Mazilu Gabriela

Raport – Validarea clasificarii cu 2 algoritmi

Prezentarea temei:

Proiectul implementat realizeaza clasificarea imaginilor in doua categorii distincte: copii si adulti, prin antrenarea unui model bazat pe algoritmii K-NN si Naive Bayes.

Prezentarea setului de date si a etapei de preprocesare:

Setul de date este unul personalizat, astfel incat l-am structurat in 3 foldere principale: train (setul de antrenament), validate (setul de validare), test (setul de testare). Fiecare la randul lui contine 2 subfoldere ce indica etichetele clasificarii ('adults' , 'children').

Pentru clasa „children”:	Pentru clasa „adults”:
Imagini pentru antrenare: 1518	Imagini pentru antrenare: 1810
Imagini pentru validare: 325	Imagini pentru validare: 388
Imagini pentru testare: 325	Imagini pentru testare: 388

Această împărțire respectă distribuția de aproximativ 70% pentru antrenare, 15% pentru validare și 15% pentru testare. Astfel, nu a mai fost nevoie sa apelez la functia „train_test_split()” in cadrul implementarii codului.

Pentru o mai buna clasificare, setul de date ales contine exclusiv fetele persoanelor, acestea fiind decupate din restul imaginii. Deși setul de date conține imagini în tonuri de gri, procesul de încărcare a imaginilor folosește cv2.imread() fără flag-ul cv2.IMREAD_GRAYSCALE, pentru a gestiona un posibil canal suplimentar care apare în formatul specific al acestor imagini. Ulterior, imaginile sunt convertite în tonuri de gri folosind cv2.cvtColor(img, cv2.COLOR_BGR2GRAY). Acest pas asigură că imaginile sunt procesate corect pentru detectarea fețelor și extragerea caracteristicilor HOG.

Pentru preprocesare, am utilizat detectorul Haar Cascade pentru localizarea și extragerea fețelor din imagini, urmată de redimensionarea acestora la dimensiunea de 64x64 pixeli. De asemenea, am aplicat extracția de caracteristici utilizând metoda Histogram of Oriented Gradients (HOG), esențială pentru procesul de clasificare.

```python
def load_images_and_hog_features(folder_path):
    hog_features = []
    labels = []
    class_names = os.listdir(folder_path)

    for class_name in class_names:
        class_path = os.path.join(folder_path, class_name)
        if os.path.isdir(class_path):
            for filename in os.listdir(class_path):
                if filename.endswith('.png'):
                    img_path = os.path.join(class_path, filename)
                    img = cv2.imread(img_path)
                    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

                    # Detectarea fetelor
                    faces = face_cascade.detectMultiScale(gray, 1.1, 4)
                    for (x, y, w, h) in faces:
                        # Croirea imaginii pentru a include doar fata
                        face_roi = gray[y:y+h, x:x+w]
                        face_resized = cv2.resize(face_roi, (64, 64))

                        # Calcularea caracteristicilor HOG
                        features = hog(face_resized, orientations=9, pixels_per_cell=(8, 8),
                                       cells_per_block=(2, 2), visualize=False)
                        hog_features.append(features)
                        labels.append(class_name)
                        break  # Presupunem ca exista o singura fata relevanta per imagine
    return hog_features, labels
```

Aplicabilitatea algoritmilor

Am implementat si comparat doi algoritmi de clasificare: k-Nearest Neighbors (k-NN) si Naive Bayes. Ambii algoritmi fac parte din categoria algoritmilor bazati pe invatare supervizata si urmaresc acelasi lucru: determinarea clasei in care se incadreaza o noua instanta. In cazul aplicatiei implementate, atat k-NN, cat si Naive Bayes reusesc sa realizeze clasificarea imaginilor pentru distinctia intre copii si adulti, si isi dovedesc performanta printr-o acuratete considerabila.

Procesul de Antrenare si Validare

Pentru fiecare algoritm, setul de date a fost impartit in trei subseturi: antrenare, validare si testare. Am antrenat algoritmii pe setul de antrenare, iar ajustarile si optimizarile au fost realizate utilizand setul de validare.

```python
train_path = "C:\\Users\\Gabriela\\Desktop\\set de date\\train"
validate_path = "C:\\Users\\Gabriela\\Desktop\\set de date\\validate"
test_path = "C:\\Users\\Gabriela\\Desktop\\set de date\\test"


# Incarcarea datelor
X_train, y_train = load_images_and_hog_features(train_path)
X_val, y_val = load_images_and_hog_features(validate_path)
X_test, y_test = load_images_and_hog_features(test_path)
```

```python
# Crearea si antrenarea clasificatorului Naive Bayes
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
# Evaluarea clasificatorului Naive Bayes
y_pred_test_nb = nb_classifier.predict(X_test)
y_pred_train_nb = nb_classifier.predict(X_train)
```

```python
# Crearea si antrenarea clasificatorului KNN
knn_classifier = KNeighborsClassifier(n_neighbors=3)
knn_classifier.fit(X_train, y_train)
# Evaluarea clasificatorului KNN
y_pred_test_knn = knn_classifier.predict(X_test)
y_pred_train_knn = knn_classifier.predict(X_train)
```

Mazilu Gabriela

Procesul de Testare

Testarea a fost efectuata pe setul de testare, care nu a fost expus algoritmilor in timpul antrenarii modelului. Scopul testarii este de a evalua performanta reala a algoritmilor in conditii neprevazute si de a verifica capacitatea acestora de a generaliza pe date noi.

Rezultatele au fost salvate intr-un fisier .csv ce include eticheta reala si cea prezisa a datelor.

```
def save_results(algorithm, y_true, y_pred,results_folder):
    results_folder = 'C:\\Users\\Gabriela\\Desktop\\Anu 3 - sem 1\\TIA\\results'
    if not os.path.exists(results_folder):
        os.makedirs(results_folder)
    results_df = pd.DataFrame({'True Label': y_true, 'Predicted Label': y_pred})
    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}')
```

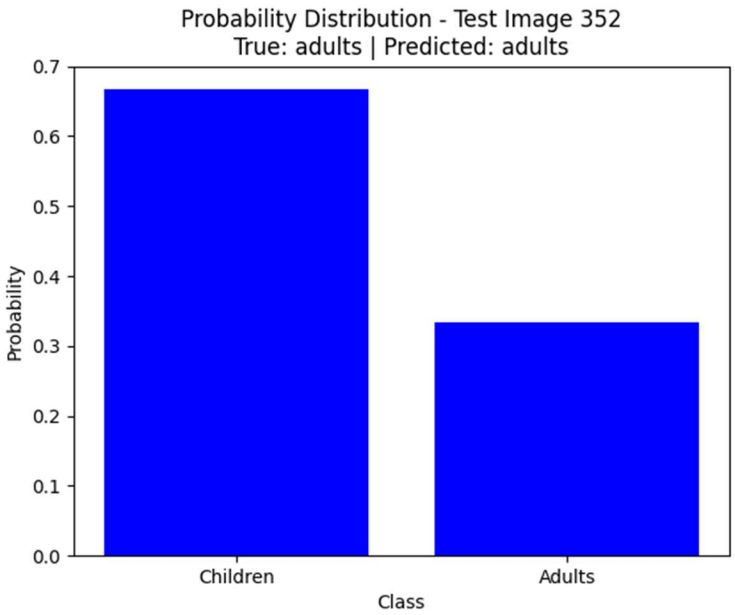
Am salvat, de asemenea, si Distributia Probabilitatii in folderul cu rezultate.

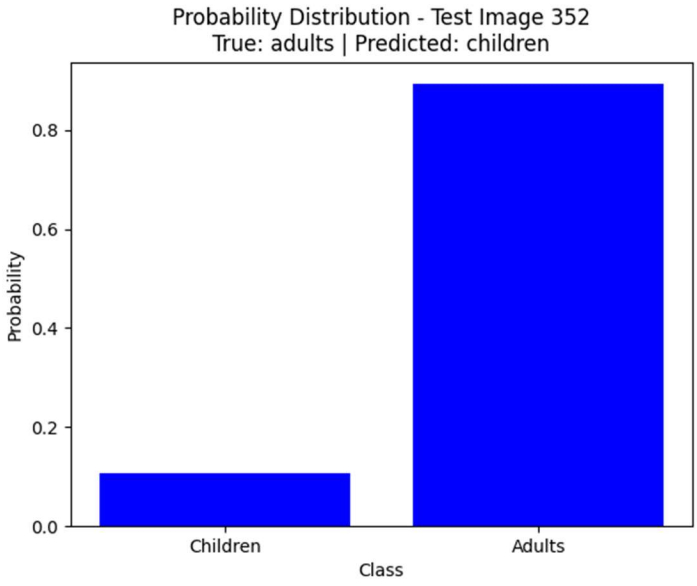
```
def save_probability_distributions(algorithm, y_true, y_pred, results_folder, probabilities):

    if not os.path.exists(results_folder):
        os.makedirs(results_folder)

    for idx, (true_label, pred_label, prob_dist) in enumerate(zip(y_true, y_pred, probabilities)):
        plt.bar(['Children', 'Adults'], prob_dist, color='blue')
        plt.title(f'Probability Distribution - Test Image {idx}\nTrue: {true_label} | Predicted: {pred_label}')
        plt.xlabel('Class')
        plt.ylabel('Probability')
        plt.savefig(os.path.join(results_folder, f'{algorithm}_probability_distribution_{idx}.png'))
        plt.close()

    print(f'Probability distributions saved for {algorithm} in {results_folder}')
```

```
results_folder = 'C:\\Users\\Gabriela\\Desktop\\Anu 3 - sem 1\\TIA\\results'

save_results('naive_bayes', y_test, y_pred_test_nb, results_folder)

save_results('knn', y_test, y_pred_test_knn, results_folder)


probabilities_nb = nb_classifier.predict_proba(X_test)
save_probability_distributions('naive_bayes', y_test, y_pred_test_nb, results_folder, probabilities_nb)

probabilities_knn = knn_classifier.predict_proba(X_test)
save_probability_distributions('knn', y_test, y_pred_test_knn, results_folder, probabilities_knn)
```

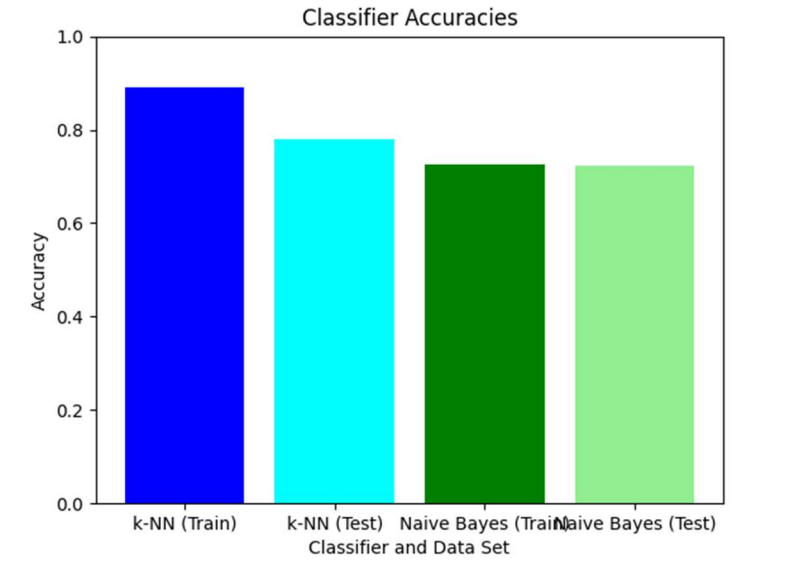
Mazilu Gabriela

Avantaje si implicatii ale Metodologiei Propuse





Analiza distribuțiilor de probabilitate pentru o imagine test (Imaginea 352) arată cum estimeaza fiecare algoritm probabilitățile claselor. În cazul k-NN, vedem o probabilitate mai mare pentru clasa corectă („adults), în timp ce pentru Naive Bayes, clasificatorul prezice fals, atribuind o probabilitate mai mare clasei „children”.

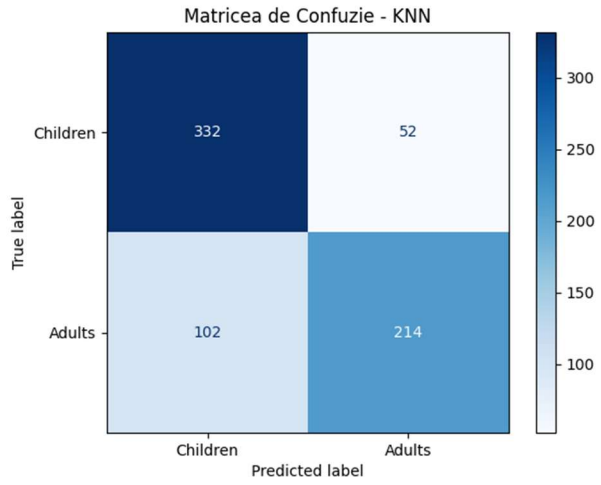


Mazilu Gabriela

```
# Calculul acuratetii pentru fiecare clasificator
accuracy_train_knn = accuracy_score(y_train, y_pred_train_knn)
accuracy_test_knn = accuracy_score(y_test, y_pred_test_knn)

accuracy_train_nb = accuracy_score(y_train, y_pred_train_nb)
accuracy_test_nb = accuracy_score(y_test, y_pred_test_nb)
```

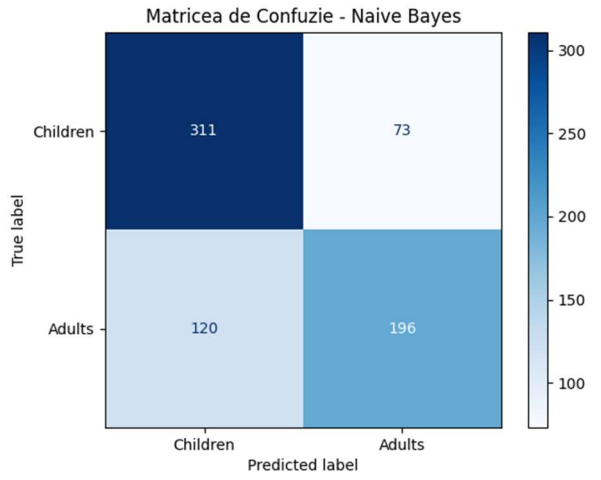
Se observa ca:

- k-NN are o acuratete mai mare pe setul de test decat Naive Bayes, indicand ca acesta poate sa generalizeze mai bine pe datele noi
- Acuratețea pe setul de antrenament pentru k-NN este mai mică decât pe setul de test, sugerând că nu există overfitting
- Naive Bayes are o acuratețe relativ constantă între setul de antrenament și cel de test, ceea ce indica un model bine echilibrat



Matricea de confuzie pentru Naive Bayes arata numarul de predictii corecte și incorecte:

- Majoritatea claselor "Children" sunt corect clasificate, dar exista și o rata semnificativa de fals negativ, unde "Children" sunt clasificați gresit ca "Adults".
- In ceea ce privește clasa "Adults", modelul pare sa aiba o performanță mai slaba, cu destul de multe clasificari gresite.



Matricea de confuzie pentru k-NN arata :

- O performanta mai buna pentru clasa “Children”, cu mai putine clasificari gresite comparative cu Naïve Bayes.
- De asemenea, pentru clasa “Adults”, k-NN pare sa aiba o performanta mai buna decat Naïve Bayes.

Mazilu Gabriela

Biblioteci Python utilizate:

```python
import cv2
from skimage.feature import hog
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, ConfusionMatrixDisplay
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

Concluzii personale:

In urma evaluarii facute, s-a dovedit performanta atat a algoritmului k-NN, cat si a algoritmului Naive-Bayes. Insa, pe problematica aleasa, algoritmul care este capabil sa recunoasca si sa clasifice cu mai mare precizie imaginile cu fete de copii si adulti este k-NN, avand in vedere acuratetea pe seturile de date, dar si matricea de confuzie.

Nu pot fi de neglijat nici detectorul Haar Cascade si metoda HOG, intrucat integrarea acestora in cadrul preprocesarii datelor a facut ca acuratetea pe setul de antrenare si cel de testare sa creasca semnificativ. Astfel, am observat ca preprocesarea datelor este un pas fundamental in crearea unui model performant.

De asemenea, am observat o imbunatatire a performantei modelului si in urma alegerii unui set de date care decupeaza fetele persoanelor.