

RAPPORT FINAL PYTHON
ANALYSE DE DONNÉES DVF

CHESTERIKOFF AYMERIC | DANSI LEXIA | ROMMELFANGEN SASHA

Mai 2023

Bienvenue dans ce rapport passionnant qui met en lumière notre incroyable travail en trinôme sur l'analyse des données DVF (Demandes de valeurs foncières) ! Inspirés par le TD 7.2 axé sur la visualisation des données COVID, nous plongeons cette fois-ci dans l'exploration des transactions immobilières des cinq dernières années.

Notre terrain de jeu s'étend sur le territoire métropolitain et les DOM-TOM, à l'exception de l'Alsace, de la Moselle et de Mayotte. Les données que nous avons exploitées sont mises à disposition en OpenData sur le site <https://www.data.gouv.fr/fr/datasets/demandes-de-valeurs-foncieres/>.

Préparez-vous à une aventure captivante à travers le monde fascinant des biens immobiliers !



Plan du rapport :

- Introduction : contexte ,objectif et répartition du projet

I. ANALYSE DES DONNEES

- A. Récupération des dataframe
- B. Chargement et nettoyage des données
- C. Explications du code

II. NOTEBOOK

- voir notebook détaillé (fichier HTML)

III. DJANGO

- Présentation du projet utilisant le framework Django pour la création d'une vitrine d'analyse et de visualisation des données

A. Analyse de données

Introduction

Ce rapport présente notre travail en trinôme sur l'analyse des données DVF
(Demandes de valeurs foncières).

Notre objectif était d'effectuer une analyse approfondie de ces données pour différentes années, en utilisant Python pour créer des visualisations et interprétations pertinentes.

Les données DVF sont librement accessibles et nous avons exploité plusieurs critères pour explorer et représenter ces données.

Répartition des tâches et ratio de contribution :

Notre équipe était composé de trois membres, et nous avons réparti les tâches au volontariat et de manière assez équitable.

- > Deux membres se sont concentrés sur le notebook (analyse des données, nettoyage des données , création des graphiques) . + rapport
- > Le troisième membre s'est consacré à la partie Django, permettant la création d'une interface interactive pour notre projet.

Chaque membre du trinôme a apporté des efforts proportionnels à la réalisation du projet selon ses compétences , en effectuant des recherches approfondies et en contribuant aux lignes de code en Python. Globalement, nous estimons le ratio suivant :

Aymeric : 45%

Sacha : 28%

Lexia : 27 %

À travers ce rapport, nous souhaitons présenter notre travail collaboratif et les résultats obtenus lors de notre analyse des données DVF. Nous sommes fiers de notre implication et de la qualité de notre travail, et nous espérons que ce rapport reflètera notre engagement et nos compétences dans ce domaine.

I. ANALYSE DES DONNEES

A. Récupération des données

- Les dataframe récupérées contenaient des millions de données non triés. Nous avons mis un peu de temps avant de les comprendre et de les interpréter . Une fois que nous avons bien assimilé nous avons pu mettre en place un code en python pour trier les fichiers et ne garder que les informations qui nous intéressaient .

• B. Chargement et nettoyage des données

Cette séquence de code constitue la première étape de l'analyse des données DVF. Les données ont été chargées, nettoyées, filtrées et préparées pour des analyses ultérieures. Le fichier de sortie au format .txt contient les colonnes sélectionnées qui seront utilisées dans les étapes suivantes de l'analyse (pour le notebook)

The screenshot shows a Mac OS X desktop with several windows open. In the foreground, there's an iPython console window with some code and output. Behind it, a Jupyter Notebook cell contains Python code for reading a CSV file and creating a DataFrame. To the right of the code, a 'split_data - DataFrame' viewer window is open, displaying a table with columns: Index, /Q voie, Code postal, Ville, Commune, Code departeme, and type de bien. The table has 9 rows, indexed from 986 to 990. The data includes entries like YOLLET 1310 POLLAT A 1109, RVE 1310 POLLAT A 746, and YOLLET 1310 POLLAT A 1110. Below the table, there are buttons for Format, Resize, Background color, Column min/max, Save and Close, and Close. At the bottom of the viewer, it says '5 rows x 44 columns' and 'In [26]'. The status bar at the bottom of the screen shows 'conda: base (Python 3.10.9)' and 'Completions: conda(base)'. The overall environment suggests a data science workflow.

```
1 #!/usr/bin/python
2 # Importer les bibliothèques nécessaires
3 import pandas as pd
4 import numpy as np
5 import matplotlib.pyplot as plt
6 from sklearn import datasets
7
8 # Charger le fichier dans un DataFrame
9 #data = pd.read_csv('valeursfoncieres-2018.csv')
10 data = pd.read_csv('valeursfoncieres-2018.txt')
11
12 # Diviser chaque ligne en utilisant le symbole ","
13 split_data = data['Identifiant du document'].str.split(',')
14
15 # Nommer chaque colonne créée
16 split_data.columns = ['colonne' + str(i) for i in range(1, len(split_data[0]) + 1)]
17
18 # Concaténer les données d'origine avec les nouvelles colonnes
19 data = pd.concat([data, split_data], axis=1)
20
21 # Sélectionner toutes les colonnes à partir de la 7e
22 data = data.iloc[:, 7:]
23
24 # Supprimer les colonnes vides
25 data = data.dropna(subset=['colonne1', 'colonne2'])
26
27 # Créer un dictionnaire pour mapper les noms
28 mapping = {'colonne3': 'No disposition',
29             'colonne4': 'Date mutation',
30             'colonne5': 'Nature mutation',
31             'colonne6': 'Type local commercial',
32             'colonne7': 'No vote',
33             'colonne8': 'Code postal',
34             'colonne9': 'Ville',
35             'colonne10': 'Commune',
36             'colonne11': 'Code departement',
37             'colonne12': 'Type de bien'}
```

C. Explications du code

Nous avons effectué un tri et un nettoyage des données de demandes de valeurs foncières en utilisant effectuant un code python

```
50 # Charger le fichier dans un DataFrame
51 data = pd.read_csv('C://Users//paulr//OneDrive//Sasha//ESILV//S6//Langage Python//TD//Data//valeursfoncieres-2018.txt', delimiter='|', dtype=dtype, decimal=',', nrow
52
53 # Supprimer les valeurs foncières inférieures à 10 000
54 data = data[data['Valeur fonciere'] > 10000]
55
56
57 # Sélectionner un nombre aléatoire fixe de lignes
58 random_rows = random.sample(range(len(data)), 100000)
59 data = data.iloc[random_rows]
60
61 # Supprimer les colonnes inutiles
62 cols_to_keep = ['No disposition', 'Date mutation', 'Nature mutation', 'Valeur fonciere',
63                 'No voie', 'B/T/Q', 'Code postal', 'Commune', 'Voie', 'Code departement',
64                 'No plan', 'No Volume', '1er lot', 'Surface Carréz du 1er lot', '2eme lot',
65                 'Surface Carréz du 2eme lot', '3eme lot', 'Surface Carréz du 3eme lot',
66                 '4eme lot', 'Surface Carréz du 4eme lot', '5eme lot',
67                 'Surface Carréz du 5eme lot', 'Nombre de lots', 'Code type local',
68                 'Type local', 'Identifiant local', 'Surface reelle bati',
69                 'Nombre pieces principales', 'Nature culture', 'Nature culture speciale',
70                 'Surface terrain']
71 data = data[cols_to_keep]
72
73 # Générer les fichiers .txt
74 data.to_csv("C://Users//paulr//OneDrive//Sasha//ESILV//S6//Langage Python//TD//Projets//Tri_2//tris2021.txt", sep='|', index=False)
75
76 # Afficher les 5 premières lignes du DataFrame mis à jour
77 print(data.head())
```

1. Nous avons chargé les données à partir d'un fichier CSV en utilisant la bibliothèque pandas. Nous avons spécifié le chemin d'accès au fichier, le délimiteur 'l' et le type de données pour chaque colonne.
2. Ensuite, nous avons éliminé les valeurs foncières inférieures à 10 000, ce qui nous permet de nous concentrer sur les transactions immobilières de plus grande valeur.
3. Pour réduire la taille des données, nous avons sélectionné un échantillon aléatoire de 1 millions de lignes (sur la capture d'écran il y a marqué 100000 afin de gagner du temps d'exécution pour nos essais) à partir du DataFrame initial. Cela nous permet d'avoir un ensemble de données plus gérable tout en conservant sa représentativité.

4. Nous avons ensuite supprimé les colonnes inutiles qui ne sont pas nécessaires pour notre analyse. Nous avons défini une liste de noms de colonnes à conserver et nous avons filtré le DataFrame en utilisant cette liste.

```
Users > babymama > Downloads > 📁 data 1.py > ...
4
5     dtype = {
6         'Identifiant de document': str,
7         'Reference document': str,
8         '1 Articles CGI': str,
9         '2 Articles CGI': str,
10        '3 Articles CGI': str,
11        '4 Articles CGI': str,
12        '5 Articles CGI': str,
13        'No disposition': int,
14        'Date mutation': str,
15        'Nature mutation': str,
16        'Valeur fonciere': float,
17        'No voie': str,
18        'B/T/Q': str,
19        'Type de voie': str,
20        'Code voie': str,
21        'Voie': str,
22        'Code postal': str,
23        'Commune': str,
24        'Code departement': str,
25        'Code commune': str,
26        'Prefixe de section': str,
27        'Section': str,
28        'No plan': str,
29        'No Volume': str,
30        '1er lot': str,
31        'Surface Carrez du 1er lot': float,
32        '2eme lot': str,
33        'Surface Carrez du 2eme lot': float,
34        '3eme lot': str,
35        'Surface Carrez du 3eme lot': float,
36        '4eme lot': str,
37        'Surface Carrez du 4eme lot': float,
38        '5eme lot': str,
39        'Surface Carrez du 5eme lot': float,
40        'Nombre de lots': int,
41        'Code type local': str,
42        'Type local': str,
43        'Identifiant local': str,
44        'Surface reelle bati': float,
45        'Nombre pieces principales': float,
46        'Nature culture': str,
47        'Nature culture speciale': str,
48        'Surface terrain': float
49
50
```

0

5. Enfin, nous avons généré un fichier .txt à partir du DataFrame traité en utilisant la méthode `to_csv` de pandas. Le fichier contient les colonnes sélectionnées et est enregistré avec le séparateur 'l' spécifié.

Cette approche nous permet de trier les données, de les nettoyer en supprimant les valeurs inférieures à 10 000 euros, de sélectionner un échantillon d'1 millions de lignes aléatoires afin que cela soit tout de même représentatif à l'échelle de la France , de supprimer les colonnes inutiles et de générer un fichier final pour nos analyses ultérieures.

II. LE NOTEBOOK

Cf : Livrable Fichier HTML annexe

III. DJANGO

Nous avons créé une interface Django afin de pouvoir mieux visualiser les données. Cette interface est dynamique et permet à l'utilisateur de sélectionner uniquement les graphiques sur l'année qu'il souhaite étudier à l'aide d'un bouton sur lequel peut cliquer l'utilisateur du site.

Notre interface Django est composé de 10 pages HTML, chacune de ces pages permet de visualiser une donnée graphique que l'on peut voir évoluer

sur différentes années (de 2018 à 2022). On peut ainsi observer les évolutions tendances dans l'immobilier au cours des dernières années.

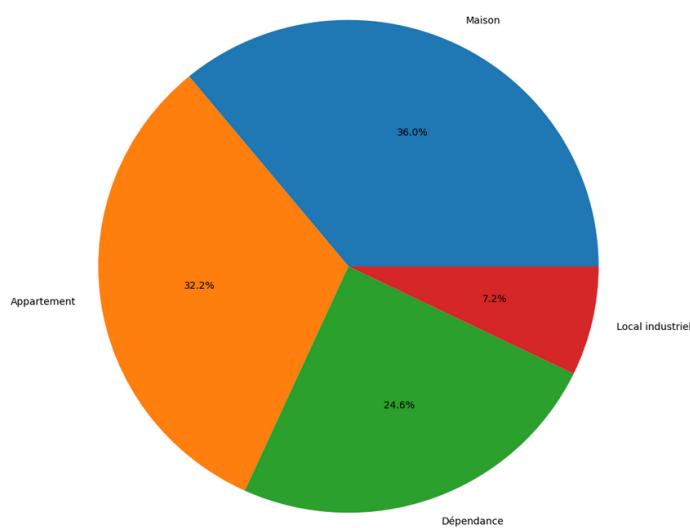
Répartition des types de biens immobiliers vendus lors des 5 dernières années

Choose a Model:
Choose a model:

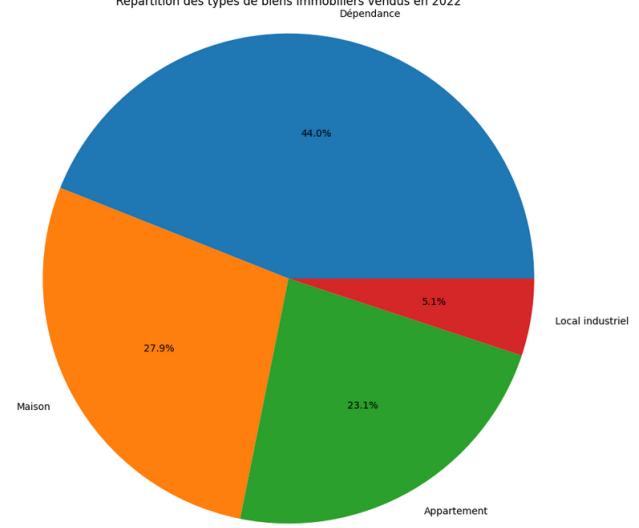
Répartition des types de biens immobiliers vendus lors des 5 dernières années

Choose a Model:
Choose a model:

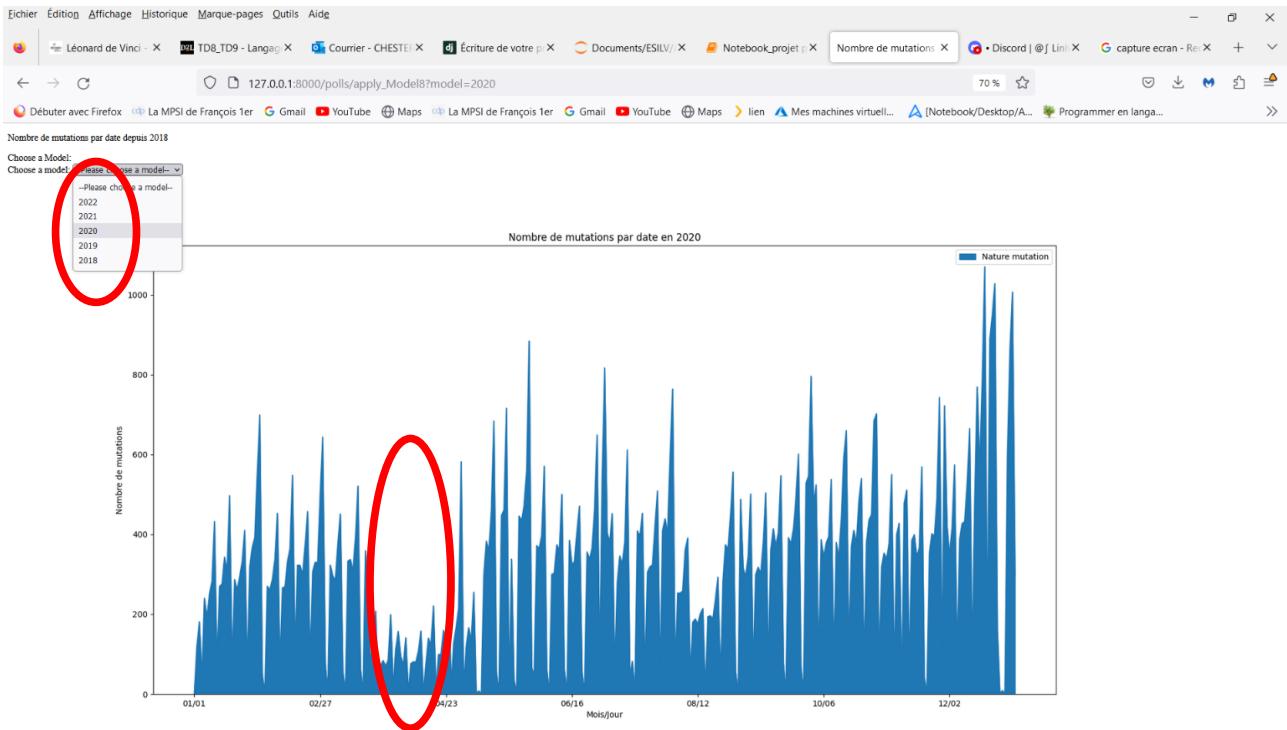
Répartition des types de biens immobiliers vendus en 2018



Répartition des types de biens immobiliers vendus en 2022

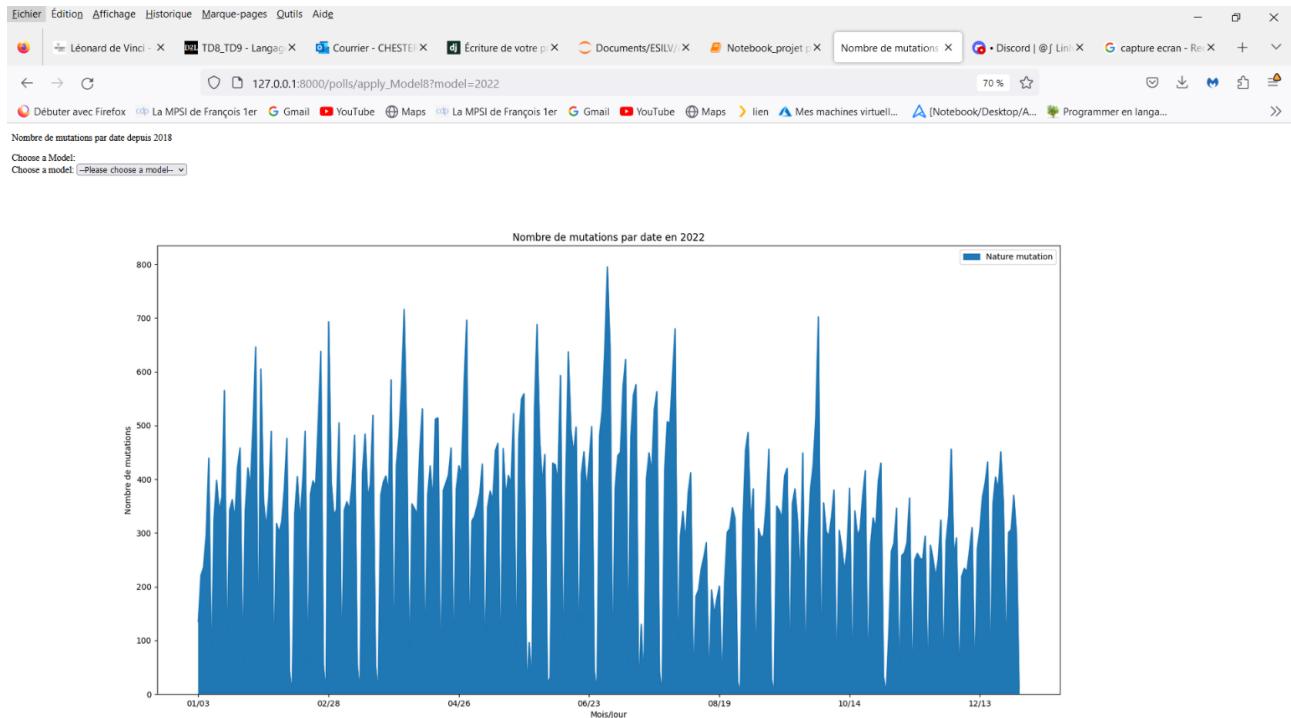


Par exemple sur ces graphiques Django ci-dessus, nous pouvons constater que depuis 2018, le pourcentage des ventes de maison et d'appartement a fortement diminué au profit des dépendances.

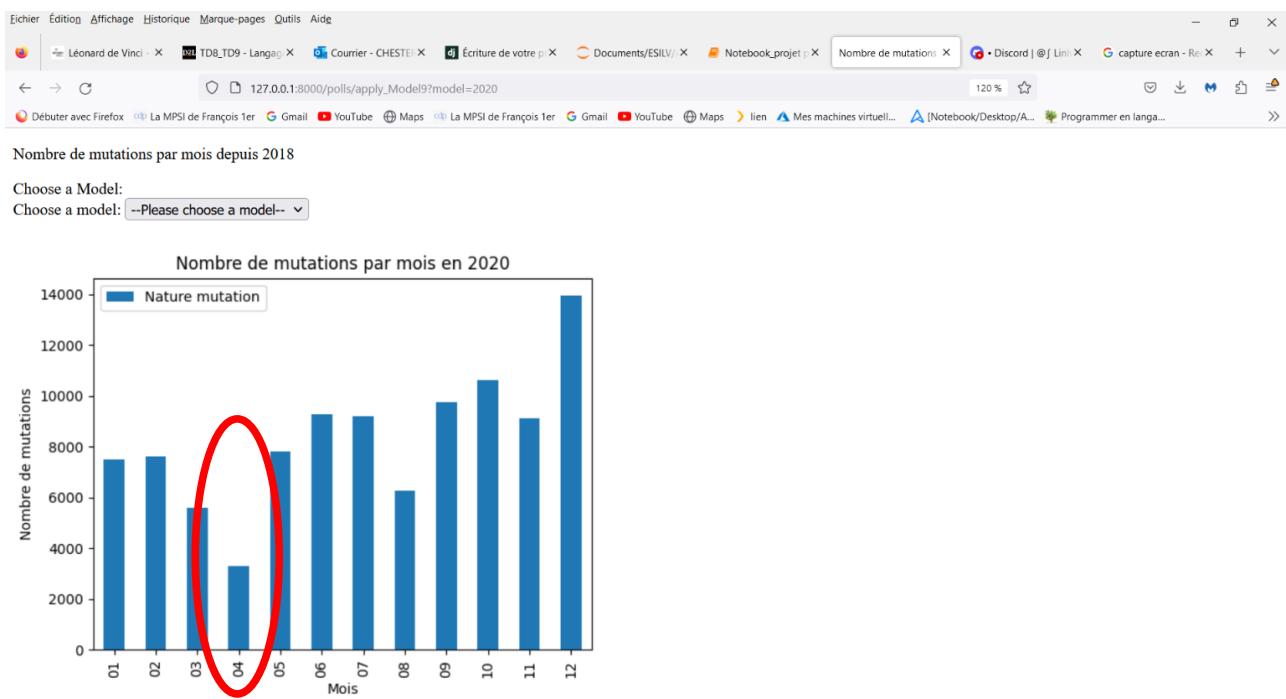


Sur le graphique ci-dessus, nous voyons qu'il est possible de sélectionner l'année d'affichage graphique que l'on veut observer sur Django à l'aide de création de form et label HTML ainsi que de méthode get en python.

De plus, ce graphique représentant le nombre de mutation au fil de l'année sélectionnée nous montre bien l'effet du confinement sur le nombre de mutation. En effet, l'ellipse entourée en rouge ci-dessus correspond au date du 1^{er} confinement de 2020 (mi-mars à début mai). Nous constatons sur cette période une très forte chute du nombre de mutation par rapport à la normale ou après lors du déconfinement. Ces données sont effectivement du au Covid et non à la période car nous n'observons pas de telle chute les autres années à la même période (voir ci-dessous pour 2022)



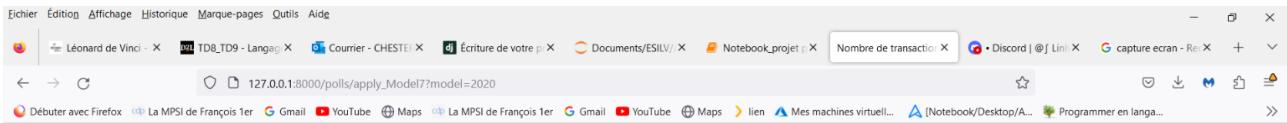
Cette baisse se constate également sur ce graphique de 2020 lors du mois d'avril qui était sous confinement en intégralité :



Alors qu'en 2022, sans confinement aucune baisse n'est observé lors du même mois.



Cette baisse se traduit donc par une chute du nombre de transactions lors du confinement :

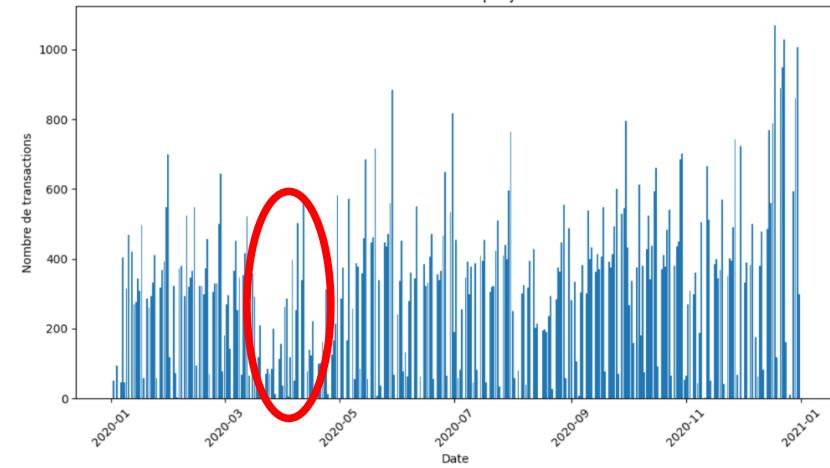


Nombre de transactions par jour depuis 2018

Choose a Model:

Choose a model:

Nombre de transactions par jour en 2020



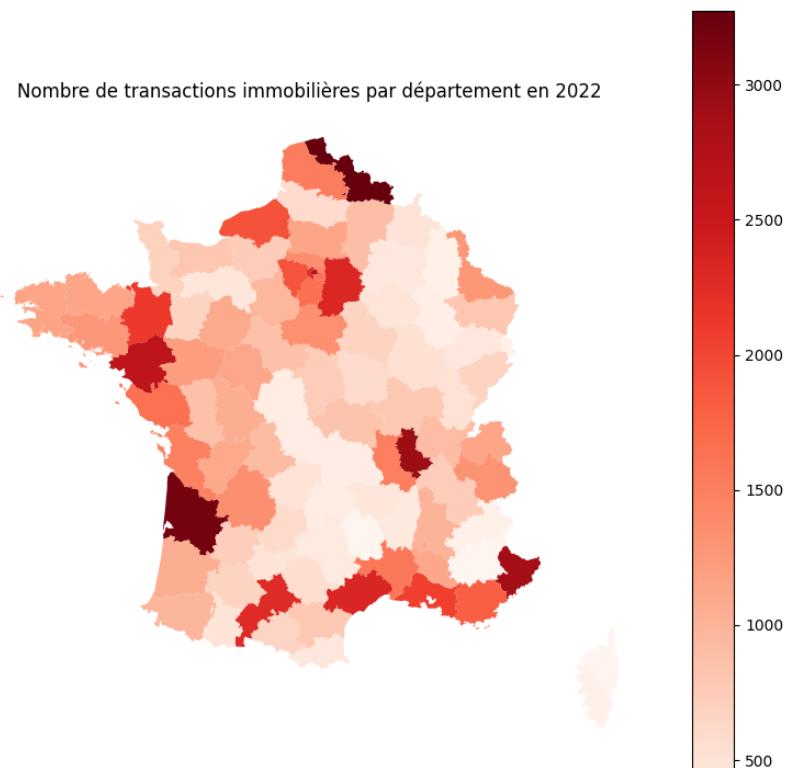
Mais paradoxalement, cette chute n'est pas observé à l'échelle du pays sur une année complète avec très peu de différence à l'échelle des département comme on peut le voir sur les 3 graphes ci-dessous de 2022, 2020 et 2018 :

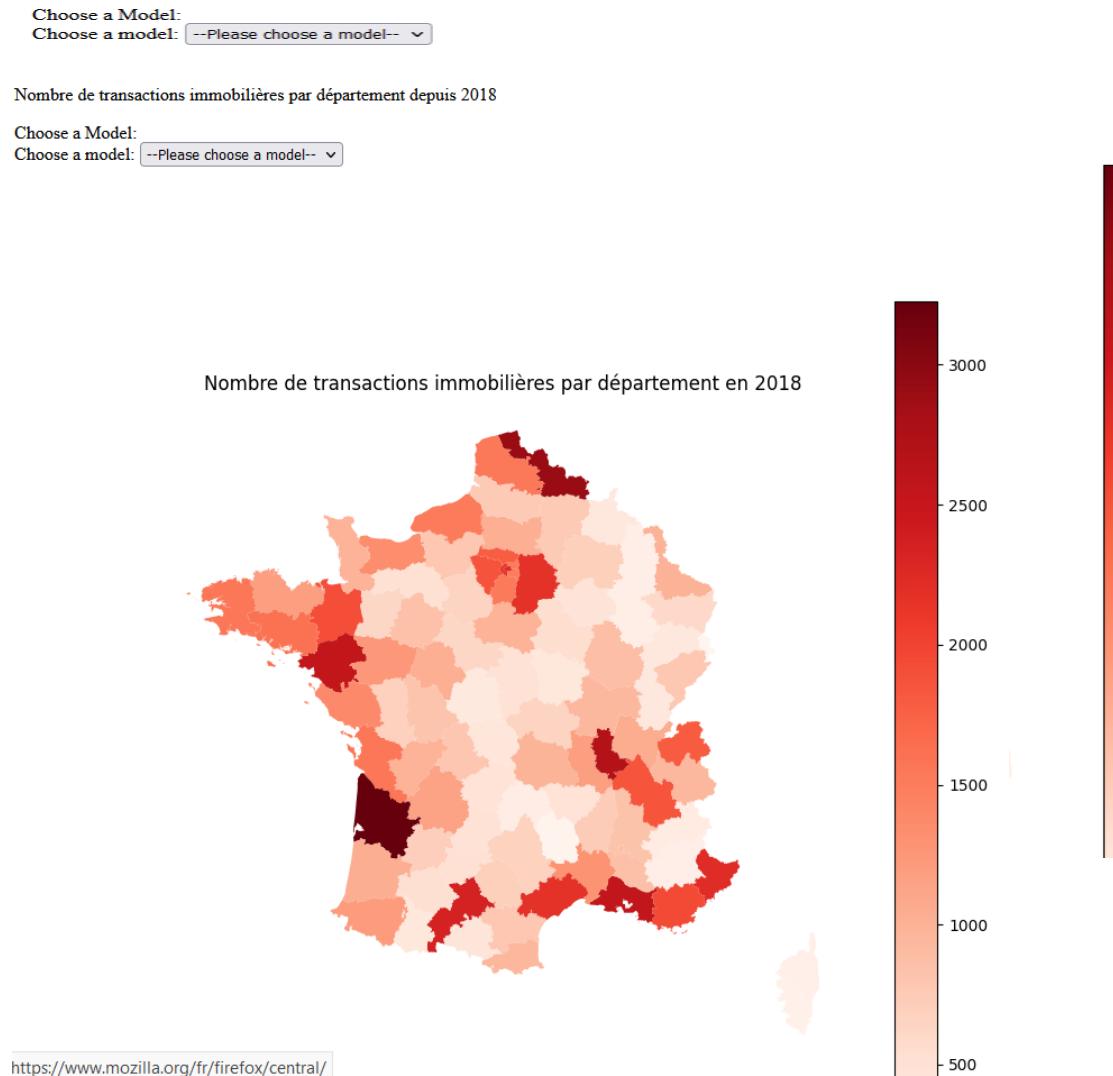
Nombre de transactions immobilières par département depuis 2018

Choose a Model:

Choose a model:

Nombre de transactions immobilières par département en 2022

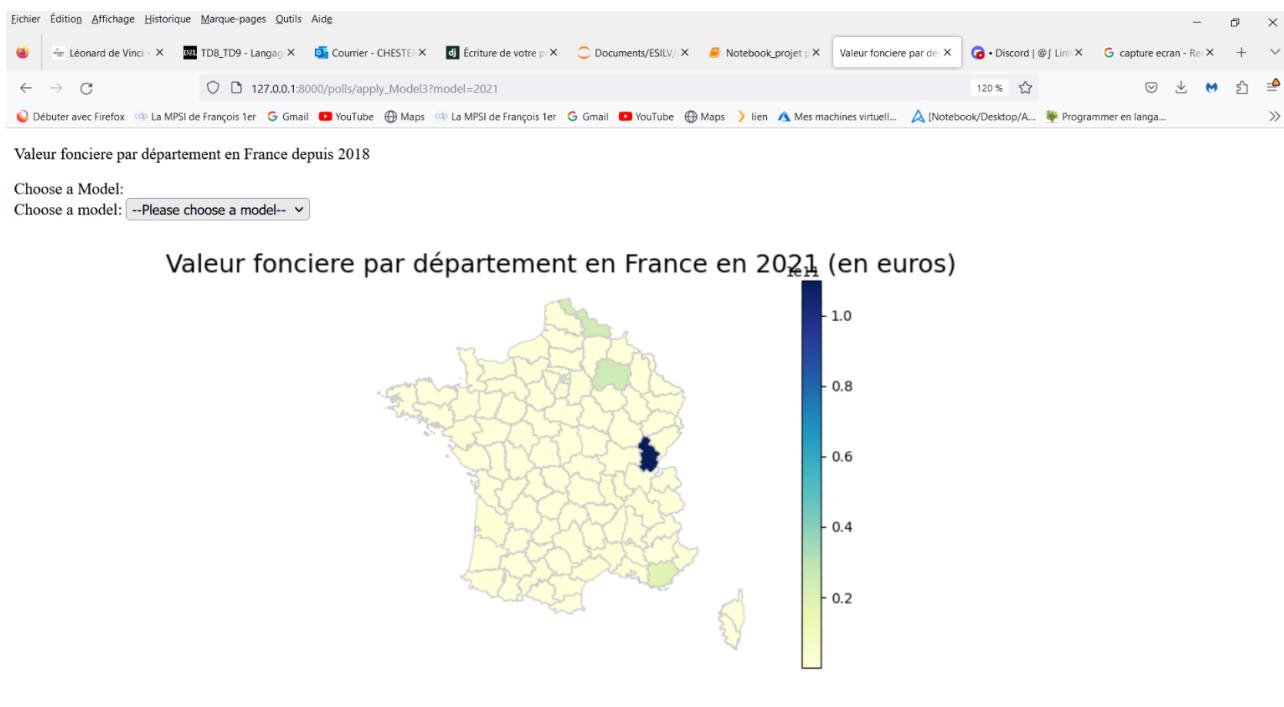




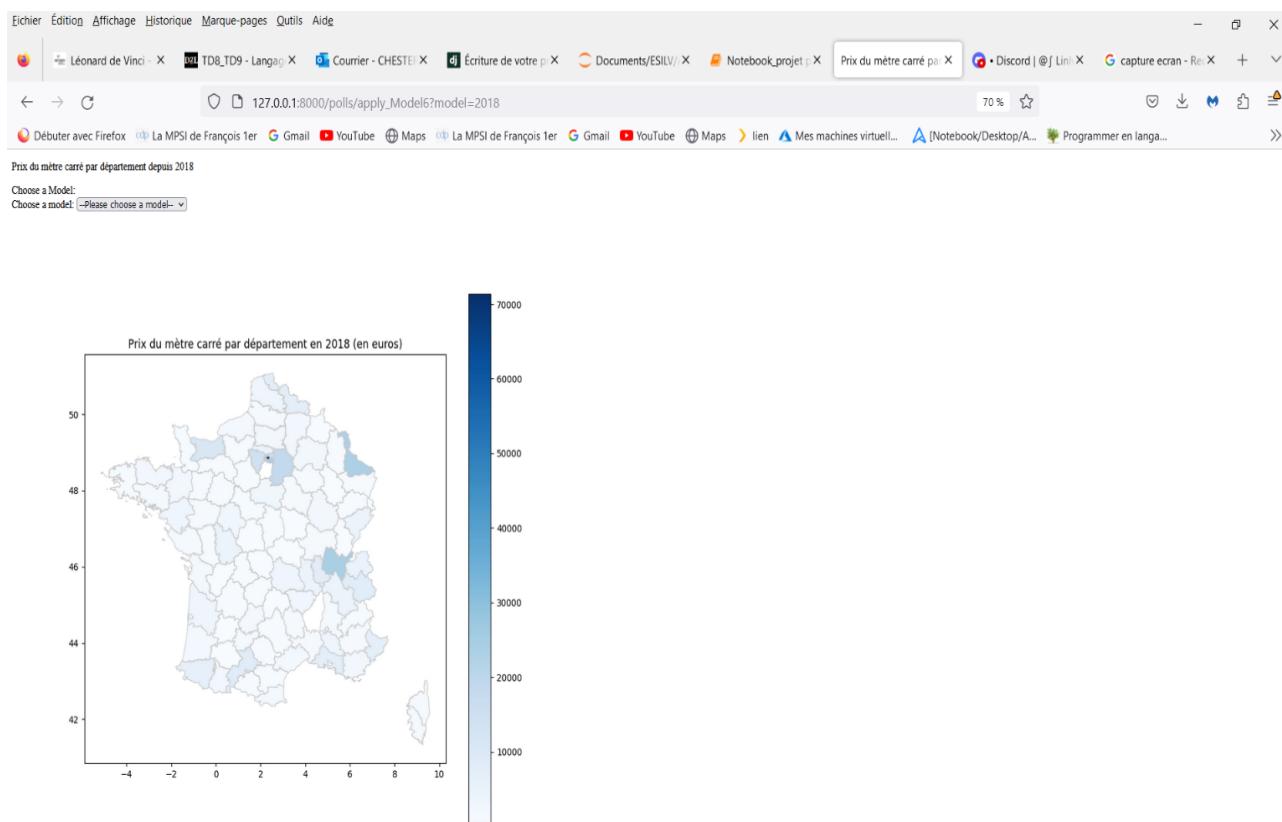
Pour ce qui du prix des valeurs foncières, nous pouvons constater que celle-ci étaient très élevés dans les grandes villes, notamment Paris où c'était le plus élevé de l'hexagone avant le Covid (voir ci-dessous).



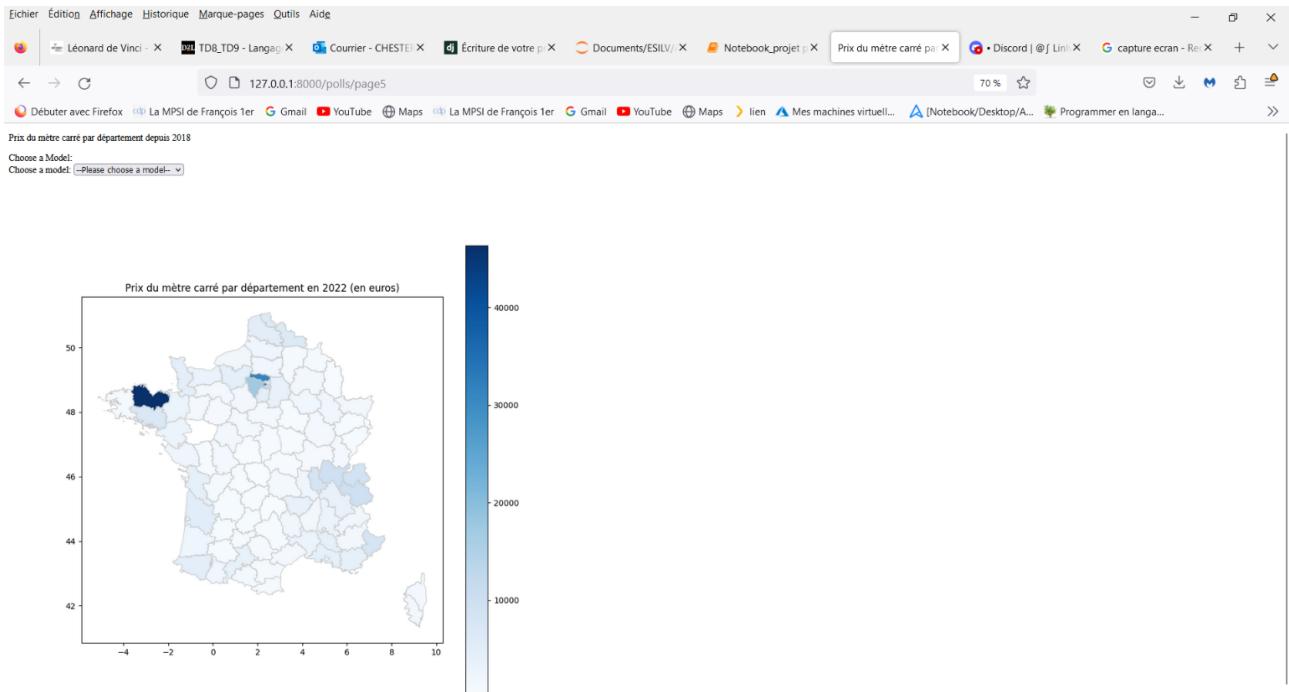
Cela n'est désormais plus le cas, comme nous pouvons le voir pour l'année 2021 ci-dessous, où certains département campagnard comme l'Ain ont eu une très nette hausse de leur valeur foncières.



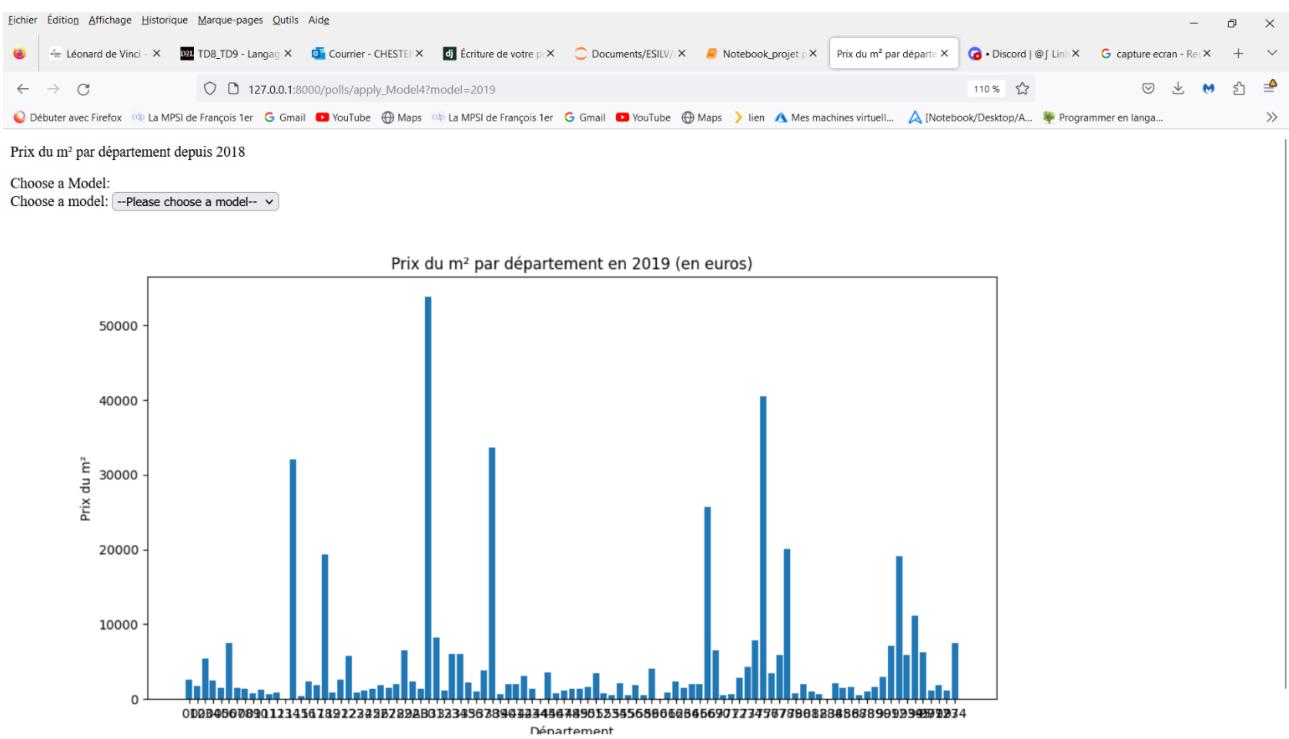
Ces données sont corroborées par celle de l'évolution du prix au mètre carré par département, Paris était le département le plus cher en 2018, mais ce n'est plus le cas aujourd'hui.

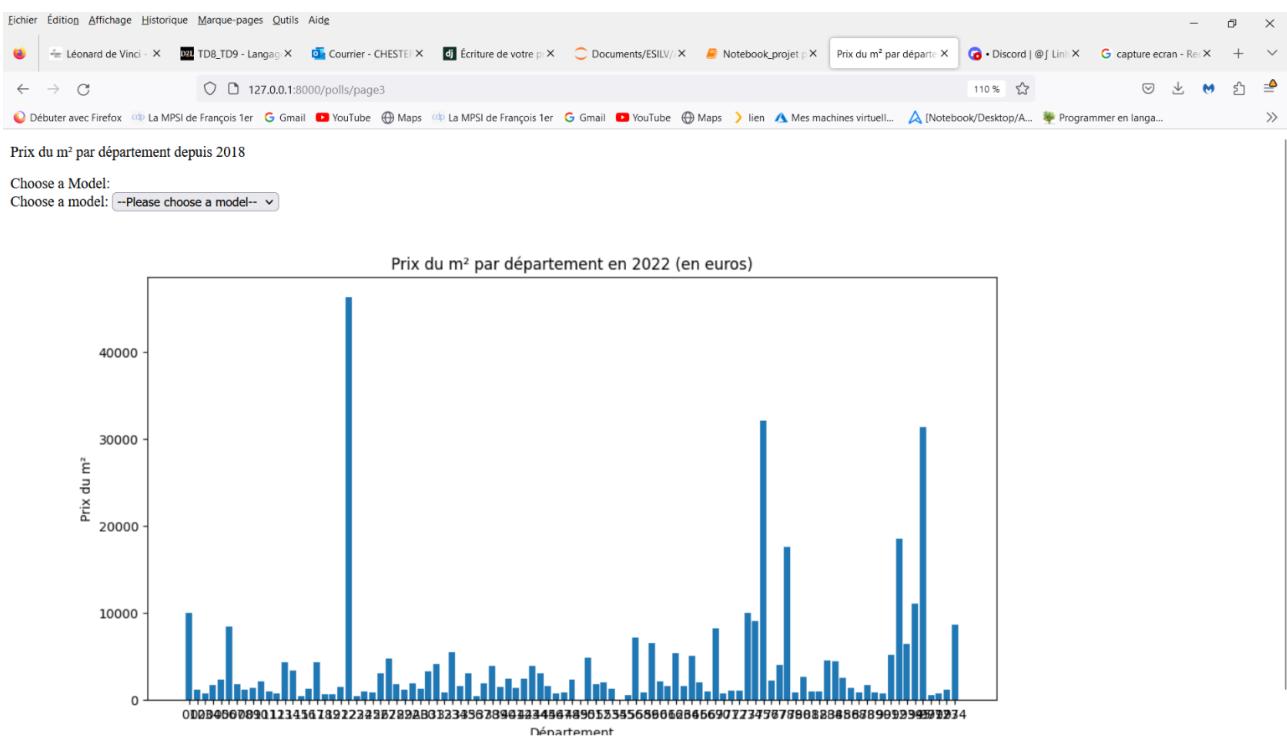


Cette carte de 2022, montre bien que Paris n'est plus le département avec le prix au mètre carré le plus élevé, de gros achat en 2022 dans les côtes d'Armor on néanmoins quelque peu faussée les données observées.

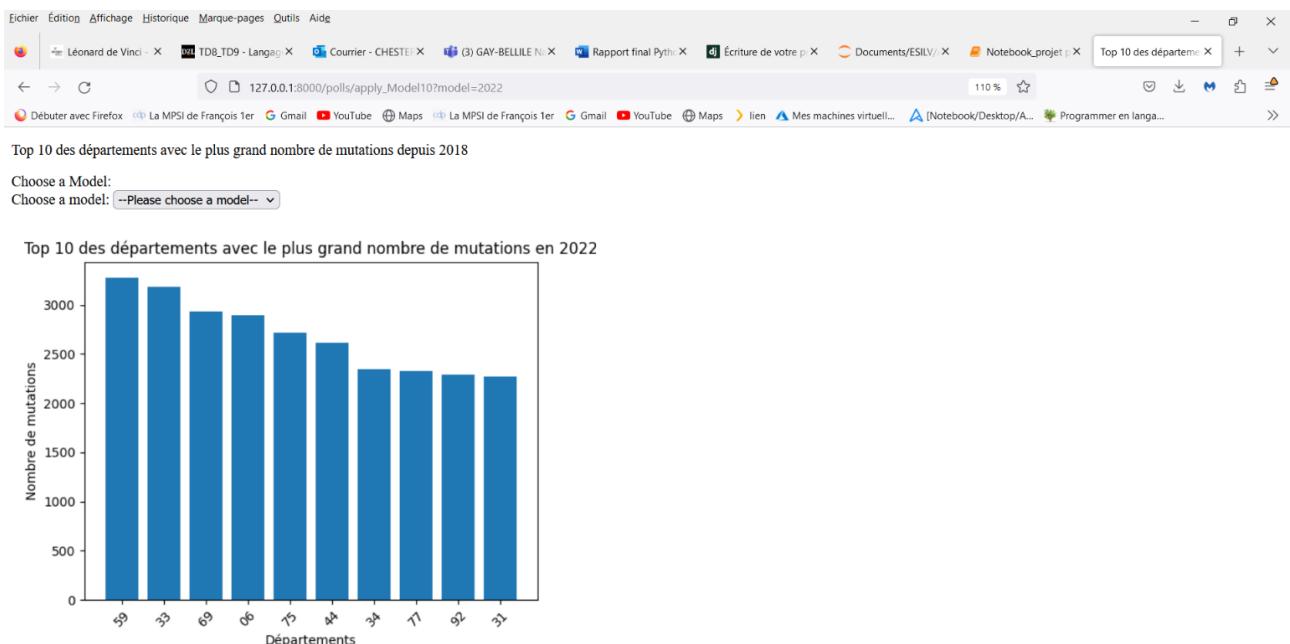


Même données sous forme d'un histogramme :





Enfin ce graphe, nous permet d'observer dans quel département, il y a le plus de mutation, inter-annuelle. Nous constatons très peu d'évolutions entre les années avant et après Covid. Cela montre peu de différence dans le nombre de déménagement par département avant et après Covid.



enfin,

L'intégralité du code Django se trouve dans le fichier zip joint

Merci pour votre temps

Conclusion :

Ce rapport présente notre travail en trinôme sur l'analyse des données DVF. Nous abordons le chargement, le nettoyage, l'interprétation et la combinaison des données, en mettant l'accent sur les visualisations pertinentes autour de l'année 2022 et les comparaisons avec d'autres années notamment avec la période de confinement en avril 2020 qui a fortement influencé le marché immobilier. Les livrables comprennent un notebook détaillé avec son exportation en HTML, ainsi qu'un projet utilisant le framework Django pour une vitrine dynamique d'analyse et de visualisation des données DVF.

Nous avons aussi implémenter l'interprétation de nos graphiques

