UNIVERSITÉ PARIS I PANTHÉON-SORBONNE

Logements  dans les grandes villes

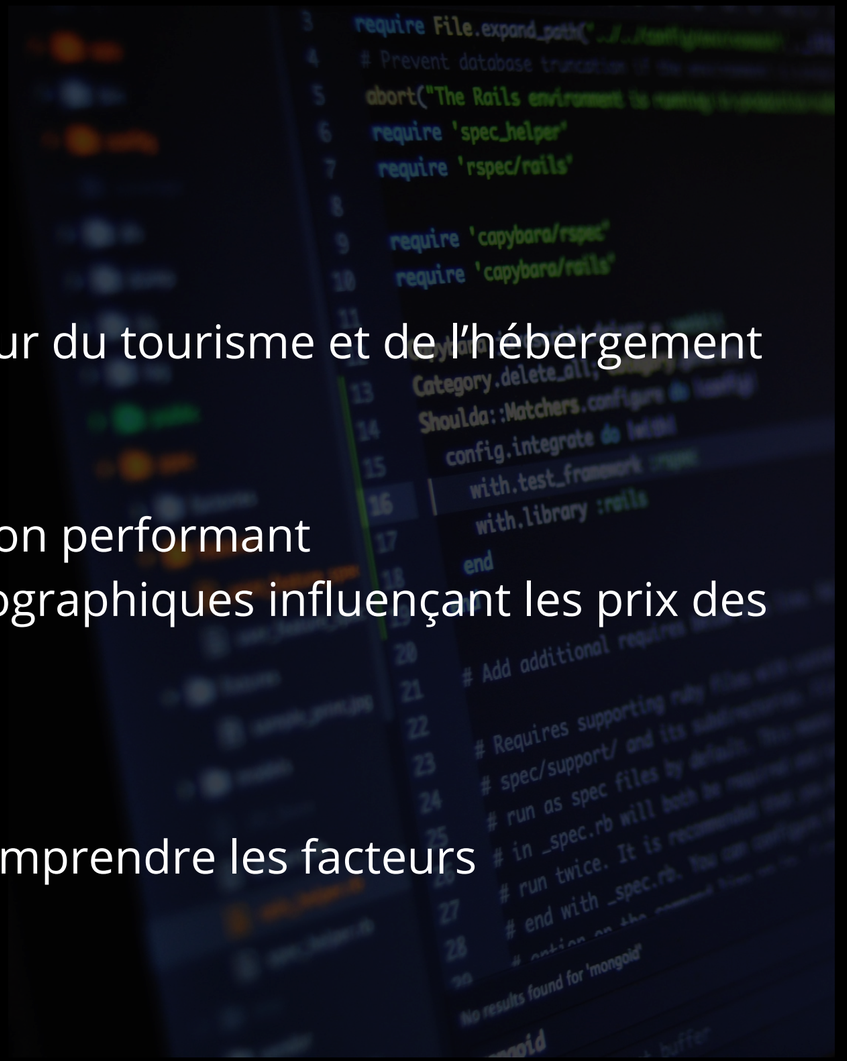
<h2> Outil de recommandation pour la recherche de locations Airbnb</h2>
Auteurs : Moheb Nazeer, Victoria Vidal, Anastasiia Savelkova, Théo Linale
Date : Novembre 2024

01 Introduction

- **Airbnb : une plateforme clé**
  - 5,6 millions de visites quotidiennes
  - Alternative majeure aux hôtels traditionnels dans le secteur du tourisme et de l’hébergement

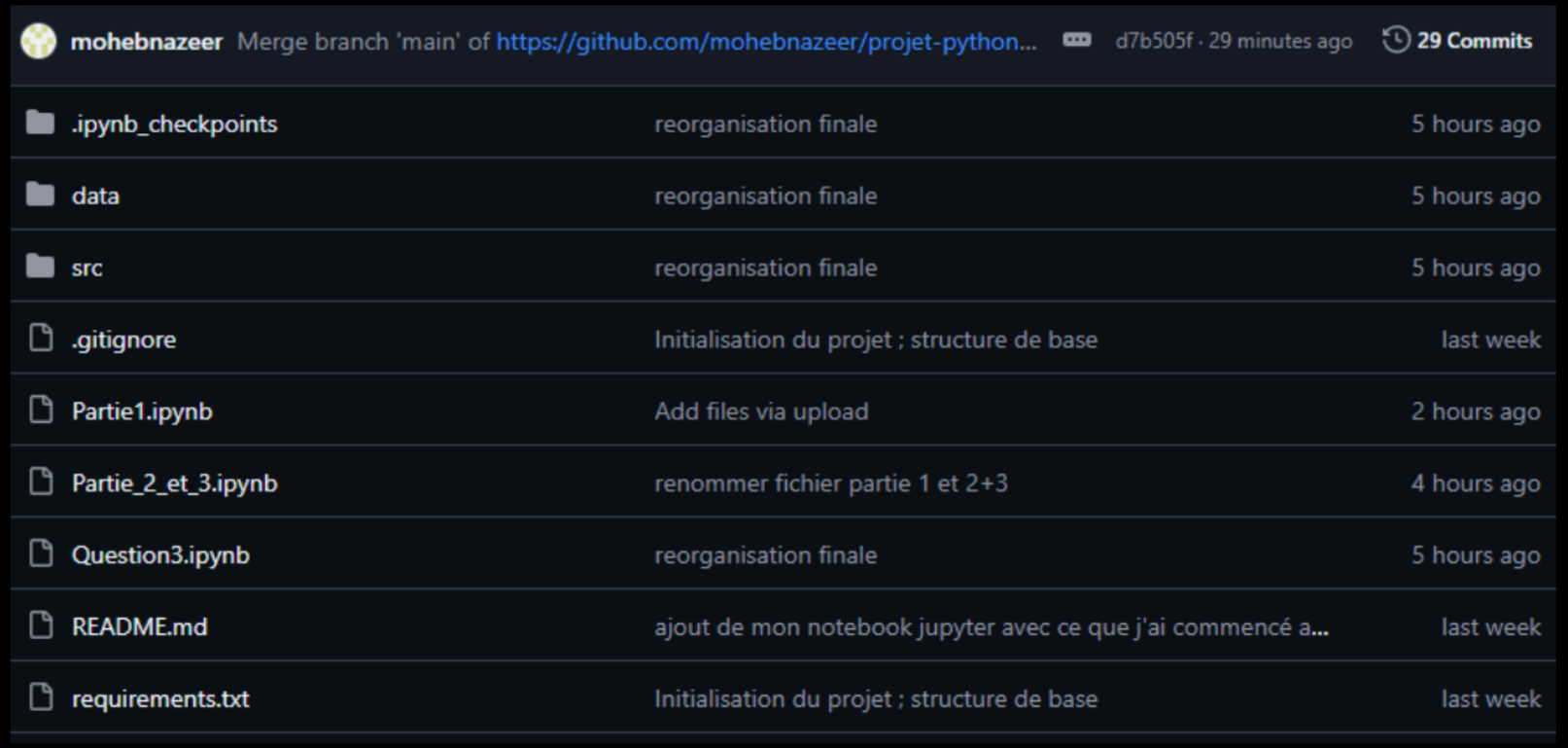
- **Objectif du projet**
  - Développer un moteur de recherche et de recommandation performant
    - Analyser les dynamiques sociales, économiques et géographiques influençant les prix des logements et les préférences des utilisateurs

- **Approche méthodologique**
  - Analyse basée sur 18 variables explicatives pour mieux comprendre les facteurs déterminants



Sommaire

02 Méthodologie et organisation du projet



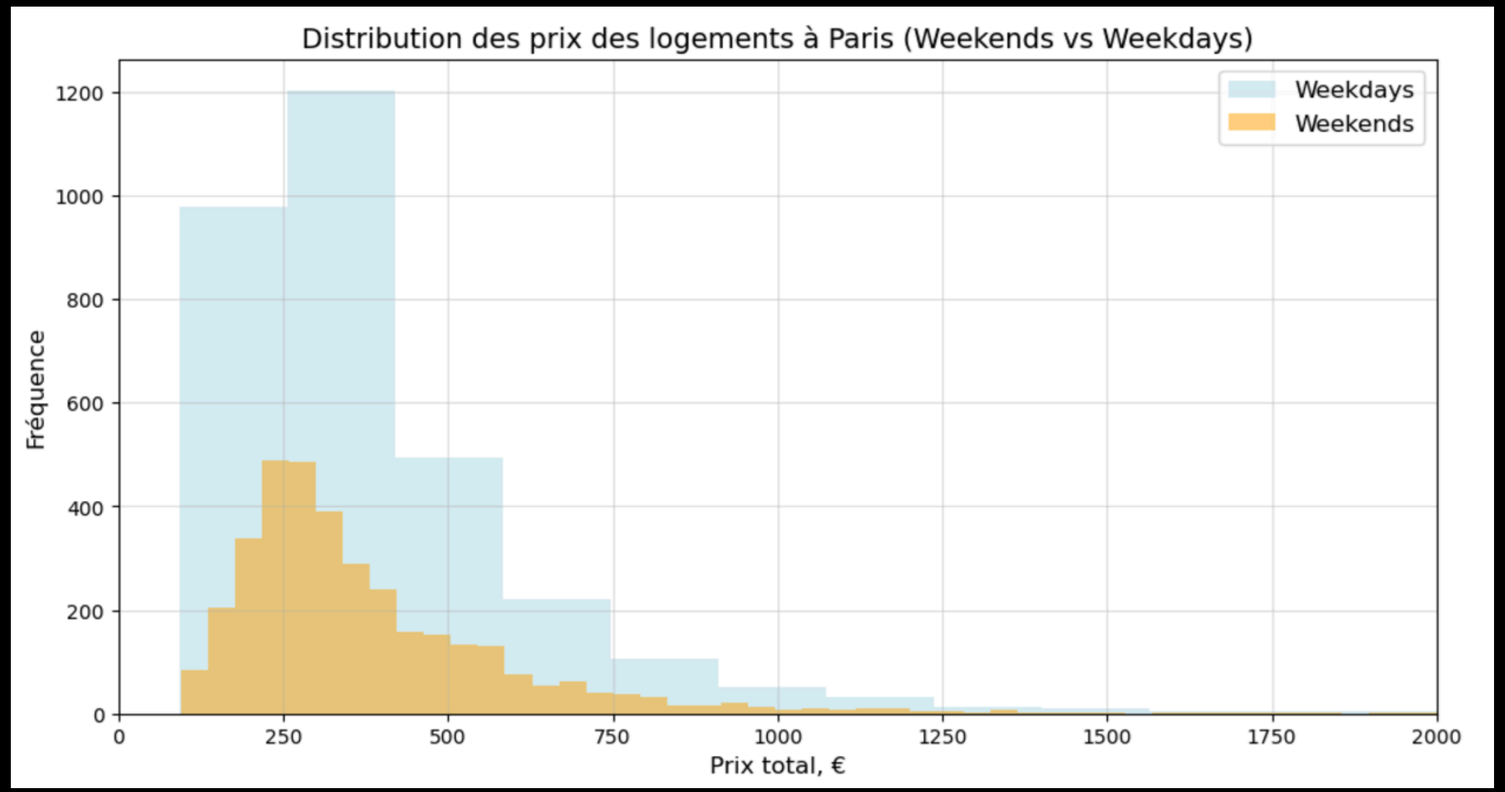
mohebnazeer Merge branch 'main' of https://github.com/mohebnazeer/projet-python...	d7b505f · 29 minutes ago	**29 Commits**
.ipynb_checkpoints	reorganisation finale	5 hours ago
data	reorganisation finale	5 hours ago
src	reorganisation finale	5 hours ago
.gitignore	Initialisation du projet ; structure de base	last week
Partie1.ipynb	Add files via upload	2 hours ago
Partie_2_et_3.ipynb	renommer fichier partie 1 et 2+3	4 hours ago
Question3.ipynb	reorganisation finale	5 hours ago
README.md	ajout de mon notebook jupyter avec ce que j'ai commencé a...	last week
requirements.txt	Initialisation du projet ; structure de base	last week

Utilisation de GitHub

- Centralisation des fichiers et des codes
- Facilitation du travail collaboratif en parallèle
- Synchronisation efficace des contributions de l’équipe

03 Etude des caractéristiques influant sur les prix

Distribution des prix des logements à Paris



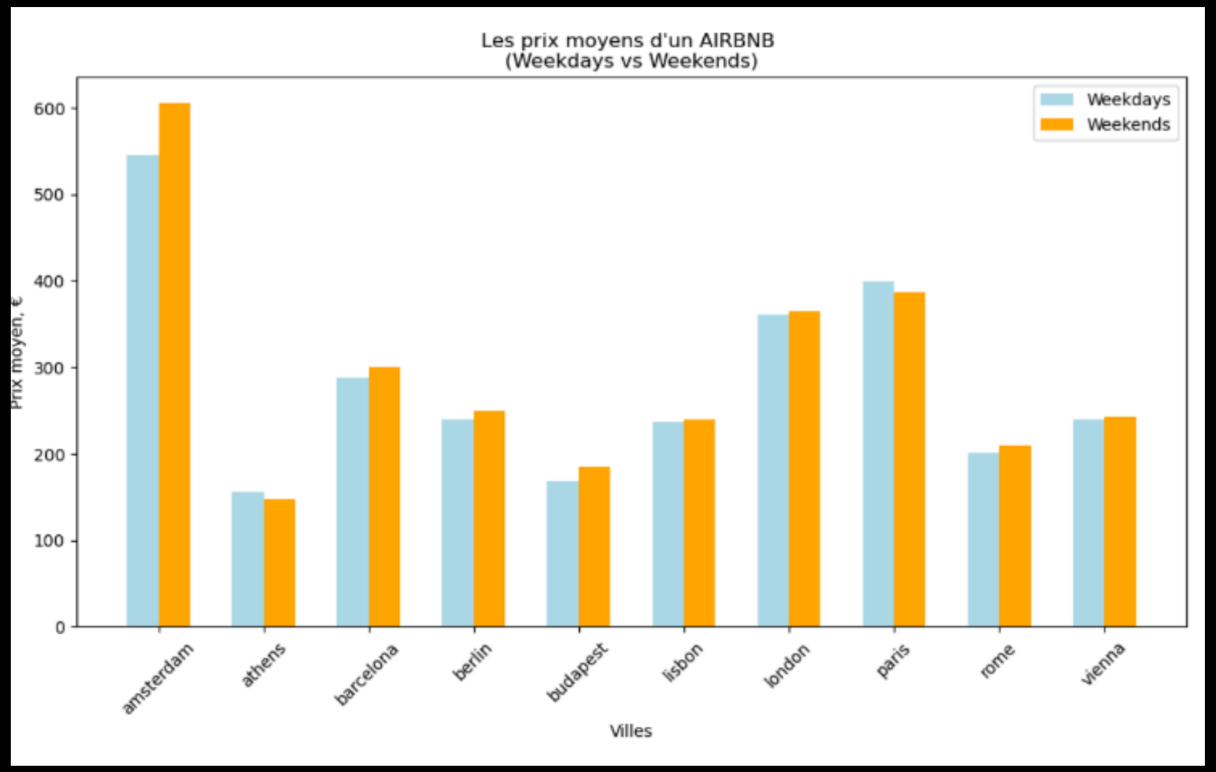
- **Prix moyen à Paris :**
  - Weekends : 387,03 €
  - Weekdays : 398,79 €

- **Écart-type :**
  - Weekends : 260,08 €
  - Weekdays : 396,37 €

- **Prix minimum :**
  - Weekends : 95,30 €
  - Weekdays : 92,74 €

- **Prix maximum :**
  - Weekends : 4 188,41 €
  - Weekdays : 16 445,61 €

Distribution des prix moyens d’un Airbnb (Weekdays vs Weekends)



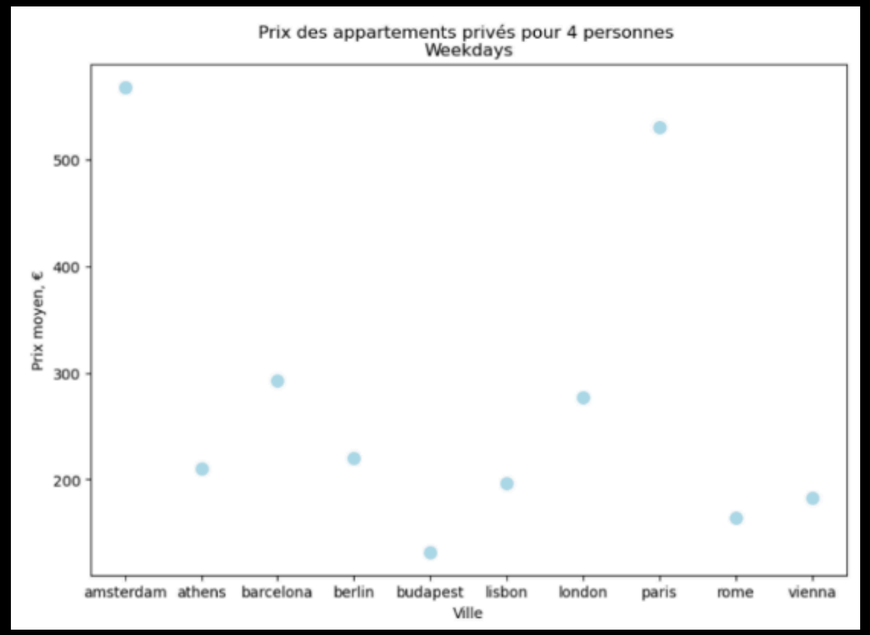
Tendance générale

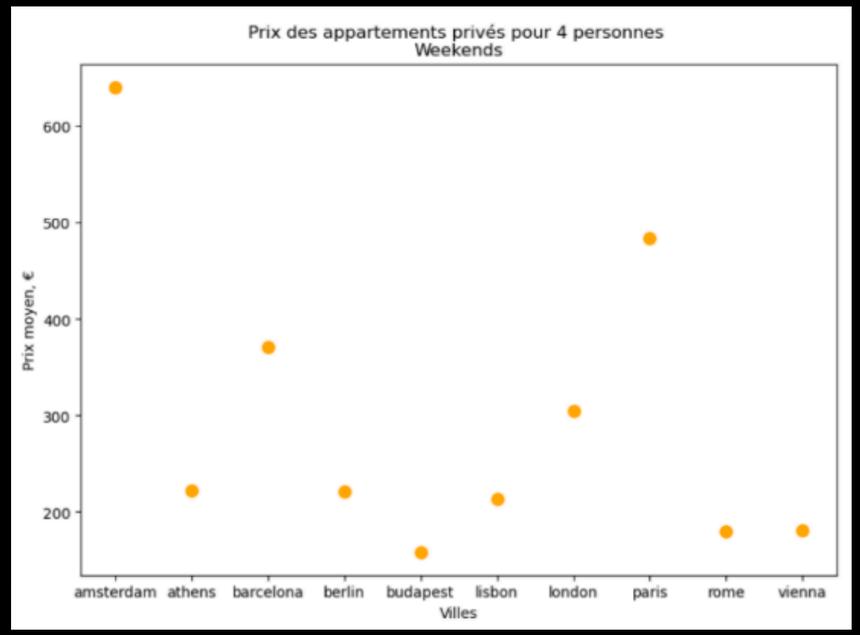
- Prix des Airbnb souvent plus élevés le weekend (afflux de touristes)
- **Barcelone, Lisbonne, Budapest, Vienne** : Prix augmentent le weekend (court séjour touristique)
- **Amsterdam** : Écart notable semaine/weekend (+50€)

Tendance inverse

- **Athènes et Paris** : Prix plus élevés en semaine (clientèle académique/professionnelle)

Prix moyen global des appartements privés (au moins 4 personnes)





Weekdays

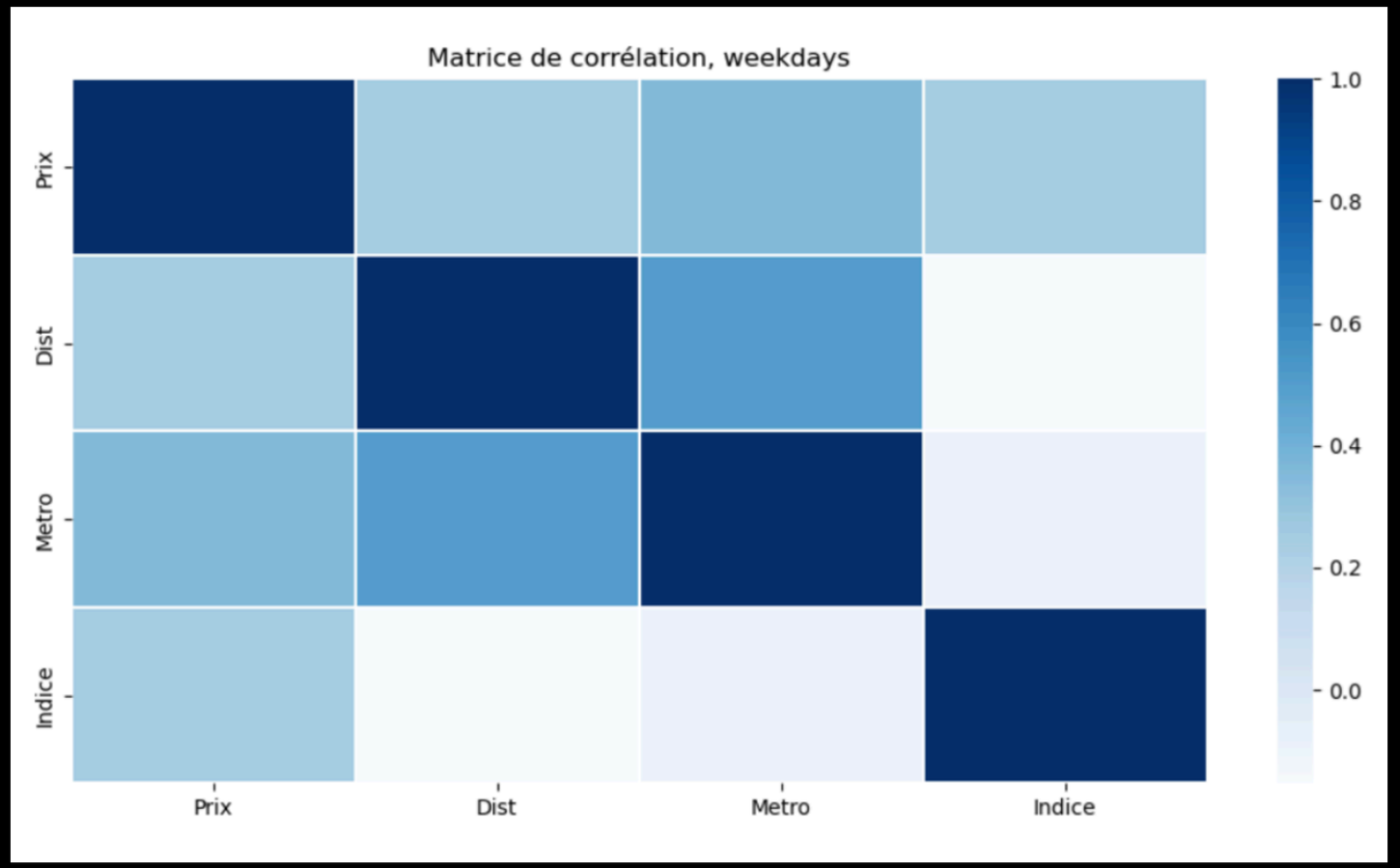
- **Amsterdam et Paris** : Prix les plus élevés (500-600 €)
- **Londres et Barcelone** : Prix intermédiaires (300-400 €)
- **Athènes et Budapest** : Prix les plus bas (100-200 €)

Weekends

- **Amsterdam** : Toujours la plus chère, dépassant 600 €.
- **Paris** : Prix baissent légèrement
- **Barcelone, Londres, Lisbonne et Berlin** : Légère hausse le week-end, liée au tourisme.
- **Athènes et Budapest** : Toujours les plus abordables (100-200 €).

L’impact des éléments géographiques

Analyse de corrélation entre les prix et les éléments géographiques

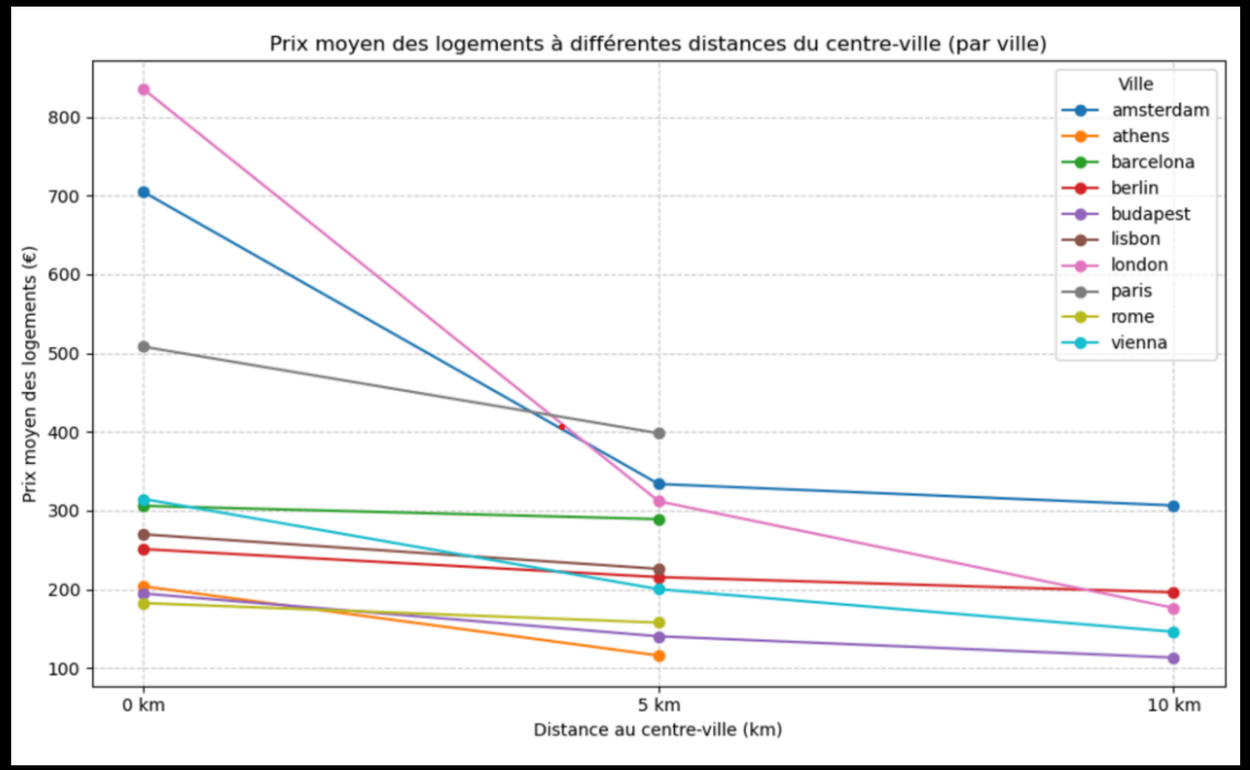


Variables	Prix	Dist
Prix	1.000000	0.254798
Dist	0.254798	1.000000

Variables	Prix	Metro
Prix	1.000000	0.354382
Metro	0.354382	1.000000

Variables	Prix	Indice
Prix	1.0000	0.2535
Indice	0.2535	1.0000

Relation entre les prix et les distances du centre-ville



- **Les prix décroissent avec la distance**
- **La pente de la diminution varie selon les villes :**
  - Forte baisse des prix avec la distance (Amsterdam/Londres)
  - Prix relativement stables avec la distance (Berlin, Budapest, Athènes, Barcelone, Lisbonne)

Régression linéaire multiple

Le modèle économétrique ajusté : $$\text{realSum} = \beta_0 + \beta_1 \cdot \text{dist} + \beta_2 \cdot \text{metro\_dist}$$

Dep. Variable :	Prix	**R-squared :**	0.133
Model :	OLS	**Adj. R-squared :**	-0.114
Method :	Least Squares	**F-statistic :**	0.5391
Date :	Sun, 24 Nov 2024	**Prob (F-statistic) :**	0.606
Time :	19 :55 :08	**Log-Likelihood :**	-60.828
No. Observations :	10	**AIC :**	127.7
Df Residuals :	7	**BIC :**	128.6
Df Model :	2		
Covariance Type :	nonrobust		

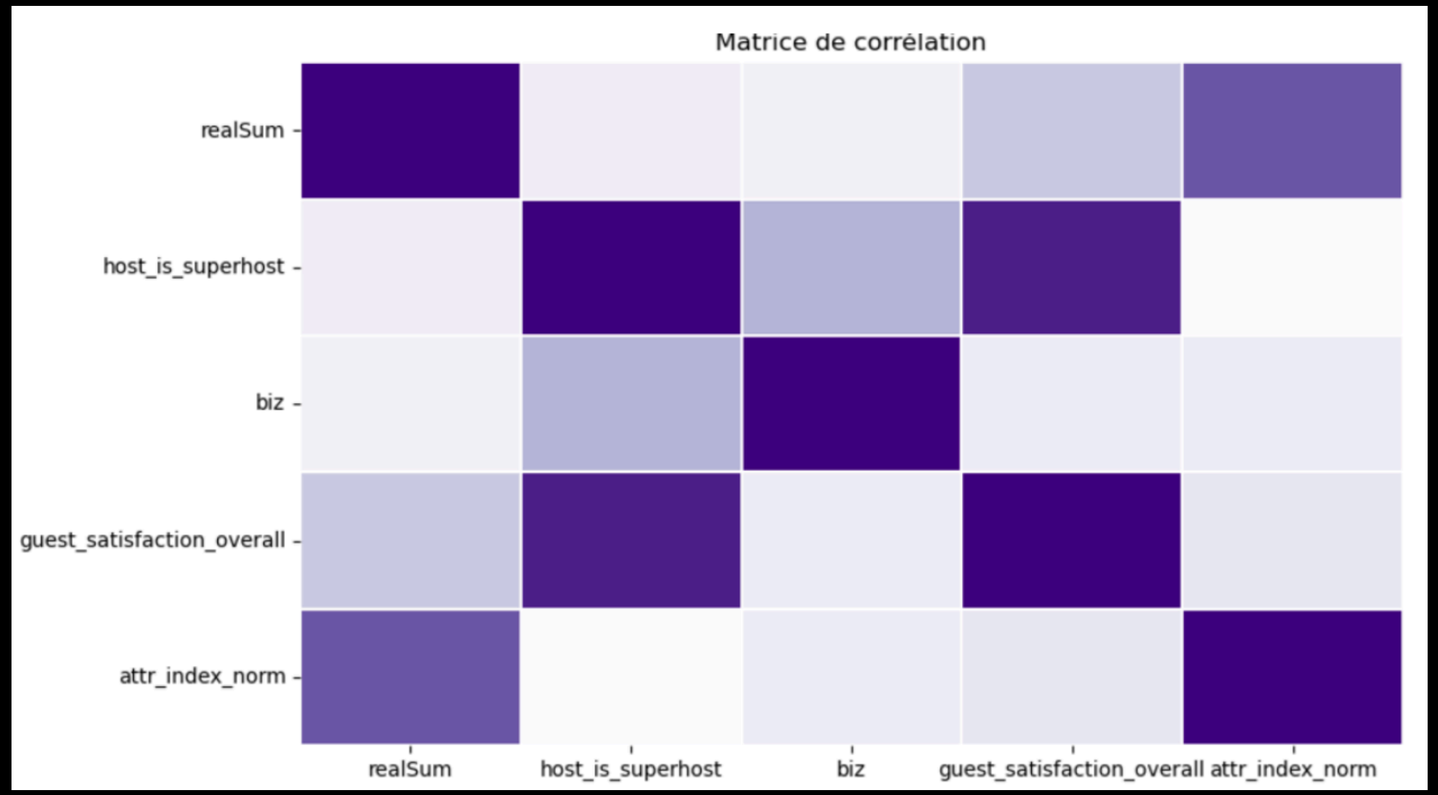
TABLE 4 – Résultats de la régression OLS

Variable	Coefficient	Erreur standard	t	P>|t|	[0.025, 0.975]
const	164.4443	124.379	1.322	0.228	[-129.666, 458.555]
Dist	9.5192	37.709	0.252	0.808	[-79.648, 98.687]
Metro	134.7111	181.023	0.744	0.481	[-293.339, 562.761]

- **Variance expliquée**
  - R-squared = 0.133 : Faible pouvoir explicatif du modèle.
- **Significativité des coefficients**
  - Les coefficients sont positifs : une augmentation supplémentaire de 1 km de distance est associée à une augmentation moyenne du prix.
  - Aucun coefficient n'est statistiquement significatif aux seuils de 1 %, 5 % ou 10 %.
- **Recommandation**
  - Augmenter la taille de l'échantillon pour des conclusions plus fiables.

L’impact des dynamiques sociales

Analyse de corrélation entre les prix et les dynamiques sociales (effet des évaluations/popularité/hôtes)



Variables	realSum	host_is_superhost
realSum	1.000000	-0.504611
host_is_superhost	-0.504611	1.000000

Variables	realSum	biz
realSum	1.000000	-0.570504
biz	-0.570504	1.000000

Variables	realSum	multi
realSum	1.000000	-0.203259
multi	-0.203259	1.000000

Variables	realSum	guest_satisfaction_overall
realSum	1.000000	-0.165855
guest_satisfaction_overall	-0.165855	1.000000

Variables	realSum	attr_index_norm
realSum	1.000000	0.539408
attr_index_norm	0.539408	1.000000

Régression linéaire multiple

$$\text{realSum} = \beta_0 + \beta_1 \cdot \text{host\_is\_superhost} + \beta_2 \cdot \text{multi} + \beta_3 \cdot \text{biz} + \beta_4 \cdot \text{guest\_satisfaction\_overall}$$

TABLE 10 – Résultats de la régression OLS

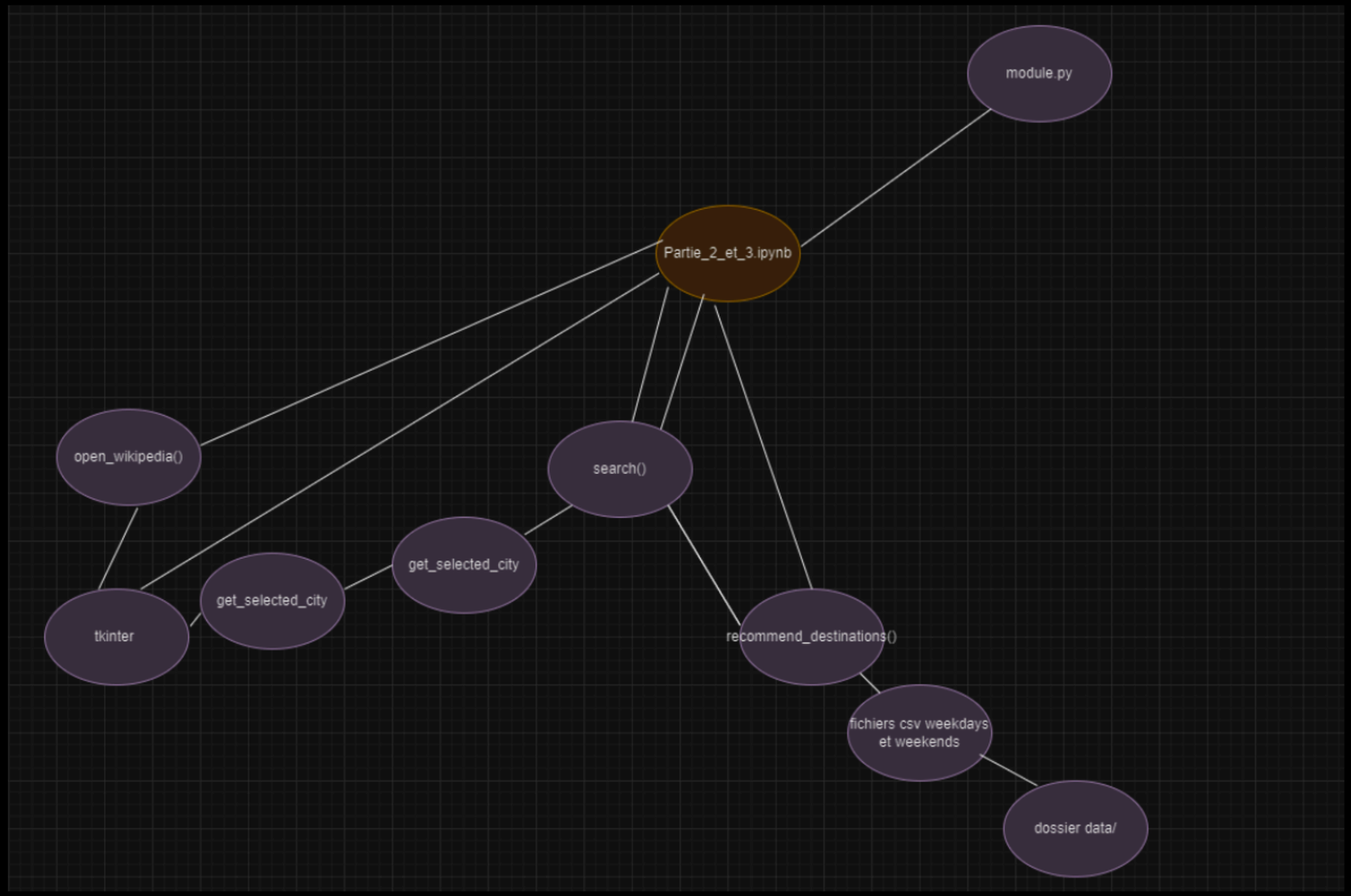
Variable	Coefficient	Erreur standard	t	P>|t|	[0.025, 0.975]
const	14500.0000	9661.966	1.501	0.194	[-10300.000, 39300.000]
host_is_superhost	1613.1391	1610.957	1.001	0.363	[-2527.958, 5754.236]
multi	-1423.0599	920.058	-1.547	0.183	[-3788.144, 942.024]
biz	-1494.5257	691.706	-2.161	0.083	[-3272.612, 283.561]
guest_sat_overall	-147.7377	103.782	-1.424	0.214	[-414.518, 119.043]

R-squared :	0.723	**Adj. R-squared :**	0.501
F-statistic :	3.255	**Prob(F-statistic) :**	0.114
Log-Likelihood :	-55.588	**AIC :**	121.2
BIC :	122.7	**No. Observations :**	10
Df Residuals :	5	**Df Model :**	4
Durbin-Watson :	1.840	**Omnibus :**	3.078
Prob(Omnibus) :	0.215	**Jarque-Bera (JB) :**	1.305
Skew :	0.885	**Prob(JB) :**	0.521
Kurtosis :	2.974	**Cond. No. :**	3.26e+04

- **Variance expliquée**
  - R-squared = 0.723 : Bonne capacité explicative.
  - Adj. R-squared = 0.553 : Performance moindre, en raison de la taille réduite de l'échantillon (N = 10) -> limite la fiabilité des résultats.
- **Significativité des coefficients**
  - Aucun coefficient significatif à 5 %, sauf biz, marginalement significatif à 10 %.
- **Recommandation**
  - Augmenter la taille de l'échantillon pour des conclusions plus fiables.

04 Architecture du programme

Intéractions des scripts



05

Moteur de recherche et interface graphique

Fonction de recommandation

```python
# Fonction de recommandations
def recommend_destinations(budget=None, capacite=None, distance=None, travel_type=None):
    filtered_data = all_data.copy()

    if budget is not None:
        filtered_data = filtered_data[filtered_data['realSum'] <= budget]                    # filtres
    if capacite is not None:
        filtered_data = filtered_data[filtered_data['person_capacity'] >= capacite]
    if distance is not None:
        filtered_data = filtered_data[filtered_data['dist'] <= distance]
    if travel_type:
        filtered_data = filtered_data[filtered_data['Type'].str.lower() == travel_type.lower()]

    # Groupement par ville et calcul des statistiques
    ville_stats = filtered_data.groupby('City').agg({          #groupement par ville
        'realSum': ['mean', 'min', 'max'],                    #moy, min et max des prix
        'cleanliness_rating': 'mean',                         #moyenne de la note de propreté
        'guest_satisfaction_overall': 'mean'                  #moyenne de la satisfaction overall
    }).reset_index()


    ville_stats.columns = ['Ville', 'Prix Moyen (€)', 'Prix minimum (€)', 'Prix maximum (€)',       #on renomme les colonnes
                           'Propreté / 10', 'Satisfaction client / 100']


    ville_stats = ville_stats.sort_values(by='Prix Moyen (€)').head(10)      # on tri par le prix
    return ville_stats
```

Cette fonction filtre les données en fonction des critères saisis par l’utilisateur et renvoie les résultats sous forme de DataFrame.

Fonction de recherche

```python
def search():
    try:
        # Récupérer les valeurs des entrées utilisateur
        budget = float(budget_entree.get()) if budget_entree.get() else None    #on stock dans budget : une valeure budget_entree que l'ut
        capacite = int(capacite_entree.get()) if capacite_entree.get() else None
        distance = float(distance_entry.get()) if distance_entry.get() else None            # .get() pour utilisation dans tkinter ensui
        travel_type = travel_type_var.get()                                          #   rpz type de voyage choisie   (type de séjour dans t

        recommendations = recommend_destinations(budget, capacite, distance, travel_type)          # on utilise la fonction de recommendc


        # Afficher les résultats dans le tableau
        for row in tree.get_children():       # permet d'obtenir les lignes présentes dans tree
            tree.delete(row)                #reinitialise le tableau a chaque iteration
        for _, row in recommendations.iterrows():        #iterer sur chaque ligne du df     (ligne par ligne)
            tree.insert("", "end", values=list(row))        #insère les valeurs dans le tableau treeview

        if recommendations.empty:
            messagebox.showinfo("Recommandations", "Aucune destination ne correspond à vos critères.")
    except Exception as e:
        messagebox.showerror("Erreur", f"Une erreur s'est produite : {e}")
```

Elle effectue une recherche basée sur des critères donnés par l'utilisateur, on y utilise la fonction de recommandation montrée au dessus et affiche les résultats dans une interface graphique sous forme de tableau.

Fonction Wikipédia

```python
def open_wikipedia(city):
    wikipedia_links = {
        "London": "https://fr.wikipedia.org/wiki/London",
        "Paris": "https://fr.wikipedia.org/wiki/Paris",
        "Rome": "https://fr.wikipedia.org/wiki/Rome",
        "Vienne": "https://fr.wikipedia.org/wiki/Vienne_(Autriche)",
        "Amsterdam": "https://fr.wikipedia.org/wiki/Amsterdam",
        "Athènes" : "https://fr.wikipedia.org/wiki/Athènes",
        "Barcelone" : "https://fr.wikipedia.org/wiki/Barcelone",
        "Berlin" : "https://fr.wikipedia.org/wiki/Berlin",
        "Budapest" : "https://fr.wikipedia.org/wiki/Berlin",
        "Lisbonne" : "https://fr.wikipedia.org/wiki/Lisbonne",
    }
    if city in wikipedia_links:
        webbrowser.open(wikipedia_links[city])
    else:
        messagebox.showinfo("Info", f"Aucun lien disponible pour {city}")
```

On a introduit une fonction supplémentaire wikipedia qui permet de diriger l'utilisateur vers une page Wikipedia qui décrit une ville sélectionnée dans un navigateur.

Tkinter

```python
fenetre = tk.Tk()
fenetre.title("Recommandations Airbnb")
fenetre.geometry("800x600")

# ----------------------------------------Section des critères----------------------------------------------------
critere_fenetre = tk.LabelFrame(fenetre, text="Entrez vos critères de recherche", padx=10, pady=10)       #section critere de recherceh d
critere_fenetre.pack(padx=10, pady=10, fill="x")
#critere_fenetre.configure(background='white')

# Budget
tk.Label(critere_fenetre, text="Budget maximal (€) :").grid(row=0, column=0, sticky="w", padx=5, pady=5)      #label textuel qui s'affich
budget_entree = tk.Entry(critere_fenetre)
budget_entree.grid(row=0, column=1, padx=5, pady=5)

# Capacité
tk.Label(critere_fenetre, text="Capacité minimale :").grid(row=1, column=0, sticky="w", padx=5, pady=5)
capacite_entree = tk.Entry(critere_fenetre)
capacite_entree.grid(row=1, column=1, padx=5, pady=5)

# Distance
tk.Label(critere_fenetre, text="Distance maximale (km) :").grid(row=2, column=0, sticky="w", padx=5, pady=5)
distance_entry = tk.Entry(critere_fenetre)
distance_entry.grid(row=2, column=1, padx=5, pady=5)

# Type de séjour
tk.Label(critere_fenetre, text="Type de séjour :").grid(row=3, column=0, sticky="w", padx=5, pady=5)
travel_type_var = tk.StringVar()
travel_type_dropdown = ttk.Combobox(critere_fenetre, textvariable=travel_type_var, values=["Weekdays", "Weekends"])    # menu déroulant
travel_type_dropdown.grid(row=3, column=1, padx=5, pady=5)

# Bouton de recherche
search_button = tk.Button(critere_fenetre, text="Rechercher", command=search)
search_button.grid(row=4, column=0, columnspan=2, pady=10)
```

```
#---------------------------------Section résultats-----------------------------------------------#

results_frame = tk.LabelFrame(fenetre, text="Résultats des recommandations", padx=10, pady=10)
results_frame.pack(padx=10, pady=10, fill="both", expand=True)                # remplit horizantelemnt et verticalement //
```

1

```
# Tableau des résultats
columns = ["Ville", "Prix Moyen (€)", "Prix minimum (€)", "Prix maximum (€)", "Propreté / 10", "Satisfaction client / 100"]
tree = ttk.Treeview(results_frame, columns=columns, show="headings", height=10)          #crée un tableau pour afficher le
for col in columns:
    tree.heading(col, text=col)
    tree.column(col, anchor="center")
tree.pack(fill="both", expand=True)
```

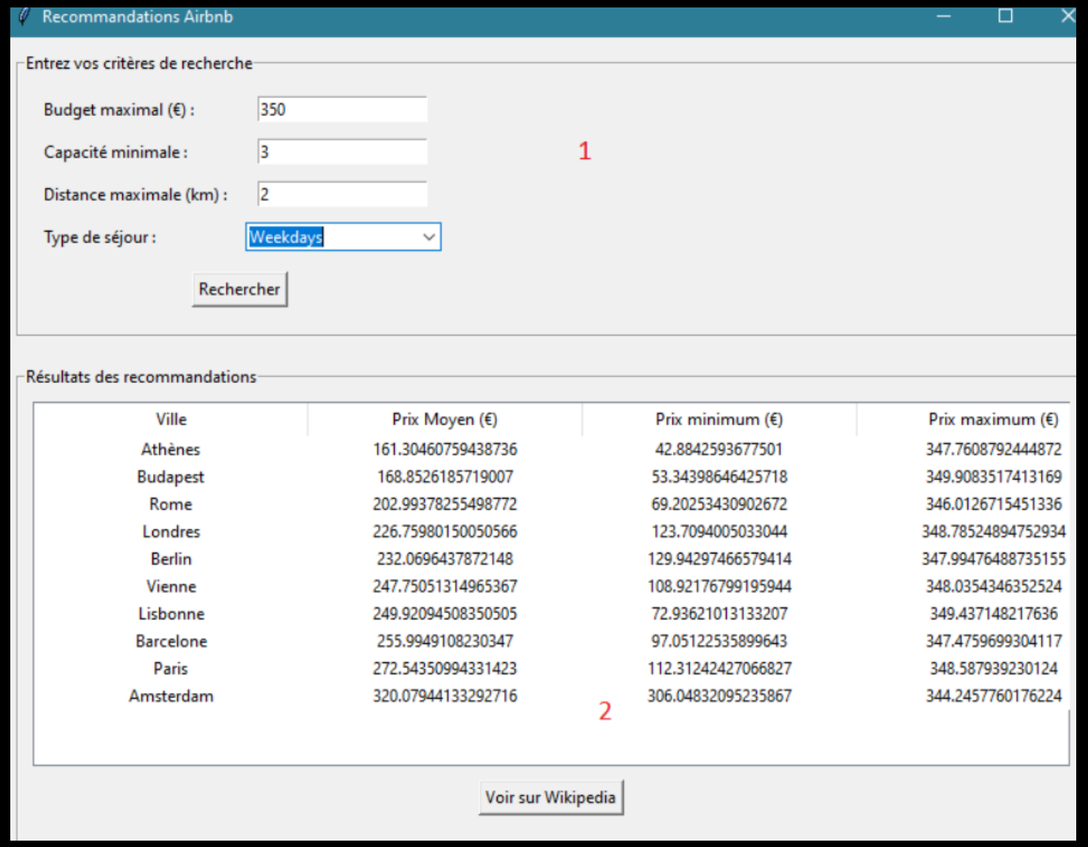
2

```
#Bouton pour Wikipédia
wikipedia_button = tk.Button(results_frame, text="Voir sur Wikipedia", command=lambda: open_wikipedia(get_selected_city()))
wikipedia_button.pack(pady=10)

# Fonction qui recupere la ville sélectionnée dans le tableau
def get_selected_city():
    selected_item = tree.focus()  # Récupère l'élément sélectionné dans le tableau
    if selected_item:
        return tree.item(selected_item)['values'][0]  # Retourne la première colonne (Ville)
    else:
        messagebox.showinfo("Info", "Veuillez sélectionner une ville.")
```

3

Interface utilisateur



L’interface utilisateur du programme se divise en deux sections principales : les critères de recherche et les résultats des recommandations.

Conclusion

Grâce à ce projet, on a pu :

-Travailler en groupe (répartition des tâches et utilisation de GitHub)

-Apprendre à manipuler de grandes quantités de données avec des bibliothèques, créer
et gérer des DataFrames à partir de différentes structures et fusionner plusieurs bases de données

-Structurer le code avec des fonctions réutilisables

-Créer une interface graphique

UNIVERSITÉ PARIS I PANTHÉON-SORBONNE

Merci !

```
1 <h2> Merci pour votre attention ! </h2>
2 <p>Auteurs : Moheb Nazeer, Victoria Vidal,
3 Anastasiia Savelkova, Théo Linale </p>
  <p>Date : Novembre 2024 </p>
```