1 Metrici de performanta pentru clasificatori: accuracy, precision, recall, F1 score

Resurse:

- Understanding Accuracy, Recall, Precision, F1 Scores, and Confusion Matrices (https://towardsdatascience.com/understanding-accuracy-recall-precision-f1-scores-and-confusion-matrices-561e0f5e328c)
- Accuracy, Precision, Recall or F1? (https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9)

In [1]:

```
import numpy as np
import sklearn
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score
```

executed in 486ms, finished 12:22:17 2021-04-20

In [2]:

```
print(f'numpy version: {np.__version__}')
print(f'sklearn version: {sklearn.__version__}')
```

executed in 5ms, finished 12:22:17 2021-04-20

```
numpy version: 1.19.2
sklearn version: 0.24.1
```

2 Matrice de confuzie

Sa presupunem ca avem de construit un clasificator care sa eticheteze date care apartin la doua clase: 0 (negativ) si 1 (pozitiv). Exemple: mail spam (pozitiv) versus nonspam (negativ), o imagine contine un catelus (pozitiv) sau nu contine etc.

Clasificatorul face anumite predictii de clasa, care pot fi sau nu conforme cu realitatea. Se obtine urmatoarea **matrice de confuzie**:

			Prezis (clasificat, inferat)	
		Positive	Negative	**Total**
Actual (real, ground truth)	Positive	True positive (TP)	False negative (FN)	**P**
	Negative	False positive (FP)	True negative (TN)	**N**

Avem:

- P = TP + FN
- N = FP + TN

Ne dorim ca in afara diagonalei (valorile FP, FN) sa fie cat mai mici, ideal 0.

Pe baza matricei de confuzie se calculeaza in mod direct acuratetea:

$$Accuracy = \frac{TP + TN}{P + N}$$

Considerarea doar a acurateti poate fi inselatoare in cazurile in care clasele sunt debalansate (numar foarte mare de exemplare intr-o clasa in comparatie cu numarul de date din cealalta clasa).

3 Exemplu numeric 1

		Prezis (clasificat, inferat)		
		Positive	Negative	**Total**
Actual (real, ground truth)	Positive	1	1	**P=2**
	Negative	0	998	**N=999**

In acest caz, acuratetea este:

$$Accuracy = \frac{998 + 1}{1000} = 0.999 = 99.9\%$$

Mai mult, pentru o astfel de situatie debalansata, un clasificator care prrezice mereu "negativ" are rata de succes de 99.8%!

Daca acele 2 cazuri pozitiv sunt insa importante (incercare de frauda/intruziune, recunoasterea de virusi/virusuri etc), clasificarea este de fapt slaba. Precision, recall si F1 score sunt niste metrici senzitive la clase dezechilibrate.

4 Precision

Definitia preciziei unui clasificator binar (precision) este:

$$Precision = \frac{TP}{TP + FP}$$

Pentru exemplul numeric de mai sus obtinem:

$$Precision = \frac{1}{1 + 0} = 1$$

5 Recall

Recall se defineste ca:

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{P}$$

Pentru exemplul numeric de mai sus obtinem:

$$Recall = \frac{1}{1 + 1} = 0.5$$

6 F1 score

Scorul F1 se defineste ca medie armonica a precision si recall:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Pentru exemplul numeric de mai sus obtinem:

$$F1 = 2 \cdot \frac{1 \cdot 0.5}{1 + 0.5} \approx 0.66$$

7 Exemplu numeric 2

In [3]:

```
# ground truth
labels =      np.array([1, 0, 0, 1, 1, 1, 0, 1, 1, 1])
# predicted
predictions = np.array([0, 1, 1, 1, 1, 0, 1, 0, 1, 0])
```

executed in 6ms, finished 12:22:18 2021-04-20

7.1 Calcul folosind numpy

In [4]:

```
# calcul manual de TP, FP etc

accuracy = np.sum(labels == predictions)
positive_mask = labels == 1
negative_mask = labels == 0
P = np.sum(positive_mask)
N = np.sum(negative_mask)
TP = np.sum(predictions[positive_mask] == 1)
print(f'true positive: {TP}')
TN = np.sum(predictions[negative_mask] == 0)
print(f'true negative: {TN}')
FP = np.sum(predictions[negative_mask] == 1)
print(f'false positive: {FP}')
FN = np.sum(predictions[positive_mask] == 0)
print(f'false negative: {FN}')

accuracy = (TP+TN)/(P+N)
precision = TP/(TP+FP)
recall = TP/(TP+FN)
print(f'accuracy: {accuracy}')
print(f'precision: {precision}')
print(f'recall: {recall}')
print(f'F1 score: {2 * precision * recall / (precision + recall)}')
```

executed in 5ms, finished 12:22:21 2021-04-20

```
true positive: 3
true negative: 0
false positive: 3
false negative: 4
accuracy: 0.3
precision: 0.5
recall: 0.42857142857142855
F1 score: 0.4615384615384615
```

In [5]:

```
assert TP + FN == P, 'TP si FN trebuie sa dea P'
assert TN + FP == N, 'TN si FP trebuie sa dea N'
```

executed in 6ms, finished 12:22:28 2021-04-20

8 Calcul folosind sklearn

In [6]:

```
from sklearn.metrics import confusion_matrix
confusion = confusion_matrix(labels, predictions)
print(confusion)
# FN = confusion[1][0]
# TN = confusion[0][0]
# TP = confusion[1][1]
# FP = confusion[0][1]
```

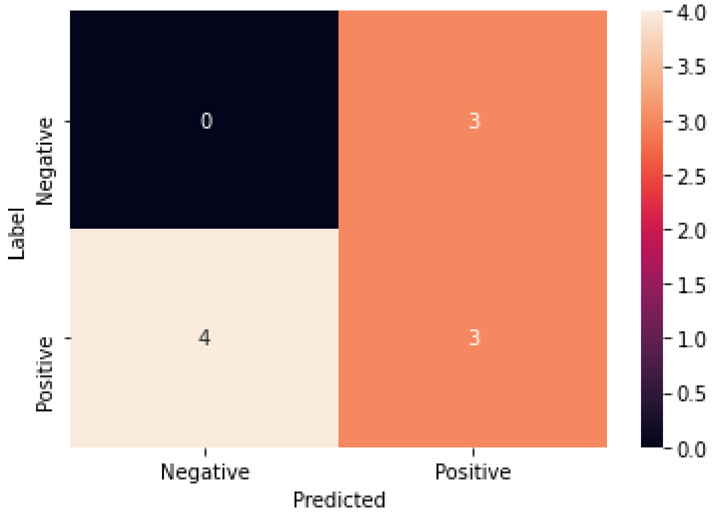
executed in 8ms, finished 12:22:34 2021-04-20

```
[[0 3]
 [4 3]]
```

In [7]:

```
# heatmap
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(confusion , annot=True , xticklabels=['Negative' , 'Positive'] , ytickl
plt.ylabel("Label")
plt.xlabel("Predicted")
plt.show()
```

executed in 635ms, finished 12:23:03 2021-04-20



In [8]:

```
print(f'Acuratete: {accuracy_score(labels , predictions)}')
```

executed in 7ms, finished 12:23:09 2021-04-20

```
Acuratete: 0.3
```

In [9]:

```
print(f'Precision: {precision_score(labels , predictions)}')
```

executed in 9ms, finished 12:23:10 2021-04-20

```
Precision: 0.5
```

In [10]:

```
print(f'Recall: {recall_score(labels , predictions)}')
```

executed in 8ms, finished 12:23:10 2021-04-20

```
Recall: 0.42857142857142855
```

In [11]:

```
print(f'F1 score: {f1_score(labels, predictions)}')
```

executed in 8ms, finished 12:23:16 2021-04-20

```
F1 score: 0.4615384615384615
```

In [12]:

```
from sklearn.metrics import classification_report
print(classification_report(labels,predictions))
```

executed in 13ms, finished 12:23:17 2021-04-20

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.50      0.43      0.46         7

    accuracy                           0.30        10
   macro avg       0.25      0.21      0.23        10
weighted avg       0.35      0.30      0.32        10
```

In [ ]: