Mélanie Donne

# Rapport Etape 1

## Analyse de l'immobilier à Paris

Mélanie Donne

## Introduction

Ce rapport détaille les progrès réalisés dans le cadre du projet de l’analyse de l'immobilier à Paris.

L'objectif principal est de collecter des données brutes à partir de différentes sources d'API, puis de les nettoyer, de les prétraiter et de les préparer pour une analyse ultérieure.

Dans cette première étape, nous avons concentré nos efforts sur trois principaux aspects : la préparation des données foncières, l'extraction des données via des requêtes API et le nettoyage des résultats obtenus. À cette fin, nous avons développé trois scripts Python distincts : data-cleaning.py pour les données foncières, api-requests.py pour l'extraction des données et api-cleaning-results.py pour leur nettoyage et leur préparation.

Dans ce rapport, nous détaillerons les différentes étapes impliquées dans le processus de nettoyage et de prétraitement des données, en fournissant des explications détaillées sur le fonctionnement des scripts et en illustrant chaque étape avec des exemples concrets.

Mélanie Donne

# 1. Le nettoyage des données

Pour commencer, j'ai codé le script `data-cleaning.py` qui effectue plusieurs opérations de nettoyage et de prétraitement sur un jeu de données relatives aux valeurs foncières de l'année 2022.

Voici les différentes étapes réalisées par ce script :

## Chargement des Données :

Le script utilise la librairie Pandas pour charger les données du fichier `valeursfoncieres-2022.txt` dans un DataFrame, en spécifiant que les colonnes sont séparées par le caractère "|".

```
# Charger les données de 2022
vf2022 = pd.read_csv("./valeursfoncieres-2022.txt", sep="|", low_memory=False)
```

## Nettoyage des Colonnes avec des Données Manquantes :

Les colonnes contenant des valeurs manquantes sont identifiées, puis celles dont le pourcentage de valeurs manquantes dépasse 50% sont supprimées du DataFrame. Cette opération vise à éliminer les colonnes avec des données insuffisantes pour une analyse significative.

```
# Nettoyage des colonnes avec des données manquantes
# Identifier les colonnes avec des valeurs manquantes
columns_with_nan = vf2022.columns[vf2022.isna().any()]

# Parcourir les colonnes avec des valeurs manquantes
for column in columns_with_nan:
    # Calculer le pourcentage de valeurs manquantes dans la colonne
    percentage_missing = vf2022[column].isna().sum() * 100.0 / vf2022.shape[0]
      # Supprimer la colonne si le pourcentage de valeurs manquantes est
supérieur à 50%
    if percentage_missing > 50:
        vf2022.drop(column, axis=1, inplace=True)
```

Mélanie Donne

### Suppression des Colonnes Inutiles :

Certaines colonnes jugées inutiles pour l'analyse sont supprimées du DataFrame. Ces colonnes comprennent des informations telles que la section, le numéro de plan, le nombre de lots, etc.

```
# Supprimer les colonnes inutiles
useless_columns = ['Section', 'No plan', 'Nombre de lots', 'Code type local',
'Nature culture', 'No voie', 'No disposition']
vf2022.drop(useless_columns, axis=1, inplace=True)
```

### Nettoyage des Lignes avec des Données Manquantes :

Les lignes contenant au moins 14 valeurs manquantes sont supprimées du DataFrame. Cette opération vise à éliminer les lignes ayant un nombre significatif de valeurs manquantes.

```
# Nettoyage des lignes avec des données manquantes

# Supprimer les lignes avec au moins 14 valeurs manquantes

vf2022 = vf2022.dropna(axis=0, thresh=14)
```

### Conversion du Format de Données :

La colonne "Valeur fonciere" est nettoyée en remplaçant les virgules par des points pour assurer un format numérique approprié.

```
# Remplacer les virgules par des points dans la colonne "Valeur fonciere"

vf2022["Valeur fonciere"] = vf2022["Valeur fonciere"].str.replace(',', '.')
```

Mélanie Donne

### Conversion des Types de Données :

Certaines colonnes sont converties dans des types de données appropriés. Par exemple, la colonne "Valeur fonciere" est convertie en type numérique, la colonne "Surface reelle bati" est également convertie en type numérique, et la colonne "Code postal" est convertie en type entier (int32). De plus, la colonne "Date mutation" est convertie en type de données datetime.

```
# Convertir certaines colonnes en types appropriés

vf2022["Valeur fonciere"] = pd.to_numeric(vf2022["Valeur fonciere"])

vf2022["Surface reelle bati"] = pd.to_numeric(vf2022["Surface reelle bati"])

vf2022["Code postal"] = vf2022["Code postal"].astype({'Code postal': 'int32'})

vf2022["Date mutation"] = pd.to_datetime(vf2022["Date mutation"])
```

### Filtrage des Données pour Paris :

Les données sont filtrées pour inclure uniquement les codes postaux de Paris, définis dans la liste paris_postal_codes.

```
# Déterminer les codes postaux de Paris
paris_postal_codes = [75001, 75002, 75003, 75004, 75005, 75006, 75007, 75008,
75009, 75010,
                      75011, 75012, 75013, 75014, 75015, 75016, 75017, 75018,
75019, 75020]

# Filtrer les données pour n'inclure que les codes postaux de Paris
vf_paris2022 = vf2022[vf2022['Code postal'].isin(paris_postal_codes)]
```

Mélanie Donne

### Affichage des données de Paris :

Les cinq premières lignes des données de Paris sont affichées pour inspection.

```
# Afficher les cinq premières lignes des données de Paris

print(vf_paris2022.head())
```

### Enregistrement des Données dans un Fichier CSV :

Les données nettoyées et filtrées sont enregistrées dans un fichier CSV nommé "valeursfoncieres_paris_2022.csv", avec l'index des lignes omis et le format numérique des valeurs défini pour une précision décimale de 1, et en utilisant une virgule comme séparateur de champ et un retour à la ligne comme terminateur de ligne.

```
# Enregistrer les données organisées dans un fichier CSV
vf_paris2022.to_csv("valeursfoncieres_paris_2022.csv",index=False,float_format
='%.1f', sep=',', line_terminator='\n')
```

Ce script permet donc de nettoyer et de préparer les données relatives aux valeurs foncières de l'année 2022 pour une analyse ultérieure, en particulier en se concentrant sur les propriétés localisées à Paris.

Mélanie Donne

## 2. La récupération et la sauvegarde de données

Dans le cadre de notre analyse de l'immobilier à Paris, nous avons exploité diverses sources de données, comprenant notamment des données gouvernementales et municipales, ainsi que des API fournissant des informations pertinentes sur les biens immobiliers et leur environnement.
Ce script présente une solution pratique pour automatiser le processus de téléchargement de données CSV à partir de l'API Open Data de Paris. Il offre une manière efficace et rapide d'extraire des ensembles de données spécifiques, prêts à être utilisés pour des analyses approfondies, des visualisations ou d'autres applications dans le domaine de la science des données.

### Fonction download_csv_from_url(url, destination) :

Cette fonction gère le processus de téléchargement des fichiers CSV. Elle prend deux paramètres en entrée : url (l'URL du fichier CSV à télécharger) et destination (le nom du fichier de destination où enregistrer le fichier téléchargé). La fonction effectue une requête HTTP GET vers l'URL spécifiée, vérifie si la réponse est réussie (code de statut HTTP 200), puis crée le répertoire 'api_data' s'il n'existe pas déjà. Enfin, elle enregistre le contenu de la réponse dans un fichier binaire avec le nom de fichier de destination spécifié.

```
def download_csv_from_url(url, destination):
    try:
        response = requests.get(url)
        if response.status_code == 200:
            # Obtenir le chemin complet du répertoire 'api_data' relativement
au dossier du script Python
            api_data_dir = os.path.join(os.path.dirname(__file__), 'api_data')
            # Créer le répertoire 'api_data' s'il n'existe pas
            if not os.path.exists(api_data_dir):
                os.makedirs(api_data_dir)
            # Obtenir le chemin complet du fichier destination dans le dossier
'api_data'
            destination_path = os.path.join(api_data_dir, destination)
            with open(destination_path, 'wb') as f:
                f.write(response.content)
            print(f"Le fichier CSV '{destination}' a été téléchargé avec
succès dans le dossier 'api_data'.")
        else:
            print(f"Erreur lors du téléchargement du fichier '{destination}':
{response.status_code}")
    except Exception as e:
        print(f"Une erreur s'est produite lors du téléchargement du fichier
'{destination}': {str(e)}")
```

Mélanie Donne

### Liste des fichiers à télécharger (files) :

Dans cette partie du script, une liste `files` est définie, contenant des tuples où chaque tuple représente une paire d'URL de téléchargement et de nom de fichier de destination. Ensuite, une boucle itère à travers cette liste et appelle la fonction download_csv_from_url pour télécharger chaque fichier.

```
# Les URL des différents fichiers CSV avec leurs destinations dans le dossier
'api_data'
files = [

("https://opendata.paris.fr/api/explore/v2.1/catalog/datasets/arrondissements/
exports/csv?lang=fr&timezone=Europe%2FBerlin&use_labels=true&delimiter=%3B",
"arrondissements.csv"),

("https://opendata.paris.fr/api/explore/v2.1/catalog/datasets/quartier_paris/e
xports/csv?lang=fr&timezone=Europe%2FBerlin&use_labels=true&delimiter=%3B",
"quartier_administratifs.csv"),

("https://opendata.paris.fr/api/explore/v2.1/catalog/datasets/logement-encadre
ment-des-loyers/exports/csv?lang=fr&timezone=Europe%2FBerlin&use_labels=true&d
elimiter=%3B", "logement_encadrement_des_loyers.csv"),

("https://opendata.paris.fr/api/explore/v2.1/catalog/datasets/plu-secteurs-de-
risques-delimites-par-le-ppri/exports/csv?lang=fr&timezone=Europe%2FBerlin&use
_labels=true&delimiter=%3B",
"plu_secteurs_de_risques_delimites_par_le_ppri.csv")
]

# Télécharger chaque fichier dans 'api_data'
for file in files:
    download_csv_from_url(file[0], file[1])
```

Mélanie Donne

## 3. Les APIs

Les API sélectionnées ont été choisies en fonction de leur pertinence par rapport aux objectifs du projet.

En effet, ces dernières fournissent des données essentielles liées à l'urbanisme, à l'aménagement du territoire, au logement et à d'autres aspects de la vie urbaine qui sont au cœur de l'objet du projet. Par exemple, les données sur les arrondissements, les quartiers administratifs, les secteurs de risques, les espaces verts et les logements encadrés par des loyers sont tous des éléments cruciaux à prendre en compte lors de l'analyse de la dynamique urbaine de Paris.

De plus, ces informations sont mises à disposition par le gouvernement et sont accessibles publiquement, ce qui garantit la fiabilité et l'authenticité des données.

En résumé, ces API ont été choisies en raison de leur pertinence thématique, de la disponibilité de leurs données, de leur compatibilité avec les objectifs de l'analyse et de leur potentiel pour générer des insights significatifs pour le projet.

### Arrondissements

Cette source fournit des informations sur les arrondissements municipaux de Paris.

### Établissements Scolaires Maternelles

Cette source de données fournit des informations sur les écoles maternelles à Paris. Ces données comprennent des informations telles que L'année scolaire, le Code INSEE, le type d'établissement, son arrondissement, son adresse, etc.

### Établissements Scolaires Élémentaires

Cette source de données fournit des informations sur les écoles élémentaires à Paris. Ces données comprennent des informations telles que L'année scolaire, le Code INSEE, le type d'établissement, son arrondissement, son adresse, etc.

### Etablissements Scolaires Collèges

Cette source de données fournit des informations sur les collèges à Paris. Ces données comprennent des informations telles que L'année scolaire, le Code INSEE, le type d'établissement, son arrondissement, son adresse, etc.

### Logement - Encadrement des Loyers

Cette source fournit des informations sur les loyers de référence par quartier à Paris depuis la mise en place de l'encadrement des loyers en 2019.

### PLU - Espaces Libres à Végétaliser (ELV)

Cette source de données fournit des informations sur les espaces libres à végétaliser prescrits par le Plan Local d'Urbanisme (PLU) de Paris.

Mélanie Donne

## Espaces Verts protégés

Cette source fournit des informations sur les espaces verts protégés de la Ville de Paris.

## PLU - Secteurs de Risques Délimités par le PPRI

Cette source fournit des informations sur les secteurs de risques délimités par le Plan de Prévention du Risque d'Inondation (PPRI) mentionnés dans le PLU de Paris.

## Quartiers Administratifs

Cette source fournit des informations sur les quartiers administratifs de Paris.