Inteligență Artificială: Tema 2 - ML aplicat

Seria CD

Mihai Trăscău, Alexandru Sorici, Ana-Maria Simion,
Vlad Drăghici, Florin Dumitrescu, Vlad Florea

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 atribute 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 atribute (feature extraction)** pentru folosirea algoritmilor de clasificare discutați la curs (folosind implementările voastre de laborator, precum și cele dintr-o bibliotecă existentă, **scikit-learn**.

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**)

Salary Classification

Setul de date conține informații personale, educaționale și profesionale ale diferitor angajați. Obiectivul setului de date este clasificarea BINARĂ a angajaților în:

- angajați care câștigă sub $50K per an (<=50K).
- angajați care câștigă peste $50K per an (>50K).

Atributele exemplelor sunt următoarele:

Nume atribut	Tip de date	Semnificație
fnl	numeric	Caracteristică socio-economică a populației din care provine individul
hpw	numeric	Număr de ore de muncă pe săptămână

relation	categoric	Tipul de relație în care este implicat individul
gain	numeric	Câștigul de capital
country	categoric	Țara de origine
job	categoric	Meseria individului
edu_int	numeric	Numărul de ani de studiu
years	numeric	Vârsta individului
loss	numeric	Pierderea de capital
work_type	categoric	Tipul de meserie
partner	categoric	Tipul de partener pe care îl are individul
edu	categoric	Tipul de educație al individului
gender	categoric	Genul individului
race	categoric	Rasa individului
prod	numeric	Producerea de capital
gtype	categoric	Tipul contractului de munca

Clasificare Risc de Creditare

Obiectivul acestui set de date este de a determina dacă în urma acordării unui împrumut unei persoane fizice, aceasta ar putea să își îndeplinească obligațiile contractuale de a restitui banii în perioada alocată. Setul de date conține 10000 de înregistrări, scopul fiind de a prezice valoarea atributului ***loan_approval_status*** (clasificare binară). Atributele după care se face clasificarea sunt următoarele:

Atribut	**Tip**	**Semnificatie**	**Valori posibile**
applicant_age	numeric	Vârsta solicitantului	
applicant_income	numeric	Venitul solicitantului	
residential_status	categoric	Statutul rezidențial al solicitantului	'Owner', 'Renter', 'Mortgage', 'Unknown'
job_tenure_years	numeric	Numărul de ani de activitate pe piața muncii a solicitantului.	
loan_purpose	categoric	Scopul împrumutului.	'Study', 'Business 'Personal', 'Health', 'Hom Improvement', 'Deb Consolidation'
loan_rating	categoric	Clasificarea împrumutului.	'Excellent', 'Very Good 'Good', 'Fair', 'Poor', 'Ver Poor', 'Extremely Poor'
loan_amount	numeric	Suma împrumutului.	
loan_rate	numeric	Rata dobânzii la împrumut.	
loan_income_ratio	numeric	Raportul dintre suma împrumutului și venitul solicitantului.	
credit_history_default_status	categoric	Starea implicită a istoricului de credit al solicitantului.	'Yes', 'No'
credit_history_length_years	numeric	Lungimea istoricului de credit al solicitantului în ani.	
credit_history_length_months	numeric	Lungimea istoricului de credit al solicitantului în luni.	
stability_rating	categoric	Stabilitatea financiară a solicitantului de credit.	'A', 'B', 'C', 'D'
loan_approval_status	**ategoric**	**Starea de aprobare a împrumutului (atributul țintă).**	**'Approved', 'Declined'**

3. Cerințe

3.1. Explorarea Datelor (Exploratory Data Analysis) [3p]

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 / ne-informative**

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 exemplelor 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

Afișati 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 atribute 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 atribute

1. **Realizați analize** de **corelație** (a se vedea indicațiile din Anexă: Explorarea Datelor) între **atributele numerice continue**, pentru a identifica dacă există atribute redundante.

2. **Realizați analize** de **corelație** între **atributele categorice,** pentru a identifica dacă existe atribute redundante.

Două atribute 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 valoarea 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 atribute 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 [5p]

Arbori de Decizie [2.5p]

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

În cazul unui arbore de decizie, **nu este obligatorie** conversia variabilelor categorice la valori numerice. Tipul lor trebuie însă avut în vedere atunci când se crează split-ul într-un nod al unui arbore de decizie.

Se cer următoarele două abordări.

1. Antrenați și evaluați un model de arbore de decizie folosind biblioteca scikit-learn. Documentația pentru DecisionTreeClassifier este disponibilă aici. [1p]
2. Implementați un model de arbore de decizie plecând de la codul din laborator [1.5p]
    - **Atenție:** pentru că atributele din setul de date pot fi și numerice, este nevoie să considerați crearea split-ului de atribute numerice, folosind metoda împărțirii binare pe intervale. Găsiți exemple de implementare în acest blog și în acest video explicativ.

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 opțiuni de ponderare a claselor (class_weight) utilă în cazul unui set de date cu clase dezechilibrate

Multi-Layered Perceptron (MLP) [2.5p]

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

Înaintea rulării propriu-zise a algoritmului țineți cont de următoarele:

- Pentru a putea utiliza un MLP, variabilele **categoriale** (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.

Se cer următoarele două abordări:

1. Implementare manuală [1.5p]
   - Pornind de la rezolvarea din laborator, implementați modelul de MLP pentru problema dată.
2. Implementare folosind biblioteca scikit-learn [1p]
   - 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 contine 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 laboratatorului (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 atribute 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 atribute 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 atribute

Analiza corelației între **atribute 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 atribute categorice, corelația între două atribute 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 atribute 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 atribute 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.**