Prédiction
Guide de fonctionnement et d'utilisation

Introduction

L'application *Prediction* du projet a été développée dans l'optique de fournir une **interface graphique pour tester facilement le modèle**.
En utilisant le modèle déjà entrainé développé par l'équipe Muzik Patch Generator en 2022, cette application **prédit si un son est écoutable ou non**.
Pour cela elle se base sur la base de données que nous lui avons fournie.

Guide d'utilisation

Pour utiliser l'application, situez-vous dans le répertoire *prediction_streamlit* avec un terminal et entrez la commande suivante :

« streamlit run Prediction.py »

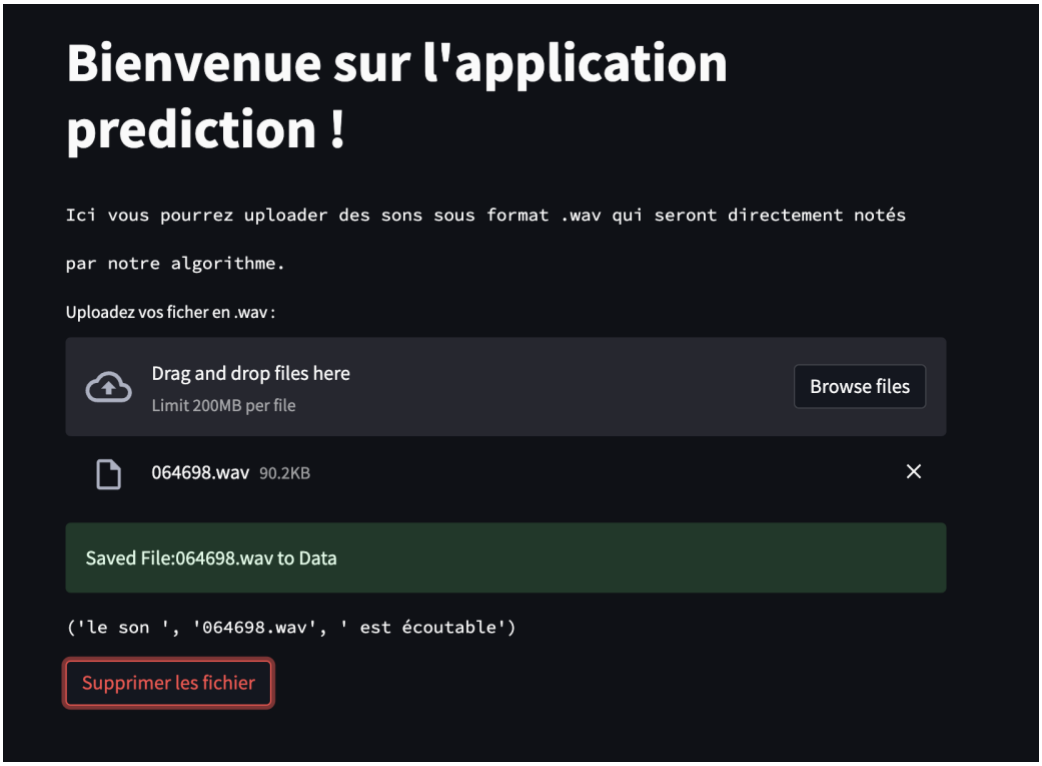
Si cette commande ne marche pas, vous n'avez probablement pas installé correctement Streamlit, dans ce cas vous pouvez facilement l'installer avec la commande

« pip install streamlit »

Assurez-vous d'avoir toutes les bibliothèques nécessaires installées sur votre ordinateur.

Interface de l'application

Une fois l'application lancée, vous devriez voir la fenêtre suivante apparaître :



Test d'écoutabilité d'un son

- Glisser le fichier audio .wav dans la zone **drag&drop** OU cliquer sur le bouton « **Browse files** »
- Sélectionner manuellement les fichiers que vous voulez tester.

⚠ Si vous avez ajouté trop de fichiers ⚠

Vous n'avez pas à les supprimer dans le répertoire (bien que cela marche),
Vous pouvez simplement cliquer sur le bouton

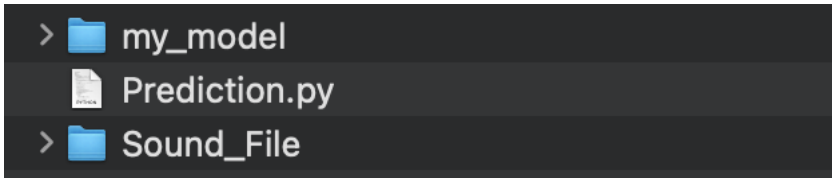
supprimer les fichiers

et tous les fichiers du répertoire *sound_Files* seront supprimés.

Guide de fonctionnement

Pour faciliter les explications, le guide de fonctionnement sera accompagné de capture d'écrans faites sur le logiciel Visual studio code

Composantes de l'application



Prédisposition pour le fonctionnement

Pour fonctionner, l'application a besoin de **deux dossiers** ainsi que du **fichier python** contenant l'application.

- Le dossier *my_model* :

Qui contient le modèle de Machine Learning **sauvegardé** à partir de la commande *model.save* de Keras, qui **créée directement le dossier correspondant.**

- Le dossier *Sound_File* :

Qui contient les **sons qui seront notés par l'application**
Vous n'avez pas besoin d'interagir avec ce dossier que Streamlit propose directement d'y mettre vos fichiers avec un système de **drag&drop**, cependant, si vous vouliez mettre vos fichiers dans ce dossier avant de lancer l'application, les fichiers seraient analysés au lancement de l'application.

Import des librairies

Pour que l'application puisse correctement marcher nous avons besoins de plusieurs bibliothèque python, vous trouverez ici une brève description de chacune d'entre elle :

- OS

Ce module fournit une façon portable d'utiliser les fonctionnalités dépendantes du système d'exploitation.

- Numpy

La bibliothèque NumPy (http://www.numpy.org/) permet d'effectuer des calculs numériques avec Python. Elle introduit une gestion facilitée des tableaux de nombres.

- Librosa

Bibliothèque python la plus utilisé dans le traitement sonore, ses fonctions permettent de facilement extraire des données sur les fichiers audio (Spectre audio, analyse de fourier, etc..)

- Tqdm

Bibliothèque utilisée pour créer des métriques de progression sur python, dans ce code, elle est uniquement utilisée pour itérer à travers une liste de fichier.

- Tensorflow

Outil open source d'apprentissage automatique développé par Google. Le code source a été ouvert le 9 novembre 2015 par Google et publié sous licence Apache.

- Streamlit

Dont le post de lancement a été diffusé à la mi-octobre 2019, a pour vocation la création de web-apps de Machine Learning.

Le code

```
def save_uploadedfile(uploadedfile):
    with open(os.path.join("./Sound_File/", uploadedfile.name), "wb") as f:
        f.write(uploadedfile.getbuffer())
    return st.success("Saved File:{} to Data".format(uploadedfile.name))
```

La fonction *save_uploadedfile* est ici pour s'assurer que le fichier sera **lisible par le modèle**, et qu'il est au **bon format**

```
def main():
    model = keras.models.load_model('my_model')
    feature_list = []
    Name_list=[]

    st.title('Bienvenue sur l\'application prediction !')
    st.text("Ici vous pourrez uploader des sons sous format .wav qui seront directement notés")
    st.text(" par notre algorithme.")

    Files=st.file_uploader("Uploadez vos ficher en .wav :",accept_multiple_files=True)

    if Files is not None:
        for file in Files:
            save_uploadedfile(file)

    else:
        st.text('You have not uploaded any files yet')
```

Ici nous chargeons le modèle à partir du dossier cité plus haut en utilisant la commande Keras associée.
Toutes les commande commençant par st.**** servent à instancier l'application et l'interface graphique correspondante.

St.file uploader va créer le système de **drag&drop** avec lequel vous pourrez poser vos fichiers

La suite est un processus de vérification afin de s'assurer de ne pas recevoir de message d'erreur si aucun fichier n'est sélectionné.

```
print('Preparing feature dataset and labels.')
for file in tqdm(os.listdir('./Sound_File')):
    # Skip if it's not a wav file
    if not file.endswith('.wav'):
        continue
    # Load audio and stretch it to length 1s

    audio_path = os.path.join('./Sound_File/', file)

    audio, sr = librosa.load(path=audio_path, sr=None)
    audio = librosa.effects.time_stretch(y=audio, rate=len(audio)/sr)
    # Calculate features and get the label from the filename
    mels = librosa.feature.melspectrogram(y=audio, sr=sr, n_fft=2048, hop_length=512)
    mels_db = librosa.power_to_db(S=mels, ref=1.0)
    if mels_db.shape == (128,87):
        feature_list.append(mels_db.reshape((128, 87, 1)))
        Name_list.append(file)
```

Ce bloc de code a pour but de faire du **preprocessing** pour les fichiers audios.
En réalité, le modèle n'effectue pas de prédiction sur le fichier audio en brut, il le converti d'abord en mel spectrogramme, qui est une interprétation graphique du fichier audio.

Ces différentes commandes servent à normaliser la longueur du fichier audio, le convertir en mel spectrogramme, puis convertir le spectrogramme en dB.

Le fichier traité est ensuite ajouté à la liste des fichiers qui seront traités par le modèle.

```
if(len(feature_list)!=0):
    features = np.array(feature_list)
    x=model.predict(features)
    for i in range(len(x)):

        if(x[i][0]>0.5):
            st.text(body=('le son ',Name_list[i],' est écoutable'))
            print(i," : ", "son écoutable")
        else:
            st.text(body=('le son ',Name_list[i],' n\'est pas écoutable'))
            print(i," : ", "son non écoutable")

else:
    st.write("Aucun fichier selectionné pour le moment")
```

Ce bloc de code va utiliser la fonction *predict* du modèle pour lui attribuer une note à chaque fichier dans la liste *feature_list*, la liste de note est donc contenue dans la variable x.

La note est comprise entre 0 et 1, on considère que si elle est supérieure à 0.5, le fichier est écoutable et vice-versa sinon.

```
if st.button('Supprimer les fichier'):
    for file in tqdm(os.listdir('./Sound_File')):
        if file == None:
            continue
        audio_path = os.path.join('./Sound_File/', file)
        os.remove(audio_path)
```

Ce bloc de code créer un bouton qui permet de supprimer tous les fichiers situés dans le répertoire *sound_files*.