**Tema 1 TIA**

# Antrenarea unui algoritm pentru clasificarea radiografiilor cu oase sănătoase și oase fracturate

Gălățanu Alexia Ștefania

Grupa 334AA

# Cuprins

## Prezentare generală

În ansamblu, acest algoritm antrenează și evaluează un model de clasificare a unor imagini îndouă categorii. Astfel se vor distinge radiografiile cu oase sănătoase de cele cu oase fracturate. Pentru realizarea acestei clasificări am utilizat algoritmul Random Forest.

Automatizarea examinării radiografiilor pentru a distinge între oasele sănătoase și cele fracturate este un aspect esențial în diagnosticul asistat de computer în domeniul imagisticii medicale. Acest proces poate ajuta la eficientizarea evaluării radiografiilor, facilitând astfel diagnosticul și tratamentul pacienților.

Obiectivul proiectului este acela de a dezvolta un algoritm care să ofere o performantă cât mai bună în distingerea radiografiilor cu oase sanătoase de cele cu oase fracturate. Utilitatea practică a unei astfel de aplicații este de a sprijinii specialiștii în diagnosticul precis al fracturilor și reducerea sarcinilor personalului medical.

## Setul de date

Setul de date utilizat este construit din 2 surse găsite pe www.kaggle.com și www.FracAtlas.com. Integrarea acestor două surse de inspirație a condus la dezvoltarea unui set de date cuprinzător și echilibrat. Acesta cuprinde 1300 de imagini în format JPG, împărțite în 3 fișiere:

• train( 930 de imagini ~72% ) pentru training

• val( 240 de imagini ~18% ) pentru validare

• test( 130 de imagini ~10%) pentru testare

fiecare dintre acestea conținând alte două fișiere (Not_fractured și Fractured).

Declararea căilor fișierelor:

```
data_folder = './train'
test_folder = './test'
validation_folder = './val'
```

## Biblioteci utilizate

• **OS** - manipularea sistemului de operare și gestionarea eficientă a fișierelor;

• **OpenCV** - funcționalități avansate pentru procesarea imaginilor, inclusiv încărcarea,

redimensionarea și transformările acestora;

• **Numpy** - manipularea datelor în format matrice și vectori;

• **scikit-learn (sklearn)** - instrumente oferite pentru construirea modelului : RandomForestClassifier pentru clasificare și StandardScaler pentru scalarea datelor.

• **matplotlib.pyplot** permite vizualizarea grafică a rezultatelor

## Încărcarea și prelucrarea imaginilor

Funcția pentru incărcarea imaginilor din fișierul train:

```
def load_train_images_from_folder(folder, target_shape=None):
    images = []
    labels = []
    for subfolder in os.listdir(folder):
        subfolder_path = os.path.join(folder, subfolder)
        if os.path.isdir(subfolder_path):
            for filename in os.listdir(subfolder_path):
                if filename.endswith('.jpg'):
                    img = cv2.imread(os.path.join(subfolder_path, filename))
                    if img is not None:

                        if target_shape is not None:
                            img = cv2.resize(img, target_shape)
                        images.append(img)
                        labels.append(subfolder)
                        print('Labels \n', labels)
                    else:
                        print(f"Warning: Unable to load {filename}")
    return images, labels
```

Analog au fost create funcțiile pentru încărcarea imaginilor în fișierul val și test.

```
> def load_test_images_from_folder(folder, target_shape=None): ···

> def load_validation_images_from_folder(folder, target_shape=None): ···
```

## Prelucrarea imaginilor:

- redimensionate la 200x200 pixeli;
- convertirea în alb negru;
- datele sunt standardizate folosind StandardScaler

```
# Load validation images and labels from the 'val' folder
validation_images, validation_labels = load_validation_images_from_folder(validation_folder, target_shape=

# Combine training and validation data
images += validation_images
labels += validation_labels

# Load test images and labels from the 'test' folder
test_images, test_labels = load_test_images_from_folder(test_folder, target_shape=(200, 200))

# Convert labels to binary (0 or 1)
labels_binary = [1 if label == 'Not_fractured' else 0 for label in labels]
test_labels_binary = [1 if label == 'Not_fractured' else 0 for label in test_labels]


# Reshape the images and convert them to grayscale
image_data = [cv2.cvtColor(image, cv2.COLOR_BGR2GRAY).flatten() for image in images]
test_image_data = [cv2.cvtColor(image, cv2.COLOR_BGR2GRAY).flatten() for image in test_images]

# Convert the list of 1D arrays to a 2D numpy array
image_data = np.array(image_data)
test_image_data = np.array(test_image_data)

# Scale the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(image_data)
scaled_test_data = scaler.transform(test_image_data)
```

## Modelul de învățare automată

Am utilizat *Random Forest Classifier* pentru sarcina de clasificare a imaginilor.

```
# Split the data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(scaled_data, labels_binary, test_size=0.2, random_state=42)

# Train a Random Forest model
random_forest_model = RandomForestClassifier(random_state=42, n_estimators=100)
random_forest_model.fit(X_train, y_train)
```

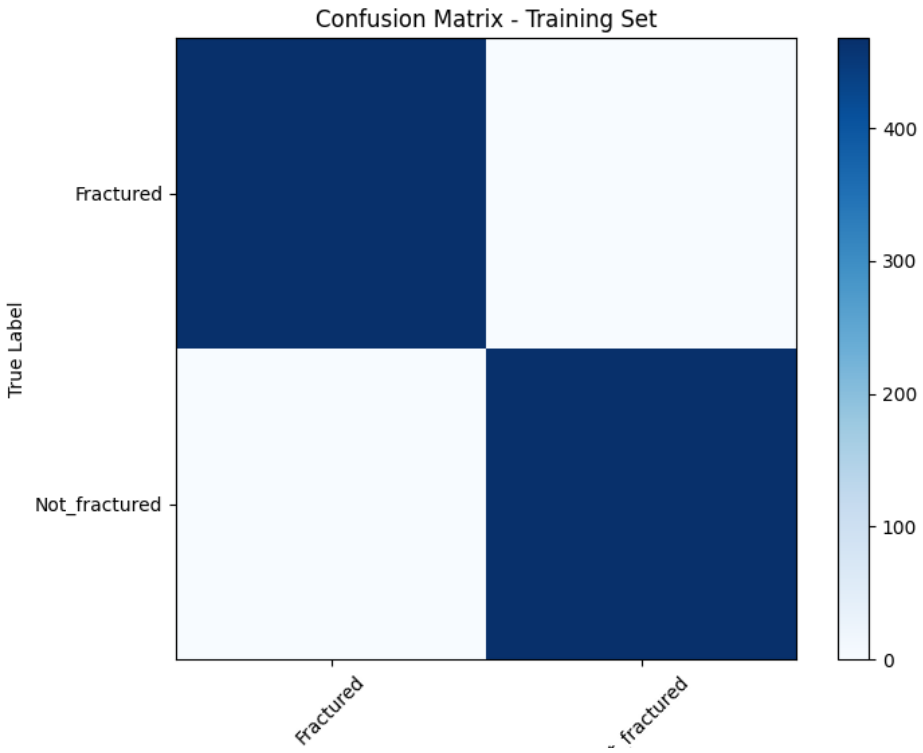
În acest pas, am creat un obiect RandomForestClassifier cu 100 de arbori de decizie și am antrenat modelul pe datele de antrenare (X_train și y_train).

## Evaluarea modelului

```
Test Image 1 - ==Fractured==
Test Image 2 - ==Not_fractured==
Test Image 3 - ==Not_fractured==
Test Image 4 - ==Fractured==
Test Image 5 - ==Fractured==
Test Image 6 - ==Fractured==
Test Image 7 - ==Fractured==
Test Image 8 - ==Fractured==
Test Image 9 - ==Fractured==
Test Image 10 - ==Fractured==
Test Image 11 - ==Fractured==
Test Image 12 - ==Fractured==
Test Image 13 - ==Fractured==
Test Image 14 - ==Not_fractured==
Test Image 15 - ==Fractured==
Test Image 16 - ==Fractured==
Test Image 17 - ==Fractured==
Test Image 18 - ==Not_fractured==
Test Image 19 - ==Not_fractured==
Test Image 20 - ==Fractured==
Test Image 21 - ==Fractured==
Test Image 22 - ==Fractured==
Test Image 23 - ==Fractured==
Test Image 24 - ==Not_fractured==
Test Image 25 - ==Not_fractured==
Test Image 26 - ==Not_fractured==
Test Image 27 - ==Not_fractured==
Test Image 28 - ==Fractured==
Test Image 29 - ==Fractured==
Test Image 30 - ==Fractured==
Test Image 31 - ==Fractured==
Test Image 32 - ==Fractured==
Test Image 33 - ==Fractured==
Test Image 34 - ==Fractured==
Test Image 35 - ==Fractured==
Test Image 36 - ==Fractured==
Test Image 37 - ==Fractured==
Test Image 38 - ==Fractured==
Test Image 39 - ==Fractured==
Test Image 40 - ==Fractured==
Test Image 41 - ==Not_fractured==
Test Image 42 - ==Fractured==
Test Image 43 - ==Fractured==
Test Image 44 - ==Fractured==
Test Image 45 - ==Fractured==
Test Image 46 - ==Fractured==
Test Image 47 - ==Fractured==
Test Image 48 - ==Fractured==
Test Image 49 - ==Fractured==
Test Image 50 - ==Fractured==
Test Image 51 - ==Fractured==
Test Image 52 - ==Fractured==
Test Image 53 - ==Fractured==
Test Image 54 - ==Fractured==
Test Image 55 - ==Fractured==
Test Image 56 - ==Fractured==
Test Image 57 - ==Fractured==
Test Image 58 - ==Fractured==
Test Image 59 - ==Not_fractured==
Test Image 60 - ==Not_fractured==
Test Image 61 - ==Fractured==
Test Image 62 - ==Not_fractured==
Test Image 63 - ==Fractured==
Test Image 64 - ==Fractured==
Test Image 65 - ==Fractured==
Test Image 66 - ==Not_fractured==
Test Image 67 - ==Not_fractured==
Test Image 68 - ==Fractured==
Test Image 69 - ==Not_fractured==
Test Image 70 - ==Not_fractured==
Test Image 71 - ==Not_fractured==
Test Image 72 - ==Not_fractured==
Test Image 73 - ==Not_fractured==
Test Image 74 - ==Not_fractured==
Test Image 75 - ==Not_fractured==
Test Image 76 - ==Fractured==
```

```
Test Image 77 - ==Fractured==
Test Image 78 - ==Fractured==
Test Image 79 - ==Fractured==
Test Image 80 - ==Fractured==
Test Image 81 - ==Fractured==
Test Image 82 - ==Fractured==
Test Image 83 - ==Fractured==
Test Image 84 - ==Fractured==
Test Image 85 - ==Not_fractured==
Test Image 86 - ==Fractured==
Test Image 87 - ==Not_fractured==
Test Image 88 - ==Not_fractured==
Test Image 89 - ==Fractured==
Test Image 90 - ==Not_fractured==
Test Image 91 - ==Fractured==
Test Image 92 - ==Fractured==
Test Image 93 - ==Fractured==
Test Image 94 - ==Not_fractured==
Test Image 95 - ==Fractured==
Test Image 96 - ==Not_fractured==
Test Image 97 - ==Not_fractured==
Test Image 98 - ==Fractured==
Test Image 99 - ==Fractured==
Test Image 100 - ==Not_fractured==
Test Image 101 - ==Not_fractured==
Test Image 102 - ==Fractured==
Test Image 103 - ==Not_fractured==
Test Image 104 - ==Not_fractured==
Test Image 105 - ==Not_fractured==
Test Image 106 - ==Fractured==
Test Image 107 - ==Fractured==
Test Image 108 - ==Not_fractured==
Test Image 109 - ==Not_fractured==
Test Image 110 - ==Not_fractured==
Test Image 111 - ==Not_fractured==
Test Image 112 - ==Not_fractured==
Test Image 113 - ==Fractured==
Test Image 114 - ==Not_fractured==
Test Image 115 - ==Fractured==
Test Image 116 - ==Fractured==
Test Image 117 - ==Not_fractured==
Test Image 118 - ==Not_fractured==
Test Image 119 - ==Fractured==
Test Image 120 - ==Fractured==
Test Image 121 - ==Fractured==
Test Image 122 - ==Fractured==
Test Image 123 - ==Fractured==
Test Image 124 - ==Fractured==
Test Image 125 - ==Fractured==
Test Image 126 - ==Not_fractured==
Test Image 127 - ==Fractured==
Test Image 128 - ==Not_fractured==
Test Image 129 - ==Not_fractured==
Test Image 130 - ==Not_fractured==
```

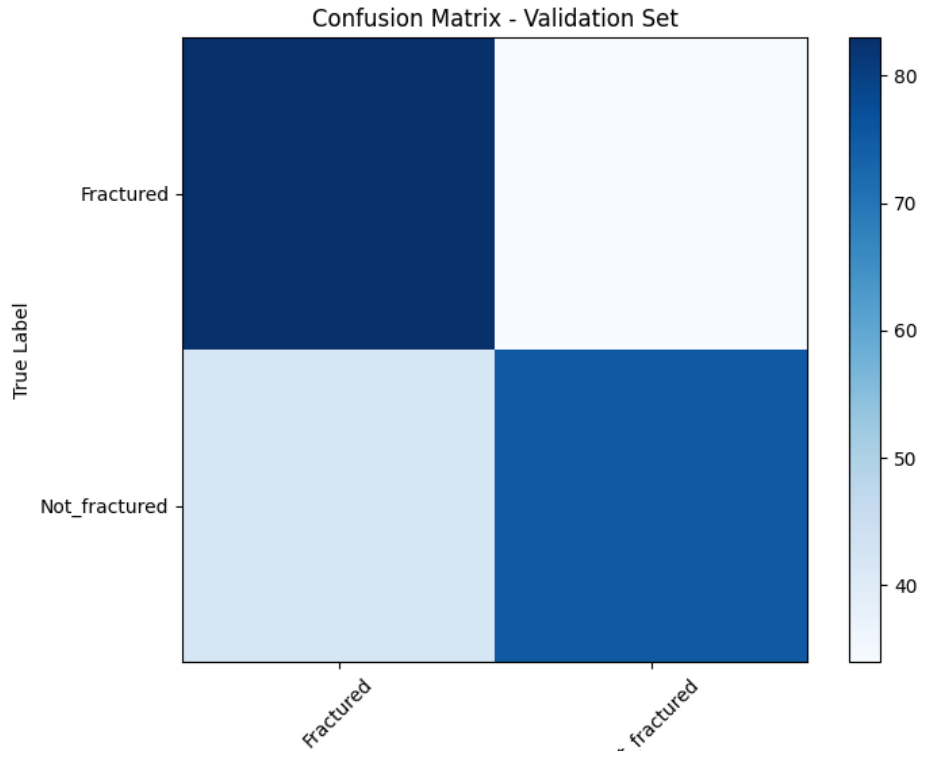
## Setul de antrenare




Acuratețea: 97.48% - indică că algoritmul a clasificat corect aproape toate datele din setul de antrenare. Această valoare poate sugera o învățare eficientă a caracteristicilor specifice ale acestui set particular. În ciuda acestei valori ridicate pe datele de antrenare, am conștientizat riscul de supraînvățare. Modelul ar putea să se adapteze excesiv la particularitățile setului de antrenare și să nu generalizeze bine pe date noi. Pentru a valida eficacitatea reală a algoritmului am efectuat modificări pe setul de date( am exclus imagini, am introdus imagini ), dar rezultatele au fost foarte apropriate, desi cantitatea și calitatea datelor din set suportase modificări semnificative.

## Setul de validare

```
Performance on Validation Set:
Accuracy: 0.6752
Precision: 0.6881
Recall: 0.6410
F1 Score: 0.6637
Confusion Matrix:
[[83 34]
 [42 75]]
```




Acuratețea: 67.52% - Procentul total de predicții corecte. În acest caz, 67.52% dintre cazuri au fost clasificate corect.

Precizia: 68.81% - Procentul de cazuri clasificate corect ca pozitive din totalul cazurilor clasificate ca pozitive.Procentul de imagini „Not_fractured” identificate corect este unul moderat.

Sensibilitate: 64.10% - Procentul de cazuri pozitive clasificate corect din totalul cazurilor pozitive reale.

Scor F1: 66.37% - O măsură a balansului între precizie și sensibilitate, utilă în cazul unor seturi de date neechilibrate.Este o performanță moderată.

Matricea de confuzie prezintă numărul de predicții corecte și incorecte făcute de model, împărțite în categoriile adevărat pozitive (TP), fals pozitive (FP), adevărat negative (TN) și fals negative (FN).

Analizând matricea de confuzie generată de evaluarea modelului pe setul de validare am constatat :

• 83 de adevărat negative (TN) (83 de cazuri în care algoritmul a identificat correct „Fractured”)

• 34 de fals pozitive (FP) (34 de cazuri în care algoritmul a identificat greșit „Not_fractured”)

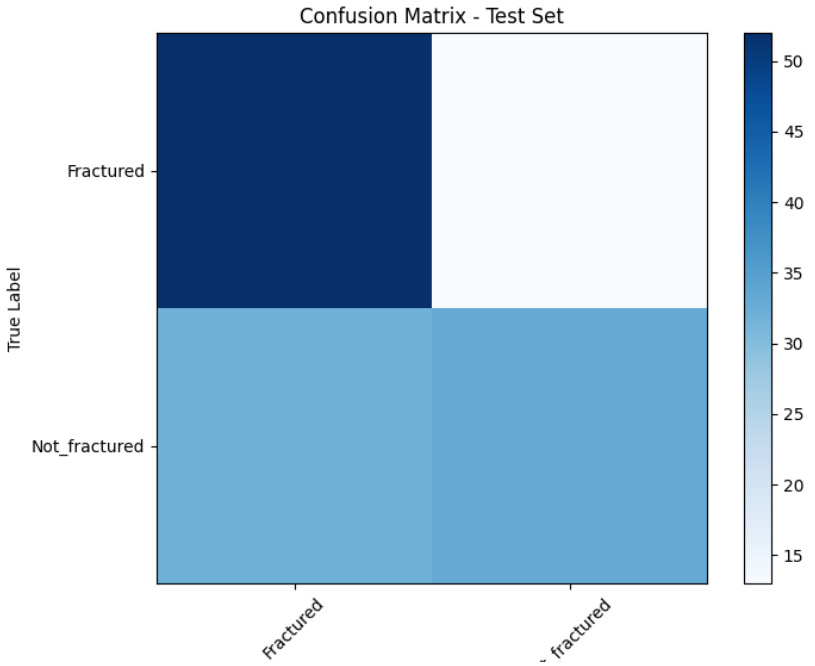
• 42 de fals negative (FN) (42 de cazuri în care algoritmul a identificat greșit „Fractured”)

• 75 de adevărat pozitive (TP) (75 de cazuri în care algoritmul a identificat corect „Not_fractured”).

Algoritmul are o performanță moderată pe setul de validare, cu un echilibru relative între identificarea corectă a cazurilor „Not_fractured” și ”Fractured”.

## Setul de testare

```
Performance on Test Set:
Accuracy: 0.6538
Precision: 0.7174
Recall: 0.5077
F1 Score: 0.5946
Confusion Matrix:
[[52 13]
 [32 33]]
```




Acuratețe: 65.38%

Precizie: 71.74% arată cât de precis este modelul atunci câbd prezice că un exemplu este „Not_fractured”. Aproape ¾ dintre predicțiile „Not_fractured” sunt corecte.

Sensibilitate: 50.77%

Scor F1: 59.46% indică echilibrul dintre precizie și sensibilitate. Rezultatul este unul moderat.

Analizând matricea de confuzie generată de evaluarea modelului pe setul de testare am constatat :

• 52 de adevărate negative (TN) – numărul de cazuri „Fractured” identificate correct ca „Fractured”

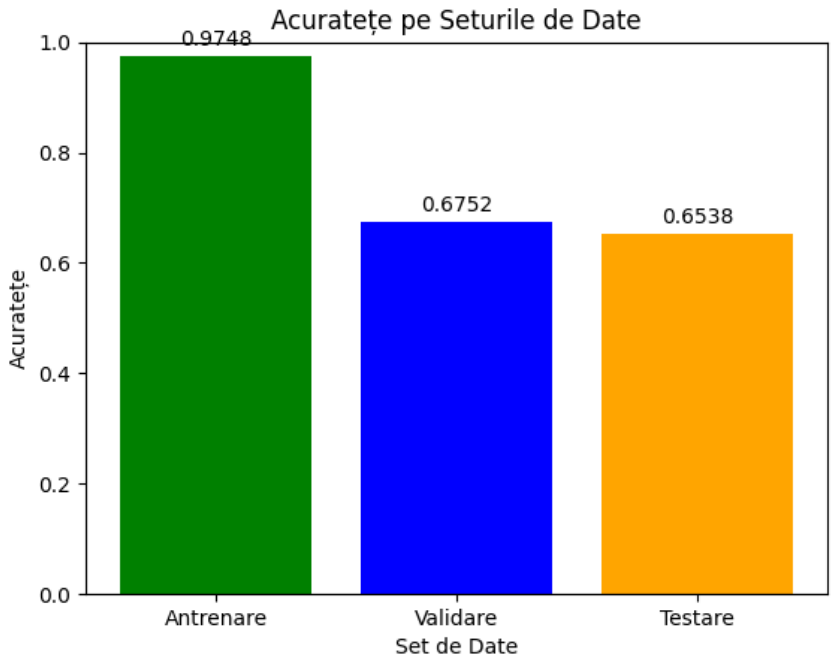
• 13 de fals pozitive (FP) – numărul de cazuri „Fractured” identificate greșit ca „Not_fractured”

• 32 de fals negative (FN) – numărul de cazuri „Not_fractured” identificate greșit ca „Fractured”

• 33 de adevărate pozitive (TP)- numărul de cazuri „Not_fractured” indentificate corect ca „Not_fractured”.

Algoritmul are o precizie relativ bună pe setul de testare în identificarea imaginilor cu radiografii sănătoase(„Not_fractured”). Sensibilitatea este mai mica, ceea ce indică că algoritmul ratează o proporție semnificativă dintre cazurile care erau în realitate „Not_fractured”.

### Reprezentarea grafică a performanțelor pe cele 3 seturi de date




## Concluzii

Discrepanța între valorile obținute pentru acuratețea pe setul de antrenare și cele obținute pe seturile de validare și testare indică o posibilă necesitate de optimizare a modelului și ajustare a setului de date. Întrucât utilizarea algoritmul Random Forest a oferit un rezultat mai favorabil pentru tema și setul de date ales față de alte variante precum K-Means, posibilele îmbunătățiri se concentrează către ajustarea setului de date. Pentru acest aspect, ar fi de preferat ca aplicarea modelului să se facă pe un set de date cu imagini care surprind o singură regiune corporală ( de exemplu: imagini cu radiografii ale membrului superior drept ), astfel se reduce semnificativ numărul de caracteristici.

Analiza matricelor de confuzie obținute pe seturile de antrenare, validare și testare dezvăluie informații semnificative pentru performanța modelului. În nicio situație, numarul cazurilor fals negative și fals pozitive nu îl depășește pe cel al cazurilor adevărate negative și adevărate pozitive. Se observă că modelul este mai probabil să identifice corect cazurile de oase sănătoase (Not_fractured).

Modelul dezvoltat poate avea potențial aplicabil în domeniul medical pentru clasificarea imaginilor cu radiografii ale oaselor sănătoase și fracturate. Cu ajustările corespunzătoare și validarea suplimentară, modelul ar putea oferi sprijin în diagnosticarea fracturilor osoase, facilitând procesul medical și reducând timpul de evaluare. Deoarece datele pot evolua în timp, adaptările la noile date sau la schimbările efetuate în distribuția datelor sunt necesare pentru menținerea unei performanțe ridicate.