina dacă o valoare a unui atribut numeric este o valoare extremă (a se vedea indicațiile din [Anexă: Preprocesare Datelor](#))
- A înlocui acea valoare printr-una obținută prin imputare

### 3. Atribute redundante (puternic corelate)

Dacă analiza de la punctul 3.1 indică prezența unor atribute numerice sau categorice puternic corelate, se va proceda prin eliminarea acelor atribute din setul celor folosite în predicție pentru a simplifica calculul algoritmului folosit.

**Sarcina este** de a argumenta în raportul dacă ați eliminat vreun atribut pe seama analizei de redundanță a acestuia.

### 4. Plaje valorice de mărimi diferite pentru atributele numerice

Atributele numerice din setul de date pot avea scale valorice semnificativ diferite (e.g. valori de ordinul miilor pe un atribut, și de ordinul unităților pe un altul). În acest caz, algoritmi precum Regresia Logistică

vor fi puternic afectați pentru că o combinație liniară a valorilor de atribut va fi dominată de atributul cu valori numerice foarte mari). Ca atare, dorim să standardizăm valorile atributelor numerice.

**Sarcina este** de a efectua operația de **standardizare** a valorilor atributelor numerice (a se vedea indicațiile din [Anexă: Preprocesare Datelor](#))

### 3.3 Utilizarea algoritmilor de Învățare Automată [6p]

- Pentru a putea utiliza DecisionTreeClassifier, RandomForestClassifier, LogisticRegression sau MLP din sklearn, variabilele **categorice** (incluzând-o pe cea țintă) trebuie convertite într-o formă numerică. Utilizați în acest scop clase disponibile în biblioteca scikit-learn: [LabelEncoder](#) sau [OneHotEncoder](#). De exemplu, puteți folosi LabelEncoder pentru a encoda variabila țintă și OneHotEncoder pentru a encoda variabilele predictor de tip categoric. Un exemplu de encodare în stil one-hot există și în laboratorul de Arbori de Decizie, unde s-a utilizat metoda [pandas.get\\_dummies](#).

#### Arbori de Decizie [1.5p]

---

Dezvoltați un model de arbore de decizie pentru predicția pe seturile de date furnizate.

Antrenați și evaluați un model de arbore de decizie folosind biblioteca scikit-learn. Documentația pentru DecisionTreeClassifier este [disponibilă aici](#).

**Documentați** în raportul vostru hiperparametrizarea folosită pentru algoritmul de arbore de decizie:

- adâncime maximă
- număr minim de exemple într-o frunză
- tipul criteriului de decizie (entropy, gini, log\_loss)
- utilizarea opțiunii de ponderare a claselor (class\_weight) utilă în cazul unui set de date cu clase dezechilibrate

#### Păduri Aleatoare [1.5p]

---

Dezvoltați un model de tip pădure aleatoare (Random Forest) pentru predicția pe seturile de date furnizate.

Antrenați și evaluați un model de RandomForest folosind biblioteca scikit-learn. Documentația pentru RandomForestClassifier este [disponibilă aici](#).

**Documentați** în raportul vostru hiperparametrizarea folosită pentru algoritmul de tip RandomForest, e.g.:

- Adâncime maximă a unui arbore,
- Număr minim de exemple într-o frunză
- Tipul criteriului de decizie (entropy, gini, log\_loss)
- Utilizarea opțiunii de ponderare a claselor (class\_weight) utilă în cazul unui set de date cu clase dezechilibrate
- Numărul de estimatori folosiți,
- Dimensiunea proporției din setul de date inițial utilizată în antrenarea fiecărui estimator individual
- Proporția de atribute folosită în antrenarea fiecărui estimator individual

## Regresie Logistică [1.5p]

---

Dezvoltați un model de regresie logistică pentru predicția pe seturile de date furnizate.

### Implementare manuală

- Pornind de la rezolvarea din laborator, implementați modelul de regresie logistică pentru problema dată.

**Documentați** în raportul vostru abordarea folosită pentru algoritmul de regresie logistică:

- Tipul de encodare folosit pentru fiecare atribut categoric
- Setările algoritmului de optimizare de tip gradient descent folosit: e.g. tip de optimizator, learning rate
- Metoda de regularizare folosită

## Multi-Layered Perceptron (MLP) [1.5p]

---

Dezvoltați un model de rețea neurală de tip MLP pentru predicția pe seturile de date furnizate.

### Implementare folosind biblioteca scikit-learn

- Folosiți documentația din biblioteca scikit-learn pentru a antrena și evalua un model de [clasificator MLP](#).

**Documentați** în raportul vostru hiperparametrizarea folosită pentru rețeaua neurală de tip MLP:

- Arhitectura: numărul și dimensiunea fiecărui strat din rețea, tipul funcțiilor de activare folosite
- Configurarea optimizatorului: tip de optimizator folosit, learning rate, număr de epoci de antrenare, dimensiunea batch-urilor de antrenare
- Metode de regularizare folosite: utilizare early\_stopping, coeficient de regularizare L2 pe ponderi

## Evaluarea algoritmilor

---

În raportul vostru trebuie să prezentați **cel puțin** următoarele:

- Descrieți **setul de hiperparametrii** final pe care i-ați folosit **pentru fiecare algoritm** în parte
- Pentru **fiecare set de date** și **fiecare algoritm**, realizați o [matrice de confuzie](#) peste clase, pentru a observa care sunt clasele pentru care algoritmul greșește în clasificare
- Realizați un tabel comparativ al algoritmilor pe fiecare set de date. În tabel prezentați **acuratețea generală de clasificare, precizie / recall / F1 la nivelul fiecărei clase în parte** ([definiții ale metricilor pe Wikipedia](#))
  - Pe linii va fi indexată numele algoritmului
  - Pe coloane vor fi prezentate metricile cerute
  - **Relevați prin bolduire** valorile maxime pentru fiecare metrică
- Pentru rețeaua de tip MLP furnizați graficele **curbelor de eroare și de performanță** (acuratețea) pentru seturile de date de antrenare și test. Trasați curbele de train și test **pe același grafic** pentru a observa dacă modelul vostru intră în overfit.

- **Comentați asupra rezultatelor, explicând de ce credeți că algoritmul cu rezultatul cel mai bun obține această performanță.**

## 4. Predarea temei

Tema se trimite ca o arhivă .zip care conține cel puțin 2 fișiere:

- un **fișier Python** sau **Jupyter Notebook** care conține implementarea analizelor cerute
- un **fișier PDF** care prezintă **raportul de rezultate și interpretare** a analizelor cerute
  - **Cerința 3.1** - cuprinde toate vizualizările și statisticile cerute. **Este obligatorie** prezența în text a **unei interpretări / analize** a diagramelor sau tabelelor rezultante.
  - **Cerința 3.3** - include **raportarea preprocesării datelor** și a evaluării algoritmilor de clasificare pentru cele două tipuri de seturi de date propuse. **Este obligatorie** prezența în text a **unei interpretări / analize** a rezultatelor obținute (e.g. care este impactul dezechilibrului de clase - dacă el există - asupra performanțelor, cât de puternic este impactul hiper-parametrilor asupra performanței fiecărui algoritm considerat, care sunt clasele cu cele mai bune predicții).

## 5. Anexă

---

În rezolvarea temei este încurajată utilizarea a două biblioteci pentru sarcinile de analiză, preprocesare și aplicare a algoritmilor de predicție: [pandas](#) și [scikit-learn](#), ambele în python.

**Biblioteca Pandas** este foarte frecvent utilizată pentru citirea, procesarea și analiza datelor livrate în format tabelar, așa cum este cazul seturilor de date din tema.

Revizitați indicațiile din laboratorul despre Arbori de Decizie și Paduri Aleatoare pentru a vedea exemple tipice de lucru cu date în pandas, îndeosebi, lucrul cu [pandas.DataFrame](#).

**Scikit-learn** este o bibliotecă ce conține implementări a multor proceduri de prelucrare statistică a datelor, precum și a algoritmilor clasici de ML care au fost studiați și în cadrul laboratorului (e.g. Decision Trees, Random Forest, Logistic Regression, Multi-Layered Perceptron).

### 5.1. Explorarea datelor

#### Citirea datelor

---

Citirea datelor din fișiere CSV se poate face cu [pandas.read\\_csv](#) ceea ce va rezulta în obținerea unui DataFrame.

**Atenție:** datele lipsă din seturile de date furnizate sunt trecute ca spațiu gol în CSV-uri. Pandas va interpreta valoarea goală drept **numpy.NaN**, iar existența unei valori de tip NaN poate fi verificată prin [pandas.isna](#).

## Analiza atributelor numerice

---

Pentru a determina statistici generale despre coloane cu attribute numerice se poate utiliza apelul [`pandas.DataFrame.describe`](#).

Afișarea statisticilor în grafice de tipul **boxplot** se poate face direct dintr-un DataFrame folosind metoda [`pandas.DataFrame.boxplot`](#).

## Analiza atributelor categorice sau ordinale

---

Pentru a determina valorile unice ale unei coloane de attribute categorice dintr-un DataFrame se poate utiliza metoda [`pandas.unique`](#).

Realizarea unei histograme cu frecvența de apariție a fiecărei valori de atribut se poate face direct dintr-un DataFrame prin metoda [`pandas.Series.hist`](#) (HINT: Fixați și parametrul `width` pentru histogramă pentru afișare mai plăcută).

Pentru obținerea unei serii Pandas indexată după valorile unui atribut și numărând exemplele care au acea valoare poate fi folosită metoda [`pandas.DataFrame.value\_counts`](#).

**De notat:** Aceste metode pot fi folosite și pentru **analiza dezechilibrului de clase**.

## Analiza corelației între attribute

---

Analiza corelației între **attribute numerice** se poate face direct dintr-un DataFrame prin metoda [`pandas.DataFrame.corr`](#), utilizând criteriul Pearson.

O interpretare vizuală a gradului de corelare se poate obține cu metoda `matshow()` din **matplotlib**, ca în [exemplul de aici](#).

Pentru attribute categorice, corelația între două attribute se poate evalua prin [testul statistic Chi-Pătrat](#), a cărui ipoteză nulă este că variabilele sunt independente (nu sunt corelate), iar testul poate confirma sau infirma această ipoteză.

Un exemplu de evaluare se poate [vedea aici](#), unde sunt utilizate metode din două biblioteci (pandas și [scipy](#)).

## 5.2. Preprocesarea datelor

### Determinarea valorilor extreme

---

Determinarea valorilor extreme pentru attribute numerice se face pe baza statisticilor de bază ale acestora obținute prin metode precum `pandas.describe()`.

În particular, metoda ce utilizează **diferența inter-quartilă (inter-quartile range - IQR)** poate fi folosită în cazul seturilor de date furnizate.



Un exemplu de utilizare [se găsește aici](#). De notat că se poate încerca setarea quantilelor și la valori mai extreme, precum  $Q1 = 0.1$  și  $Q3 = 0.9$ , în cazul în care variabilele au o plajă foarte largă de valori, pentru a distinge valori outlier, de simple valori mari.

## Imputarea valorilor lipsă sau extreme

---

În procedura de imputare se pot utiliza două clase tipice din biblioteca **scikit-learn**: [SimpleImputer](#) (imputare univariată) și [IterativeImputer](#) (imputare prin raport la celelalte attribute din eșantion).

Exemple de utilizare a acestora se pot observa în [documentația de aici](#).

**De notat** că puteți încerca aplicarea mai multor metode de imputare pe un anumit set de date, obținând astfel diferite transformări ale acestuia. Algoritmii de predicție pot fi rulați apoi pe fiecare transformare în parte, păstrând metoda de imputare ce oferă cele mai bune rezultate.

## Standardizarea datelor

---

Pentru procedura de standardizare a datelor se poate recurge la clasele tipice existente în **scikit-learn**. Exemple de utilizare se [găsesc aici](#).

Acordați atenție îndeosebi claselor **StandardScaler**, **MinMaxScaler** sau **RobustScaler**.

## 5.3. Descrierea Atributelor

### Pirvision

Nume Atribut	Tip	Descriere
Timestamp	string	Momentul de timp (ora-minut-secundă) la care a început măsurătoarea
Day Index	int	A câta zi este în cadrul experimentului (corelată cu coloana Day)
Day	string	Ziua în care a fost efectuată măsurătoarea (format: an-lună-zi)
Temp (F)	int	Temperatura senzorului în grade Fahrenheit
Temp (C)	int	Temperatura senzorului în grade Celsius
OBS_1 - OBS_57	int	Valoarea observațiilor senzorului (intensitatea curentului), indici 1–57
Class	int	Clasa atribuită setului de observații ale senzorului

Poluarea aerului

Atribut	Tip	Semnificatie
Country Name	Numele țării în care au fost colectate datele.	Categoric
City Name	Numele orașului în care au fost colectate datele.	Categoric
Overall AQI	Valoarea generală a indexului calității aerului (Air Quality Index).	Numeric
Overall AQI Level	Nivelul general al AQI .	Categoric Target
CO AQI Value	Valoarea indexului de calitate a aerului pentru monoxid de carbon (CO).	Numeric
CO AQI Category	Categorie calitativă pentru AQI-ul CO (ex. Scăzut, Ridicat).	Categoric
Ozone AQI Value	Valoarea AQI pentru ozon (O <sub>3</sub> ).	Numeric
Ozone AQI Category	Categorie calitativă pentru ozon.	Categoric
NO2 AQI Value	Valoarea AQI pentru dioxid de azot (NO <sub>2</sub> ).	Numeric
NO2 AQI Category	Categorie calitativă pentru NO <sub>2</sub> .	Categoric
PM2.5 AQI Value	Valoarea AQI pentru particule PM2.5.	Numeric
PM2.5 AQI Category	Categorie calitativă pentru PM2.5.	Categoric
Volatile_Organic_Com pounds	Concentrația de compuși organici volatili (VOCs).	Numeric