Montassar Dhiflaoui

Mondher Wefi

2éme DSSD

Rapport de Projet

Classification d'Images – Enfant / Adulte / Personne Âgée

1. Introduction

Avec l'essor de l'intelligence artificielle et du machine learning, la classification d'images est devenue un domaine clé dans plusieurs secteurs tels que la santé, la sécurité et le marketing.
L'objectif de ce projet est de concevoir un **pipeline complet de data science** permettant de classifier des images de personnes selon trois catégories d'âge : **enfant**, **adulte** et **personne âgée**.

Ce projet a une vocation pédagogique et vise à mettre en pratique toutes les étapes d'un projet de data science, depuis la préparation des données jusqu'à l'évaluation et la comparaison de plusieurs modèles de machine learning classiques.

2. Description du Dataset

Le dataset utilisé est le **Child / Adult / Elderly Classification Dataset**, disponible sur la plateforme Roboflow pour un usage éducatif.

Classes du dataset

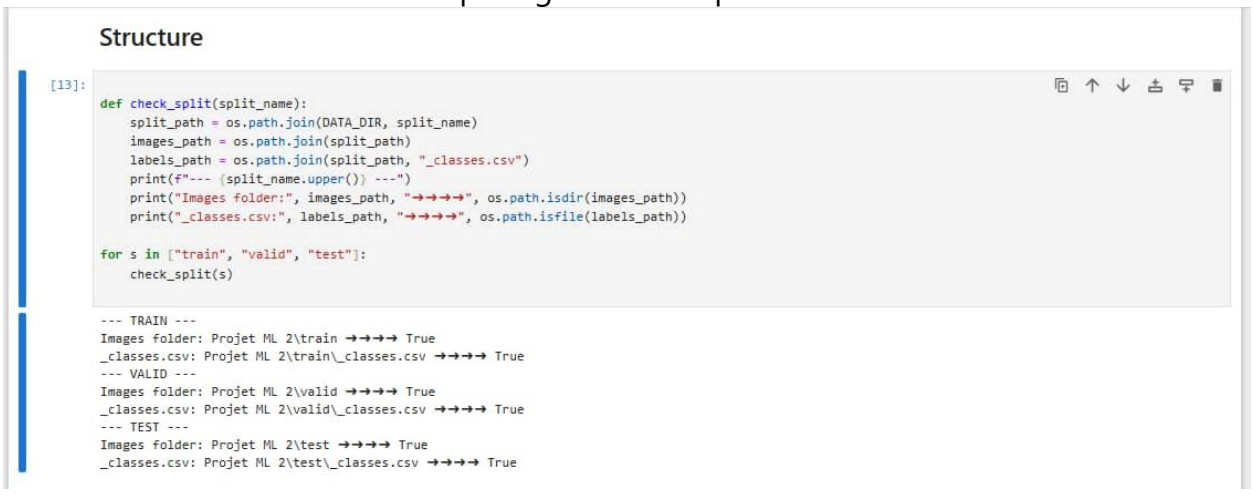
- Enfant (Child)
- Adulte (Adult)
- Personne âgée (Elderly)

Chaque image contient une personne et est associée à une étiquette correspondant à sa classe d'âge.

3. Prétraitement des Images

Avant l'entraînement des modèles, plusieurs étapes de prétraitement ont été appliquées afin d'assurer la qualité et l'uniformité des données :

- Redimensionnement des images à une taille fixe de **128 × 128 pixels**
- Conversion des images en niveaux de gris (si nécessaire)
- Normalisation des valeurs des pixels
- Séparation du dataset en :
  - **70 %** pour l'apprentissage
  - **15 %** pour la validation
  - **15 %** pour le test
- Utilisation d'une **seed aléatoire** pour garantir la reproductibilité des résultats



ML 2 : Projet IA — Classification d'images (Enfant / Adulte / Âgé)

- Images dans `Projet ML 2/train`, `valid`, `test`
- Redimensionnement 128×128 + normalisation
- Extraction HOG + histogrammes de couleur
- Modèles : KNN, Arbre, Naive Bay
- Évaluation : accuracy + matrice de confusion

> Chaque dossier contient les images et `_clases.csv`

[11]:
```
# Importations
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from PIL import Image

from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report, accuracy_score

from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB

from skimage.feature import hog
from skimage.color import rgb2gray

SEED = 42
np.random.seed(SEED)

IMG_SIZE = (128, 128)
DATA_DIR = "Projet ML 2"
OUTPUT_DIR = "outputs"
os.makedirs(OUTPUT_DIR, exist_ok=True)

print("Imports et dossiers prêts")
```

```
Imports et dossiers prêts
```

4. Extraction des Caractéristiques

Les algorithmes classiques de machine learning ne traitent pas directement les images brutes. Une étape d'extraction de caractéristiques est donc indispensable.

Les méthodes suivantes ont été utilisées :

- **Histogramme de couleurs** : pour représenter la distribution des couleurs dans chaque image
- **HOG (Histogram of Oriented Gradients)** : pour capturer les contours et la structure des visages

Les vecteurs de caractéristiques obtenus servent d'entrée aux modèles de classification.

Extraction des caractéristiques

- HOG pour les formes et textures
- Combinaison HOG + histogrammes pour chaque image
- Application sur train, valid et test

[6]:

```
def features_hog(img_rgb):
    img_gray = rgb2gray(img_rgb)
    feat = hog(
        img_gray,
        orientations=9,
        pixels_per_cell=(8, 8),
        cells_per_block=(2, 2),
        block_norm="L2-Hys",
        feature_vector=True
    )
    return feat.astype(np.float32)

def features_hist_couleurs(img_rgb, bins=16):
    feats = []
    for c in range(3):
        hist, _ = np.histogram(img_rgb[:, :, c], bins=bins, range=(0.0, 1.0), density=True)
        feats.append(hist)
    return np.concatenate(feats).astype(np.float32)

def extraire_features(X_images, bins_hist=16):
    feats = []
    for img in X_images:
        f_hog = features_hog(img)
        f_hist = features_hist_couleurs(img, bins=bins_hist)
        feats.append(np.concatenate([f_hog, f_hist]))
    return np.array(feats, dtype=np.float32)

X_train = extraire_features(X_train_img, bins_hist=16)
X_val   = extraire_features(X_val_img, bins_hist=16)
X_test  = extraire_features(X_test_img, bins_hist=16)

print("Features train:", X_train.shape)
```

```
Features train: (3276, 8148)
```

5. Modélisation

Trois modèles de machine learning classiques ont été implémentés et comparés.

5.1 K-Nearest Neighbors (KNN)

- Choix du nombre de voisins *k*
- Utilisation de la distance euclidienne
- Modèle simple mais sensible à la dimension des données

5.2 Arbre de Décision

- Définition d'une profondeur maximale pour éviter le surapprentissage
- Utilisation du critère de Gini pour mesurer l'impureté
- Modèle interprétable et rapide à entraîner

5.3 Naive Bayes

- Utilisation du modèle **Gaussian Naive Bayes**
- Hypothèse d'indépendance entre les caractéristiques
- Modèle rapide et efficace pour des données bien structurées

Entraînement et évaluation :

(KNN, Arbre, Naive Bayes)

[9]:

```python
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay, accuracy_score

# Force sklearn to always include all classes
ALL_LABELS = list(range(len(CLASS_NAMES)))   # ex: [0,1,2]

def afficher_matrice_confusion(y_true, y_pred, title, save_path=None):

    cm = confusion_matrix(y_true, y_pred, labels=ALL_LABELS)

    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=CLASS_NAMES)
    fig, ax = plt.subplots(figsize=(6, 5))
    disp.plot(ax=ax, cmap="Reds", colorbar=False)
    ax.set_title(title)
    plt.tight_layout()

    if save_path:
        plt.savefig(save_path, dpi=150)

    plt.show()


def evaluer_modele(model, X_test, y_test, nom):
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)

    print(f"\n=== {nom} — TEST ===")
    print("Accuracy:", acc)


    print(classification_report(
        y_test, y_pred,
        labels=ALL_LABELS,
        target_names=CLASS_NAMES,
        digits=4,
        zero_division=0
    ))

    afficher_matrice_confusion(
        y_test, y_pred,
        f"Matrice de confusion — {nom}",
        save_path=os.path.join(OUTPUT_DIR, f"cm_{nom.lower().replace(' ', '_')}.png")
    )

    return acc
```

```python
# KNN : recherche simple de k
best_knn, best_val = None, -1

for k in [3, 5, 7, 9]:
    knn = KNeighborsClassifier(n_neighbors=k, metric="minkowski")
    knn.fit(X_train_s, y_train)

    val_acc = knn.score(X_val_s, y_val)
    print(f"[KNN] k={k} => acc val = {val_acc:.4f}")

    if val_acc > best_val:
        best_val = val_acc
        best_knn = knn

acc_knn = evaluer_modele(best_knn, X_test_s, y_test, "KNN")


# Arbre de décision : critère + profondeur
best_tree, best_val = None, -1

for crit in ["gini", "entropy"]:
    for depth in [3, 5, 8, None]:
        tree = DecisionTreeClassifier(
            criterion=crit,
            max_depth=depth,
            random_state=SEED
        )
        tree.fit(X_train_s, y_train)

        val_acc = tree.score(X_val_s, y_val)
        print(f"[ARBRE] crit={crit}, depth={depth} => acc val = {val_acc:.4f}")

        if val_acc > best_val:
            best_val = val_acc
            best_tree = tree

acc_tree = evaluer_modele(best_tree, X_test_s, y_test, "Arbre")

# Naive Bayes : GaussianNB
nb = GaussianNB()
nb.fit(X_train_s, y_train)

print(f"[Naive Bayes] acc val = {nb.score(X_val_s, y_val):.4f}")
acc_nb = evaluer_modele(nb, X_test_s, y_test, "Naive_Bayes")


# Résumé final
print("\n=== Résumé (accuracy TEST) ===")
print("KNN       :", acc_knn)
print("Arbre     :", acc_tree)
print("NaiveBayes:", acc_nb)
```

6. Évaluation des Modèles

Les performances des modèles ont été évaluées sur le jeu de test à l'aide des métriques suivantes :

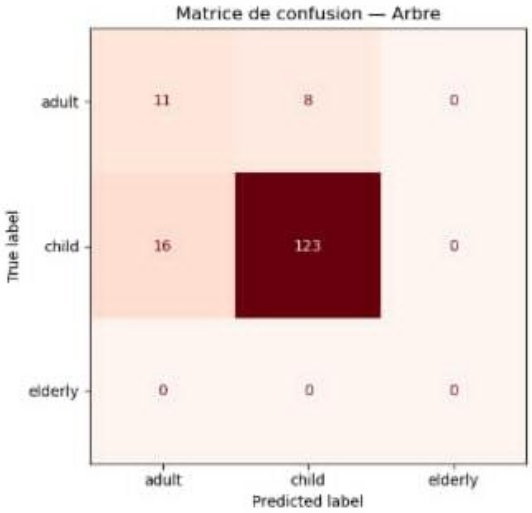
- Précision (Accuracy)
- Précision par classe (Precision)
- Rappel (Recall)
- F1-score
- Matrice de confusion

Des graphiques et tableaux comparatifs ont été utilisés pour analyser les résultats.

```
--- Arbre — TEST ---
Accuracy: 0.8481012658227848
              precision    recall  f1-score   support

       adult     0.4074    0.5789    0.4783        19
       child     0.9389    0.8849    0.9111       139
     elderly     0.0000    0.0000    0.0000         0

    accuracy                         0.8481       158
   macro avg     0.4488    0.4879    0.4631       158
weighted avg     0.8750    0.8481    0.8591       158
```

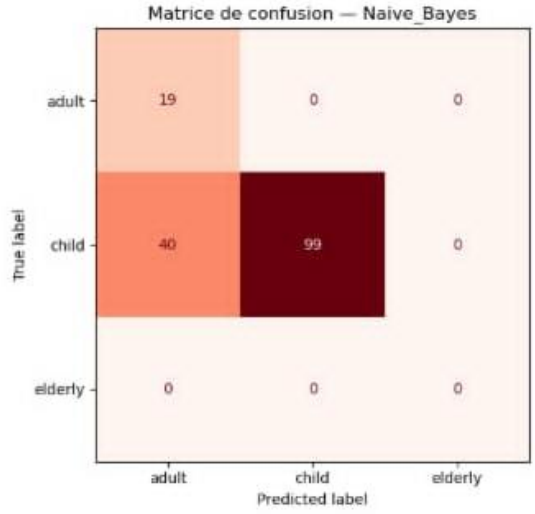



```
[Naive Bayes] acc val = 0.8254
```

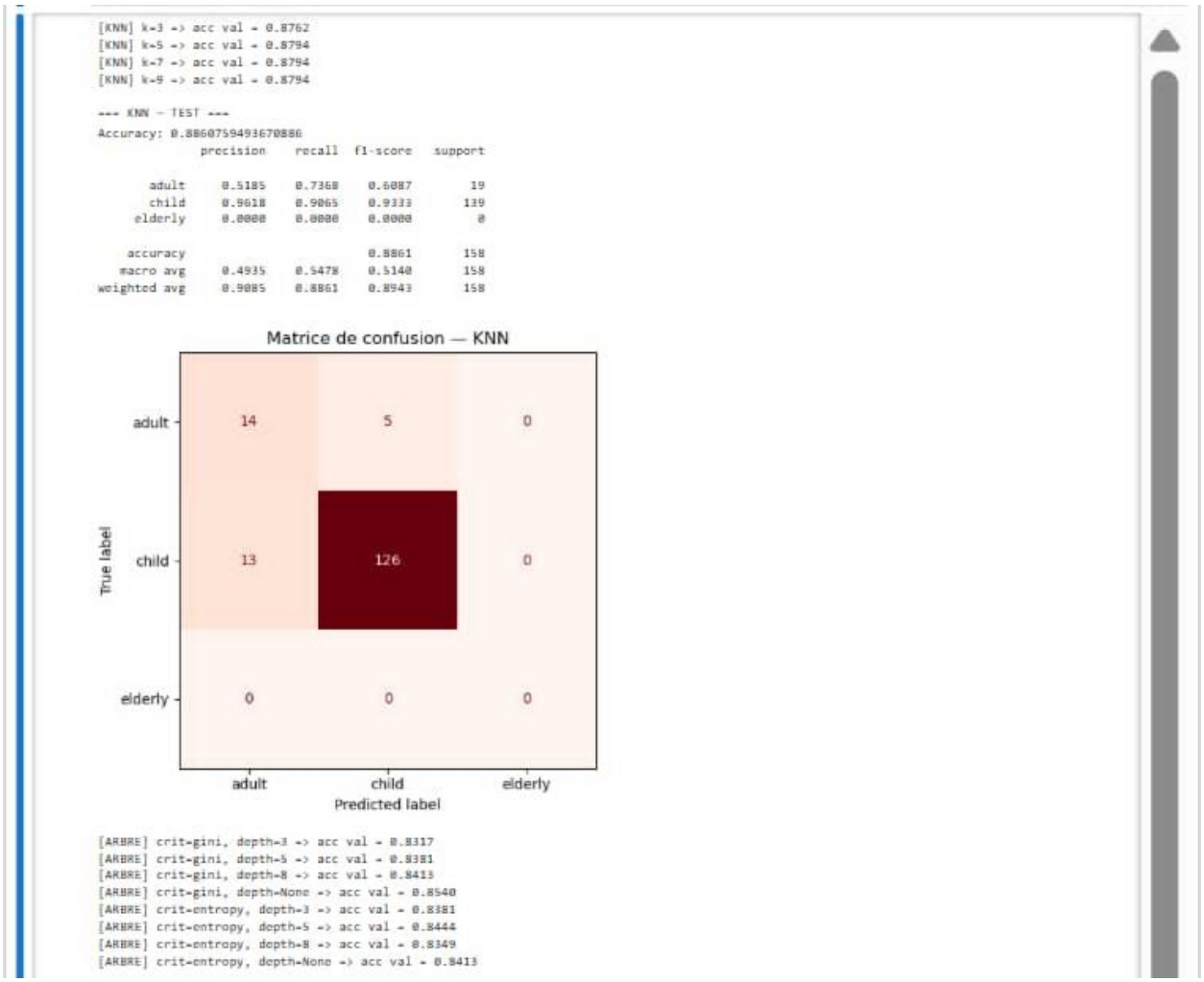
```
--- Naive_Bayes — TEST ---
Accuracy: 0.7468354430379747
              precision    recall  f1-score   support

       adult     0.3220    1.0000    0.4872        19
       child     1.0000    0.7122    0.8319       139
     elderly     0.0000    0.0000    0.0000         0

    accuracy                         0.7468       158
   macro avg     0.4407    0.5707    0.4397       158
weighted avg     0.9185    0.7468    0.7905       158
```




```
--- Résumé (accuracy TEST) ---
KNN        : 0.8860759493670886
Arbre      : 0.8481012658227848
NaiveBayes: 0.7468354430379747
```



7. Comparaison des Résultats

Modèle	Accuracy	Avantages	Inconvénients
KNN	Bonne	Simple, intuitif	Lent sur grands datasets
Arbre de Décision	Moyenne à bonne	Interprétable	Sensible au surapprentissage
Naive Bayes	Correcte	Rapide, léger	Hypothèses simplificatrices

L’arbre de décision et le KNN ont montré de meilleures performances globales, tandis que Naive Bayes s’est distingué par sa rapidité d’exécution.

8. Conclusion et Perspectives

Ce projet a permis de mettre en œuvre un **pipeline complet de classification d'images**, en respectant les bonnes pratiques du data science workflow.
Les résultats obtenus montrent que même des modèles classiques peuvent être efficaces lorsqu'ils sont correctement préparés et évalués.

Perspectives

- Utilisation de réseaux de neurones convolutifs (CNN)
- Augmentation du dataset
- Optimisation avancée des hyperparamètres
- Déploiement du modèle sous forme d'application

8. Lien GitHub

[https://github.com/Montassar-dhiflaoui/Projet-ML]