# Identificarea Focului din Imagini

De Stoica Vlad Ionut

Focul este una dintre cele mai usoare de recunoscut chestii pentru oameni. Datorita registrului sau vizual, culorile in care apare, dar mai ales forma lui, il face foarte usor de recunoscut.

Daca noua ne este atat de usor sa recunoastem focul, oare este usor si pentru un algoritm de invatare automata?

Librariile folosite sunt: os, OpenCV (cv2), Numpy, StandardScaler din sklearn.preprocessing, KNeighboursClassifier din sklearn.neighbours si matplotlib.pyplot

Algoritmul pe care l-am ales sa-l folosesc pentru acest proiect este K-Nearest-Neighbors, un algoritm care nu are nevoie de foarte multe date de antrenare, acestea fiind alcatuite din:

- Setul “fire” are 17 imagini, 4 dintre ele fiind poze luate de mine, restul fiind luate de pe internet.
- Setul “no_fire” are 12 imagini, una este luata si editata de mine, iar restul sunt de pe internet

Setul de testare are:

- Setul “fire” are 7 imagini, doar una este luata de pe internet, restul de mine
- Setul “no_fire” 6 imagini, toate luate de mine

Pentru ca algoritmul KNN sa functioneze, imaginile trebuie sa fie procesate intr-un de data mai “digestibil” pentru algoritm. Acest process contine:

- Transformarea imaginii color intr-una gri
- Aplatizarea imaginii intr-un vector 1-Dimensional
- Retransformarea lui intr-un vector 2-Dimensional de tip “numpy”
- Standardizarea datelor folosind StandardScaler

Dupa Preprocesare, algoritmul primeste imaginile de antrenare impreuna cu etichetele lor. Pentru a verifica performanta antrenarii, se foloseste algoritmul pentru pentru a face preziceri asupra setului de antrenare.

Aceste predictii sunt folosite impreuna cu etichetele actuale pentru a gasi elementele matricei de confuzie, True Positive(TP), True Negative(TN), False Positive(FP), False Negative(FN), “True” insemnand cand predictia este egala cu eticheta actuala, “False” cand nu sunt egale, “Positive”, reprezinta cand predictia este “fire”, “Negative” cand predictia este “no_fire”.

Acestea fiind folosite apoi pentru a calcula alte masurari semnificative clasificarilor: Acuratetea ((TP+TN)/(TP+TN+FN+FP)), Precizia (TP/(TP+FP)), Sensibilitatea (TP/(TP+FN)), Specifitatea (TN/(TN+FP)) si Scorul F1 (Sensibilitate*Precizie/(Sensibilitate+Precizie)).

Pentru diferite valori ale lui K, rezultatele antrenarii sunt:

2:

```
Rezultatele antrenarii:
True Positive: 17
True Negative: 7
False Positive: 5
False Negative: 0

Acuratetea: 0.8275862068965517
Precizia: 0.7727272727272727
Sensibilitatea: 1.0
Specifitatea: 0.5833333333333334
Scor F1: 0.8717948717948718
```

3:

```
Rezultatele antrenarii:
True Positive: 11
True Negative: 10
False Positive: 2
False Negative: 6

Acuratetea: 0.7241379310344828
Precizia: 0.8461538461538461
Sensibilitatea: 0.6470588235294118
Specifitatea: 0.8333333333333334
Scor F1: 0.7333333333333334
```

4:

```
Rezultatele antrenarii:
True Positive: 12
True Negative: 5
False Positive: 7
False Negative: 5

Acuratetea: 0.5862068965517241
Precizia: 0.631578947368421
Sensibilitatea: 0.7058823529411765
Specifitatea: 0.4166666666666667
Scor F1: 0.6666666666666667
```

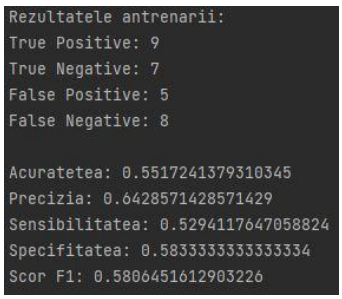
5:

```
Rezultatele antrenarii:
True Positive: 10
True Negative: 8
False Positive: 4
False Negative: 7

Acuratetea: 0.6206896551724138
Precizia: 0.7142857142857143
Sensibilitatea: 0.5882352941176471
Specifitatea: 0.6666666666666666
Scor F1: 0.6451612903225806
```

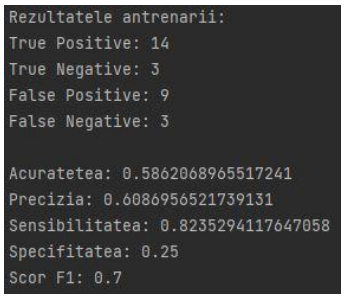
6:

```
Rezultatele antrenarii:
True Positive: 12
True Negative: 6
False Positive: 6
False Negative: 5

Acuratetea: 0.6206896551724138
Precizia: 0.6666666666666666
Sensibilitatea: 0.7058823529411765
Specifitatea: 0.5
Scor F1: 0.6857142857142857
```

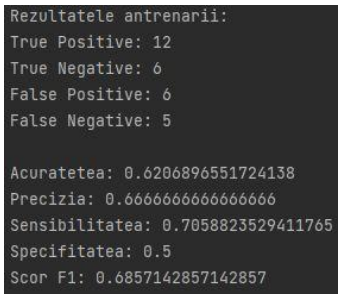
7:




8:




9:




Dintre toate aceste iteratii, algoritmul valoarea K = 2 are cele mai favorabile valori, in special Sensibilitatea, care este foarte importanta din motive ce vor fi explicate mai tarziu; doar Specifitatea are o valoare mai mica, aproape de 0.5.

Valoriile medii ale masuratoriilor sunt:

Acuratetea – 0.642

Precizia – 0.694

Sensibilitate – 0.713

Specifitate – 0.542

Scor F1 – 0.696

Se poate observa ca, in mare, Sensibilitatea are valorile cele mai mari, in timp ce Specifitatea are valorile mai mici.

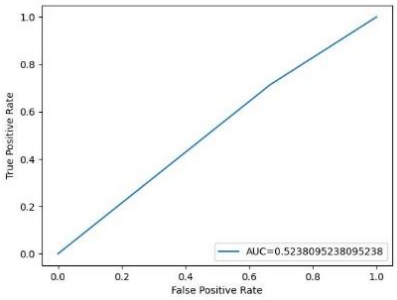
Apoi, algoritmul knn primeste imaginile de testare, preprocesate, pentru a face predictii asupra etichetei care ar trebui sa le apartina.

Similar ca inainte, trebuie sa se verifice performanta algoritmului, calculand matricea de confuzie si apoi restul masuratorilor. Pe langa acele masuratori se adauga si curba ROC, un graphic ce ajuta la masurarea eficientei algoritmului, pe axa X se afla Specifitatea, iar pe Y, Sensibilitatea. Dedesubtul acestui graphic va fi afisat AUC, aria de sub graphic.

Pentru fiecare valoare K, rezultatele verificarii sunt:

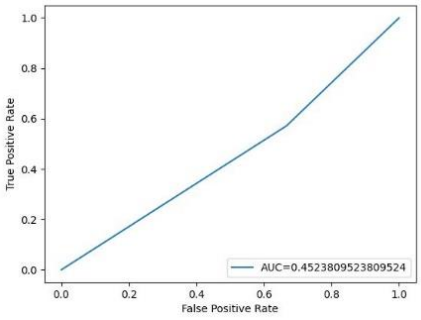
2.

```
True Positive: 5
True Negative: 2
False Positive: 4
False Negative: 2

Acuratetea: 0.5384615384615384
Precizia: 0.5555555555555556
Sensibilitatea: 0.7142857142857143
Specifitatea: 0.3333333333333333
Scor F1: 0.6250000000000001
```



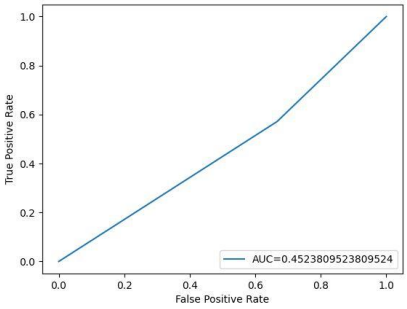

3.

```
True Positive: 4
True Negative: 2
False Positive: 4
False Negative: 3

Acuratetea: 0.46153846153846156
Precizia: 0.5
Sensibilitatea: 0.5714285714285714
Specifitatea: 0.3333333333333333
Scor F1: 0.5333333333333333
```



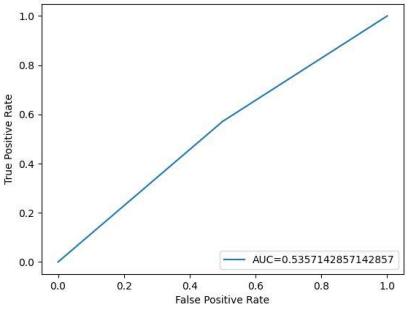
4.

```
True Positive: 4
True Negative: 2
False Positive: 4
False Negative: 3

Acuratetea: 0.46153846153846156
Precizia: 0.5
Sensibilitatea: 0.5714285714285714
Specifitatea: 0.3333333333333333
Scor F1: 0.5333333333333333
```



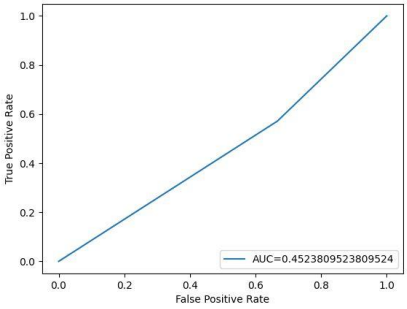

5.

```
True Positive: 4
True Negative: 3
False Positive: 3
False Negative: 3

Acuratetea: 0.5384615384615384
Precizia: 0.5714285714285714
Sensibilitatea: 0.5714285714285714
Specifitatea: 0.5
Scor F1: 0.5714285714285714
```



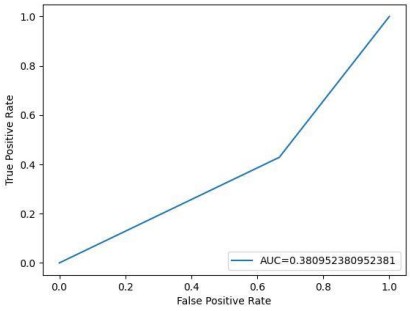

6.

```
True Positive: 4
True Negative: 2
False Positive: 4
False Negative: 3

Acuratetea: 0.46153846153846156
Precizia: 0.5
Sensibilitatea: 0.5714285714285714
Specifitatea: 0.3333333333333333
Scor F1: 0.5333333333333333
```



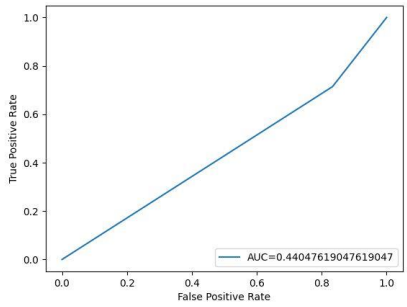

7.

```
True Positive: 3
True Negative: 2
False Positive: 4
False Negative: 4

Acuratetea: 0.38461538461538464
Precizia: 0.42857142857142855
Sensibilitatea: 0.42857142857142855
Specifitatea: 0.3333333333333333
Scor F1: 0.42857142857142855
```



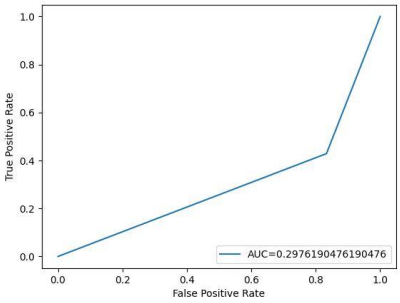
8.

```
True Positive: 5
True Negative: 1
False Positive: 5
False Negative: 2

Acuratetea: 0.46153846153846156
Precizia: 0.5
Sensibilitatea: 0.7142857142857143
Specifitatea: 0.16666666666666666
Scor F1: 0.588235294117647
```




9.

```
True Positive: 3
True Negative: 1
False Positive: 5
False Negative: 4

Acuratetea: 0.3076923076923077
Precizia: 0.375
Sensibilitatea: 0.42857142857142855
Specifitatea: 0.16666666666666666
Scor F1: 0.39999999999999997
```




Algoritmul are cele mai rotunjite masuratori pentru k=5, toate acestea fiind mai mari sau egali cu 5, pana si valoarea AUC este cea mai mare. Dar cand vine vorba de detectia focului, esuarea detectiei acestuia (False Negative) poate avea consecinte mai grave decat detectia acestuia acolo unde nu se afla (False Positive), astfel Sensibilitatea ar trebui sa fie maximizata.

Cele mai mari valori sunt gasite pentru K=2 si K=8, iar intre aceste doua, K=2 pare a fi mai bun din celelalte puncta de vedere, mai mult, valoarea AUC a iteratiei este a doua cea mai mare, Specifitatea pierzand cel mai mult, sa speram ca asta nu conduce la un caz al “baiatului care a strigat lup”.

Valoriile medii ale masuratoriilor sunt:

Acuratetea – 0.452

Precizia – 0.491

Sensibilitatea – 0.571

Specifitatea – 0.312

Scor F1 – 0.527

Aceste rezultate sunt consistente, desi mult mai mici decat, rezultatele antrenarii: Sensibilitatea are valoriile cele mai mari, iar Specifitatea le are mai mici.

Algoritmul knn pare sa favorizeze Sensibilitatea cand vine vorba de detectia focului, desi, Specifitatea fiind defavorizata , nu as putea spuna ca “detecteaza” focul pe cat “nu il rateaza”, vede focul in aproape toate imaginile in care se afla, dar il vede si acolo unde nu este.

In teorie, asta ar insemna ca multe tragedii ar fi evitate, dar multe resurse vor fi pierdute pe alarme false, resurse care ar fi putut fi folosite la aparitia unei tragedii reale. Ultimul cui batut in sicriul acestui algoritm este valoarea AUC, chiar si cea mai mare este mai mica de 0.6, aceeia fiind 0.536, ceea ce inseamna ca acest model se respinge.

In concluzie, algoritmul KNearestNeighbour este prea ineficient pentru a putea fi folosit la detectia focului.

Nota: in folderul ‘results’ s-au salvat screenshot-uri cu toate masuratorile la antrenare si la testare, precum si rezultatele predictiilor imaginiilor pentru iteratia K=2.

Ce as putea comenta asupra imaginilor este ca, in mare parte, algoritmul a avut probleme detectand pozele de calitate mai proasta (si nu ma refer la densitatea de pixeli, ci la alte efecte cum ar fi cele de lumina prezente in imaginile de test 4 si 5), deci, performanta algoritmului poate sa se imbunatateasca odata cu dezvoltariile tehnologice in domeniul fotografiilor.