

MAI 2024

Kevin LE GRAND

# PROJET CHEF D'OEUVRE

## Estimation de biens immobiliers

Développeur en intelligence artificielle

Certification RNCP34757

# SIMPLON.CO

**ISEN**

ALL IS DIGITAL!

OUEST



**Microsoft**

*En adressant ce document à l'enseignante, je certifie que ce travail est le mien et que j'ai pris connaissance des règles relatives au référencement, au plagiat, ainsi qu'à l'usage d'une intelligence artificielle d'aide à la rédaction de type ChatGPT*

## ***Liens vers les différentes ressources et applications produites pour ce projet***

Lien Github :

[https://github.com/rastakoer/certif\\_app\\_immo/](https://github.com/rastakoer/certif_app_immo/)

Lien de l'application :

<https://immoappkevleg2-d1a29ee73317.herokuapp.com/>

Lien de l'API :

<https://apiimmoappkevleg-7337fa262339.herokuapp.com/>

Lien de MLflow :

<https://mlflowimmoappkevleg-737621d410d0.herokuapp.com/>

Lien Grafana :

<https://kevinlegrand.grafana.net/public-dashboards/cde8ec56de054eb295d3f68e0039aa63>

# SOMMAIRE

<b>1. Introduction</b>	<b>5</b>
a. Contexte	5
b. Objectifs	6
c. Approche de la solution	7
d. Technologies utilisées	7
e. Schéma de l'application	8
<b>2. Récupération, nettoyage et stockage des données</b>	<b>9</b>
a. Web scraping et api	9
b. Nettoyage	10
c. Schéma de la base de données	11
d. Création et remplissage de la base de données	12
<b>3. EDA</b>	<b>13</b>
a. Répartition et visualisation des données	13
i. Étendue et nombre de ventes	13
ii. Répartition des prix en fonction du type de bien	14
iii. Visualisation de la répartitions des prix en fonction de la localisation	15
iv. Visualisation de l'évolution des prix en fonction de sa surface habitable	15
v. Visualisation de l'évolution des prix en fonction du nombre de pièces	16
vi. Evolution des prix dans le temps	17
b. Visualisation et traitements des valeurs aberrantes	18
i. Étude sur les prix élevés	18
ii. Étude sur les prix bas	19
iii. Traitements des valeurs aberrantes	19
c. Corrélations entre les variables	19
d. Conclusion EDA	21
<b>4. Veille technique</b>	<b>22</b>
a. Introduction	22
i. Définition de la régression linéaire (tiré du site Kobia.fr)	22
ii. La fonction coût ou perte (loss function)	22
iii. Les algorithmes de minimisation de l'erreur	23
iv. Métrique de performances	24
b. Les différents modèles d'intelligence artificielle disponibles	26
i. RandomForestRegressor	26
ii. XGBoost	29
iii. Réseau de neurones	30
c. Conclusion	31

---

<b>5. Les modèles</b>	<b>32</b>
a. Choix des modèles et méthodologie	32
b. Entraînement des modèles	32
i. Récupération des données	33
ii. Nettoyage de données	34
iii. Splittage des données	35
iv. Préparation des données	35
v. Entraînement d'un modèle	36
c. Comparaisons des modèles	37
<b>6. Mise en place de l'environnement</b>	<b>40</b>
a. Outils et utilisation pour l'intégration et déploiement continu (CI/CD)	40
i. Création du fichier test	40
ii. Configuration dans GitHub	41
iii. Configuration dans Heroku et fonctionnement	41
iv. Confirmation de l'intégration et du déploiement	43
b. Front-end	43
c. Back-end	45
i. Gestion des utilisateurs	45
ii.API	46
d. Monitoring	46
<b>7. Conclusion</b>	<b>48</b>
a. La réalisation du projet	48
b. Les difficultés rencontrées	48
c. Les améliorations et perspectives d'évolution	49
<b>8. Annexes</b>	<b>50</b>
<b>9. Glossaire</b>	<b>53</b>
<b>10. Index des graphiques</b>	<b>55</b>
<b>11. Bibliographie</b>	<b>57</b>
<b>12. Citations</b>	<b>58</b>

# 1. Introduction

## a. Contexte

Ce projet chef d'œuvre s'articule autour d'une application pouvant permettre l'estimation d'un bien immobilier. Il a été réalisé en dehors d'une alternance en entreprise et pour les besoins de validations de compétences les entreprises citées seront fictives.

Pour vous permettre de comprendre le contexte du projet, je vous propose de commencer par lire le premier entretien qui s'est déroulé entre l'entreprise FNAIM (Fédération Nationale de l'Immobilier) et Normand'IA (entreprise spécialisée dans le déploiement d'applications contenant de l'intelligence artificielle). Les deux autres comptes rendus suivants sont disponibles en annexe page 51 et 52.

*Caen le 27 Novembre 2023,*

*Compte rendu du premier échange : **Démarrage du projet***

*Parties prenantes :*

- La société demandeuse FNAIM représentée par Caroline Kern Richard
- La société prestataire Normand'IA représentée par Kevin LE GRAND

**Objectifs :**

- Permettre de faire le point avec ce qui est attendu par la FNAIM, de définir les besoins et d'établir une feuille de route entre les deux parties.

**Les attentes :**

- Réaliser une application Web intuitive permettant de faire une prédition sur le prix de vente d'une maison ou d'un appartement partout en France, de voir sur une carte les biens qui ont été vendus ainsi que les statistiques et l'évolution des ventes.
- L'application devra permettre de suivre l'évolution du nombre de prédictions ainsi qu'un processus d'alerting en cas de dysfonctionnement.

**Questions/réponses :**

- Q.Normand'IA : Disposez-vous de données historiques des ventes ?  
R.FNAIM : Non nous ne disposons pas de bases de données mais vous pouvez trouver les données de ventes sur le site data.gouv.fr
- Q.FNAIM : La livraison de l'application pourrait-elle avoir lieu avant avant les J.O 2024
- R.Normand'IA : Je vous propose de vous rencontrer régulièrement pour vous informer de l'avancement du projet et ainsi apporter des correctifs si besoin.

**Prochaines étapes :**

- Une première livraison de l'application est attendue pour le 15 Janvier avec un rendez-vous convenu entre les deux parties à 14H dans les bureaux de Normand'IA.
- Cette première étape devra rendre compte d'une application fonctionnelle sans pour autant avoir de bonnes performances aux niveaux des prédictions.

LE GRAND Kevin

Caroline Kern Richard

X

X

## b. Objectifs

Suite à ce premier échange, l'équipe de Normand'IA s'est rassemblé pour élaborer les différentes étapes du projet.

Un planning a été établi avec un diagramme de Gantt (figure 1).

Un tableau Kanban (figure 2) a été réalisé pour avoir une visualisation directe des étapes en cours, terminées, à venir ou bloquées.

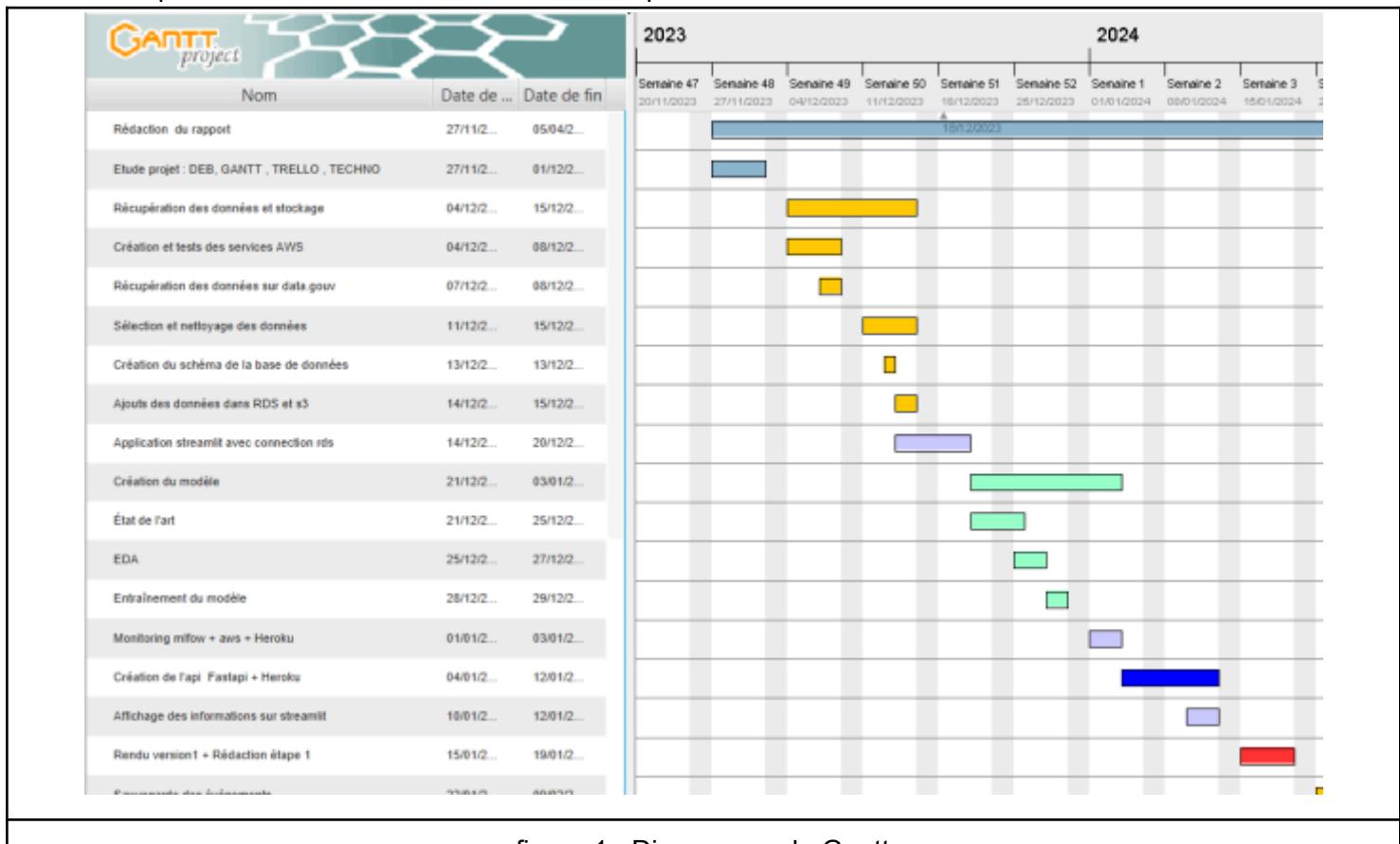


figure 1 : Diagramme de Gantt



Figure 2 : Kanban réalisé sur Trello

### c. Approche de la solution

Les données seront récupérées sur le site data.gouv avec deux méthodes. La première sera avec du web scraping pour récupérer et extraire les fichiers csv<sup>1</sup> contenant les ventes réalisées. La deuxième méthode utilisera l'API<sup>2</sup> data.gouv pour récupérer les informations sur les départements et régions.

L'application sera déployée et intégrée en continu à l'aide de GitHub et Heroku, elle utilisera également les services d'AWS pour la gestion de bases de données et de stockage.

L'objectif principal de ce projet étant l'estimation du prix d'un bien immobilier nous utiliserons, dans un premier temps un modèle de régression linéaire simple afin d'atteindre l'étape 1 dans les délais et de se concentrer sur tout l'environnement à mettre en place. Par la suite un état de l'art sera réalisé afin de choisir un modèle performant en prenant le temps de trouver les meilleurs paramètres.

### d. Technologies utilisées



- Le développement de l'application sera effectué en python.
- L'application sera déployée et intégrée en continu sur Heroku à l'aide de github et Docker. Elle sera conçue avec le Framework python Streamlit.
- Le modèle sera créé avec scikit-learn et/ou TensorFlow.
- L'api sera créée et documentée avec FastAPI.
- Le monitoring se fera à l'aide de MLFlow pour le modèle et de Grafana pour l'application.
- Les services AWS sont :
  - IAM : pour la gestion des accès.
  - Gestion de la facturation et des coûts.
  - RDS : pour les bases de données MySQL pour les ventes et Postgre pour MLFlow.
  - s3 : pour le stockage.
  - Lambda et EventBridge : pour l'automatisation d'un code à intervalle régulier ou lors du déclenchement d'un événement.

<sup>1</sup> CSV (Comma Separated Values) est un format de fichier texte utilisé pour stocker des données tabulaires

<sup>2</sup> API (Application Programming Interface) est une Interface de Programmation Applicative, est un ensemble de règles, de protocoles et de définitions qui permettent à différents logiciels et systèmes de communiquer entre eux.

### e. Schéma de l'application

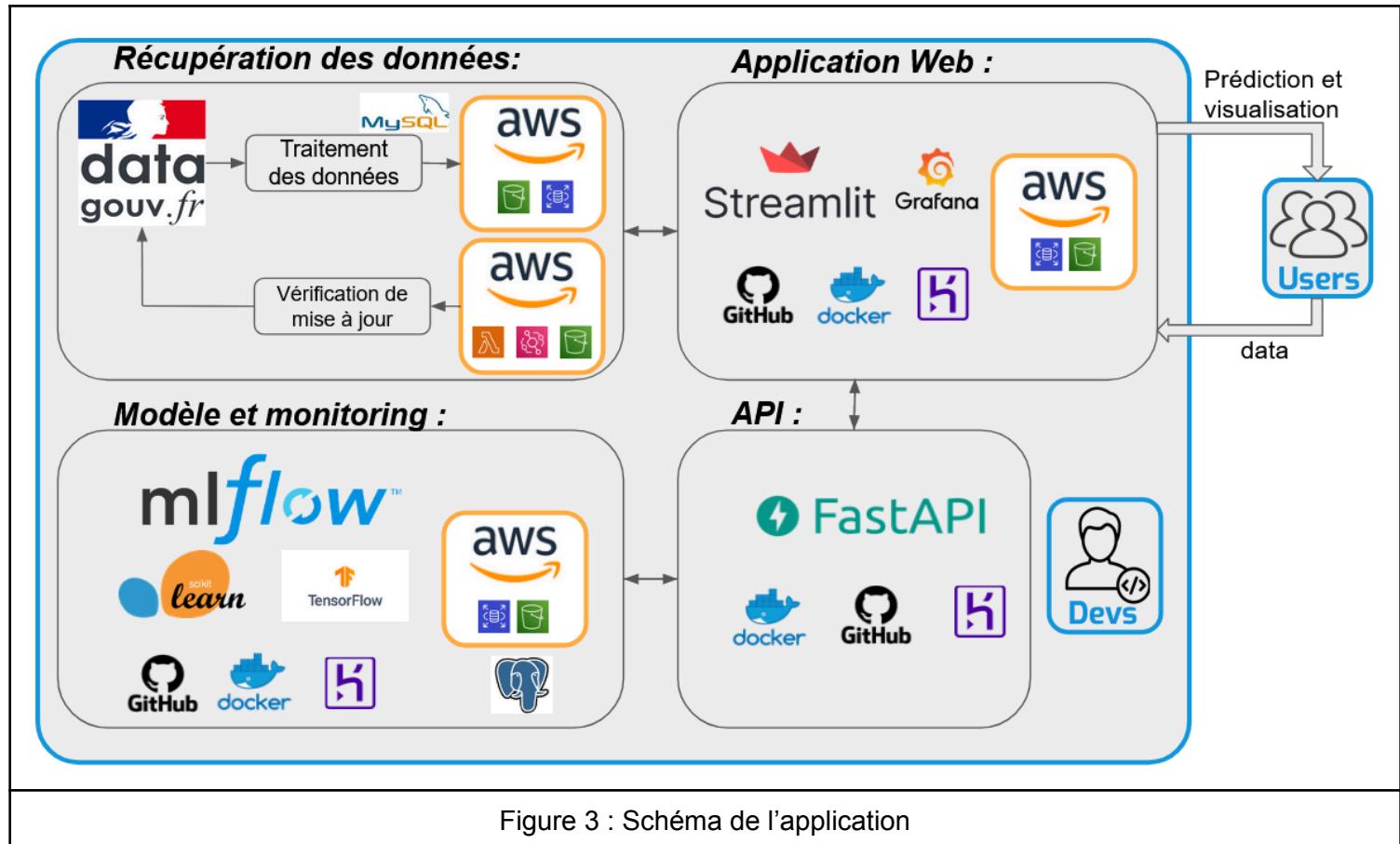


Figure 3 : Schéma de l'application

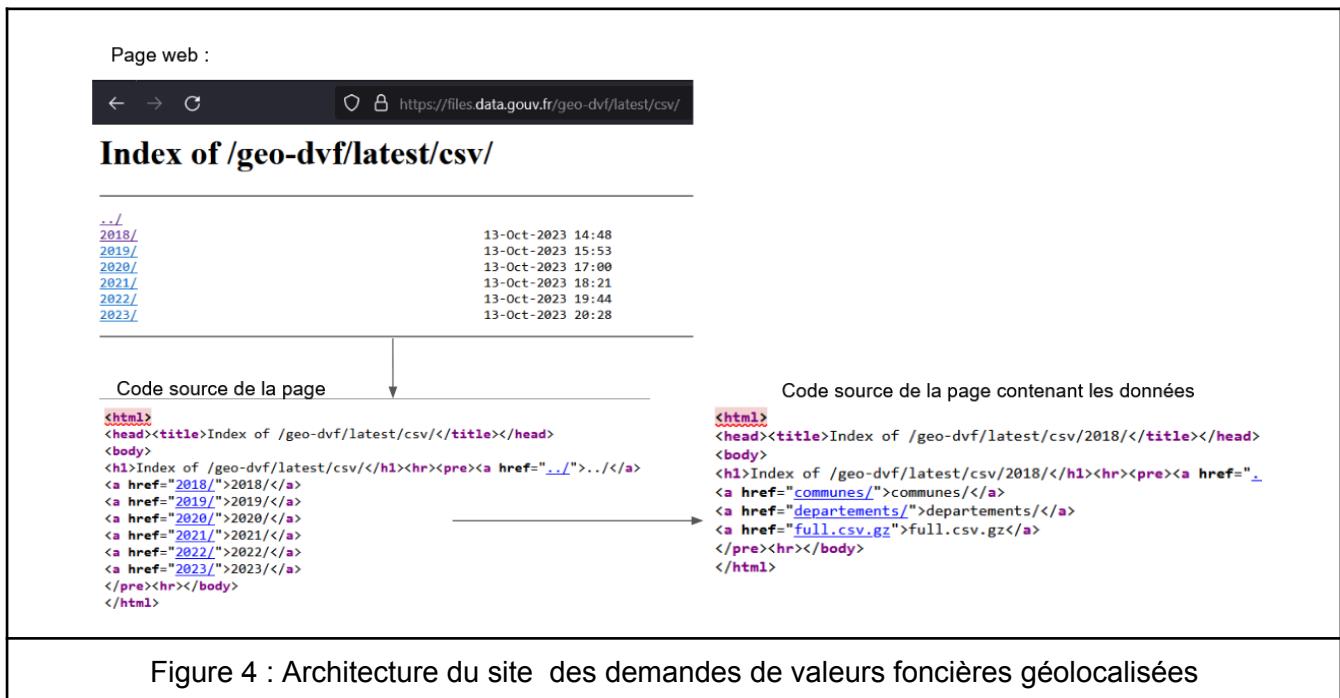
Le schéma montre bien les quatres blocs distincts de l'application. Le dépôt GitHub<sup>3</sup> aura donc 4 branches avec chacune un environnement indépendant permettant une intégration et un déploiement continu indépendant des uns des autres. Un schéma fonctionnel plus détaillé est disponible en annexe figure 46 .

<sup>3</sup> **GitHub** est une plateforme de développement collaboratif de logiciels basée sur Git. Elle permet aux développeurs de travailler ensemble sur des projets, de gérer des versions, de suivre les modifications, et de coordonner le travail en équipe. Mon dépôt : [https://github.com/rastakoer/certif\\_app\\_immo/](https://github.com/rastakoer/certif_app_immo/)

## 2. Récupération, nettoyage et stockage des données

### a. *Web scraping et api*

Pour récupérer les données nécessaires au projet, l'information donnée par la FNAIM est d'utiliser le site du gouvernement data.gouv. Le choix des données s'est donc porté sur les "Demandes de valeurs foncières géolocalisées"<sup>4</sup>. Le site fournit une description succincte des variables composant le dataset et son architecture simplifiée permet de récupérer facilement les fichiers avec du web scraping<sup>5</sup> comme le montre l'image figure 4:



L'utilisation de la bibliothèque BeautifulSoup est donc idéale dans la récupération de données. Afin de rendre ces données exploitables et ne pas prendre trop de place sur l'ordinateur, les classes BytesIO, GzipFile et DataFrame, respectivement issues des modules io, gzip et pandas de python ont été utilisées pour décompresser les fichiers et transformer les données en un dataframe pandas.

Il y a malheureusement des données manquantes dans ces données, les informations sur les départements ou les régions. Le gouvernement met à disposition un service d'APIs gratuites dans lesquelles il existe l'api Géo<sup>6</sup>. L'utilisation de cette api permet de faire correspondre le nom des départements et régions avec le code des communes présentes dans les données scrapées.

<sup>4</sup> Lien du site :<https://www.data.gouv.fr/fr/datasets/demandes-de-valeurs-foncieres-geolocalisees/>

<sup>5</sup> **webscraping** : Il s'agit d'une technique informatique utilisée pour extraire automatiquement des données d'un site web.

<sup>6</sup> Lien de l'api : <https://geo.api.gouv.fr/>

## b. Nettoyage

Une fois les données récupérées en local sur l'ordinateur, un premier nettoyage était nécessaire avant le stockage en base de données afin de conserver uniquement les données utiles au projet. Voici ci-dessous, un tableau des données disponibles :

Variables	Explications
<ul style="list-style-type: none"> <li>• id_mutation : Identifiant de mutation (non stable, sert à grouper les lignes)</li> <li>• date_mutation : Date de la mutation au format ISO-8601 (YYYY-MM-DD)</li> <li>• numero_disposition : Numéro de disposition</li> <li>• nature_mutation : Nature de la mutation</li> <li>• valeur_fonciere : Valeur foncière ( séparateur décimal = point)</li> <li>• adresse_numero : Numéro de l'adresse</li> <li>• adresse_suffixe : Suffixe du numéro de l'adresse (B, T, Q)</li> <li>• adresse_code_voie : Code FANTOIR de la voie (4 caractères)</li> <li>• adresse_nom_voie : Nom de la voie de l'adresse</li> <li>• code_postal : Code postal (5 caractères)</li> <li>• code_commune : Code commune INSEE (5 caractères)</li> <li>• nom_commune : Nom de la commune (accentué)</li> <li>• ancien_code_commune : Ancien code commune INSEE (si différent lors de la mutation)</li> <li>• ancien_nom_commune : Ancien nom de la commune (si différent lors de la mutation)</li> <li>• code_departement : Code département INSEE (2 ou 3 caractères)</li> <li>• id_parcelle : Identifiant de parcelle (14 caractères)</li> <li>• ancien_id_parcelle : Ancien identifiant de parcelle (si différent lors de la mutation)</li> <li>• numero_volume : Numéro de volume</li> <li>• lot_1_numero : Numéro du lot 1</li> <li>• lot_1_surface_carrez : Surface Carrez du lot 1</li> <li>• lot_2_numero : Numéro du lot 2</li> <li>• lot_2_surface_carrez : Surface Carrez du lot 2</li> <li>• lot_3_numero : Numéro du lot 3</li> <li>• lot_3_surface_carrez : Surface Carrez du lot 3</li> <li>• lot_4_numero : Numéro du lot 4</li> <li>• lot_4_surface_carrez : Surface Carrez du lot 4</li> <li>• lot_5_numero : Numéro du lot 5</li> <li>• lot_5_surface_carrez : Surface Carrez du lot 5</li> <li>• nombre_lots : Nombre de lots</li> <li>• code_type_local : Code de type de local</li> <li>• type_local : Libellé du type de local</li> <li>• surface_reelle_bati : Surface réelle du bâti</li> <li>• nombre_pieces_principales : Nombre de pièces principales</li> <li>• code_nature_culture : Code de nature de culture</li> <li>• nature_culture : Libellé de nature de culture</li> <li>• code_nature_culture_spéciale : Code de nature de culture spéciale</li> <li>• nature_culture_spéciale : Libellé de nature de culture spéciale</li> <li>• surface_terrain : Surface du terrain</li> <li>• longitude : Longitude du centre de la parcelle concernée (WGS-84)</li> <li>• latitude : Latitude du centre de la parcelle concernée (WGS-84)</li> </ul>	<p><u>Données conservées :</u></p> <ul style="list-style-type: none"> <li>- L'id_mutation représentant une vente unique</li> <li>- La date de vente</li> <li>- Toutes les données concernant la position GPS et l'adresse</li> <li>- Les informations sur la surface du bâti, du terrain et le nombres de pièces</li> <li>- Le type de local (Maison, appartement..)</li> <li>- La nature de la mutation (Vente, adjudication..)</li> <li>- La valeur foncière qui est le montant de la vente (la target du futur modèle)</li> </ul> <p><u>Données inutiles :</u></p> <ul style="list-style-type: none"> <li>- les informations sur les lots et/ou parcelles composants une vente</li> <li>- les informations sur les cultures</li> <li>- les anciens codes et noms de communes</li> </ul> <p><u>Remarques :</u></p> <ul style="list-style-type: none"> <li>- Un id_mutation peut apparaître sur plusieurs lignes car une vente est décomposée en lots ou parcelles.</li> <li>- Pour un même id_mutation il peut y avoir plusieurs type de local sur un lot différent, exemple (maison et dépendances)</li> </ul>

Après cette première analyse, la première étape était de récupérer toutes les colonnes entourées en vert sur le tableau ci-dessus où les id\_mutations avait pour nature de mutation une valeur "Vente", une variable est aussi créée pour indiquer si le bien posséde une dépendance ou non.

Par la suite plusieurs fonctions ont été créées afin d'insérer des données pertinentes et cohérentes en base de données:

- La gestion des valeurs manquantes avec des stratégies différentes suivant les variables (pour les valeurs foncières : suppression de ligne, numéro de rue : 0 ...)
- Un data processing afin d'adapter les données au bon format dans notre base de données (exemple date\_mutation au format date, surface\_bâti en integer...)
- Le groupement des données par id\_mutation en utilisant différentes fonction d'aggrégations suivants les variables (exemple longitude : mean, surface\_terrain : sum)

### c. Schéma de la base de données

Le schéma de la base de données a été créé avec JMerise<sup>7</sup>. JMerise est un outil dédié à la modélisation des modèles conceptuels de données (MCD). Il permet les relations réflexives (cardinalités), la généralisation et la spécialisation des entités. Il génère le modèle logique de données (MLD) et le script Mysql.

Pour générer le MCD il faut suivre les étapes suivantes :

#### 1. Créations des entités

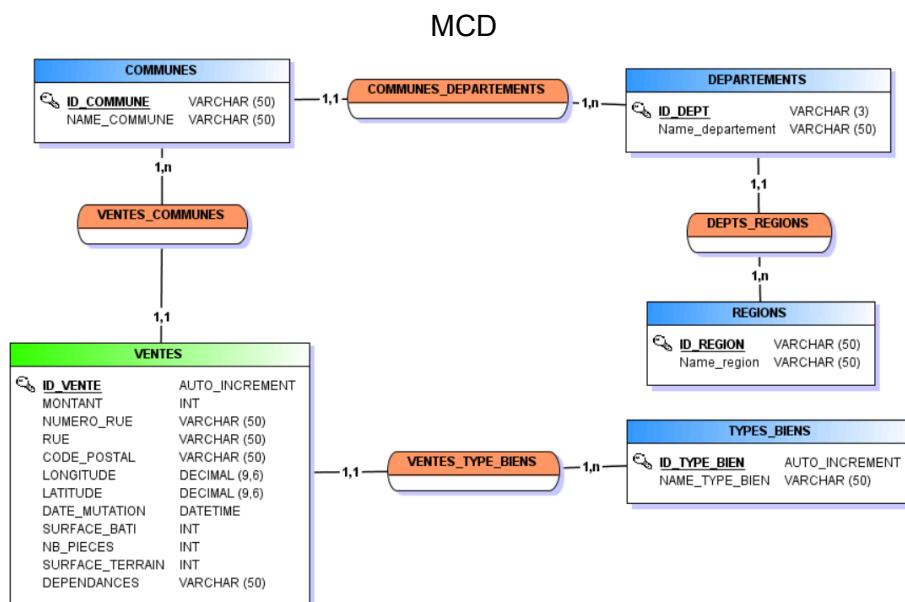
- VENTES : avec la description du bien (id\_mutation, l'adresse du bien, la surface du bâti et du terrain si le bien est une maison, le nombre de pièces, sa position GPS et le montant de la vente)
- COMMUNES , DEPARTEMENTS, REGIONS : avec un identifiant unique pour chaque spécialisation ainsi que leur dénomination
- TYPES\_BIENS : contenant un identifiant et la dénomination du type de bien vendu

#### 2. Définition des variables : Pour chaque entité il faut définir les variables ainsi que leur type. Exemple pour l'entité COMMUNES nous avons deux variables, ID\_COMMUNES avec le type VARCHAR (ici ce n'est pas un auto increment puisque l'identifiant est normé par l'API) et la variable NAME\_COMMUNE représentant le nom de la commune en type VARCHAR(50)

#### 3. Assignation des cardinalités, exemple :

- Une vente ne pouvait appartenir qu'à un type de bien(1,1) et un type de bien pouvait appartenir à plusieurs ventes(1,n).
- Une région peut avoir plusieurs départements(1,n) tandis qu'un département n'appartient qu'à une seule région(1,1)

La modélisation de notre modèle conceptuel de données (MCD) et représenté par la figure ci-dessous:



<sup>7</sup> Lien JMerise : <https://www.jfreesoft.com/JMerise/>

La force de JMerise est qu'une fois le modèle conceptuel de données créé, on va pouvoir vérifier la cohérence du MCD et s'il est correct, JMerise va nous générer le MLD (Modèle Logique de Données) en générant lui même les clés étrangères (Figure 5) et le script SQL<sup>8</sup> permettant de créer notre base de données relationnelle.

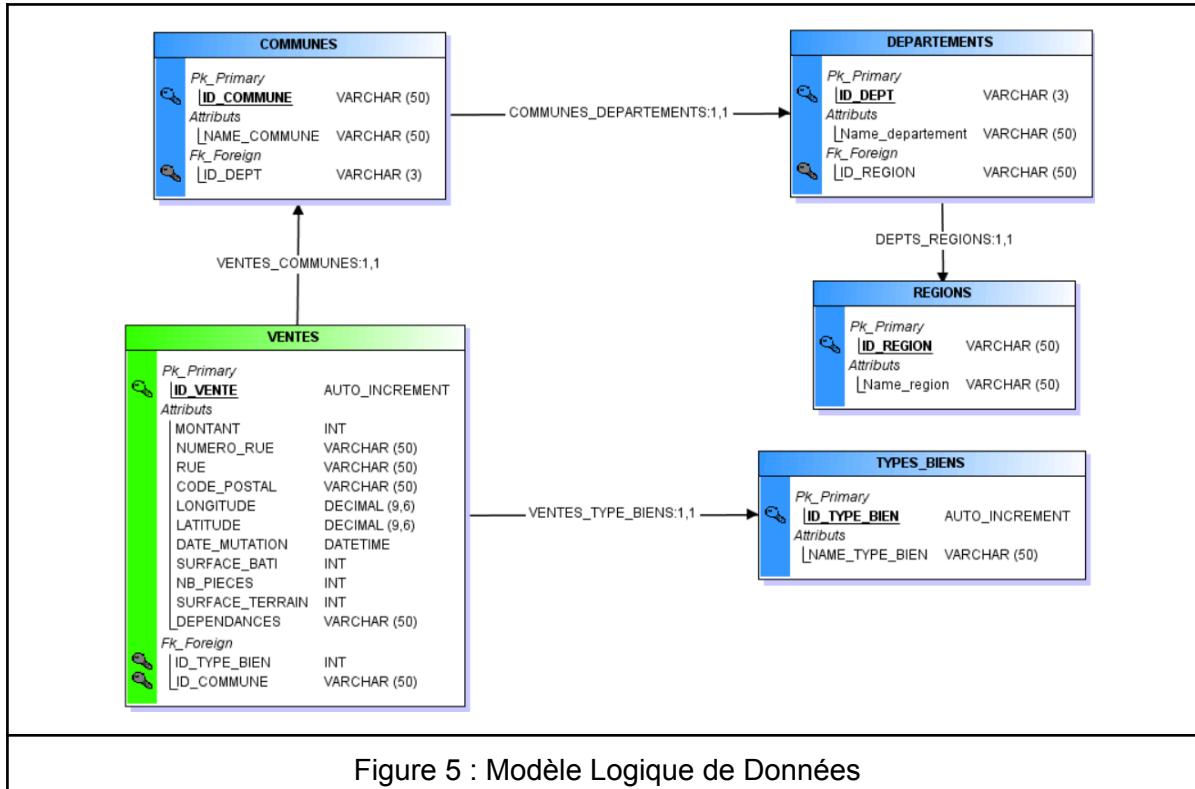


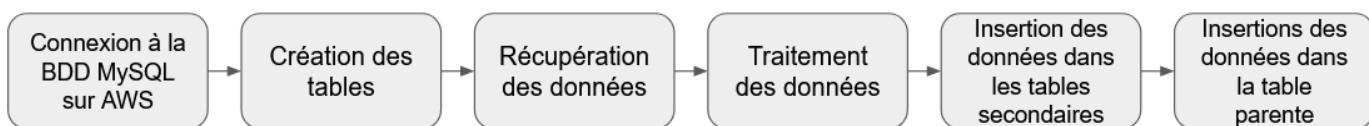
Figure 5 : Modèle Logique de Données

#### d. Création et remplissage de la base de données

La base de données a été créée à l'aide du fichier sql fourni par JMerise sur une base de données Mysql hébergée sur AWS<sup>9</sup> à l'aide des services RDS<sup>10</sup> et IAM<sup>11</sup>.

Le script complet permettant de créer et de remplir la base données est présenté dans le notebook “loading\_first\_data\_in\_rds” de mon dépôt Github dans la branche datagouv\_to\_rds. Ce notebook comporte deux imports de fichier, connection.py permettant d'établir une connexion locale avec la base de données distante AWS, ainsi que le fichier data\_processing.py permettant le nettoyage des données vu dans la partie 2.b.

Le processus est le suivant :



<sup>8</sup> **SQL** : Structured Query Language est un langage de programmation utilisé pour gérer et manipuler des bases de données relationnelles

<sup>9</sup> **AWS** : Amazon Web Service. Du cloud computing permettant de fournir des services comme du stockage, des bases de données, de la puissance de calcul ... plus de 200 services.

<sup>10</sup> **RDS** : Relational Database Service

<sup>11</sup> **IAM** : Identity and Access Management

### 3. EDA

L'EDA (Analyse Exploratoire de Données) a été réalisée en récupérant les données précédemment stockées en base. L'EDA consiste à :

- Comprendre la structure des données :
  - Examiner la distribution
  - Examiner les statistiques
  - Identifier les anomalies ou valeurs aberrantes
- Explorer les relations entre les variables
- Visualiser les données : Utilisation de graphiques permettant de représenter des tendances.

L'ensemble de cette EDA est disponible sur la branche model du dépôt Github.

#### a. Répartition et visualisation des données

##### i. Étendue et nombre de ventes

Le jeu de données s'étend du 1er Juillet 2018 au 31 Décembre 2022. L'année 2023 disponible sur le site data.gouv n'a dans un premier temps pas été ajoutée en base de données afin de pouvoir, par la suite, tester l'import de nouvelles données automatiquement.

Le nombre de ventes recensées durant cette période est de 4 205 071. La figure 6 permet de visualiser la répartition des ventes par type de bien et par année :

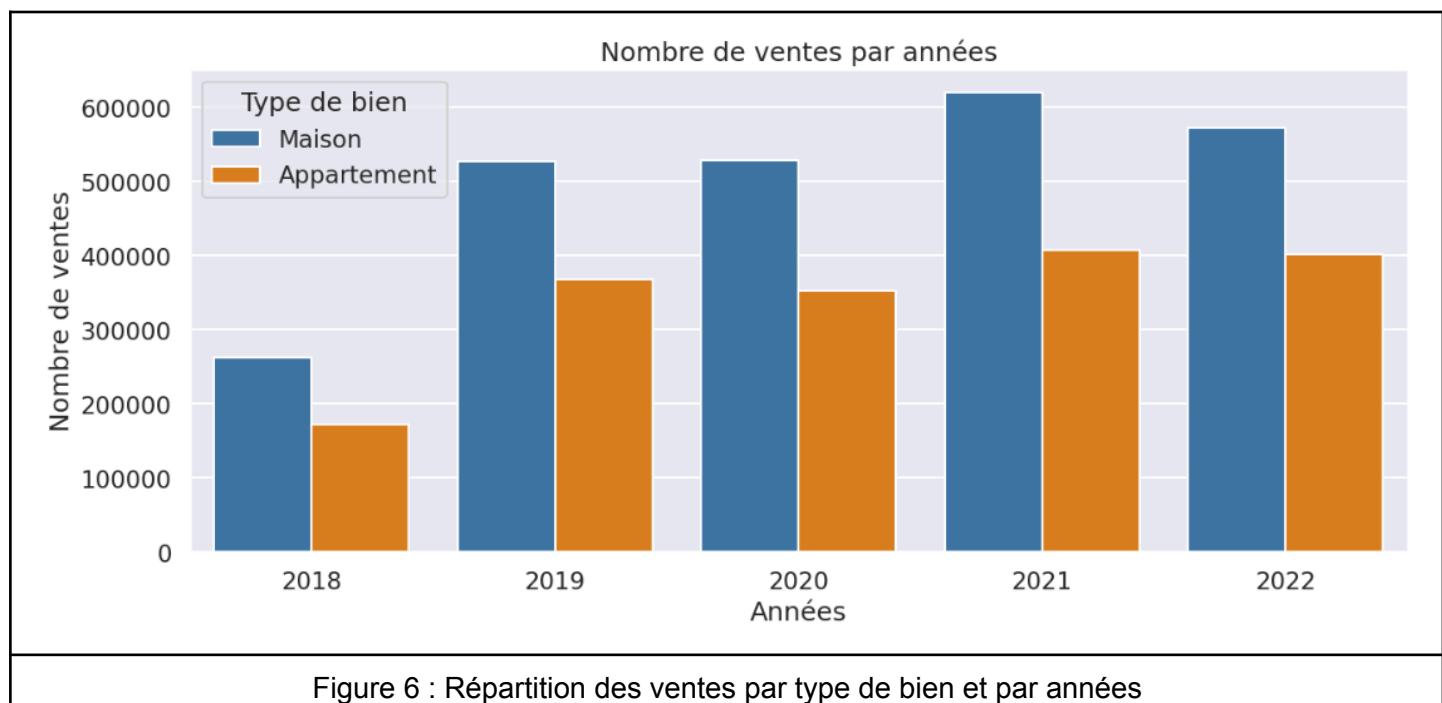
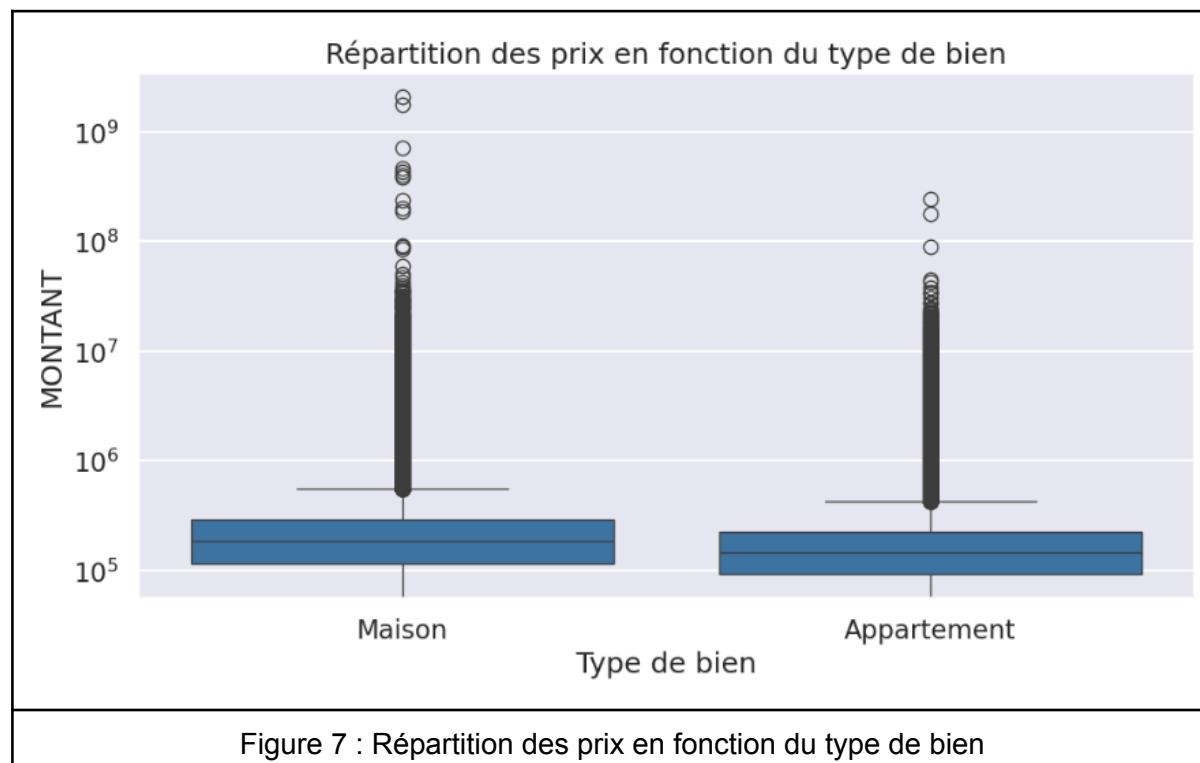


Figure 6 : Répartition des ventes par type de bien et par années

On remarque sur ce graphique que pour une année complète, de 2019 à 2022, le nombre de ventes est assez similaire, de 500 000 à 600 000 pour les maisons et de 350 000 à 400 000 pour les appartements. On remarque également que les ventes de maisons sont supérieures à celles des appartements

## ii. Répartition des prix en fonction du type de bien

Pour visualiser la répartition des prix en fonction du type de bien, la première méthode choisie est le boxplot. Cette méthode permet également d'avoir un premier aperçu de valeurs aberrantes. Le graphique figure 7 est représenté en échelle logarithmiques car la différence de prix entre les biens peut-être significative, exemple entre un studio en banlieue et un manoir Parisien.



Ce graphique ne permet pas de visualiser correctement la répartition des prix de vente en fonction du type de bien car il y a un nombre important de valeurs extrêmes (appelés outliers).

L'impression des quartiles donne des informations plus précises:

Répartition pour Maison :

[0, 116000, 186000, 290000, 2086000000]

Répartition pour Appartement :

[0, 92600, 146500, 224750, 245365904]

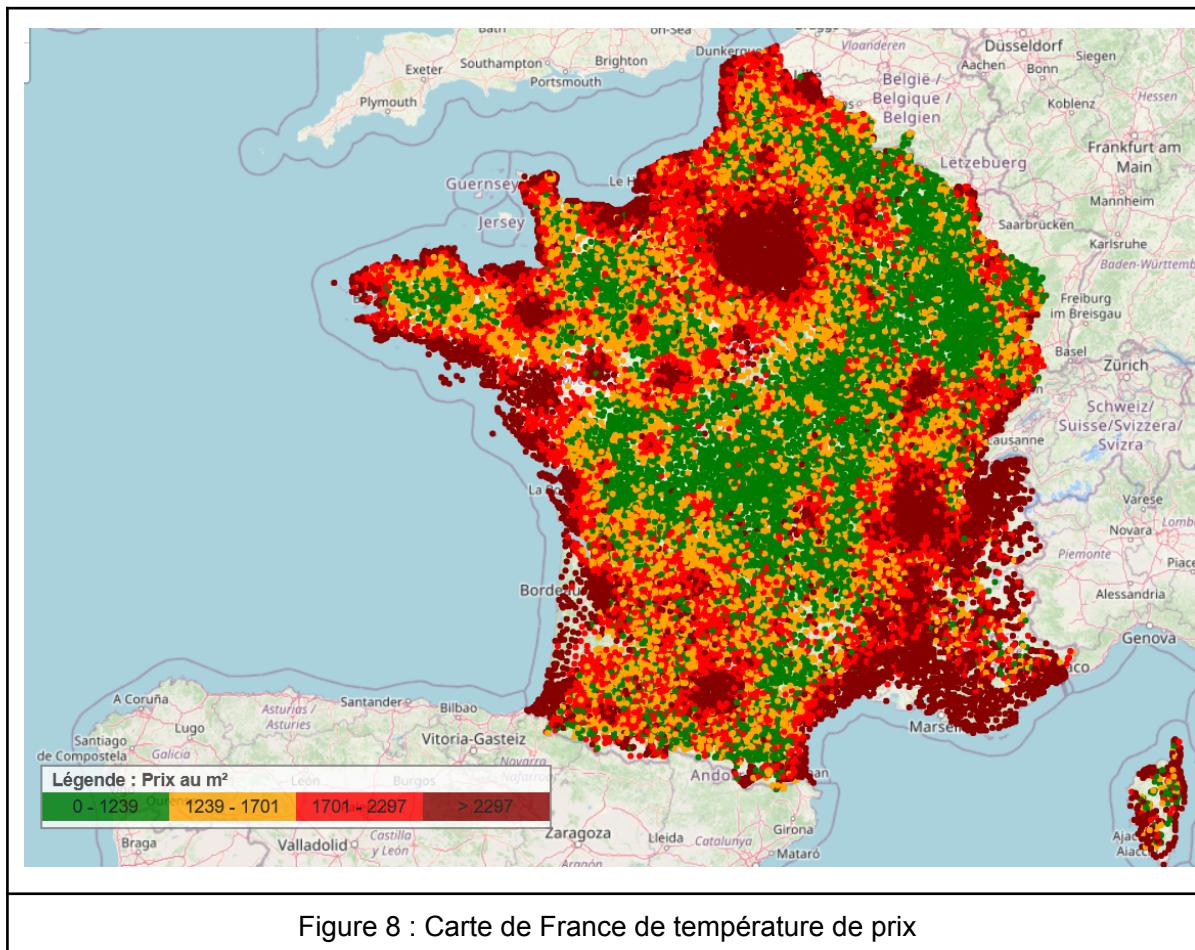
Ces données nous donnent les informations suivantes :

- Des biens ont été vendus à 0€ (après recherche 3 maisons et 7 appartements)
- Le prix médian est de :
  - 186 000€ pour une maison
  - 146 500€ pour un appartement.
- Le prix le plus élevé est de :
  - Plus de 2 milliards pour une maison
  - Plus de 200 millions pour un appartement

**Si un seul modèle est créé alors il faudra supprimer ces outliers, qui sont des biens d'exception ou des erreurs de saisie. Afin de se rapprocher au mieux des standards de ventes et de maximiser les performances.**

### iii. Visualisation de la répartitions des prix en fonction de la localisation

Pour réaliser cette visualisation, la bibliothèque folium de python a permis de créer une carte interactive des prix en fonction de la géolocalisation des biens disponibles dans le jeu de données. Une carte par type de bien a été créée en affectant le prix moyen au mètre carré par commune et en utilisant les quantiles de toutes ces moyennes pour créer une "température" des prix, du vert (prix moyen le moins cher) à bordeaux(prix moyen le plus cher). La figure 8 représente une capture d'écran de la carte des températures de prix :



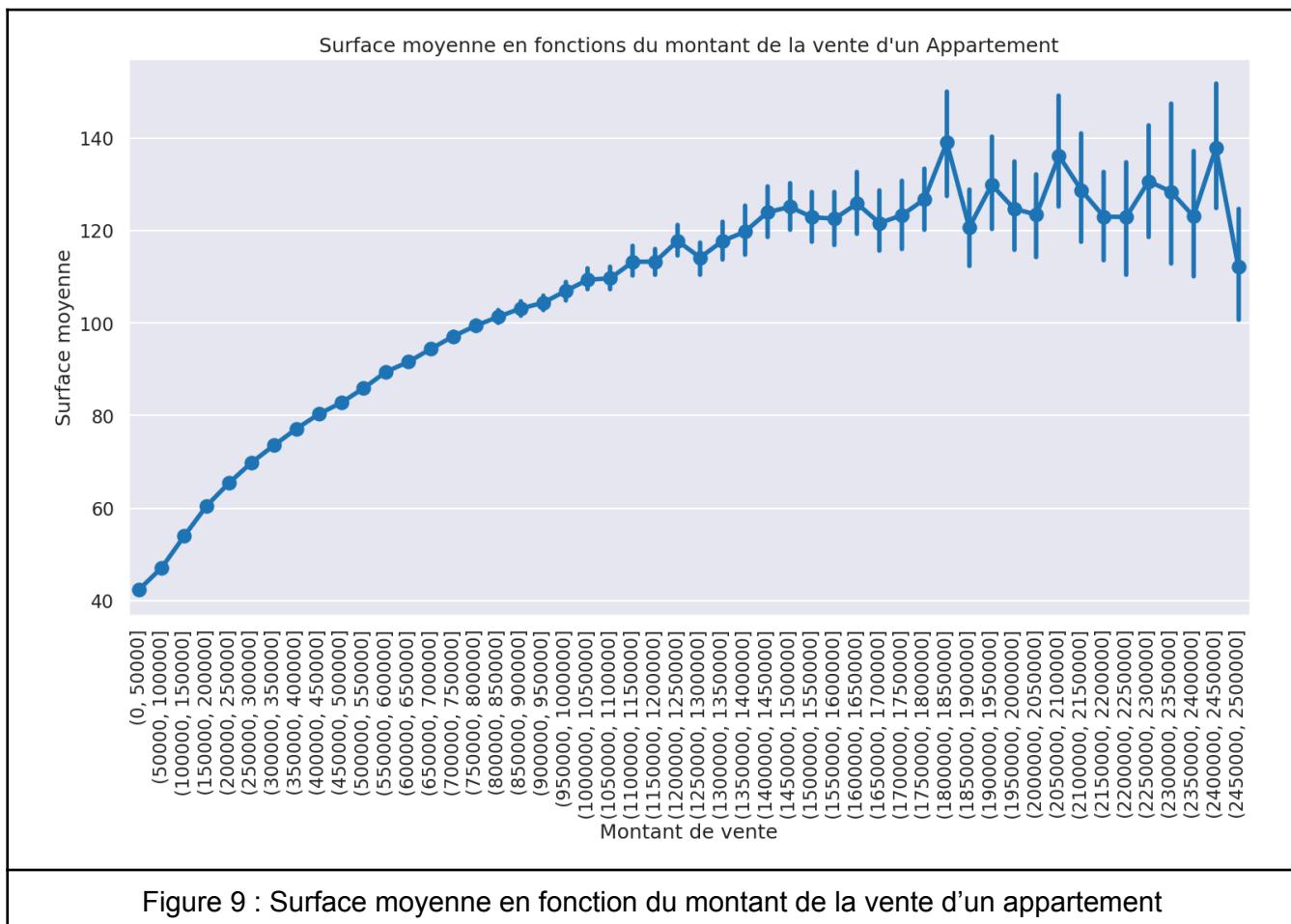
Un bien situé dans une grande agglomération, sur la côte ou dans les Alpes aura un prix moyen plus élevé qu'un même bien situé dans le département de la Creuse.

**Cette carte met en évidence l'importance de la localisation d'un bien pour déterminer son prix.**

### iv. Visualisation de l'évolution des prix en fonction de sa surface habitable

La figure 9 montre que plus un logement est grand, plus son prix est élevé. Ce qui est vrai jusqu'à une certaine limite, par exemple pour un appartement, au-delà d'un prix de vente moyen d'environ 150 000 € la surface n'a plus beaucoup d'influence. La différence de prix au-delà des 100 120 m<sup>2</sup> est sûrement dûe à sa localisation ou son standing.

**La surface corrélée à la localisation pourra donc être un élément important dans la construction de notre modèle.**



#### v. Visualisation de l'évolution des prix en fonction du nombre de pièces

La figure 10 représente l'évolution du prix moyen d'un bien immobilier (en région Île-de-France) en fonction du nombre de pièces que compose ce logement :

Ce graphique montre que plus un bien a de pièces, plus l'estimation d'un bien sera difficile à déterminer. Plus le logement est composé de pièces plus le violinplot s'allonge. L'évolution du prix moyen d'un bien en fonction du nombre de pièces qui le compose est très similaire à celle de l'évolution du prix en fonction de la surface. Ce qui est logique, plus il y a de pièces plus le bien est grand.

**Lors de la création du modèle il sera peut-être pertinent de ne prendre que la surface ou que le nombre de pièces afin de réduire la complexité du modèle, réduire le temps d'entraînement ou même améliorer les performances du modèle en évitant le surajustement.**

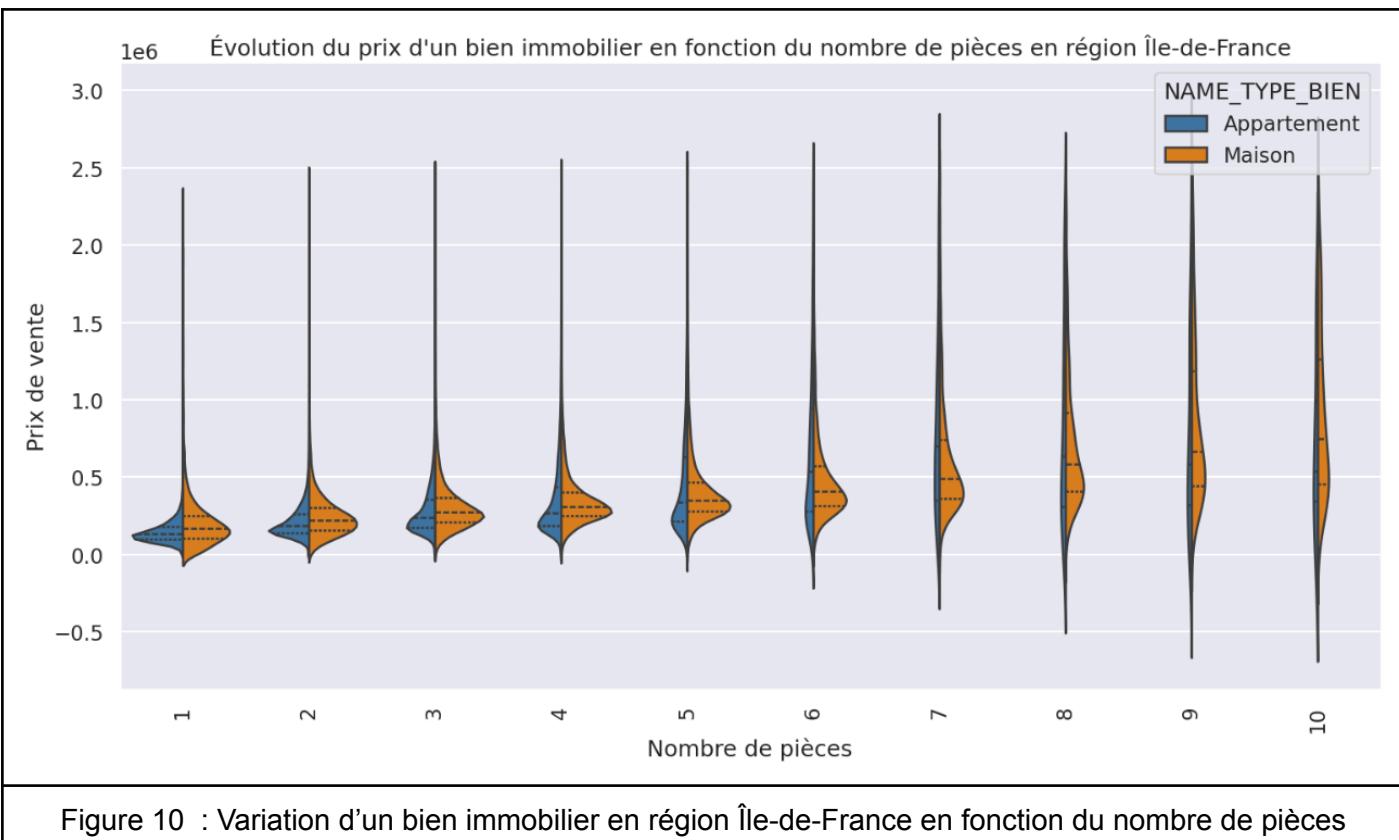


Figure 10 : Variation d'un bien immobilier en région Île-de-France en fonction du nombre de pièces

#### vi. Evolution des prix dans le temps

On entend souvent parler de la “flambée” du prix de l’immobilier, qu’en est-il réellement ? La figure 11 montre l’évolution du prix moyen d’un logement en France réparti en trimestre de Juin 2018 à Décembre 2022 :

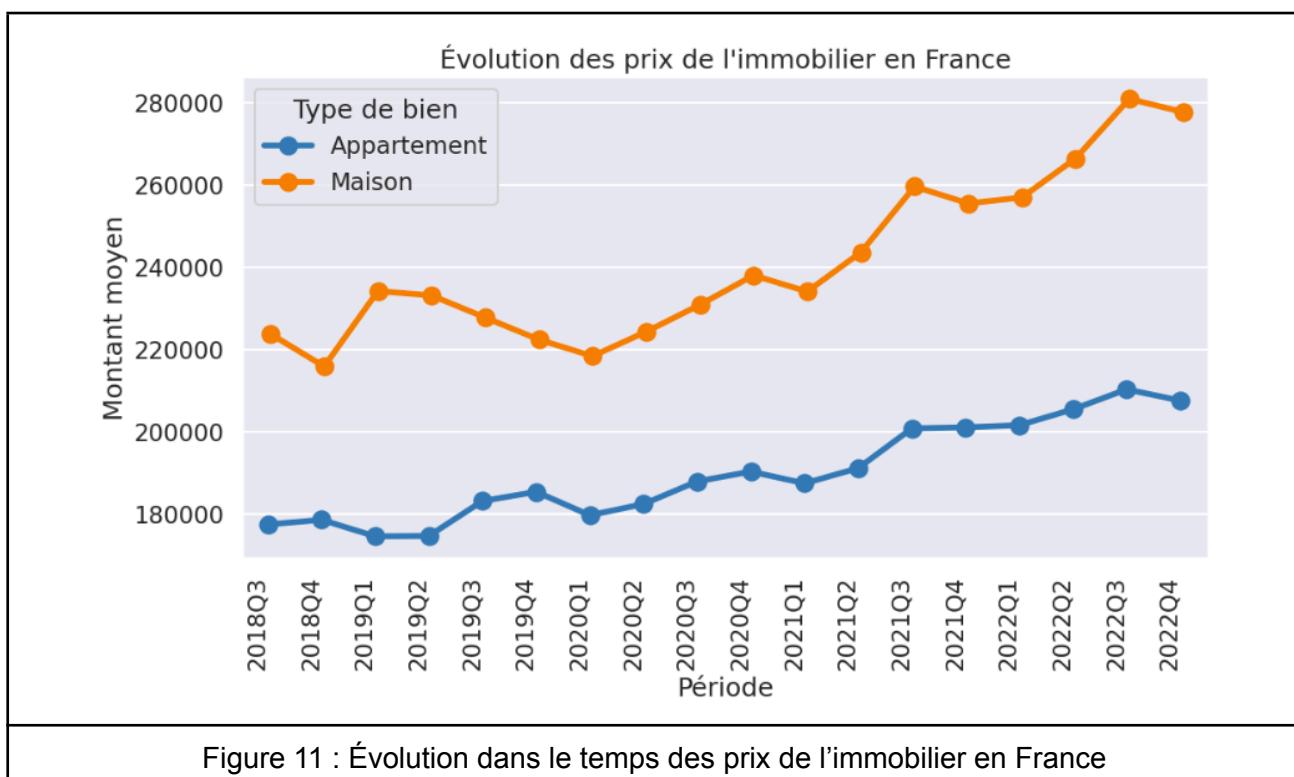


Figure 11 : Évolution dans le temps des prix de l’immobilier en France

Effectivement, ce graphique montre une évolution à la hausse des prix de l'immobilier. Entre le troisième trimestre 2018 et fin 2024, le prix moyen d'une maison a augmenté d'environ 27% contre 20% pour un appartement. On remarque d'ailleurs une évolution plus rapide du prix moyen d'une maison par rapport à celui d'un appartement à partir du premier trimestre 2021 correspondant à la fin du deuxième confinement COVID.

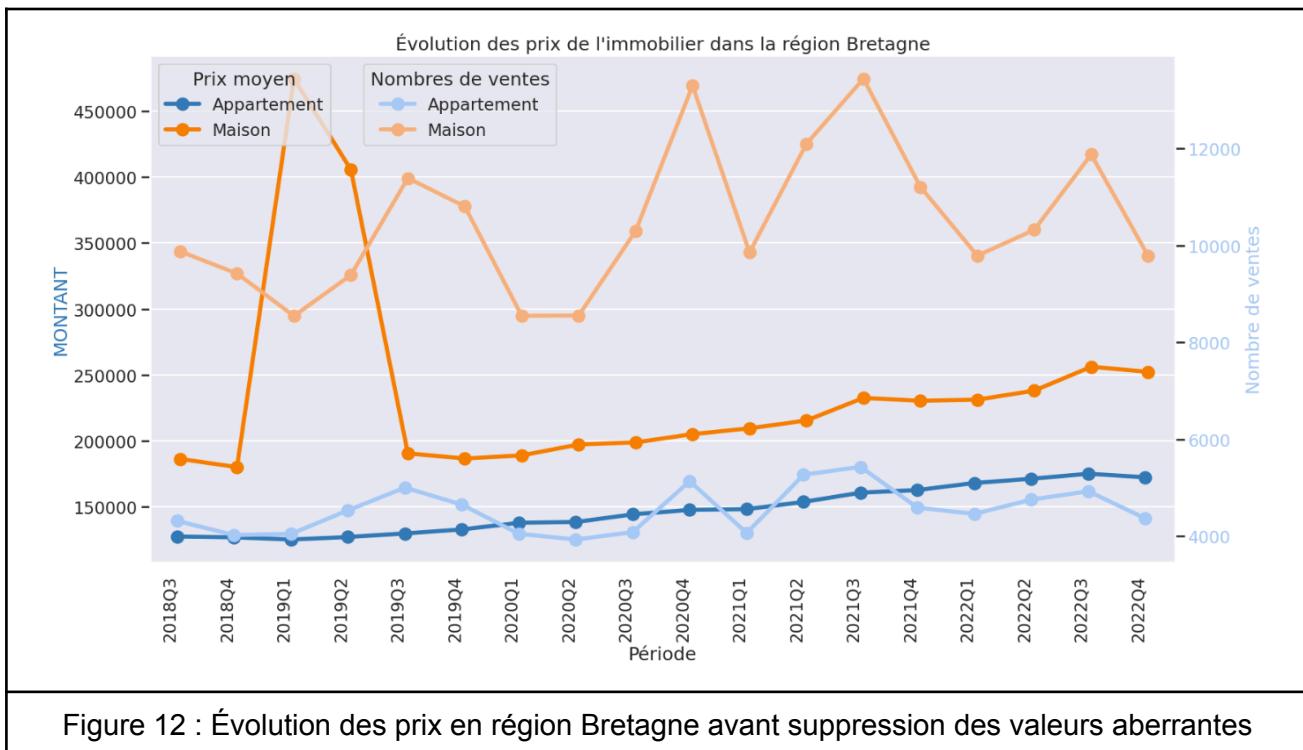
Pour la création du modèle plusieurs solutions peuvent être envisagées. L'utilisation d'un réseau de neurones basé sur les séries temporelles ou la création de plusieurs modèles de régression à des intervalles différents (exemple un modèle pour 2019 ,un pour 2020...) et de choisir un modèle en fonction des tendances de marchés.

### b. Visualisation et traitements des valeurs aberrantes

L'étude sur la répartition des prix en fonction du type de bien, chapitre 3.a.i , a pu mettre en évidence des valeurs aberrantes. Quelle en est l'origine et quelles sont les actions à mener ?

#### i. Étude sur les prix élevés

La récupération des données pour le bien ayant le montant le plus élevé nous renseigne sur le bien. Il s'agit d'une maison de 135m<sup>2</sup> vendue en 2019 située à Plouzévédé dans le Finistère. En rentrant l'adresse du bien dans Google map on peut constater que ce bien est situé dans un lotissement et, à moins que sa fabrication soit en or, il est impossible que sa valeur atteigne les 2 milliards. Il s'agit donc d'une erreur de saisie.



La figure 12 avant suppression de valeurs aberrantes montre bien une évolution anormale pour le premier et deuxième trimestre 2019, faisant passer le prix d'une moyenne de 200 000 à 450 000€. En supprimant les biens supérieurs à 7,5 millions l'évolution de la tendance redéveloppe normale comme le montre la figure 13 :

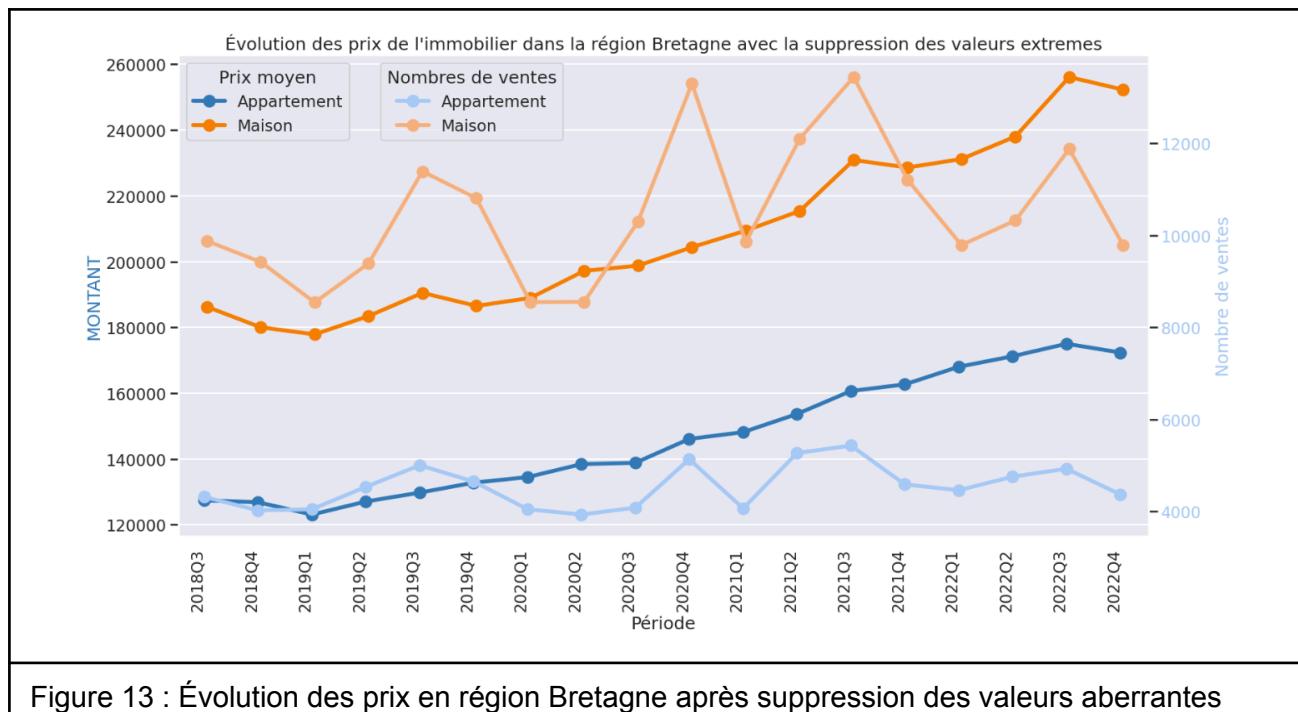


Figure 13 : Évolution des prix en région Bretagne après suppression des valeurs aberrantes

## ii. Étude sur les prix bas

Précédemment on a pu remarquer que certain bien était vendu à 0€. Il est plus difficile de détecter des valeurs aberrantes avec un montant faible ou nul car la différence entre la moyenne des prix et 0 est moins importante qu'avec le prix de vente d'un bien faisant 10 000 fois le prix moyen, comme vu dans l'étude des prix élevés.

Après une étude sur "le bon coin", on peut constater que des appartements peuvent-être vendus à "temps partagé", c'est-à-dire qu'on dispose des parts de l'appartement et qu'on peut l'occuper seulement pour X semaines par an. D'autres biens peuvent aussi fausser le futur modèle, par exemple, un garage ou un cabanon vendu sous l'étiquette d'une maison ou appartement.

## iii. Traitements des valeurs aberrantes

Le traitement des valeurs aberrantes est donc plutôt subjectif, il n'existe pas de valeurs précises d'une valeur minimum ou maximum d'un bien. On peut estimer qu'un appartement à moins de 15 000€ est pratiquement impossible et qu'une maison à plus de 5 millions est plutôt rare. Une autre possibilité serait de supprimer les outliers présents sur les boxplots de chaque région.

## c. Corrélations entre les variables

Les points précédents de l'analyse exploratoire de données permettent de mettre en évidence l'importance de la situation géographique et de la surface d'un bien immobilier sur son prix de vente. Une variable représentant le prix au mètre carré a donc été créée afin d'établir une relation entre la situation géographique et le prix d'un bien immobilier.

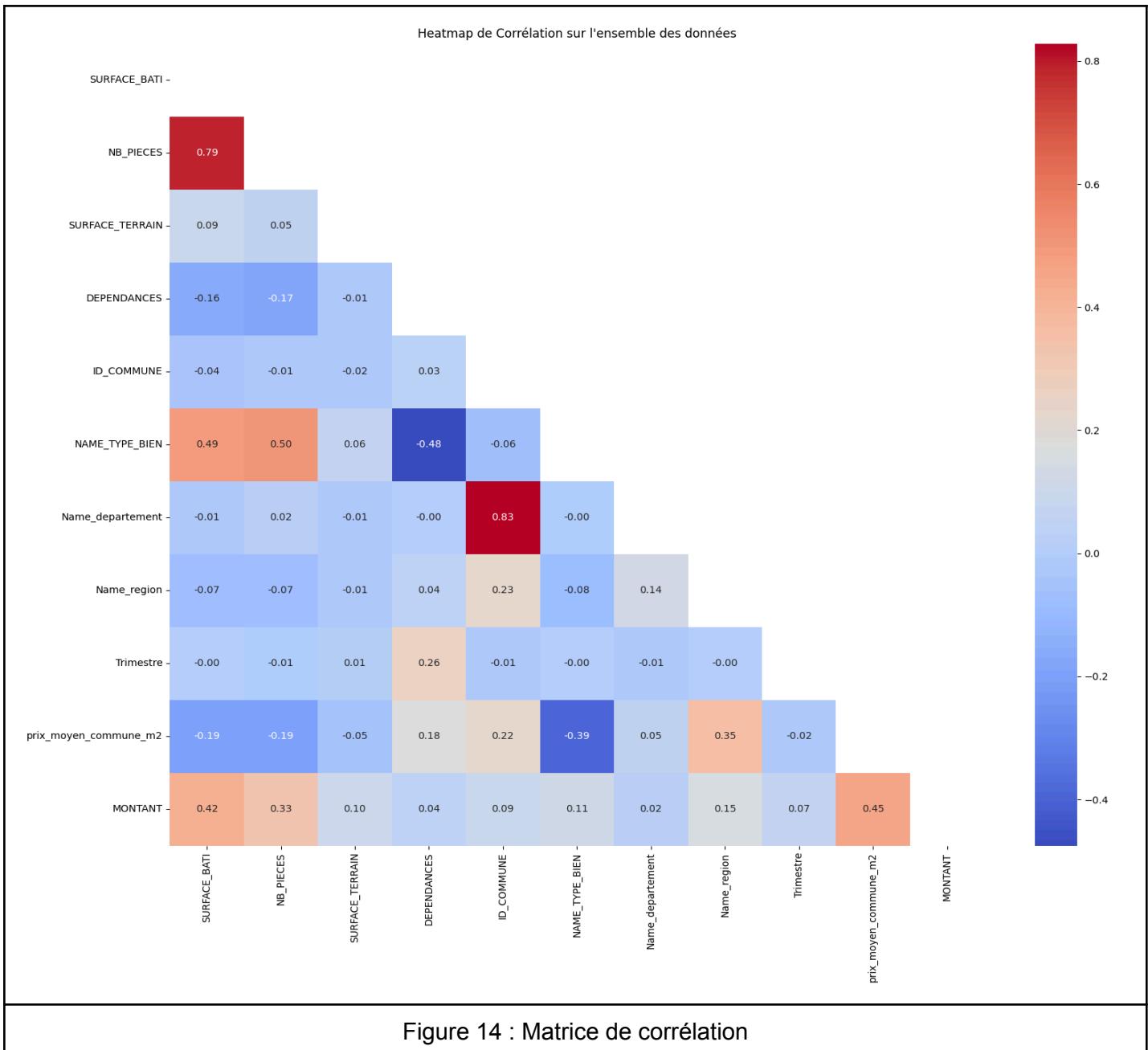


Figure 14 : Matrice de corrélation

Cette matrice de corrélation figure 14, confirme les points précédents vus pendant l'analyse exploratoire de données. Les variables ayant le plus d'importance pour l'estimation d'un bien immobilier sont :

- La surface (0.42)
- Le prix au mètre carré par commune(0.45)
- Le nombre de pièces(0.33)
- La région(0.15)
- Le type de bien(0.11)
- La surface du terrain(0.1)

Cependant il faut aussi mettre en évidence les corrélations entre les variables n'incluant pas le montant :

- Le nombre de pièces et la surface du bien(0.79)
- La région et le prix au mètre carré(0.35)
- Le type de bien et le prix au mètre carré(-0.39)

Afin d'éviter un sur-ajustement et un temps de calcul trop long, la surface du bien sera choisie au détriment du nombre de pièces car cette variable a une plus forte corrélation avec le prix d'un bien.

La surface du terrain n'est corrélée qu'à 10% avec le montant mais elle n'a pas de forte corrélation avec d'autres variables. Il sera donc intéressant de voir si elle peut améliorer les performances d'un modèle.

#### *d. Conclusion EDA*

Cette analyse exploratoire de données a permis de mettre en évidence :

- Le traitement de données à effectuer pour éliminer les valeurs aberrantes.
- De faire un choix pertinent de variables pour la construction des futurs modèles d'intelligence artificielle tout en optimisant les performances et en minimisant le risque de surajustement.
- L'évolution du prix de l'immobilier en fonction du temps. Malheureusement, la variable "Trimestre" qui a été créée n'a pas beaucoup d'influence sur le montant. Pour ces raisons des modèles seront entraînés sur des durées différentes (toute la plage de données et seulement sur une année) pour voir si les performances sont améliorées.

## 4. Veille technique

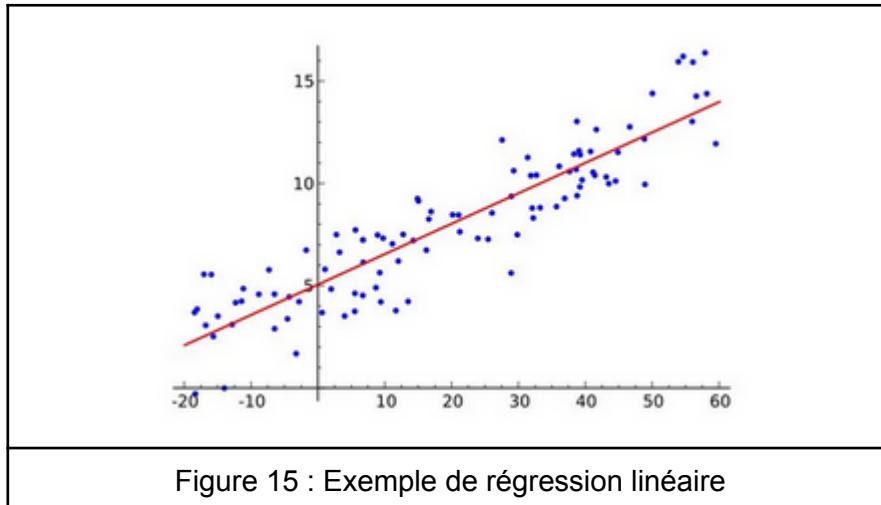
### a. Introduction

Dans ce projet, ce qu'on cherche à déterminer est le prix d'une maison en fonction de ses caractéristiques. Le prix étant une variable continue nous allons donc nous intéresser aux modèles de régression linéaire.

#### i. Définition de la régression linéaire (tiré du site Kobia.fr)

Le site Kobia.fr, spécialisé dans le domaine de l'intelligence artificielle et de la data sciences et auteur de multiples articles pour les débutants comme pour les experts, permet de réaliser une première approche sur la régression linéaire(lien disponible dans le chapitre 11.Bibliographies) .

La régression linéaire est une technique statistique de modélisation des relations entre différentes variables (dépendantes ou indépendantes). Utilisée pour décrire et analyser les valeurs ou données, la régression linéaire a pour objectif de réaliser des prédictions ou des prévisions.



La figure 15 ci-dessus nous montre la représentation d'une régression linéaire simple, les points bleus représentent les données et la droite en rouge, la droite de régression qui peut être déterminée par l'équation suivante :

$$y = \beta_0 + \beta_1 x$$

y est la valeur dépendante  
x est la valeur indépendante  
B0 est l'ordonnée à l'origine  
B1 est le coefficient directeur de la droite

#### ii. La fonction coût ou perte (loss function)

La CNIL, Commission Nationale de l'Informatique et des Libertés, une autorité administrative indépendante, dit que, dans le domaine de l'intelligence artificielle, la fonction de perte ou de coût est la quantification de l'écart entre les prévisions du modèle et les observations réelles(lien disponible dans le chapitre 11.Bibliographies).

L'erreur se mesure entre notre modèle  $f(x)$ , dont on a choisi les valeurs de  $a$  et  $b$  aléatoirement, et les différentes valeurs présentes dans le dataset  $y$ . Le modèle est  $f(x)=ax+b$  et les valeurs  $y$ . On fait alors la différence entre les deux pour calculer l'erreur. Ci-dessous la formule pour calculer une erreur :

$$(f(x^{(i)}) - y^{(i)})^2$$

Formule de la fonction coût pour l'ensemble des données :

$J(\theta)$  est la fonction coût,  
 $\theta$  représente les paramètres du modèle,  
 $m$  est le nombre d'exemples d'entraînement,  
 $f_{\theta}(x^{(i)})$  est la prédiction du modèle (fonction hypothèse) pour l'exemple  $i$ ,  
 $y^{(i)}$  est la valeur réelle associée à l'exemple  $i$ .

L'objectif étant d'obtenir une fonction coût proche de 0, afin de limiter le nombre d'erreurs.

### iii. Les algorithmes de minimisation de l'erreur

- La méthode des moindres carrés :

La méthode des moindres carrés donne directement les paramètres d'optimisation de la fonction coût mais dans un jeu de données contenant énormément de données, ce qui est le cas dans avec le projet d'estimation de biens immobilier, le temps de calcul devient extrêmement long et il devient donc plus judicieux d'opter pour la méthode de descente de gradient.

- La descente de gradient :

La descente de gradient va chercher pas à pas les meilleurs paramètres pour minimiser la fonction coût. La vidéo réalisée par Machine Learnia permet de bien comprendre le fonctionnement de la descente de gradient. Dans la vidéo, la recherche de minimisation de la fonction coût est illustrée par la vallée d'une montagne où la fonction coût la plus réduite serait le bas de la vallée.

- Gradients :

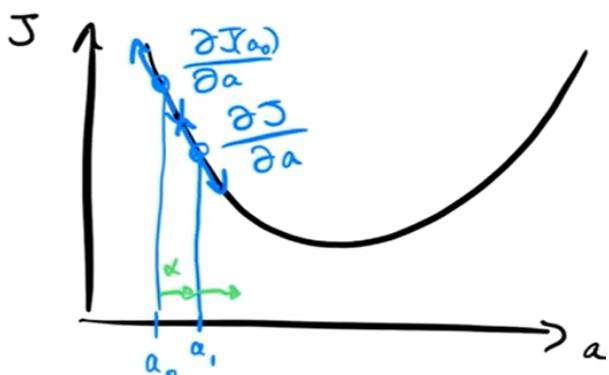
$$\frac{\partial J(a, b)}{\partial a} = \frac{1}{m} \sum_{i=1}^m x^{(i)}(ax^{(i)} + b - y^{(i)})$$

$$\frac{\partial J(a, b)}{\partial b} = \frac{1}{m} \sum_{i=1}^m (ax^{(i)} + b - y^{(i)})$$

- Algorithme de Gradient Descent :

$$a = a - \alpha \frac{\partial J(a, b)}{\partial a}$$

$$b = b - \alpha \frac{\partial J(a, b)}{\partial b}$$



1. L'utilisateur doit préalablement déterminer le coefficient alpha appelé learning rate.
2. L'algorithme commence au point a0 et calcul la pente de la fonction coût à cet endroit (la pente est la dérivée de la fonction coût).
3. Le point a1 va être déterminé par la formule suivante :  $a1=a0-(\text{alpha} \times \text{la pente})$
4. L'algorithme va itérer cette opération jusqu'à trouver le bon gradient pour lequel  $a_{n+1}=a_n$

#### iv. Métrique de performances

Pour déterminer les performances d'un modèle de régression linéaire il existe plusieurs métriques. Le site Kobia.fr (lien disponible dans le chapitre 11.Bibliographies) permet de montrer et d'expliquer les différences entre les métriques disponibles les plus utilisées. Ainsi il permet de faire un choix éclairé en fonction de nos données, afin d'obtenir la meilleure représentation des performances de nos modèles.

- La MAE (Mean Absolute Error):

La MAE ou erreur absolue moyenne , est la moyenne des valeurs absolues des erreurs. La MAE est dans la même unité que la variable à prédire. Par conséquent, elle est facilement interprétable.

- La MSE et RMSE (Root Mean Squared Error):

La Root Mean Squared Error (RMSE) et la Mean Squared Error (MSE) sont les métriques de régression les plus courantes. Du fait de leurs propriétés de régularité, ce sont les métriques historiques pour optimiser les modèles de régression comme la régression linéaire.

La RMSE, ou racine de l'erreur quadratique moyenne, est – comme son nom l'indique – la racine carrée de la MSE. Mathématiquement, elle est définie par :

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Contrairement à la MSE, la RMSE s'exprime dans la même unité que la variable à prédire et est par conséquent plus facile à interpréter.

La MSE et la RMSE sont donc particulièrement utiles lorsque l'on préfère éviter de faire de grandes erreurs. Elles seront aussi utiles lorsque l'on cherche une solution explicite au problème donné.

- LA MAPE (Mean Absolute Percentage Error):

La Mean Absolute Percentage Error (MAPE) est la métrique de régression utilisée lorsque l'on considère l'erreur du modèle en proportion de la valeur prédite. Pour la comprendre intuitivement, prenons un exemple : vous prédisez le nombre de personnes qui viendront à la soirée que vous organisez, et vous prédisez correctement ce nombre à un invité près. L'effet de votre erreur n'est pas le même selon que vous avez 5 invités ou 100 invités : un invité de plus ou de moins dans une soirée de 100 invités a beaucoup moins d'importance que dans une soirée de 5 invités.

- La MSLE (Mean Squared Logarithmic Error):

La MSLE est proche de la MSE, mais les valeurs réelles et prédites sont remplacées par leurs logarithmes pour prendre en compte des variations exponentielles. Cela en fait une métrique adaptée lorsque les valeurs prédites varient sur une grande échelle.

La MSLE ne s'exprime pas dans une unité simple, ce qui la rend difficile à interpréter. Mais grâce à l'utilisation du logarithme, la MSLE réduit l'importance des erreurs pour de grandes valeurs réelles comme le fait la MAPE. Cette métrique quantifie l'erreur réalisée par le modèle. Plus elle est élevée, moins le modèle est performant.

Métrique	Avantages	Inconvénients	Exemple
MSE/RMSE	Accentue les fortes erreurs, régulière, optimisable	Sensible aux outliers	On accepte 5 erreurs de 1°C plus qu'une seule erreur de 5°C
MAE	Homogène, interprétable	Sensible aux outliers	5 erreurs de 1°C sont équivalentes à une seule erreur de 5°C
MAPE	Robuste aux changements d'échelle, interprétable	Sensible aux faibles valeurs, utilisable sur des valeurs non nulles uniquement	Une erreur de 10% sur une réalité de 10€ (9 ou 11€) est équivalente à une erreur de 10% sur une réalité de 100€ (90 ou 110€)
MSLE	Robuste aux changements d'échelle, régulière, optimisable	Peu interprétable, utilisable sur des valeurs positives, non symétrique	On préfère surestimer de 10% que de sous estimer de 10% peu importe la valeur de la réalité.

Figure 16 : Tableau récapitulatif des métriques

- R<sup>2</sup> score

Le R<sup>2</sup> score, aussi appelé coefficient de détermination, quantifie la performance d'un modèle de régression. Le R<sup>2</sup> score est l'une des métriques les plus utilisées pour la régression linéaire. Cette métrique est une version "normalisée" de la MSE.

Le R<sup>2</sup> score est défini par la formule :

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

- La variable à prédire prend les valeurs  $y_i$  pour  $1 \leq i \leq n$
- Les prédictions sont les  $\hat{y}_i$

On peut voir le R<sup>2</sup> comme l'erreur du modèle divisé par l'erreur d'un modèle qui prédit tout le temps la moyenne de la variable à prédire. Le score R<sup>2</sup> est d'autant plus élevé que le modèle est performant, et vaut au maximum 100%, lorsque toutes les prédictions sont exactes. Il n'y a pas de score minimum, mais un modèle simple prédisant tout le temps

la valeur moyenne atteint un score R2 de 0%. Par conséquent un score R2 négatif signifie que les prédictions sont moins bonnes que si l'on prédisait systématiquement la valeur moyenne.

Le R2 à trois spécificités :

- Il facilite la comparaison entre différents modèles.
- Il permet de situer le modèle entre un modèle non performant et parfait.
- Il est en revanche peu interprétable et ne donne pas d'information sur l'erreur moyenne du modèle.

## b. Les différents modèles d'intelligence artificielle disponibles

Il existe une multitude de modèles d'intelligence artificielle pouvant répondre à notre problème. Dans cette étude on va s'intéresser à trois modèles en utilisant python :

- RandomForestRegressor de scikit Learn
- XGBoost
- Un réseau de neurones Sequential avec Tensor Flow et Keras

### i. RandomForestRegressor

Pour expliquer le principe de fonctionnement d'un modèle Random Forest Regressor (forêt d'arbres aléatoires pour la régression), il faut déjà commencer par comprendre le fonctionnement d'un arbre de décision.

Définition simple par Arthur Flor (data engineer et contributeur actif à la communauté technologique, il a écrit plusieurs articles sur le site medium) : Un arbre de décision est une structure de données dans laquelle la combinaison de conditions (nœuds) conduit à une décision (nœuds feuille). On utilise donc l'apprentissage automatique pour créer le meilleur arbre de décision en fonction des données en notre possession(lien disponible dans le chapitre 11.Bibliographies).

Structure : La structure d'un arbre de décision est hiérarchique et arborescente. Elle est composée d'un nœud racine, de branches, de nœuds internes et de nœuds feuilles (IBM, société multinationale américaine spécialisée dans la technologie et les services informatiques). L'arbre de décision est aussi caractérisé par sa profondeur, c'est le nombre de niveaux entre le nœud racine et les feuilles. Une illustration, figure 17, adaptée du graphique d'IBM dont le lien est disponible dans le chapitre 11.Bibliographies :

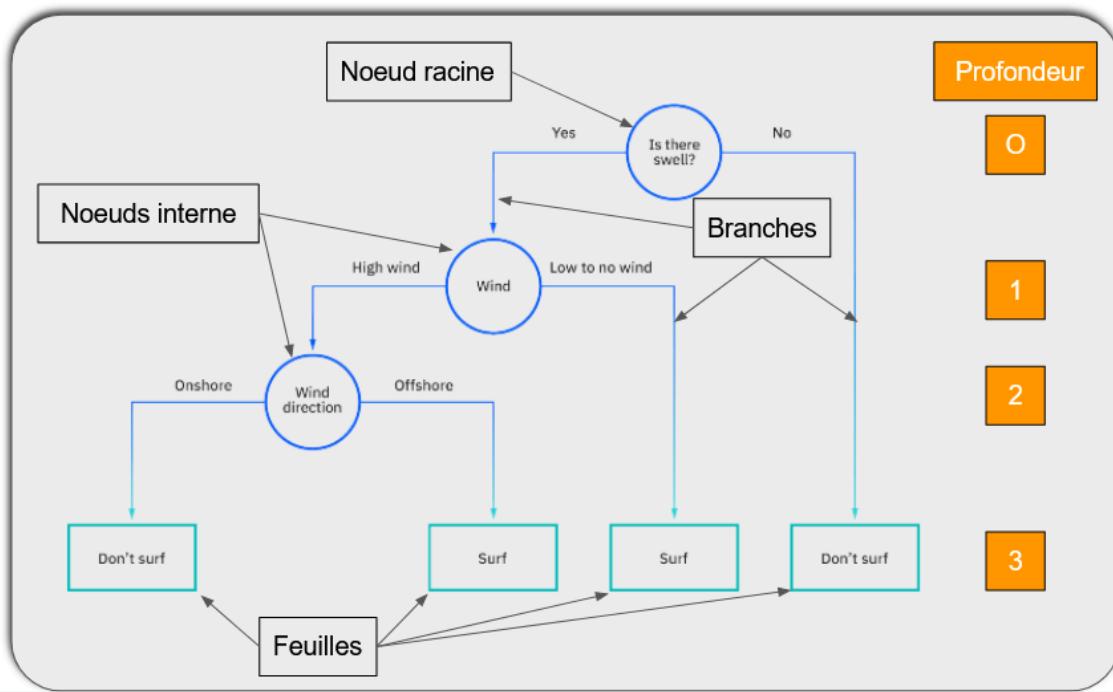


Figure 17 : Structure d'un arbre de décision

### Construction d'une forêt d'arbre aléatoire :

La forêt aléatoire ou random forest est un algorithme de machine learning proposé par Leo Breiman et Adèle Cutler en 2001[Citations Ch.12]. C'est un algorithme qui se base sur l'assemblage d'arbres de décision.

Chaque random forest est constitué d'un ensemble d'arbres de décision indépendants où chaque arbre (Figure 18) dispose d'une vision parcellaire du problème du fait d'un double tirage aléatoire :

- o Un tirage aléatoire avec remplacement sur les observations (les lignes de la base de données) ce que l'on appelle le tree bagging.
- o Un tirage aléatoire sur les caractéristiques (les colonnes de la base de données), ce que l'on appelle le feature sampling.

A la fin tous ces arbres de décisions indépendants sont assemblés(Figures 19 et 20). Le résultat final est réalisé à l'aide d'un vote (ou voting) qui prend par défaut, dans un problème de classification, la réponse la plus fréquente parmi tous les arbres de décision aléatoires indépendants.

Illustration :

### 1 Construction de l'arbre N°1 du random forest

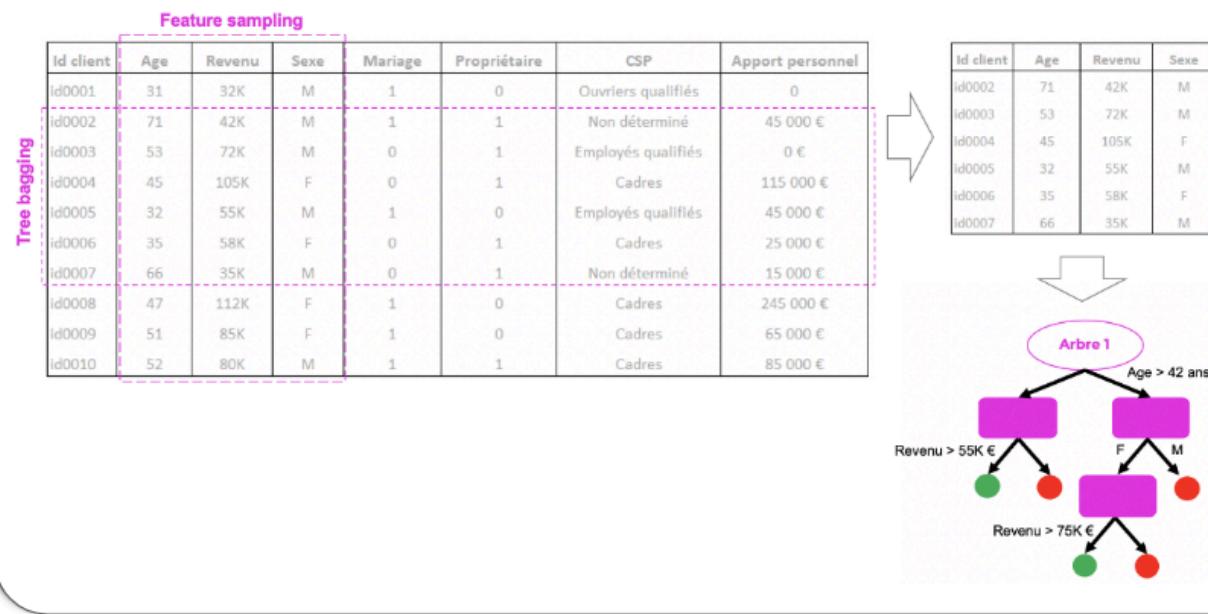


Figure 18 : Construction d'un arbre de décision

### 2 Mise en œuvre d'une forêt d'arbres

Id client	Age	Revenu	Sexe
id0002	71	42K	M
id0003	53	72K	M
id0004	45	105K	F
id0005	32	55K	M
id0006	35	58K	F
id0007	66	35K	M

Id client	Age	CSP	Apport personnel
id0001	31	Ouvriers qualifiés	0
id0007	66	Non déterminé	15 000 €
id0008	47	Cadres	245 000 €
id0009	51	Cadres	65 000 €
id0010	52	Cadres	85 000 €

Id client	Mariage	Propriétaire	CSP
id0001	1	0	Ouvriers qualifiés
id0003	0	1	Employés qualifiés
id0005	1	0	Employés qualifiés
id0007	0	1	Non déterminé
id0009	1	0	Cadres

Figure 19 : Mise en oeuvre d'une forêt d'arbres aléatoires

### 3 Prédiction du random forest

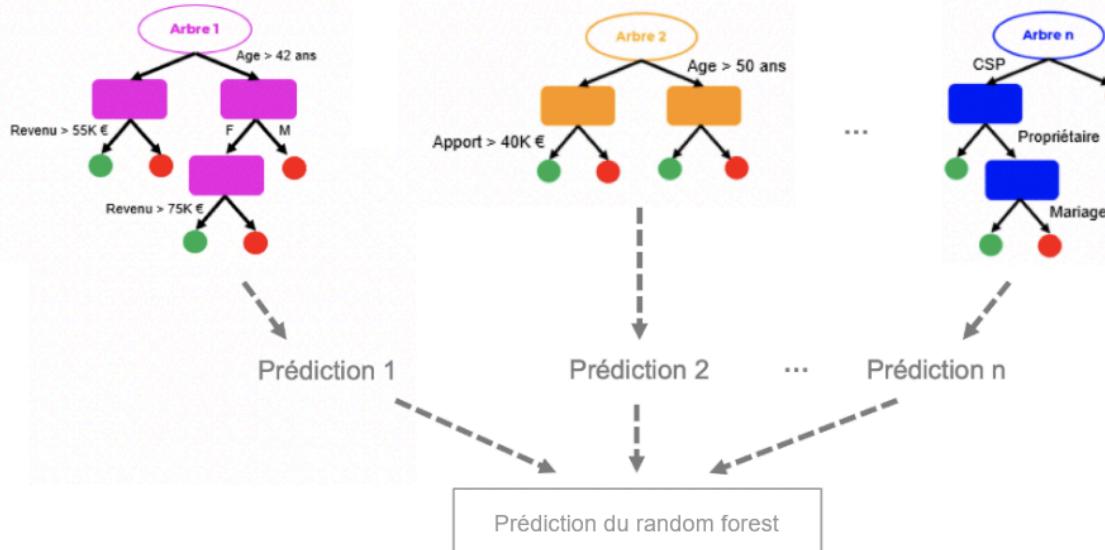


Figure 20 : Prédiction d'une forêt d'arbres aléatoires

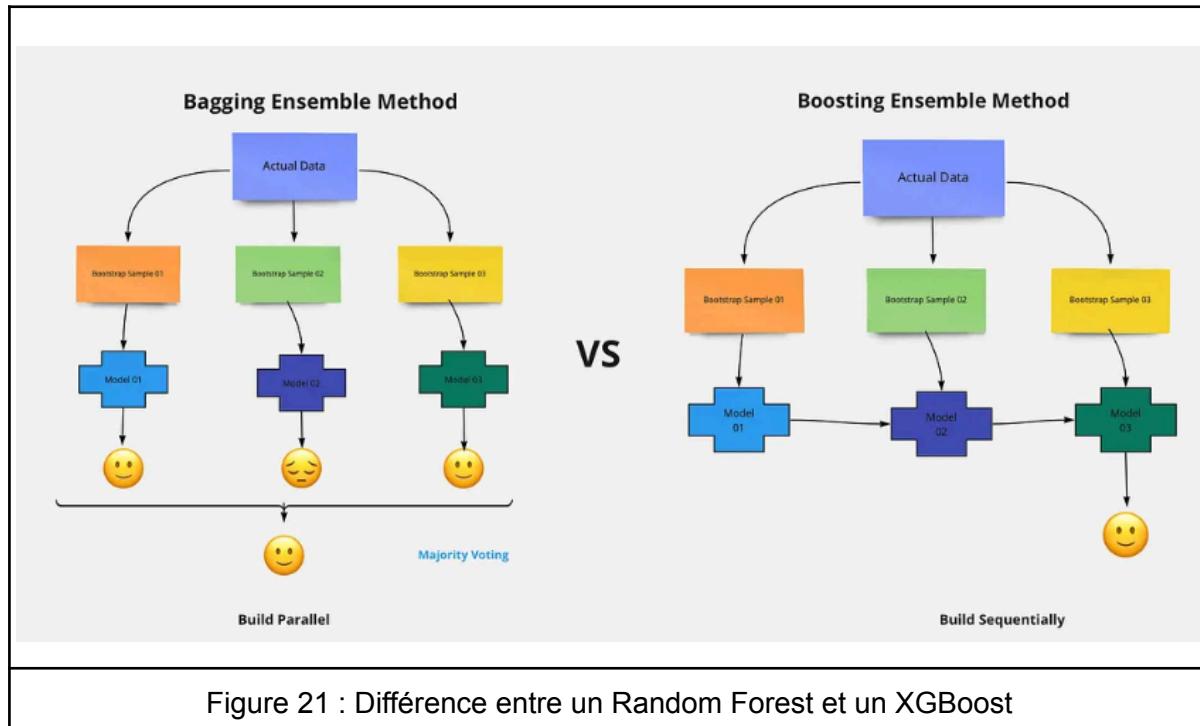
### ii. XGBoost

La présentation de XGBoost, eXtreme Gradient Boosting peut être réalisée à partir d'un article écrit par ActuIA, magazine et un média en ligne spécialisé dans l'intelligence artificielle. XGBoost est un algorithme plus récent que le Random Forest Regressor puisqu'il est apparu en 2016 dans une publication de Tianqi Chen et Carlos Guestrin[Citations Ch.12].

#### Concept :

L'idée principale de l'apprentissage avec XGBoost est d'entraîner de façon séquentielle plusieurs modèles et de les combiner successivement en corrigeant itération après itération les erreurs afin d'obtenir le modèle d'ensemble le plus puissant possible. Le résultat de la prédiction est donc constitué de la prédiction de l'ensemble des arbres de décision enchaînés. Cette méthode augmente les performances et la stabilité du modèle tout en minimisant sa variance.

La figure 21, tirée d'un article rédigé par Aman Gupta (lien disponible dans le chapitre 11.Bibliographies) Manager d'un groupe d'experts en IA sur LinkedIn et auteur de plusieurs articles sur medium.com, permet de visualiser la différence entre Random Forest et XGBoost :



### iii. Réseau de neurones

La vidéo réalisée par Machine Learnia permet de faire une bonne introduction aux réseaux de neurones artificiels (lien vers la vidéo dans le chapitre 11.Bibliographies).

## Historique :

Les premiers neurones artificiels ont été créés en 1943 par deux mathématiciens et neuroscientifiques Warren McCulloch et Walter Pitts[Citations Ch.12] en s'inspirant des neurones biologiques. Ces neurones ne pouvaient traiter que des entrées logiques.

Plus tard en 1957 un psychologue Américain, Frank Rosenblatt[Citations Ch.12], l'inventeur du perceptron, proposa le premier algorithme de l'histoire du deep learning ou apprentissage profond.

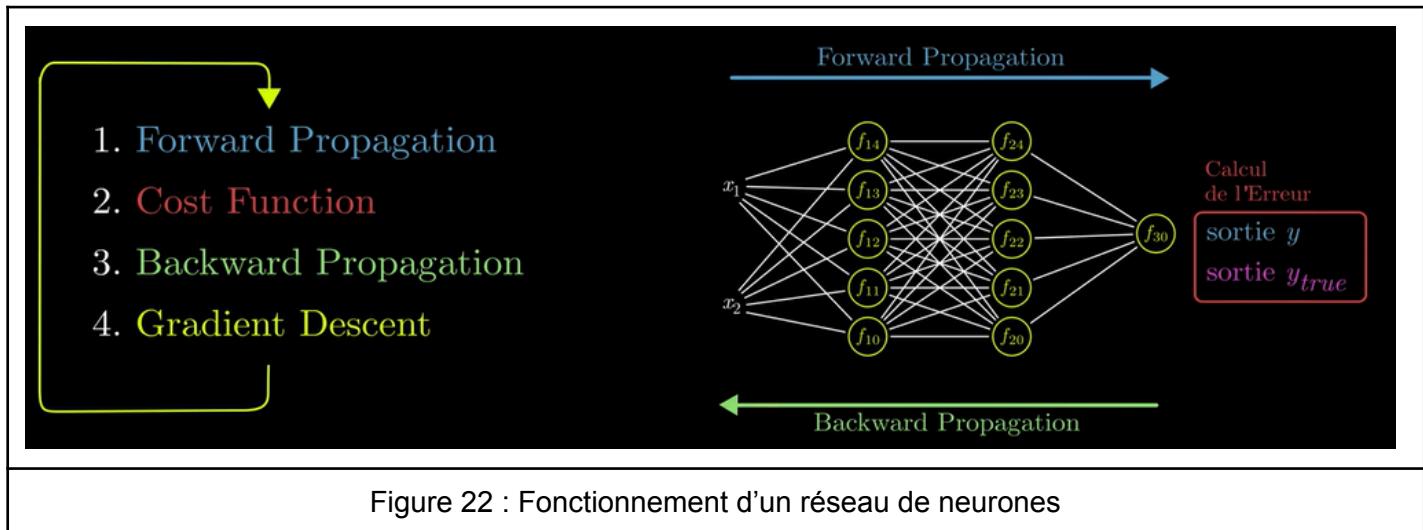
En 1986, Geoffrey Hinton inventa le perceptron multicouche[Citations Ch.12], le premier véritable réseau de neurones artificiels capable de traiter les problèmes non linéaires.

Dans les années 90 sont apparues les premières variantes du perceptron multicouches. Les réseaux de neurones convolutifs, par Yann LeCun[Citations Ch.12], permettant de reconnaître et le traitement d'images. Par la suite, les réseaux de neurones récurrents permettant de traiter les problèmes de séries temporelles, la lecture de textes ou encore la reconnaissance vocale.

L'intelligence artificielle telle qu'on la connaît aujourd'hui a été permise grâce à l'émergence d'internet, des smartphones et du nombre croissant de données disponibles.

Le second facteur est l'évolution de l'informatique et sa puissance de calcul qui est nécessaire pour entraîner des modèles de réseaux de neurones.

### Fonctionnement d'un réseau de neurones :



1. Forward Propagation : On fait circuler les données de la première couche jusqu'à la dernière afin de produire une sortie  $y$ .
2. Cost function : Calculée  $y$  et la réalité  $y_{true}$  en utilisant la fonction coût vu au chapitre 4.a.ii.
3. Back Propagation : On mesure comment chaque couche varie par rapport à chaque couche de notre modèle en partant de la dernière en remontant jusqu'à la toute première
4. Gradient Descent : On corrige chaque paramètre du modèle grâce à l'algorithme de descente de gradient vu au chapitre 4.a.iii.

### En pratique :

Pour entraîner un réseaux de neurones nous avons la possibilité de :

- Définir le nombre de couches à donner à notre modèle
- Paramétrer un nombre d'époques (une époque ou epoch est un cycle vu précédemment dans le "Fonctionnement d'un réseau de neurones")
- Configurer des arrêts anticipés (early stopping) permettant au modèle d'arrêter de s'entraîner pour prévenir du surajustement
- Prévoir un changement du critère alpha vu dans le chapitre 4.a.iii sur la descente de gradient, pour réguler le processus d'optimisation et améliorer les performances du modèle.
- ...

### **c. Conclusion**

Le dataset concernant le projet d'estimation de bien immobilier, qui est un problème de régression linéaire multiple, dispose de suffisamment de données pour être entraîné sur les trois modèles vus précédemment. Il sera intéressant de comparer les performances de ces modèles en s'appuyant sur les différentes métriques vu au chapitre 4.a.iv. Un choix sera

---

ensuite effectué pour appliquer ce modèle à l'application en fonction des performances et de la rapidité de prédition.

## 5. Les modèles

### a. Choix des modèles et méthodologie

Suite à l'analyse exploratoire de données ainsi qu'à la réalisation de l'état de l'art, la création et la recherche de performances des modèles va suivre le schéma suivant :

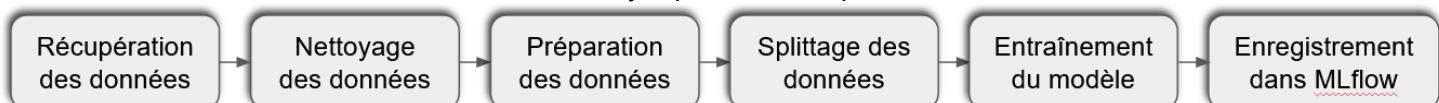
1. Pour chaque technologie, 3 modèles seront créés : uniquement avec les maisons, uniquement avec les appartements, avec les deux types de bien. Dans un premier temps, le modèle sera entraîné sur l'ensemble des données.
2. La même méthode que le point 1 mais avec période d'entraînement plus réduite (exemple la dernière année) afin que le modèle ne voit que les prix les plus récents.
3. Dans un dernier temps des variables seront ajoutées au modèle le plus performant entre le point 1 et 2 pour essayer d'améliorer le modèle sans le sur-entraîner.

Ce schéma n'est pas fixe et peut évoluer en fonction des données et des résultats. Si les métriques de performances sont satisfaisantes alors il sera inutile de continuer à entraîner un modèle et si pour un modèle les données ne sont pas suffisantes alors le modèle ne sera pas entraîné (exemple si une région ne dispose pas beaucoup de vente d'appartement sur la dernière année écoulée).

Les comparaisons des performances des modèles seront effectuées à l'aide de MLflow<sup>12</sup> dont l'utilisation et le déploiement sera expliqué en chapitre 6.d.

### b. Entraînement des modèles

Avant d'entraîner un modèle il y a plusieurs étapes à réaliser :



Les étapes de récupération et de nettoyage des données peuvent se faire en une seule étape dans la requête SQL en indiquant :

- Les variables nécessaires
- Le nombre de pièces minimum voulu
- Le prix maximum et minimum d'un bien comme nous avons pu le voir dans l'analyse exploratoire des données.

La préparation des données consiste à rendre les données exploitables pour l'entraînement d'un modèle. Il faudra donc labelliser les variables catégorielles puis standardiser<sup>13</sup> toutes les variables afin de les mettre à la même échelle.

<sup>12</sup> **MLflow** est une plateforme open source fournissant des outils pour suivre, gérer et déployer des modèles, ainsi que pour expérimenter différentes approches de modélisation.

<sup>13</sup> **Standardiser** consiste à une mise à l'échelle des données pour les rendre comparables et cohérentes les unes des autres afin de garantir que les algorithmes de machine learning traitent les différentes caractéristiques de manière équitable et efficace, sans être influencés par l'échelle initiale des données.

Pour le splitting des données, le prix de l'immobilier étant en hausse régulière depuis plusieurs années, le jeu de données sera trié par ordre croissant. Le jeu d'entraînement représente 80% des ventes les plus anciennes et le jeu de test les 20% restants.

### i. Récupération des données

La récupération des données se fait par une requête SQL(Figure 23). La création d'un algorithme a été créé afin de formater plus facilement la requête et d'en extraire les données voulues(figure 24). Les paramètres configurables sont :

- La région : Permet d'indiquer ou non une région spécifique.
- Le type de bien : Permet d'indiquer ou non un type de bien.
- Le nombre de ventes minimum : Le nombre de ventes minimum par commune.
- La surface du terrain : La surface de terrain minimum.
- Le nombre de mois : Le nombre de mois voulu pour réaliser la création du modèle. Les mois récupérés sont les mois les plus récents de la base de données.

```
# Table pour compter le nombre de ventes par commune
WITH nb_ventes_mini AS(
SELECT
    ID_COMMUNE AS ID_COMMUNE,
    count(*) nb_ventes_par_commune
FROM VENTES
WHERE DATE_MUTATION >= DATE_SUB((SELECT MAX(DATE_MUTATION) FROM VENTES), INTERVAL 15 MONTH)
GROUP BY ID_COMMUNE
)

# Selection des variables voulues pour l'entraînement du modèle
SELECT
    V.SURFACE_BATI,
    V.ID_COMMUNE,
    V.DATE_MUTATION,
    # T.NAME_TYPE_BIEN,
    # R.Name_region,
    V.SURFACE_TERRAIN,
    V.MONTANT
FROM VENTES V
INNER JOIN TYPES_BIENS AS T ON V.ID_TYPE_BIEN = T.ID_TYPE_BIEN
INNER JOIN COMMUNES AS C ON V.ID_COMMUNE = C.ID_COMMUNE
INNER JOIN DEPARTEMENTS D ON C.ID_DEPT = D.ID_DEPT
INNER JOIN REGIONS R ON D.ID_REGION = R.ID_REGION
WHERE R.Name_region NOT IN('Martinique', 'Guyane', 'La Réunion', 'Mayotte', 'Guadeloupe')
    AND T.Name_Type_Bien='Maison'
    AND V.DATE_MUTATION >= DATE_SUB((SELECT MAX(DATE_MUTATION) FROM VENTES), INTERVAL 15 MONTH)
    AND V.MONTANT>15000 AND V.MONTANT<6500000
    AND V.SURFACE_BATI>0
    AND V.NB_PIECES>0
    AND V.ID_COMMUNE IN (
        SELECT |
            ID_COMMUNE
        FROM nb_ventes_mini
        WHERE nb_ventes_par_commune>=10)
```

Cette sous-requête permet de compter le nombre de ventes par communes sur la période donnée.

Variables qui seront récupérées .

Jointure entre les différentes tables.

Filtrage des données :

- montants, surface bâti et le nombre de pièces qui sont paramétrés par rapport au résultat de l'EDA.
- Les autres filtres sont ajoutés par l'algorithme.

Figure 23: Requête pour la récupération de données.

	SURFACE_BATI	ID_COMMUNE	DATE_MUTATION	SURFACE_TERRAIN	MONTANT
0	68	01004	2022-07-22	0	155000
1	80	07042	2022-08-31	18679	143000
2	104	62065	2022-06-21	1869	93285
3	90	62041	2022-06-18	110	159000
4	41	62839	2022-06-09	1659	105000

Figure 24 : Aperçu du résultat de la requête

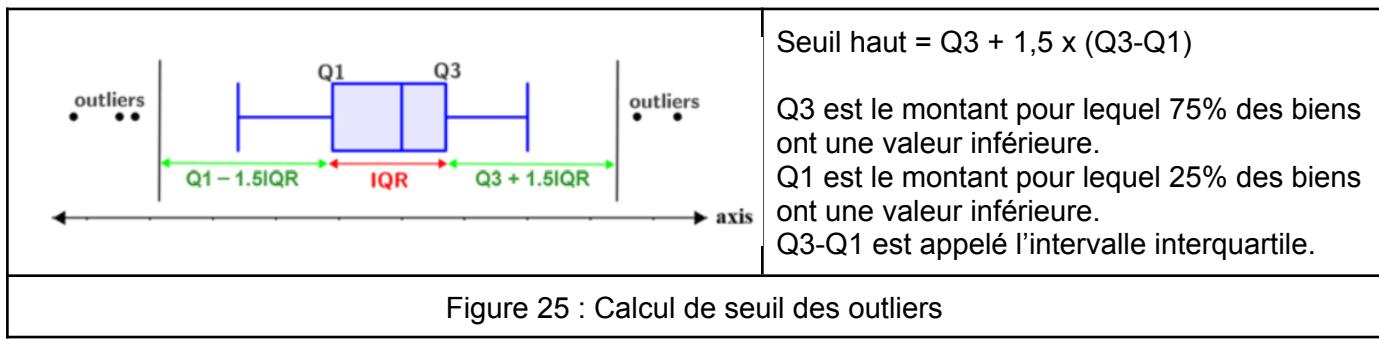
## ii. Nettoyage de données

Le nettoyage de données dans le processus de la création du modèle d'intelligence artificielle consiste à supprimer les valeurs aberrantes, dit outliers. Dans notre problème d'estimation d'un bien immobilier, les outliers sont des valeurs nettement supérieures à la moyenne nationale. Pour la gestion des outliers un algorithme a été créé. Cet algorithme prend en entrée :

- Le dataset créé dans le chapitre précédent
- Une variable tx\_filtrage prenant deux arguments. Le premier étant le nombre de ventes minimum que l'on désire conserver après retrait des outliers et le deuxième étant le pourcentage d'outliers dans une commune pour lequel on ne conservera pas la commune dans l'entraînement du modèle.
- Le dernier paramètre étant la partie de la requête du chapitre précédent permettant le filtrage, la clause WHERE.

Le déroulement de l'algorithme est le suivant :

- 1) Le calcul du montant pour lequel la valeur d'un bien est considéré comme une valeur aberrante. Ce calcul de seuil (illustré dans la figure 25) a été réalisé à l'aide du dataset et de la méthode de la marge interquartile.



- 2) La deuxième étape consiste à créer un second dataset à l'aide d'une requête SQL, en utilisant le seuil pour les outliers ainsi que la clause WHERE précédente. Ce nouveau dataset contiendra le nombre de ventes restantes après suppression des outliers ainsi que le pourcentage de ventes retirées par commune. La figure 26 montre un aperçu du data frame créé.

ID_COMMUNE	nb_outliers	total_ventes	ventes_restantes	NAME_COMMUNE	pourcentage_ventes_retirees
33063	1108	6797	5689	Bordeaux	16.30
06088	1059	11579	10520	Nice	9.15
92051	943	1364	421	Neuilly-sur-Seine	69.13
92012	932	2461	1529	Boulogne-Billancourt	37.87
06029	778	4103	3325	Cannes	18.96

Figure 26 : Dataset montrant l'impact du retrait des outliers

- 3) La dernière étape va utiliser la variable tx\_filtrage qui pour rappel contient 2 arguments, nombre de ventes et taux de filtrage. Par exemple, si tx\_filtrage=[10,30], les communes conservées seront les communes dont il restera minimum 10 ventes et moins de 30% d'outliers.

L'exécution de cet algorithme à donc permis de retirer les ventes considérées comme valeurs aberrantes, les communes contenant un taux d'outliers importants et de retirer les communes ayant moins de 10 ventes restantes après le retrait des outliers. L'algorithme permet également l'affichage de l'impact final sur cette gestion des outliers comme le montre la figure 27.

```
Nombre d'outliers : 60157
Nombre de commune avec des outliers : 5680
Nombre de communes contenant des outliers et pour lesquels ilreste plus de 10 ventes après suppression des outliers : 4526
Nombre de communes avec plus de 30% de ventes retirées : 641
Nombre de communes qui seront retirées : 1415
Nombre de ventes restantes après suppression des outlier et après filtrage : 905178
Il y a donc eu 8.47% de lignes supprimées après filtrage
```

Figure 27 : Impact final du retrait des outliers

### iii. Splittage des données

Pour la séparation en jeux d'entraînement et de test la solution était de trier le dataset par ordre chronologique, de prendre les 80% des premières lignes pour l'entraînement et les 20% restants pour le test. Ainsi on simule la réalité en entraînant sur des ventes passées, les données de tests représentants les données futures.

On va également s'assurer que les communes présentes dans les données de test le sont aussi dans les données d'entraînement afin que la normalisation des données puisse être effectuée.

### iv. Préparation des données

La première chose à faire dans la préparation de données est de créer la variable du prix au mètre carré par commune. Le prix au mètre carré par commune a été réalisé en calculant le prix au mètre carré de chaque vente dans les données d'entraînement, puis, un dataframe à été créé en groupant les données par identifiant commune et en faisant la moyenne du prix au mètre carré.

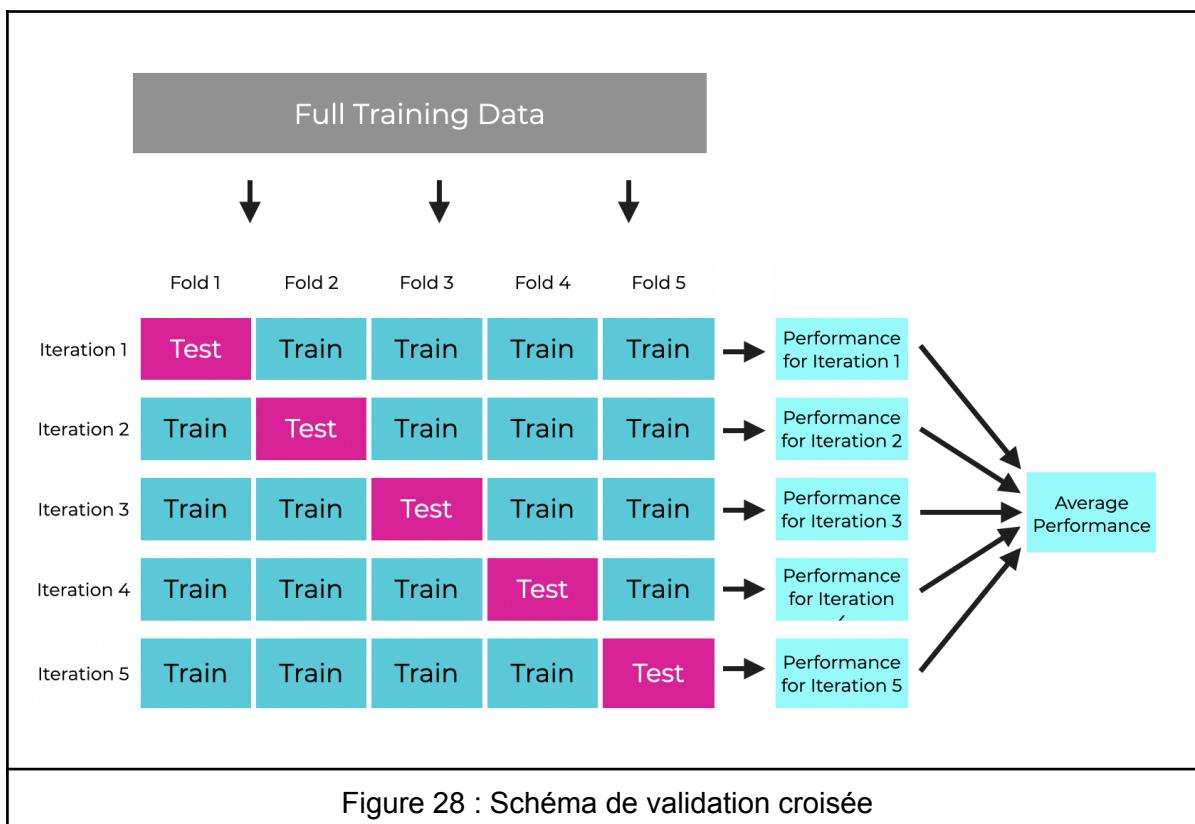
Ensuite les données d'entraînement sont dans un premier temps labellisées s'il existe des variables catégorielles puis standardisées afin de mettre toutes les variables sur la même échelle.

Enfin on va utiliser le data frame avec le prix au mètre carré par commune réalisé avec les données d'entraînement pour l'appliquer aux données de test puis appliquer la même normalisation.

### v. Entraînement d'un modèle

Après différents tests sur les trois algorithmes présentés dans la veille technique (RandomForest, XGBoost et un modèle de réseaux de neurones), la méthode choisie est celle de XGBoost pour sa rapidité d'exécution et ses performances.

La méthode utilisée pour entraîner cet algorithme est Grid Search CV de scikit learn. Cette méthode permet d'entraîner un modèle sur différents hyperparamètres en utilisant la validation croisée (cross validation). La validation croisée permet d'obtenir une estimation plus fiable des performances du modèle sur des données non vues comme le montre la figure 28.



Une fois tous les entraînements réalisés, il est possible de récupérer les paramètres pour lesquels le modèle a obtenu les meilleures performances et de réentraîner ce modèle avec ces paramètres . Cependant il est important de s'assurer que le modèle ne soit pas en sur-apprentissage (overfitting) ou sous-apprentissage (underfitting). Un phénomène de sur-apprentissage arrive lorsque le modèle s'adapte trop aux données d'entraînement et ne généralise pas bien ce qui provoque de mauvaises prédictions sur les données de test non vues. Ces phénomènes sont illustrés figure 29.

Pour s'assurer que le modèle ne soit pas en sur-apprentissage il est possible de tracer une courbe de validation pour s'assurer que les performances de test sont en corrélation avec les performances de test.

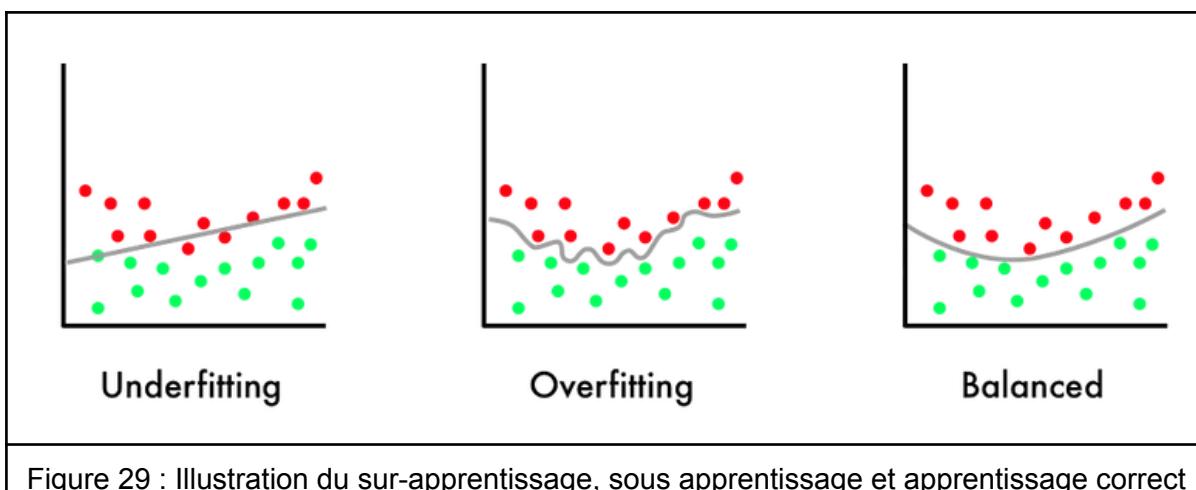


Figure 29 : Illustration du sur-apprentissage, sous apprentissage et apprentissage correct

### c. Comparaisons des modèles

Les différents tests effectués ont montré qu'une durée plus courte avec les variables sur le prix au mètre carré, la surface du bien ainsi que celle du terrain ont apporté de meilleures performances.

Le monitoring avec MLFlow permet de visualiser les différentes métriques entre les différents modèles ainsi que de visualiser les courbes qui y ont été stockées. Le graphique figure 30 montre les métriques de 4 modèles.

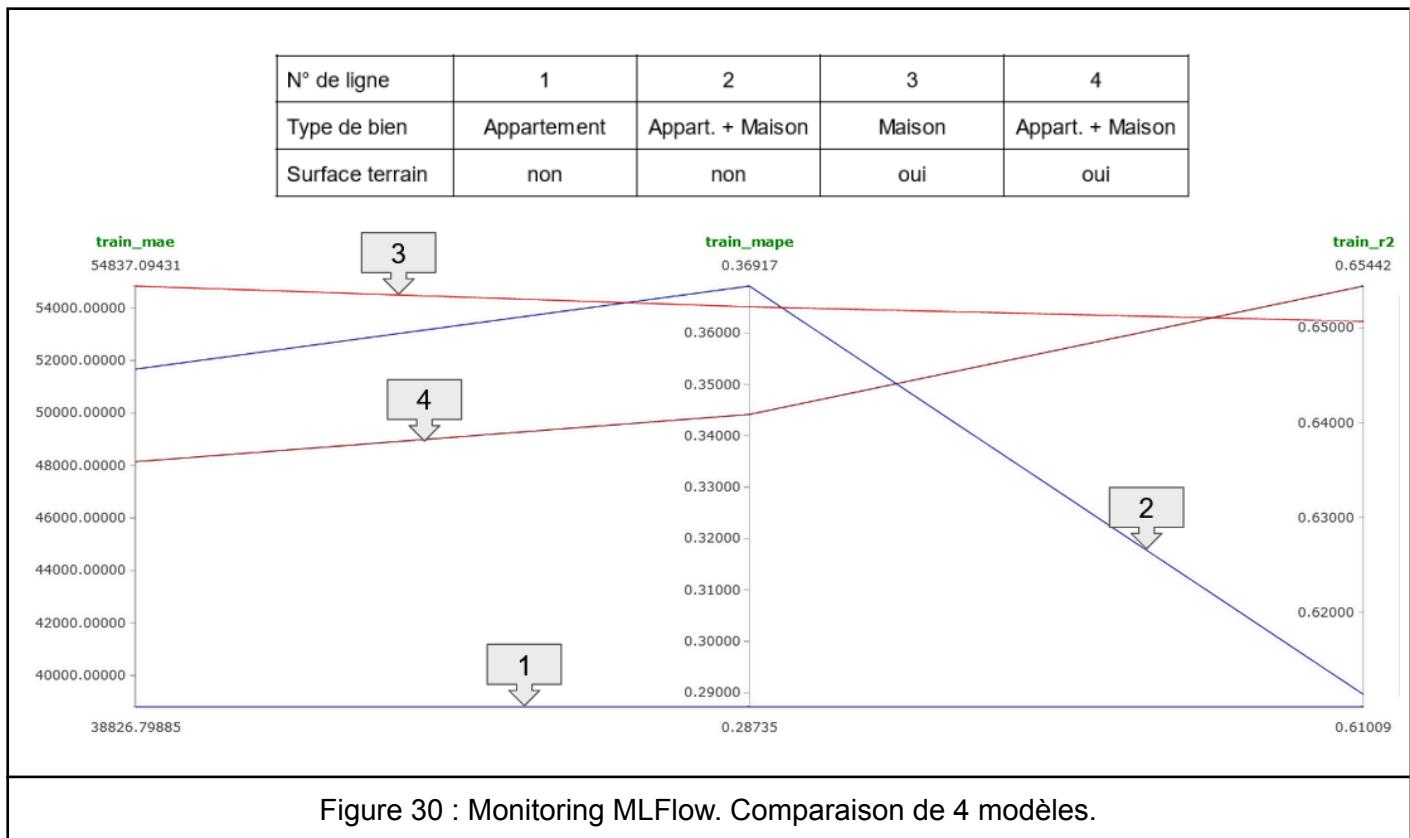


Figure 30 : Monitoring MLFlow. Comparaison de 4 modèles.

La conclusion que l'on peut faire de la figure 30 est que le modèle ayant le plus bas écart de prix absolu moyen ainsi qu'un pourcentage d'écart moyen le plus faible est le modèle entraîné sur les appartements uniquement (ligne 1). Malheureusement le modèle entraîné sur les maisons (ligne 3) n'obtient pas les mêmes performances avec une erreur moyenne absolue et un pourcentage d'erreurs beaucoup plus élevé. Le modèle retenu sera donc celui entraîné avec les maisons et les appartements ainsi que la surface du terrain (ligne 4). Ce modèle est le plus résilient avec une erreur moyenne de 48 000€, un pourcentage d'erreur de 34,4% et un r2 score de 0,65.

Avant d'implémenter ce modèle dans l'application, la courbe de validation va nous permettre de repérer si le modèle ne contient pas de sous ou sur-ajustement. La figure 31 nous montre qu'au-delà d'une quarantaine d'estimateurs le modèle commence à légèrement sur-ajuster car le r2 score diminue légèrement au-delà de cette zone.

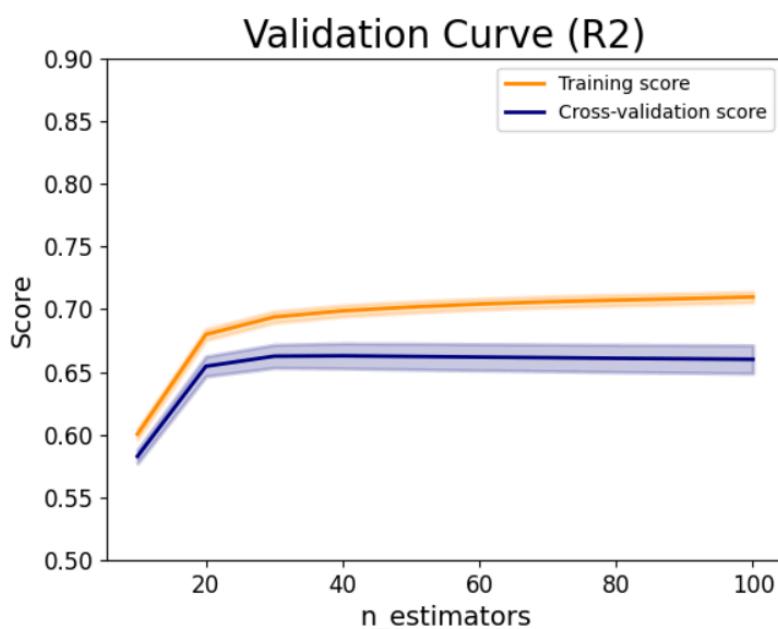
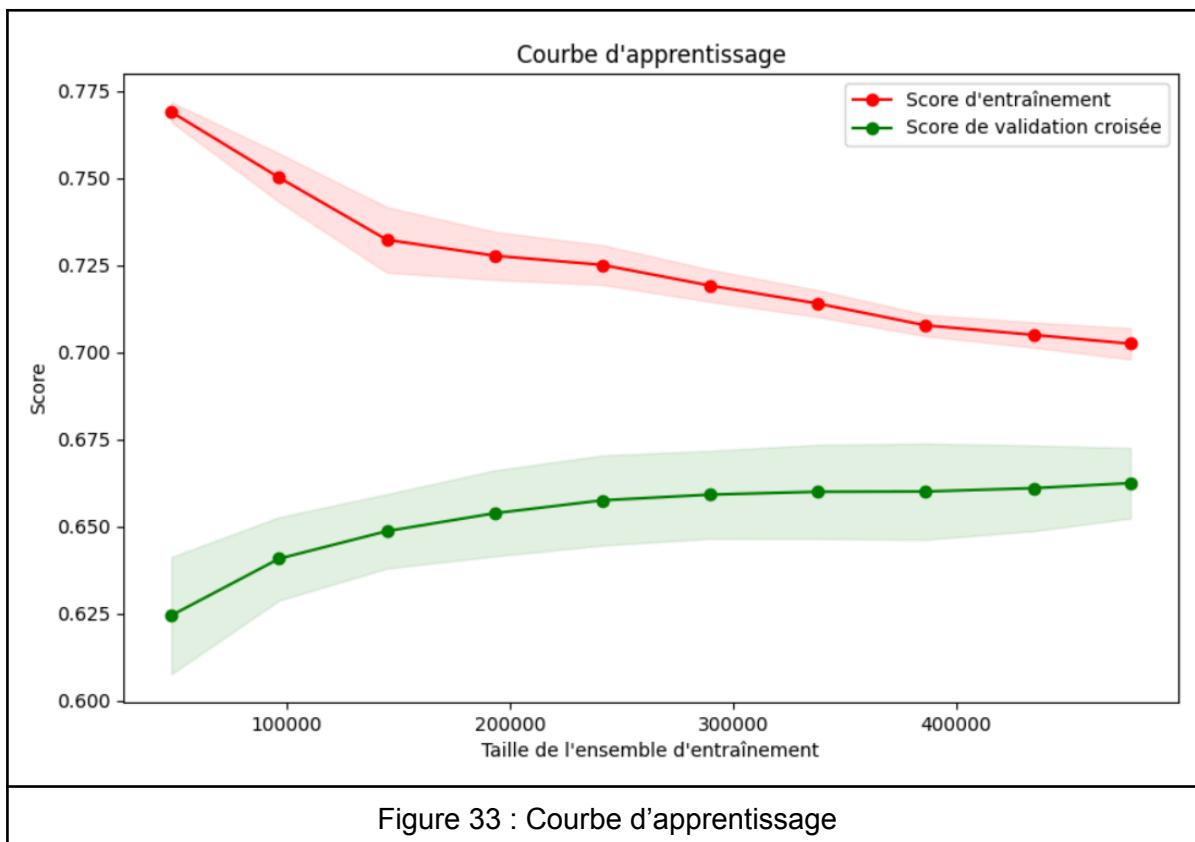
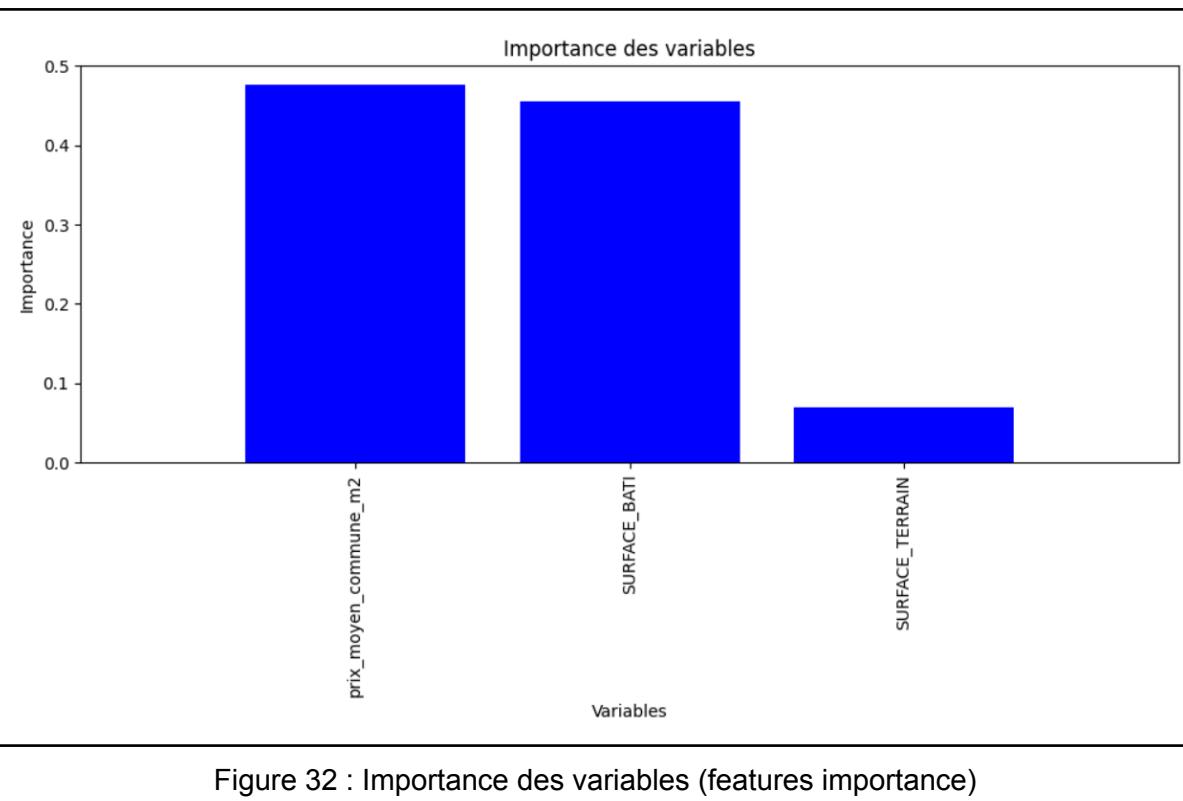


Figure 31 : Courbe de validation (validation curve)

Le graphique figure 32 montrent l'importance des variables dans la détermination du prix par le modèle et nous indique que le prix moyen au mètre carré ainsi que la surface du bâti sont prédominantes et que la surface du terrain n'a qu'environ 8% d'importance. Malgré cette prédominance faible, cette variable est utile dans l'amélioration des performances comme on peut le remarquer dans la figure 30 entre les lignes 2 et 4 où le simple fait d'ajouter cette variable, permet de faire diminuer la moyenne de l'erreur absolue et d'augmenter le r2 score.

La figure 33 illustrant la courbe d'apprentissage nous permet d'affirmer que le modèle dispose de suffisamment de données. On remarque sur ce graphique qu'à partir d'environ 300 000 échantillons les performances du modèle n'évoluent plus.



## 6. Mise en place de l'environnement

### a. *Outils et utilisation pour l'intégration et déploiement continu (CI/CD)*

L'intégration et le déploiement continu, dans la création de l'application et de l'API, a consisté à vérifier automatiquement un code lorsqu'il était déposé sur un dépôt sources GitHub puis à déployer sur Heroku lorsque les tests sont corrects. Pour la mise en place de cette pratique, plusieurs étapes étaient nécessaires.

#### i. Création du fichier test

Les tests ont été réalisés avec unittest, qui est une bibliothèque python qui permet d'écrire des tests unitaires pour vérifier le bon fonctionnement des fonctions, méthodes ou classes dans un programme. Les fonctions testées, figure 34, pour notre application étaient :

- Le bon fonctionnement de la connexion à la base de données RDS d'amazon contenant toutes les ventes.
- Le retour de l'API renvoie à ce qui est attendu.

```
import unittest
import os
from functions import api_predict,create_connection

class TestAPICalls(unittest.TestCase):
    def test_create_connection(self):
        # Les informations de connection doivent être renseigné dans le dépôt GitHub
        host = os.environ['DB_HOST']
        user = os.environ['DB_USER']
        password = os.environ['DB_PASSWORD']
        port = int(os.environ['DB_PORT'])
        database = "datagouv"

        # Appel de la fonction à tester
        conn = create_connection(host, user, password, port, database)

        # Vérification que le cursor n'est pas None (indicatif d'une connexion réussie)
        self.assertIsNotNone(conn)

        # Fermeture de la connexion après les tests
        conn.close()

    def test_api_predict_success(self):
        # Appel de la fonction de l'API avec des données type
        data = {"SURFACE_BATI": 150,
                "SURFACE_TERRAIN": 750,
                "prix_moyen_communne_m2": 1356}
        result = api_predict(data)

        # Vérification si le résultat est un dictionnaire
        self.assertEqual(dict, type(result))

        # Vérification que la clé 'reponse' est présente dans le JSON
        self.assertIn('reponse', result)

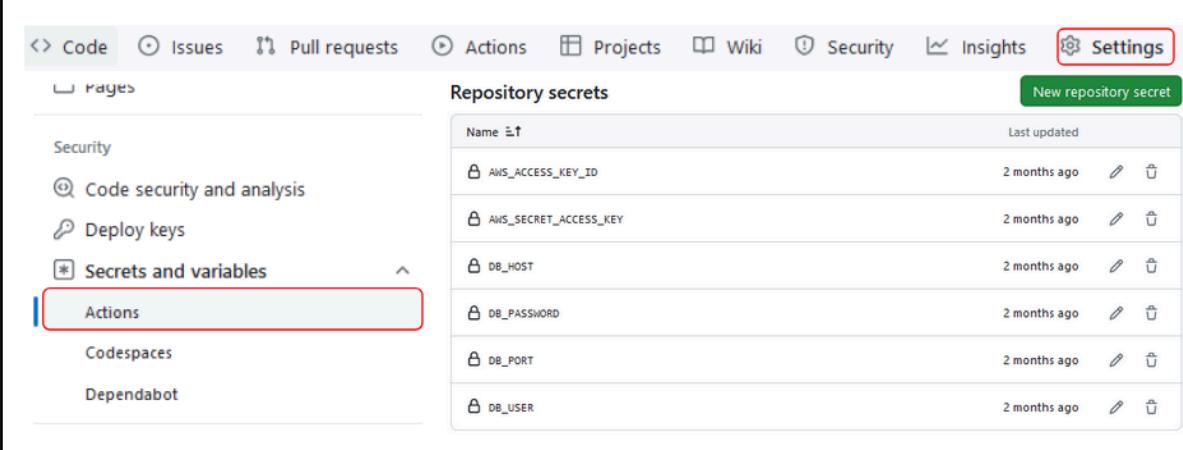
        # Vérification la valeur de la clé 'reponse' est de type float
        self.assertIsInstance(result['reponse'], float)

if __name__ == '__main__':
    unittest.main()
```

Figure 34 : Code pour les tests unitaires

## ii. Configuration dans GitHub

Pour que les tests unitaires puissent être exécutés dans GitHub, deux étapes étaient nécessaires. La première consiste à renseigner les informations de connexions dans la base de données. Les informations de connexions sont stockées dans des variables d'environnement que l'on peut configurer dans l'onglet “Setting”, “Security”, “Secrets and variables” comme l'illustre la figure 35.



The screenshot shows the GitHub interface for managing repository secrets. The top navigation bar includes Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The Settings tab is highlighted with a red border. On the left, a sidebar lists Pages, Security (Code security and analysis, Deploy keys, Secrets and variables), Actions (highlighted with a red border), Codespaces, and Dependabot. The main content area is titled "Repository secrets" and displays a table of environment variables:

Name	Last updated
AWS_ACCESS_KEY_ID	2 months ago
AWS_SECRET_ACCESS_KEY	2 months ago
DB_HOST	2 months ago
DB_PASSWORD	2 months ago
DB_PORT	2 months ago
DB_USER	2 months ago

At the bottom of the screenshot, the caption "Figure 35 : Variables d'environnement sur GitHub" is visible.

La deuxième étape étant de créer un workflow<sup>14</sup> pour indiquer à GitHub les étapes à suivre lors du dépôt d'un nouveau code. Ce workflow doit être placé impérativement avec le chemin “.github/worflow/mon\_fichier.yml”. Ce Fichier va permettre de définir un environnement dans lequel seront exécutés les tests unitaires, de nommer les étapes à effectuer afin de retrouver facilement leurs traces dans GitHub Actions ainsi que de lui spécifier le chemin où se trouve le fichier de test.

## iii. Configuration dans Heroku et fonctionnement

Pour que le déploiement se fasse automatiquement, il faut indiquer à Heroku la méthode pour déployer l'application. Pour réaliser ce déploiement automatique, deux étapes sont nécessaires.

La première consiste à lui indiquer l'endroit de notre dépôt, lui autoriser la connexion et enfin à lui demander d'attendre que les tests d'intégration réussissent avant de déployer l'application. La figure 36 permet de voir cette configuration.

La seconde chose à faire est d'indiquer à Heroku comment il doit déployer l'application. Pour réaliser cette étape, il suffit de créer un fichier Heroku.yml à la racine du dépôt dans laquelle les fichiers de l'application sont stockés. Ce fichier heroku.yml, indique qu'il faut construire l'application à l'aide d'un Dockerfile ainsi que le chemin de ce dernier.

Un Dockerfile est un ensemble d'instructions pour créer une image Docker, qui encapsule à la fois l'application et ses dépendances dans un conteneur, permettant un déploiement portable et cohérent sur diverses machines sans qu'un utilisateur ait besoin de

<sup>14</sup> Workflow est une manière systématique de gérer et d'accomplir des tâches en suivant un ensemble prédéfini de règles ou de directives.

faire de mises à jour. La figure 37 montre les différentes étapes pour la création d'une image Docker<sup>15</sup>.



The screenshot shows the Heroku deployment configuration interface. At the top, there are three sections: 'Deployment method' (Heroku Git, GitHub Connected, Container Registry), 'App connected to GitHub' (connected to rastakoer/certif\_app\_immo by rastakoer, with options to disconnect or view activity feed), and 'Automatic deploys' (enabling automatic deploys from application branch). A note says: 'You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions [here](#)'. Below that, under 'Automatic deploys', there are two checkboxes: 'Automatic deploys from application are enabled' (checked) and 'Every push to application will deploy a new version of this app. Deploys happen automatically: be sure that this branch in GitHub is always in a deployable state and any tests have passed before you push. [Learn more](#)' (unchecked). The last checkbox is 'Wait for CI to pass before deploy' (unchecked).

Figure 36 : Configuration sur Heroku pour le déploiement continu

```
FROM continuumio/miniconda3

WORKDIR /home/app

# Install git
RUN apt-get update && apt-get install -y git

# Clone the repository
RUN git clone https://github.com/rastakoer/certif_app_immo.git --branch application .

# Copiez les fichiers nécessaires
COPY requirements.txt .

# Install Python dependencies
RUN pip install -r requirements.txt

# Copy the application code
COPY . /home/app

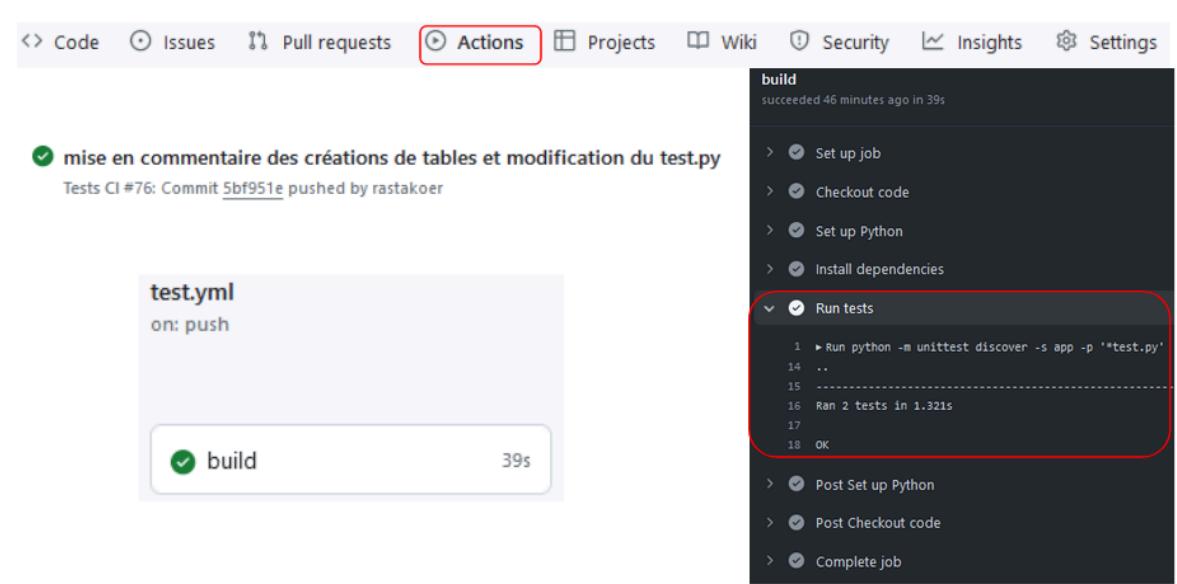
# Set the command to run on container start
CMD streamlit run app/app.py --server.port $PORT
```

Figure 37 : Construction de l'image Docker pour l'application

<sup>15</sup> Les images Docker sont créées à partir de fichiers appelés Dockerfiles et contiennent tout le nécessaire pour exécuter une application, y compris le code source, les bibliothèques, les dépendances, les fichiers de configuration, etc. C'est comme une boîte virtuelle prête à l'emploi qui peut être exécutée sur n'importe quel système compatible avec Docker, assurant ainsi une portabilité et une cohérence des environnements d'exécution logicielle.

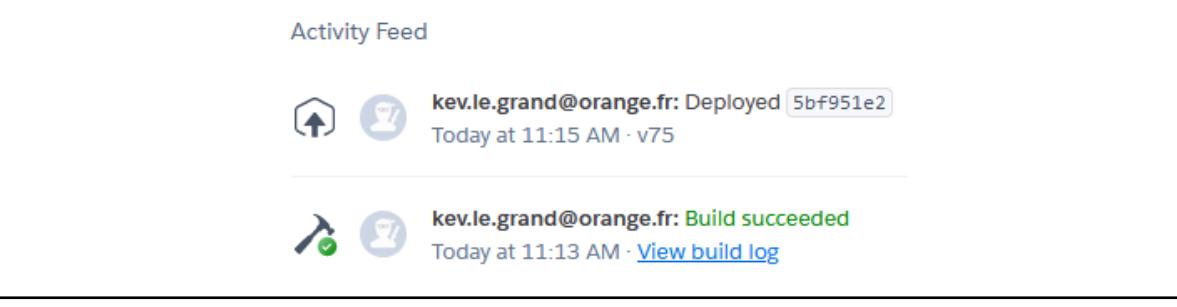
#### iv. Confirmation de l'intégration et du déploiement

Une fois cette configuration établie, il est possible de visualiser que les tests se sont bien déroulés dans GitHub Actions (figure 38) ainsi que le déploiement dans Heroku dans l'onglet Activity (figure 39).



The screenshot shows the GitHub Actions interface. The 'Actions' tab is selected. A green checkmark indicates a successful build named 'build' that completed in 39s. The build log details the steps: Set up job, Checkout code, Set up Python, Install dependencies, Run tests (which is expanded and highlighted with a red box), Post Set up Python, Post Checkout code, and Complete job. The 'Run tests' section shows command-line output for running unit tests.

Figure 38 : Affichage du test d'intégration continue



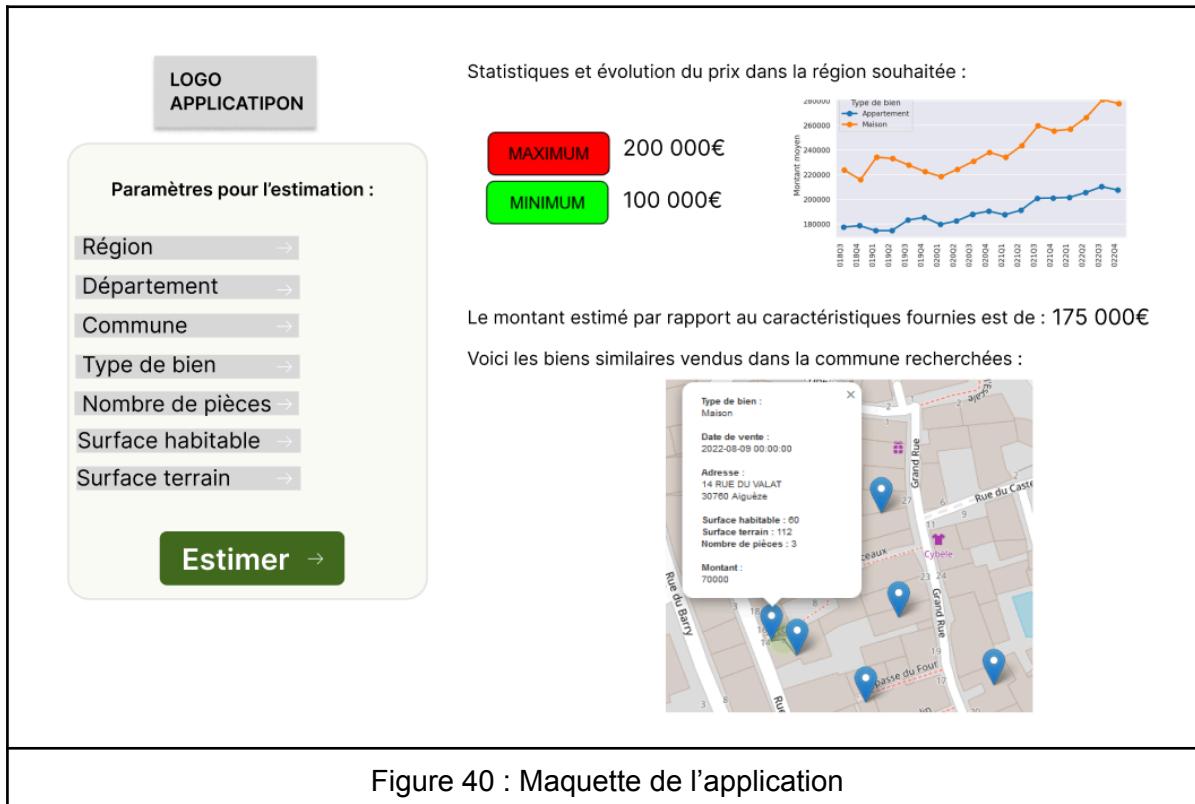
The screenshot shows the Heroku Activity Feed. It displays two entries: a deployment by 'kev.le.grand@orange.fr' at 11:15 AM which succeeded, and a build log entry by the same user at 11:13 AM which also succeeded. Both entries include a link to 'View build log'.

Figure 39 : Affichage de la confirmation du déploiement

#### b. Front-end

Pour la réalisation du frontend une maquette, figure 40, a été réalisée à l'aide du site Figma.com. Cette maquette montre approximativement le rendu que devra avoir l'application. Les outils utilisés seront :

- Pour le squelette de l'application, Streamlit, qui permet une construction simplifiée d'une application web, ce qui est largement suffisant pour les fonctionnalités que doit contenir l'application.
- La visualisation des biens vendus sera réalisée avec la librairie Folium de python qui permet un affichage dynamique d'une carte et de placer des popups en fonction de coordonnées GPS.
- L'affichage des statistiques se fera à l'aide d'une connexion à la base de données RDS en lecture seule (configuration IAM sur AWS).



Le résultat final de l'application permet d'afficher les statistiques sur les ventes dès que l'utilisateur fait un choix. La figure 41 représente la page lorsque l'utilisateur a saisi toutes les caractéristiques.



## c. Back-end

### i. Gestion des utilisateurs

Lors de la connexion à l'application, une demande de login est réalisée. Cette page permet de créer un nouvel utilisateur ou de se connecter si l'utilisateur possède déjà un compte. Lorsqu'un nouvel utilisateur renseigne son identifiant et son mot de passe, ceux-ci sont stockés dans une base de données, le mot de passe étant chiffré à l'aide de la librairie hashlib de python. L'administrateur peut ensuite donner l'autorisation à l'utilisateur d'accéder au dashboard de grafana en paramétrant son niveau dans la base de données. La figure 42 montre la base de données ainsi que les différences sur l'application entre deux utilisateurs de niveau différent.

base de données				
<b>id</b>	<b>username</b>	<b>password</b>	<b>level</b>	
5	kevin3	0f1b445587009b8a7dd2a4c1270f2f9a	1.0	
7	new	22af645d1859cb5ca6da0c484f1f37ea	NaN	

**Page de connexion**

**Login Immoapp**

**Connexion**

Nom d'utilisateur

Mot de passe

 (œil)

**Créer un nouvel utilisateur**

Nouveau nom d'utilisateur

Nouveau mot de passe

 (œil)

**utilisateur : kevin3**



Cette application web permet d'estimer les prix immobiliers dans la métropolitaine.

Le modèle a été entraîné sur une période de 12 à partir du 21/01/2020.

Il est possible que votre commune ne figure pas dans la liste :

- N'ont été retenues que les communes ayant eu plus de 1000 transactions.
- N'ont été retenues que les communes ayant moins de 300 habitants.

Grafana

**utilisateur : new**



Cette application web permet d'estimer les prix immobiliers dans la métropolitaine.

Le modèle a été entraîné sur une période de 12 à partir du 21/01/2020.

Il est possible que votre commune ne figure pas dans la liste :

- N'ont été retenues que les communes ayant eu plus de 1000 transactions.
- N'ont été retenues que les communes ayant moins de 300 habitants.

**Figure 42 : Gestion des utilisateurs**

ii.API

La création d'une API permet d'utiliser le modèle depuis n'importe quelle application ou d'être testé directement depuis le lieu où elle est hébergée. Lors d'un travail en équipe, l'utilisation d'une API permet une meilleure collaboration et un développement concurrentiel, car les équipes peuvent travailler sur l'API et l'application principale de manière indépendante, accélérant ainsi le processus de développement global. L'API de l'application a été développée avec la librairie python FastApi qui permet de documenter facilement ses fonctionnalités et déployée sur heroku (en CI/CD) permettant un accès facile. La figure 42 montre une capture d'écran lors d'un test de l'API.

The screenshot shows the FastAPI documentation for the 'predict' endpoint. At the top, it says 'API prédictions d'un bien immobilier' with version '0.1.0' and 'OAS 3.1'. Below that, it says 'openapi.json' and 'Api développée par Kevin LE GRAND'. A note states: 'Obtenez une prédiction de la valeur d'un bien grâce aux éléments suivants : - La surface d'un bien - La surface du terrain - Le prix au m<sup>2</sup> de la commune'. The main section is titled 'POST /predict Prédiction'. It says 'Route permettant de réaliser une prédiction'. Under 'Parameters', it says 'No parameters'. Under 'Request body required', it shows a JSON schema: { "SURFACE\_BATTU": 250, "SURFACE\_TERRAIN": 1000, "prix\_moyen\_commune\_m2": 2123.13 }. To the right, there's a table showing a response for code 200: 'Response body' containing { "reponse": 394082.6875 }.

Figure 43 : Capture d'écran FastApi

***d. Monitoring***

Le monitoring de l'application a été réalisé à l'aide de Grafana. Le monitoring permet:

- D'afficher les meilleures performances des modèles.
- D'afficher les KPI du site (nombre de prédictions mensuel, par type de biens...).
- D'envoyer des mails en fonction des alertes paramétrées.

Pour effectuer ce monitoring on se sert de la base de données de MLflow ainsi que des tables créées spécifiquement pour Grafana. Depuis l'application, on enregistre les

paramètres de chaque prédiction voulus par un utilisateur dans une table nommée "kpis", ainsi que chaque plantage dans la table nommée "crash" grâce à un try-except.

Dans l'outil de Grafana il suffit ensuite de requêter, selon les besoins, sur les tables de Mlflow, kpis ou crash afin de produire un module. L'ensemble des modules permet de fournir un tableau de bord (figure 44). Il est possible ensuite de configurer une alerte mail si un dysfonctionnement a lieu sur l'application (figure 45) ou sur n'importe quelle autres mesures présentent dans les bases de données. Il serait possible de créer une alerte si le nombre d'estimations dépasse un certain seuil, si les performances d'un modèle s'améliorent...



figure 44 : Tableau de bord permettant le monitoring de l'application avec Grafana

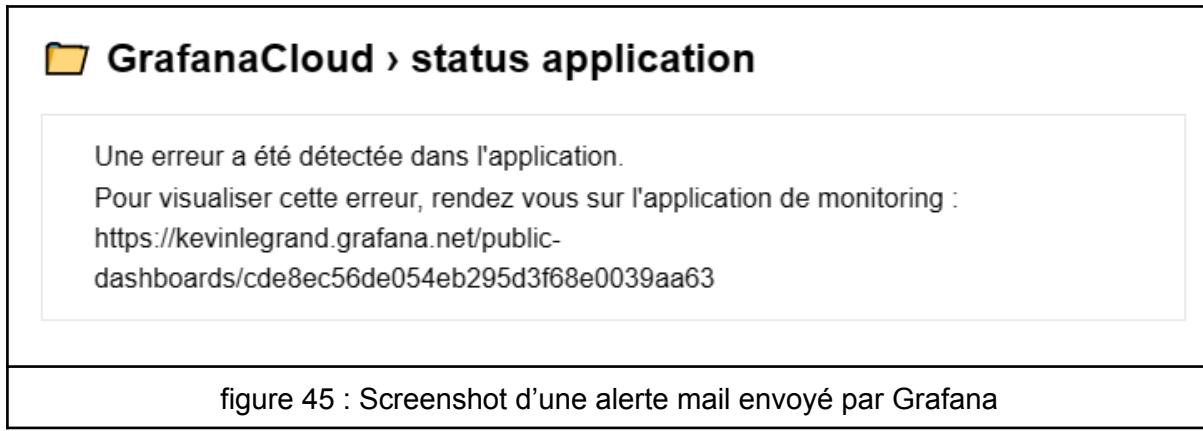


figure 45 : Screenshot d'une alerte mail envoyé par Grafana

## 7. Conclusion

### a. La réalisation du projet

Ce projet était très enrichissant par la diversité des tâches à accomplir. Il m'a permis d'appliquer un grand nombre des compétences acquises durant cette formation :

- Gestion de projet (diagramme de Gantt plus Kanban sur Trello)
- web scraping.
- Analyse de données.
- Création et gestion de bases de données.
- Hébergements sur services cloud (AWS : S3, RDS...).
- La programmation orientée objet avec la création de classes et de fonctions.
- Création et recherche de performances de modèles d'intelligence artificielle.
- Déploiement et intégration continue à l'aide de Github et Heroku.
- Création d'une API (fastapi).
- Création d'une application(streamlit).
- Monitoring des modèles créés avec MLflow.
- Monitoring d'application avec Grafana.

La réalisation de ce projet de bout en bout m'a permis de me rendre compte de l'intérêt de tous les outils utilisés qui facilitent grandement le travail en équipe, de la vision du projet et de son avancement ainsi que les méthodes efficaces pour déployer une application contenant de l'intelligence artificielle.

Pour finir, ce projet est la conclusion de plusieurs mois de formation et il permet de mesurer le chemin parcouru et le résultat d'un travail intensif partant de pratiquement zéro en Octobre 2022.

### b. Les difficultés rencontrées

La première difficulté était de trouver un projet avec des données disponibles et pouvant remplir toutes les compétences demandées pour la certification ainsi que les compétences acquises durant la formation et l'alternance.

Une fois le projet et les données trouvées, la plus grande difficulté de ce projet était de maintenir les délais. Ce projet n'étant pas un projet d'alternance, ayant pratiquement quatre heures de transport par jour pour me rendre en formation ou sur mon lieu de travail et mon fils en garde alternée, ce projet a donc été réalisé pendant les congés payés.

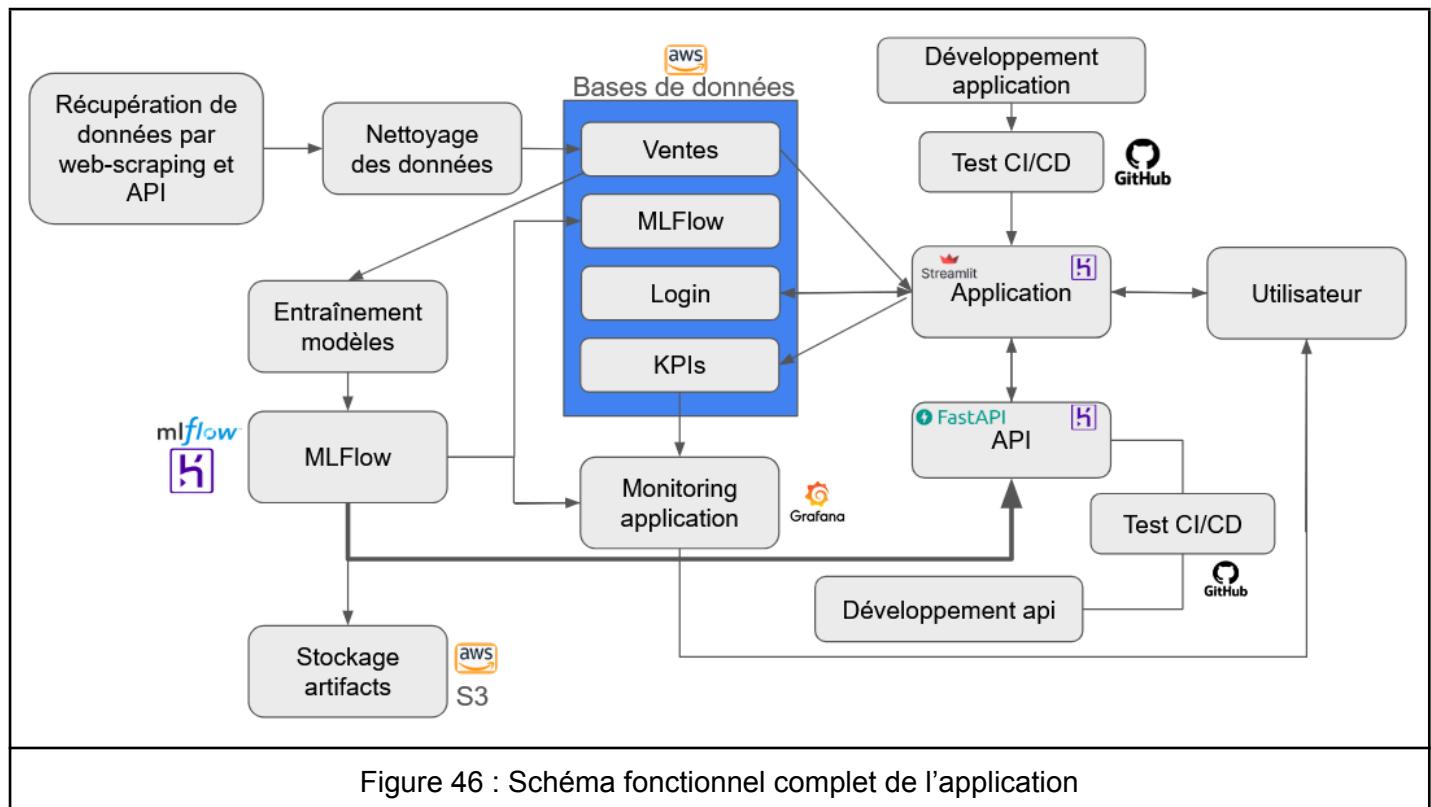
Concernant la réalisation du projet deux difficultés ont été rencontrées. La première était le manque d'informations sur les biens en vente ne permettant pas d'obtenir des performances convenables et d'obtenir une estimation fiable. La deuxième était la gestion des coûts par les différents services, Heroku, AWS, Google Colab.

### c. Les améliorations et perspectives d'évolution

Pour cette application qui réalise des prédictions de prix de vente de biens immobiliers, l'objectif serait de trouver des données où il y aurait plus d'informations sur le bien comme par exemple l'année de construction, les types de matériaux utilisés, le diagnostic de performances énergétiques, les services à proximité (écoles, commerces...), si le bien est à rafraîchir ou à rénover... Ce projet pourrait être réalisé à plus petite échelle, dans une agence départementale ou régionale dans lesquelles il y aurait plus d'information sur les biens vendus. Ces agences pourraient également mieux cibler leurs prospections en s'appuyant sur le monitoring de l'application réalisé avec Grafana qui permet de visualiser les demandes de prédictions par communes.

Un second modèle pourrait aussi être créé pour les communes côtières et les grandes agglomérations, car ces communes sont prises par le modèle en production comme des valeurs aberrantes.

## 8. Annexes



Caen le 12 Février 2024,

**Compte rendu du second échange : Situation du projet à mi parcours**

**Parties prenantes :**

- La société demandeuse FNAIM représentée par Caroline Kern Richard
- La société prestataire Normand'IA représentée par Kevin LE GRAND

**Objectifs :**

- Permettre de faire le point sur l'avancement du projet et de voir la cohérence avec ce qui est attendu.

**Déroulement :**

- L'entreprise Normand'IA réalise une démonstration de l'application en cours de développement. Elle a permis de démontrer que l'application permettait de réaliser une prédition en indiquant différents critères.

**Questions/réponses :**

- Q.FNAIM : Est ce que les performances de prédictions pourront être améliorées ?
- R.Normand'IA : Oui, le but de cette première étape était de mettre en œuvre toutes les ressources nécessaires à l'application (bases de données, le déploiement des divers services, api, mlflow, application et monitoring de l'application).
- Q.FNAIM : Serait-il possible d'ajouter des statistiques sur les ventes par régions, départements et communes ainsi que des KPI sur l'utilisation de l'application.
- R.Normand'IA : Oui nous pouvons afficher les différentes statistiques en fonction des choix des utilisateurs. Pour les KPI nous pouvons créer un login qui permettra d'accéder ou non aux KPIs.

**Prochaines étapes :**

- Un rendez-vous a été fixé au 5 Avril pour la livraison de la version finale du projet.

LE GRAND Kevin

Caroline Kern Richard

X

X

Figure 47 : Compte-rendu N°2

Caen le 05 Avril 2024,

Compte rendu du second échange : **Situation du projet à mi parcours**

Parties prenantes :

- La société demandeuse FNAIM représentée par Caroline Kern Richard
- La société prestataire Normand'IA représentée par Kevin LE GRAND

**Objectifs :**

- Permettre de faire le point sur la livraison de l'application.

**Déroulement :**

- L'entreprise Normand'IA réalise une démonstration de l'application. Elle a permis de démontrer que l'application permettait de :
  - Réaliser une prédition en indiquant différents critères.
  - Afficher les différentes statistiques en fonction des choix des utilisateurs
  - L'accès aux KPIs de l'application.

**Questions/réponses :**

- Q.FNAIM : Merci pour votre travail et les délais tenus. Serait-il possible d'avoir de meilleures performances sur le résultat des prédictions ?
- R.Normand'IA : Malheureusement non, notre équipe a fait tout son maximum pour atteindre les meilleures performances possibles. Les données disponibles sur le site du gouvernement ne donnent pas assez d'informations sur les biens vendus. La solution serait d'avoir une base de données plus complète avec une information sur le DPE, l'état des biens en vente (à rénover, à rafraîchir, à rénover), les services à proximité...
- Q.Normand'IA : Pourriez-vous nous constituer une telle base de données ?
- R.FNAIM : oui certainement mais pas dans l'immédiat.
- Normand'IA : Notre équipe dispose de personnes qualifiées en bigdata et nous pouvons donc vous proposer nos services si besoin.

**Conclusion :**

- La société FNAIM représentée par Caroline Kern Richard est satisfaite du travail réalisé par la société Normand'IA.
- La société FNAIM reprendra contact avec la société Normand'IA dès qu'elle aura constitué une base de données plus complète.
- La société Normand'IA reste disponible pour la maintenance de l'application et de nouvelles missions.

LE GRAND Kevin

Caroline Kern Richard

X

X

Figure 48: Compte-rendu N°3

## 9. Glossaire

**GitHub** : GitHub est une plateforme de développement collaboratif, permettant aux développeurs de partager, collaborer et gérer des projets.

**CSV** : Un fichier csv (Comma Separated Values) est un format de fichier texte utilisé pour stocker des données tabulaires. Chaque ligne du fichier représente une ligne de données où les valeurs de chaque ligne sont séparées par une virgule.

**EDA (Exploratory Data Analysis)**: Une Analyse Exploratoire de Données (EDA) est une approche statistique et graphique visant à explorer et à comprendre les caractéristiques et les relations présentes dans un ensemble de données. L'objectif principal de l'EDA est de révéler des tendances, des schémas, des anomalies et des relations potentielles entre les variables sans imposer de suppositions ou de modèles prédefinis.

**API (Application Programming Interface)**: Une API, ou Interface de Programmation Applicative, est un ensemble de règles, de protocoles et de définitions qui permettent à différents logiciels et systèmes de communiquer entre eux.

**Webscraping** : Le "webscraping" en français est souvent traduit par "l'extraction de données web" ou "l'aspiration de données web". Il s'agit d'une technique informatique utilisée pour extraire automatiquement des données d'un site web.

**MCD (Modèle Conceptuel de Données)**: Le Modèle Conceptuel de données est une formalisation d'une structure et la signification des informations pouvant décrire des objets et des associations qui sont perçus comme ayant un intérêt dans le domaine étudié tout en faisant abstraction des solutions et des contraintes techniques informatiques d'implantation en base de données.

**MLD (Modèle Logique de Données)** : il décrit les entités, les attributs, les relations et les contraintes de manière à ce qu'elles puissent être traduites directement en schéma de base de données. Le modèle logique de données se situe donc entre le modèle conceptuel et le modèle physique, servant de pont entre la conception conceptuelle et la mise en œuvre pratique d'une base de données.

**SQL (Structured Query Language)** : SQL est un langage de programmation utilisé pour gérer et manipuler des bases de données relationnelles. Il permet de communiquer avec une base de données afin d'effectuer des opérations telles que l'insertion, la modification, la suppression et la récupération de données.

**AWS** : Amazon Web Service. Du cloud computing permettant de fournir des services comme du stockage, des bases de données, de la puissance de calcul... Plus de 200 services.

**RDS** : Relational Database Service. Permet de créer et de gérer différentes bases de données relationnelles comme MySql, Postgre, MariaDB, Oracle...

---

**IAM** : Identity and Access Management. C'est un identifiant utilisé pour se connecter à AWS et effectuer des actions autorisées en fonction des permissions accordées.

**MLflow** : MLflow est une plateforme open source conçue pour aider les développeurs et les ingénieurs en données. Elle fournit des outils pour suivre, gérer et déployer des modèles, ainsi que pour expérimenter différentes approches de modélisation. En bref, MLflow facilite le développement, la gestion et la mise en production des modèles de machine learning.

**Workflow** : est une manière systématique de gérer et d'accomplir des tâches en suivant un ensemble prédéfini de règles ou de directives.

**Images Docker** : Elles sont créées à partir de fichiers appelés Dockerfiles et contiennent tout le nécessaire pour exécuter une application, y compris le code source, les bibliothèques, les dépendances, les fichiers de configuration, etc. C'est comme une boîte virtuelle prête à l'emploi qui peut être exécutée sur n'importe quel système compatible avec Docker, assurant ainsi une portabilité et une cohérence des environnements d'exécution logicielle.

## 10. Index des graphiques

N°	Titre	Page
1	Diagramme de Gantt	6
2	Kanban Réalisé sur Trello	6
3	Schéma de l'application	8
4	Architecture du site des demandes de valeurs foncières géolocalisées	9
5	Modèle Logique de Données	12
6	Répartition des ventes par type de bien et par années	13
7	Répartition des prix en fonction du type de bien	14
8	Carte de France de température de prix	15
9	Surface moyenne en fonction du montant de la vente d'un appartement	16
10	Variation d'un bien immobilier en région Île-de-France en fonction du nombre de pièces	17
11	Évolution dans le temps des prix de l'immobilier en France	17
12	Évolution des prix en région Bretagne avant suppression des valeurs aberrantes	18
13	Évolution des prix en région Bretagne après suppression des valeurs aberrantes	19
14	Matrice de corrélation	20
15	Exemple de régression linéaire	22
16	Tableau récapitulatif des métriques	25
17	Structure d'un arbre de décision	27
18	Construction d'un arbre de décision	28
19	Mise en oeuvre d'une forêt d'arbres aléatoires	29
20	Prédiction d'une forêt d'arbres aléatoires	29
21	Différence entre un Random Forest et un XGBoost	30
22	Fonctionnement d'un réseau de neurones	31
23	Requête pour la récupération de données	33
24	Aperçu du résultat de la requête	33
25	Calcul de seuil des outliers	34
26	Dataset montrant l'impact du retrait des outliers	34

N°	Titre	Page
27	Impact final du retrait des outliers	35
28	Schéma de validation croisée	36
29	Illustration du sur-apprentissage, sous apprentissage et apprentissage correct	37
30	Monitoring MLFlow. Comparaison de 4 modèles	37
31	Courbe de validation (validation curve)	38
32	Importance des variables (features importance)	39
33	Courbe d'apprentissage	39
34	Code pour les tests unitaires	40
35	Variables d'environnement sur GitHub	41
36	Configuration sur Heroku pour le déploiement continue	42
37	Construction de l'image Docker pour l'application	42
38	Affichage du test d'intégration continue	43
39	Affichage de la confirmation du déploiement	43
40	Maquette de l'application	44
41	Capture d'écran de la page lors d'une prédiction	44
42	Gestion des utilisateurs	45
43	Capture d'écran FastApi	46
44	Tableau de bord permettant le monitoring de l'application avec Grafana	47
45	Screenshot d'une alerte mail envoyé par Grafana	47
46	Schéma fonctionnel complet de l'application	50
47	Compte-rendu N°2	51
48	Compte-rendu N°3	52

## 11. Bibliographie

Définition régression linéaire	<a href="https://kobia.fr/quest-ce-quune-regression-lineaire/">https://kobia.fr/quest-ce-quune-regression-lineaire/</a>
Fonction perte ou coût	<a href="https://www.cnil.fr/fr/definition/fonction-de-perte-ou-de-cout-loss-function">https://www.cnil.fr/fr/definition/fonction-de-perte-ou-de-cout-loss-function</a>
Descente de gradients	<a href="https://www.youtube.com/watch?v=rcl_YRyoLIY">https://www.youtube.com/watch?v=rcl_YRyoLIY</a>
Métrique de performances RMSE/MSE,MAE,MAPE,MSLE	<a href="https://kobia.fr/regression-metricsquelle-metrique-choisir/">https://kobia.fr/regression-metricsquelle-metrique-choisir/</a>
Métrique de performances R <sup>2</sup> score	<a href="https://kobia.fr/regression-metrics-r2-score/">https://kobia.fr/regression-metrics-r2-score/</a>
Définition arbre de décision Arthur Flor	<a href="https://arthurflor23.medium.com/prediction-of-late-payments-using-decision-tree-based-algorithms-ce72a2fbccab">https://arthurflor23.medium.com/prediction-of-late-payments-using-decision-tree-based-algorithms-ce72a2fbccab</a>
Illustration Random Forest IBM	<a href="https://www.ibm.com/fr-fr/topics/decision-trees">https://www.ibm.com/fr-fr/topics/decision-trees</a>
Présentation XGBoost ActuIA	<a href="https://www.actuia.com/vulgarisation/dcouvrir-et-comprendre-xgboost/">https://www.actuia.com/vulgarisation/dcouvrir-et-comprendre-xgboost/</a>
Random Forest vs XGBoost Aman Gupta medium.com	<a href="https://medium.com/geekculture/xgboost-versus-random-forest-898e42870f30">https://medium.com/geekculture/xgboost-versus-random-forest-898e42870f30</a>
Réseau de neurones Machine Learnia	<a href="https://www.youtube.com/watch?v=XUFLq6dKQok">https://www.youtube.com/watch?v=XUFLq6dKQok</a>

## 12. Citations

[P.27] BREIMAN, Leo. Random forests. *Machine learning*, 2001, vol. 45, p. 5-32.

[P.29] CHEN, Tianqi et GUESTRIN, Carlos. Xgboost: A scalable tree boosting system. In : *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016. p. 785-794.

[P.30] MCCULLOCH, Warren S. et PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 1943, vol. 5, p. 115-133.

[P.30] ROSENBLATT, Frank. Perceptron simulation experiments. *Proceedings of the IRE*, 1960, vol. 48, no 3, p. 301-309.

[P.30] RUMELHART, David E., HINTON, Geoffrey E., et WILLIAMS, Ronald J. Learning Internal Representations by Error Propagation, Parallel Distributed Processing, Explorations in the Microstructure of Cognition, ed. DE Rumelhart and J. McClelland. Vol. 1. 1986. *Biometrika*, 1986, vol. 71, p. 599-607.

[P.30] LECUN, Yann, BENGIO, Yoshua, *et al.* Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 1995, vol. 3361, no 10, p. 1995.