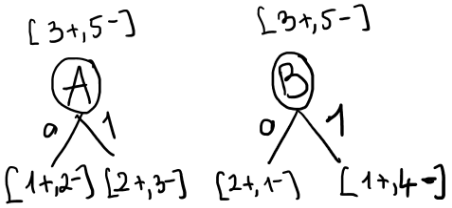
Disciplina: Învățare automată Timp: 1 oră și 10 minute Punctaj maxim: 4p	**TEST**	**Arbori de decizie (2)** **Clasificare bayesiană (1)**

1. Fie codul următor:

```
import pandas as pd
features = ['A', 'B', 'C', 'Y']
data = pd.DataFrame([
  (0,0,0,0),
  (0,0,1,1),
  (0,1,0,1),
  (0,1,1,0),
  (1,0,0,0),
  (1,0,1,1),
  (1,1,0,0),
  (1,1,1,1)
],
columns=features)

from sklearn import tree
X = data[['A', 'B', 'C']]
y = data['Y']
dt = tree.DecisionTreeClassifier(criterion='entropy').fit(X,y)
error = 1-dt.score(X,y)
print(error)
```

   a. (0.1p) Ce va afișa codul? Justificați, dar **NU faceți arborele!**
   b. (0.1p) Răspundeți la întrebarea de la a. pentru cazul în care în tabel mai adăugăm 2 linii:
   (A=0,B=0,C=0,Y=1),
   (A=0,B=0,C=1,Y=0). **NU faceți arborele!**
2. (0.1p) Comparați H[2+,3-] cu H[10+,5-]. **Justificați**.
3. (0.1p) Comparați entropiile condiționale medii asociate următorilor compași de decizie. **Justificați** folosind raționamente calitative [sau calcule...].



4. În cadrul algoritmului **ID3**, fie următorul tabel în care doar atributele C și D sunt **continue**:

A	B	C (continuu)	D (continuu)	Y (output)
0	0	1	1	0
0	1	2	2	1
1	0	1	3	1
1	1	5	0	1
0	1	5	6	0

   a. (0.2p) Câte praguri distincte trebuie să considerăm pentru C atunci când căutăm atributul (optim) care trebuie pus în rădăcină? Dar pentru D? **Justificați**.
   b. (0.1p) Care sunt nodurile ce trebuie luate în calcul atunci când se caută nodul rădăcină? **NU mai scrieți partițiile [a+,b-], ci doar numele din cercuri.**
   c. Presupunem că valorile entropiilor condiționale medii dintre atributul de ieșire și fiecare atribut candidat pentru nodul rădăcină sunt: 0.8 pentru A, 0.7 pentru B, 0.9 pentru restul.
      i. (0.1p) Ce atribut/nod va fi ales ca rădăcină? **Justificați**.
      ii. (0.1p) Care sunt nodurile ce trebuie luate în calcul atunci când se caută nodul corespunzător ramurii *Rădăcină* == 0? **NU mai scrieți partițiile [a+,b-], ci doar numele din cercuri.**
   d. (0.1p) Fie tabelul în care apar doar coloanele C și Y. Aplicăm ID3 **doar** pe aceste date. Cum va fi clasificată instanța (C=0)? **Justificați fără a face arborele!**

5. (0.3p) Ce va afișa următorul cod? Justificați!

```
import pandas as pd
features = ['A', 'B', 'Y']
data = pd.DataFrame([
  (0,0,0),
  (1,0,1),
  (1,1,1)
],
columns=features)

X = data[['A', 'B']]
y = data['Y']

from sklearn.model_selection import cross_val_score
from statistics import mean
from sklearn.model_selection import LeaveOneOut
from sklearn import tree

classifier = tree.DecisionTreeClassifier(criterion='entropy')
loo = LeaveOneOut()
scores = cross_val_score(classifier, X, y, cv=loo)
error = 1-mean(scores)
print(error)
```

6. Fie următorul set de date, unde A,B,Y sunt discrete:

A	B	Y (output)
0	0	1
1	1	1
2	0	1
1	1	0

   a. (0.1p) Estimați P(A = 0|Y=1) în sensul verosimilității maxime (MLE).
   b. (0.1p) Estimați P(A = 0|Y=1) folosind regula *add-one* de netezire a lui Laplace.

(VEZI PAGINA URMĂTOARE)

7. Fie următoarea celulă din Google Colab:

```
import pandas as pd
import numpy as np
from sklearn.naive_bayes import CategoricalNB
data = pd.DataFrame([
  (1,0,0),
  (1,1,1),
  (0,1,1),
  (1,1,2),
  (2,1,2)
],
columns=["A","B","Y"])

X = data[["A","B"]]
y = data["Y"]

cl = CategoricalNB(alpha=0).fit(X,y)

print(np.exp(cl.class_log_prior_))
np.set_printoptions(suppress=True)
print([np.exp(x) for x in cl.feature_log_prob_])
```

```
[0.2 0.4 0.4]
[array([[0. , 1. , 0. ],
       [0.5, 0.5, 0. ],
       [0. , 0.5, 0.5]]), array([[1., 0.],
       [0., 1.],
       [0., 1.]])]
```

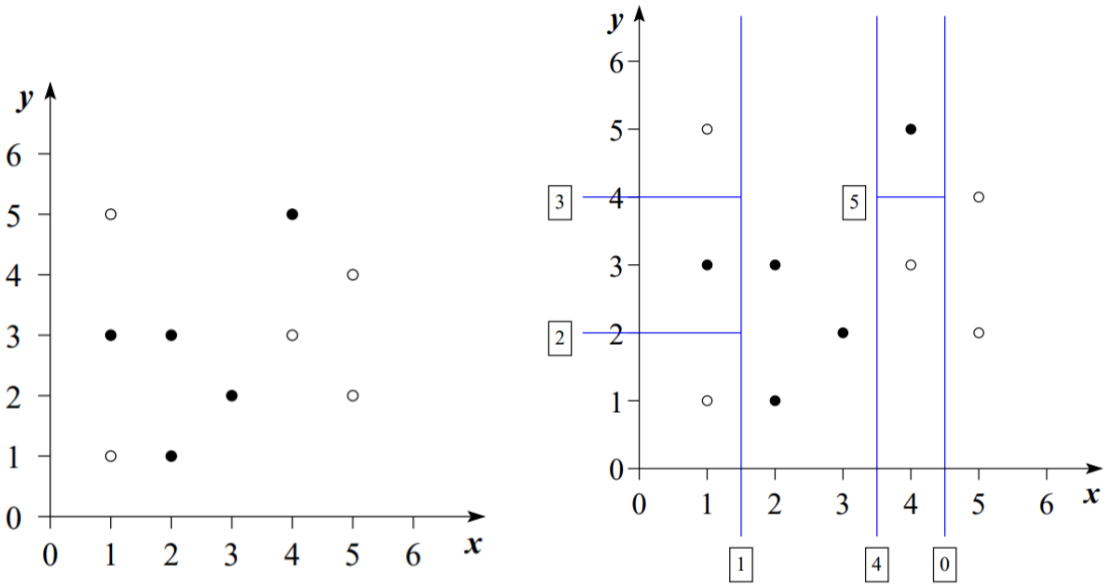
   a. (0.7p) Explicați în detaliu cum s-a ajuns la outputul respectiv.
   b. (0.1p) Care este numărul minimal de parametri pe care a trebuit să-l estimeze CategoricalNB în codul de mai sus [pentru a face apoi predicții pe un set oarecare de instanțe de test]?
   c. Fie intrarea (A=0,B=0). La următoarele trei exerciții ne referim la această intrare și la algoritmul **Bayes OPTIMAL (NU NAIV) (NU folosiți regula lui Laplace.)**:
      i. (0.2p) Derivați regula de decizie a algoritmului pentru intrarea dată și pentru datele de antrenament din variabila numită *data* din codul de mai sus. Scrieți toți pașii intermediari.
      ii. (0.2p) Care va fi decizia algoritmului pentru intrarea dată? Justificați.
      iii. (0.1p) Precizați cu ce probabilitate este luată această decizie.

8. În cadrul unui experiment de ML un student a împărțit setul de date de care dispunea în 3: antrenare, validare și testare. Studentul trebuie să aleagă între 3 modele: ID3, ID3 cu index Gini, Bayes Naiv și calculează următoarele erori la antrenare și validare după ce a antrenat pe setul de antrenare.

Model	Eroare la antrenare	Eroare la validare
Bayes Naiv	0.05 = 5%	0.06 = 6%
ID3	0.05 = 5%	0.4 = 40%
ID3 cu index Gini	0.54 = 54%	0.55 = 55%

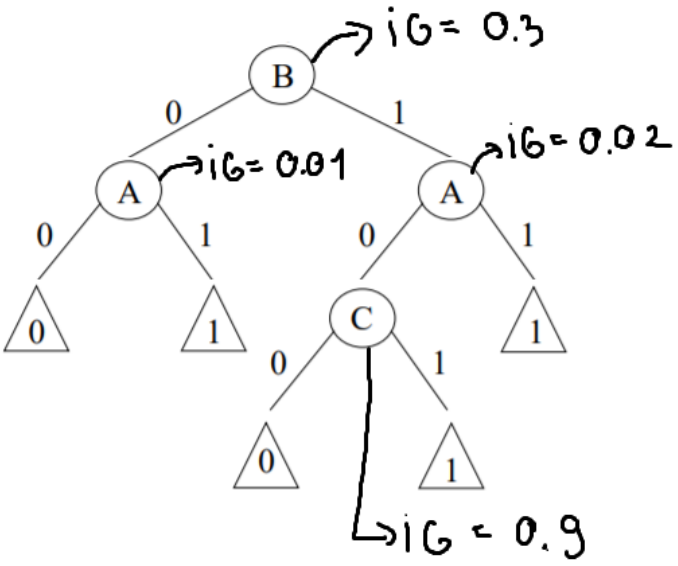
   a. (0.1p) Ce model va alege din cele 3? De ce?
   b. (0.2p) Ce probleme (underfitting/overfitting) prezintă celelalte două modele?
   c. (0.1p) Pentru modelul cu overfitting ce s-ar putea face pentru a scăpa de această problemă?

9. Pe setul de date de mai jos se aplică **ID3 (cu atribute continue)**. În partea dreaptă, am vizualizat toate spliturile realizate de algoritmul ID3 pentru a învăța complet arborele de decizie. Ordinea spliturilor este dată de un indice într-un dreptunghi; numerotarea începe de la 0.



a. (0.1p) Câte frunze va avea arborele produs de algoritmul ID3?
b. (0.2p) Desenați arborele produs de algoritmul ID3. NU mai scrieți partițiile [a+,b-].
c. (0.2p) Desenați granițele de decizie produse de algoritmul ID3 pe acest set de date.

10. Fie următorul arbore produs de ID3 pe un set de date. Pentru fiecare nod este trecut și IG-ul corespunzător calculat în cadrul algoritmului.



a. (0.1p) Pentru un astfel de arbore de decizie, o strategie simplă de trunchiere (engl., pruning) în vederea contracarării fenomenului de “overfitting” constă în a parcurge arborele de sus în jos, începând deci cu nodul-rădăcină şi identificând fiecare nod de test pentru care câştigul de informaţie are o valoare mai mică decât o valoare pozitivă, mică, fixată de la început, ε. Orice astfel de nod de test este imediat înlocuit — împreună cu subarborele corespunzător lui — cu un nod de decizie, conform etichetei majoritare a instanţelor asignate nodului de test **(în acest exercițiu veți lăsa nodul de decizie, adică triunghiul, gol, pentru că nu aveți informații suficiente)**. Această strategie se numeşte “top-down pruning”.

Care este arborele de decizie obţinut aplicând această strategie pe arborele de mai sus, dacă se consideră ε = 0.03?

b. (0.1p) O altă posibilitate de a face pruning este să parcurgem arborele de decizie începând cu părinţii nodurilor-frunză şi să eliminăm în mod recursiv acele noduri de test pentru care câştigul de informaţie (sau un alt criteriu ales) este mai mic decât ε. Aceasta este strategia de "bottom-up pruning".
Observaţie: Spre deosebire de strategia top-down, în varianta de pruning de tip bottom-up nu vor fi eliminate noduri (cu IG < ε) pentru care există descendenţi al căror câştig de informaţie este mai mare sau egal cu ε.
Ce arbore se obţine făcând "bottom-up pruning" pe arborele dat mai sus, dacă se consideră ε = 0.03?

c. (0.1p) În acest exercițiu am folosit următoarea metodă pentru a determina dacă un atribut de intrare este independent de atributul de ieșire: IG < ε, unde ε e valoare mică, pozitivă apropiată de zero. Există o altă metodă de a verifica o astfel de independență în contextul trunchierii arborilor de decizie. Menționați numele ei.