Auteurs
Patrice BANAÏAS - Talib BATOU

Décembre 2025

Projet Python avancé

Analyse du marché immobilier
à partir de données issues du web scraping

Formation : SDA 2025 – 2026
Université Paris 1 Panthéon-Sorbonne – Formation Continue

Sommaire

INTRODUCTION

Contexte

Le marché immobilier français présente d'importantes variations de prix selon les territoires, les types de biens et les localisations. Comprendre ces disparités nécessite l'accès à des données fiables et leur analyse structurée.

Source des données

Nous avons collecté des annonces immobilières (maisons et appartements) à partir du site :

immobilier.notaires.fr

reconnu pour la qualité de ses informations.

Le scraping couvre 22 départements représentatifs du territoire français :

01, 06, 13, 21, 29, 30, 31, 33, 34, 35, 38, 42, 44, 49, 51, 59, 62, 69, 74, 75, 76, 83

Architecture du projet

Le projet s'articule autour de 4 composants Python :

Fichier	*Rôle*
scraper.py	Collecte des annonces
exploration.ipynb	Exploration et validation
clean_data.py	Nettoyage et géocodage
dashboard.py	Application streamlit interactive

Objectif

Proposer un outil simple, visuel et interactif permettant d'explorer les tendances du marché immobilier : carte des biens, analyses par territoire, comparaisons de prix au m^2.

scraper.py - Collecte des données

Source : immobilier.notaires.fr

Celui-ci reposant en partie sur un chargement dynamique des pages, l'utilisation de Selenium a été retenue afin d'accéder au contenu HTML complet des annonces.

Une fois les pages chargées, le contenu est analysé à l'aide de BeautifulSoup afin d'extraire les informations essentielles de chaque annonce (titre, description, prix, surface, nombre de pièces, localisation et type de bien).

Le scraping est réalisé de manière automatisée sur plusieurs pages et départements, puis les données collectées sont exportées dans un fichier CSV pour les étapes suivantes du projet.

Extrait de scraper.py

```python
# --- 1. Configuration du Projet ---
# Liste des départements à scraper
departements = ['75','01','13','69','31','06','44','34','33','59','83','42','76','21','38','35','51','49','30','29','74','62']
pages_to_scrape = 5
base_url = "https://www.immobilier.notaires.fr/fr/annonces-immobilieres-liste"

# Liste pour stocker tous les résultats
all_data = []

# --- 2. Configuration de Selenium (Mode Headless) ---
chrome_options = Options()
# Mode Headless (sans interface graphique) pour la rapidité et la stabilité
chrome_options.add_argument("--headless")
chrome_options.add_argument("--no-sandbox")
chrome_options.add_argument("--disable-dev-shm-usage")
# User-Agent pour éviter d'être bloqué
chrome_options.add_argument(
    "user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36")

```

Cet extrait présente le périmètre du scraping (départements et pagination) ainsi que la configuration de Selenium en mode headless pour charger les pages dynamiques du site et limiter les risques de blocage.

```python
        # --- Niveau 2 : Boucler sur chaque lien pour aller chercher le DETAIL ---
        for link in liens_annonces:
            try:
                driver.get(link)
                # Pause pour charger la page de détail dynamique
                time.sleep(random.uniform(2, 4))

                soup_detail = BeautifulSoup(driver.page_source, 'html.parser')
```

Pour chaque annonce, le script accède à la page de détail via le navigateur automatisé. Un temps d'attente est appliqué pour assurer le chargement correct de la page, puis le contenu HTML est analysé avec BeautifulSoup afin d'accéder à la structure de l'annonce.

Les informations essentielles de chaque annonce sont extraites à partir de la structure HTML de la page, notamment le titre, le type de bien, la localisation et la description lorsque celle-ci est disponible.

```python
                # --- Sauvegarde des données ---
                all_data.append({
                    'Departement': dept,
                    'URL': link,
                    'Titre': titre,
                    'Type_Bien': type_de_bien,
                    'Localisation': location,
                    'Prix': prix,
                    'Surface_m2': surface,
                    'Nb_Pieces': nb_pieces,
                    'Description': description
                })
```

Les données extraites sont regroupées dans une structure intermédiaire afin de constituer un ensemble homogène prêt à être exporté et analysé.

```python
    df = pd.DataFrame(all_data)
    df.to_csv('annonces_completes_notaires.csv', index=False, encoding='utf-8-sig')
```

Les données collectées sont structurées dans un DataFrame pandas puis exportées dans un fichier CSV, qui servira de base aux étapes de nettoyage, d'analyse et de visualisation.

exploration.ipynb - Exploration des données

Rôle du notebook

Le notebook exploration.ipynb permet d'analyser les données brutes issues du scraping avant leur traitement automatique.
Il sert à comprendre la structure, évaluer la qualité des données et valider les règles de nettoyage.

Visualisation des premières annonces collectées (prix, surface, localisation, description).

1. Aperçu général des données

```
df.head(3)
```

0.0s

	Departement	URL	Titre	Type_Bien	Localisation	Prix	Surface_m2	Nb_Pieces	Description
0	75	https://www.immobilier.notaires.fr/fr/annonce-...	Achat :Appartement T4- Paris 8 - Paris (75)	Appartement T4	- Paris 8 - Paris (75)	955000	72.12	4	Notaire - F4 Rue Berryer PARIS VIIIème. Au coe...
1	75	https://www.immobilier.notaires.fr/fr/annonce-...	Achat :Appartement T2- Paris 16 - Paris (75)	Appartement T2	- Paris 16 - Paris (75)	745500	70.00	2	Notaire vend un appartement T2 de 70.58m² Carr...
2	75	https://www.immobilier.notaires.fr/fr/annonce-...	Achat :Appartement T4- Paris 15 - Paris (75)	Appartement T4	- Paris 15 - Paris (75)	600000	74.15	4	Exclusivité, au sein d'un bel immeuble d'angle...

Extrait de : exploration.ipynb

Analyse des valeurs manquantes (nombre et pourcentage)

2.2 Valeurs manquantes

```
missing = df.isna().sum()
missing_pct = (missing / len(df) * 100).round(1)

pd.DataFrame({
    'Manquantes': missing,
    'Pourcentage': missing_pct
})
```

0.0s

	Manquantes	Pourcentage
Departement	0	0.0
URL	0	0.0
Titre	0	0.0
Type_Bien	0	0.0
Localisation	0	0.0
Prix	0	0.0
Surface_m2	150	11.6
Nb_Pieces	0	0.0
Description	0	0.0

150 valeurs manquantes sur Surface_m2 (11.6%)

→ À traiter dans `clean_data.py`

Analyse globale du jeu de données :
– volume d'annonces collectées,
– structure des colonnes et types de données,
– détection des doublons,
– identification des valeurs manquantes.

3. Traitement des surfaces manquantes

Hypothèse : La surface peut être extraite de la colonne `Description` via regex

3.1 Test du pattern regex principal

```
# Extraction de la surface depuis le texte à l'aide d'une expression régulière

pattern_m2 = r"(\d+(?:[.,]\d+)?)\s*m²"

missing_surface = df[df['Surface_m2'].isna()]

for i, row in missing_surface.head(3).iterrows():
    desc = row['Description'][:200]
    match = re.search(pattern_m2, desc)
    print(f"Description: {desc}...")
    print(f"Surface extraite : {match.group(1) if match else 'NON'}\n")
```

0.0s

```
Description: APPARTEMENT A VENDRE A PARIS 15EME EN EXCLUSIVITE ! N'attendez plus pour découvrir cet agréable appartement de type 3 d'environ 67 m2, au troisième étage avec ascenseur d'un immeuble datant de 1976 et...
Surface extraite : NON

Description: Vendu LOUÉ. Paris 20ème, Gambetta, dans un immeuble ancien à deux pas de la Place, appartement de 33,6m² au 5ème étage avec ascenseur. Plan en étoile pour cet appartement vendu loué 9870 +1050 de char...
Surface extraite : 33,6

Description: À VENDRE : Appartement idéalement situé dans le 15ème arrondissement de Paris, à 5 minutes de la Médiathèque Marguerite-Yourcenar, à 4 minutes du métro Vaugirard, et à seulement 22 minutes à pied de l...
Surface extraite : NON
```

Extraction des surfaces à l'aide d'expressions régulières (Regex) directement depuis la description des annonces, afin de corriger les valeurs manquantes.

```
# Pattern observé : "- Ville - Département (XX)"
# Exemple : "- Paris 8 - Paris (75)"

def extraire_ville(localisation):
    """Extrait le nom de la ville depuis la localisation"""
    # Retirer le tiret initial
    texte = localisation.lstrip('- ').strip()
    # Prendre la partie avant le second tiret
    if ' - ' in texte:
        return texte.split(' - ')[0].strip()
    return texte

# Test
print("Test d'extraction de ville :\n")
for loc in df['Localisation'].head(5):
    print(f"{loc} → {extraire_ville(loc)}")
```

0.0s

```
Test d'extraction de ville :

- Paris 8 - Paris (75) → Paris 8
- Paris 16 - Paris (75) → Paris 16
- Paris 15 - Paris (75) → Paris 15
- Paris 14 - Paris (75) → Paris 14
- Paris 16 - Paris (75) → Paris 16
```

Préparation de l'extraction de la ville pour le géocodage

Les informations de localisation étant fournies sous forme textuelle, cette étape vise à extraire le nom de la ville afin de préparer le géocodage (latitude / longitude) utilisé dans les analyses territoriales et la cartographie.

Cette phase d'exploration permet de formaliser précisément l'ensemble des règles de transformation nécessaires à la fiabilisation des données.
Ces règles seront ensuite implémentées dans le script clean_data.py afin de produire un jeu de données propre, structuré et prêt pour les analyses et la visualisation.

clean_data.py - Nettoyage et géocodage des données

Rôle du script clean_data.py

Le script clean_data.py transforme les données brutes issues du scraping en un jeu de données propre, structuré et exploitable, prêt à être utilisé dans l'application Streamlit.

Contrairement au notebook d'exploration, ce script constitue un pipeline de traitement automatisé, reproductible et orienté production.

Extrait de clean_data.py

```
"""
clean_data.py - Script de nettoyage des données immobilières

Ce script transforme les données brutes du scraping en un fichier propre
prêt pour l'application Streamlit.

Entrée  : data/annonces_raw.csv
Sortie  : data/annonces_clean.csv

Transformations effectuées :
1. Extraction des surfaces manquantes depuis les descriptions
2. Suppression des lignes sans surface
3. Création de la colonne Ville
4. Création de la colonne prix_m2
5. Géocodage des villes (latitude, longitude)

"""
```

Extraction intelligente des surfaces

```
    # --- Priorité 1 : Surface Carrez (mention légale, très fiable) ---
    carrez = re.search(r"(\d+(?:[.,]\d+)?)\s*m[²2]\s*(?:carrez|loi carrez)", desc, re.IGNORECASE)
    if carrez:
        return float(carrez.group(1).replace(',', '.'))

    # --- Priorité 2 : Surface habitable ---
    habitable = re.search(r"(\d+(?:[.,]\d+)?)\s*m[²2]\s*habitables?", desc, re.IGNORECASE)
    if habitable:
        return float(habitable.group(1).replace(',', '.'))

    # --- Priorité 3 : Formulations courantes "de X m²" ---
    # Capture : "appartement de 75 m²", "d'environ 80 m²", "d'une surface de 65 m²"
    formulation = re.search(
        r"(?:de|d'environ|d'une surface de)\s*(\d+(?:[.,]\d+)?)\s*m[²2]",
        desc, re.IGNORECASE
    )
    if formulation:
        val = float(formulation.group(1).replace(',', '.'))
        if val >= seuil_min:
            return val
```

L'une des étapes clés du nettoyage consiste à corriger les surfaces manquantes à partir du texte des annonces.
Le script implémente une extraction hiérarchisée et robuste à l'aide d'expressions régulières.
La logique repose sur :
une priorisation des sources fiables (Surface Carrez, surface habitable), **des seuils adaptatifs selon le type de bien** (studios vs T2+), **l'évitement des pièges courants** (balcons, caves, terrasses).

Structuration des données géographiques

```
def extraire_ville(localisation: str) -> str:
    """
    Extrait le nom de la ville depuis la colonne Localisation.

    Format attendu : "- Ville - Département (XX)"
    Exemple : "- Paris 8 - Paris (75)" → "Paris 8"

    Args:
        localisation: Chaîne de localisation brute

    Returns:
        Nom de la ville nettoyé
    """
    # Retirer le tiret initial et les espaces
    texte = str(localisation).lstrip('- ').strip()

    # Prendre la partie avant le second tiret
    if ' - ' in texte:
        return texte.split(' - ')[0].strip()
    return texte
```

Afin de permettre les analyses territoriales et la cartographie, le script extrait et normalise les informations de localisation.
Les transformations incluent :
extraction du nom de la ville depuis la colonne Localisation,
extraction du nom du département pour lever les ambiguïtés,
préparation des données pour le géocodage.

Géocodage des villes

```
def construire_query_geocodage(ville: str, nom_departement: str = "") -> str:

    # Pattern pour détecter Paris/Lyon/Marseille + arrondissement
    match = re.search(r"(paris|lyon|marseille)\s*0*(\d+)", str(ville).lower().strip())

    if match:
        nom_ville = match.group(1)
        arrondissement = int(match.group(2))

        # Codes postaux de base
        base_cp = {
            'paris': 75000,
            'lyon': 69000,
            'marseille': 13000
        }

        cp = base_cp.get(nom_ville, 0)
        if cp > 0:
            return f"{cp + arrondissement}, France"

    # Cas standard : ajouter le département pour plus de précision
    if nom_departement:
        return f"{ville}, {nom_departement}, France"
    else:
        return f"{ville}, France"
```

Le script enrichit ensuite les données avec des coordonnées GPS :
géocodage via Nominatim (OpenStreetMap),
gestion spécifique des arrondissements de Paris, Lyon et Marseille,
utilisation d'un cache pour limiter les requêtes et respecter l'API.

Résultat final du pipeline

À l'issue du traitement :
les annonces sans surface exploitable sont supprimées,
le prix au m² est calculé,
les coordonnées géographiques sont ajoutées,

un fichier final est généré pour l'analyse et la visualisation : annonces_clean.csv

Synthèse de la méthodologie

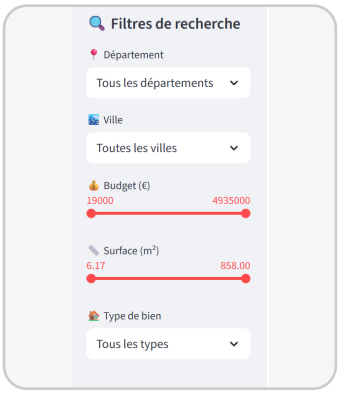
Le projet repose sur une chaîne de traitement complète et structurée des données immobilières. Il commence par le scraping automatisé des annonces depuis le site immobilier.notaires.fr, puis se poursuit par une phase d'exploration des données dans un notebook afin de mieux comprendre leur structure, leur qualité et les éventuels problèmes à corriger.

Les règles identifiées lors de cette exploration sont ensuite intégrées dans un script de nettoyage dédié, chargé de fiabiliser les données, de les enrichir (notamment sur le plan géographique) et de produire un jeu de données propre et cohérent. Cette organisation permet de bien séparer les rôles de chaque étape : exploration, nettoyage, enrichissement et préparation des données.

Les données finales ainsi obtenues peuvent alors être exploitées sereinement pour proposer des analyses statistiques et des visualisations interactives. La section suivante s'appuie sur ce jeu de données pour présenter une application Streamlit permettant d'explorer visuellement les tendances du marché immobilier.

dashboard.py – Application Streamlit

Les résultats du projet sont présentés à travers une application web interactive développée avec Streamlit, permettant d'explorer visuellement les données immobilières collectées et nettoyées. L'interface repose sur des filtres dynamiques et des visualisations interactives, offrant une analyse à la fois globale et détaillée du marché immobilier.



L'interface propose plusieurs sélecteurs interactifs permettant de filtrer les annonces par département, ville, type de bien (maison ou appartement), ainsi que par fourchette de prix et de surface à l'aide de curseurs.

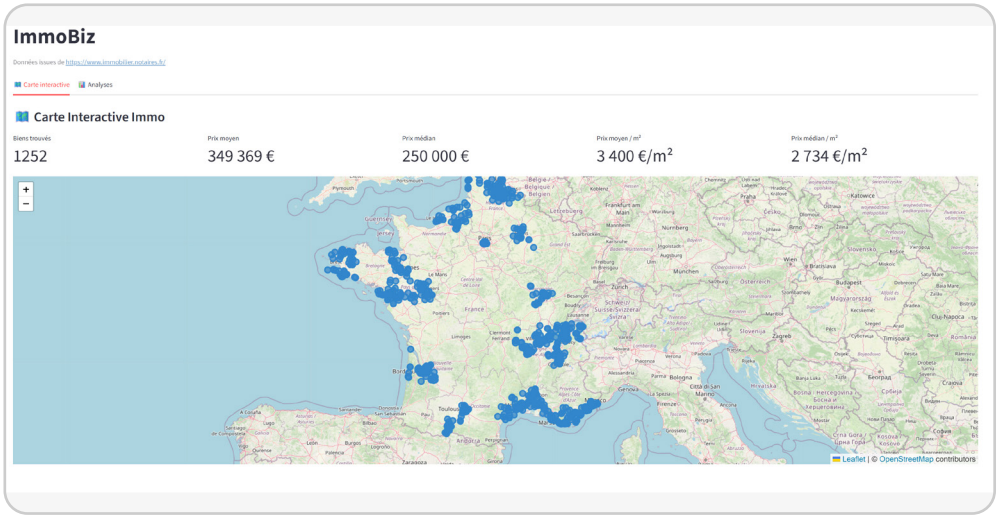
L'ensemble des visualisations se met à jour automatiquement en fonction des filtres sélectionnés, aussi bien pour la carte géolocalisée que pour les graphiques d'analyse.

L'application est organisée en deux onglets complémentaires.

Le premier onglet, **Carte interactive**, permet une exploration géographique des annonces, tandis que le second onglet, **Analyses,** propose des visualisations statistiques et comparatives des données.

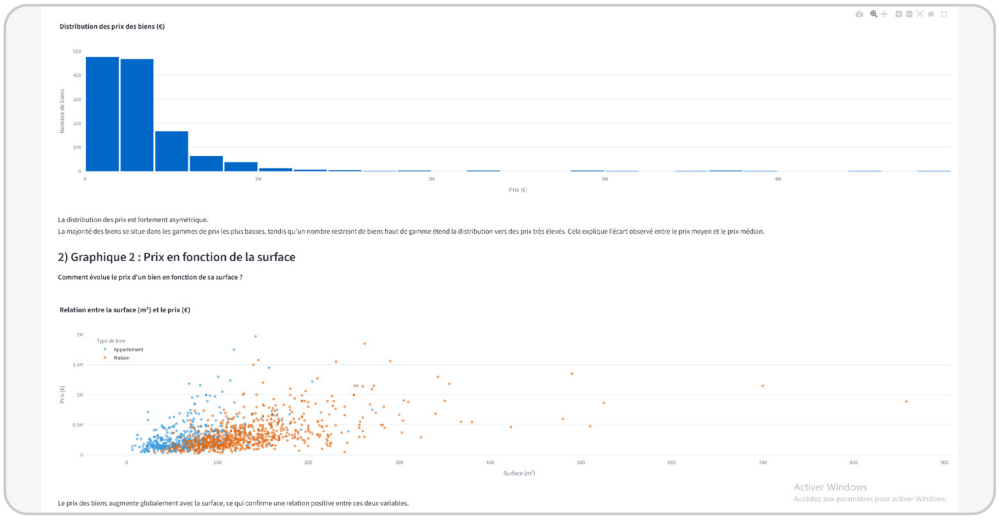
Carte interactive

Extrait de dashboard.py



Au-dessus de la carte, des indicateurs synthétiques permettent d'obtenir une lecture rapide du marché pour la sélection en cours.
Sont notamment affichés le nombre de biens trouvés, le prix moyen, le prix médian, ainsi que le prix moyen et le prix médian au m^2.

Analyses



L'onglet Analyses regroupe plusieurs visualisations statistiques permettant d'explorer les données en profondeur.
Il propose notamment la distribution des prix, la relation entre prix et surface, ainsi que des comparaisons du prix au m^2 par territoire, sous forme de graphiques et de distributions.

Conclusion

Ce projet a permis de mettre en place une chaîne complète de traitement et d'analyse de données immobilières, depuis la collecte automatisée des annonces jusqu'à leur visualisation au sein d'une application web interactive. Chaque étape a été pensée pour répondre à des problématiques concrètes de qualité des données, de structuration et de lisibilité des résultats.

La séparation claire entre le scraping, l'exploration, le nettoyage et la visualisation garantit une approche robuste, maintenable et évolutive. Les données finales obtenues sont fiables et suffisamment structurées pour permettre aussi bien des analyses globales que des explorations locales et détaillées du marché immobilier.

L'application Streamlit constitue l'aboutissement du projet en offrant une interface simple, interactive et accessible, permettant d'explorer les tendances du marché à travers des filtres dynamiques, une carte géolocalisée et des graphiques d'analyse. Ce projet illustre ainsi la capacité à concevoir une solution data complète, alliant traitement automatisé, analyse et restitution visuelle des données.