MACHINE LEARNING

MISE EN PRODUCTION ET DÉPLOIEMENT CONTINU SUR DES DONNÉES MULTIMÉDIA

Sommaire

Introduction

- Analyser et comprendre les appels d'urgence au 911
- Développer un modèle capable de traiter les fichiers audio
- Kaggle : “911 Recordings: The First 6 Seconds”

Préparation des données

1) Utilisation de Librosa

2) Fréquence d’echatillonage de 50 (contre 11kHz)

```
# Répertoire contenant les fichiers audio
train_directory = "ref_data/"
taux_echantillonage = 50

# Liste pour stocker les noms de fichiers et les vecteurs de données
file_names = []
data_list = []

# Parcourir tous les fichiers dans le répertoire
for filename in os.listdir(train_directory):
    if filename.endswith(".wav"):
        # Chemin complet du fichier
        file_path = os.path.join(train_directory, filename)

        # Charger le fichier audio avec Librosa
        data, _ = librosa.load(file_path, sr=taux_echantillonage)

        # Stocker le nom du fichier et le vecteur de données dans les listes
        file_names.append(filename)
        data_list.append(data)

# Créer un DataFrame avec les noms de fichiers
audiodata = pd.DataFrame({'filename': file_names})
```

Mise en production du modèle

1) Chargement et prétraitement des données

2) Entraînement du modèle de prédiction

```
def comparaison_classifieurs(X,Y,X_norm,clfs):
    meilleur_model,meilleur_score=run_classfieurs_cv(X,Y,clfs)
    meilleur_model_norm,meilleur_score_norm=run_classfieurs_cv(X_norm,Y,clfs)
    if(meilleur_score>=meilleur_score_norm):
        strategie='no_norm'
        pickle.dump(strategie,open('strategie.pkl','wb'))

        return meilleur_model,strategie

    else:
        strategie='norm'
        pickle.dump(strategie,open('strategie.pkl','wb'))

        return meilleur_model_norm
```

```
X = X.values
scaler = StandardScaler()
X_norm = scaler.fit_transform(X)
```

```
best_model, strategie=comparaison_classifieurs(X,Y,X_norm,clfs)
```

```
 50%|████      | 1/2 [00:01<00:01,  1.59s/it]
balanced_accuracy Moyen for CART is: 0.499 +/- 0.057
100%|████████| 2/2 [00:03<00:00,  1.61s/it]
balanced_accuracy Moyen for ID3 is: 0.473 +/- 0.053
 50%|████      | 1/2 [00:01<00:01,  1.50s/it]
balanced_accuracy Moyen for CART is: 0.497 +/- 0.058
100%|████████| 2/2 [00:03<00:00,  1.57s/it]
balanced_accuracy Moyen for ID3 is: 0.473 +/- 0.053
```

```
best_model
```

DecisionTreeClassifier

```
DecisionTreeClassifier(random_state=1)
```

Mise en production du modèle

3)Sélection des features importantes

300 -> 221 colonnes

4) Optimisation des paramètres du modèle

Gridsearch

Mise en production du modèle

5) Automatisation du modèle final avec pickle

```
def automatisation(X,X_norm,Y,strategie,classifieur,selected_features):
    RF=RandomForestClassifier(n_estimators=1000,random_state=1)

    P = Pipeline([('FS',SelectFromModel(RF,max_features=len(selected_features))),
                  ('classifieur',classifieur)
                      ])

    if(strategie=='no_norm'):
        P.fit(X,Y)
    else:
        P.fit(X_norm,Y)

    pickle.dump(P,open('model_final_audio.pkl','wb'))
```

6) Scoring et mise à jour continus

balanced accuracy

```
# pour chaque fichier du dossier data_test, faire la préparation et la prédiction
for filename in os.listdir('data_test/'):
    if filename.endswith(".wav"):
        file_path = os.path.join('data_test/', filename)
        to_predict = prepa_audio(file_path, taux_echantillonage)
        prediction = pipeline.predict(to_predict)
        print(filename, prediction)
```

```
call_123_0.wav [1]
call_125_0.wav [1]
call_134_0.wav [1]
call_139_0.wav [1]
call_13_0.wav [1]
call_185_0.wav [1]
call_218_0.wav [1]
```

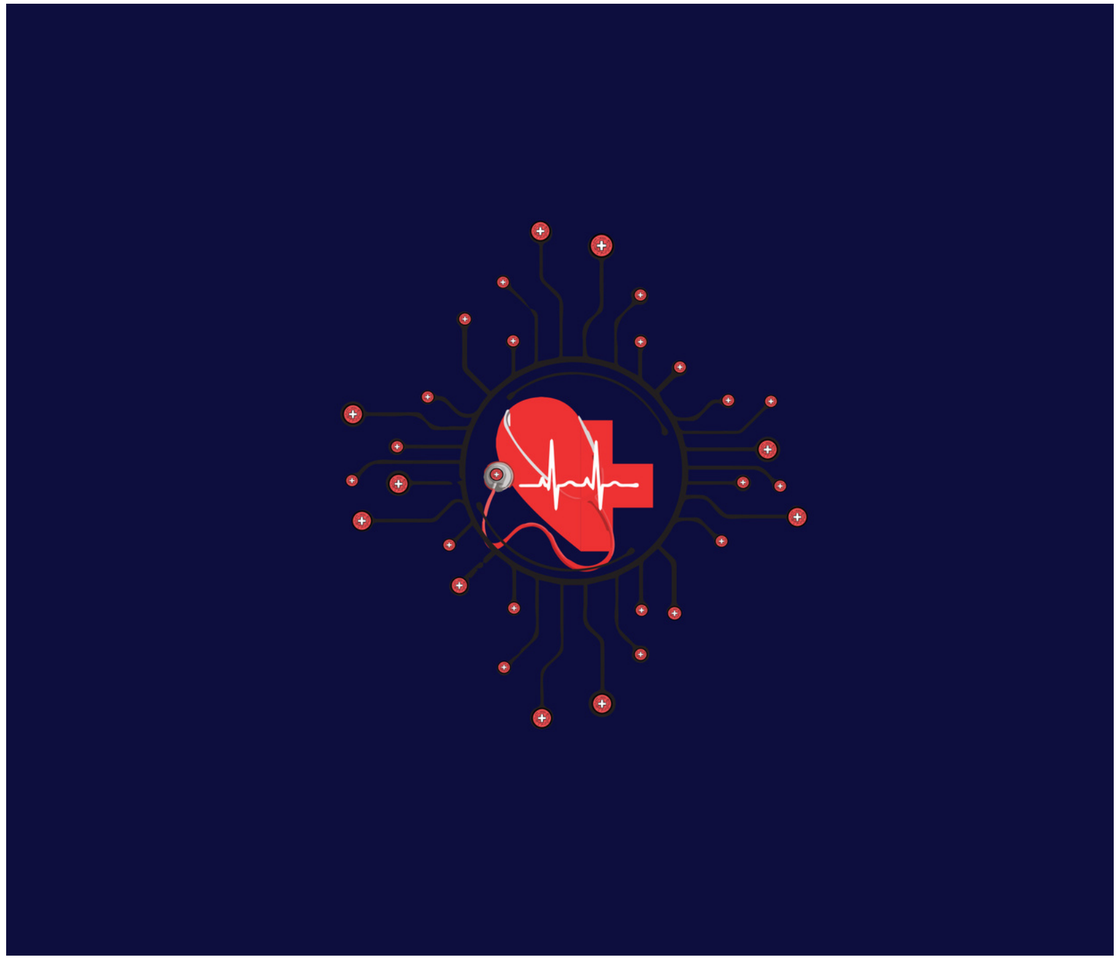
Conteneurisation Docker

- API de serving : FastAPI
- Dockerizing l'API avec Dockerfile
- Interface web : Streamlit
- Gestion des IP dynamiques avec Docker-Compose : Serving API - WebApp
- Démarrage :

  docker-compose -f webapp/docker-compose.yml --up

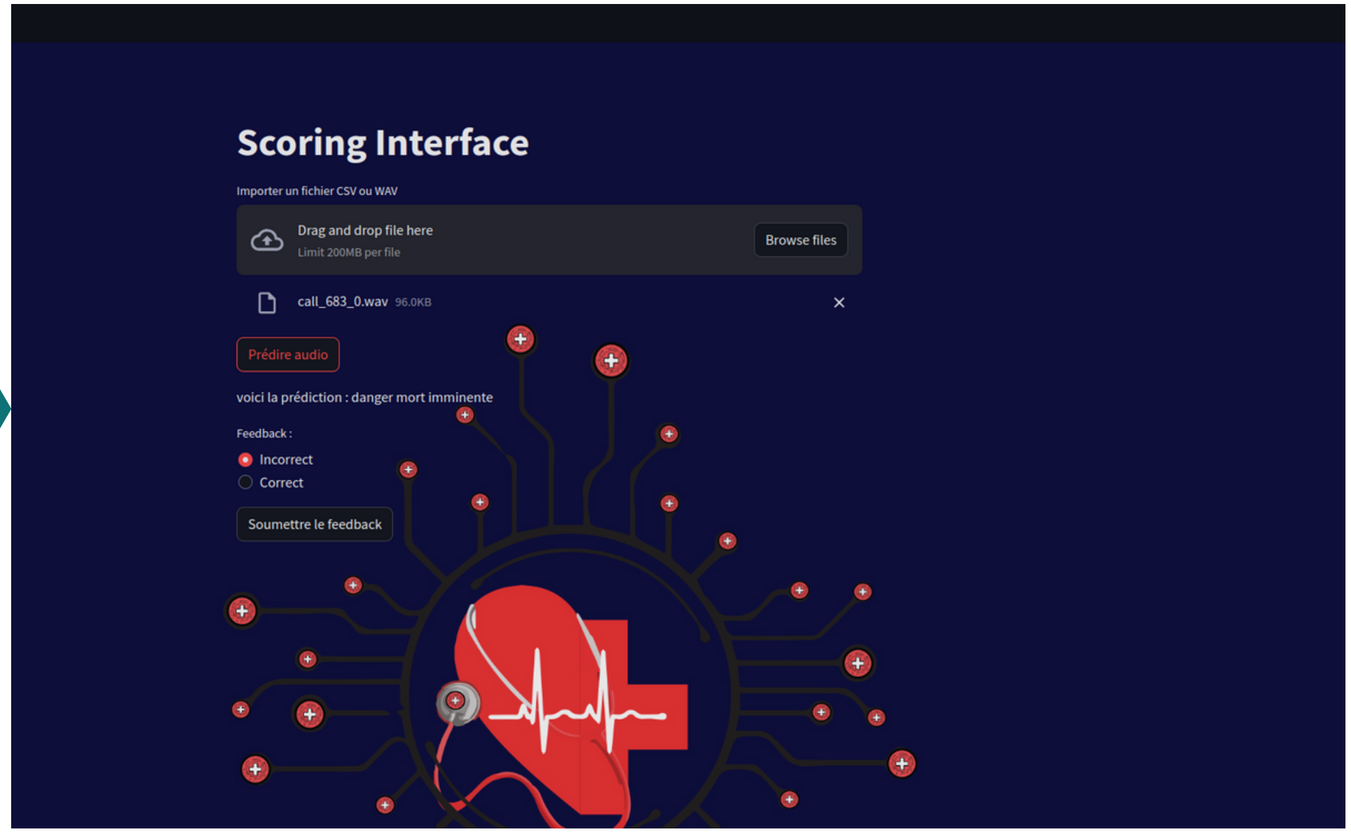
  docker-compose -f serving/docker-compose.yml --up

Webapp & Serving



Webapp

1) DEMO



Webapp

2) Contenu

- Vérification des extensions des fichiers
- Prédictions sur les CSV
- **Prédiction sur l’audio**
- Bouton feedback

Serving

Roots :

- Pour les csv

```python
# Prediction endpoint
@app.post("/predict")
async def predict(file: UploadFile = File(...)):
    try:
        df = pd.read_csv(file.file, sep=';')
        X_final_scoring,X_final_norm_scoring=preparation_scoring(df,scaler,encoder)

        try:
            if(strategie=='no_norm'):
                predictions=model.predict(X_final_scoring)
            else:
                predictions=model.predict(X_final_norm_scoring)
        except Exception as e:
            raise HTTPException(status_code=501, detail=str(e))

        # Format predictions as JSON
        predictions_json =  predictions.tolist()

        return predictions_json

    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Serving

Roots :

- Pour les wav

```python
@app.post("/predictaudio")
async def predict(file: UploadFile = File(...)):
    try:
        print("debut du read")
        with open("uploaded_file.wav", "wb") as f:
            f.write(await file.read())
        print("fin du read")

        with open("../artifacts/model_final_audio.pkl", 'rb') as f:
            model_final_audio = pickle.load(f)

        to_predict = prepa_audio('uploaded_file.wav', 50)
        prediction = model_final_audio.predict(to_predict)

        predictions_json =  prediction.tolist()

        list_data=[file.filename]+to_predict.values.tolist()[0]+[prediction[0],prediction[0]]

        with open("../data/prod-data.csv", 'a',newline= '') as f:
            writer_object = writer(f,delimiter=';')
            writer_object.writerow(list_data)

        return predictions_json

    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Serving

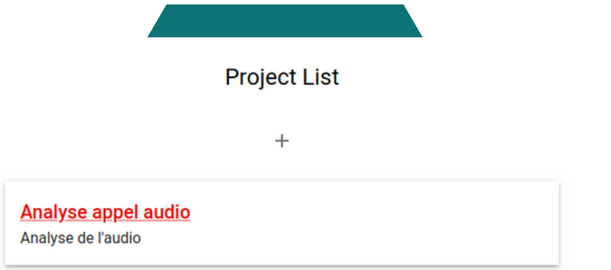
Roots :

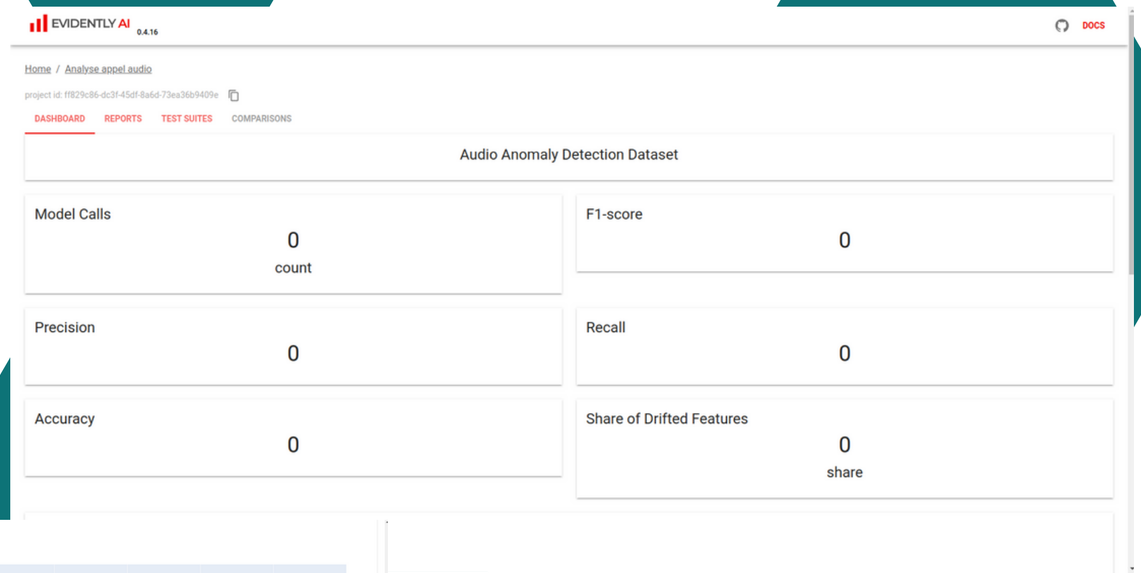
- Pour le feedback

```python
@app.get("/feedback")
async def feedback():
    # ouvrir le fichier prod_data.csv, modifier la dernière colonne de la d
    # si la valeur est 1, la mettre à 0, sinon la mettre à 1
    data = pd.read_csv("../data/prod-data.csv", sep=";",header=None)
    if data.iloc[-1, -1] == 0:
        data.iloc[-1, -1] = 1
    elif data.iloc[-1, -1] == 1:
        data.iloc[-1, -1] = 0
    data.to_csv("../data/prod-data.csv", index=False, sep=";",header=False)
```

Rapports & Monitoring

Evidently





MERCI