```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Projet fin détude Simplon

Codé en mars 2022

@auteur: Jean-Pierre Maffre
"""

#-------------- Import, variables globales et fonctions  ----------------------
#
#------------------------------------------------------------------------------

import streamlit as st
from streamlit_drawable_canvas import st_canvas

import os

import time

from PIL import Image, ImageOps

import tensorflow as tf

from joblib import load

import numpy as np

def charge_modeles(repertoire= None):
    """
    Retourne un dictionnaire dont la clé est le nom de l'algorithme et la valeur
        associée à la clé le nom du modèle binaire à charger.
    Entrée:
        nom du répertoire contenant les fichiers modeles
    Sortie:
        le dictionnaire{algo:nom_du_fichier}
    """
    assert type(repertoire)== str, "Le paramètre repertoire doit être de "+\
        "type chaine de caractères"

    liste_fichiers= os.listdir(repertoire)
    # On ne récupère que les fichiers d'extensions .modele
    liste_fichiers = [f for f in liste_fichiers if ".modele" in f]
    # Recherche du nom de l'algorithme contenue dans le nom fichier
    # Les fichiers doivent être écrit avec la syntaxe: algo_caractéristique.modele
    algo= [a.split("_")[0] for i,a in enumerate(liste_fichiers)]
    return {cle: valeur for cle, valeur in zip(algo, liste_fichiers)}

def sauvegarde(nom):
    imgpil= Image.open("./images/tmp.png")
    imgpil.save(nom)

ERR_CHARGEMENT_DICO= False
ERR_DICO_VIDE= False
ERR_TYPE_ALGO= False
ERR_FICHIER_TEST_ABS= False
TEST_LANCER= False

#------------------------- Programme principal -------------------------------
#
#------------------------------------------------------------------------------

st.set_page_config(page_title= "Lecture caractères manuscrits",
                   page_icon= "./images/icone.png")

dict_algo_fichier = charge_modeles(repertoire= "./modeles/")
algos = list(dict_algo_fichier.keys())
fichiers= list(dict_algo_fichier.values())
```

```
#---------------- Construction de la page - zone latérale ---------------------
st.sidebar.markdown("**Options:**")
# Epaisseur du tracé
epaisseur = st.sidebar.slider(label= "Épaisseur du trait", min_value= 1,
                              max_value= 25, value= 15, key= "ept")
nom_algo = st.sidebar.selectbox(label="Choix du modèle d'algorithme",
                                options= algos, key="bteselect")
fichier_modele= dict_algo_fichier[nom_algo]

st.sidebar.markdown("**Sauvegarde:**")

with st.sidebar.container():
    st.markdown(
        "Pour effectuer une sauvegarde de l'image appuyez sur le bouton "
        "correspondant à la vérité terrain. L'image sera enregistrée sous "
        "la forme 'chiffre_numero_unique.png'."
    )
    c21, c22, c23, c24, c25= st.columns(5)
    with c21:
        chif0= st.button(("0"), key="b0")
        chif5= st.button(("5"), key="b5")
    with c22:
        chif1= st.button(("1"), key="b1")
        chif6= st.button(("6"), key="b6")
    with c23:
        chif2= st.button(("2"), key="b2")
        chif7= st.button(("7"), key="b7")
    with c24:
        chif3= st.button(("3"), key="b3")
        chif8= st.button(("8"), key="b8")
    with c25:
        chif4= st.button(("4"), key="b4")
        chif9= st.button(("9"), key="b9")

    if chif0:
        sauvegarde(f"0_{int(time.time())}.png")
    if chif1:
        sauvegarde(f"1_{int(time.time())}.png")
    if chif2:
        sauvegarde(f"2_{int(time.time())}.png")
    if chif3:
        sauvegarde(f"3_{int(time.time())}.png")
    if chif4:
        sauvegarde(f"4_{int(time.time())}.png")
    if chif5:
        sauvegarde(f"5_{int(time.time())}.png")
    if chif6:
        sauvegarde(f"6_{int(time.time())}.png")
    if chif7:
        sauvegarde(f"7_{int(time.time())}.png")
    if chif8:
        sauvegarde(f"8_{int(time.time())}.png")
    if chif9:
        sauvegarde(f"9_{int(time.time())}.png")

st.sidebar.markdown("**Zone de test:**")
st.sidebar.markdown(
    "Pour effectuer lancer un test de non régression... un clic suffit :) ! "
    "On chargera l'ensemble des modèles disponibles que l'on testera. **Le "
    "résultat des tests s'affiche en bas de page**."
)
test= st.sidebar.button("Test", key="btst")

#---------------- Construction de la page - zone principale -------------------

st.title("🍀 Lecture caractères manuscrits")
Image_illustration = Image.open("./images/image_titre.png")
```

```
st.image(Image_illustration)
st.markdown(
    "Aprés la théorie et l'étude sur les différents algorithmes "
    "sur des données provenant du même dataset, il est temps de "
    "passer à la pratique !\n \n"
    "Comme on peut le voir sur l'image illustrant cette page il y "
    "une trés grande diversité dans les écritures. Nos modèles ne "
    "devraient donc pas avoir de difficultés majeures pour s'adapter "
    "à un style d'écriture qu'il n'a jamais rencontré...\n\n"
)

st.subheader("De la théorie, à la pratique:")

with st.form("Prédiction"):
    c1, c2= st.columns(2)
    with c1:
        recon_canvas = st_canvas(
            #epaisseur = st.slider("Epaisseur du trait: ", 1, 25, 15)
            # Fixed fill color with some opacity
            fill_color="rgba(0, 165, 0, 0.3)", #"rgba(255, 165, 0, 0.3)"
            stroke_width= epaisseur,
            stroke_color="#FFFFFF",
            background_color="#000000",
            update_streamlit=True,
            height=280,
            width=280,
            drawing_mode= "freedraw",
            key="recon_canvas",
        )
    with c2:
        irma_dit = st.form_submit_button("Prédiction")

    if irma_dit:
        # on charge l'image et on l'a converti
        img= recon_canvas.image_data
        image= Image.fromarray(img)
        image= image.resize((28,28))
        # Convertion de l'image au format rgba en tableau numpy 2d
        image= ImageOps.grayscale(image)
        image.save("./images/tmp.png")

        chiffre= np.array(image)/255.0

        # On charge le modèle
        id_algo= nom_algo[0:3]
        if id_algo== "Reg" or id_algo== "KNN":
            modele= load(f'./modeles/{fichier_modele}')
            tab= list(modele.predict_proba([chiffre.ravel()])[0])
        elif id_algo== "CNN":
            modele = tf.keras.models.load_model(f'./modeles/{fichier_modele}')
            tab= list(modele.predict(chiffre.reshape(1,28,28)).reshape(10))

        prediction= tab.index(max(tab))
        # On affiche la prédiction et le tableau des probabilités
        ch= ""
        for i in range(len(tab)):ch+= f"p({i})= {100*tab[i]:.2f} %, "
        col1, col2= st.columns(2)
        with col1:
            st.image(image)
        with col2:
            st.write(f"Chiffre lu: {prediction}")

        st.write(f"Tableau des probabilités:\n{ch[:-2]}")
        st.write(f"fichier modèle: {fichier_modele}")

st.markdown(
    "Pour retrouver ce projet, ainsi que les projets réalisés lors de la "
    "formation 'Développeur Data IA' effectué à simplon: "
    " [jpphi - github](https://github.com/jpphi)")
```

```
#----------------------- tests de non régression ----------------------------
#
#----------------------------------------------------------------------------

if test:
    TEST_LANCER= True
    st.markdown(
        "**Zone de test**<br>Déroulement du test:<br/>"
        "1- Les modèles peuvent-ils être chargés ?<br/>"
        "2- Chaque modèle sera testé sur une image test."
        , unsafe_allow_html= True)

    try:
        dict_algo_fichier = charge_modeles(repertoire= "./modeles/")
        if dict_algo_fichier== {}: ERR_DICO_VIDE= True
    except:
        ERR_CHARGEMENT_DICO= True

    if ERR_CHARGEMENT_DICO:
        st.markdown(
           "**Erreur:**<br/><span style='color:red'> Le répertoire dans "
           "lequel les modèles sont sauvegardés n'existe pas !</span><br/>"
           , unsafe_allow_html= True)

    elif ERR_DICO_VIDE:
        st.markdown(
            "**Erreur:**<br/><span style='color:red'> Aucun algorithme n'a pu "
            "être chargé !</span><br/>", unsafe_allow_html= True)

    else: # on continu les tests
        algos = list(dict_algo_fichier.keys())
        fichiers= list(dict_algo_fichier.values())
        try:
            chiffre_test= Image.open("./images/image_test.png")
            chiffre_test= np.array(chiffre_test)/255.0

        except:
            st.markdown(
                "**Erreur:**<br/><span style='color:red'> Fichier test absent "
                "!</span><br/>", unsafe_allow_html= True)
            ERR_FICHIER_TEST_ABS= True

        if not(ERR_FICHIER_TEST_ABS):

            for i in range(len(algos)):
                # on charge le modèle
                id_algo= nom_algo[0:3]
                if id_algo== "Reg" or id_algo== "KNN":
                    modele= load(f'./modeles/{fichier_modele}')
                    tab= list(modele.predict_proba([chiffre_test.ravel()])[0])
                elif id_algo== "CNN":
                    modele = tf.keras.models.load_model(f'./modeles/{fichier_modele}')
                    tab= list(modele.predict(chiffre_test.reshape(1,28,28)).reshape(10))
                else:
                    st.markdown(
                        "**Erreur:**<br/><span style='color:red'> Algorithme de "
                        "type inconnu !</span><br/>", unsafe_allow_html= True)
                    ERR_TYPE_ALGO= True
                    break

                prediction= tab.index(max(tab))
                if prediction!= 5:
                    st.write(f"Modèle: {algos[i]}, fichier: {fichiers[i]}:")
                    st.markdown(
                        "**Erreur:**<br/><span style='color:blue'> Erreur de "
                        "prédiction ! Ceci ne remet par pour autant en cause "
                        "le programme, une part d'aléatoire existe dans les "
```

```
                        "prédictions </span><br/>", unsafe_allow_html= True)

if not(ERR_CHARGEMENT_DICO) and not(ERR_DICO_VIDE) and not(ERR_TYPE_ALGO) and \
    not(ERR_FICHIER_TEST_ABS) and TEST_LANCER:
    st.markdown("**Tests passés avec succès, pas d'erreur de fonctionnement**")
    TEST_LANCER= False
```