

PROJET CHEF D'ŒUVRE

ESTIMATIONS DE BIENS IMMOBILIERS

Développeur en intelligence artificielle

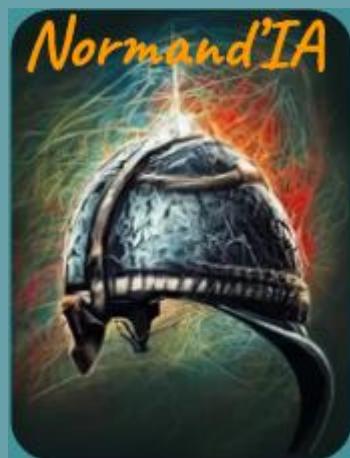
Certification RNCP34757

SOMMAIRE

• Introduction	3
• Gestion de projet	4
• Outils et schéma fonctionnel	6
• Récupération de données	7
• Base de données relationnelle	9
• Insertion des données dans la bdd	10
• Analyse exploratoire de données	11
• Veille technique	17
• Création du modèle	22
• Mise en place de l'environnement CI/CD	28
• Frontend	30
• Backend	31
• Monitoring	33
• Démonstration	35
• Conclusion	36
• Questions/réponses	

Introduction

Premier compte rendu de l'échange
entre les parties prenantes



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 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

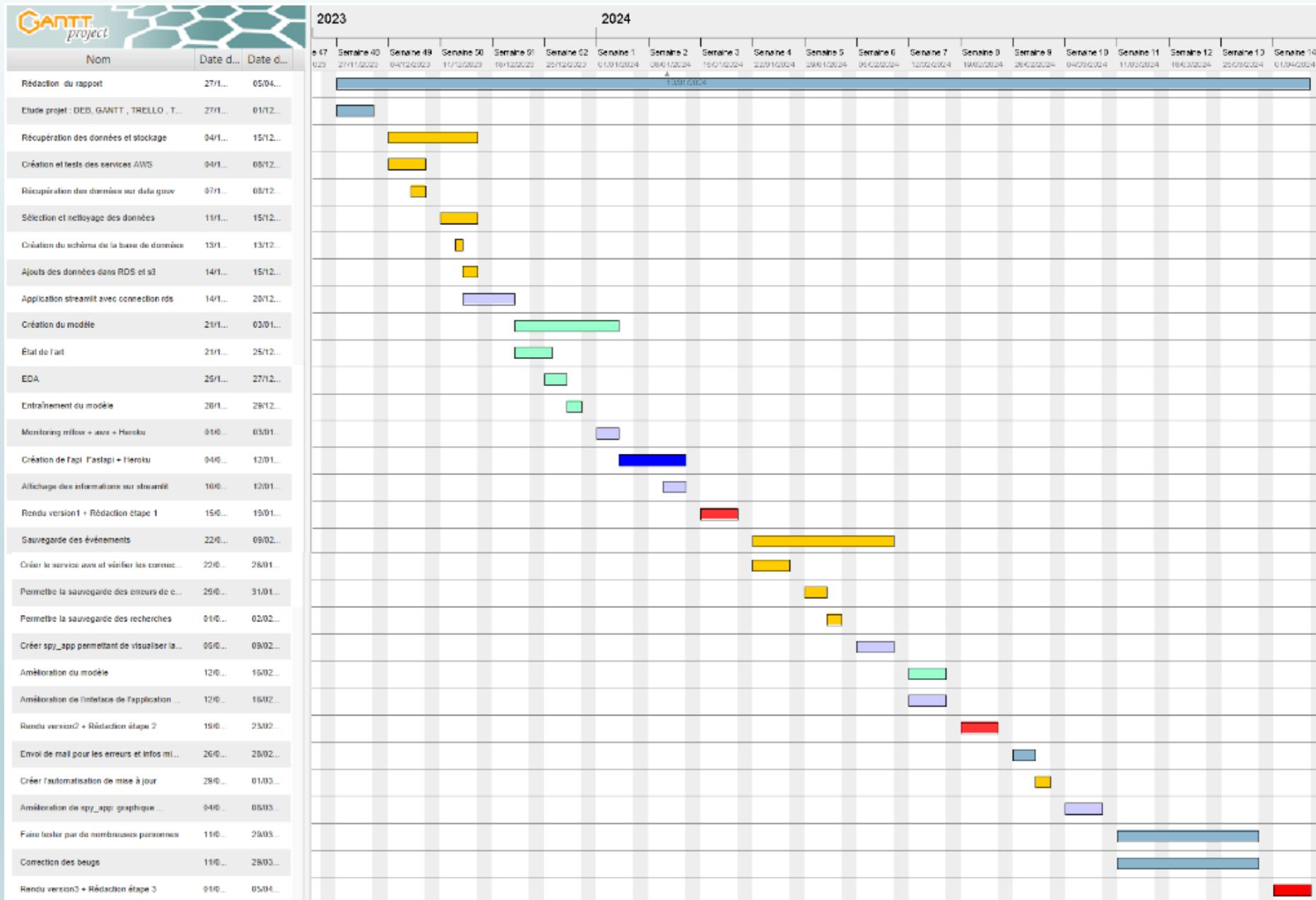
Caroline Kern Richard

X

X

Gestion de projet :

À la suite du premier échange entre les 2 parties prenantes, l'équipe de Normand'IA a mis en **diagramme de Gantt** avec les différentes étapes afin de planifier les étapes du projet et de prévoir la complétion dans les temps impartis.



Gestion de projet :

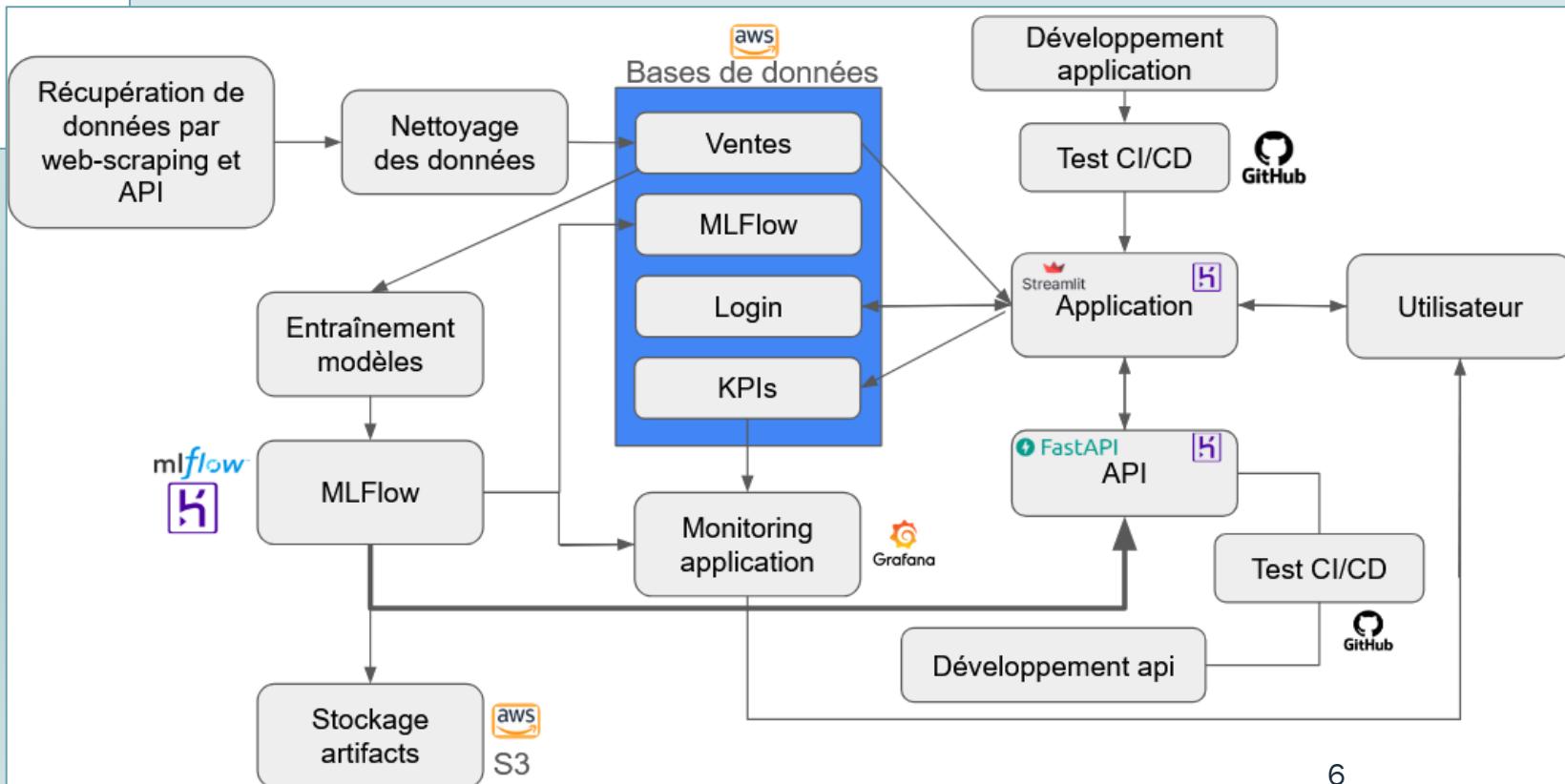
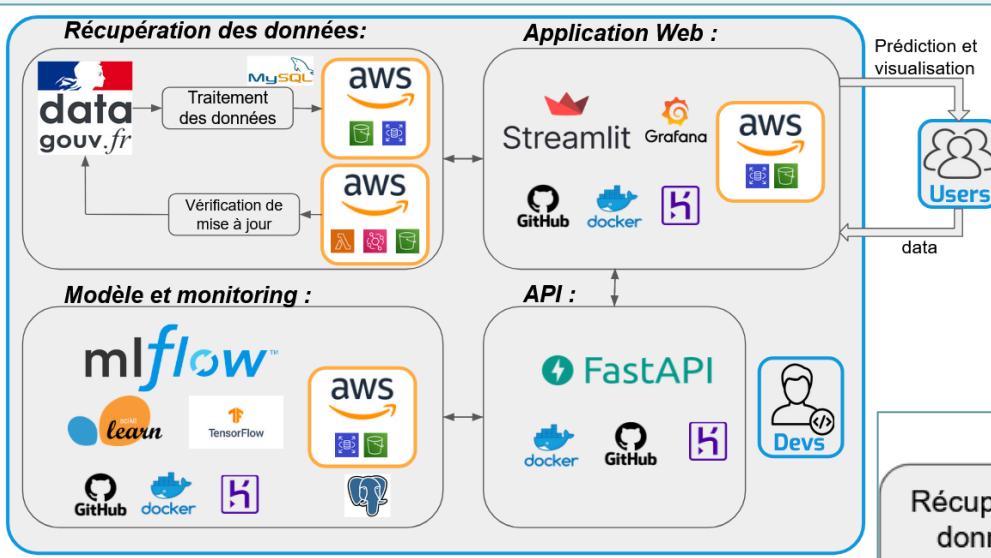
Pour mener à bien ce projet l'équipe a décidé de le gérer en **méthode agile** avec la méthode **Kanban** et de se réunir chaque début de semaine pour voir l'avancement du projet, les étapes à venir, en cours, bloquées ou terminées.

Un tableau Kanban a été réalisé sur Trello permettant à toute l'équipe de visualiser les différentes tâches et leurs positionnements.

The screenshot shows a Trello board with the following structure:

- Backlog:**
 - Modèle: Réaliser l'état de l'art
 - Modèle application: Adapter la taille du point sur la carte interactive lors d'un zoom
 - Modèle: Choix du modèle
 - BDD: Ajouts de la superficie et du nombre d'habitants dans les tables communes
 - application: Réaliser le monitoring de l'application. Grafana
- À suivre:**
 - Modèle application: Stocker la heatmap de la carte de France sur S3 et l'ajouter à l'application
 - application: Ajouter une alerte mail en cas de plantage de l'application
- En cours:**
 - Livraison !!!: Livraison V1 + rédaction étape1
 - + Ajouter une carte
- Bloqués:**
 - BDD: Créer une base de données nosql DocumentDB sur AWS et la rendre disponible depuis une application externe.
 - + Ajouter une carte
- Terminé:**
 - rédaction de projet: Rédaction de l'expression des besoins
 - rédaction de projet: Diagramme de Gantt
 - BDD: Crédit d'un service Aws RDS
 - BDD: Schéma de base de données avec merise
 - BDD: Récupération des données avec les api :

Outils et schéma fonctionnel



Récupération de données

La récupération des données s'est faite à partir du site data.gouv.fr pour la partie concernant les ventes et à partir d'une API pour récupérer les informations sur les communes.

L'utilisation des librairies BeautifulSoup et requests de python ont été utilisées pour la récupération de données.

Extrait de webscraping

```
# URL de la page web
url = 'https://files.data.gouv.fr/geo-dvf/latest/csv/'

response = requests.get(url)
html_content = response.content

# Utilisez BeautifulSoup pour analyser le HTML
soup = BeautifulSoup(html_content, 'html.parser')

# Trouver toutes les balises <a>
a_tags = soup.find_all('a')
annees=[]

# Parcourir chaque balise <a> et extraire le texte ainsi que la date/heure
for a_tag in a_tags:
    name = a_tag.text.strip()
    if name !='..':
        annees.append(name)
print(annees)

['2018/', '2019/', '2020/', '2021/', '2022/', '2023/']
```

Code source de la page

```
<html>
<head><title>Index of /geo-dvf/latest/csv/</title></head>
<body>
<h1>Index of /geo-dvf/latest/csv/</h1><hr><pre><a href=".//..>..</a>
<a href="2018/">2018/</a>
<a href="2019/">2019/</a>
<a href="2020/">2020/</a>
<a href="2021/">2021/</a>
<a href="2022/">2022/</a>
<a href="2023/">2023/</a>
</pre><hr></body>
</html>
```

- Les communes depuis l'api

```
# Récupération des codes communes, noms communes et code départements
url = "https://geo.api.gouv.fr/communes"
response = requests.get(url)
data_comunes = response.json()
df_comunes = pd.DataFrame(data_comunes)
df_comunes = df_comunes.loc[:,["code","nom","codeDepartement"]]
df_comunes = df_comunes.drop_duplicates()
df_comunes.columns = ["ID_COMMUNE","NAME_COMMUNE","ID_DEPT"]
df_comunes.head()
```

	ID_COMMUNE	NAME_COMMUNE	ID_DEPT
0	01001	L'Abergement-Clémenciat	01
1	01002	L'Abergement-de-Varey	01
2	01004	Ambérieu-en-Bugey	01
3	01005	Ambérieux-en-Dombes	01
4	01006	Ambléon	01

- Les départements depuis l'api

```
# Récupération des codes départements, noms départements et code régions
url = "https://geo.api.gouv.fr/departements"
response = requests.get(url)
data_dpts = response.json()
df_dpts = pd.DataFrame(data_dpts)
df_dpts = df_dpts.loc[:,["code","nom","codeRegion"]]
df_dpts = df_dpts.drop_duplicates()
df_dpts.columns = ['ID_DEPT', 'Name_departement', 'ID_REGION']
df_dpts.head()
```

	ID_DEPT	Name_departement	ID_REGION
0	01	Ain	84
1	02	Aisne	32
2	03	Allier	84
3	04	Alpes-de-Haute-Provence	93
4	05	Hautes-Alpes	93

Structure du site

Code source de la page contenant les données

```
<html>
<head><title>Index of /geo-dvf/latest/csv/2018/</title></head>
<body>
<h1>Index of /geo-dvf/latest/csv/2018/</h1><hr><pre><a href="..>..</a>
<a href="communes/">communes/</a>
<a href="departements/">departements/</a>
<a href="full.csv.gz">full.csv.gz</a>
</pre><hr></body>
</html>
```

Récupération de données

Avant l'insertion en base de données un premier traitement était nécessaire :

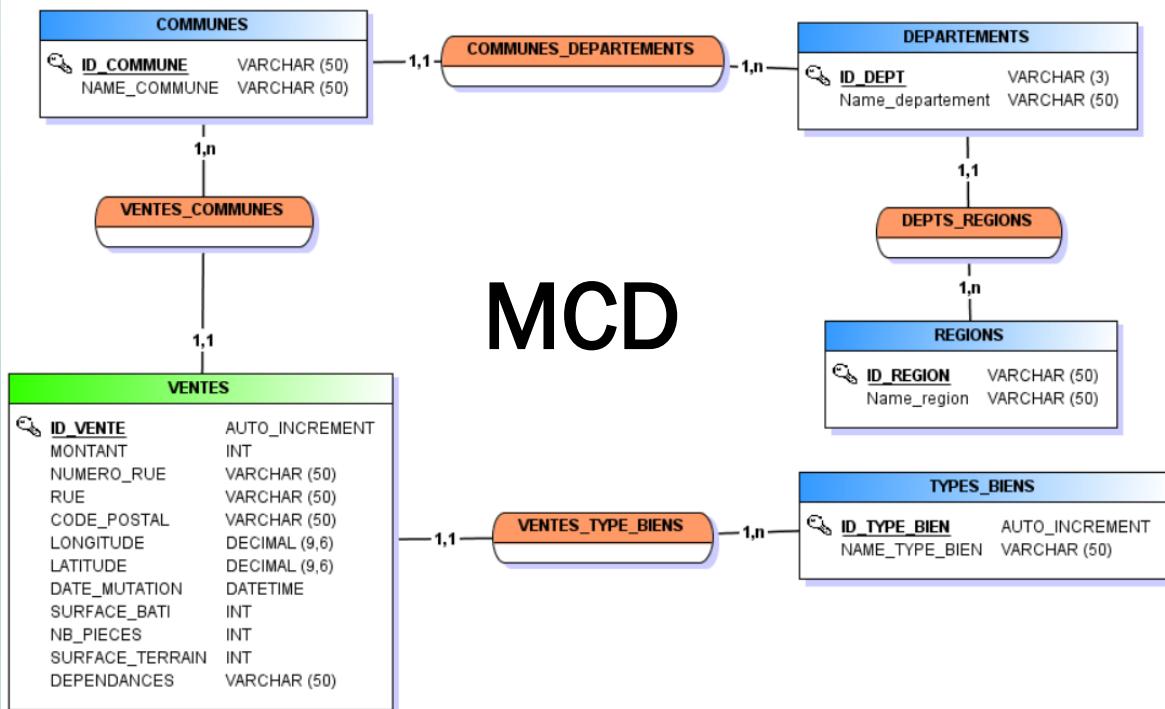
- Sélectionner les données conformément aux attentes de la FNAIM et les caractéristiques utiles à la création d'un modèle d'intelligence artificielle.
- Gérer les valeurs nulles (exemple suppression des biens sans date de vente ou de montant, remplacement par 0 d'un numéro de rue...)
- Formater les données au bon format (Date, string, integer...)
- Grouper les données par identifiant de vente et adapter une stratégie pour chaque colonne (valeur moyenne pour la géolocalisation, la somme pour la surface du terrain...)
- Adapter les colonnes qui serviront de clé étrangère dans la future base de données

- `id_mutation` : Identifiant de mutation (non stable, sert à grouper les lignes)
- `date_mutation` : Date de la mutation au format ISO-8601 (YYYY-MM-DD)
- `numero_disposition` : Numéro de disposition
- `nature_mutation` : Nature de la mutation
- `valeur_fonciere` : Valeur foncière (séparateur décimal = point)
- `adresse_numero` : Numéro de l'adresse
- `adresse_suffixe` : Suffixe du numéro de l'adresse (B, T, Q)
- `adresse_code_voie` : Code FANTOIR de la voie (4 caractères)
- `adresse_nom_voie` : Nom de la voie de l'adresse
- `code_postal` : Code postal (5 caractères)
- `code_commune` : Code commune INSEE (5 caractères)
- `nom_commune` : Nom de la commune (accentué)
- `ancien_code_commune` : Ancien code commune INSEE (si différent lors de la mutation)
- `ancien_nom_commune` : Ancien nom de la commune (si différent lors de la mutation)
- `code_departement` : Code département INSEE (2 ou 3 caractères)
- `id_parcelle` : Identifiant de parcelle (14 caractères)
- `ancien_id_parcelle` : Ancien identifiant de parcelle (si différent lors de la mutation)
- `numero_volume` : Numéro de volume
- `lot_1_numero` : Numéro du lot 1
- `lot_1_surface_carrez` : Surface Carrez du lot 1
- `lot_2_numero` : Numéro du lot 2
- `lot_2_surface_carrez` : Surface Carrez du lot 2
- `lot_3_numero` : Numéro du lot 3
- `lot_3_surface_carrez` : Surface Carrez du lot 3
- `lot_4_numero` : Numéro du lot 4
- `lot_4_surface_carrez` : Surface Carrez du lot 4
- `lot_5_numero` : Numéro du lot 5
- `lot_5_surface_carrez` : Surface Carrez du lot 5
- `nombre_lots` : Nombre de lots
- `code_type_local` : Code de type de local
- `type_local` : Libellé du type de local
- `surface_reelle_bati` : Surface réelle du bâti
- `nombre_pieces_principales` : Nombre de pièces principales
- `code_nature_culture` : Code de nature de culture
- `nature_culture` : Libellé de nature de culture
- `code_nature_culture_speciale` : Code de nature de culture spéciale
- `nature_culture_speciale` : Libellé de nature de culture spéciale
- `surface_terrain` : Surface du terrain
- `longitude` : Longitude du centre de la parcelle concernée (WGS-84)
- `latitude` : Latitude du centre de la parcelle concernée (WGS-84)

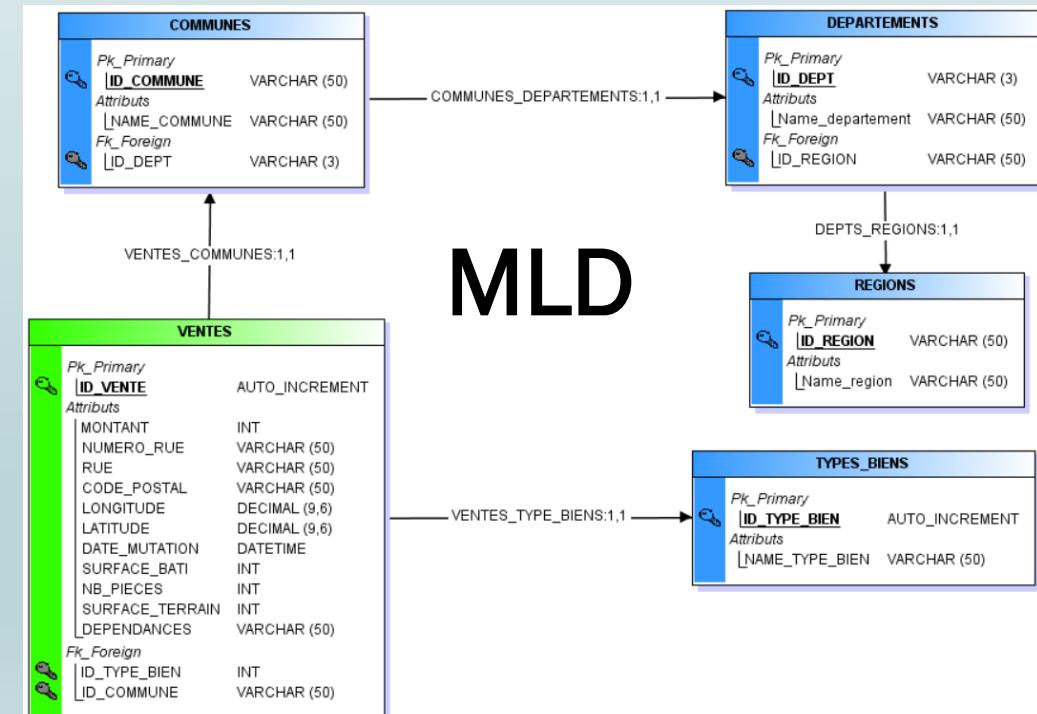
Création de la base de données



La base de données a été modélisée à l'aide de Jmerise, puis créée et hébergée sur le service RDS d'AWS. La technologie retenue pour la base de données fut MySQL pour sa stabilité et ses performances.



Jmerise est un outil permettant de créer facilement une base de données. Il suffit de créer le **model conceptuel de données** (MCD) avec les tables voulues, les clés primaires et les points de cardinalités. Ensuite Jmerise crée lui-même le **modèle logique de données** (MLD) en ajoutant les clés étrangères ainsi qu'un fichier SQL permettant de créer les tables.



Insertion en base de données

Pour l'insertion des données dans la base de données la méthode a été de les insérer des tables les plus éloignées à la table principale.

```
query="""  
SELECT  
    V.*,  
    T.NAME_TYPE_BIEN,  
    C.NAME_COMMUNE,  
    D.Name_departement,  
    R.Name_region  
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 AS D ON C.ID_DEPT = D.ID_DEPT  
INNER JOIN REGIONS R ON D.ID_REGION = R.ID_REGION  
LIMIT 10;  
"""  
df = pd.read_sql(query, engine)  
df.head(10)
```

```
# Insertion dans la table REGIONS  
for index, row in df_regions.iterrows():  
    query=""  
    INSERT IGNORE INTO REGIONS (ID_REGION,Name_region)  
    VALUES (%s,%s)  
    """  
    cursor.execute(query,(row["ID_REGION"],row["Name_region"]))  
db.commit()  
# Insertion dans la table DEPARTEMENTS  
for index, row in df_depts.iterrows():  
    query=""  
    INSERT IGNORE INTO DEPARTEMENTS (ID_DEPT, Name_departement, ID_REGION)  
    VALUES (%s,%s,%s)  
    """  
    cursor.execute(query,(row["ID_DEPT"],row["Name_departement"],row["ID_REGION"]))  
db.commit()  
# Insertion dans la table COMMUNES  
for index, row in df_communes.iterrows():  
    query=""  
    INSERT IGNORE INTO COMMUNES (ID_COMMUNE, NAME_COMMUNE, ID_DEPT)  
    VALUES (%s,%s,%s)  
    """  
    cursor.execute(query,(row["ID_COMMUNE"],row["NAME_COMMUNE"],row["ID_DEPT"]))  
db.commit()  
# Insertion dans la table TYPES_BIENS  
liste_type_local = ["Appartement","Maison"]  
for local in liste_type_local:  
    query=""  
    INSERT IGNORE INTO TYPES_BIENS (NAME_TYPE_BIEN)  
    VALUES(%s)  
    """  
    cursor.execute(query,(local))  
db.commit()  
df.to_sql(name='VENTES', con=engine, if_exists='append', index=False)
```

ID_VENTE	MONTANT	NUMERO_RUE	RUE	CODE_POSTAL	LONGITUDE	LATITUDE	DATE_MUTATION	SURFACE_BATI	NB_PIECES	SURFACE_TERRAIN	DEPENDANCES	ID_TYPE_BIEN	ID_COMMUNE	NAME_TYPE_BIEN	NAME_COMMUNE	Name_departement	Name_region
2	37000	5	RUE D YPRES	01000	5.220403	46.197326	2018-07-06	30	1	0	1	1	01053	Appartement	Bourg-en-Bresse	Ain	Auvergne-Rhône-Alpes
43	74000	5670	BD MORINET	16260	0.450180	45.825141	2018-08-02	32	2	621	1	1	16085	Appartement	Chasseneuil-sur-Bonnieure	Charente	Nouvelle-Aquitaine
48	50000	13	GR GRANDE RUE	16110	0.386824	45.740376	2018-08-01	113	5	0	1	1	16281	Appartement	La Rochefoucauld-en-Angoumois	Charente	Nouvelle-Aquitaine
87	45000	13	GR GRANDE RUE	16110	0.386824	45.740376	2018-08-09	103	3	0	1	1	16281	Appartement	La Rochefoucauld-en-Angoumois	Charente	Nouvelle-Aquitaine
118	120000	21	BD VICTOR HUGO	01000	5.229090	46.201418	2018-09-28	82	4	0	0	1	01053	Appartement	Bourg-en-Bresse	Ain	Auvergne-Rhône-Alpes
167	123000	2	RUE THEOPHILE GIBOUIN	16500	0.670092	46.015351	2018-08-31	48	3	2262	0	1	16106	Appartement	Confolens	Charente	Nouvelle-Aquitaine
179	254000	3	CHAMPS DE LA MONTEE	16510	0.225537	45.993701	2018-09-17	100	4	61625	0	1	16400	Appartement	Verteuil-sur-Charente	Charente	Nouvelle-Aquitaine
218	16000	41	RUE GABRIEL PERI	02100	3.283414	49.844160	2018-11-09	39	2	0	0	1	02691	Appartement	Saint-Quentin	Aisne	Hauts-de-France
233	60000	154	BD GAMBETTA	02100	3.292034	49.852991	2018-10-25	51	3	0	0	1	02691	Appartement	Saint-Quentin	Aisne	Hauts-de-France
265	47000	26	RUE DU VIEUX PORT	02100	3.277169	49.841145	2018-10-26	48	2	0	0	1	02691	Appartement	Saint-Quentin	Aisne	Hauts-de-France

Analyse exploratoire de données (EDA)

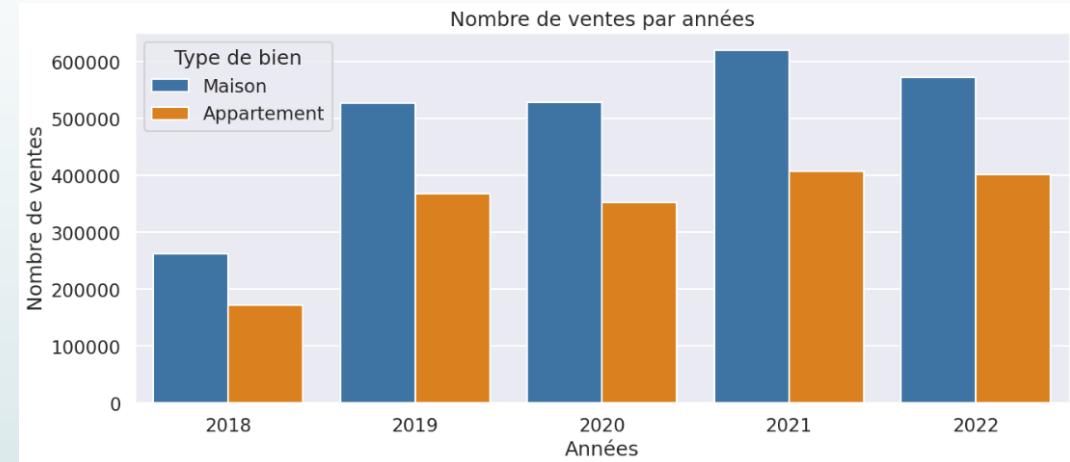
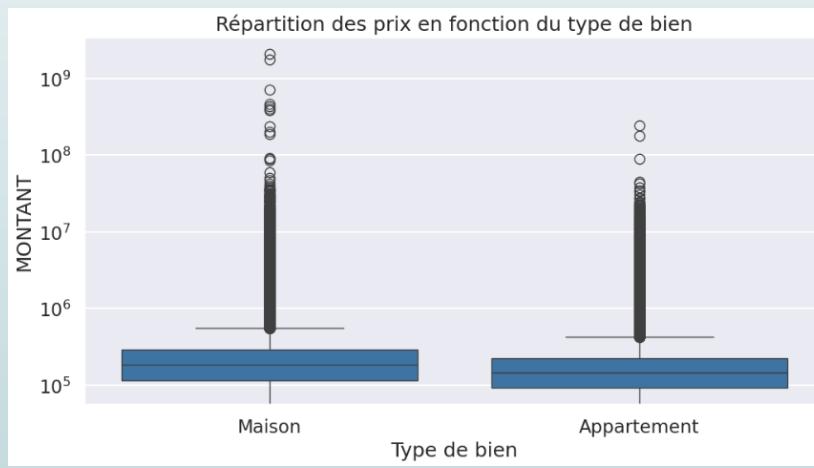
L'analyse exploratoire de données permet de :

- La compréhension des données (distribution, type, relation les unes avec les autres).
- L'identification des tendances.
- La détection de valeurs aberrantes (les données manquantes étant gérés avant l'insertion en base de données).
- La validation des hypothèses.
- La sélection des variables pertinentes.
- Communiquer des résultats préliminaires.

EDA : Distribution

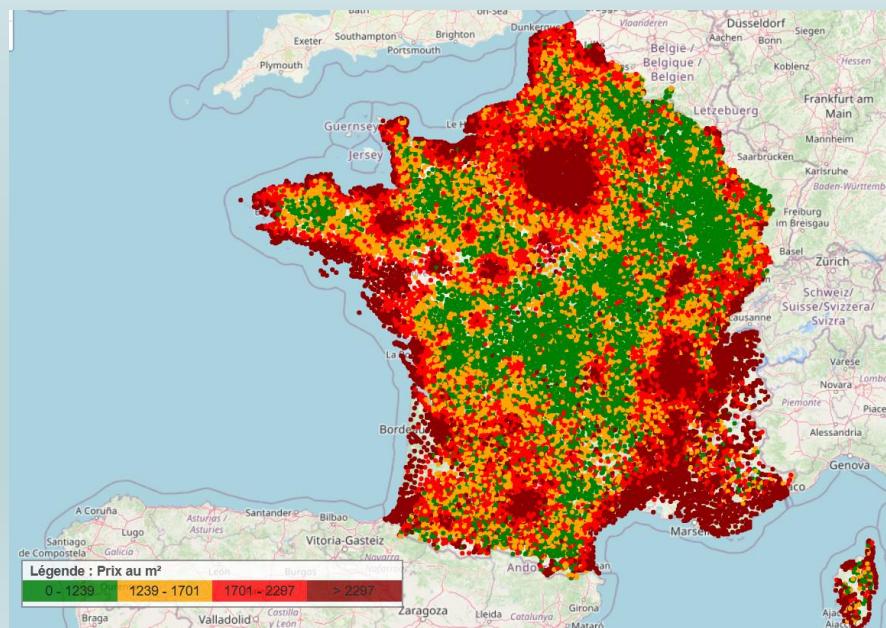
Le nombre de ventes augmentes d'années en années.

Le nombre de ventes de maisons sont toujours supérieures à celle des appartements.



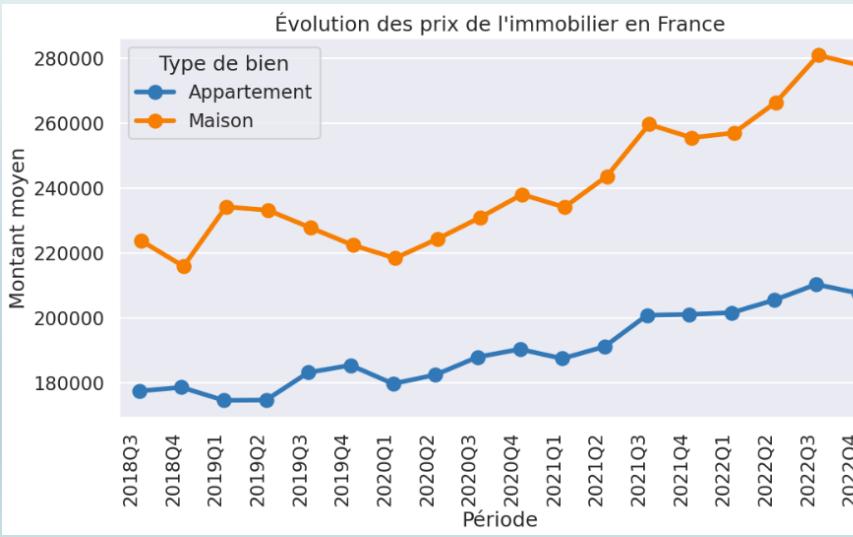
Le graphique en échelle logarithmique montre une grande disparité des prix de ventes. Ces valeurs aberrantes peuvent être dues à une région où les prix sont beaucoup plus chers que la moyenne ou une erreur de saisie.

Cette carte de température, réalisée avec la librairie folium de python, montre que plus on se rapproche des grandes agglomérations ainsi que les villes touristiques (villes côtières, Sud, Alpes) plus le prix de vente moyen d'un bien est élevé.

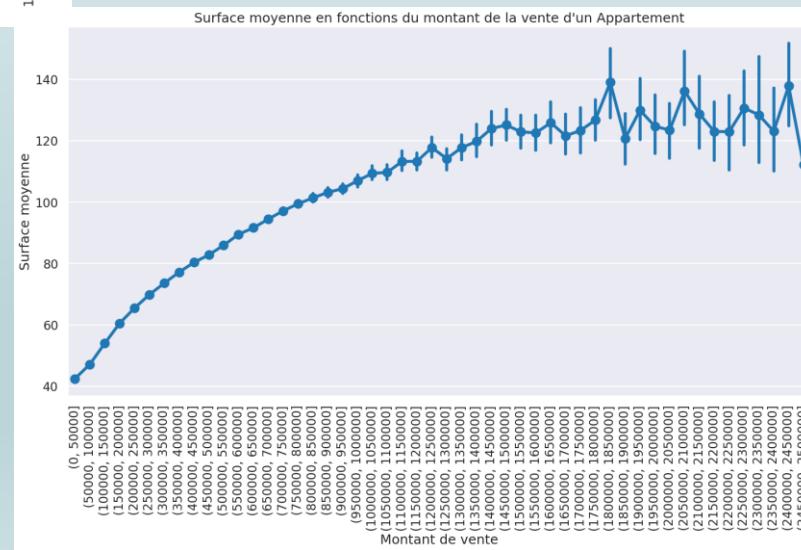
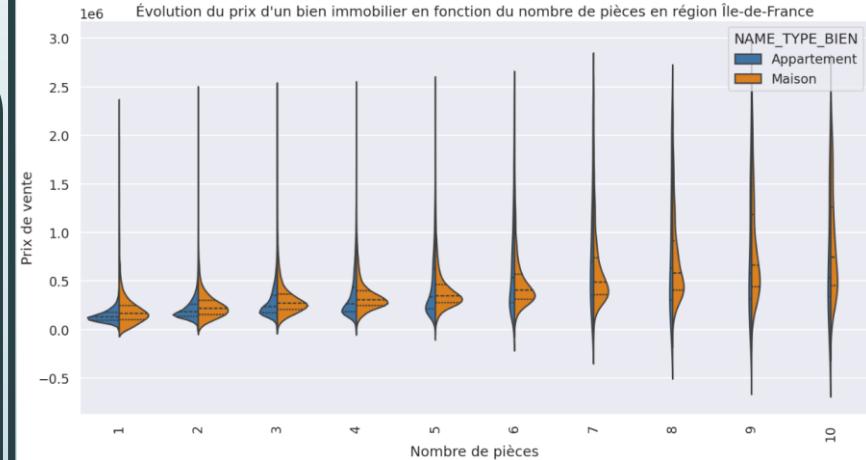


EDA : Tendances

Ce graphique confirme que l'évolution du prix de l'immobilier est en tendance haussière.

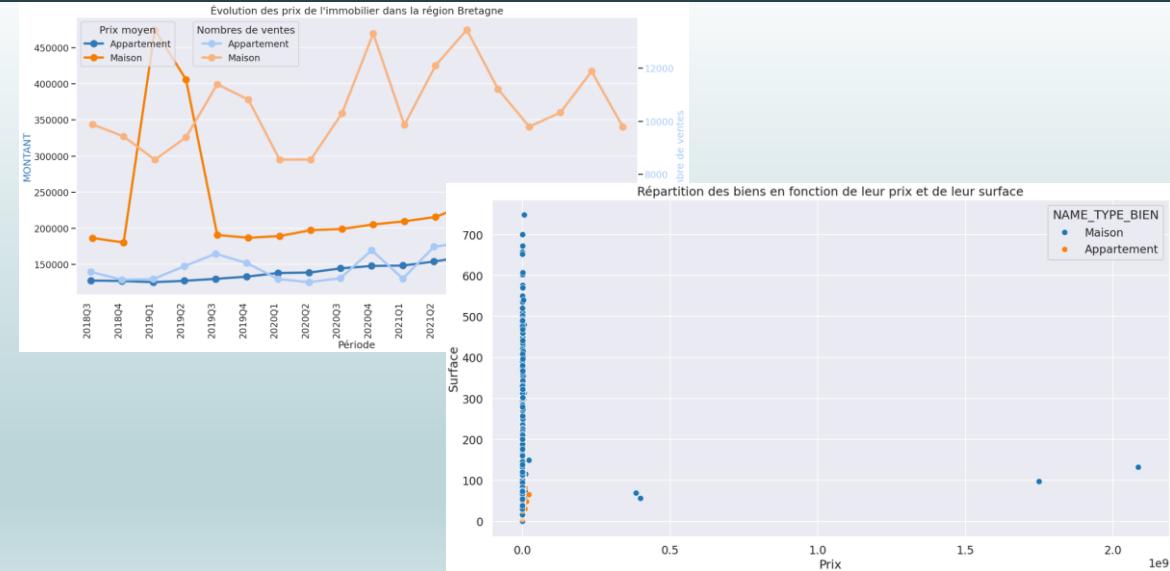


Ces deux graphiques montrent que plus un logement est grand plus la variation du prix est importante.

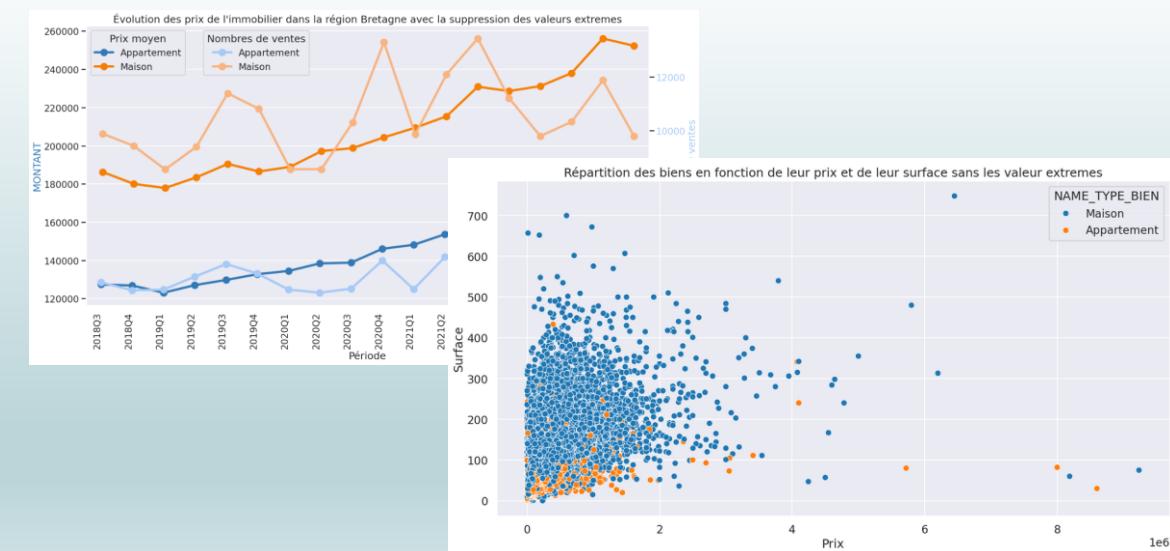


EDA : Valeurs aberrantes

Le graphique sur l'évolution du prix moyen de vente par trimestre et par type de bien pour la région Bretagne, montre des valeurs nettement supérieures à la normale pour le premier et second trimestre 2019.

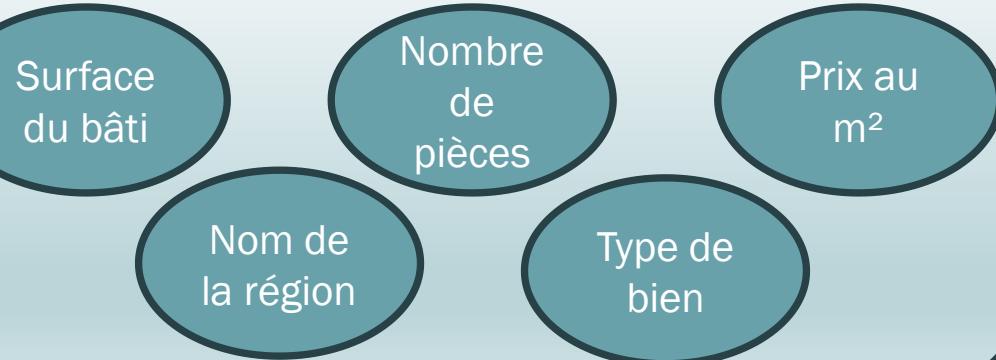


La suppression de ses valeurs aberrantes permet de revenir à une évolution cohérente du prix de vente moyen d'un bien immobilier.

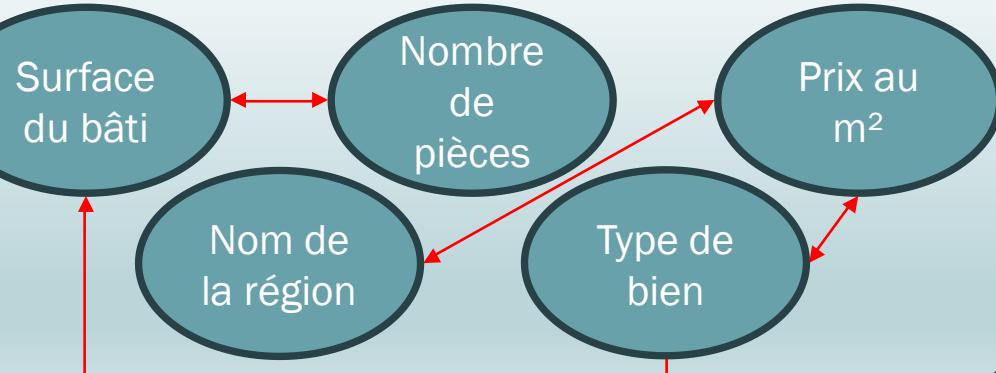


EDA : Corrélation entre les variables

Variables les plus corrélées avec le montant



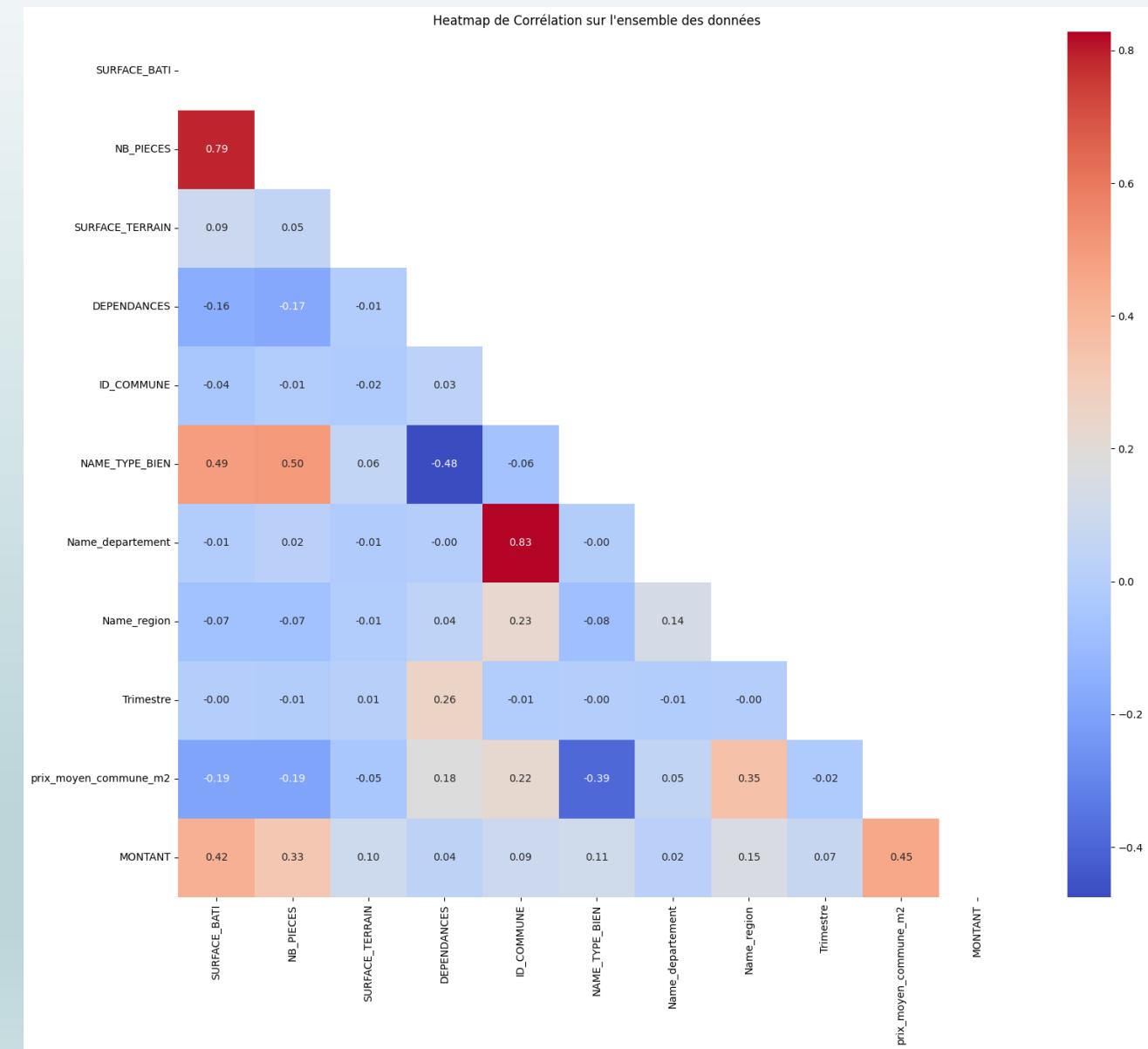
Variables corrélées entre elles



Variable faiblement corrélée avec le montant mais aucune corrélation avec une autre variable

Surface du terrain

Heatmap de Corrélation sur l'ensemble des données



EDA : Conclusion

Pour la création du modèle d'intelligence artificielle et afin d'obtenir de meilleures performances et un ajustement correct, les variables choisies pour estimer le prix d'un bien immobilier seront :

La surface du bâti

Le prix au m²

La surface du terrain

La récupération des données depuis la base de données se fera en ajoutant un filtre permettant d'enlever les biens ayant un montant aberrant par rapport à la moyenne des biens vendus.

Veille technique : régression linéaire et fonction coût

L'estimation d'un bien par rapport aux données existantes est un problème de régression linéaire.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (f_\theta(x^{(i)}) - y^{(i)})^2$$

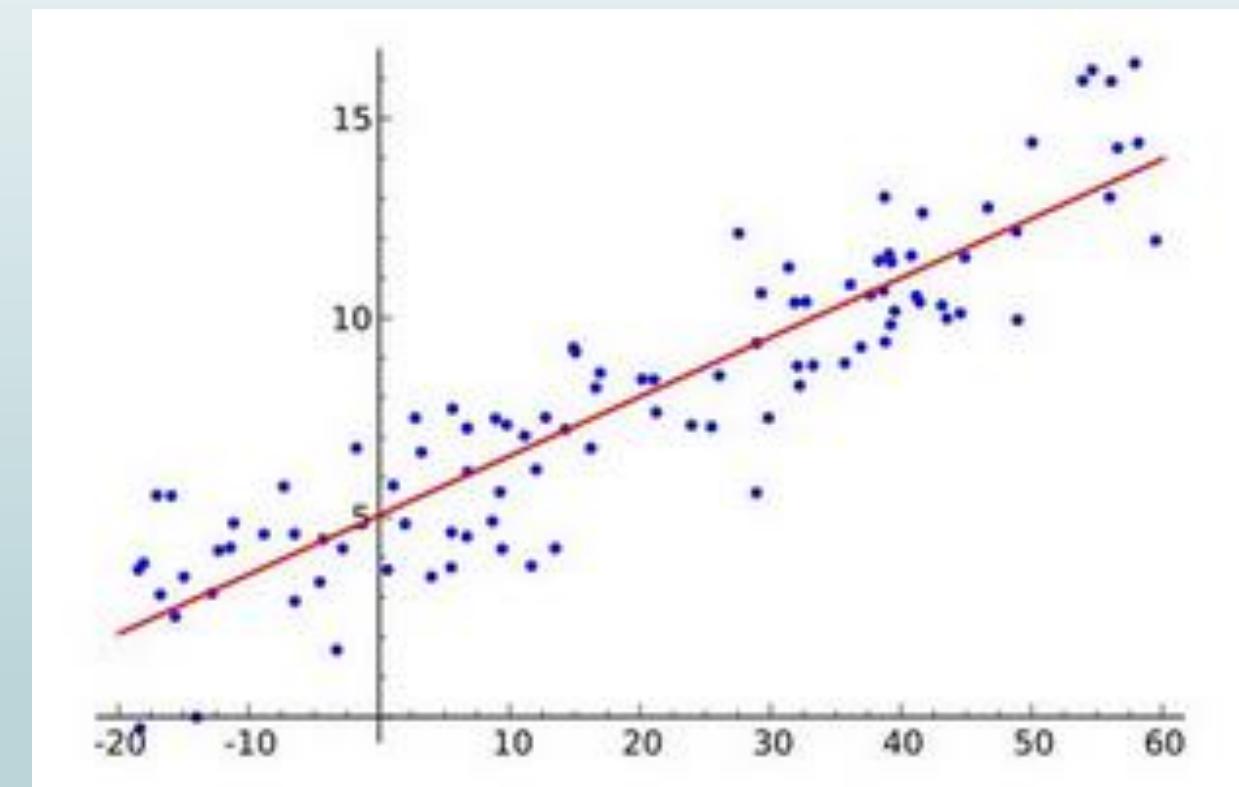
$J(\theta)$ est la fonction coût,

θ 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 .



Veille technique : métriques de performances

Afin de déterminer modèles à de bonnes performances nous allons utiliser 3 métriques facilement interprétables :



Mean Absolute Error est la moyenne des erreurs en valeur absolues, exemple :

Pour une erreur de +10k€ et une autre de -20k€

La MAE est de $(10k+20k)/2 \Rightarrow 15k€$

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |Y_i - \hat{Y}_i|$$



Mean Absolute Percentage Error suit le même principe que la MAE mais donne une erreur en pourcentage, exemple

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^n \left| \frac{Y_i - \hat{Y}_i}{Y_i} \right|$$



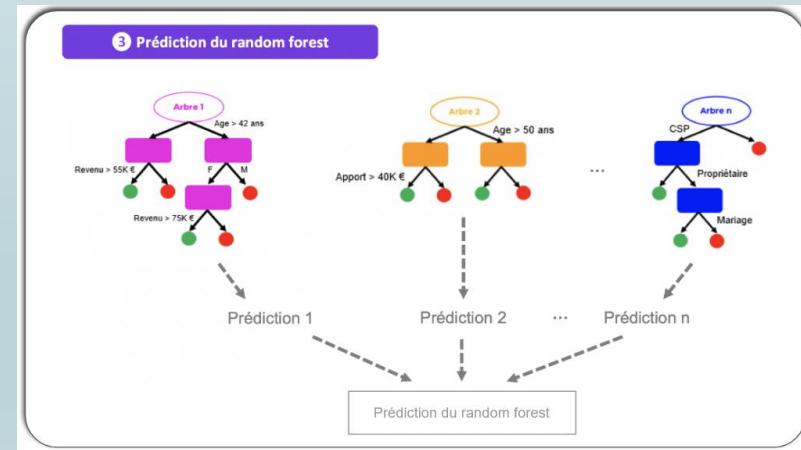
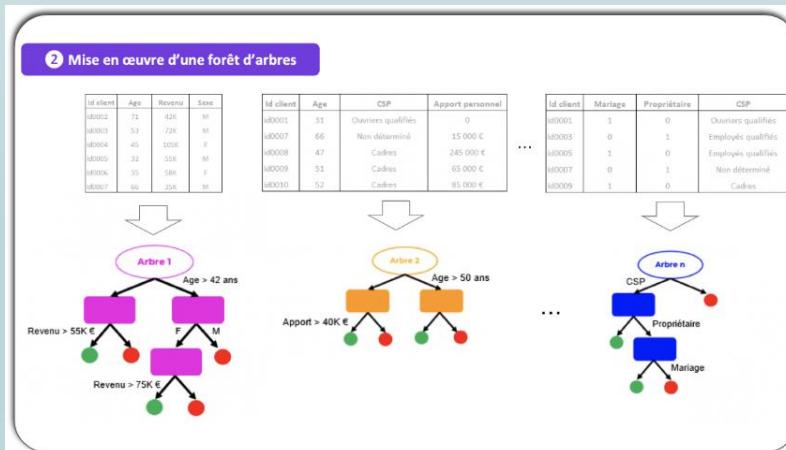
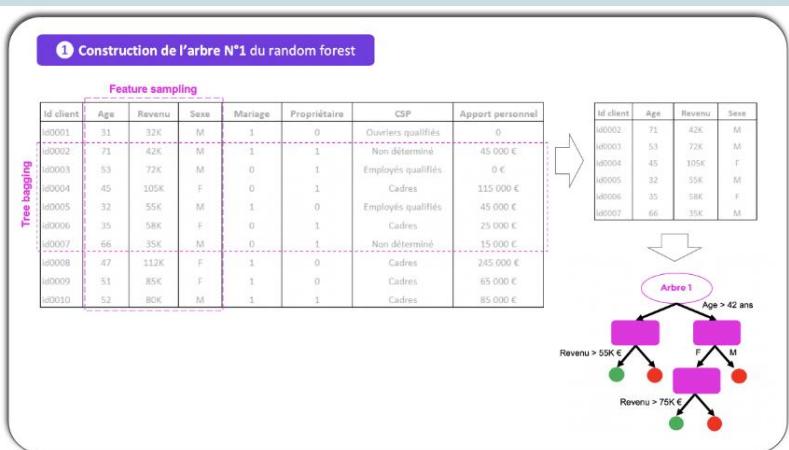
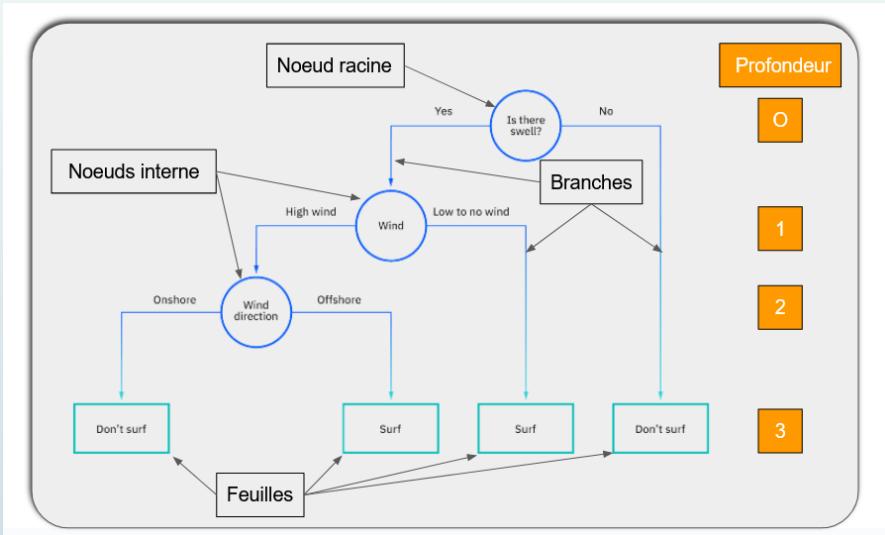
Le R² score permet de vérifier la pertinence de notre modèle. Le R² est une valeur comprise entre 1 (le modèle est parfait) et une valeur négative (le modèle est moins performant que si on prenait la valeur moyenne)

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

Veille technique : Le RandomForest



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



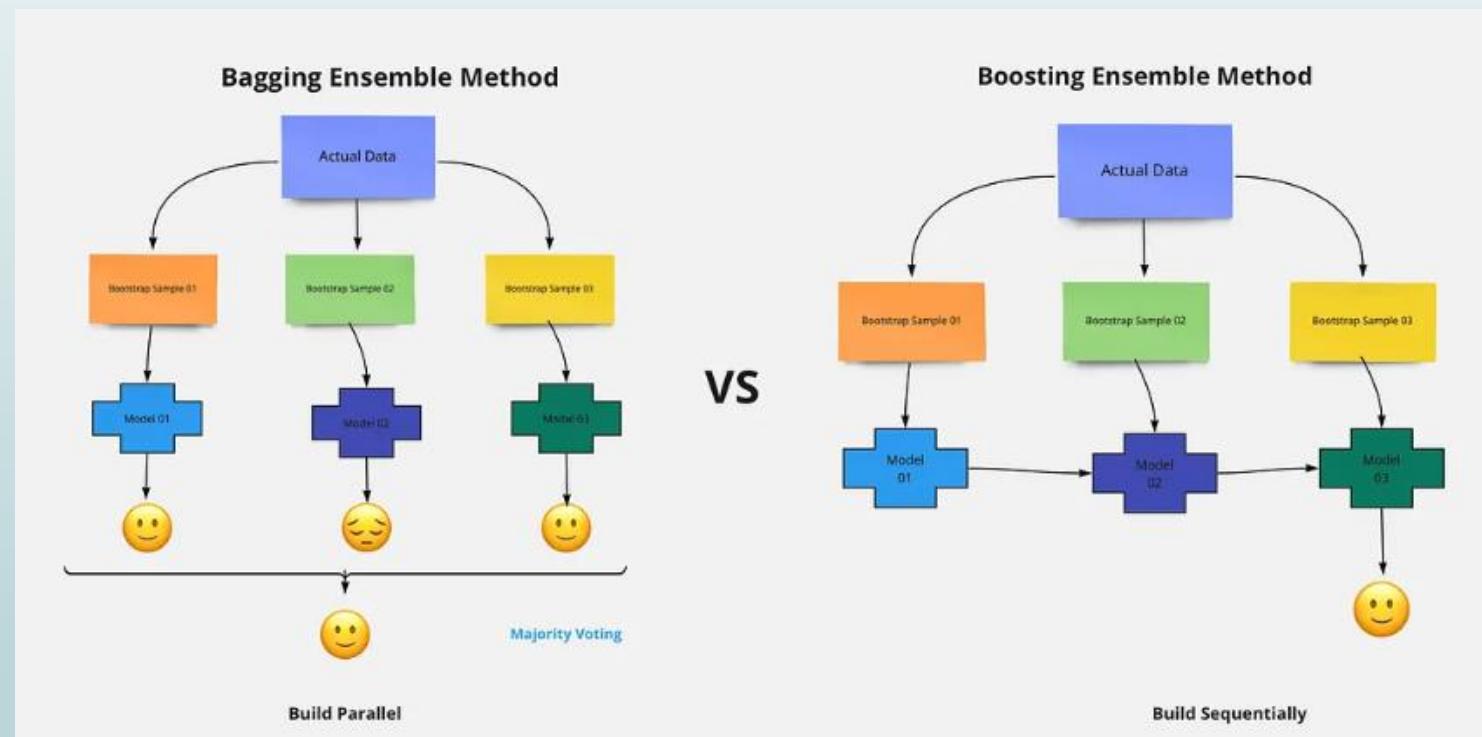
Veille technique : XGBoost

Apparu en 2016 dans une publication de Tianqi Chen et Carlos Guestrin.

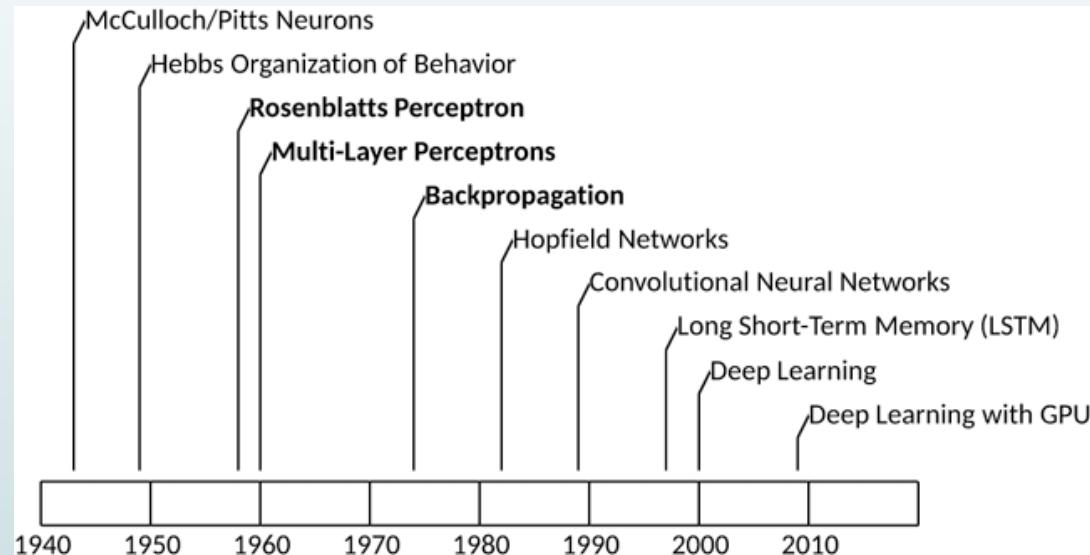
Entraînement de façon séquentielle plusieurs modèles et de les combiner successivement en corrigeant itération après itération

Le résultat de la prédiction est donc constitué de la prédiction de l'ensemble des arbres de décision enchaînés.

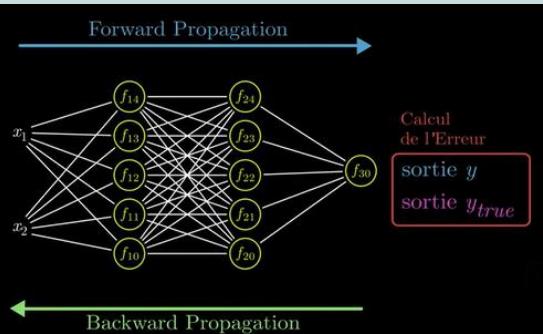
Cette méthode augmente les performances et la stabilité du modèle tout en minimisant sa variance.



Veille technique : Réseau de neurones



- 1. Forward Propagation
- 2. Cost Function
- 3. Backward Propagation
- 4. Gradient Descent



1-Forward Propagation : Circulation des données de la première couche jusqu'à la dernière afin de produire une sortie y .

2-Cost function : Calcul l'erreur de la sortie. Calculée avec y et la valeur réelle en utilisant la fonction coût.

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.

Création du modèle : méthodologie

Chaque technologie sera entraîné sur :

- différentes unités de temps (toutes les données ou seulement les 15 derniers mois)
- Avec et sans la variable de la surface du terrain
- Différents tests seront effectués (un modèle uniquement pour les maisons, un pour les appartements ou un modèle unique)

Une technologie pourra être abandonnée si le temps d'entraînement est trop long et/ou si les performances sont moins bonne qu'une autre technologie.

Les modèles devront être entraînés en validation croisées afin de d'optimiser le réglage des hyperparamètres et d'avoir un modèle plus fiable.

Les métriques, courbes de validations, courbes d'apprentissages et l'importance des variables devront être enregistrées afin de permettre l'interprétation des résultats

Le processus pour la création d'un modèle sera le suivant



Création du modèle : Récupération des 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
)
```

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

```
# 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
```

Variables qui seront récupérées .

```
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 AS 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)
```

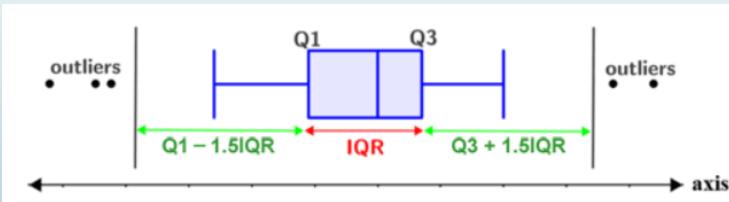
Jointure entre les différentes tables.

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.

	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

Création du modèle : Nettoyage des données



$$\text{Seuil haut} = Q3 + 1,5 \times (Q3-Q1)$$

Q3 est le montant pour lequel 75% des biens ont une valeur inférieure.

Q1 est le montant pour lequel 25% des biens ont une valeur inférieure.

Q3-Q1 est appelé l'intervalle interquartile.

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

Nombre d'outliers : 60157
Nombre de commune avec des outliers : 5680
Nombre de communes contenant des outliers et pour lesquels il reste 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 outliers et après filtrage : 905178
Il y a donc eu 8.47% de lignes supprimées après filtrage

Création du modèle : Préparation des données

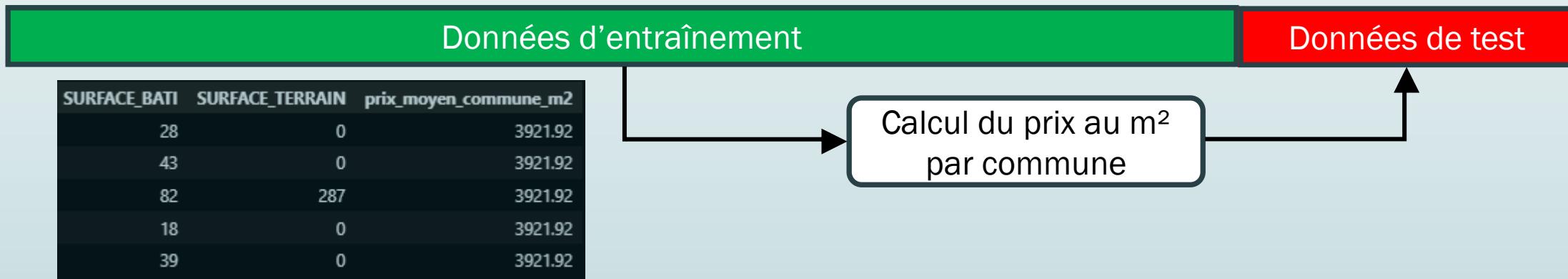


Splitage des données :

Janvier 2022

Décembre 2022

Mars 2023



Normalisation des données :

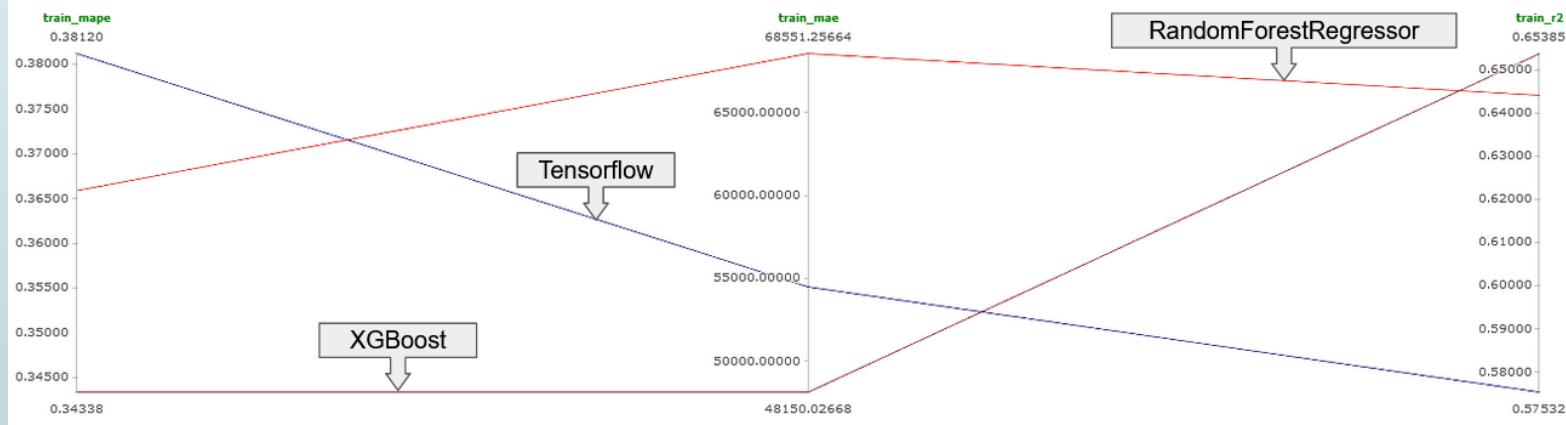
Une fonction a été créée afin de labelliser (si besoin) et de standardiser les données, le but est de mettre à la même échelle toutes les variables.

SURFACE_BATI	SURFACE_TERRAIN	prix_moyen_commune_m2
-1.348370	-0.116862	0.897053
-0.956129	-0.116862	0.897053
0.063698	-0.094724	0.897053
-1.609864	-0.116862	0.897053
-1.060726	-0.116862	0.897053

Création du modèle : Entraînement d'un modèle



Chaque entraînement a pu être logué avec Mlflow permettant ainsi de visualiser les hyperparamètres du modèle, ses métriques et les différents graphiques pour l'interprétation.



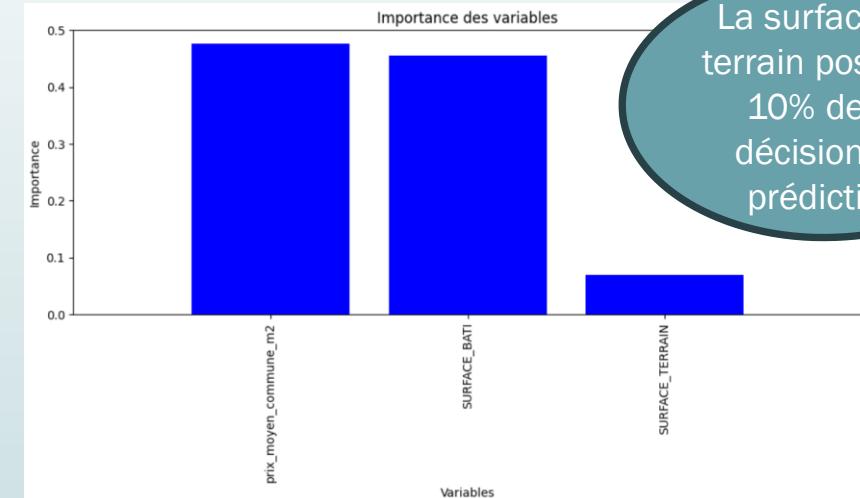
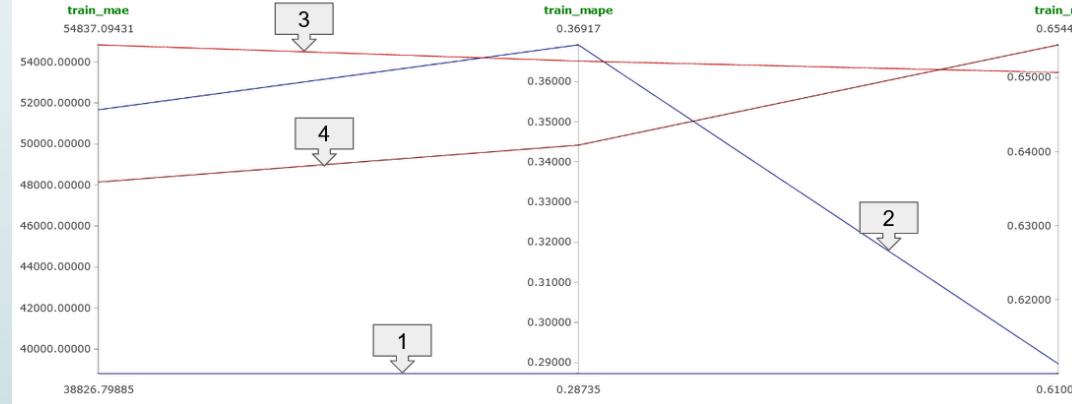
Suites aux tests sur les différentes technologies, la méthode d'entraînement avec XGBoost a été retenu pour ses performances et sa rapidité d'exécution.

Le model ainsi que les scalers et encoders sont stockés dans un bucket S3 permettant l'accès depuis n'importe quelle machine. Une base de données Postgre permet d'enregistrer les chemins d'accès.

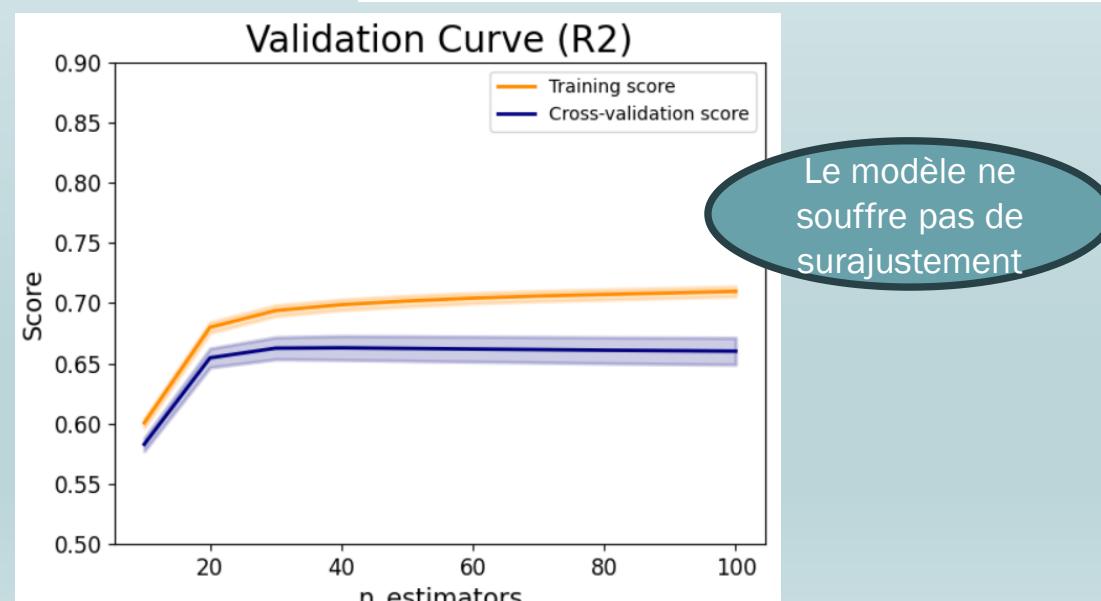
Création du modèle : Interprétations

Le modèle N°4 est le plus résilient

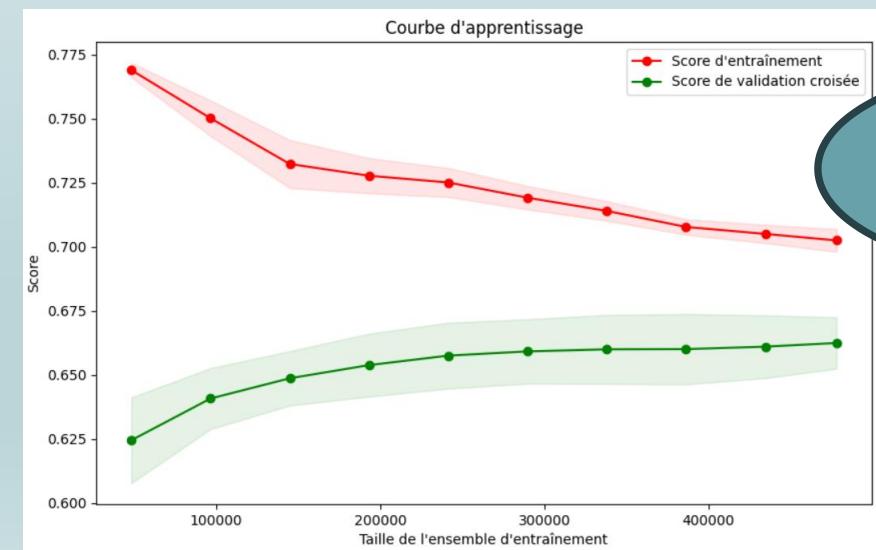
N° de ligne	1	2	3	4
Type de bien	Appartement	Appart. + Maison	Maison	Appart. + Maison
Surface terrain	non	non	oui	oui



La surface du terrain possède 10% de la décision de prédiction



Le modèle ne souffre pas de surajustement



Le modèle dispose de suffisamment de données

Mise en place de l'environnement CI/CD



```
name: Tests CI

on: ['push']

jobs:
  build:
    runs-on: ubuntu-latest
    env:
      DB_HOST: ${{ Secrets.DB_HOST}}
      DB_USER: ${{ secrets.DB_USER }}
      DB_PASSWORD: ${{ secrets.DB_PASSWORD }}
      DB_PORT: ${{ secrets.DB_PORT }}
      AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
      AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
      URL_POSTGRE: ${{ secrets.URL_POSTGRE }}

    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Set up Python
        uses: actions/setup-python@v2
        with:
          python-version: 3.9

      - name: Install dependencies
        run:
          python -m pip install --upgrade pip
          pip install -r app/requirements.txt

      - name: Run tests
        run:
          python -m unittest discover -s app -p '*test.py'
```

```
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()
```

The screenshot shows the GitHub Actions interface. On the left, the 'Repository secrets' sidebar is open, listing various secrets like AWS keys, database credentials, and URLs. The 'Actions' tab is selected. On the right, a workflow named 'test.yml' is shown for a push event. The workflow consists of several steps: setting up the job, checking out code, setting up Python (version 3.9), installing dependencies (using pip), and running tests. The 'Run tests' step is expanded, showing the command: 'Run python -m unittest discover -s app -p "*test.py"'. The workflow has completed successfully, taking 39 seconds.

Mise en place de l'environnement



Deployment method

Heroku Git Use Heroku CLI GitHub Connected Container Registry Use Heroku CLI

App connected to GitHub
Connected to rastakoer/certif_app_immo by rastakoer
Disconnect...
Releases in the activity feed link to GitHub to view commit diffs
Automatically deploys from application

Automatic deploys
You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.
Automatic deploys from application are enabled
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.
Wait for CI to pass before deploy

Overview Resources Deploy Metrics Activity Access Settings

App Information
App Name: miflowimmoappkevleg
Region: Europe
Stack: container
Framework: Container
Heroku git URL: https://git.heroku.com/miflowi...

Config Vars
Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.
Config Vars:
ARTIFACT_STORE_URI
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
BACKEND_STORE_URI

```
build:  
  docker:  
    web: app/Dockerfile
```

```
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
```

Activity Feed

kev.le.grand@orange.fr: Deployed 5bf951e2 Today at 11:15 AM · v75

kev.le.grand@orange.fr: Build succeeded Today at 11:13 AM · View build log

Front-end

Maquette



LOGO
APPLICATION

Paramètres pour l'estimation :

Région →
Département →
Commune →
Type de bien →
Nombre de pièces →
Surface habitable →
Surface terrain →

Estimer →

MAXIMUM 200 000€
MINIMUM 100 000€

Statistiques et évolution du prix dans la région souhaitée :

Le montant estimé par rapport au caractéristiques fournies est de : 175 000€

Voici les biens similaires vendus dans la commune recherchées :

IMMO APP

Sélectionnez une région
Normandie

Sélectionnez un département
Calvados

Sélectionnez une commune
Caen

Sélectionnez le type de logement :
Appartement

De combien de pièces est composé le bien
3

Surface habitable
65

Valider

Application

Le bien est estimé à 176054 €

Statistiques sur la commune de Caen pour un appartement sont :

Prix moyen du m ²	Montant moyen	Montant minimum	Montant maximum
3066 €	147886 €	1 €	42395720 €

Voici les Appartements vendus dans la commune de Caen

Back-end : Niveau utilisateur



Login Immoapp

Connexion

Nom d'utilisateur

Mot de passe

Se connecter

Créer un nouvel utilisateur

Nouveau nom d'utilisateur

Nouveau mot de passe

Créer utilisateur

Utilisation de hashlib pour le cryptage de mot de passe

Accès à différentes fonctions suivant le niveau utilisateur

Création d'une table pour stocker les informations utilisateurs

config_bdd.py
app.py
bdd_grafan.ipynb
functions.py

C12

p.45



application

id	username	password	level
5	kevin3	0f1b445587009b8a7dd2a4c1270f2f9a	1.0
7	new	22af645d1859cb5ca6da0c484f1f37ea	NaN

kevin3



Cette application web permet d'estimer la valeur immobilière d'une commune métropolitaine.

Le modèle a été entraîné sur une période de 12 à partir du 21/01/2022.

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 3000 transactions

Grafana

new



Cette application web permet d'estimer la valeur immobilière d'une commune métropolitaine.

Le modèle a été entraîné sur une période de 12 à partir du 20/01/2022.

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 3000 transactions

Back-end : API



API prédictions d'un bien immobilier 0.1.0 OAS 3.1

/openapi.json

Api développée par Kevin LE GRAND

Obtenez une prédition de la valeur d'un bien grâce aux éléments suivants :

- La surface d'un bien
- La surface du terrain
- Le prix au m² de la commune

POST /predict Prédictions

Route permettant de réaliser une prédition

Parameters

No parameters

Request body required

```
{  
    "SURFACE_BATI": 250,  
    "SURFACE_TERRAIN": 1000,  
    "prix_moyen_commune_m2": 2123.13  
}
```

Code	Details
200	Response body <pre>{"reponse": 394082.6875}</pre>

Appel depuis l'application

```
#//////////  
#           échange avec l'API  
#//////////  
def api_predict(data: dict) -> dict:  
    """  
        Fonction permettant de faire l'appel à l'api et de recevoir une prédition  
  
    Args :  
        - data (dict) : Dictionnaire avec les valeurs saisies par l'utilisateur  
  
    Returns  
        -  
    """  
    response = requests.post('https://apiimmoappkevleg-7337fa262339.herokuapp.com/predict',  
                             json=data)  
  
    return response.json()
```

La création d'une API permet

Un développement indépendant de l'application

La réutilisation par une autre application

Avantages FastAPI

Documentation claire pour les utilisateurs

Simple d'utilisation

Monitoring : Fonctions et requêtes



```
#///////////  
#           enregistrement de la recherche dans grafana  
#///////////  
  
def log_grafana() -> None:  
    """  
  
    Fonction permettant d'enregistrer les paramètres de la recherche dans Grafana.  
  
    Cette fonction ne prend aucun argument en entrée et utilise st.session_state  
    pour stocker les données dans postgres.  
  
    Cette fonction ne produit aucune sortie dans l'application.  
    """  
  
    database.add_row("kpis", date_pred=datetime.now(),  
                    type_de_bien=f"{st.session_state.type_de_bien}",  
                    region=f"{st.session_state.region}",  
                    departement=f"{st.session_state.departement}",  
                    commune=f"{st.session_state.commune}",  
                    surface_bati=f"{st.session_state.surface_bati}",  
                    surface_terrain=f"{st.session_state.surface_terrain}", #///////////  
                    nb_piece=f"{st.session_state.nb_pieces}",  
                    user=f"{st.session_state.username}",  
                    pred=float(st.session_state.pred))  
  
    return
```

Args:
- texte (str) : Texte de l'erreur avec try: ... except Exception as e

Returns:
- None

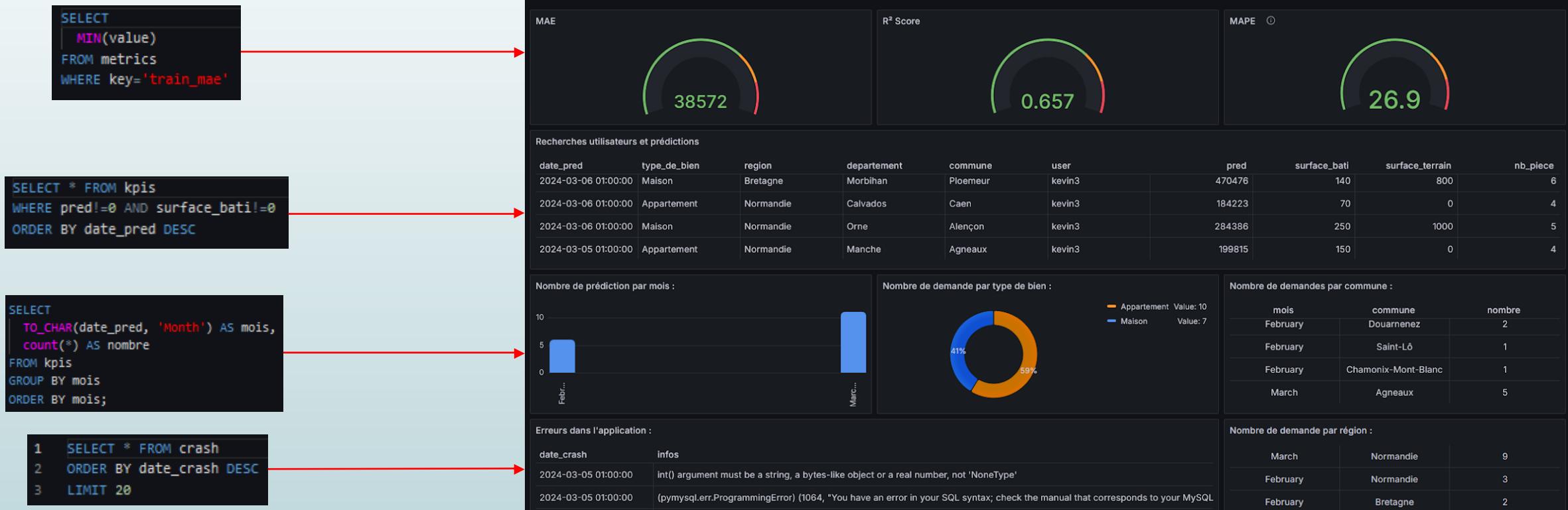
Remarque:

Cette fonction a pour but de visualiser les erreurs dans Grafana puis d'envoyer un mail à l'aide de grafana aux développeurs.

```
"""  
  
database.add_row('crash',  
                 date_crash=datetime.now(),  
                 infos = texte)  
  
return
```

```
#///////////  
#           enregistrement du status de l'application  
#///////////  
  
def log_status_grafana(status : int) -> None:  
    """  
  
    Fonction permettant d'enregistrer un status de l'application dans Postgre.  
  
    Args:  
    - status (int) : 0 si une prédiction a réussi sinon 1  
  
    Returns:  
    - None  
  
    Remarque:  
  
    Cette fonction a pour but de visualiser les erreurs dans Grafana puis d'envoyer un mail à l'aide de grafana aux développeurs.  
    """  
  
    database.add_row('app_status',  
                    date_status=datetime.now(),  
                    status = status)  
  
return
```

Monitoring : Aperçu



2. Define query and alert condition

Define query and alert condition [Need help?](#)

A mlflow

Format: Table

```
1 SELECT STATUS FROM app_status
2 ORDER BY date_status DESC
3 LIMIT 1
```

B Threshold

Takes one or more time series returned from a query condition.

Input A

IS ABOVE 0

GrafanaCloud > status application

Une erreur a été détectée dans l'application.
Pour visualiser cette erreur, rendez vous sur l'application de monitoring : <https://kevinlegrand.grafana.net/public-dashboards/cde8ec56de054eb295d3f68e0039aa63>

Démo de l'application

The screenshot displays the IMMO APP application interface. On the left, a sidebar menu lists various locations: Bourgogne-Franche-Comté, Bourgogne, Bourg-en-Bresse, Mâcon, Chalon-sur-Saône, and Franche-Comté. Below these are dropdown menus for 'Type de logement' (House, Apartment), 'Surface' (100m²), 'Nombre de pièces' (3), and 'Orientation' (South). A large blue button at the bottom of the sidebar says 'Valuer'. The main content area features a large blue circle with the text 'IMMO APP'. Above the map, a bold statement reads 'Le bien est estimé à 184132 €'. Below this, a sub-section titled 'Statistiques sur la commune de Frahier-et-Chatebier pour une maison sont :' includes four small graphs. A larger section below is titled 'Voici les Maisons vendus dans la commune de Frahier-et-Chatebier' and shows a map of the commune with several green dots indicating sold properties. One dot is highlighted with a yellow box containing detailed information: 'Ville : Frahier-et-Chatebier', 'Surface : 100 m²', 'Nombre de pièces : 3', 'Orientation : Sud', 'Prix : 184 132 €', and 'Date de vente : 2021-07-10'. At the bottom of the map area is a 'Retour' button.

Conclusions

La réalisation du projet

Enrichissant par la diversité des tâches à accomplir et les technologies utilisées

L'intérêt de tous les outils utilisés même si le projet a été réalisé seul de bout en bout

Les connaissances acquises durant cette formation

Les difficultés rencontrées

L'idée de projet et les données pouvant inclure toutes les connaissances acquises

Les délais à respecter, ce projet n'étant pas un projet d'alternance

Le manque d'informations sur les biens vendus

La gestion des coûts

Améliorations et perspectives d'évolution

Réaliser ce projet pour une agence avec des biens mieux décrits

Créer un second modèle pour les grandes agglomérations et les villes touristiques

Poursuivre les études en IA

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 prédiction 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

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 prédiction 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

MERCI
DE VOTRE
ATTENTION

Questions/Réponses