**🧠 Projet : Détection de la Dépression via Apprentissage Fédéré Multimodal (Texte + Audio)**

**1. 🎯 Objectif général**

Développer un modèle fédéré pour la **détection automatique de la dépression** en utilisant des **caractéristiques audio** (YAMNet) et **textuelles** (BERT), à partir du **DAIC-WOZ dataset**, tout en respectant la **confidentialité des données** grâce à **l’apprentissage fédéré** (FL).

**2. 📁 Données utilisées**

* **Fichier source** : daic\_paths.csv  
  Contient les chemins vers :
  + fichiers **audio WAV**
  + transcriptions **texte TSV**
  + étiquettes binaires : PHQ8\_Binary
* **Test set prédéfini** :  
  Chargé via test\_ids (1).npy, utilisé pour **évaluation finale uniquement**.
* **Dataset complet (audio + texte)** :
  + Audio : encodé avec **YAMNet** → vecteurs (100, 1024)
  + Texte : encodé avec **BERT** → vecteurs (64, 768)

**3. 🧪 Séparation des ensembles**

* **Test set** : défini par test\_ids, non utilisé pendant l'entraînement.
* **Dev set (validation)** :
  + 10% des données restantes (après exclusion du test set)
  + Échantillonné **de façon équilibrée entre les classes (0/1)**
* **Train set** :
  + Données restantes après soustraction du dev set
  + Réparties entre **10 clients** de manière uniforme

**4. 🏗️ Architecture du modèle**

* Entrées :
  + audio\_input → 3 couches LSTM
  + text\_input → Flatten puis concaténation
  + Fusion audio + texte → LSTM → Dense → sortie sigmoïde
* Sortie : prédiction de la probabilité de dépression (binaire)

**5. 🧪 Entraînement fédéré**

**Nombre de clients : 10**

**Nombre de rounds : 5**

**Évaluation :**

* À chaque round sur le **dev set**
* À la fin sur le **test set**

**6. ⚙️ Stratégies testées**

Nous avons comparé plusieurs stratégies de fédération avec des **paramètres variés** :

**✅ 6.1. FedAvg (standard)**

* **Algorithme de base** : Moyenne pondérée des poids clients
* **Batch size utilisé** : 32
* **Évaluation à chaque round sur dev set**

**✅ 6.2. SCAFFOLD**

* Objectif : corriger les **drifts locaux** sur les clients via des **correcteurs de gradient**.
* **Batch sizes testés** :
  + 8
  + 16
  + 32

Pour chaque batch size, la stratégie est testée sur 5 rounds, avec suivi des métriques (Accuracy, F1, MAE, RMSE…)

**✅ 6.3. FedProx**

* Objectif : ajouter une pénalité μ \* ||w - w\_t||² pour réduire la divergence locale des modèles
* **Valeurs de μ testées** :
  + 0.001
  + 0.01
  + 0.1
* Pour **chaque μ**, on teste avec :
  + batch size = 8
  + batch size = 16
  + batch size = 32

Cela donne **9 variantes de FedProx** testées

**7. 📊 Évaluation**

Pour chaque stratégie, on mesure :

* **Accuracy**
* **F1 Score**
* **Précision**
* **Recall**
* **MAE**
* **RMSE**

Les métriques sont :

* Enregistrées sur **dev set à chaque round**
* Mesurées **une seule fois sur le test set à la fin**

**✅ Résumé des combinaisons testées**

| **Stratégie** | **μ (si applicable)** | **Batch Sizes testés** |
| --- | --- | --- |
| FedAvg | - | 32 |
| SCAFFOLD | - | 8, 16, 32 |
| FedProx | 0.001, 0.01, 0.1 | 8, 16, 32 |

Nombre total de **configurations testées** :

* FedAvg (1)
* SCAFFOLD (3)
* FedProx (3x3 = 9)  
  → **Total : 13 configurations**

**📁 Stockage & logs**

* Résultats sauvegardés pour chaque round
* Les performances du **modèle final** (après 5 rounds) sont mesurées **sur le test set (défini par test\_ids.npy)**