UNIVERSITÉ PARIS 1

PANTHÉON SORBONNE

État des lieux du marché locatif français

Web scraping, modélisation prédictive et interprétabilité des loyers par Random Forest et valeurs SHAP

Paulo Sergio Garcia Rodriguez

Numéro étudiant : 12522052

Paris, 2025

Table des matières

Table de figures

1. Introduction

1.1. Contexte général

Le marché locatif français présente une forte hétérogénéité, influencée par des facteurs géographiques, structurels et économiques. L'objectif de ce projet est de construire un pipeline complet permettant de collecter automatiquement des données issues d'une plateforme immobilière, de les nettoyer, d'extraire les variables pertinentes, puis de modéliser le loyer mensuel. Cette approche suit les méthodologies modernes de la data science, reposant sur l'automatisation, la reproductibilité et l'évaluation rigoureuse des modèles.

1.2. Etat de l'art

La modélisation des loyers s'inscrit dans une littérature ancienne portant sur l'analyse des prix immobiliers. Les approches classiques reposent sur les modèles hédoniques, formalisés notamment par Rosen (1974), qui expliquent le prix d'un logement comme une combinaison de ses caractéristiques structurelles et de sa localisation. Ces modèles ont longtemps constitué un cadre de référence en économie immobilière, mais ils reposent sur des hypothèses de linéarité et d'homogénéité des effets, souvent insuffisantes pour capturer la complexité et l'hétérogénéité des marchés réels.

Face à ces limites, des travaux plus récents ont mis en évidence l'intérêt des méthodes de Machine Learning pour améliorer la performance prédictive dans des contextes économiques complexes. Les modèles non paramétriques, et en particulier les Random Forests (Breiman, 2001), permettent de modéliser des relations non linéaires et des interactions entre variables, tout en offrant une bonne robustesse face aux valeurs atypiques et à la forte variabilité des données. Ces approches ont été largement mobilisées dans des travaux appliqués en économie lorsque l'objectif principal est la qualité prédictive plutôt que l'identification causale (Mullainathan & Spiess, 2017 ; Athey & Imbens, 2019).

Cependant, l'un des principaux inconvénients des modèles de Machine Learning réside dans leur interprétabilité limitée. Afin de répondre à cette contrainte, des méthodes d'explication locale ont émergé, parmi lesquelles les valeurs SHAP (Lundberg & Lee, 2017), qui s'appuient sur la théorie des jeux pour décomposer chaque prédiction individuelle en une valeur de référence et des contributions associées à chaque variable explicative. Ces méthodes permettent ainsi de concilier performance prédictive et compréhension fine du comportement du modèle.

Dans ce contexte, le présent travail se positionne comme une approche appliquée combinant collecte automatisée des données par web scraping**,** modélisation prédictive non linéaire et analyse d'interprétabilité**,** afin d'étudier le marché locatif français. L'objectif est de proposer une analyse à la fois performante sur le plan prédictif et éclairante quant aux mécanismes sous-jacents aux loyers observés, dans un marché caractérisé par une forte hétérogénéité géographique et structurelle.

1.3. Objectif du projet

Le projet vise à concevoir un modèle de Machine Learning afin de prédire le loyer total d'un logement à partir de données collectées en ligne. L'ensemble du pipeline — scraping, nettoyage, exploration, visualisation, modélisation et sélection finale du modèle — est entièrement automatisé grâce à une architecture modulaire en Python. Ce rapport présente et justifie chaque étape de cette démarche.

1.4. Approche méthodologique

Le travail repose sur quatre étapes principales : (i) extraction automatisée des annonces, (ii) nettoyage structurel et création de variables dérivées, (iii) analyse exploratoire permettant d'identifier les relations entre variables, (iv) modélisation via régression linéaire et Random Forest, suivie d'une analyse des résidus et du choix final du modèle.

1.5. Architecture du projet

L'architecture du projet repose sur une organisation modulaire claire. Le diagramme ci-dessous illustre le flux principal : collecte des données, nettoyage, analyse/statistiques et modélisation, puis orchestration générale assurée par le script principal.

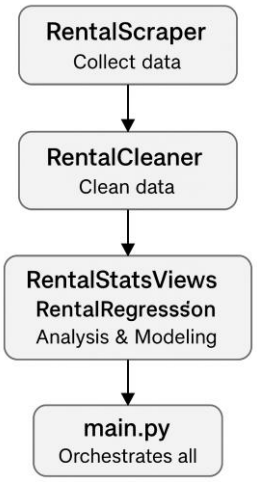

Figure 1- Diagramme UML de l'architecture du projet

- **RentalScraper** : module chargé de l'extraction automatisée des annonces immobilières depuis la plateforme *locamoi.fr* et de la constitution d'un premier jeu de données brut.
- **RentalCleaner** : assure le nettoyage des données, l'extraction des codes postaux, la génération des départements et la normalisation des variables.
- **RentalStatsViews** : produit l'ensemble des visualisations utilisées dans l'analyse exploratoire des données (EDA).
- **RentalRegression** : construit le pipeline de modélisation, entraîne les modèles, calcule les métriques, analyse les résidus et détermine l'importance des variables.
- **main.py** : script orchestrateur exécutant séquentiellement l'ensemble du pipeline au sein d'une interface Streamlit.

2. Collecte de données - Web Scraping

2.1. Source de données

Les données exploitées dans ce projet proviennent du site *locamoi.fr*, une plateforme spécialisée dans la diffusion d'annonces de locations immobilières en France. Ce site centralise un volume important d'offres issues de différentes zones géographiques et couvrant une large diversité de biens (appartements, maisons, studios, etc.), ce qui en fait une source pertinente pour analyser les dynamiques du marché locatif à l'échelle nationale.

Au total, environ 94 000 annonces de loyers ont été collectées lors de la phase de web scraping, sur un volume estimé à près de 100 000 annonces disponibles sur la plateforme. L'écart entre ces deux valeurs s'explique par la complexité croissante du processus de collecte pour les derniers pourcentages d'annonces, qui auraient exigé un effort technique disproportionné au regard du gain marginal en représentativité. L'échantillon obtenu, couvrant plus de 90 % des données accessibles, peut ainsi être considéré comme largement suffisant et statistiquement significatif pour les analyses menées dans ce projet.

Après un processus de nettoyage rigoureux — comprenant la suppression des doublons, l'élimination des valeurs aberrantes et la correction des incohérences — environ 80 000 observations exploitables ont été retenues pour l'analyse.

Cette volumétrie offre une représentation fiable du marché locatif français et garantit une robustesse statistique suffisante pour l'ensemble des étapes ultérieures, qu'il s'agisse de l'analyse exploratoire ou de la modélisation.

2.2. Outils utilisés

La collecte des données a été implémentée à l'aide d'une classe Python dédiée, nommée *RentalScraper*, conçue pour structurer, automatiser et sécuriser l'ensemble du processus de web scraping, depuis l'envoi des requêtes HTTP jusqu'à l'extraction, la validation et la sauvegarde des informations collectées.

```python
@dataclass
class RentalScraper:
    """Scraper for rental properties from the locamoi.fr website."""

    general_url: str = "https://locamoi.fr/location"
    url_suffixes: List[str] = field(default_factory=list)

    soup: Optional[BeautifulSoup] = None

    rows: List[Dict[str, Optional[str]]] = field(default_factory=list)
    dataframe: Optional[pd.DataFrame] = None

    session: requests.Session = field(init=False)
    logger: logging.Logger = field(init=False)
```

Figure 2 - RentalScraper class and attributs

2.2.1. Requêtes HTTP avec requests et gestion des erreurs

La bibliothèque Python *requests* a été mobilisée pour l'envoi des requêtes HTTP vers locamoi.fr. Une session persistante est initialisée dans la méthode __post_init__, garantissant une meilleure stabilité du scraping. Un mécanisme de *retry* a également été implémenté afin de gérer les timeouts, les erreurs serveur (HTTP 5xx) et les limitations de requêtes (HTTP 429), assurant ainsi la continuité du processus de collecte.

```python
self.session = requests.Session()
retries = Retry(
    total=5,
    backoff_factor=0.5,
    status_forcelist=[429, 500, 502, 503, 504],
    allowed_methods=["GET"],
    raise_on_status=False,
)
adapter = HTTPAdapter(max_retries=retries)
self.session.mount("https://", adapter)
self.session.mount("http://", adapter)
self.logger.info("Requests session initialized with retry adapters.")
```

Figure 3 - Création de la session requests

2.2.2. Analyse du contenu HTML avec BeautifulSoup

Après la récupération des pages web, l'extraction des données a été réalisée à l'aide de BeautifulSoup (bs4), permettant d'analyser la structure HTML et d'extraire les informations essentielles (prix, surface, localisation, pièces, type de bien). Une fonction dédiée, fetch_soup, automatise cette étape en vérifiant le code de retour HTTP avant le chargement du contenu ; en cas d'erreur, un objet vide est retourné pour éviter toute interruption. Le processus est complété par un système de journalisation (logging) assurant la traçabilité et la fiabilité de la collecte.

```python
def fetch_soup(self, url: str) -> None:
    """
    Fetch HTML content from `url` and update `self.soup`.

    - Performs a GET request using the session configured with retry logic.
    - Network-level exceptions (timeouts, connection errors) may bubble up and
      must be handled by the caller.
    - If the server returns a non-200 HTTP status, the response is logged and
      `self.soup` is set to an empty BeautifulSoup instance so that downstream
      code can safely run CSS selectors without raising errors.
    - On successful responses (HTTP 200), `self.soup` is populated with the parsed HTML.
    """
    self.logger.debug("Fetching URL: %s", url)


    response = self.session.get(
        url,
        timeout=(10, 40),
    )

    if response.status_code != 200:
        self.logger.warning("Non-200 status for %s: %s", url, response.status_code)
        self.soup = BeautifulSoup("", "html.parser")

        return

    html = response.text
    self.soup = BeautifulSoup(html, "html.parser")

    self.logger.debug(
        "Fetched %s (status=%s, bytes=%s)",
        url,
        response.status_code,
        len(html),
    )
```

Figure 4 - Méthode fetch_soup assurant la récupération sécurisée du contenu HTML avec gestion des erreurs et journalisation

2.3. Méthodologie et logique du scraping

Le processus de scraping repose sur une logique séquentielle permettant d'obtenir l'ensemble des informations nécessaires à partir du site locamoi.fr. La première étape consiste à récupérer les liens individuels des annonces. Pour cela, les pages de recherche du site sont parcourues par tranches de prix, et la méthode **get_url_suffixes** extrait puis enregistre, pour chaque annonce, le suffixe de son URL dans l'attribut url_suffixes. Ces suffixes sont ensuite convertis en URLs complètes grâce à la méthode **build_full_url**.

Une fois cette liste d'URLs constituée, les données de chaque annonce sont extraites à l'aide de la méthode **get_rental_data**, qui renvoie un dictionnaire structurant les informations essentielles du bien (prix, surface, localisation, type de propriété et nombre de pièces). Cette méthode met également à jour l'attribut rows, qui s'enrichit progressivement et permet, en fin de processus, de construire le dataframe final rassemblant l'ensemble des observations collectées.

L'ensemble du déroulement est orchestré par la méthode run_scraping, qui coordonne successivement la récupération des liens, la génération des URLs complètes et l'extraction détaillée des données pour chaque annonce. Le pipeline est par ailleurs renforcé par des mécanismes de gestion des erreurs, de tentatives automatiques (retry) et de journalisation (logging), garantissant la stabilité, la traçabilité et la fiabilité du processus de collecte sur un volume important de données.

```python
def run_scraping(self, resume: bool = False, start_index: int = 0) -> None:
    """
    Run the scraping process.

    If resume=False (default):
        - Collect URL suffixes
        - Scrape all properties from scratch

    If resume=True:
        - url_suffixes must already be populated
        - Resume scraping from start_index

    Results are accumulated in self.rows and converted to self.dataframe at the end.
    """
    if not resume:
        self.logger.info("Starting scraping run.")

        self.get_url_suffixes()

        self.logger.info("Starting properties scraping run.")
        print("\nStarting properties scraping run.\n")
        start_index = 0
        total = len(self.url_suffixes)

    else:
        total = len(self.url_suffixes)
        if start_index < 0 or start_index >= total:
            raise ValueError(
                f"start_index must be between 0 and {total - 1}, got {start_index}"
            )

        self.logger.info(
            "Resuming property scraping from %s/%s",
            start_index + 1,
            total,
        )
        print(f"\nResuming scraping from index {start_index} of {total}\n")
```

```python
    for counter in range(start_index, total):
        suffix = self.url_suffixes[counter]

        print(f"\nScraping progress: {counter + 1}/{total}")
        self.logger.info(
            "Scraping progress: %s/%s",
            counter + 1,
            total,
        )

        url = self.build_full_url(suffix)

        try:
            rental_dict_data = self.get_rental_data(url)

            if rental_dict_data.get("Address") is None and len(rental_dict_data) == 1:
                self.logger.warning(
                    "Empty or invalid data for URL %s at page %s. Skipping.",
                    url, counter + 1
                )

                continue

            self.rows.append(rental_dict_data)

        except Exception as e:
            self.logger.warning(
                "Failed to process URL %s at page %s. Skipping. Details: %s",
                url, counter + 1, e
            )

            continue

    self.dataframe = pd.DataFrame(data=self.rows)

    self.logger.info(
        "Scraping finished. Total rows=%s",
        len(self.dataframe) if self.dataframe is not None else 0,
    )
```

Figure 5 - Pipeline de Scraping de données

2.4. Problèmes rencontrés

Le site locamoi.fr impose une contrainte structurelle majeure : la consultation est limitée à 50 pages par recherche, soit un maximum de 1 000 annonces (20 par page). Pour contourner cette restriction, les requêtes ont d'abord été segmentées en intervalles de prix d'un euro, permettant d'accéder à des ensembles d'annonces distincts pour chaque tranche fine. Cette stratégie a permis de collecter environ 94 % des annonces disponibles. Les 6 % restants auraient exigé des segmentations additionnelles (par type de bien ou surface), dont la combinatoire aurait entraîné un volume de requêtes disproportionné au regard du gain analytique attendu.

Afin d'optimiser davantage le processus, une seconde logique de segmentation a été introduite pour les loyers élevés. Conformément à la structure montrée dans le code, dès que le seuil de 5 000 € était atteint, l'algorithme augmentait automatiquement l'amplitude des intervalles de prix : les tranches passaient alors de 1 € à 1 000 €, et ce jusqu'à 40 000 €**,** correspondant à la catégorie des biens les plus onéreux (notamment les immeubles). Cette accélération contrôlée permettait d'éviter une explosion du nombre de requêtes tout en garantissant la couverture intégrale des annonces très haut de gamme.

En outre, le scraping a été réalisé en mode synchrone, les solutions asynchrones testées se heurtant à des limitations opérationnelles : erreurs non interceptées, timeouts multiples, blocages serveur et difficulté accrue à assurer la traçabilité des logs. Malgré un temps d'exécution plus élevé, l'approche synchrone offrait une stabilité nettement supérieure et un contrôle plus rigoureux du pipeline de collecte.

Finalement, la nature dynamique du site a rendu certains liens temporairement inaccessibles, ce qui a nécessité la mise en place de mécanismes de sécurisation : système de *retry*, contrôle systématique des statuts HTTP, création d'un objet *BeautifulSoup* vide en cas d'erreur et journalisation complète afin d'assurer une reprise fiable du scraping.

3. Nettoyage et preparation de données

3.1. Aperçu du dataset brut

L'inspection initiale des données met en évidence plusieurs problématiques classiques associées au web scraping. On observe notamment la présence de doublons, résultant de la republication d'annonces identiques, ainsi que de nombreuses valeurs manquantes dans certaines variables essentielles, telles que la surface ou le type de bien. Les formats se révèlent également hétérogènes, certaines valeurs intégrant par exemple des unités textuelles comme « 30 m² ». Enfin, les données comportent des valeurs extrêmes ou incohérentes — prix égaux à zéro, surfaces négatives, etc. — caractéristiques d'erreurs de saisie ou de collecte.

Indice	Address	Type de bien	Pièces	Surface	Prix
0	14 Rue du Hoc, 76610 Le Havre, France	appartement	1	26 m²	380 €
1	23 Rue de l'Abbaye, 91800 Brunoy, France	appartement	1	33 m²	695 €
2	60000 Beauvais, France	appartement	2	59 m²	770 €
3	1155 Route de la Guitterie, 49370 Bécon-les-Granits, France	appartement	1	95 m²	1 250 €
4	81600 Gaillac, France	appartement	1	20 m²	500 €
5	88310 Cornimont, France	appartement	4	82 m²	610 €
6	74 Avenue du Maréchal Leclerc, 33400 Talence, France	appartement	1	25 m²	520 €
7	25 Rue du Moulin, 92800 Puteaux, France	studio	1	125 m²	3 210 €
8	2 Rue du General Leclerc, 16120 Châteauneuf-sur-Charente, France	maison	1	66 m²	630 €
9	1819 Route de la Meynardie, 24410 Saint Privat en Périgord, France	appartement	1	52 m²	550 €

Figure 6 - Dataset brut obtenu du processus de scraping

3.2. Traitement de doublons et nettoyage de variables numériques

L'étape de préparation des données a d'abord consisté à identifier et supprimer les doublons, fréquents dans les données issues du web scraping, qu'il s'agisse d'annonces répétées sur plusieurs pages ou associées à des URL différentes mais décrivant le même logement. Leur suppression a permis de réduire sensiblement le volume initial tout en préservant l'intégrité statistique du jeu de données.

Le travail s'est ensuite concentré sur le nettoyage et l'harmonisation des variables numériques principales — prix, surface, prix au m² et nombre de pièces. Les formats textuels (tels que « 45 m² » ou « 1 200 € ») ont été standardisés afin d'extraire uniquement les valeurs numériques, puis convertis dans un format exploitable pour l'analyse. La variable département a été reconstruite automatiquement à partir de l'adresse complète, grâce à la correspondance entre codes postaux et départements.

Plusieurs règles de filtrage ont été appliquées pour éliminer les observations impossibles ou incohérentes : retrait des logements de moins de 9 m², conformément à l'article R.111-2 du Code de la construction et de l'habitation ; exclusion des loyers inférieurs à 300 €, généralement liés à des erreurs de saisie ou à des biens non résidentiels ; suppression des annonces parisiennes dépassant 3 500 €, valeurs extrêmes associées à des anomalies de scraping ; enfin, élimination des enregistrements pour lesquels le loyer total était inférieur à cinq fois la surface, seuil permettant de détecter des incohérences manifestes.

Des statistiques descriptives et des visualisations avant/après nettoyage ont permis de valider l'efficacité de ces corrections et d'identifier d'éventuelles anomalies résiduelles. Cette étape assure ainsi la cohérence, la fiabilité et la robustesse du dataset, constituant une base solide pour les analyses exploratoires et les modèles de régression développés par la suite.

3.3. Création de nouvelles variables

Afin d'améliorer la pertinence des analyses statistiques et des modèles de régression, plusieurs variables dérivées ont été créées :

- Prix au m², permettant une normalisation par la surface ;
- Log (Prix) et log (Prix/m²), afin de réduire l'asymétrie et la variabilité des distributions ;

Ces transformations ont démontré un effet notable sur la stabilisation des distributions et des résidus des modèles.

3.4. Dataset final après nettoyage

À l'issue du processus de nettoyage, le dataset final contient environ 80 000 observations et un ensemble harmonisé de variables numériques et catégorielles, prêtes à être analysées. Les distributions y apparaissent plus régulières, et les relations entre variables sont plus faciles à interpréter.

Ce dataset constitue désormais une base solide pour la phase d'analyse exploratoire **(chapitre 4)** ainsi que pour la modélisation prédictive **(chapitre 5 et suivants)**.

Indice	Dept	Property Type	Rooms	Surface	Price	log (Price)	Price per m²	log (Price per m²)
0	76	appartement	1	26	380	5.94017	14.6154	2.68207
1	91	appartement	1	33	695	6.54391	21.0606	3.0474
2	60	appartement	2	59	770	6.64639	13.0508	2.56885
3	49	appartement	1	95	1250	7.1309	13.1579	2.57702
4	81	appartement	1	20	500	6.21461	25	3.21888
5	88	appartement	4	82	610	6.41346	7.43902	2.00674
6	33	appartement	1	25	520	6.25383	20.8	3.03495

Figure 7 - Aperçu du dataset final

4. Analyse Exploratoire de Données (EDA)

4.1. Statistiques descriptives

Indice	metrics	count	mean	std	min	25%	50%	75%	max	mode	skewness	kurtosis
0	Rooms	81872	1.86434	1.24446	1	1	1	3	10	1	1.60257	2.80028
1	Surface	81872	66.1687	70.136	9	39	56	78	4760	40	20.0876	778.857
2	Price	81872	1220.19	1618.72	300	565	789	1250	40000	850	8.39531	113.064
3	log (Price)	81872	6.80586	0.669427	5.70378	6.33683	6.67077	7.1309	10.5966	6.74524	1.17777	1.82492
4	Price per m²	81872	21.1023	23.2016	5	10.8333	14.0299	21.6667	2407.08	12.5	22.6481	1813.83
5	log (Price per m²)	81872	2.80296	0.627906	1.60944	2.38263	2.64119	3.07577	7.78617	2.52573	1.03248	1.08406

Figure 8 - Tableau de statistiques descriptives

Dans le tableau ci-dessus (**figure 8**) figurent les statistiques descriptives du jeu de données final présenté à la section **3.4** dans la **figure 7**.

On observe tout d'abord un écart particulièrement important entre les valeurs minimale et maximale du prix (300 € contre 40 000 €). Cette amplitude témoigne de la présence de biens extrêmement haut de gamme au sein d'un marché majoritairement dominé par des logements plus standards. Dans ce contexte, la valeur moyenne (1220,19 €) apparaît peu représentative de la tendance centrale, car fortement influencée par les outliers. La médiane (789 €) constitue ainsi un indicateur plus fiable de la distribution réelle des loyers.

La surface présente une dynamique similaire. Bien que la moyenne atteigne 66 m², la distribution se caractérise par une forte dispersion (écart-type de 70 m²) et par la présence de biens atypiques, comme en témoigne le maximum de 4760 m². Là encore, la médiane (56 m²) reflète plus fidèlement la réalité du parc immobilier, composé majoritairement de logements de petite taille.

Concernant le nombre de pièces, la mode égale à 1 indique que le type de logement le plus répandu est le studio ou T1. Cette observation est cohérente avec la distribution des surfaces, puisque 50 % des biens possèdent une superficie inférieure à 56 m².

Les mesures d'asymétrie (skewness) renforcent ces constats. Les variables telles que *Surface*, *Price* et *Price per m²* présentent une skewness fortement positive, indiquant des distributions étalées vers la droite, dominées par un petit nombre de valeurs exceptionnellement élevées. Cela confirme une fois de plus que la moyenne est un estimateur biaisé dans ce contexte et qu'il convient de privilégier la médiane ou des transformations logarithmiques pour obtenir une représentation plus robuste des données.

Les coefficients d'aplatissement (kurtosis) mettent également en évidence une distribution leptokurtique, caractérisée par une concentration d'observations proches de la moyenne et la présence concomitante d'outliers extrêmes. Les kurtosis particulièrement élevées de *Surface* et *Price per m²* montrent que ces variables contiennent un nombre anormalement important de valeurs atypiques, ce qui renforce l'intérêt de transformations logarithmiques ou de méthodes robustes pour les analyses ultérieures.

Enfin, les variables transformées par le logarithme (*log (Price)* et *log (Price per m²)*) affichent une asymétrie et une kurtosis nettement plus faibles, ce qui confirme que ces transformations permettent de stabiliser la variance, réduire l'influence des outliers et rapprocher les distributions de la normalité. Elles seront donc particulièrement adaptées dans le cadre de modélisations prédictives ou d'analyses statistiques avancées.

4.2. Distribution de loyers et effet de la transformation logarithmique

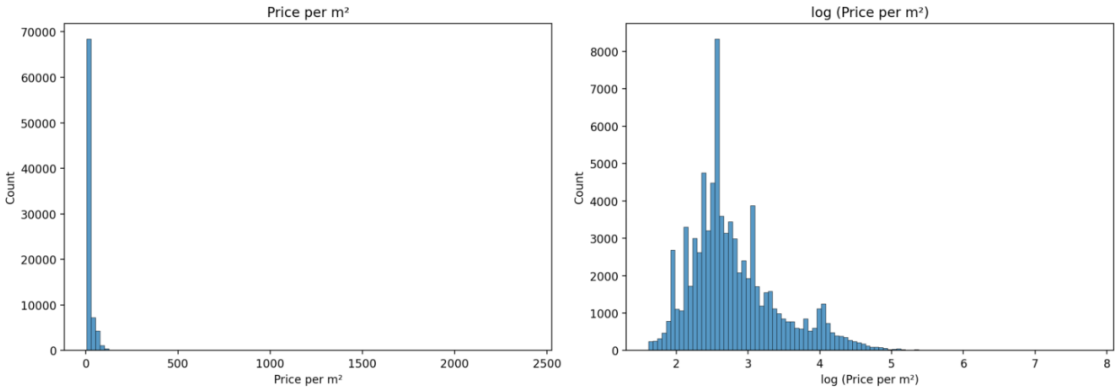

Figure 9 - Comparaison des distributions de prix avant et après la transformation logarithimique

La **figure 9** ci-dessus présente la distribution du prix au mètre carré avant et après transformation logarithmique.

À gauche, on observe que les valeurs brutes sont fortement asymétriques, avec une masse concentrée dans les faibles niveaux de prix au m², tandis que quelques observations très élevées étirent la distribution vers la droite. Ce comportement est caractéristique des données économiques, où les phénomènes de marché génèrent souvent des distributions à queue lourde.

L'application d'une transformation logarithmique — visualisée à droite — permet de réduire l'influence des valeurs extrêmes et de rapprocher la distribution d'une forme plus régulière. Bien que la distribution ne soit pas parfaitement normale, elle devient nettement plus symétrique, plus compacte, et présente une variabilité plus stable.

Cette transformation facilite l'interprétation statistique des données et constitue une étape couramment utilisée pour analyser des variables économiques fortement asymétriques comme les prix immobiliers.

4.3. Relation surface – prix

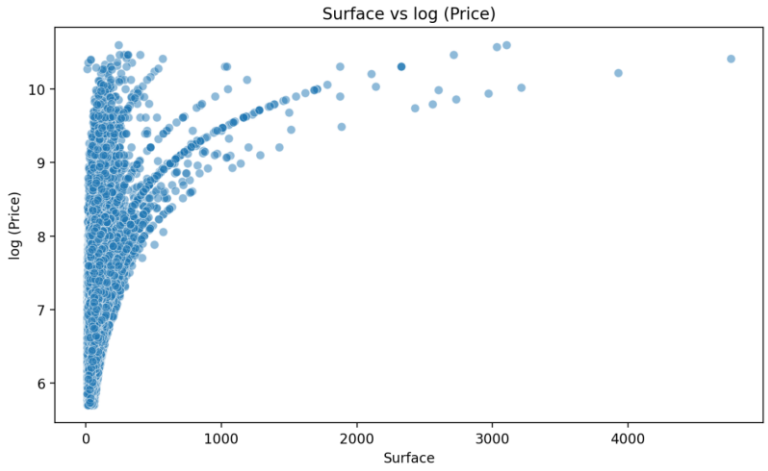

Figure 10 - Relation de la surface versus le log (Price)

La **figure 10** ci-dessus illustre la relation entre la surface du logement et le logarithme du loyer total. Pour la grande majorité des biens (entre 9 et 300 m²), on observe une tendance croissante nette : les logements plus grands présentent des loyers plus élevés, ce qui reflète logiquement la structure du marché locatif.

Cependant, cette progression n'est pas linéaire en valeur brute. L'utilisation de log (Price) permet de révéler une relation plus régulière, montrant que le prix augmente avec la surface, mais à un rythme décroissant. Autrement dit, le coût supplémentaire par mètre carré diminue pour les grandes surfaces, un phénomène classique dans l'immobilier.

Cette forme est cohérente avec les propriétés d'une distribution log-normale : La réduction de l'influence des valeurs extrêmes et ce qui met en évidence la structure fonctionnelle sous-jacente entre surface et prix.

Enfin, quelques valeurs très élevées (> 1000 m²) se démarquent, mais elles restent marginales. Leur légère irrégularité s'explique par le très faible nombre de biens concernés et par leur forte hétérogénéité (bâtiments complets, biens atypiques, locaux transformés, etc.).

4.4. Analyse du type de propriété

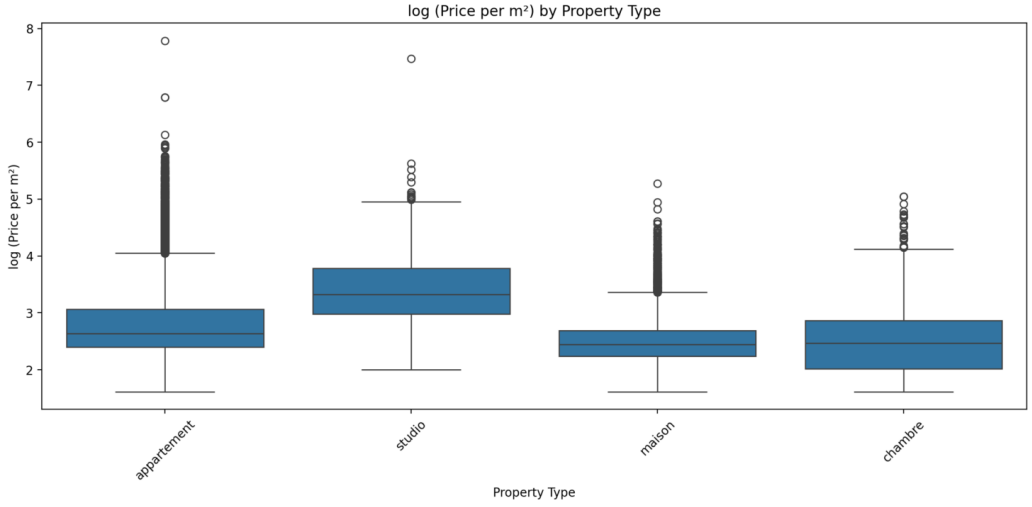

Figure 11 - Boxplot de Prix par m² selon le type de propriété

Le boxplot du logarithme du prix au mètre carré par type de propriété met en évidence des différences structurelles marquées entre les segments du marché locatif. Les studios présentent la médiane la plus élevée, confirmant une tendance bien établie du marché français selon laquelle les petites surfaces affichent généralement un prix au mètre carré plus important. Cette catégorie se caractérise également par une dispersion notable dans la partie supérieure de la distribution, comme en témoigne une distance interquartile $Q3 -$ médiane relativement élevée, sans pour autant être dominée par des valeurs extrêmes. Cette configuration reflète la diversité des studios, fréquemment situés dans des zones très urbanisées, où des facteurs tels que la localisation ou le niveau de standing peuvent entraîner des niveaux de prix élevés.

Les appartements, en comparaison, présentent une médiane de prix au mètre carré inférieure à celle des studios, mais une dispersion plus importante, caractérisée par un intervalle interquartile plus large et la présence de nombreuses valeurs extrêmes. Cette distribution traduit une forte hétérogénéité de ce segment, qui regroupe des biens très variés en termes de surface, de localisation et de caractéristiques qualitatives, conduisant à des écarts de prix au mètre carré particulièrement marqués.

Les maisons affichent les valeurs médianes les plus faibles, ce qui est cohérent avec l'existence d'un effet de décroissance du prix marginal au mètre carré pour les grandes surfaces.

Enfin, les logements de type « chambre » constituent un cas spécifique. Pour cette typologie, le loyer correspond au prix de la chambre louée individuellement, tandis que la surface utilisée pour le calcul du prix au mètre carré est généralement celle du logement dans son ensemble. Ce mode de calcul conduit mécaniquement à des valeurs de prix au mètre carré plus faibles et rend ces observations difficilement comparables à celles des logements occupés de manière exclusive. Cette mesure ne traduit donc pas un prix intrinsèquement plus bas, mais reflète une mutualisation de l'espace et un droit d'usage partiel du logement. La dispersion relativement limitée observée pour les chambres suggère par ailleurs un segment de marché plus homogène, caractérisé par des pratiques locatives standardisées et des niveaux de loyers moins dispersés que pour les studios ou les appartements.

4.5. Analyse du nombre de pièces

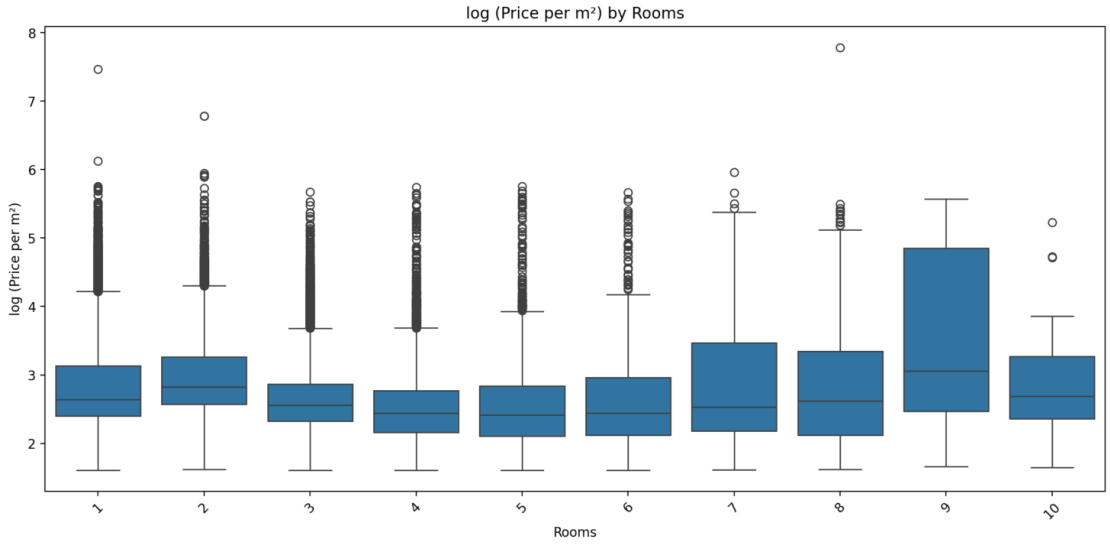

Figure 12 - Boxplot de Prix par m² selon le nombre de pièces

Le boxplot du log(prix/m²) par nombre de pièces montre que la majorité des catégories (de 1 à 8 pièces) présentent des niveaux très proches, avec des médianes et des dispersions globalement similaires. Cela suggère que, une fois la surface prise en compte, le nombre de pièces n'influence que marginalement le prix au m². La seule exception notable concerne les logements de 9 pièces, qui affichent une médiane sensiblement plus élevée et une dispersion plus large. Cette catégorie correspond probablement à des biens atypiques ou haut de gamme, pour lesquels le prix au m² ne suit plus la structure classique observée dans les logements plus standards.

4.6. Analyse du département (localisation du bien)

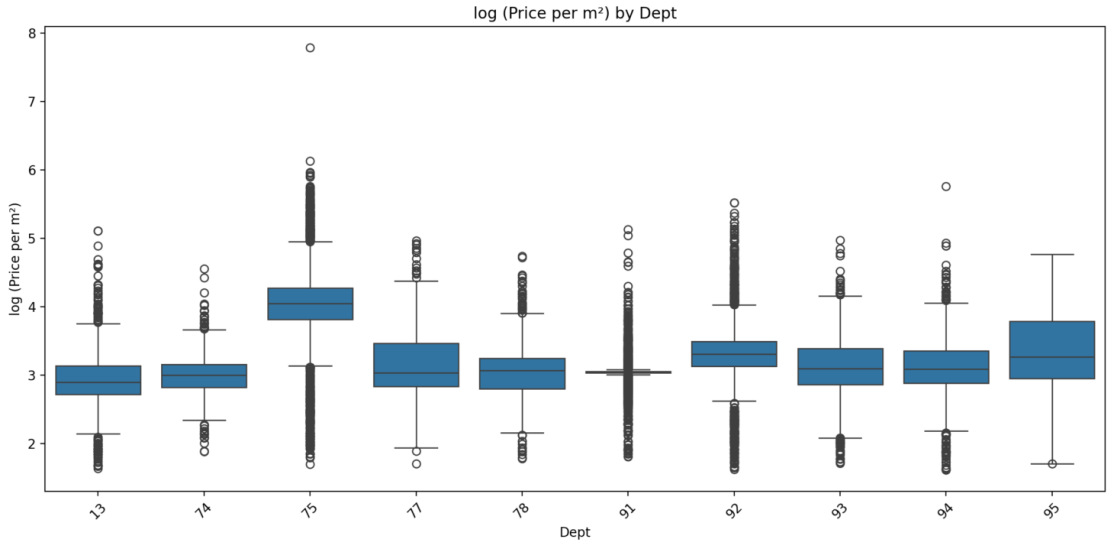

Figure 13 - Boxplot de prix par m² selon le departement

L'analyse du log(price/m²) par département (top 10 selon la médiane) montre sans surprise que Paris (75) présente les valeurs les plus élevées du pays. La médiane y est nettement supérieure à celle des départements voisins d'Île-de-France (92, 93, 94, 95) ainsi qu'à celles des départements plus éloignés. Toutefois, malgré ce niveau de prix très élevé, la distribution parisienne apparaît relativement homogène : l'écart entre la médiane et le troisième quartile (Q3) est comparable à celui entre Q1 et la médiane. Cette symétrie indique un marché parisien à la fois onéreux mais moins dispersé, où les variations de prix au m² sont plus contenues que dans les départements voisins, pourtant eux aussi caractérisés par des niveaux de prix élevés mais plus hétérogènes.

5. Modélisation et Machine Learning

5.1. Problème de régression

L'objectif du modèle est de prédire le loyer total à partir de caractéristiques structurales du logement (surface, nombre de pièces), de variables catégorielles (type de propriété, département) et de transformations statistiques. Il s'agit d'un problème de régression supervisée où la variable cible est continue.

5.2. Séparation train/test

Les données nettoyées sont divisées en deux ensembles : un ensemble d'entraînement (80 %) pour ajuster les modèles et un ensemble de test (20 %) pour évaluer leur performance sur des observations inédites. La fonction *train_test_split* est utilisée avec un *random_state* afin de garantir la reproductibilité des résultats.

5.3. Stratégie de validation croisée (Cross-Validation)

Afin d'obtenir une estimation plus robuste et moins dépendante d'une unique partition des données, une validation croisée de type *k-fold* a été intégrée au processus de modélisation. Cette technique consiste à diviser l'ensemble d'entraînement en k sous-ensembles, à entraîner le modèle sur $k-1$ d'entre eux et à l'évaluer sur le sous-ensemble restant, en répétant l'opération pour chaque fold. La validation croisée permet ainsi de réduire le risque de surapprentissage, d'améliorer la stabilité des métriques et de comparer les modèles de manière plus rigoureuse. Les scores moyens obtenus sur les différents folds servent de base pour sélectionner les modèles les plus performants avant leur évaluation finale sur l'ensemble de test.

5.4. Modèles testés

- **Régression linéaire :** Modèle de base supposant une relation linéaire entre les variables explicatives et la cible. Il sert de référence pour évaluer les gains apportés par des modèles plus complexes.
- **Random Forest :** Modèle d'ensemble construit à partir d'arbres de décision. Il gère les relations non linéaires, est robuste aux valeurs aberrantes et limite le surapprentissage grâce au bootstrap (rééchantillonnage avec remise du jeu de données)
- **Random Forest optimisé :** Une recherche d'hyperparamètres via *RandomizedSearchCV*, couplée à la validation croisée, permet d'optimiser la profondeur des arbres, le nombre d'estimateurs et d'autres paramètres sensibles afin d'améliorer la performance et la stabilité du modèle.
- **Modèles restreints (sans certaines variables) :** Plusieurs variantes ont été entraînées afin d'évaluer l'importance réelle de certaines caractéristiques, notamment en supprimant des variables comme *Rooms* ou certaines catégories. Cela permet d'étudier la robustesse du modèle face à des architectures de données simplifiées.

5.5. Prétraitement des variables

- **StandardScaler :** Le *StandardScaler* est appliqué aux variables numériques afin d'harmoniser leur échelle et d'améliorer la stabilité de modèles sensibles comme la régression linéaire.
- **OneHotEncoder :** Les variables catégorielles (type de propriété, département) sont encodées sous forme binaire via le *OneHotEncoder*, permettant leur intégration efficace dans les modèles, notamment les approches non linéaires telles que la Random Forest.

6. Evaluation des modèles

6.1. Performance des modèles

Model	Target	MAE_train	RMSE_train	R2_train	MAE_test	RMSE_test	R2_test	R2_CV_mean	R2_CV_std
Linear Regression	log (Price)	0.2734	0.3829	0.6715	0.2721	0.3806	0.6821	0.6718	0.0158
Random Forest	log (Price)	0.1201	0.1999	0.9105	0.171	0.2749	0.8342	0.8296	0.0039
Random Forest without Rooms	log (Price)	0.1474	0.2369	0.8743	0.1825	0.2845	0.8224	0.8165	0.0035
Random Forest without Property	log (Price)	0.1201	0.1998	0.9105	0.1711	0.2753	0.8337	0.8294	0.0038
Random Forest without Outliers (2.5/97.5)	log (Price)	0.113	0.1855	0.8939	0.1619	0.257	0.7973	0.7972	0.004
Random Forest without Outliers (tuned) (tuned)	log (Price)	0.1308	0.205	0.8704	0.1582	0.2471	0.8126	0.8124	0.0034

Figure 14 - Tableau comparatif des métriques d'évaluation

Les métriques d'évaluation (R^2, MAE, RMSE) pour les ensembles d'entraînement et de test sont présentées ci-dessus. Elles montrent que la régression linéaire constitue un modèle inadéquat : le R^2 très faible et les erreurs élevées en grande partie liée à la violation des hypothèses de Gauss-Markov (linéarité, homoscédasticité, indépendance des résidus). L'écart limité entre les performances train et test confirme par ailleurs un sous-apprentissage (underfitting) : le modèle est trop simple pour décrire la variabilité réelle du marché locatif.

À l'inverse, la forêt aléatoire obtient un R^2 d'environ 0,83 sur le jeu de test, avec des erreurs nettement plus faibles. Bien que les performances sur le jeu d'entraînement soient très élevées ($R^2 \approx 0{,}91$), l'écart avec le R^2 test demeure modéré, indiquant un léger overfitting structurel, typique des modèles d'arbres, mais restant contenu. Ce comportement suggère que la Random Forest parvient à capturer la complexité des données tout en conservant une bonne capacité de généralisation.

En outre, la validation croisée corrobore cette stabilité : le *R2_CV_mean* avoisine également 0,83, tandis que l'écart-type (*R2_CV_std*) est extrêmement faible (≈ 0,004). Cette homogénéité entre folds montre que le modèle ne dépend pas d'une partition particulière du jeu d'apprentissage. Autrement dit, la variance du modèle est faible et l'overfitting bien maîtrisé.

Les modèles alternatifs, construits en excluant certaines variables comme *Rooms* ou *Property Type*, montrent que ces caractéristiques ne contribuent que marginalement à la qualité prédictive : leur retrait entraîne une baisse infime du R^2. À l'inverse, le département et la surface concentrent l'essentiel du signal prédictif, comme le confirme l'analyse d'importance des variables estimée par permutation sur le jeu de test, présentée dans la **figure 15**.

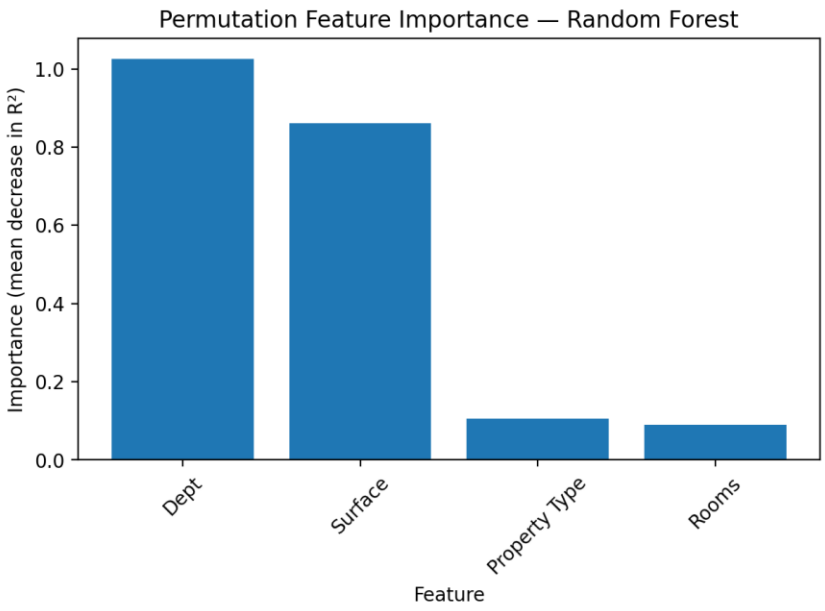

Figure 15 - Importance de features dans le benchmark (Random Forest)

6.2. Analyse des résidus

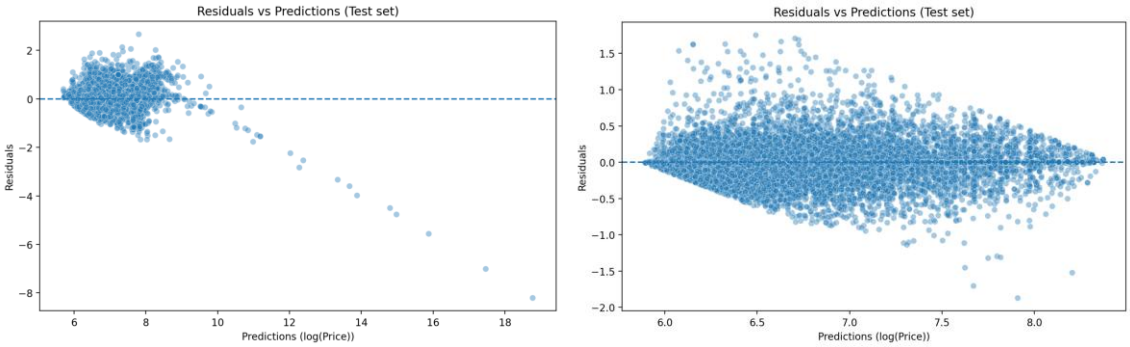

Figure 16 - Graphique de résidus pour la régression linéaire (à gauche) et forêt aléatoire (à droite)

L'analyse des résidus au travers de la **figure 16** présentée ci-dessus permet d'évaluer la validité des hypothèses sous-jacentes aux modèles. Pour la régression linéaire, on observe une forte hétéroscédasticité : la variance des résidus augmente avec les prédictions, formant une bande ouverte caractéristique. De plus, les résidus ne sont pas centrés autour de zéro, ce qui indique un biais et la violation de l'hypothèse $E[\varepsilon|X] = 0$ de Gauss-Markov. Ainsi, la régression linéaire ne constitue pas un modèle adéquat pour ce problème.

En revanche, le modèle de type Forêt Aléatoire présente une meilleure stabilité globale : les résidus demeurent majoritairement centrés autour de zéro et leur dispersion reste relativement constante sur l'ensemble des valeurs prédites, suggérant une hétéroscédasticité limitée. On observe toutefois un biais conditionnel, caractérisé par des résidus plutôt positifs pour les loyers faibles et plutôt négatifs pour les loyers élevés, traduisant une légère compression des valeurs extrêmes. Ce biais a un impact réel sur les prédictions individuelles, le modèle ayant tendance à légèrement surestimer les loyers les plus faibles et à sous-estimer les loyers les plus élevés, ce qui peut être problématique lorsque l'analyse porte spécifiquement sur les valeurs extrêmes. Malgré ce phénomène, l'erreur moyenne reste globalement faible, et la Forêt Aléatoire parvient à capturer efficacement les relations complexes entre les variables explicatives, produisant des erreurs plus stables et moins biaisées qu'un modèle linéaire.

6.3. Analyse du comportement non linéaire du modèle

Afin d'interpréter correctement certaines prédictions inhabituelles du modèle, il convient de rappeler que la Forêt Aléatoire n'apprend pas une relation globale entre les variables et le loyer. Contrairement à la régression linéaire, qui impose des effets monotones et homogènes, la Forêt Aléatoire reproduit des structures locales présentes dans les données, même lorsque celles-ci proviennent d'annonces atypiques ou mal classées.

Cette propriété apparaît clairement dans l'analyse d'un logement hypothétique situé à Paris (75), de 45 m², déclaré comme *studio*, pour lequel seul le nombre de pièces est modifié.

Dept	Rooms (pièces)	Surface (m²)	Property Type	Predicted Rent (€)
75	1	45	studio	2414
75	2	45	studio	1619
75	3	45	studio	1432

Figure 17 - Prédiction pour le loyer d'un immeuble à Paris (75)

Le modèle prédit environ 2414 € pour 1 pièce, 1619 € pour 2 pièces, et 1432 € pour 3 pièces. La hiérarchie observée peut sembler contre-intuitive, notamment l'effet inverse de l'augmentation du nombre de pièces, entraînant une prédiction moins élevée.

Cette dynamique s'explique directement par la structure du dataset. Dans les données collectées, un bien de 45 m² – 1 pièce ne correspond presque jamais à un studio classique mais plutôt à un logement atypique ou de standing (loft, atelier, espace ouvert), caractérisé par des loyers nettement plus élevés. À l'inverse, la configuration 45 m² – 2 pièces correspond au T2 parisien standard, segment très fréquent et donc statistiquement plus modéré en termes de prix.

La configuration 45 m² – 3 pièces est extrêmement rare et renvoie généralement à des biens anciens ou mal distribués. Cette rareté entraîne une forte variabilité des loyers : certaines annonces sont très bon marché, d'autres légèrement au-dessus du T2 selon leur état ou leur localisation précise. Le modèle apprend donc cette dispersion et produit une estimation légèrement inférieure au T2, tout en restant très éloignée du segment valorisé des grands 1 pièce.

Ainsi, ces prédictions ne reflètent pas une incohérence du modèle mais la réalité complexe du marché locatif et la présence de segments atypiques. Cette analyse confirme l'intérêt d'un modèle non linéaire pour capturer des comportements impossibles à représenter avec une régression linéaire.

6.4. Choix du modèle final

Le choix du modèle final repose sur la stabilité, la précision et l'analyse des résidus. La régression linéaire est exclue en raison de son biais, de sa forte hétéroscédasticité et de ses faibles performances en termes de R^2, MAE et RMSE (**figure 14 – chapitre 6.1**). La forêt aléatoire (benchmark) constitue déjà un modèle robuste et performant, mais la version optimisée offre un meilleur compromis entre précision et homogénéité. Bien que le R^2 diminue légèrement, cette baisse s'explique par une moindre sensibilité aux valeurs extrêmes, ce qui améliore la cohérence globale du modèle. Le modèle retenu pour une utilisation finale et un éventuel déploiement est donc la Forêt Aléatoire optimisée (tuned) entraînée après suppression des outliers.

6.5. Analyse de l'interprétabilité du modèle (SHAP)

Les performances élevées de la Forêt Aléatoire optimisée (tuned) sans outliers confirment sa capacité à modéliser la complexité du marché locatif français, bien que sa nature non linéaire rende son interprétation directe délicate. Afin de comprendre les mécanismes décisionnels du modèle et le rôle des variables explicatives, une analyse d'interprétabilité est menée à l'aide des valeurs SHAP, qui permettent de décomposer chaque prédiction en une valeur de référence correspondant à la prédiction moyenne du modèle et en contributions locales associées à chaque variable, mesurant leur effet marginal à la hausse ou à la baisse sur la prédiction finale.

$$Prédiction = Baseline + \sum SHAP$$

La **figure 18** ci-dessous présente un graphique de synthèse (beeswarm) des valeurs SHAP pour les principales variables du modèle.

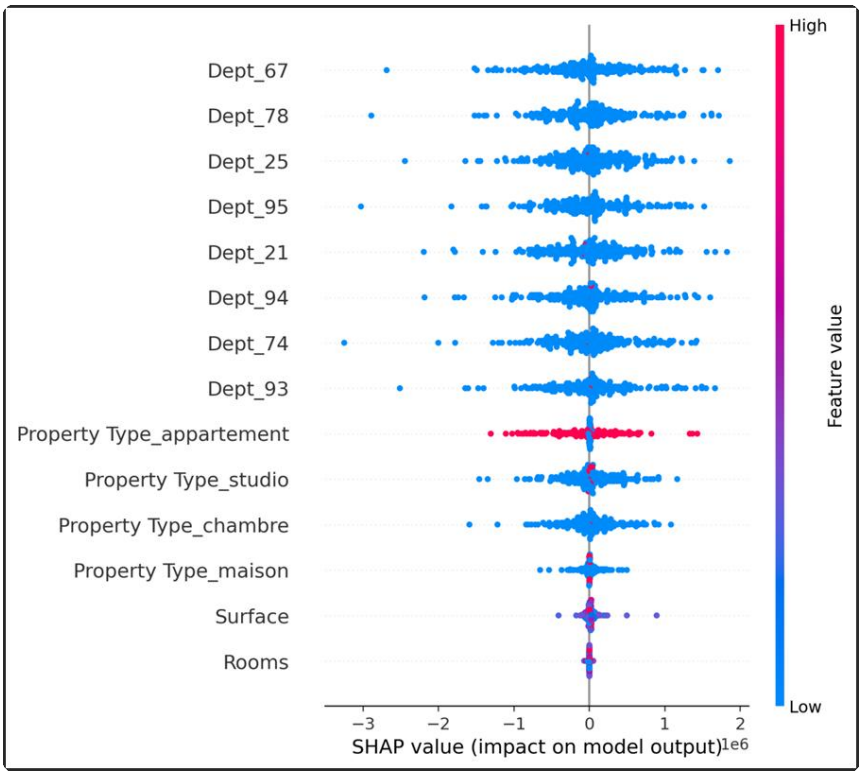

Figure 18 - Graphique des valeurs SHAP

Ce type de visualisation permet d'analyser simultanément l'importance globale des variables, mesurée par la magnitude moyenne des valeurs SHAP, ainsi que la variabilité de leurs effets locaux selon les observations.

L'analyse met en évidence que certains départements — notamment les départements 67, 78, 25, 95, 21, 94, 74 et 93 — présentent des valeurs SHAP de forte magnitude. Cela indique que la localisation dans ces départements conduit le modèle à appliquer des corrections particulièrement importantes à la prédiction de référence, à la hausse comme à la baisse, en fonction des autres caractéristiques du logement telles que la surface, le type de propriété et le nombre de pièces. Ces départements correspondent à des zones où le modèle est confronté à des configurations de biens moins fréquentes ou atypiques au regard de la structure globale du jeu de données. La forte amplitude des valeurs SHAP traduit ainsi une hétérogénéité marquée des marchés locatifs concernés, caractérisés par une grande diversité de situations locales et de niveaux de loyers pour des biens pourtant comparables sur le plan structurel.

Cette observation est cohérente avec les résultats de l'analyse exploratoire, qui a mis en évidence une forte dispersion des loyers et la présence de segments non standards dans certaines zones géographiques. Les variables départementales jouent ainsi un rôle central dans la discrimination locale des biens et constituent le principal levier d'ajustement fin des prédictions individuelles du modèle.

À l'inverse, la variable *Surface* se distingue par une importance globale élevée dans l'apprentissage du modèle (**figure 15 – Chapitre 6.1**), tout en présentant des valeurs SHAP relativement faibles et peu dispersées. Ce comportement indique que la surface constitue une variable structurelle essentielle à la formation du niveau général des loyers, indispensable à la performance globale du modèle. Cependant, son effet local sur la prédiction demeure stable et homogène, induisant des ajustements progressifs et prévisibles par rapport à la prédiction de référence, sans générer de corrections extrêmes. Ainsi, tandis que la surface structure globalement le prix des logements, elle contribue peu à la variabilité fine des prédictions individuelles.

Enfin, les variables liées au nombre de pièces présentent des valeurs SHAP globalement faibles, confirmant leur influence marginale une fois la surface prise en compte. Ce résultat est cohérent avec l'analyse exploratoire, qui montrait que le nombre de pièces n'apporte qu'une information limitée sur le prix au mètre carré lorsque la superficie du logement est déjà connue.

Au-delà de son rôle explicatif, l'analyse SHAP ouvre également des perspectives méthodologiques pour l'amélioration du modèle. L'identification de départements associés à des corrections locales extrêmes suggère l'existence de segments géographiques atypiques susceptibles de justifier des stratégies de modélisation complémentaires, telles que des regroupements exploratoires de départements rares ou hétérogènes. De telles approches pourraient permettre d'évaluer si un traitement différencié de ces segments améliore la stabilité ou la précision des prédictions, tout en conservant la robustesse du modèle global.

7. Limites du modèle, pistes d'amélioration et déploiement

7.1. Limitations du modèle et du dataset

Le modèle conçu dans ce travail détient notamment des limitations concernant la source de données et les limites du propre modèle. Au préalable, il est essentiel de souligner le bruit acquis en conséquence des annonces mal remplies, des biens atypiques et des outliers présents dans l'échantillon. Ces irrégularités, inhérentes au web scraping, introduisent une variabilité non contrôlée qui peut influencer la manière dont le modèle apprend certaines relations. De plus, certaines variables descriptives importantes pour l'estimation du loyer — telles que l'étage, la présence d'un ascenseur, l'année de construction ou l'état général du logement — ne sont pas disponibles dans le dataset, ce qui limite la capacité prédictive globale du modèle.

Un autre point notable concerne l'interprétation des pièces : la variable « Rooms » ne distingue pas entre pièces de vie et chambres, ce qui peut conduire à des ambiguïtés, notamment pour les logements atypiques où la classification ne suit pas les conventions du marché français. Enfin, la Forêt Aléatoire, bien que robuste, reste sensible à la structure locale des données et ne généralise pas toujours bien pour des configurations rares ou absentes du dataset. Ces limitations rappellent que les performances du modèle dépendent étroitement de la qualité et de la représentativité des données utilisées.

7.2. Améliorations futures

Par rapport aux améliorations futures, on peut souligner plusieurs pistes concrètes. Tout d'abord, l'implémentation d'un webscraping asynchrone, doté de mécanismes de protection pour gérer les erreurs (timeouts, pages inaccessibles, changements de structure HTML), permettrait d'augmenter le volume de données collectées tout en améliorant la robustesse du pipeline. Il serait également pertinent de diversifier les sources en intégrant plusieurs sites d'annonces, afin d'enrichir la base avec de nouvelles variables explicatives, telles que l'arrondissement précis, l'étage, la présence d'un ascenseur, l'année de construction, l'état général du bien, le caractère meublé ou non, ainsi que la proximité des transports et des services. Ces informations supplémentaires pourraient fortement améliorer la capacité du modèle à capturer les différences de standing entre logements a priori comparables en termes de surface et de nombre de pièces.

En parallèle, un nettoyage avancé des données pourrait être mis en place pour traiter plus finement les outliers et les incohérences, notamment en détectant les annonces dont la typologie (nombre de pièces) est manifestement incompatible avec la surface ou le niveau de loyer.

Enfin, il serait intéressant d'explorer des modèles alternatifs, tels que des méthodes de gradient boosting (XGBoost, LightGBM) afin de comparer leurs performances et leur interprétabilité à celles de la Forêt Aléatoire. Ces axes d'amélioration ouvrent la voie à un système plus robuste, plus précis et mieux adapté à la complexité du marché locatif français.

7.3. Perspectives de déploiement et MLOps

Dans la perspective d'un déploiement réel du modèle, plusieurs considérations techniques doivent être prises en compte afin d'assurer une utilisation fiable, réplicable et évolutive. Le modèle actuel, ainsi que son pipeline de prétraitement, pourraient être exportés au format sérialisé (par exemple avec joblib) et intégrés dans une API légère développée avec FastAPI ou Flask. Une telle API permettrait de recevoir en entrée les caractéristiques d'un logement, d'appliquer automatiquement les transformations nécessaires, puis de renvoyer une estimation du loyer. Ce type d'architecture rend possible l'intégration du modèle au sein d'une application web ou d'un système interne d'aide à la décision.

Dans une optique plus avancée, relevant des pratiques modernes de MLOps, il serait pertinent de mettre en place un système de suivi continu du modèle. Cela inclurait le versionnement des données et du modèle, la surveillance de ses performances dans le temps (détection de dérive ou "data drift"), ainsi que l'automatisation de certaines étapes du cycle de vie, comme le réentraînement périodique à partir de données nouvelles. Des outils tels que MLflow, DVC ou Prefect pourraient faciliter la traçabilité et la reproductibilité des expérimentations, tandis qu'une containerisation via Docker garantirait un environnement d'exécution stable, indépendamment de la plateforme cible.

Enfin, un déploiement à plus grande échelle pourrait reposer sur des services cloud permettant l'orchestration, la montée en charge et la gestion d'accès sécurisés. Ces bonnes pratiques de MLOps assureraient que le modèle reste non seulement performant au moment de sa création, mais également durable et fiable dans le temps, même face à l'évolution rapide du marché locatif et des sources de données.

8. Conclusion

Ce projet a permis de développer une chaîne complète de prédiction des loyers, depuis la collecte automatisée des données jusqu'à l'entraînement et l'évaluation de modèles de régression. La structuration du pipeline, fondée sur une architecture modulaire, a rendu possible l'intégration fluide du nettoyage, de la préparation et de la modélisation, tout en assurant la reproductibilité de l'ensemble du processus. L'analyse exploratoire et l'usage de la transformation logarithmique ont mis en évidence les caractéristiques spécifiques du marché locatif, marqué par des valeurs extrêmes et des distributions hétérogènes.

La comparaison des approches a montré les limites des modèles linéaires pour capter les dynamiques complexes du marché, tandis que la Forêt Aléatoire optimisée s'est distinguée par sa capacité à modéliser des relations non linéaires et à reproduire fidèlement la structure réelle des données. Certaines prédictions atypiques, notamment sur des biens de 45 m² selon la typologie déclarée, illustrent l'importance de disposer de données riches et cohérentes, mais confirment également la faculté du modèle à apprendre les comportements effectivement présents dans le dataset.

Au-delà de la performance prédictive, ce travail ouvre des perspectives claires pour une industrialisation du pipeline. L'intégration d'outils tels que DVC pour le versionnement des données, MLflow pour le suivi des expériences, Docker pour la reproductibilité des environnements et Prefect pour l'orchestration automatisée permettrait de transformer ce prototype en un système pleinement opérationnel. De même, l'enrichissement du dataset à partir de sources multiples, ainsi que l'exploration de modèles de type gradient boosting, constitueraient des pistes prometteuses pour améliorer la précision et la robustesse du système.

9. Bibliographie

- Rosen, S. (1974). *Hedonic prices and implicit markets: Product differentiation in pure competition*. *Journal of Political Economy*, 82(1), 34–55.
- Breiman, L. (2001). *Random forests*. *Machine Learning*, 45(1), 5–32.
- Mullainathan, S., & Spiess, J. (2017). *Machine learning: An applied econometric approach*. *Journal of Economic Perspectives*, 31(2), 87–106.
- Athey, S., & Imbens, G. (2019). *Machine learning methods that economists should know about*. *Annual Review of Economics*, 11, 685–725.
- Lundberg, S. M., & Lee, S.-I. (2017). *A unified approach to interpreting model predictions*. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.