## 📝 **Rapport Technique : Détection de la Dépression par Apprentissage Fédéré Multimodal**

### 📚 ****Contexte****

Ce projet vise à développer un modèle de classification binaire permettant de détecter des signes de **dépression** à partir de données **audio (MFCC)** et **textuelles** (extraits de dialogues).  
L’approche adoptée repose sur l’**apprentissage fédéré (Federated Learning)** pour garantir la confidentialité des données, tout en exploitant un modèle d’apprentissage profond multimodal.

### 🧩 ****Données utilisées****

Le jeu de données est issu du corpus **DAIC-WOZ**, équilibré et prétraité.  
Les données sont divisées en trois ensembles distincts :

| **Ensemble** | **Utilisation** |
| --- | --- |
| **Train** | Entraînement local sur les clients |
| **Dev** | Validation globale (pendant les rounds) |
| **Test** | Évaluation finale après tous les rounds |

Le dataset d’entraînement est réparti entre **10 clients**, chacun recevant une portion équitable des données.

### 🧠 ****Architecture du modèle****

Le modèle est un réseau **LSTM multimodal**, composé de deux branches d’entrée :

* 🎙️ **Branche audio** : 3 couches LSTM successives pour extraire les traits à partir de MFCC.
* 💬 **Branche texte** : Vecteurs aplatis via une couche Flatten.
* 🔗 **Fusion** des deux branches via Concatenate, suivie d’un LSTM global, puis de couches Dense.

**Caractéristiques techniques :**

* **Fonction de perte** : RMSE (Root Mean Squared Error)
* **Optimiseur** : RMSProp (learning rate : 1e-5)
* **Métriques** : accuracy, mae, rmse
* **Activation finale** : sigmoid (car classification binaire)

### ⚙️ ****Paramètres d'entraînement****

* **Nombre de clients** : 10
* **Rounds fédérés** : 5
* **Batch sizes testés** : 8, 16, 32
* **Epochs locaux** : 2 par round
* **Échantillonnage** :
  + fraction\_fit = 0.8
  + fraction\_evaluate = 0.5
* **Clients requis** :
  + min\_fit\_clients = 10
  + min\_evaluate\_clients = 5

### 🔀 ****Stratégies fédérées testées****

| **Stratégie** | **Description** |
| --- | --- |
| **FedAvg** | Moyenne pondérée des poids des clients après chaque round. Méthode classique. |
| **FedProx** | Ajoute une pénalisation sur l’éloignement par rapport aux poids globaux. Robuste aux données non i.i.d. |
| **Scaffold** | Ajoute des vecteurs de correction (control variates) pour compenser les dérives locales. Utile si les données sont fortement hétérogènes. |

### 🔄 ****Algorithme utilisé : Apprentissage Fédéré Multimodal****

#### Étapes de l’algorithme :

1. **Initialisation du modèle global** sur le serveur.
2. **Répartition des données** sur les 10 clients (train uniquement).
3. **Entraînement fédéré** pendant plusieurs rounds :
   * Le serveur sélectionne les clients.
   * Chaque client entraîne son modèle localement.
   * Les modèles sont renvoyés et **agrégés selon la stratégie choisie**.
   * Le modèle global est **évalué sur le jeu de validation dev**.
4. **Évaluation finale** du modèle global sur le **jeu de test test**.
5. Génération d’une **matrice de confusion** pour visualiser les erreurs.

### 📊 ****Évaluation****

#### 🔁 Pendant les rounds :

* Évaluation à chaque round sur le **jeu de validation (dev)** centralisé.
* Métriques collectées :
  + accuracy
  + f1-score
  + precision
  + recall
  + mae
  + rmse

#### ✅ Après les rounds :

* Évaluation finale sur le **jeu de test**.
* Génération de la **matrice de confusion**.
* Visualisation via seaborn.heatmap.

### 🧮 ****Matrice de confusion (exemple)****

|  | **Prédit : 0** | **Prédit : 1** |
| --- | --- | --- |
| Réel : 0 | TN | FP |
| Réel : 1 | FN | TP |

### 🔍 ****Pseudo-code simplifié de l’algorithme****

python

CopierModifier

init global\_model

for each round in N\_rounds:

selected\_clients = sample\_clients(fraction\_fit)

client\_updates = []

for client in selected\_clients:

client\_model = global\_model

client\_model.train(local\_data, epochs=2, batch\_size=B)

client\_updates.append(client\_model.weights)

global\_model.weights = aggregate(client\_updates, strategy)

evaluate(global\_model, dev\_data)

evaluate(global\_model, test\_data)

show\_confusion\_matrix()