**BreastScreen-AI :**

**Un outil de détection précoce du cancer du sein à Djibouti par l’Intelligence Artificielle**

**Préparation des données / Ingénierie des caractéristiques**

**Membres de l’équipe : Yacin MOUHOUMED ELMI**

1. **Aperçu**

La phase de préparation des données constitue une étape essentielle dans le succès d’un projet d’apprentissage automatique. Dans le cas de **BreastScreen-AI**, elle permet de transformer des images échographiques brutes en données exploitables par le modèle de deep learning. Cette phase comprend la **collecte, le nettoyage, l’analyse exploratoire (AED), l’ingénierie des caractéristiques et la transformation**. Elle est cruciale pour assurer la robustesse, la précision et la généralisation du modèle CNN utilisé pour la détection précoce du cancer du sein.

1. **Collecte de données**

* **Source principale** : le jeu de données public **BUSI (Breast Ultrasound Images Dataset, 2020)** comprenant **780 images échographiques** classées en trois catégories :
  + Normal : 133 images
  + Bénin : 437 images
  + Malin : 210 images
* **Format** : images en niveaux de gris, dimensions variables.
* **Prétraitements initiaux** :
  + Conversion en format RGB (3 canaux) pour compatibilité avec les architectures CNN.
  + Normalisation des noms de fichiers et création de labels numériques (0 = Normal, 1 = Bénin, 2 = Malin).

1. **Nettoyage des données**

* **Suppression des images de mauvaise qualité** : élimination manuelle d’images floues ou bruitées.
* **Gestion des déséquilibres** : le dataset présente un déséquilibre de classes (beaucoup plus de cas bénins que normaux ou malins). Pour réduire ce biais, nous avons appliqué une **augmentation de données (data augmentation)**.
* **Labels manquants** : aucun cas de valeurs manquantes observé.

1. **Analyse exploratoire des données (AED)**

Une analyse exploratoire a permis de comprendre la distribution des classes et d’examiner la variabilité visuelle des images échographiques :

* **Distribution des classes** :
* Normal : 133 images
* Bénin : 437 images
* Malin : 210 images

→ Les classes sont déséquilibrées, ce qui justifie l’usage de la data augmentation.

* **Exemple de visualisation** : Affichage de quelques images représentatives de chaque classe pour vérifier la qualité et observer les différences visuelles.

import matplotlib.pyplot as plt

import random

classes = ['Normal', 'Bénin', 'Malin']

plt.figure(figsize=(10,5))

for i, c in enumerate(classes):

idx = random.choice([j for j, y in enumerate(y\_train) if y == i])

plt.subplot(1,3,i+1)

plt.imshow(x\_train[idx])

plt.title(c)

plt.axis("off")

plt.show()

Les images montrent que les tumeurs bénignes et malignes peuvent avoir des textures très similaires, ce qui justifie l’utilisation d’un modèle profond (CNN) pour capturer ces différences.

1. **Ingénierie des caractéristiques**

Dans ce projet, les **caractéristiques discriminantes** ne sont pas extraites manuellement mais automatiquement par les couches convolutionnelles. Néanmoins, certaines transformations ont été appliquées en amont :

* **Redimensionnement** : toutes les images en **256×256×3** pixels.
* **Data Augmentation** : rotation, translation, zoom et symétrie horizontale, permettant au modèle de mieux généraliser et de simuler une variabilité clinique.

**6. Transformation des données**

* **Mise à l’échelle** : normalisation des pixels entre **[0,1]** via rescale=1./255.
* **Encodage des labels** : transformation en labels numériques (0, 1, 2).
* **Séparation des ensembles** :
  + Train : 70%
  + Validation : 15%
  + Test : 15%

**Exploration du modèle**

1. **Sélection du modèle**

Nous avons choisi un **réseau neuronal convolutionnel (CNN)** car il est particulièrement adapté à l’analyse d’images médicales. Les CNN apprennent automatiquement les caractéristiques discriminantes (bords, textures, formes tumorales) sans nécessiter d’ingénierie manuelle complexe.

* **Forces** : bonne performance sur images médicales, capacité de généralisation.
* **Faiblesses** : besoin de beaucoup de données (compensé par la data augmentation).

1. **Entraînement du modèle**

* **Hyperparamètres utilisés** :
  + Optimiseur : Adam
  + Fonction de perte : sparse\_categorical\_crossentropy
  + Batch size : 32
  + Epochs : 50
* **Validation croisée** : validation sur un ensemble indépendant (15% du dataset).

1. **Évaluation du modèle**

Les performances du modèle ont été évaluées via :

* **Accuracy** : mesure la proportion de bonnes prédictions.
* **Matrice de confusion** : permet de visualiser les erreurs par classe.
* **Courbe ROC & AUC** : évaluation fine de la capacité discriminante du modèle.

Exemple de matrice de confusion (à générer dans le notebook) :

*from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay*

*y\_pred = model.predict(x\_test\_resized).argmax(axis=1)*

*cm = confusion\_matrix(y\_test, y\_pred)*

*disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=classes)*

*disp.plot(cmap=plt.cm.Blues)*

*plt.show()*

**4. Implémentation du code (extraits clés)**

*# Générateurs avec data augmentation*

*from tensorflow.keras.preprocessing.image import ImageDataGenerator*

*train\_datagen = ImageDataGenerator(*

*rescale=1./255,*

*rotation\_range=15,*

*width\_shift\_range=0.1,*

*height\_shift\_range=0.1,*

*zoom\_range=0.1,*

*horizontal\_flip=True,*

*fill\_mode='nearest'*

*)*

*val\_datagen = ImageDataGenerator(rescale=1./255)*

*train\_generator = train\_datagen.flow(x\_train\_resized, y\_train, batch\_size=32)*

*val\_generator = val\_datagen.flow(x\_val\_resized, y\_val, batch\_size=32)*

*# Entraînement*

*hist = model.fit(train\_generator,*

*epochs=50,*

*validation\_data=val\_generator)*