# 1. Configuration et Prérequis

#### Configuration de l'Environnement

- Python 3.8 ou supérieur est installé.
- Installez les dépendances nécessaires listées dans requirements.txt.

## **Configuration du Projet**

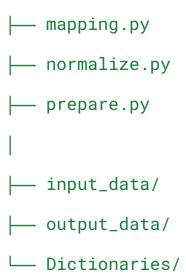
1. Naviguez vers le répertoire du projet.

## Installation des Dépendances

pip install -r requirements.txt

# 2. Structure du Projet

#### Structure du Répertoire



#### **Description des Fichiers**

- main.py: Orchestre l'ensemble du pipeline en exécutant séquentiellement les étapes et en gérant les chemins d'entrée/sortie.
- **constants.py**: Stocke toutes les constantes utilisées dans le projet, y compris les chemins de fichiers et les seuils par défaut.
- **Dockerfile**: Définit la configuration du conteneur Docker pour le déploiement du projet.
- **requirements.txt**: Liste des dépendances Python nécessaires pour exécuter le projet.
- **README.md**: Fichier de documentation fournissant des instructions et des détails sur le projet.
- **calculate.py**: Implémente la résolution des entités et le calcul de similarité entre les enregistrements.
- **describe.py**: Fournit des capacités de description des données, y compris des résumés statistiques et la génération de tables synthétiques.
- mapping.py: Gère les tâches de mapping des données, appliquant des mappings prédéfinis aux colonnes du DataFrame.
- **normalize.py**: Normalise les adresses en utilisant la géocodification et pré-traite les colonnes de chaînes dans les DataFrames.
- **prepare.py**: Traite les DataFrames en supprimant les colonnes avec un grand nombre de valeurs NaN et en filtrant les colonnes en fonction des mappings.
- input\_data/: Répertoire contenant les fichiers CSV d'entrée à traiter.

- **output\_data/**: Répertoire où les fichiers de sortie générés par le pipeline sont sauvegardés.
- **Dictionaries/**: Répertoire contenant les fichiers JSON utilisés pour les mappings de données.

#### 3. Instructions d'Utilisation

#### **Exécution du Pipeline**

- Assurez-vous que les fichiers CSV d'entrée sont placés dans le répertoire input\_data/.
- 2. Ajustez les mappings et les paramètres dans les fichiers de dictionnaire respectifs situés dans Dictionaries/.
- 3. Ouvrez un terminal et naviguez vers le répertoire du projet.
- 4. Exécutez la commande suivante pour lancer le pipeline :

#### python main.py

- 5. Suivez les sorties du terminal pour surveiller le progrès et la complétion de chaque étape du pipeline.
- 6. Les fichiers de sortie seront enregistrés dans le répertoire output\_data/ tel que spécifié dans constants.py.

# 4. Explication Détaillée des Composants

## Étape 1 : Chargement des Données et Mapping (main.py, mapping.py)

- main.py: Commence par charger les données d'entrée depuis input\_data/, puis applique les mappings définis dans mapping.py.
- mapping.py: Implémente DataFrameMapper pour charger les dictionnaires de mapping depuis Dictionaries/, appliquer les renommages de colonnes au DataFrame et le préparer pour un traitement ultérieur.

# Étape 2 : Description des Données (describe.py)

• describe.py: Fournit des insights détaillés sur les données traitées, y compris les

résumés statistiques (DataFrameDescriber) et génère des tables synthétiques à partir des fichiers CSV dans input\_data/.

#### Étape 3 : Préparation des Données (prepare.py)

• **prepare.py**: Nettoie et prépare les données en supprimant les colonnes avec un grand nombre de valeurs NaN (DataFrameProcessor) et filtre les colonnes en fonction des mappings.

## Étape 4: Normalisation des Adresses (normalize.py)

• **normalize.py**: Normalise les adresses en utilisant la géocodification (AddressNormalization), convertit les colonnes en minuscules et supprime les caractères non alphanumériques.

# Étape 5 : Résolution des Entités et Calcul des Similarités (calculate.py)

• calculate.py: Implémente le rapprochement en utilisant la vectorisation TF-IDF (EntityResolution) et calcule les scores de similarité (cosinus et Jaccard).