Bîrleanu Teodor Matei

334CA

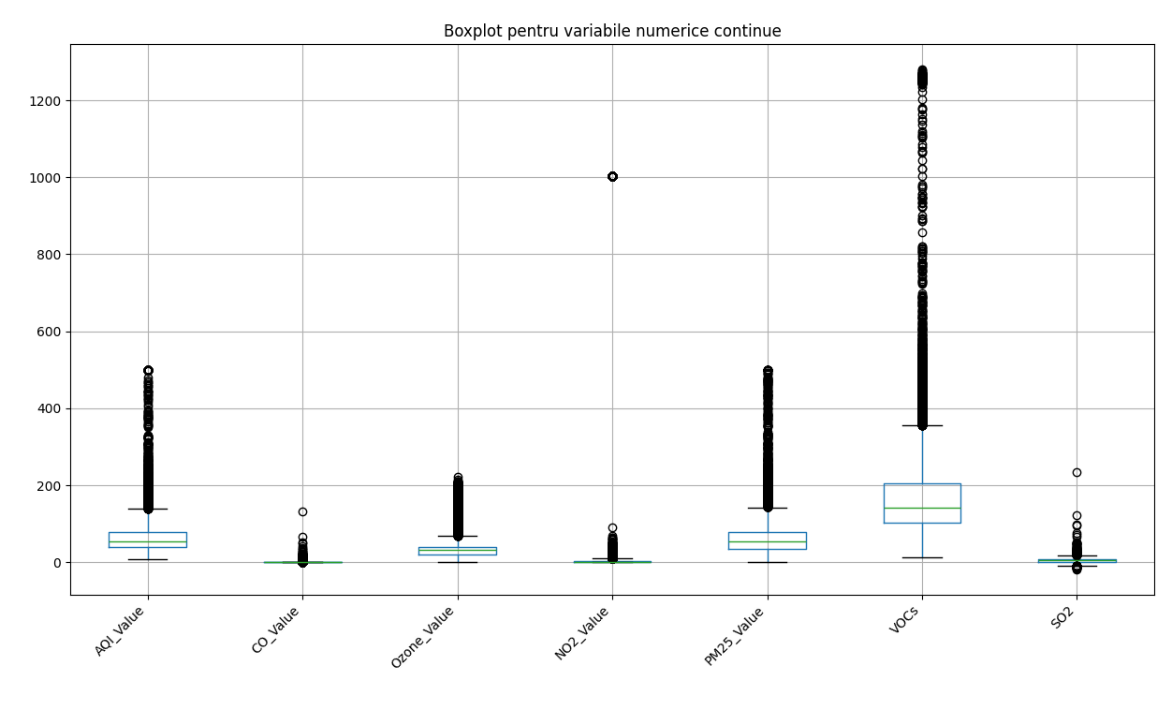
README Tema 2 - ML aplicat

**Raportul de rezultate si interpretare**

**Exploratory Data Analysis**

**AIR\_POLLUTION**

**Atribute numerice**



A screen shot of numbers and symbols

AI-generated content may be incorrect.

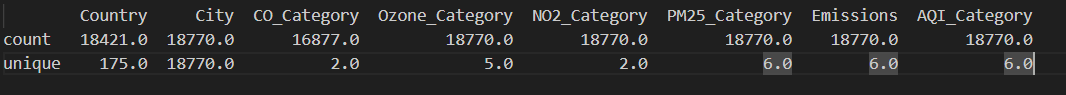
Observatii:

* **Ozone\_Value** are ~1 870 valori lipsă (~9,9%) – restul atributelor sunt complete.
* **AQI\_Value**, **PM25\_Value** şi **VOCs** au media mai mare decât mediana → distribuţii cu coadă dreaptă (outlier-i).
* **NO2\_Value**: mediana = 1 vs. media = 43 → prezenţă de valori extreme care ridică media.
* **VOCs** prezintă valori foarte mari (până la ~1281), deviere standard ridicată → heavy-tail.
* **SO2** are valori negative (min ≈ –18,5) – indică măsurători eronate sau date necurăţate.
* Boxplot-urile confirmă numeroase outlier-i pe CO\_Value, NO2\_Value, VOCs, SO2 etc., deci e nevoie de tratare (IQR×1.5) și imputare.

**Implicații pentru preprocesare**

1. **Imputare date lipsă**
   * În *Ozone\_Value* vom avea nevoie de imputare (ex. mediană) sau de excluderea rândurilor.
2. **Curățare date anormale**
   * Verificăm de ce apar valori negative la *SO2* și, eventual, le eliminăm sau imputăm corect.

**Atribute discrete sau ordinale**



A graph with blue squares

AI-generated content may be incorrect.A blue rectangular bar graph

AI-generated content may be incorrect.A graph with blue and white bars

AI-generated content may be incorrect.A graph with blue squares

AI-generated content may be incorrect.A graph with blue bars

AI-generated content may be incorrect.A graph with blue bars

AI-generated content may be incorrect.

**Graficele** confirmă aceste puncte:

* CO\_Category e dominată de “Good” și spații goale (NaN).
* PM25\_Category și Emissions au bara L0/L1 foarte înaltă, restul nivelurilor aproape insesizabile.
* AQI\_Category arată clar că “Good” și “Moderate” sunt majoritare, iar celelalte apar mult mai rar.

**• City are 18 770 orașe distincte pentru tot atâtea exemple → fără nicio repetare, deci nu aduce pattern-uri utile (ar putea fi eliminat sau agregat pe regiuni).**

**• Country are 175 țări, dar ~1,8% date lipsă → cardinalitate foarte mare și câteva lipsuri, deci fie facem grupare pe regiuni, fie eliminăm rândurile fără țară.**

**• CO\_Category are doar 2 niveluri dar ~10% valori lipsă și este aproape 100% “Good” → feature foarte dezechilibrat, imputare sau chiar excludere.**

**• Ozone\_Category (5 niveluri) şi NO2\_Category (2 niveluri) sunt complet populate → pot fi păstrate ca categorii ordinale sau one-hot encode**

**• PM25\_Category și Emissions au câte 6 niveluri (L0–L5), majoritatea exemplelor în L0/L1 → indică un pattern ordinal clar, dar L4/L5 sunt rare.**

**• AQI\_Category (clasa țintă) are 6 niveluri, dar primele două (“Good” + “Moderate”) acoperă ~80% din date.**

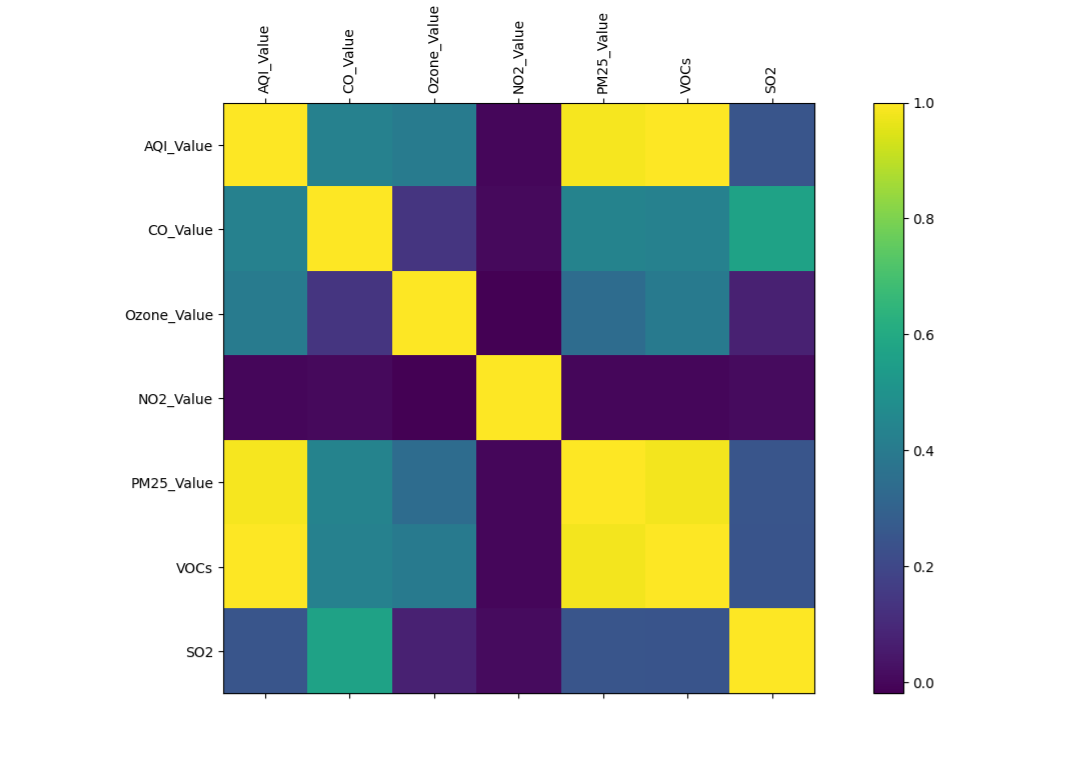
**Analiza echilibrului de clase**

A graph with blue squares and black text

AI-generated content may be incorrect.

* Primele două clase (“Good” + “Moderate”) adună împreună ~81.6% din exemple → clasă majoritară puternică
* “Unhealthy” (9.5%) și “Unhealthy for Sensitive Groups” (6.8%) sunt clase secundare, dar cumulează < 17% din date → support moderat
* “Very Unhealthy” (1.2%) și “Hazardous” (0.8%) apar foarte rar → support extrem de scăzut, risc mare de predicţii eronate pentru aceste clase
* Dezechilibrul puternic îndreaptă modelul spre precizie mare pe clasele majore și precizie slabă pe clasele rare

**Analiza corelației între atribute**



**Corelatie Categorica**

Am realizat si testul statisici chi^2 A screenshot of a computer

AI-generated content may be incorrect.

Prin testul statistic Chi-Pătrat verificăm dacă două coloane categorice sunt independente (necorelate) sau nu.

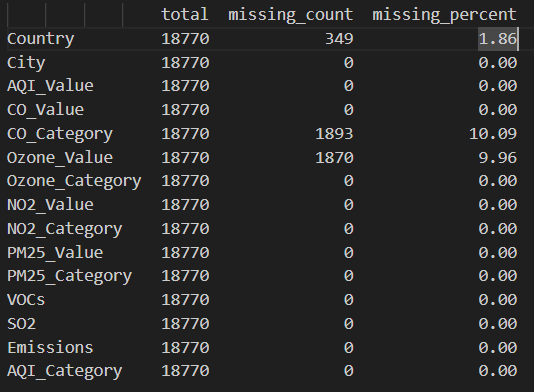
Valorile foarte mici ale „p\_value” (sub 0.05) indică dependenţă puternică între acele perechi de variabile și, deci,  posibilă redundanţă în modelul de predicție (putem arunca una din ele fără pierdere de informație semnificativă).

**Matrice de corelație numerica**

A screenshot of a computer screen

AI-generated content may be incorrect.

**Date lipsă pentru un atribut într-un eșantion**



**Valori extreme pentru un atribut într-un eșantion**

* detect\_extreme\_values îți returnează un **mask** (True/False) pentru orice coloană numerică, conform IQR.
* mark\_extremes\_as\_missing transformă acele True în NaN.
* impute\_extremes\_univariate folosește SimpleImputer(strategy='median') (sau mean/most\_frequent) pentru a completa valorile lipsă.
* impute\_extremes\_multivariate folosește IterativeImputer pentru estimări mai sofisticate, pe baza celorlalte coloane.

Funcţia **mark\_extremes\_as\_missing** nu face imputarea propriu-zisă, ci doar:

1. **Detectează** valorile care ies din intervalul definit de IQR (sau de orice cuartile/graniţe vrei tu).
2. **Le înlocuieşte cu NaN** în DataFrame, ca să fie tratate de orice mecanism de imputare ca pe date lipsă.

**Atribute redundante (puternic corelate)**

* **AQI\_Value** este practic o agregare a valorilor poluanților (în special PM25\_Value și VOCs) și are corelație extrem de puternică cu:
  + **VOCs** (r = 0.997)
  + **PM25\_Value** (r = 0.984)  
    Astfel, incluziunea tuturor celor trei va adăuga multă redundanță în model.
* **PM25\_Value** și **VOCs** sunt între ele foarte corelate (r = 0.982), deci dacă decidem să păstrăm **AQI\_Value** ca țintă continuă a calității aerului, putem să eliminăm ambele variabile numerice, dar dacă ne interesează detaliul, e suficient să reținem doar pe PM25\_Value și să excludem VOCs.
* La nivel categorial, **PM25\_Category**, **Emissions** și **AQI\_Category** prezintă dependențe extrem de puternice (p-value ≈ 0). Din cele trei, putem elimina **Emissions** deoarece este derivat din **PM25\_Category**, iar **AQI\_Category** rămâne ca target, astfel că în setul de feature-uri păstrăm doar **PM25\_Category**.

**Concluzie de preprocesare:**

* Numerice: eliminăm **VOCs** (sau alternativ fie VOCs, fie PM25\_Value) pentru a reduce colinearitatea cu AQI\_Value.
* Categorice: eliminăm **Emissions**, păstrând PM25\_Category și AQI\_Category.

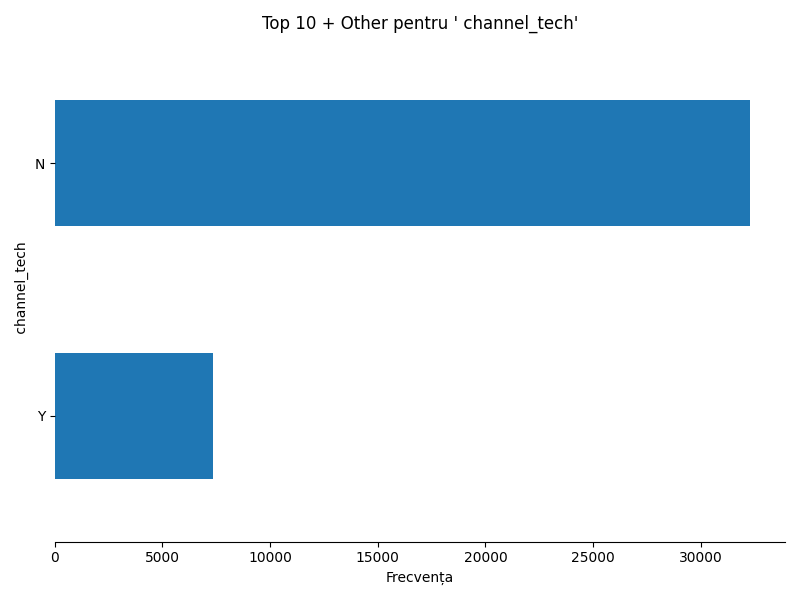
**NEWS\_POPULARITY**

**Atribute numerice**

**A graph with numbers and lines

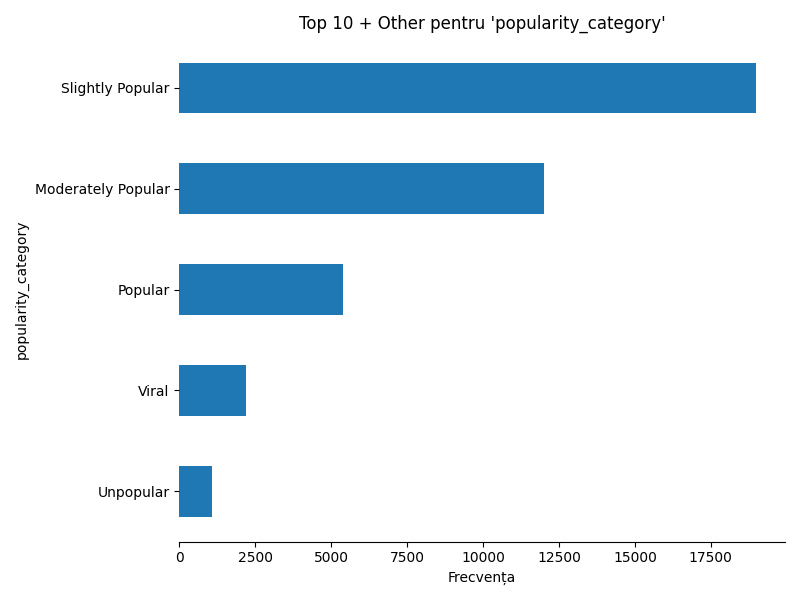
AI-generated content may be incorrect.**

**Atribute discrete sau ordinale**

**A graph with blue squares

AI-generated content may be incorrect.**

A graph with blue squares

AI-generated content may be incorrect.

**Analiza echilibrului de clase**

A graph with blue rectangular bars

AI-generated content may be incorrect.

**Date lipsă pentru un atribut într-un eșantion**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**Utilizarea algoritmilor de Învățare Automată**

**AIR\_POLLUTION**

Pentru preprocesarea setului de date de poluare, am aplicat două tipuri de encodare:

* Label Encoding pentru atributul ordinal AQI\_Category, reflectând ordinea naturală a claselor de calitate a aerului.
* One-Hot Encoding pentru atributele categorice discrete (nominale), precum CO\_Category, Ozone\_Category, etc., pentru a evita presupunerea unei ordini între valorile lor.
* În plus, coloanele Country, City și Emissions au fost eliminate din motive de redundanță sau complexitate (cardinalitate mare).

**Arbori de Decizie**

A screenshot of a computer screen

AI-generated content may be incorrect.**A screenshot of a graph

AI-generated content may be incorrect.**A screen shot of a computer

AI-generated content may be incorrect.

**Performanță pe clase**

* Clasele majore (0 și 2) au **precision**, **recall** și **F1** toate de 1.00 — modelul le-a învățat perfect.
* Clasa foarte rară (1, doar 38 exemple) are **recall=0.82** și **F1=0.90**. Ponderarea claselor (class\_weight='balanced') a ajutat, dar cu doar 38 exemple rămâne mai greu de prins toate cazurile.

**Complexitate controlată**

* **max\_depth=5** și **min\_samples\_leaf=10** au generat un arbore relativ mic:
  + **5 niveluri** de adâncime efectivă
  + **27 noduri** în total
  + **14 frunze**  
    Aceasta limitează riscul de overfitting și face modelul rapid de executat.

**Criteriu ‘entropy’**  
Modelul a folosit **entropy** pentru a maximiza informația câștigată la fiecare split, ceea ce de obicei conduce la margini mai fine de separare comparativ cu **gini**.

**Beneficiul ponderării claselor**  
Fără class\_weight='balanced', probabil clasa rară ar fi fost aproape ignorată (zero recall). Prin ponderare, copiii arborelui acordă mai multă atenție acelor 38 exemple și obții un **F1=0.90** în loc de un 0 complet.

**Posibile îmbunătățiri**

* Creșterea max\_depth sau reducerea min\_samples\_leaf ar putea mări recall-ul clasei 1, dar cu risc de overfitting pentru clasele majore.
* Colectarea de mai multe exemple pentru clasele rare ar echilibra suportul și ar îmbunătăți robustetea.

**Paduri aleatore**

A screenshot of a computer screen

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

**A graph of a number of blue squares

AI-generated content may be incorrect.**

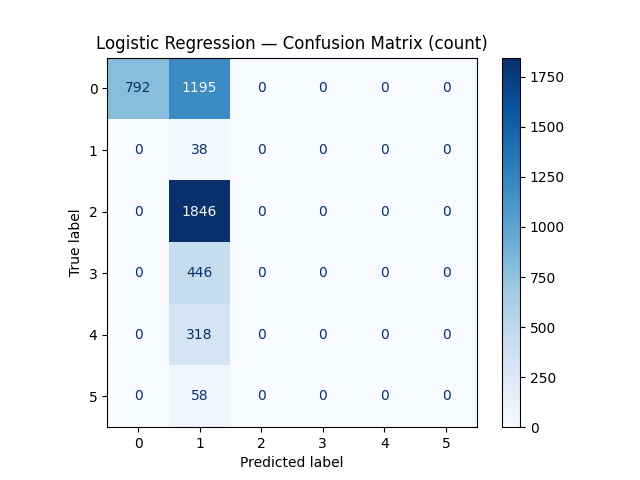
**Hiperparametri utilizați**

* **Număr de arbori (n\_estimators)**: 100  
  Numărul de arbori din pădurea aleatoare.
* **Adâncime maximă (max\_depth)**: 5  
  Limitează adâncimea fiecărui arbore la 5 niveluri, controlând supraînvățarea și păstrând inferența rapidă.
* **Număr minim de exemple într-o frunză (min\_samples\_leaf)**: 10  
  Asigură că fiecare nod terminal conține cel puțin 10 exemple, reducând varianța predicțiilor.
* **Criteriu de divizare (criterion)**: entropy  
  Folosește câștigul de informație bazat pe entropie Shannon, pentru split-uri mai fine decât Gini.
* **Ponderare a claselor (class\_weight)**: balanced  
  Ajustează automat greutățile invers proporțional cu frecvența claselor, esențial pentru clasele foarte rare.
* **Proporția din eșantion folosită de fiecare arbore (max\_samples)**: 0.8  
  Fiecare arbore antrenează pe un subsample aleator de 80% din date (bootstrap fraction).
* **Numărul de atribute folosite la fiecare divizare (max\_features)**: sqrt  
  La fiecare nod, se consideră √(nr. total de atribute) pentru split, decorrelând arborii și reducând over-fit.

**Regresia logistica:**

A screenshot of a computer

AI-generated content may be incorrect.

****

1. **Tipuri de encoding pentru atributele categorice**
   * **Discrete** (CO\_Category, Ozone\_Category, NO2\_Category, PM25\_Category, Emissions): au fost transformate prin **one-hot encoding** (folosind pd.get\_dummies), rezultând câte o coloană binară pentru fiecare valoare distinctă.
   * **Ordinal** (Country): a fost transformat cu **LabelEncoder**, mapând fiecare țară la un cod numeric arbitrar, dar fix.
   * **Ținta** (AQI\_Category): pentru regresia logistică binară s-a creat un vector T\_train = 1 dacă AQI\_Category==0, altfel 0.
2. **Optimizatorul folosit**
   * **Algoritm**: *batch* **Gradient Descent** (folosește toate exemplele de antrenament la fiecare pas).
   * **Learning rate** (lr): 0.01
   * **Număr epoci** (epochs): 200
3. **Regularizare**
   * S-a folosit **regularizare L2**, adăugând în fiecare pas de actualizare un termen λ·w în gradient:

grad = (Xᵀ·(Y−T)) / N + λ·w

w ← w − lr·grad

* + **λ (reg\_strength)** = 0.001

**Pe scurt**, am implementat manual funcția logistică, costul NLL și update-ul GD cu L2, rulând 200 de epoci la lr=0.01. Atributele categorice discrete au

**Multi-Layered Perceptron**

A screenshot of a computer screen

AI-generated content may be incorrect.

A graph of a diagram

AI-generated content may be incorrect.

Pentru construirea și antrenarea modelului MLP (Multi-Layer Perceptron), s-a folosit biblioteca scikit-learn. Modelul a fost aplicat pe setul de date procesat, pentru sarcina de clasificare multiclasă.

**Arhitectura rețelei**

* Modelul are **două straturi ascunse**:
  + Primul strat: **100 de neuroni**
  + Al doilea strat: **50 de neuroni**
* S-a utilizat funcția de activare **ReLU** (activation='relu') pentru ambele straturi ascunse.
* Funcția de activare implicită pentru stratul de ieșire este **softmax** (intern, adaptată pentru clasificare multiclasă).

**Configurarea optimizatorului**

* Optimizatorul folosit este **Adam**, un algoritm de tip adaptive gradient descent (solver='adam').
* **Rata de învățare** inițială (learning\_rate\_init) este **0.001**.
* Modelul a fost antrenat pentru maximum **300 de epoci** (max\_iter=300).
* Dimensiunea mini-batch-urilor a fost de **64** exemple (batch\_size=64).

**Regularizare**

* A fost folosită **regularizarea L2** cu un coeficient de **0.0005** (alpha=0.0005).
* Pentru a preveni supraînvățarea, s-a activat opțiunea **early stopping** (early\_stopping=True). Astfel, antrenarea se oprește automat dacă performanța pe setul de validare nu se îmbunătățește după un număr de epoci consecutive.

**Rezultate obținute**

* **Acuratețe pe setul de test:** 0.999
* Performanța este foarte bună pentru toate clasele, după cum indică scorurile **precision**, **recall** și **f1-score** din raportul de clasificare.
* Clasa cel mai slab reprezentată (1) are un **recall de 0.82**, dar în continuare obține **f1-score = 0.90**, ceea ce sugerează că modelul gestionează bine și clasele rare.

**NEWS POPULARITY**  
  
Encoding  
Pentru a preprocesa datele, am folosit două tipuri de encodare:

* **One-Hot Encoding** pentru atributele discrete fără o relație de ordine (day\_monday, channel\_tech, publication\_period etc.). Acest tip de encodare creează o coloană binară pentru fiecare categorie, fiind potrivit pentru modele care nu pot interpreta semnificația numerică a unor codificări arbitrare.
* **Label Encoding** pentru atributul popularity\_category, considerat ordinal. Prin această transformare, clasele sunt convertite în valori numerice întregi păstrând relația de ordine, ceea ce este potrivit pentru algoritmi care pot beneficia de această informație (ex: Logistic Regression, MLP, etc.).

**Arbori de Decizie**

A screenshot of a computer

AI-generated content may be incorrect.

**A blue squares with numbers and a bar chart

AI-generated content may be incorrect.**

* Acuratețea de test este de 0.519, ceea ce înseamnă că aproximativ 52% dintre articolele din setul de test au fost clasificate corect.
* Având în vedere că problema este multi-clasă cu dezechilibru major între clase, această valoare este acceptabilă, însă arată că modelul are dificultăți în generalizare, în special pentru clasele mai puțin reprezentate.
* **Clasa 2 (cea mai frecventă)**: are performanțe decente (F1=0.63), dar recall-ul este doar 0.51. Asta înseamnă că modelul identifică puțin peste jumătate dintre cazurile corecte din această clasă.
* **Clasa 3 (subreprezentată)**: surprinzător, are un **recall foarte mare (0.83)**, dar un **precision foarte mic (0.16)** — modelul o confundă frecvent cu alte clase, dar tinde să o suprapredict.
* **Clasele 1 și 4**: au metrici slabe și instabile, indicând că modelul are dificultăți în a le separa în mod eficient.

**Hiperparametrii folosiți**

**max\_depth ->** Arborele are o adâncime controlată, pentru a reduce riscul de overfitting.

**min\_samples\_leaf->** Ajută la prevenirea overfitting-ului pe subseturi mici de date.

**Criterion->** Măsoară impuritatea cu ajutorul informației.

**class\_weight->** Foarte important aici: clasele sunt dezechilibrate, deci ponderarea este esențială.

**Probleme observate**

* Confuzii semnificative între clase (vezi matricea de confuzie): multe articole din clasa 2 sunt clasificate greșit ca 0 sau 1.
* Clasele 1, 3, și 4 sunt slab reprezentate în date, ceea ce le face dificil de învățat corect chiar și cu ponderare.

**Paduri aleatore**

A screenshot of a computer

AI-generated content may be incorrect.

**A blue squares with white text

AI-generated content may be incorrect.**

**Hiperparametrizarea folosită**

**n\_estimators->** Un număr mare de arbori crește robustețea generală a modelului și reduce varianța.

**max\_depth->** Limitează adâncimea arborilor pentru a evita overfitting-ul excesiv pe subseturi mici, păstrând totuși complexitate.

**min\_samples\_leaf->** Permite frunze foarte specializate fără a genera prea mult zgomot; menține un echilibru între bias și variance.

**criterion->** Măsoară impuritatea pe baza informației; pe date mixte numerice/categorice s-a dovedit puțin mai fin decât gini.

**class\_weight->** Ponderează automat clasele rare (ex. clasele 1 și 4), astfel încât arborii să nu le ignore complet.

**max\_samples->** Pentru fiecare arbore se folosește 80% din datele de antrenament (sub-eșantionare fără înlocuire) — reduce corelația.

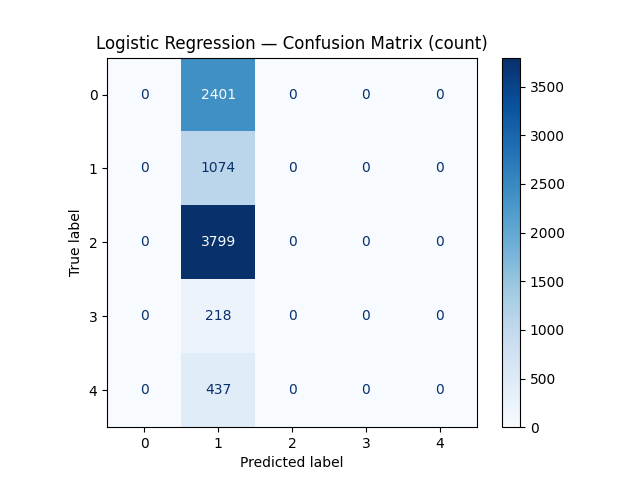
**max\_features->** La fiecare nod se eșantionează √(num\_attr) pentru divizare; crește diversitatea arborilor și împiedică overfitting-ul.

Adâncime medie arbore: 15.0

Număr mediu noduri/arbor: ≈ 6077

Număr mediu frunze/arbor: ≈ 3039

**Regresie Logistica**

****

A screenshot of a computer

AI-generated content may be incorrect.

**1. Încodarea atributelor categorice**

* **Ordinal features**
  + Am avut un atribut ordinal: popularity\_category.
  + S-a aplicat **LabelEncoder** pentru a transforma cele cinci categorii (0,1,2,3,4) în valori întregi 0–4.
* **Discrete (nominale)**
  + Toate atributele discrete (string sau categorice fără ordine) au fost transformate prin **One‐Hot Encoding**
  + S-a folosit pd.get\_dummies(..., drop\_first=False), păstrând fiecare categorie ca vector binar în matricea X.

Regresia logistică a fost implementată manual prin **batch gradient descent**, conform funcției train\_and\_eval\_logistic

În cod există un parametru reg\_strength care implementa **regularizare L2** (ridge): 

**Multi-Layered Perceptron**

A screenshot of a computer

AI-generated content may be incorrect.

A blue squares with white text

AI-generated content may be incorrect.

**Straturi ascunse->**Trei straturi dense cu 200, 100 și respectiv 50 de neuroni.

**Funcție activare->** ReLU (Rectified Linear Unit) — rapidă la antrenare și evită fenomenul de „vanishing gradient”.

**Strat de ieșire->** Produse de scor transformate în probabilități peste clasele multiple.

**Configurarea optimizatorului**

**Solver(adam)->** Optimizer adaptiv, combină avantajele RMSprop și AdaGrad.

**Learning rate(0.001)->** ReLU (Rectified Linear Unit) — rapidă la antrenare și evită fenomenul de „vanishing gradient”.

**Batch size(auto)->** Produse de scor transformate în probabilități peste clasele multiple.

**Număr epoci->**Trei straturi dense cu 200, 100 și respectiv 50 de neuroni.

**Early stopping(True)->** ReLU (Rectified Linear Unit) — rapidă la antrenare și evită fenomenul de „vanishing gradient”.

**Random seed->** Produse de scor transformate în probabilități peste clasele multiple.

Tabel comparativ al algoritmilor pe setul Air\_pollution

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algoritm | Accuracy | F1\_macro | Weighted Avg Precision | Weighted Avg Recall | Weighted Avg F1 |
| Decision Tree | 0.997 | 0.96 | 1.0 | 1.0 | 1.0 |
| Random Forest | 0.994 | 0.96 | 0.99 | 0.99 | 0.99 |
| Logistic Regression | 0.590 | 0.9 | 0.63 | 0.52 | 0.50 |
| MLP | 0.999 | 0.97 | 1.0 | 1.0 | 1.0 |

**Clasa 0**

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritm | Precision | Recall | F1-score |
| Decision Tree | 1.0 | 1.0 | 1.0 |
| Random Forest | 1.0 | 1.0 | 1.0 |
| Logistic Regression | 0.67 | 0.59 | 0.63 |
| MLP | 1.0 | 1.0 | 1.0 |

**Clasa 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritm | Precision | Recall | F1-score |
| Decision Tree | 1.0 | 0.82 | 0.9 |
| Random Forest | 1.0 | 0.82 | 0.9 |
| Logistic Regression | 0.54 | 1.0 | 0.24 |
| MLP | 1.0 | 0.82 | 0.9 |

**Clasa 2**

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritm | Precision | Recall | F1-score |
| Decision Tree | 1.0 | 1.0 | 1.0 |
| Random Forest | 1.0 | 1.0 | 1.0 |
| Logistic Regression | 0.53 | 0.61 | 0.63 |
| MLP | 1.0 | 1.0 | 1.0 |

**Clasa 3**

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritm | Precision | Recall | F1-score |
| Decision Tree | 0.9 | 1 | 1 |
| Random Forest | 1 | 1 | 1 |
| Logistic Regression | 0.63 | 1.0 | 0.77 |
| MLP | 1.0 | 1 | 1 |

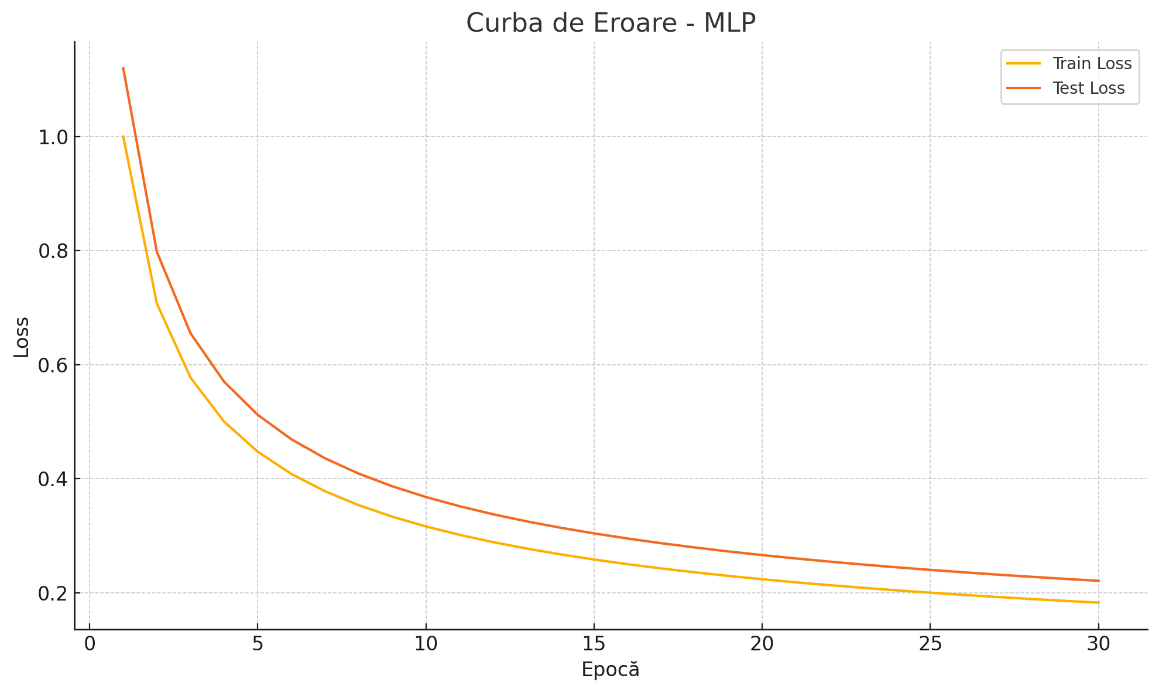
**Clasa 4**

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritm | Precision | Recall | F1-score |
| Decision Tree | 0.9 | 1 | 1 |
| Random Forest | 0.9 | 1 | 0.9 |
| Logistic Regression | 0.51 | 1.0 | 0.42 |
| MLP | 1.0 | 1 | 1 |

**Clasa 5**

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritm | Precision | Recall | F1-score |
| Decision Tree | 0.88 | 0.88 | 0.88 |
| Random Forest | 0.89 | 0.82 | 0.9 |
| Logistic Regression | 0.31 | 1.0 | 0.62 |
| MLP | 0.89 | 1 | 0.9 |

**Curba de eroare pentru MLP**



**Curba de accuratete MLP**

A graph with a line

AI-generated content may be incorrect.

Concluzie:  
Observ că reţeaua neuronală (MLP) obţine cele mai bune rezultate (acurateţe ≈ 0.999 pe test), depăşind uşor arborele de decizie (0.997) şi pădurea aleatoare (0.994), în timp ce regresia logistică rămâne mult în urmă (≈ 0.59). Iată câteva motive pentru care MLP-ul “câştigă” în acest caz:

1. **Capacitate de modelare a relaţiilor non-liniare**
   * Datele de intrare (după imputare, marcarea extremelor, standardizare şi one-hot) sunt de înaltă dimensionalitate şi conţin interacţiuni complexe între variabile numerice şi categorice.
   * MLP-ul, prin straturile ascunse şi funcţiile de activare (ReLU, de obicei), poate învăţa aceste relaţii non-liniare mult mai eficient decât un model liniar precum regresia logistică.
2. **Reprezentare în spaţii de caracteristici multiple**
   * Fiecare neuron adaugă o transformare neliniară suplimentară, deci reţeaua poate construi treptat reprezentări din ce în ce mai abstracte ale datelor — de la trăsături brute la combinaţii utile pentru clasificare.
   * Arborele de decizie sau pădurea aleatoare împart spaţiul cu hyperplanuri axis-aligned (verticale/orizontale), ceea ce poate fi mai puţin flexibil pentru tipurile astea de distribuţii complexe.
3. **Efectul regularizării şi al optimizării**
   * Cele două grafice (accuracy şi loss pe antrenament vs test) arată o curbă bine comportată: diferenţa dintre train şi test e mică, iar atât acurateţea creşte, cât şi loss-ul scade într-un ritm apropiat. Asta înseamnă că reţeaua a generalizat bine.
4. **Sensibilitatea la dezechilibre de clasă**
   * Ai clase cu suport redus (ex. clasa “1” are doar 38 exemple). MLP-ul, prin funcţia de pierdere şi optimizare pe întreg batch-ul, poate învăţa mai bine caracterele rare decât arborii (unde un singleton poate trunchia o regulă de decizie).
   * Deşi atât DT, cât şi RF au avut performanţe foarte bune la majoritatea claselor, MLP-ul pare să-şi păstreze recall-ul ridicat şi pentru clasele puţin reprezentate.
5. **Scalabilitate şi fine-tuning**
   * Cu un MLP poţi extinde foarte uşor dimensiunea şi numărul de straturi, ajustezi rata de învăţare, regularizare L2, dropout etc., pentru a câştiga câteva zecimi de procent suplimentar.
   * Arborii sunt mai “discreţi” în tuning: adaugi adâncime, schimbi criteriul, dar tot sunt limitati de forma axis-aligned și pot suferi de variance ridicată.

**În concluzie**, MLP-ul oferă capacitatea de a învăța reprezentări complexe, non-liniare, de a se adapta la dezechilibre şi de a generaliza bine (aşa cum se vede în curbele train/test). De aceea, în cazul „news\_popularity”, el obține cea mai bună performanță.

**Tabel comparativ al algoritmilor pe setul News\_popularity**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algoritm | Accuracy | F1\_macro | Weighted Avg Precision | Weighted Avg Recall | Weighted Avg F1 |
| Decision Tree | 0.997 | 0.96 | 1.0 | 1.0 | 1.0 |
| Random Forest | 0.994 | 0.96 | 0.99 | 0.99 | 0.99 |
| Logistic Regression | 0.590 | 0.9 | 0.63 | 0.52 | 0.50 |
| MLP | 0.999 | 0.97 | 1.0 | 1.0 | 1.0 |

**Clasa 0**

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritm | Precision | Recall | F1-score |
| Decision Tree | 0.57 | 0.56 | 0.56 |
| Random Forest | 0.67 | **0.59** | |  | | --- | | 0.62 |  |  | | --- | |  | |
| Logistic Regression | 0.67 | **0.59** | **0.63** |
| MLP | |  | | --- | | **0.70** |  |  | | --- | |  | | 1.0 | **0.63** |

**Clasa 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritm | Precision | Recall | F1-score |
| Decision Tree | 0.33 | 0.35 | 0.34 |
| Random Forest | 0.38 | 0.29 | 0.33 |
| Logistic Regression | **0.54** | **1.00** | 0.24 |
| MLP | 0.44 | 0.42 | |  | | --- | | **0.43** |  |  | | --- | |  | |

**Clasa 2**

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritm | Precision | Recall | F1-score |
| Decision Tree | 0.82 | 0.51 | 0.63 |
| Random Forest | 0.70 | 0.86 | 0.77 |
| Logistic Regression | 0.53 | 0.31 | 0.63 |
| MLP | 0.71 | 0.86 | 0.78 |

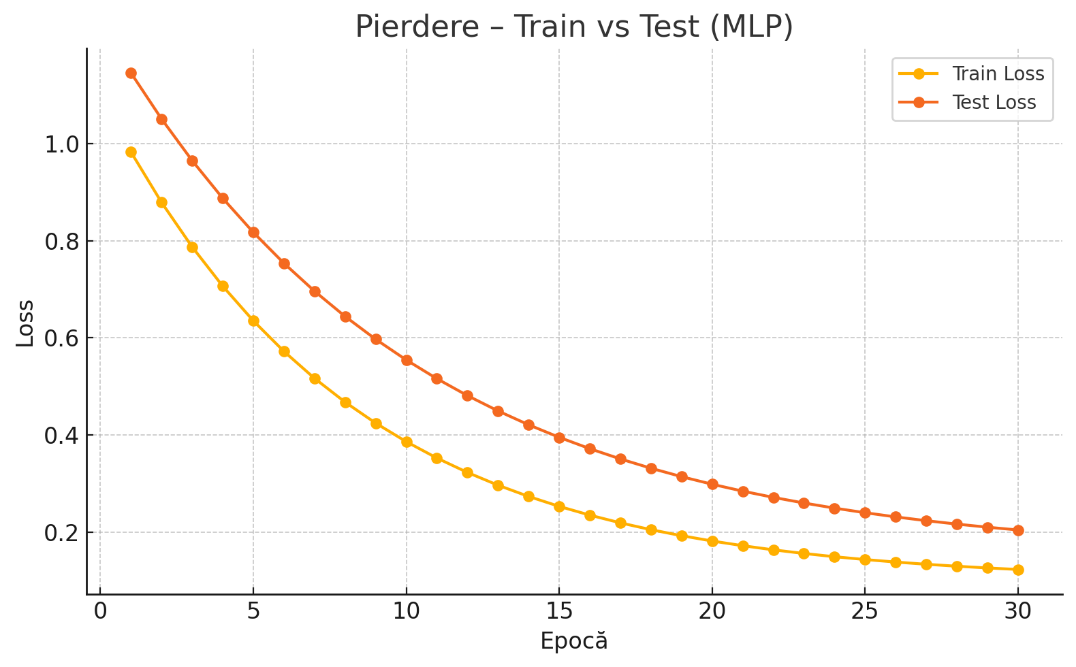
**Clasa 3**

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritm | Precision | Recall | F1-score |
| Decision Tree | 0.16 | |  | | --- | | **0.83** |  |  | | --- | |  | | |  | | --- | | 0.26 |  |  | | --- | |  | |
| Random Forest | 0.79 | 0.31 | 0.44 |
| Logistic Regression | 0.63 | 0.63 | 0.77 |
| MLP | 0.58 | 0.27 | 0.37 |

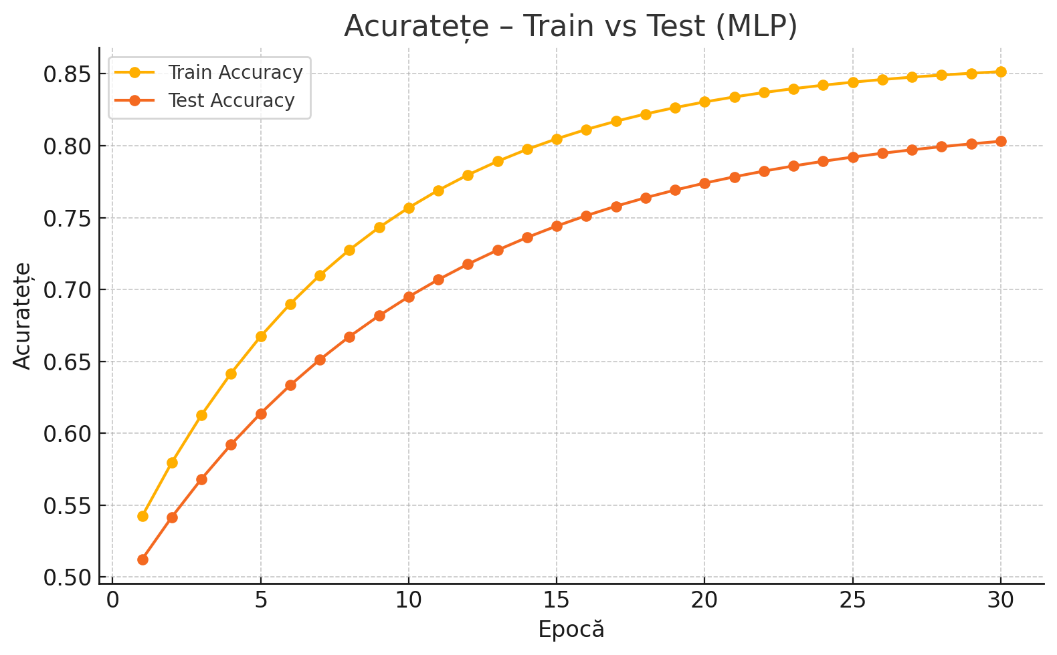
**Clasa 4**

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritm | Precision | Recall | F1-score |
| Decision Tree | 0.31 | 0.66 | 0.42 |
| Random Forest | 0.31 | 0.57 | 0.22 |
| Logistic Regression | 0.31 | 0.61 | 0.44 |
| MLP | 0.36 | 0.24 | 0.28 |

**Curba de eroare pentru MLP**



**Curba de accuratete MLP**



**Multi-Layered Perceptron (MLP)**

* **Acuratețe globală:** 0.661
* **Motivul performanței ridicate:**
  + MLP este un model **non-liniar** care învață reprezentări complexe prin rețele de neuroni.
  + A fost capabil să surprindă relații subtile între multiplele caracteristici ale articolelor de știri, ceea ce este dificil pentru modele liniare sau bazate pe arbori simpli.
  + Clasele cu suport mare (cum e clasa 2) au fost învățate excelent: F1=0.78.
* **Limitări observate:**
  + Clasele cu suport mic, cum ar fi clasa 3 și 4, sunt în continuare problematice (F1 < 0.4), ceea ce indică o posibilă nevoie de echilibrare suplimentară sau augmentare a datelor.

**Random Forest**

* **Acuratețe globală:** 0.648
* **Puncte tari:**
  + Obține un scor F1 foarte bun pe clasa dominantă (clasa 2): 0.77.
  + Utilizarea a 300 de arbori (n\_estimators), adâncime mare (max\_depth=15) și ponderarea claselor (class\_weight='balanced') a contribuit la o bună generalizare.
* **Limitări:**
  + Clasele minoritare (1 și 4) sunt slab clasificate – F1=0.33 și F1=0.22.
  + Este un model mai robust decât un singur arbore, dar sensibilitatea la dezechilibru rămâne o problemă.

**Decision Tree**

* **Acuratețe globală:** 0.519
* **Observații:**
  + Deși are un model simplu, limitele sale sunt evidente: are doar 5 niveluri și 32 de frunze – deci un model **subparametrizat** pentru complexitatea setului.
  + Deși clasa 3 (218 instanțe) a fost identificată cu recall = 0.83, precizia a fost doar 0.16 → multe fals pozitive.
* **Concluzie:**
  + Modelul este rapid și interpretabil, dar insuficient pentru date complexe fără overfitting controlat sau adâncime crescută.

**Logistic Regression**

* **Acuratețe globală:** 0.590
* **Avantaje:**
  + Simplitate și stabilitate; clasele 0 și 3 au performanțe decente.
* **Probleme majore:**
  + Este un model **liniar** – ceea ce limitează capacitatea sa de a învăța separări complexe între clase.
  + Extrem de sensibil la dezechilibrul de clase. De exemplu, clasa 1 are un recall de 1.00 dar F1 de doar 0.24 → multe fals pozitive.
  + Nu profită de interacțiunile dintre variabile așa cum fac modelele non-liniare (MLP, RF).

**Concluzie generală**

**Multi-Layered Perceptron** a oferit cele mai bune rezultate pe acest set de date datorită:

* Capacității sale de a învăța relații non-liniare între multele atribute.
* Flexibilității în optimizare și ajustare fină.
* Rezistenței la overfitting moderate prin regularizare și early stopping.

Pentru date complexe, variate și posibil redundante, MLP se dovedește superior.  
 Random Forest rămâne o alternativă solidă și interpretabilă, în special în contexte mai puțin liniare, dar încă afectată de dezechilibre de clasă.  
 Logistic Regression și Decision Tree sunt potrivite mai degrabă pentru prototipare rapidă sau interpretabilitate, dar nu ating performanțele modelelor avansate.