

# Inteligența artificială (cu Python)

- pentru ingineri roboticieni -

Stelian Brad



Învățare interactivă

## Lucrarea 2: Matricea de confuzie

# Exemplu – matrice de confuzie 2x2

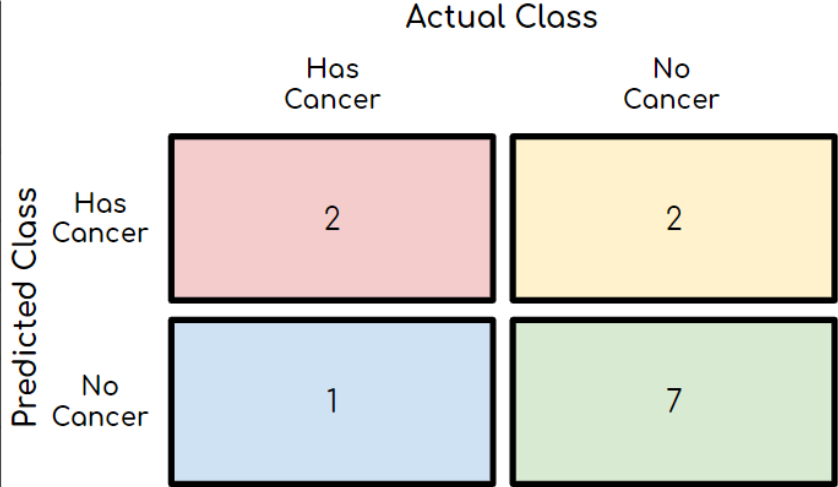
Imaginați-vă că am creat un model de învățare automată care prezice dacă un pacient are sau nu cancer. Tabelul prezintă douăsprezece predicții pe care le-a făcut modelul, precum și rezultatul real al fiecărui pacient. Cu datele asociate, putem completa matricea de confuzie. Apoi putem învăța o serie de lucruri despre modelul nostru.

Modelul nostru a prezis că 4/12 (roșu + galben) pacienți aveau cancer atunci când erau de fapt 3/12 (roșu + albastru) pacienți cu cancer.

TP = 2 (rosu)  
TN = 7 (verde)  
FP = 2 (galben)  
FN = 1 (albastru)  
TOTAL SET = 12

Acuratetea = (TP+TN)/TOTAL SET = 0.75  
Precizia<sub>p</sub> = TP/(TP+FP) = 0.5 (clasa poz.)  
Precizia<sub>n</sub> = TN/(TN+FN)=0.875 (cl. neg.)  
Sensitivitatea<sub>p</sub> = TP/(TP+FN) = 0.66666667  
Sensitivitatea<sub>n</sub> = TN/(TN+FP) = 0.77777778

Predicted Class	Actual Class
No Cancer	No Cancer
Has Cancer	Has Cancer
No Cancer	No Cancer
No Cancer	No Cancer
No Cancer	Has Cancer
Has Cancer	Has Cancer
No Cancer	No Cancer
Has Cancer	No Cancer
No Cancer	No Cancer
No Cancer	No Cancer
Has Cancer	No Cancer
No Cancer	No Cancer



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

# Exemplu – matrice de confunzie 2x2 (biblioteca Sklearn)

```
# Matricea de confuzie
import sklearn
from sklearn.metrics import confusion_matrix
# Acuratetea
from sklearn.metrics import accuracy_score
# Rechemarea
from sklearn.metrics import recall_score
# Precizia
from sklearn.metrics import precision_score
# Scorul F1
from sklearn.metrics import f1_score
import inspect
```

```
y_pred=[0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0]
y_true=[0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0]
```

```
confuzia = confusion_matrix(y_true, y_pred)
print("Matricea de confuzie=\n", confuzia)
acuratetea = accuracy_score(y_true, y_pred)
print("Acuratetea= ", acuratetea)
rechemarea = recall_score(y_true, y_pred, average=None)
print("Recall-ul= ", rechemarea)
precizia = precision_score(y_true, y_pred, average=None)
print("Precizia= ", precizia)
factorul_1 = f1_score(y_true, y_pred, average=None)
print("F1= ", factorul_1)
```

```
help(sklearn.metrics.confusion_matrix)
help(sklearn.metrics.recall_score)
```

```
confuzia = confusion_matrix(y_true, y_pred)
print("Matricea de confuzie=\n", confuzia)
acuratetea = accuracy_score(y_true, y_pred)
print("Acuratetea= ", acuratetea)
rechemarea = recall_score(y_true, y_pred, average="micro")
print("Recall-ul= ", rechemarea)
precizia = precision_score(y_true, y_pred, average="micro")
print("Precizia= ", precizia)
factorul_1 = f1_score(y_true, y_pred, average="micro")
print("F1= ", factorul_1)
```

```
Matricea de confuzie=
[[7 2]
 [1 2]]
Acuratetea= 0.75
Recall-ul= 0.75
Precizia= 0.75
F1= 0.75
```

Squeezed text (82 lines).

Squeezed text (108 lines).

```
>>> |
```

```
Matricea de confuzie=
[[7 2]
 [1 2]]
Acuratetea= 0.75
Recall-ul= [0.77777778 0.66666667]
Precizia= [0.875 0.5 ]
F1= [0.82352941 0.57142857]
```

Squeezed text (82 lines).

Squeezed text (108 lines).

```
>>>
```

Predictie

Fara cancer

Cancer

Efectiv

Fara cancer

Cancer

TN = 7

FP = 1

FN = 2

TP = 2

```
>>> import inspect
>>> linii=inspect.getsource(recall_score)
>>> print(linii)
```

Thus in binary classification, the count of true negatives is :math:'C\_{0,0}', false negatives is :math:'C\_{1,0}', true positives is :math:'C\_{1,1}' and false positives is :math:'C\_{0,1}'.

Returns

-----  
recall : float (if average is not None) or array of float, shape = [n\_unique\_labels]  
Recall of the positive class in binary classification or weighted average of the recall of each class for the multiclass task.

# pentru profesor matricea\_confuzie.py

# Exemplu – matrice de confuzie 3x3

#pentru profesor c\_m.py

```
from sklearn import metrics
```

```
y_pred = ["a", "b", "c", "a", "b"]
```

```
y_act = ["a", "b", "c", "c", "a"]
```

```
print(metrics.confusion_matrix(y_act, y_pred, labels=["a", "b", "c"]))
```

```
print(metrics.classification_report(y_act, y_pred, labels=["a", "b", "c"]))
```

Predicted Outcomes	Actual Outcomes
class a	class a
class b	class b
class c	class c
class a	class c
class b	class a

`y_pred` este o listă care conține etichetele prezise. `y_act` conține etichetele reale. `metrics.confusion_matrix()` include lista etichetelor reale, lista etichetelor previzionate și un argument opțional pentru a specifica ordinea etichetelor → calculează matricea de confuzie pentru intrările date. `metrics.classification_report()` include lista etichetelor reale, lista etichetelor previzionate și un argument opțional pentru a specifica ordinea etichetelor → calculează valori de performanță precum precizie, sensibilitate, scor F1 și asistență.

[[1 1 0] [0 1 0] [1 0 1]]		class b		class	
	precision	recall	f1-score	support	
a	0.50	0.50	0.50	2	
b	0.50	1.00	0.67	1	
c	1.00	0.50	0.67	2	
accuracy			0.60	5	
macro avg		0.67	0.67	0.61	5
weighted avg		0.70	0.60	0.60	5

`support` este numărul de eșantioane ale răspunsului adevărat care se află în acea clasă.

# Exemplu – matrice de confunzie 2x2 (biblioteca Pandas)

```
import pandas as pd
```

```
data = {'y_Actual': [1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0],  
        'y_Predicted': [1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0]  
}
```

```
df = pd.DataFrame(data, columns=['y_Actual', 'y_Predicted'])  
print(df)
```

	y_Actual	y_Predicted
0	1	1
1	0	1
2	0	0
3	1	1
4	0	0
5	1	1
6	0	1
7	0	0
8	1	1
9	0	0
10	1	0
11	0	0

```
>>>
```

```
import pandas as pd
```

```
data = {'y_Actual': [1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0],  
        'y_Predicted': [1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0]  
}
```

```
df = pd.DataFrame(data, columns=['y_Actual', 'y_Predicted'])
```

```
confusion_matrix = pd.crosstab(df['y_Actual'], df['y_Predicted'],  
                               rownames=['Actual'], colnames=['Predicted'])  
print(confusion_matrix)
```

```
Predicted 0 1  
Actual  
0         5 2  
1         1 4  
>>>
```

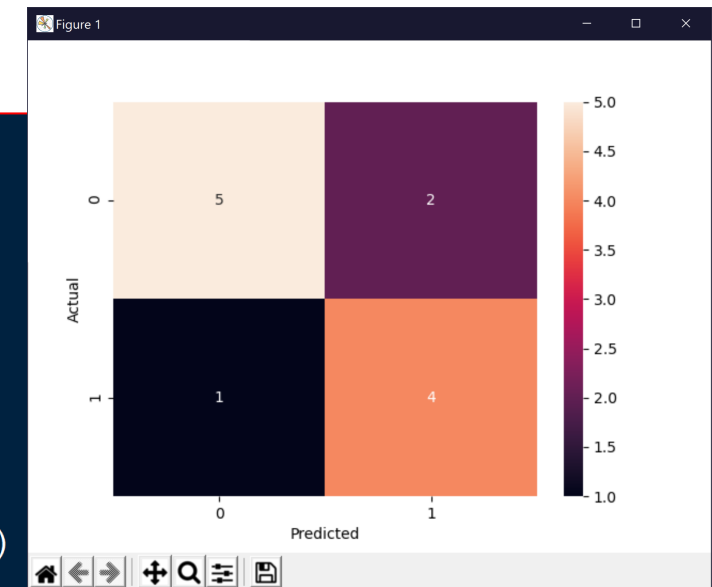
```
import pandas as pd  
import seaborn as sn  
import matplotlib.pyplot as plt
```

```
data = {'y_Actual': [1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0],  
        'y_Predicted': [1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0]  
}
```

```
df = pd.DataFrame(data, columns=['y_Actual', 'y_Predicted'])
```

```
confusion_matrix = pd.crosstab(df['y_Actual'], df['y_Predicted'],  
                               rownames=['Actual'], colnames=['Predicted'])
```

```
sn.heatmap(confusion_matrix, annot=True)  
plt.show()
```



# pentru profesor pandas\_c\_m.py



# Exemplu – matrice de confunzie 2x2 (biblioteca Pandas)

Opțional, puteți adăuga și totalurile la marginile matricei de confuzie setând `margins = True`

```
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt

data = {'y_Actual': [1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0],
        'y_Predicted': [1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0]}

df = pd.DataFrame(data, columns=['y_Actual', 'y_Predicted'])

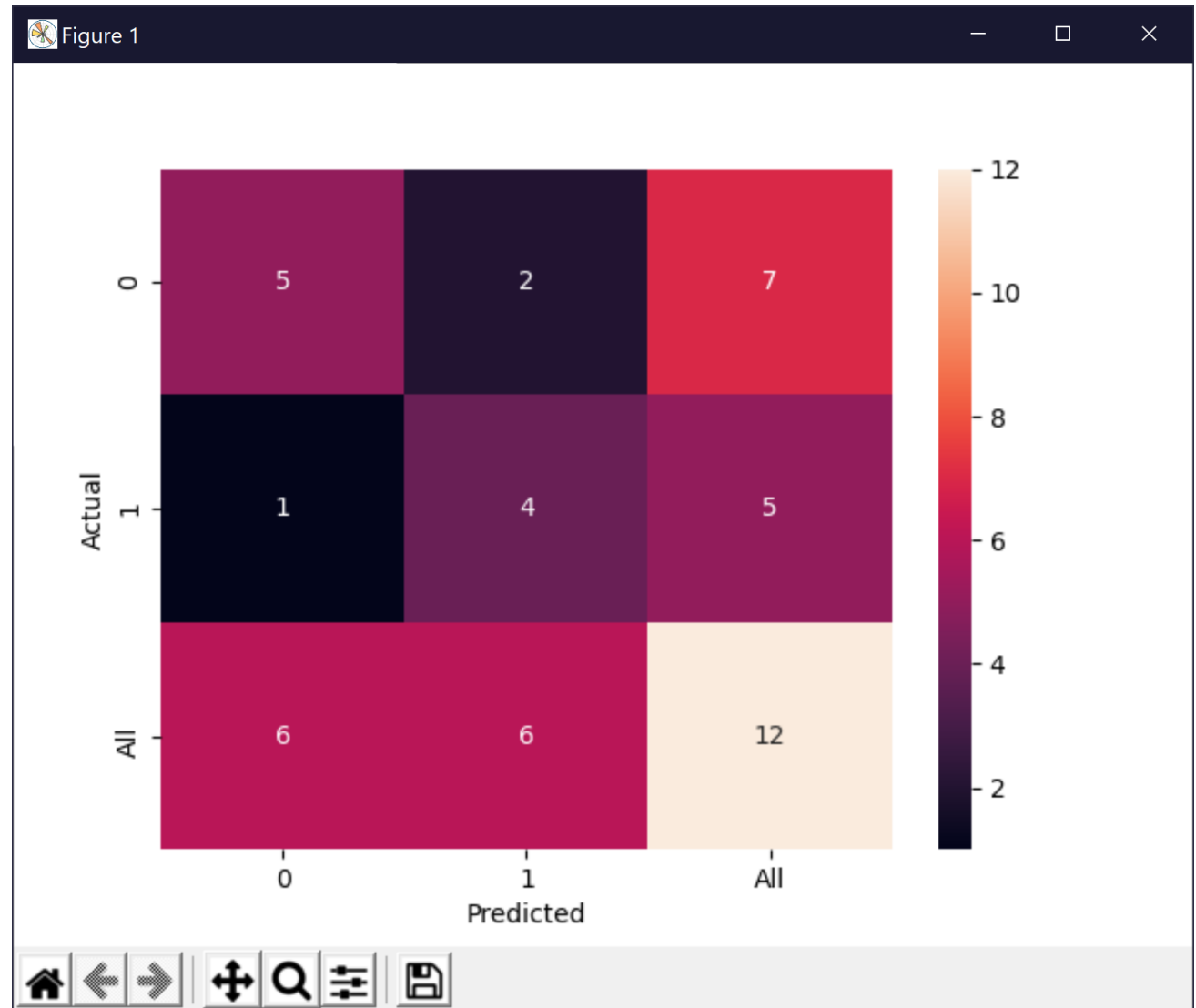
confusion_matrix = pd.crosstab(df['y_Actual'], df['y_Predicted'],
                               rownames=['Actual'], colnames=['Predicted'],
                               margins = True)

sn.heatmap(confusion_matrix, annot=True)
plt.show()
```

# pentru profesor pandas\_c\_m.py

Studiu **pandas DataFrame**

<https://pandas.pydata.org/pandas-docs/version/0.24.2/reference/frame.html>



# Exemplu – matrice de confunzie 2x2 (biblioteca Pandas)

# pentru profesor pandas\_ml\_c\_m.py

```
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt

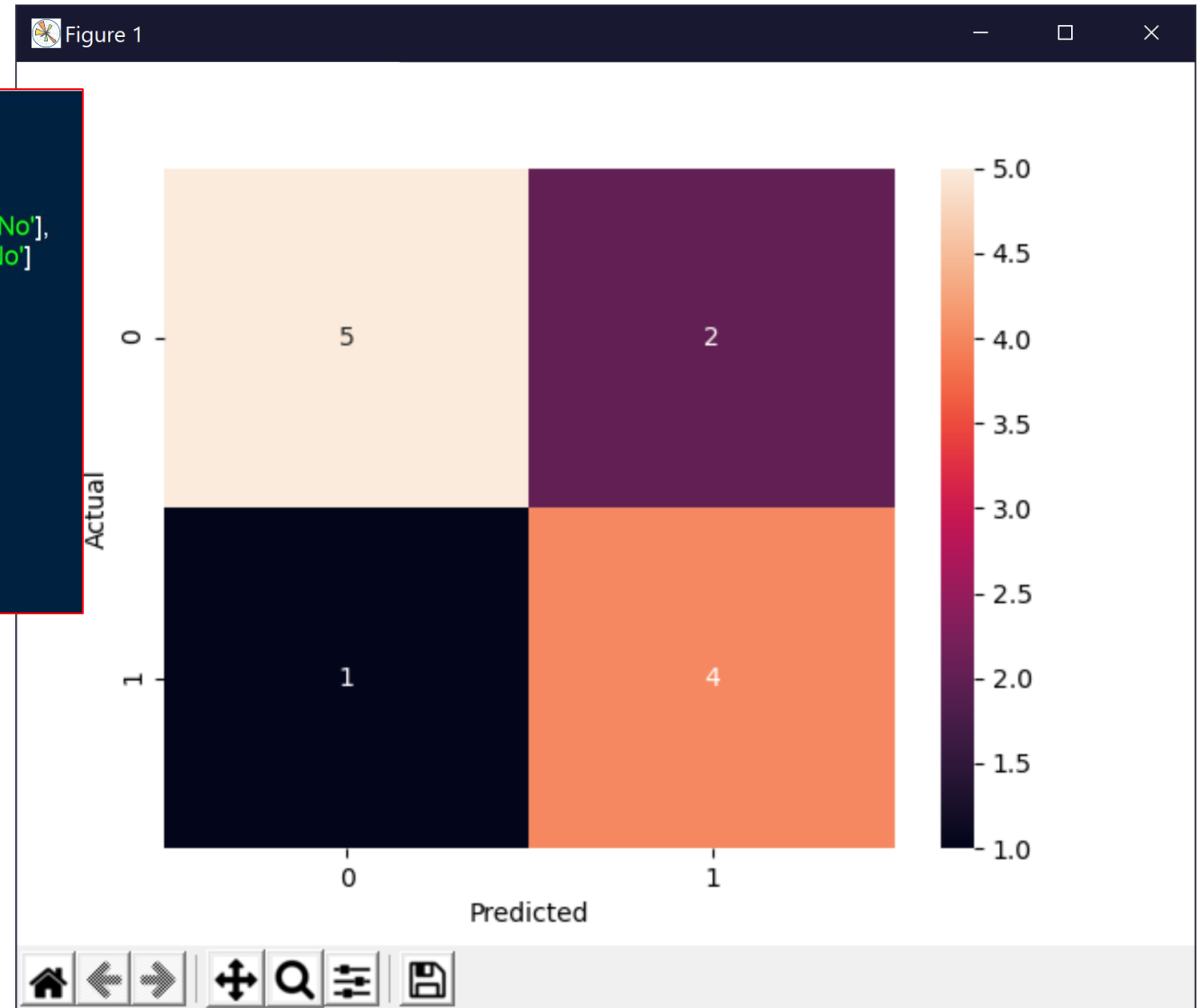
data = {'y_Actual':  ['Yes', 'No', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'Yes', 'No'],
        'y_Predicted': ['Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No', 'No']}

df = pd.DataFrame(data, columns=['y_Actual', 'y_Predicted'])
df['y_Actual'] = df['y_Actual'].map({'Yes': 1, 'No': 0})
df['y_Predicted'] = df['y_Predicted'].map({'Yes': 1, 'No': 0})

confusion_matrix = pd.crosstab(df['y_Actual'], df['y_Predicted'],
                               rownames=['Actual'], colnames=['Predicted'])

sn.heatmap(confusion_matrix, annot=True)
plt.show()
```

Conversia din Yes / No in 1/0





# Matricea de confuzie - sinteza

Exemplul de mai jos arata cum se reprezinta intr-o matrice de confuzie o situatie cu P instante pozitive si N instante negative.

		True condition			
Total population		Condition positive	Condition negative	$Prevalence = \frac{\sum \text{Condition positive}}{\sum \text{Total population}}$	$Accuracy\ (ACC) = \frac{\sum \text{True positive} + \sum \text{True negative}}{\sum \text{Total population}}$
Predicted condition	Predicted condition positive	True positive	False positive, Type I error	$Positive\ predictive\ value\ (PPV),\ Precision = \frac{\sum \text{True positive}}{\sum \text{Predicted condition positive}}$	$False\ discovery\ rate\ (FDR) = \frac{\sum \text{False positive}}{\sum \text{Predicted condition positive}}$
	Predicted condition negative	False negative, Type II error	True negative	$False\ omission\ rate\ (FOR) = \frac{\sum \text{False negative}}{\sum \text{Predicted condition negative}}$	$Negative\ predictive\ value\ (NPV) = \frac{\sum \text{True negative}}{\sum \text{Predicted condition negative}}$
		$True\ positive\ rate\ (TPR),\ Recall, Sensitivity, \text{probability of detection},\ Power = \frac{\sum \text{True positive}}{\sum \text{Condition positive}}$	$False\ positive\ rate\ (FPR),\ Fall-out, \text{probability of false alarm} = \frac{\sum \text{False positive}}{\sum \text{Condition negative}}$	$Positive\ likelihood\ ratio\ (LR+) = \frac{TPR}{FPR}$	$Diagnostic\ odds\ ratio\ (DOR) = \frac{LR+}{LR-}$
		$False\ negative\ rate\ (FNR),\ Miss\ rate = \frac{\sum \text{False negative}}{\sum \text{Condition positive}}$	$Specificity\ (SPC),\ Selectivity, \text{True negative rate}\ (TNR) = \frac{\sum \text{True negative}}{\sum \text{Condition negative}}$	$Negative\ likelihood\ ratio\ (LR-) = \frac{FNR}{TNR}$	
				$F_1\ score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$	

# Matricea de confuzie – aprofundarea conceptului

**Adevărat pozitiv (TP)** → Valoarea estimată se potrivește cu valoarea reală → Valoarea reală a fost pozitivă, iar modelul a prezis o valoare pozitivă

**Adevărat negativ (TN)** → Valoarea estimată se potrivește cu valoarea reală → Valoarea reală a fost negativă, iar modelul a prezis o valoare negativă

**Fals pozitiv (FP) - Eroare de tip 1** → Valoarea prezisă a fost pozitivă în mod fals → Valoarea reală a fost negativă, dar modelul a prezis o valoare pozitivă → Cunoscută și sub numele de eroare de tip 1

**Fals negativ (FN) - Eroare de tip 2** → Valoarea prezisă a fost negativă în mod fals → Valoarea reală a fost pozitivă, dar modelul a prezis o valoare negativă → Cunoscută și sub numele de eroare de tip 2

		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	560	60
	NEGATIVE	50	330

Adevărat pozitiv (TP) = 560; adică 560 puncte pozitive de date de clasă au fost corect clasificate de model

Adevărat negativ (TN) = 330; ceea ce înseamnă că 330 de puncte de date negative din clasă au fost corect clasificate de model

Fals pozitiv (FP) = 60; adică 60 de puncte de date negative ale clasei au fost incorect clasificate ca aparținând clasei pozitive de către model

Fals negativ (FN) = 50; ceea ce înseamnă că 50 de puncte pozitive de date de clasă au fost incorect clasificate ca aparținând clasei negative de către model

# Matricea de confuzie – de ce este importanta?

Să presupunem că doriți să preziceți câte persoane sunt infectate cu un virus contagios (COVID-19) în anumite perioade înainte ca acestea să prezinte simptomele și să le izolați de populația sănătoasă. Cele două valori pentru variabila noastră țintă ar fi: Bolnav și Nu Bolnav. De ce avem nevoie de o matrice de confuzie atunci când avem indicatorul “Acuratete”?

$$\text{Acuratetea} = \frac{TP + TN}{TP + FP + TN + FN}$$

TP = 30, TN = 930, FP = 30, FN = 10 → Acuratetea = 0.96 (96%)

ID	Bolnavi efectiv	Bolnavi predicție	Rezultat
1	1	1	TP
2	0	0	TN
3	0	0	TN
4	1	1	TP
5	0	0	TN
6	0	0	TN
7	1	0	FP
8	0	1	FN
9	0	0	TN
10	1	0	FP
:	:	:	:
1000	0	0	FN

Modelul nostru spune „Pot prezice persoanele bolnave in 96% din cazuri”. Cu toate acestea, modelul face contrariul → **prezice persoanele care nu se vor îmbolnăvi cu o precizie de 96% în timp ce bolnavii răspândesc virusul!** Credeți că aceasta este o valoare corectă pentru modelul nostru, având în vedere gravitatea problemei? Nu ar trebui să măsurăm câte cazuri pozitive putem prezice corect pentru a opri răspândirea virusului contagios? Sau poate, din cazurile prezise corect, câte sunt cazuri pozitive pentru a verifica fiabilitatea modelului nostru? Aici întâlnim conceptul dual de Precizie și Recall.

# Matricea de confuzie – de ce este importanta?

Persoane bolnave prezise  
corect ca fiind bolnave  
de catre model

Persoane sanatoase  
prezise incorrect ca fiind  
bolnave de catre model

ACTUAL VALUES			
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	TP (30)	FP (30)
	NEGATIVE	FN (10)	TN (930)

Persoane bolnave prezise  
incorrect ca fiind  
sanatoase de catre model

Persoane sanatoase prezise  
corect ca fiind sanatoase de  
catre model

$$\text{Scorul } F1 = \frac{2}{\frac{1}{\text{Sensitivitate}} + \frac{1}{\text{Precizie}}}$$

$$\text{Precizia} = \frac{TP}{TP + FP}$$

$$\text{Precizia} = \frac{30}{30 + 30} = 0.5$$

$$\text{Sensitivitatea} = \frac{TP}{TP + FN}$$

$$\text{Sensitivitatea} = \frac{30}{30 + 10} = 0.75$$

50% din cazurile prezise corect s-au dovedit a fi cazuri pozitive, iar 75% dintre cazurile pozitive au fost prezise cu succes.

**Precizia** ne spune câte dintre cazurile prezise corect s-au dovedit a fi de fapt pozitive. Precizia este o valoare utilă în cazurile în care falsul pozitiv este de o importanta mai mare decât falsul negativ.

**Recall-ul (sensitivitatea)** ne spune câte dintre cazurile pozitive reale am reușit să le prezicem corect cu modelul nostru. Sensitivitatea este o valoare utilă în cazurile în care falsul negativ este de o importanta mai mare decât falsul pozitiv.

**Scorul F1** este o medie armonică dintre precizie și sensibilitate și, prin urmare, oferă o idee combinată despre aceste două valori. Este maxim când Precizia este egala cu Sensitivitatea. Dar există o capcana aici. Interpretabilitatea scorului F1 este slabă. Aceasta înseamnă că nu știm ce maximizează clasificatorul nostru - precizia sau sensibilitatea? Deci, îl folosim în combinație cu alte metrice de evaluare, ceea ce ne oferă o imagine completă a rezultatului.

# Matricea de confuzie – de ce este importanta?

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# valori actuale
actual = [1,0,0,1,0,0,1,0,0,1]
# valori prezise
predicted = [1,0,0,1,0,0,0,1,0,0]

# matricea de confuzie
matrix = confusion_matrix(actual,predicted, labels=[1,0])
print('Confusion matrix : \n',matrix)

# rezultate
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Rezultate : \n', 'tp=', tp, 'fn=', fn, 'fp=', fp, 'tn=', tn)

# raport
matrix = classification_report(actual,predicted,labels=[1,0])
print('Raport : \n',matrix)
```

```
Confusion matrix :
[[2 2]
 [1 5]]
Rezultate :
tp= 2 fn= 2 fp= 1 tn= 5
Raport :
      precision    recall  f1-score   support

     1       0.67      0.50      0.57         4
     0       0.71      0.83      0.77         6

 accuracy          0.70         10
 macro avg       0.69      0.67      0.67         10
 weighted avg    0.70      0.70      0.69         10

>>>
```

# pentru profesor m\_c\_2.py

Sklearn are două funcții excelente: `confusion_matrix()` și `classification_report()`. Sklearn `confusion_matrix()` returnează valorile matricei de confuzie. Rezultatul este, însă, ușor diferit de ceea ce am studiat până acum. Acesta ia rândurile (liniile) ca valori reale (efective) și coloanele ca valori estimate. Restul conceptului rămâne același. Sklearn `classification_report()` produce precizie, sensibilitate și scor f1 pentru fiecare clasă țintă. În plus, are și câteva valori suplimentare: **micro medie aritmetica**, **macro medie aritmetica** și **medie ponderată**. Se calculeaza pentru toate clasele.

$$\text{Precizia micro medie} = \frac{TP1 + TP2}{TP1 + TP2 + FP1 + FP2}$$

$$\text{Precizia macro medie} = \frac{TP1 + TP2}{2}$$

# Matricea de confuzie – exemplu multi-clasa

Pentru a măsura rezultatele algoritmilor de învățare automată, matricea de confuzie are nevoie de o generalizare pentru cazul cu mai multe clase. Să presupunem că avem un eșantion de 25 de animale, de ex. 7 pisici, 8 câini și 10 șerpi. Matricea de confuzie a algoritmului nostru de recunoaștere poate arăta ca în următorul tabel:

	predicted			
		dog	cat	snake
	dog	6	2	0
	cat	1	6	0
	snake	1	1	8

În această matrice de confuzie, sistemul a prezis corect 6 dintre cei 8 câini, dar în 2 cazuri a prezis pentru câine o pisică. Cele 7 pisici au fost determinate corect în 6 cazuri, dar într-un caz o pisică a fost considerată câine. De obicei, este greu să spui ca un șarpe este un câine sau o pisică, dar asta s-a întâmplat cu clasificatorul nostru în două cazuri. Totuși, 8 din 10 șerpi au fost recunoscuți corect. Putem vedea că toate predicțiile corecte sunt situate în diagonala tabelului, astfel încât erorile de predicție pot fi găsite cu ușurință în tabel, deoarece acestea vor fi reprezentate de valori în afara diagonalei. Putem generaliza acest lucru în cazul mai multor clase. Pentru a face acest lucru, rezumăm rândurile și coloanele matricei de confuzie. Având în vedere că matricea este orientată ca mai sus, adică faptul că un rând dat al matricei corespunde valorii specifice pentru „adevăr”, avem:

$$Precision_i = \frac{M_{ii}}{\sum_j M_{ji}}$$
$$Recall_i = \frac{M_{ii}}{\sum_j M_{ij}}$$

$$precision_{dogs} = 6/(6 + 1 + 1) = 3/4 = 0.75$$
$$precision_{cats} = 6/(2 + 6 + 1) = 6/9 = 0.67$$
$$precision_{snakes} = 8/(0 + 0 + 8) = 1$$

$$recall_{dogs} = 6/(6 + 2 + 0) = 3/4 = 0.75$$
$$recall_{cats} = 6/(1 + 6 + 0) = 6/7 = 0.86$$
$$recall_{snakes} = 8/(1 + 1 + 8) = 4/5 = 0.8$$



# Matricea de confuzie – aplicare biblioteca Numpy

```
import numpy as np
```

```
cm = np.array([
[5825, 1, 49, 23, 7, 46, 30, 12, 21, 26],
[ 1, 6654, 48, 25, 10, 32, 19, 62, 111, 10],
[ 2, 20, 5561, 69, 13, 10, 2, 45, 18, 2],
[ 6, 26, 99, 5786, 5, 111, 1, 41, 110, 79],
[ 4, 10, 43, 6, 5533, 32, 11, 53, 34, 79],
[ 3, 1, 2, 56, 0, 4954, 23, 0, 12, 5],
[ 31, 4, 42, 22, 45, 103, 5806, 3, 34, 3],
[ 0, 4, 30, 29, 5, 6, 0, 5817, 2, 28],
[ 35, 6, 63, 58, 8, 59, 26, 13, 5394, 24],
[ 16, 16, 21, 57, 216, 68, 0, 219, 115, 5693]])
```

```
def precision(label, confusion_matrix):
    col = confusion_matrix[:, label]
    return confusion_matrix[label, label] / col.sum()
```

```
def recall(label, confusion_matrix):
    row = confusion_matrix[label, :]
    return confusion_matrix[label, label] / row.sum()
```

```
def precision_macro_average(confusion_matrix):
    rows, columns = confusion_matrix.shape
    sum_of_precisions = 0
    for label in range(rows):
        sum_of_precisions += precision(label, confusion_matrix)
    return sum_of_precisions / rows
```

```
def recall_macro_average(confusion_matrix):
    rows, columns = confusion_matrix.shape
    sum_of_recalls = 0
    for label in range(columns):
        sum_of_recalls += recall(label, confusion_matrix)
    return sum_of_recalls / columns
```

```
print("label precision recall")
for label in range(10):
    print(f"{label:5d} {precision(label, cm):9.3f} {recall(label, cm):6.3f}")
```

```
print("precision total:", precision_macro_average(cm))
```

```
print("recall total:", recall_macro_average(cm))
```

```
def accuracy(confusion_matrix):
    diagonal_sum = confusion_matrix.trace()
    sum_of_all_elements = confusion_matrix.sum()
    return diagonal_sum / sum_of_all_elements
```

```
print("accuracy:", accuracy(cm))
```

```
label precision recall
0 0.983 0.964
1 0.987 0.954
2 0.933 0.968
3 0.944 0.924
4 0.947 0.953
5 0.914 0.980
6 0.981 0.953
7 0.928 0.982
8 0.922 0.949
9 0.957 0.887
precision total: 0.9496885564052286
recall total: 0.9514531547877969
accuracy: 0.9503833333333334
>>> |
```

```
import numpy as np
```

```
cm = np.array([
[5825, 1, 49, 23, 7, 46, 30, 12, 21, 26],
[ 1, 6654, 48, 25, 10, 32, 19, 62, 111, 10],
[ 2, 20, 5561, 69, 13, 10, 2, 45, 18, 2],
[ 6, 26, 99, 5786, 5, 111, 1, 41, 110, 79],
[ 4, 10, 43, 6, 5533, 32, 11, 53, 34, 79],
[ 3, 1, 2, 56, 0, 4954, 23, 0, 12, 5],
[ 31, 4, 42, 22, 45, 103, 5806, 3, 34, 3],
[ 0, 4, 30, 29, 5, 6, 0, 5817, 2, 28],
[ 35, 6, 63, 58, 8, 59, 26, 13, 5394, 24],
[ 16, 16, 21, 57, 216, 68, 0, 219, 115, 5693]])
```

```
def precision(label, confusion_matrix):
    col = confusion_matrix[:, label]
    return confusion_matrix[label, label] / col.sum()
```

```
def recall(label, confusion_matrix):
    row = confusion_matrix[label, :]
    return confusion_matrix[label, label] / row.sum()
```

```
def precision_macro_average(confusion_matrix):
    rows, columns = confusion_matrix.shape
    sum_of_precisions = 0
    for label in range(rows):
        sum_of_precisions += precision(label, confusion_matrix)
    return sum_of_precisions / rows
```

```
def recall_macro_average(confusion_matrix):
    rows, columns = confusion_matrix.shape
    sum_of_recalls = 0
    for label in range(columns):
        sum_of_recalls += recall(label, confusion_matrix)
    return sum_of_recalls / columns
```

```
print("label precision recall")
for label in range(10):
    print(f"{label:5d} {precision(label, cm):9.3f} {recall(label, cm):6.3f}")
```

```
print("precision total:", precision_macro_average(cm))
```

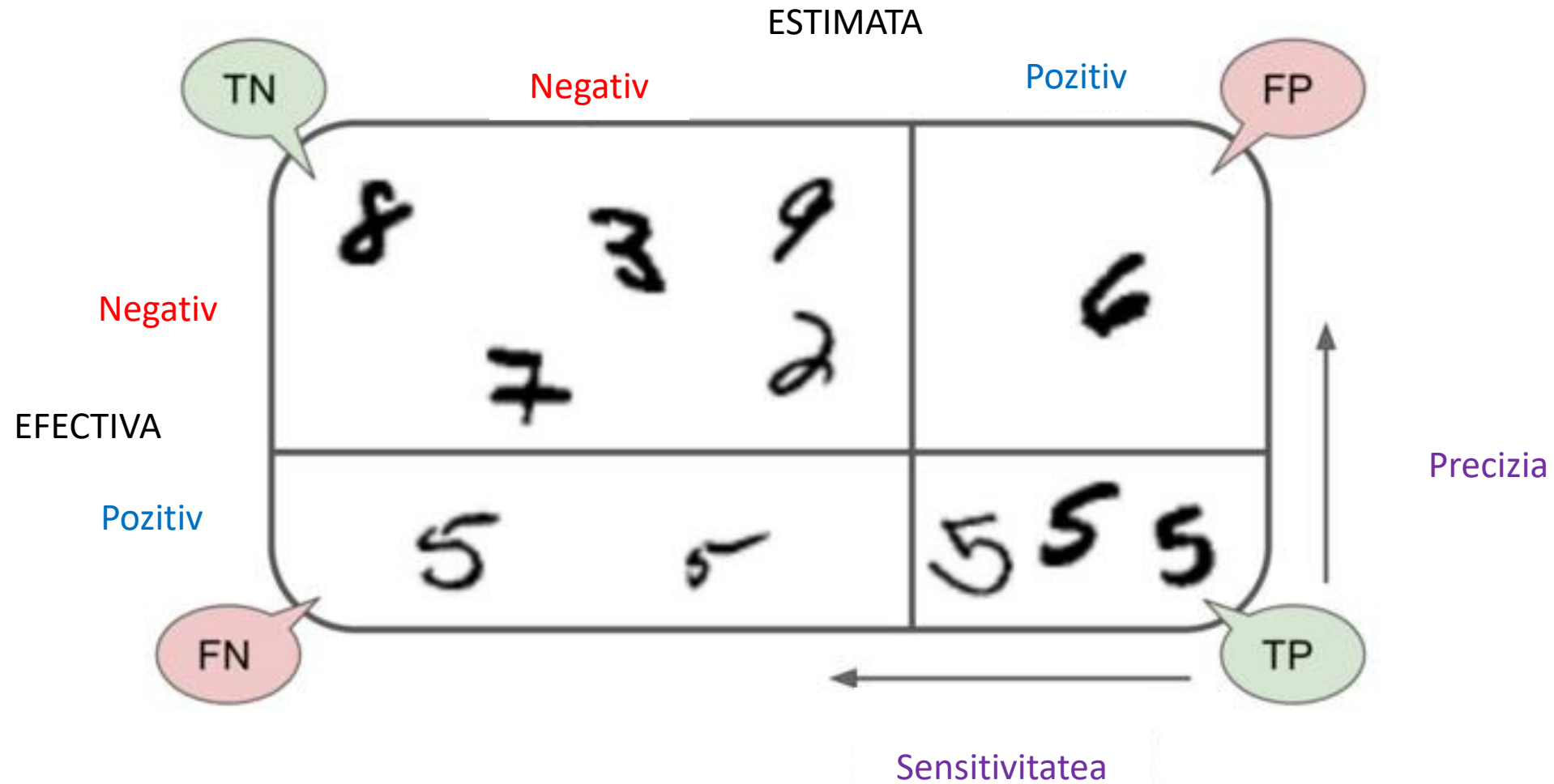
```
print("recall total:", recall_macro_average(cm))
```

```
def accuracy(confusion_matrix):
    diagonal_sum = confusion_matrix.trace()
    sum_of_all_elements = confusion_matrix.sum()
    return diagonal_sum / sum_of_all_elements
```

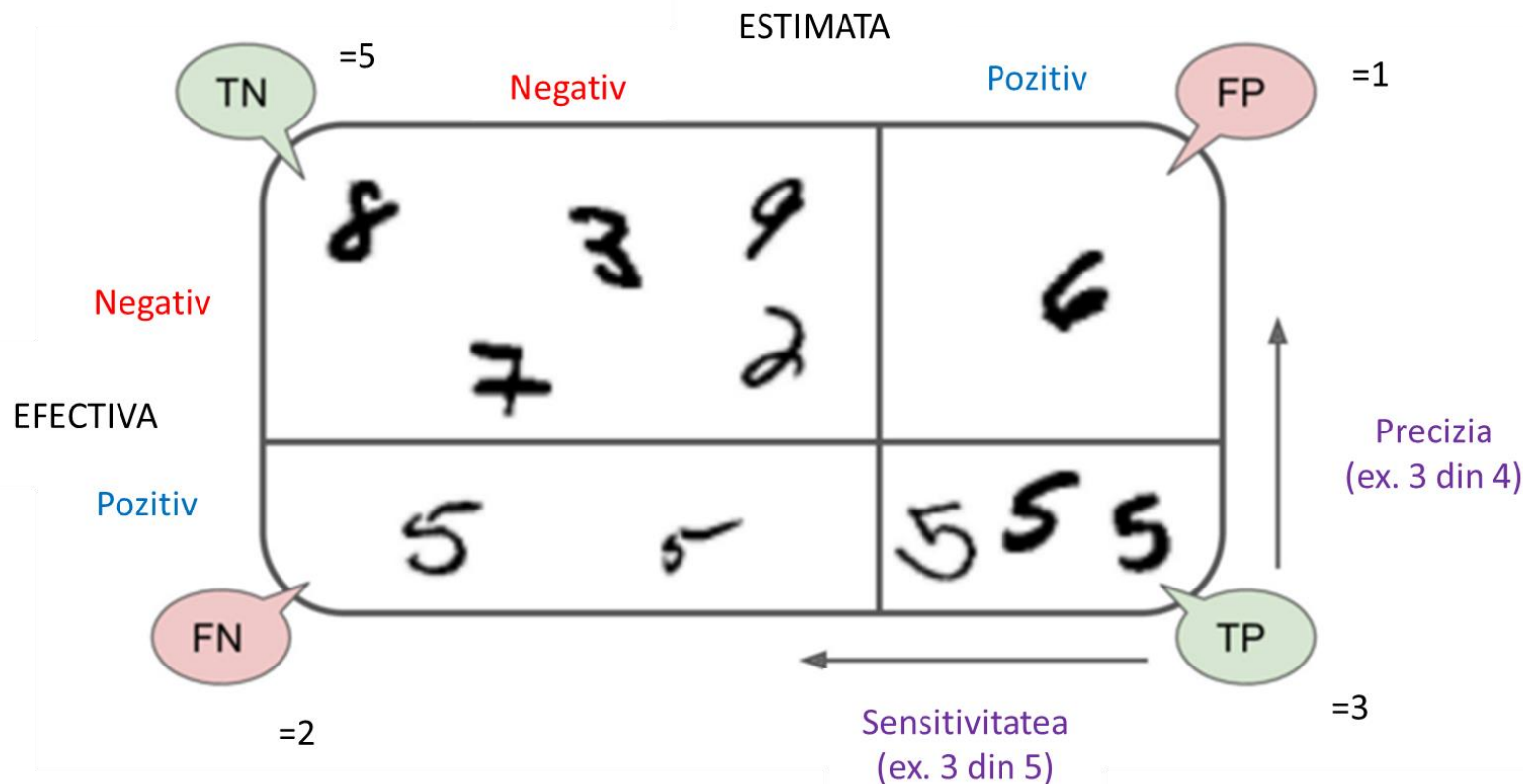
```
print("accuracy:", accuracy(cm))
```

#pentru profesor numpy\_cm.py

# Exercitiu – care este acuratetea si precizia modelului?



# Exercitiu – raspuns



$$\text{Precizia} = \frac{TP}{TP + FP} = \frac{3}{3 + 1} = \frac{3}{4} = 75\%$$

$$\text{Specificitatea} = \frac{TN}{TN + FP} = \frac{5}{5 + 1} = \frac{5}{6} = 83.33\%$$







$$\text{Sensitivitatea} = \frac{TP}{TP + FN} = \frac{3}{3 + 2} = \frac{3}{5} = 60\%$$

$$F1 = \frac{2 \times 75 \times 60}{75 + 60} = 66.66\%$$

$$\text{Acuratetea} = \frac{TN + TP}{TN + TP + FN + FP} = \frac{5 + 3}{5 + 3 + 2 + 1} = \frac{8}{11} = 72.72\%$$










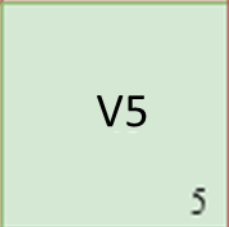





# Exercitiu pentru mai multe clase

Cum ar funcționa o matrice de confuzie pentru o problemă de clasificare multi-clasă? Avem o matrice de confuzie pentru o problemă multi-clasă în care trebuie să prezicem dacă o persoană prefera Facebook, Instagram sau Snapchat. Matricea de confuzie este de tipul 3 x 3 ca mai jos. Numerele din celule indica numarul celulei din cadrul matricii. Trebuie sa calculati TP, TN, FP, FN pentru fiecare clasa si apoi valorile pentru acuratete, precizie si recall (senzitivitate). Consideram ca numerele din cellule sunt V1, ..., V9.

		ACTUAL VALUES		
				
PREDICTED VALUES		<div>V1</div> <div>1</div>	<div>V2</div> <div>2</div>	<div>V3</div> <div>3</div>
		<div>V4</div> <div>4</div>	<div>V5</div> <div>5</div>	<div>V6</div> <div>6</div>
		<div>V7</div> <div>7</div>	<div>V8</div> <div>8</div>	<div>V9</div> <div>9</div>

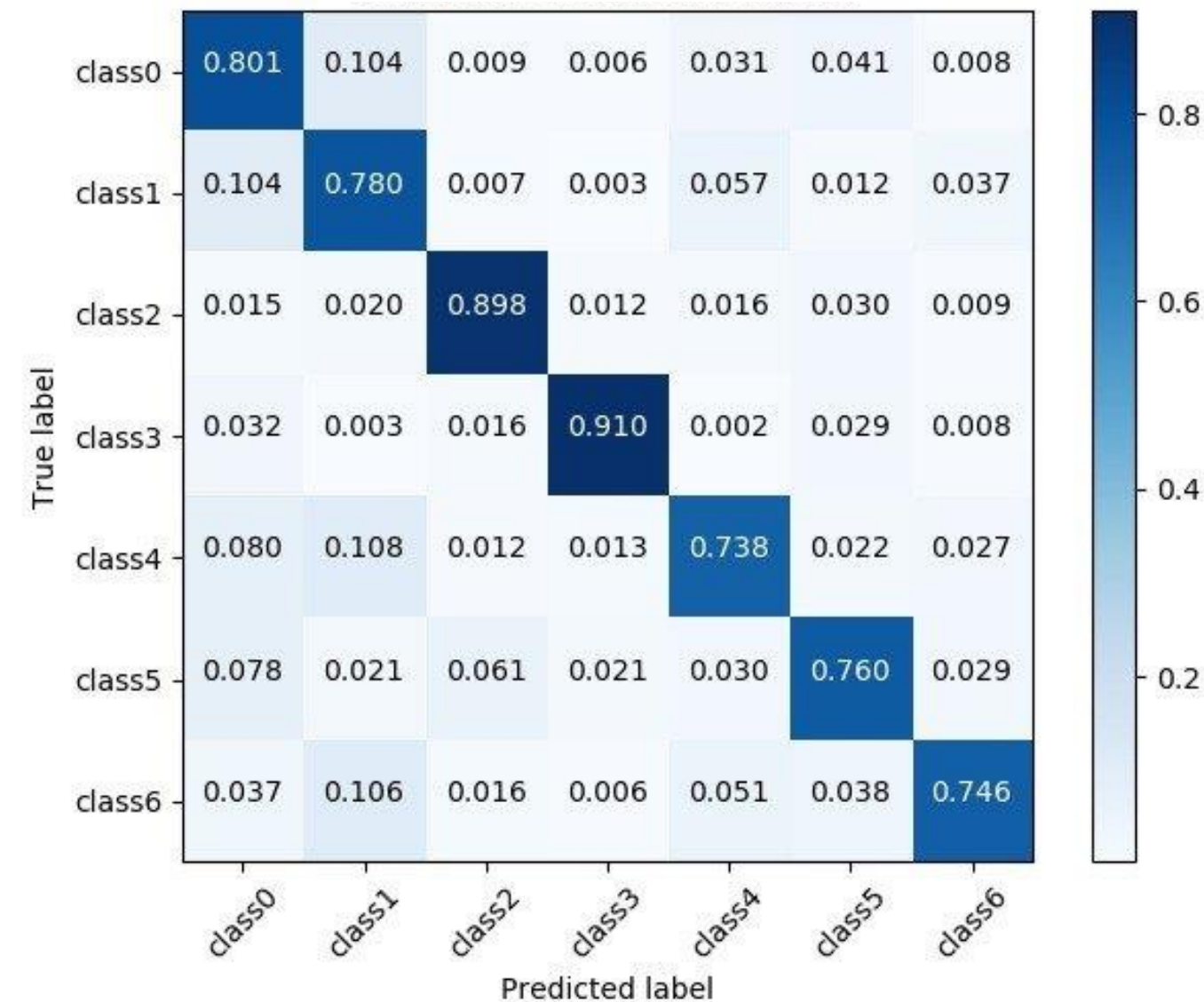
# Exercitiu pentru mai multe clase - raspuns

Adevăratul pozitiv, adevăratul negativ, falsul pozitiv și falsul negativ pentru fiecare clasă ar fi calculate după cum urmează:

		ACTUAL VALUES					
							
PREDICTED VALUES		 V1 1	 V2 2	 V3 3	Facebook	Instagram	Snapchat
		 V4 4	 V5 5	 V6 6	TP = V1 FP = V2+V3 TN = V5+V6+V8+V9 FN = V4+V7	TP = V5 FP = V4+V6 TN = V1+V3+V7+V9 FN = V2+V8	TP = V9 FP = V7+V8 TN = V1+V2+V4+V5 FN = V3+V6
		 V7 7	 V8 8	 V9 9			

# Exercitiu | descifrati matricea de mai jos

Matrice de confuzie NORMALIZATA



Calculati acuratetea, precizia si sensibilitatea pentru clasele 0 si 4.