JOURNAL DE BORD

Projet : Tableau de Bord Interactif - Open Data Logement

Amadou Aboubacar, Ndiaye Ibrahima
Master 2 MIASHS

Année universitaire 2024-2025

1 Présentation du Projet

L'objectif de ce projet était de concevoir un tableau de bord interactif permettant d'analyser l'évolution du parc de logements dans les départements du Gard (30) et de l'Hérault (34). Nous avons voulu créer un outil d'aide à la décision simple et visuel, basé sur les données ouvertes de l'INSEE.

Pour cela, nous avons exploité un historique de données allant de **2013 à 2022**, couvrant **691 communes**. Cela représente le traitement de **137 variables** différentes par commune (résidences principales, secondaires, vacance, typologie du bâti, statuts d'occupation, etc.).

Technologies Clés

Nous avons développé l'application en **Python** en utilisant les librairies suivantes :

- **Interface Web :** Streamlit (framework interactif).
- **Traitement des données :** Pandas, NumPy et GeoPandas (nettoyage et gestion spatiale).
- **Visualisation :** Plotly Express (graphiques interactifs) et Pydeck (cartographie WebGL).
- **Machine Learning :** Scikit-learn (clustering KMeans, régression) et SciPy (optimisation).

2 Architecture Technique

L'application est structurée de manière modulaire pour faciliter la maintenance et les évolutions futures. Le fichier principal `app.py` (1030 lignes) gère l'interface utilisateur et l'affichage, tandis que la logique d'Intelligence Territoriale est déportée dans le module `ml_models.py` (291 lignes).

L'interface utilisateur est organisée en **quatre onglets** pour guider l'utilisateur dans son exploration :

1. **Accueil :** Présentation du projet, indicateurs clés (KPI) et méthodologie.
2. **Cartographie :** Carte interactive permettant de visualiser 7 indicateurs différents.
3. **Analyse :** Fiches détaillées par commune avec 4 graphiques d'évolution temporelle.
4. **Intelligence Territoriale :** Clustering automatique et prédictions ML.

Les données sont stockées dans le dossier **SORTIE/** sous forme de fichiers CSV optimisés et d'un fichier GeoJSON pour la cartographie.

3 Déroulement du Développement

3.1 Phase 1 : Préparation des Données (Data Engineering)

Cette première étape a été la plus longue et la plus technique. Il a fallu nettoyer, harmoniser et transformer les données brutes pour les rendre exploitables. Nous avons travaillé en trois temps, répartis sur trois notebooks Jupyter.

1.1 Compilation des Bases Logement (Notebook : 1_compilation.ipynb)

Nous avons rassemblé les données de l'INSEE entre 2013 et 2022. La difficulté principale venait de l'hétérogénéité des fichiers sources : certains étaient en format Excel (.xlsx), d'autres en ancien format (.xls), et les noms des colonnes changeaient selon les années (par exemple `P22_LOG` pour 2022, `P21_LOG` pour 2021).

Nous avons écrit des scripts Python pour :

- Charger les 8 fichiers Excel en ignorant les 5 premières lignes (en-têtes INSEE).
- Supprimer les préfixes temporels des colonnes (`P22_`, `P21_`...) pour harmoniser les noms.
- Fusionner verticalement toutes les bases avec `pd.concat()`.
- Ajouter une colonne `AN` (année) pour identifier chaque millésime.

Ensuite, nous avons filtré les données pour ne garder que le Gard (30) et l'Hérault (34). Cela a permis de réduire la taille : nous sommes passés de **349 759 lignes** (toute la France) à **6 931 lignes** utiles (691 communes). Cette réduction de 98% était indispensable pour garantir la réactivité de l'application.

Enfin, nous avons calculé **13 variables dérivées** comme :

- **Parts dans le parc total :** Plog_RP (résidences principales), Plog_RS (résidences secondaires), Plog_VAC (logements vacants).
- **Parts dans les RP :** Prp_RP_PROP (propriétaires), Prp_RP_LOC (locataires).

Le fichier final `Compil_clean.csv` contient 147 colonnes et constitue la base de toutes nos analyses.

1.2 Traitement Cartographique (Notebook : Traitement.ipynb)

Pour afficher les communes sur une carte, nous avons utilisé les contours géographiques fournis par l'IGN. Le fichier source contenait les **35 000 communes de France**, que nous avons filtrées pour ne garder que nos 691 communes.

Deux défis techniques majeurs se sont posés :

- **Conversion des géométries :** Les formes des communes étaient codées en format WKB hexadécimal (dans une colonne nommée 'shape' avec chaînes de caractères incompréhensibles). Nous avons utilisé la librairie **Shapely** pour les transformer en objets Polygon exploitables : `shapely.wkb.loads(x, hex=True)`.
- **Reprojection cartographique :** Les coordonnées étaient en système métrique français Lambert-93 (EPSG :2154). Pour qu'elles s'affichent correctement sur une carte web, nous avons dû les convertir en système GPS standard WGS84 (EPSG :4326) avec GeoPandas.

Nous avons ensuite effectué une jointure spatiale entre les géométries et les données logement 2022 sur la clé `insee_com` (code INSEE commune). Le résultat est un fichier **GeoJSON**, optimisé pour le web et compatible avec notre application Streamlit.

1.3 Optimisation pour les Graphiques (Notebook : 2_tableaux.ipynb)

Pour que les graphiques s'affichent instantanément lors de la sélection d'une commune, nous avons préparé les données à l'avance. Au lieu de laisser l'application faire des calculs complexes en temps réel, nous avons créé **4 fichiers CSV spécifiques** (que l'on appelle des vues) où les données sont déjà restructurées au format 'long' (pivot_longer) :

- `TAB_TYPEHAB.csv` : Évolution Maisons/Appartements (13 862 lignes).
- `TAB_CATEHAB.csv` : Évolution RP/RS/Vacants (20 793 lignes).
- `RP_TYPO.csv` : Typologie T1-T5+ (34 655 lignes).
- `RP_SO.csv` : Statuts d'occupation 2022 (2 764 lignes).

Cette transformation permet à Plotly de générer les graphiques en moins d'une seconde, sans calcul intermédiaire.

3.2 Phase 2 : Développement de l'Interface (Streamlit)

Une fois les données prêtes, nous nous sommes concentrés sur la création de l'interface web avec le framework **Streamlit**. L'application est structurée en quatre onglets principaux, chacun répondant à un besoin spécifique.

2.1 Configuration Générale et Design

Nous avons configuré l'application en mode `wide` pour exploiter toute la largeur de l'écran. Un point crucial a été la mise en place du système de cache (`@st.cache_data`) : cela permet de charger le GeoJSON une seule fois en mémoire et d'éviter qu'il ne se recharge à chaque interaction utilisateur.

Pour le design, nous avons personnalisé l'interface avec du CSS :

- Palette harmonisée : Orange principal, Beige clair, Marron foncé.
- Police moderne : Google Font 'Inter' (sans-serif).
- Effets interactifs : Hover avec `translateY(-3px)`, transitions CSS fluides.

2.2 Onglet 1 : Accueil

Cet onglet sert de point d'entrée. Il affiche les **KPI départementaux** en 5 colonnes (nombre de communes, logements totaux, Résidence Principale, Résendence Secondaire, vacants) avec un design en cartes stylisées. Nous avons également ajouté des sections informatives expliquant notre méthodologie et un aperçu des données brutes pour assurer la transparence.

2.3 Onglet 2 : Cartographie Interactive (Pydeck WebGL)

C'est la partie technique la plus aboutie du projet. Au lieu d'utiliser des bibliothèques cartographiques classiques souvent lentes (comme Folium), nous avons choisi la technologie **Pydeck**, qui exploite le GPU via WebGL.

L'utilisateur peut choisir parmi **7 indicateurs** répartis en 3 groupes (Part des catégories de logement dans tout le parc, Part des catégories de résidences principales dans tout le parc, Part des types de locatifs (privé et public) dans les RP). La carte change de couleur dynamiquement selon un dégradé Jaune pâle → Rouge foncé calculé en temps réel.

Nous avons ajouté une barre de recherche pour trouver une commune facilement. Quand on la sélectionne, trois choses se passent :

- La vue se centre automatiquement sur la commune (calcul du centroïde).
- Une **bordure noire très épaisse** entoure la commune pour la rendre visible.
- Un marqueur texte affiche son nom au centre.

Une légende interactive affiche les valeurs min, max et moyenne de l'indicateur sélectionné, avec un trait vertical indiquant la position de la moyenne sur le dégradé de couleurs.

2.4 Onglet 3 : Analyse Détaillée par Commune

Cet onglet permet une exploration approfondie ville par ville. Quand on sélectionne une commune, l'application génère automatiquement **4 graphiques interactifs** avec Plotly :

i. **Évolution du parc 2013-2022 selon le type d'habitat :** Barres empilées (Maisons + Appartements) avec une ligne de tendance du total.
ii. **Répartition RP 2022 :** Camembert montrant les statuts d'occupation (Propriétaires, Locataires HLM, Locataires privé, Gratuit).
iii. **Évolution des logements selon leur catégories entre 2013-2022 :** Graphique en aires (RP, RS, Vacants).
iv. **Répartition des réqidences principales selon leur typologie en 2022 :** Barres horizontales montrant la répartition T1-T5+.

Tous les graphiques utilisent la même palette de couleurs harmonisée et sont configurés en mode `plotly_white` pour un rendu épuré.

2.5 Onglet 4 : Intelligence Territoriale (Machine Learning)

Pour aller plus loin dans l'analyse, nous avons développé deux fonctionnalités basées sur le machine learning :

- **Clustering KMeans :** Un algorithme regroupe automatiquement les communes similaires en 3 profils (K=3 optimal selon le Silhouette Score 0.55). Les variables utilisées sont : Plog_RP, Plog_RS, Plog_VAC, Plog_MAISON, Plog_APPART. Chaque profil est nommé automatiquement selon ses caractéristiques dominantes (ex : 'Profil littoral - résidences secondaires élevées').
- **Prédiction du parc de logements :** Un modèle de régression simple qui prédit l'évolution du nombre de logements sur 1 à 5 ans. Le meilleur modèle est sélectionné automatiquement selon le RMSE.

3.3 Phase 3 : Optimisation et Tests

Une fois l'application fonctionnelle, nous avons travaillé sur l'optimisation des performances et la correction des bugs.

Migration Folium → Pydeck

Au début du projet, nous utilisions la bibliothèque Folium pour la cartographie. Cependant, le temps de chargement était inacceptable : **plusieurs secondes** pour afficher 691 polygones. Après analyse, nous avons compris que Folium génère du SVG côté serveur, ce qui est très lourd.

Nous avons donc migré vers **Pydeck**, qui utilise le rendu WebGL côté client (GPU). Le gain a été important : le temps de chargement est devenu beaucoup plus rapide qu'avant.

4 Difficultés Rencontrées et Solutions

Ce projet ne s'est pas fait sans heurts. Voici les principaux obstacles que nous avons rencontrés et comment nous les avons surmontés :

1. Performance de la carte : Comme mentionné, Folium était trop lent. La migration vers Pydeck a demandé un apprentissage technique important (documentation en anglais, syntaxe complexe), mais le résultat en valait la peine.

2. Gestion des types de données : Les codes INSEE des départements étaient parfois en `int`, parfois en `str`. Cela causait des erreurs de filtrage. Nous avons systématisé la conversion en `str` avant toute comparaison : `dep_str = data['DEP'].astype(str)`.

3. Conflits Git : Travaillant en binôme sur le même fichier `app.py`, nous avons eu plusieurs conflits de fusion. Nous avons appris à mieux communiquer avant de faire des modifications importantes et à utiliser `git stash` pour sauvegarder temporairement nos changements.

4. Sélection automatique du nombre de clusters : Au début, nous fixions arbitrairement K=3 pour le clustering. Nous avons ensuite implémenté une sélection automatique basée sur le Silhouette Score (testé de K=2 à K=5), ce qui rend l'algorithme plus robuste.

5 Résultats et Insights Territoriaux

L'application finale permet d'identifier clairement des dynamiques territoriales :

- **Littoral méditerranéen :** Forte concentration de résidences secondaires (>40%), notamment à La Grande-Motte (68%). Vacance faible malgré la saisonnalité.
- **Centres urbains (Nîmes, Béziers) :** Vacance élevée (>15%), dominance des appartements (>60%), majorité de locataires.
- **Périurbain (couronnes Montpellier, Nîmes) :** Croissance soutenue (+2 à +3%/an), dominance des maisons (>80%), propriétaires majoritaires (>70%).
- **Rural (arrière-pays) :** Vacance très élevée (>20%), résidences secondaires importantes (15-30%), propriétaires ultra-dominants (>80%).

6 Bilan et Compétences Acquises

Ce projet a été très enrichissant sur le plan technique et méthodologique.

Compétences techniques développées :

- **Data Science :** Manipulation de données avec Pandas (merge, groupby, pivot), GeoPandas (reprojection CRS, jointures spatiales).
- **Machine Learning :** Clustering (KMeans, Silhouette Score), régression (linéaire), validation (R^2, RMSE).
- **Développement Web :** Streamlit (widgets, cache, session state), Pydeck (WebGL), CSS personnalisé.
- **Optimisation :** Réduction de 98% des données, cache mémoire, format long pour graphiques.

Compétences méthodologiques :

- Cycle itératif : Prototype → Test → Amélioration.
- Gestion de version avec Git (résolution de conflits).
- Documentation (code commenté, notebooks Jupyter).
- Déploiement de l'application sous streamlit (voici le lien de l'application : `https://opendataloge.streamlit.app/`).

Conclusion

L'application finale est opérationnelle, performante et répond aux objectifs fixés. Elle permet à n'importe quel utilisateur de comprendre en quelques clics la situation du logement dans sa commune. Ce projet démontre concrètement le potentiel de l'Open Data et des outils modernes de Data Science pour créer des outils d'aide à la décision accessibles et visuels.