Gălățanu Alexia-Ștefania
Grupa 334AA

Tema 2 TIA

Clasificarea imaginilor cu radiografii ale oaselor sănătoase și fracturate

Prezentare generală

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 și etapa de preprocesare

Dezvoltarea acestei aplicații a avut ca prima etapă colectarea și preprocesarea datelor. În acest sens, setul de date utilizat este construit din imagini ale două seturi de date găsite pe www.kaggle.com și www.FracAtlas.com. Integrarea acestor două surse de inspirație a condus la dezvoltarea unui set de date cuprinzător, reprezentat de un fișier denumit „dataset” ce conține 2 fișiere („Fractured” și „Not_fractured”), fiecare cuprinzând câte 650 imagini în format JPG. În etapa de preprocesare a datelor m-am asigurat că toate imaginile pe care le-am selectat pentru varianta finală a setului de date sunt clare, în format JPG, relevante pentru tema aleasă și că nu există un dezechilibru între clase.

În codul aplicației, responsabilitatea împărțirii setului de date în seturi de antrenare și testare aparține apelului funcției **train_test_split** din biblioteca scikit-learn. Împărțirea datelor se face în urma apelului acestei funcții:

```
# Load your dataset
dataset_folder = "./dataset"
X, y, class_mapping = load_dataset(dataset_folder)
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Această ultimă linie din secvența de cod de mai sus preia matricea de caracteristici **X** și vectorul de etichete **y** și le împarte în seturi de antrenare (**X_train** și **y_train**) și testare (**X_test** și **y_test**). Argumentul **test_size=0.2** indică faptul că 20% din date vor fi utilizate pentru testare, iar restul pentru antrenare. Argumentul **random_state=42** furnizează o valoare pentru reproducerea împărțirii datelor, astfel încât aceasta să fie aceeași în fiecare rulare.

Gălățanu Alexia-Ștefania
Grupa 334AA

Biblioteci utilizate:

Numele bibliotecii	Utilitate
os	Manipularea sistemului de operare și gestionarea eficientă a fișierelor
numpy (np)	Lucrul eficient cu matrici și vectori, operații necesare pentru manipularea datelor sub formă de imagini și pentru operații matematice
sklearn.model_selection.train_test_split	Împărțirea setului de date în seturi de antrenare și testare
sklearn.neighbors.KNeighborsClassifier	Implementarea algoritmului k-Nearest Neighbors (k-NN) pentru clasificare. Este un algoritm de învățare supervizată bazat pe deea că obiectele asemănătoare se găsesc în proximitatea spațială.
sklearn.naive_bayes.GaussianNB	Implementarea clasificatorului Naive Bayes (Gaussian) pentru clasificare. Este un algoritm bazat pe teorema Bayes, care descrie probabilitatea condiționată a unui eveniment, dat fiind un alt eveniment.
sklearn.metrics.accuracy_score, sklearn.metrics.classification_report	Calculul matricilor pentru evaluarea performanței modelelor, cum ar fi acuratețea.
skimage.io.imread, skimage.transform.resize	Încărcarea imaginilor și redimensionarea acestora la dimensiunile dorite. Aceste funcții sunt parte a bibliotecii scikit-image.
matplotlib.pyplot	Permite vizualizarea grafică a rezultatelor.
pandas (pd)	Manipularea, analiza și salvarea (într-un fișier CSV, în cazul acestei aplicații) datelor.

Algoritmi propuși pentru problematica aleasă

Aplicația creată folosește algoritmii de învățare automată k-Nearest Neighbors și Naive Bayes pentru a clasifica imagini cu radiografii în două categorii: „Fractured” și „Not_fractured”. În acest context, algoritmii k-Nearest Neighbors (k-NN) și Naive Bayes sunt utili din următoarele motive: Algoritmul k-NN este un algoritm simplu și intuitiv. Rezultatele generate sunt ușor de interpretat, bazându-se pe principiul că punctele similare din spațiul de caracteristici se grupează împreună. În ciuda antrenării mai îndelungate, etapa de testare este rapidă, deoarece necesită doar căutarea vecinilor apropiați în spațiul de caracteristici.

Algoritmul Naive Bayes funcționeză bine, dat fiind faptul că setul de date este unul cu o cantitate relativ mică de date. Este eficient în gestionarea relațiilor dintre diferitele atribute ale imaginilor cu radiografii osoase, chiar dacă cantitatea de date nu este foarte bogată. În plus,

este robust în fața atributelor irelevante, făcându-l potrivit pentru situații în care anumite caracteristici pot fi prezente sau absente în mod aleatoriu.

Există mai mulți algoritmi de învățare automata care pot oferi performanțe mai bune decât K-Nearest Neighbors (k-NN) și Naive Bayes pentru problematica tratată, mai ales că discutăm despre aplicabilitate în imagistică medicală, ce ar presupune un set de date mult mai amplu.

Complexitatea poate crește din mai multe motive precum: Radiografiile pot avea o mare varietate în ceea ce privește calitatea, rezoluția și condițiile de iluminare. Aceasta poate face ca identificarea caracteristicilor relevante să fie o sarcină mai dificilă pentru algoritmi mai simpli. Imaginile medicale pot avea dimensiuni ridicate, ceea ce înseamnă că un număr mare de caracteristici trebuie luate în considerare în procesul de clasificare. Algoritmii mai avansați pot gestiona mai bine această dimensionalitate ridicată și pot extrage caracteristici semnificative. Astfel, algoritmi de învățare automată precum CNN, SVM sau Random Forest pot oferi performanțe mai bune pentru tema aleasă față de cele oferite de algoritmii k-NN și Naive Bayes.

Procesul de Antrenare:

1. Încărcarea și Prelucrarea Datelor:
   Imaginile cu radiografii sunt încărcate și prelucrate pentru a standardiza dimensiunea și pentru a le converti la tonuri de gri. Deoarece nu am nevoie de informații despre culoare, verific dacă imaginea are 3 dimensiuni, (3 canale, adică în format color), în caz afirmativ, o convertesc la scală de gri.
   Mai apoi, imaginile sunt convertite într-o singură dimensiune.

```python
# Function to load images and labels from the dataset
def load_dataset(root_folder):
    images = []
    labels = []
    class_mapping = {} # To map class names to numeric labels
    for class_label, class_name in enumerate(os.listdir(root_folder)):
        class_mapping[class_label] = class_name
        class_folder = os.path.join(root_folder, class_name)

        for filename in os.listdir(class_folder):
            image_path = os.path.join(class_folder, filename)
            image = imread(image_path)


            if image.ndim == 3:
                image = np.mean(image, axis=2) # Convert RGB image to grayscale


            image = resize(image, (64, 64), anti_aliasing=True) # Adjust image size as needed

            # print("Before flattening:", image.shape, "After flattening:", image.flatten().shape)


            images.append(image.flatten()) # Flatten the image
            labels.append(class_label)
    return np.array(images), np.array(labels), class_mapping
```

Gălățanu Alexia-Ștefania
Grupa 334AA

2. Split-ul datelor:
   Setul de date este divizat într-un set de antrenare și unul de testare, asigurându-se că modelele sunt testate pe date necunoscute.

```
# Load your dataset
dataset_folder = "./dataset"
X, y, class_mapping = load_dataset(dataset_folder)
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

3. Antrenarea algoritmului k-NN
   Modelul este antrenat să învețe caracteristicile asociate imaginilor cu radiografii sănătoase și fracturate.

```
# Initialize k-NN classifier
knn_classifier = KNeighborsClassifier(n_neighbors=3)
# Train the model
knn_classifier.fit(X_train, y_train) # Make predictions on the test set
y_test_pred_knn = knn_classifier.predict(X_test)
# Save results and test images for k-NN
save_results('knn', y_test, y_test_pred_knn, class_mapping)
probabilities_knn = knn_classifier.predict_proba(X_test)
save_test_images('knn', X_test, y_test, y_test_pred_knn, class_mapping, probabilities_knn)
```

4. Antrenarea algoritmului Naive Bayes:
   Modelul Naive Bayes, de tip Gaussian, este antrenat pentru a înțelege distribuția probabilităților și a face predicții pe baza independenței condiționate a atributelor.

```
# Example 2: Naive Bayes (Gaussian Naive Bayes) # Initialize Naive Bayes classifier
nb_classifier = GaussianNB()
# Train the model
nb_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_test_pred_nb = nb_classifier.predict(X_test)

# Save results and test images for Naive Bayes
save_results('naive_bayes', y_test, y_test_pred_nb, class_mapping)
probabilities_nb = nb_classifier.predict_proba(X_test)
save_test_images('naive_bayes', X_test, y_test, y_test_pred_nb, class_mapping, probabilities_nb)
```

Gălățanu Alexia-Ștefania
Grupa 334AA

Procesul de testare:

1. Evaluarea Performanței:
   Performanța modelelor este evaluată pe setul de testare, măsurându-se acuratețea, care indică proporția de predicții corecte.

```
# Acuratețe pentru k-NN
accuracy_knn_train = accuracy_score(y_train, knn_classifier.predict(X_train))

# Acuratețe pentru Naive Bayes
accuracy_nb_train = accuracy_score(y_train, nb_classifier.predict(X_train))

# Diagrama pentru acuratețea pe datele de test și antrenament pentru k-NN și Naive Bayes
labels = ['k-NN Test', 'k-NN Train', 'Naive Bayes Test', 'Naive Bayes Train']
accuracies = [accuracy_knn, accuracy_knn_train, accuracy_nb, accuracy_nb_train]

plt.bar(labels, accuracies, color=['green', 'lightgreen', 'blue', 'lightblue'])
plt.ylabel('Acuratețe')
plt.title('Acuratețe clasificare pe date de test și antrenament')
plt.show()
```

2. Salvarea Rezultatelor și Vizualizare:
   Rezultatele, inclusiv predicțiile și etichetele reale, sunt salvate pentru a oferi o imagine detaliată asupra modului în care modelele se comportă pe datele de testare.
   Pentru o privire de ansamblu, predicțiile și etichetele reale sunt trecute 2 două fișiere, pentru fiecare dintre algoritmi, cu extensia .csv .

```
# Function to save results to a CSV file
def save_results(algorithm, y_true, y_pred, class_mapping):
    results_df = pd.DataFrame({'True Label': [class_mapping[label] for label in y_true],
    'Predicted Label': [class_mapping[label] for label in y_pred]})
    results_folder = './results'
    if not os.path.exists(results_folder):
        os.makedirs(results_folder)

    results_file = os.path.join(results_folder, f'{algorithm}_results.csv')
    results_df.to_csv(results_file, index=False)
    print(f'Results saved for {algorithm} at {results_file}')
```

Gălățanu Alexia-Ștefania
Grupa 334AA

	A	B	C
1	True Label	Predicted Label	
2	Fractured	Fractured	
3	Not_fracture	Not_fractured	
4	Fractured	Fractured	
5	Not_fracture	Not_fractured	
6	Not_fracture	Not_fractured	
7	Fractured	Not_fractured	
8	Not_fracture	Fractured	
9	Fractured	Fractured	
10	Not_fracture	Fractured	
11	Fractured	Fractured	
12	Not_fracture	Fractured	
13	Not_fracture	Fractured	
14	Fractured	Fractured	
15	Not_fracture	Not_fractured	
16	Not_fracture	Not_fractured	
17	Fractured	Fractured	
18	Not_fracture	Not_fractured	
19	Not_fracture	Not_fractured	
20	Fractured	Not_fractured	
21	Fractured	Fractured	

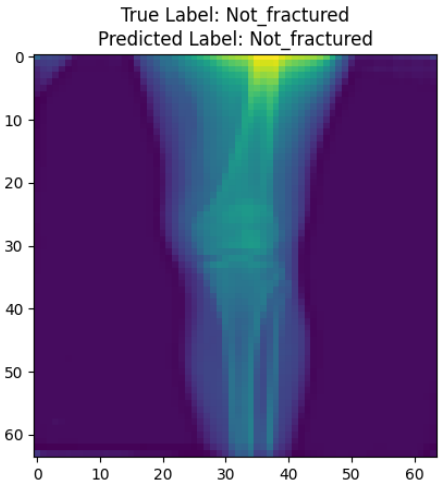
knn_results

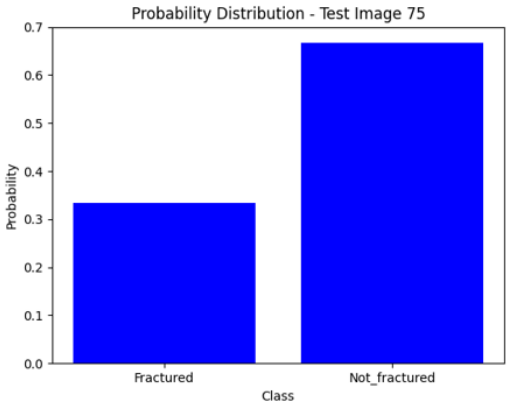
	A	B	C
1	True Label	Predicted Label	
2	Fractured	Fractured	
3	Not_fracture	Not_fractured	
4	Fractured	Not_fractured	
5	Not_fracture	Not_fractured	
6	Not_fracture	Not_fractured	
7	Fractured	Fractured	
8	Not_fracture	Fractured	
9	Fractured	Fractured	
10	Not_fracture	Not_fractured	
11	Fractured	Fractured	
12	Not_fracture	Not_fractured	
13	Not_fracture	Not_fractured	
14	Fractured	Not_fractured	
15	Not_fracture	Not_fractured	
16	Not_fracture	Fractured	
17	Fractured	Fractured	
18	Not_fracture	Not_fractured	
19	Not_fracture	Not_fractured	
20	Fractured	Fractured	
21	Fractured	Not_fractured	

naive_bayes_results

Exemple din fișierul results:

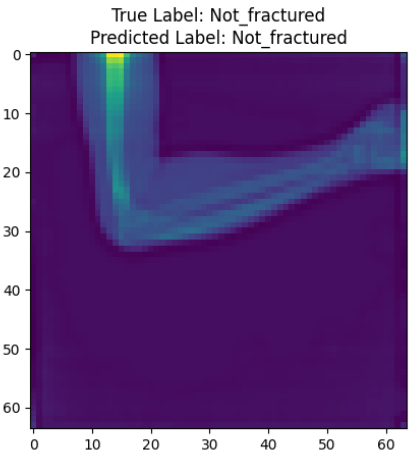
knntest_image75.png :

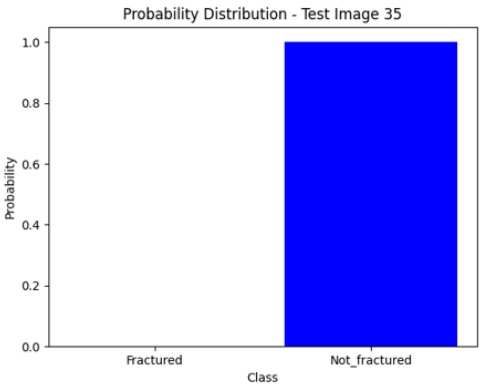




Gălățanu Alexia-Ștefania
Grupa 334AA

naive_bayestest_image35.png:





Avantaje și implicații:

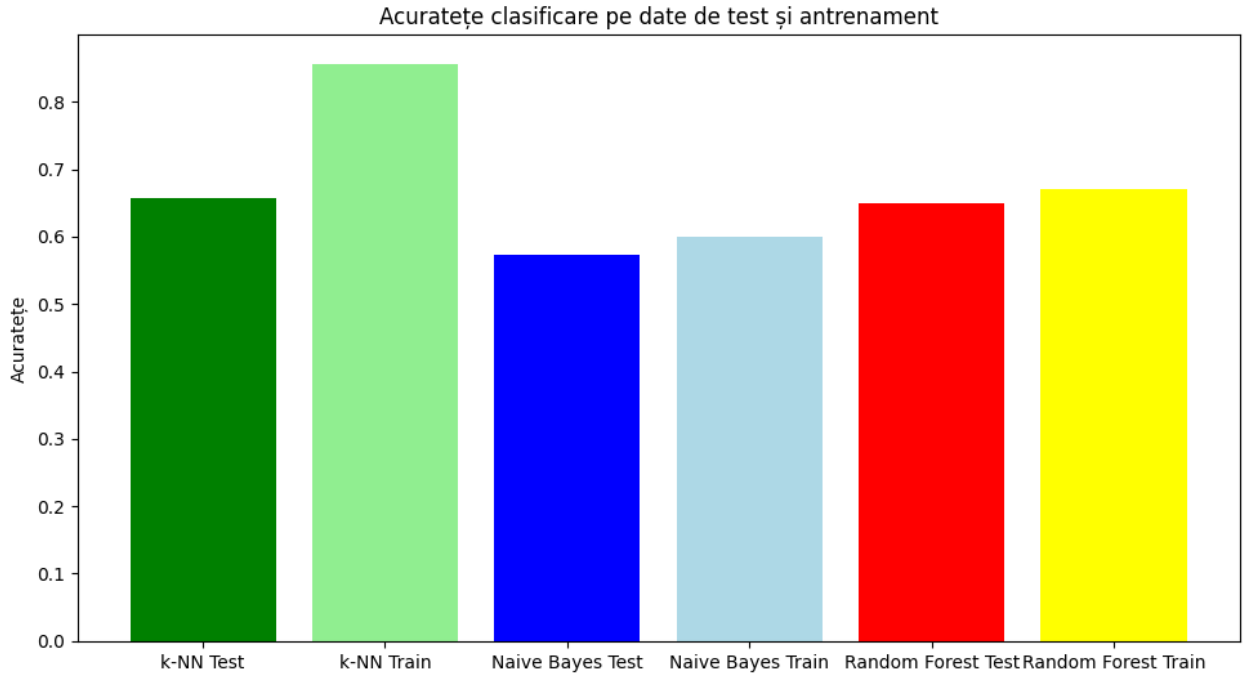
Clasificare rapidă: Modelele antrenate pot furniza un instrument de diagnostic rapid și fiabil, susținând medicul în procesul decizional.

Optimizarea resurselor: Aplicația urmărește optimizarea resurselor medicale, permițând distribuirea mai eficientă a atenției medicilor.

Gălățanu Alexia-Ștefania
Grupa 334AA

Concluzii:

În concluzie, algoritmii k-NN și Naive Bayes demonstrează aplicabilitate în clasificarea imaginilor cu radiografii osoase. Cu toate acestea, continuarea cercetărilor și îmbunătățirile constante sunt esențiale pentru a maximiza precizia și utilitatea algoritmilor de învățare automată utilizați în practica medicală. Integrarea acestor algoritmi în domeniul medical ar putea reprezenta un pas semnificativ către îmbunătățirea procesului de diagnosticare și management al afecțiunilor osoase.



În privința rezultatelor obținute în urma testării, algoritmul k-NN oferă o acuratețe mai mare la testare și antrenare față de algoritmul Naive Bayes.

În comparație cu rezultatele obținute în urma testării aplicației realizate în cadrul temei 1, pe același set de date, acuratețea la testare este apropiată între algoritmii k-NN și Random Forest, în timp ce, la antrenare, acuratețea data de algoritmul Random Forest nu se remarcă a fi mai slabă sau mai puternică decât cele oferite de utilizarea algoritmilor k-NN și Naive Bayes.

Gălățanu Alexia-Ștefania
Grupa 334AA

Diferențele dintre performanțele rezultate în urma utilizării algoritmilor k-NN și Naive Bayes sunt generate de următoarele motive:

- Diferențele dintre imaginile cu oase sănătoase și fracturate sunt complexe, iar algoritmul k-NN ar putea să le identifice mai bine.
- Valoarea de vecini aleasă pentru algoritmul k-NN (k=3) este adaptată pentru setul de date, număr ales pentru o performanță mai bună.

Așadar, pentru problematica propusă de mine, anume clasificarea imaginilor cu radiografii ale oaselor sănătoase și fracturate varianta cea mai potrivită de soluționare a algoritmilor de invatare automată pe care i-am abordat este algoritmul k-NN.