T-DEV-810
Image recognition

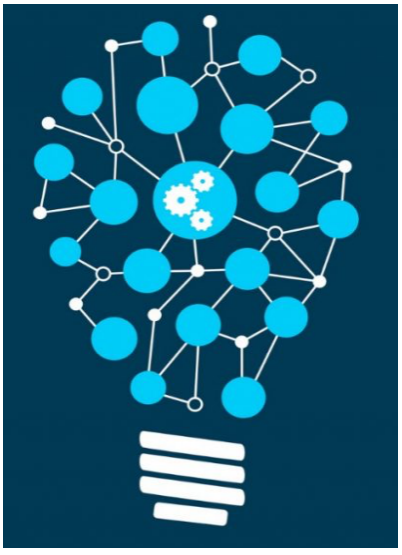
Brice TOSSIM
Hugo LEVEBVRE
Stephane OLIVIEIRA DOS SANTOS
Sébastien KELLER
Maximilien DESNOS

1. INTRODUCTION

1.1 Contexte

Ce ne sont pas des médicaments mais leur rôle en médecine est crucial. Appareils auditifs, prothèses, pacemakers, tests de grossesse, pompes à perfusion :tous font partie des dispositifs médicaux, des produits de santé dont l'action fonctionne par un moyen mécanique. Comme dans de nombreux autres domaines, là aussi, l'intelligence artificielle a permis de nouvelles avancées, si bien que de nombreux nouveaux dispositifs médicaux fonctionnent grâce à un algorithme. Du logiciel de radiologie qui arrive à détecter les tumeurs sur une imagerie médicale aux logiciels compagnons sur application mobile, qui permettent aux patients de suivre leur maladie au jour le jour, les utilisations de l'intelligence artificielle sont variées.

1.2 Le machine learning



Le Machine Learning peut être défini comme étant une technologie d'intelligence artificielle permettant aux machines d'apprendre sans avoir été au préalablement programmées spécifiquement à cet effet. Le Machine Learning est explicitement lié au Big Data, étant donné que pour apprendre et se développer, les ordinateurs ont besoin de flux de données à analyser, sur lesquelles s'entraîner.

De ce fait, le Machine Learning, issu par essence du Big Data, a précisément besoin de ce dernier pour fonctionner. Le Machine Learning et le Big Data sont donc interdépendants.

1.3 Notre projet

Ici, des médecins nous ont demandé d'utiliser le machine learning afin de les aider à détecter une pneumonie sur une radio. Pour ce faire les médecins nous ont donnée accès à 3 jeux de données correspondant à des radiologies de poumons sain et atteint de pneumonie.

1.4 Nos moyens

Ce projet de machine learning à été réalisé en Python 3 avec notamment les librairies Numpy, Skimage ou encore Sklearn.

2 Traitement des données

Dans un premier temps nous avons besoin d'importer les différentes librairies nécessaires aux traitements des données :

- Numpy: destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.
- Matplotlib : Sert à la création de graphique
- os : Ce module fournit une manière portable d'utiliser les fonctionnalités dépendantes du système d'exploitation.
- PIL : Python Imaging Library est une bibliothèque de traitement d'images pour le langage de programmation Python. Elle permet d'ouvrir, de manipuler, et de sauver différents formats de fichiers graphiques.
- Sklearn : Scikit-learn est une bibliothèque libre Python destinée à l'apprentissage automatique.

```
import numpy as np
import os
from numpy import asarray
from PIL import Image
from sklearn.ensemble import RandomForestClassifier

from sklearn import metrics
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

Ensuite nous allons définir les différents dossiers qui seront utilisés pour les jeux de données

```
mainDIR = './chest_xray/'

train_folder= mainDIR + '/train/'
val_folder = mainDIR + '/val/'
test_folder = mainDIR + '/test/'
```

Nous allons maintenant charger, stocker et transformer les images stocker dans les dossiers vus précédemment. Pour se faire on va utiliser la fonction suivante, **loadDataset**, en y passant la variable correspondant au chemin du dossier contenant les images.

```
train = loadDataset(train_folder)
test = loadDataset(test_folder)
val = loadDataset(val_folder)
```

Cette fonction génère un tableau contenant toutes les nuances de gris des images, pixel par pixel.

Le tableau x_array contiendra tous les tableaux de pixels pour chaque image et y_array est le tableau qui contiendra tous les labels qui serviront à déterminer si les tableaux seront issus d'une pneumonie ou non.

Dans un premier temps nous allons charger le contenu du dossier dans la variable listDir, elle contiendra deux autres dossiers PNEUMONIA et NORMAL.

Nous allons boucler sur le contenu de ces dossiers en omettant les fichiers cachés et traiter les images.

Dans la variable img nous allons charger le contenu de l'image avec la fonction **open** de la classe **Image**. Nous nous retrouvons avec un tableau contenant lui même un tableau correspondant aux informations RGB du du pixel. La variable img contient donc tous les pixels de l'image charger.

Mais le format obtenu n'est pas celui que l'on souhaite.

Tout d'abord nous ne pouvons pas comparer chaque pixel de chaque image pour savoir si le patient à une pneumonie.

Il va nous falloir réduire le nombre de pixel, nous allons ici réduire l'image à 50 sur 50 et stocker dans la variable img.

Le format de la variable img ne correspondant pas à nos besoins. Nous allons utiliser la fonction **asarray** pour avoir le tableau souhaité.

```
def loadDataset(folder):
    x_array = []
    y_array = []
    listDir = os.listdir(folder)
    for dir in listDir:
        if dir[0] != ".":
            for file in os.listdir(folder + dir):
                if(file[0] != "."):
                    img = Image.open(folder + dir + '/' + file).convert('L')

                    resize_img = img.resize((50, 50))
                    img = asarray(resize_img)
                    img_array = getImageValue(img)

                    x_array.append(img_array)
                    y_array.append(dir)

    return {'samples': x_array, 'labels': y_array}
```

Nous allons utiliser la fonction **getImageValue** qui prend en paramètre l'image précédemment traité. Cette fonction prend le tableau de valeur de l'image et le reformate pour qu'il puisse être lu par le modèle de l'arbre.

```
def getImageValue(img):
    return_tab = []
    for line_pixel in img:
        for pixel in line_pixel:
            return_tab.append(pixel)
    return return_tab
```

Après avoir remplie les tableau x_array avec les tableaux de pixels et y_array avec les labels, nous allons générer un classificateur de forêt aléatoire. Une forêt aléatoire est un méta-estimateur qui ajuste un certain nombre de classificateurs d'arbre de décision sur divers sous-échantillon de l'ensemble de données et utilise la moyenne pour améliorer la précision prédictive et contrôler le sur-ajustement.

```
clf = RandomForestClassifier()
clf.fit(train["samples"], train["labels"])
```

```
predicted = clf.predict(test["samples"])
print('Result:', accuracy_score(test["labels"], predicted))
```

```
Result: 0.7932692307692307
```

La dernière étape est de passer (L2 dans l'image ci-dessus) nos tableaux samples qui correspond aux images et labels à partir de train à l'arbre qui se chargera de faire le travail à partir de ces données.

Pour finir nous utilisons l'apprentissage fait à la machine pour prédire l'existence d'une pneumonie dans les images contenue dans le tableau test.

Nous pouvons enfin d'afficher le taux de réussite de reconnaissance qu'à notre machine à reconnaître des poumons sein et des poumons avec une pneumonie.