JOURNAL DE BORD

1. Les jeux de données

Source des données : Le fichier pred-mai-mef-dhup-3.csv est obtenu à partir du site https://www.data.gouv.fr/fr/datasets/carte-des-loyers-indicateurs-de-loyers-dannonce-par-commune-en-2023/.

Les données fournies représentent les indicateurs de loyers d'annonces au niveau communal, couvrant l'ensemble de la France à l'exception de Mayotte. Les limites géographiques des communes sont celles en vigueur au 1er janvier 2023.

Ces indicateurs de loyers sont calculés à partir des données d'annonces publiées sur les plateformes de leboncoin et du Groupe SeLoger sur la période 2018-2023. Ils sont présentés toutes charges comprises pour des biens types loués vides, mis en location au cours du 3ème trimestre de 2023, avec les caractéristiques de référence suivantes : pour une maison, une surface de 92 m² et une surface moyenne par pièce de 22,4 m².

2. Définition des variables

INSEE_C: Code INSEE de la commune en France
LIBGEO: Nom de la commune en France
DEP: Code de departements en France

Lwr.IPm2: représente la borne inférieure de cet intervalle. Cela signifie que dans 95% des cas, le véritable loyer moyen par mètre carré pour les biens similaires dans la même région et à la même période se situera au-dessus de cette valeur. Autrement dit, c'est la plus petite estimation plausible pour le loyer par mètre carré.

Upr.IPm2: en revanche, désigne la borne supérieure de cet intervalle. Dans 95% des cas, le véritable loyer moyen par mètre carré sera en dessous de cette valeur. C'est donc la plus grande estimation plausible pour le loyer par mètre carré.

3. Nettoyage des données

Le "Processus de Nettoyage" est une étape cruciale dans la préparation des données avant leur présentation dans l'interface utilisateur (UI). Ce processus garantit que les données sont précises, cohérentes et utilisables, améliorant ainsi la fiabilité et l'efficacité de l'application. Ci-dessous, nous explorons les étapes complexes impliquées dans le processus de nettoyage des données pour notre application Flask, qui utilise SQLAlchemy pour les opérations de base de données et Pandas pour la manipulation des données :

3.1. Configuration et Initialisation de la Base de Données :

Initialement, l'application Flask est configurée pour utiliser une base de données SQLite nommée rental.db. Cette configuration implique de spécifier l'URI de la base de données SQLAlchemy dans les paramètres de configuration de l'application.

SQLAlchemy est initialisé avec l'objet d'application Flask, liant le ORM avec Flask. Ce ORM est crucial pour traduire les classes Python en tables de base de données et vice versa.

3.2. Définition du Modèle de Données :

La classe Rental est définie comme un modèle dans SQLAlchemy. Cette classe représente la table des propriétés de location dans la base de données.

Chaque attribut de la classe correspond à une colonne dans la table de base de données, tels que INSEE_C, LIBGEO, loypredm2, lwr_IPm2 et upr_IPm2.

La clé primaire est désignée, et les types de données sont définis pour chaque colonne afin de garantir l'intégrité des données et un indexage correct.

3.3. Lecture et Transformation des Données :

Les données sont lues à partir d'un fichier CSV en utilisant Pandas, qui est réputé pour son efficacité dans la manipulation et la gestion de données structurées. Le chemin du fichier et les paramètres nécessaires comme l'encodage et le délimiteur sont spécifiés pour analyser correctement les données.

Le format décimal européen dans le CSV (la virgule comme séparateur décimal) est converti en format standard (le point comme séparateur décimal). Ceci est crucial pour les calculs numériques et le stockage dans la base de données.

3.4. Manipulation des Données avec Pandas :

Les données subissent une transformation où chaque ligne du CSV est convertie en une instance du modèle Rental. Cette étape implique de mapper les colonnes CSV aux attributs de classe. Chaque attribut, en particulier les attributs numériques, est converti d'une chaîne en un flottant, garantissant que les types de données sont cohérents avec le schéma de la base de données.

3.5. Opérations sur la Base de Données :

Chaque objet Rental est ajouté à la session SQLAlchemy, qui gère les transactions. Cette étape est répétée pour chaque ligne dans le DataFrame, transformant et chargeant efficacement l'ensemble des données dans la base de données. Après l'ajout de tous les objets, la session est validée, ce qui signifie que les transactions sont exécutées et que les données sont stockées de manière permanente dans la base de données.

3.6. Route d'Initialisation :

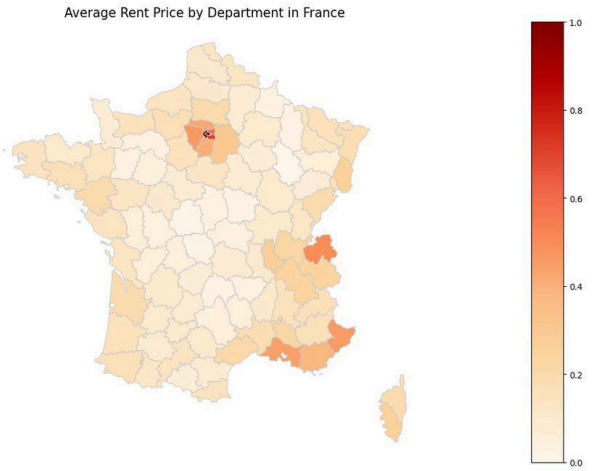
Un point de terminaison est établi pour déclencher le processus d'initialisation de la base de données et de chargement des données. Cela permet que l'opération soit exécutée facilement via une simple requête HTTP, garantissant que la base de données peut être peuplée sans interaction directe avec le code backend.

3.7. Gestion des Erreurs et Intégrité des Données :

Pendant tout le processus, la gestion des erreurs est essentielle pour garantir que les problèmes rencontrés lors de la lecture du fichier, de la conversion des données ou des opérations sur la base de données ne perturbent pas la fonctionnalité globale de l'application. Des contrôles d'intégrité sont effectués pour empêcher l'insertion de données en double, préservant l'unicité et l'exactitude des enregistrements.

À travers ces étapes méticuleuses et détaillées, le processus de nettoyage prépare non seulement les données pour leur utilisation dans l'UI, mais garantit également qu'elles respectent les normes de qualité nécessaires pour une application robuste et fiable.

4. Visualisation de Données :



La visualisation ci-dessus représente le prix moyen de location par département en France. Chaque département est coloré en fonction de son prix moyen de location, allant d'une teinte claire à une teinte foncée sur une échelle de couleur orange-rouge. Les départements avec des teintes plus foncées indiquent des prix de location plus élevés, tandis que ceux avec des teintes plus claires ont des prix de location plus bas.

Cette visualisation permet une comparaison rapide des prix de location entre les départements français. Elle met en évidence les régions où les prix de location sont les plus élevés, généralement dans les grandes villes et leurs environs, ainsi que les régions où les prix sont plus abordables, souvent dans les zones rurales ou moins densément peuplées.

La barre latérale à droite de la carte indique l'échelle des valeurs normalisées, allant de 0 à 1. Cela permet aux spectateurs de comprendre que plus la couleur est foncée, plus le loyer moyen est élevé. En outre, la légende située en haut de la carte donne un titre à la visualisation, soulignant qu'il s'agit du prix moyen de location par département en France.

Prossessus de Visualisation:

- Importation des bibliothèques : Le script commence par importer les bibliothèques Python nécessaires : pandas pour la manipulation des données, geopandas pour le traitement des données géographiques, et matplotlib.pyplot pour le traçage. Il importe également make_axes_locatable depuis mpl_toolkits.axes_grid1, qui est utilisé pour gérer les sous-graphiques et le placement de la barre de couleur.
- Chargement des données :Le code charge les données à partir d'un fichier CSV nommé pred-mai-mef-dhup-3.csv en utilisant pandas. Ce fichier devrait contenir des données immobilières incluant les prix de location, et il est délimité par des points-virgules (;) et encodé en 'ISO-8859-1' (Latin-1).
- Prétraitement des données : Il convertit les valeurs de la colonne loypredm2 de chaînes de caractères avec des virgules décimales au style français en nombres flottants, ce qui est plus pratique pour les calculs en Python.
- Regroupement des données : Le script regroupe les données par la colonne DEP (départements) et calcule le prix de location moyen par département. Ces données agrégées sont réinitialisées dans un nouveau DataFrame avg_rent_by_dep.
- Normalisation : Il normalise les prix de location moyens dans la plage de 0 à 1 en fonction des valeurs maximales et minimales trouvées dans avg_rent_by_dep. Cette étape facilite une distribution de couleur plus uniforme dans la visualisation finale de la carte.
- Chargement des données géographiques :
- Les données GeoJSON pour les départements français sont chargées dans un GeoDataFrame gdf_departments à partir d'un fichier nommé departements-version-simplifiee.geojson.
- Fusion des données : Le code prépare les données géographiques pour la fusion en s'assurant qu'une clé commune existe (DEP dans ce cas). Ensuite, il fusionne les données de loyer moyen avec les données géographiques pour combiner à la fois la localisation et les informations sur le loyer dans un GeoDataFrame gdf_map.
- Mapping des couleurs : Une carte de couleurs nommée 'OrRd' (Orange-Rouge) est choisie pour la visualisation des données. Ce type de carte

de couleurs est généralement bon pour montrer des plages de données, avec des couleurs plus chaudes indiquant des valeurs plus élevées.

- Traçage : Les données fusionnées sont tracées sous forme de carte où la colonne normalisée dicte l'intensité de couleur de chaque département. Le traçage inclut des personnalisations telles que le réglage des largeurs de ligne, des couleurs de bordure, et l'ajout d'une légende qui montre l'échelle des prix de location normalisés.
- Ajustement de la figure : Les axes sont supprimés pour nettoyer visuellement le graphique, et un titre est défini avec des paramètres de police spécifiques.
- Affichage du graphique : Enfin, la carte est affichée en utilisant plt.show(). Cette visualisation montrera les prix de location moyens à travers les départements français, codés en couleur de clair à foncé en fonction des valeurs normalisées.

5. Fonncionne d'application Flask:

L'application Flask décrite dans le README est conçue pour fournir un aperçu des prix de location par région. Elle utilise une base de données SQLite pour stocker et gérer les données relatives aux prix de location. Les utilisateurs peuvent accéder à une liste des régions avec les prix moyens de location de maison en France, ainsi qu'à des informations détaillées sur les prix pour chaque région.

Pour commencer, les utilisateurs doivent cloner le dépôt du projet sur leur machine locale et créer un environnement virtuel pour gérer les dépendances du projet. Une fois l'environnement virtuel configuré et les dépendances installées, ils peuvent lancer l'application Flask en utilisant la commande `flask run`.

Une fois l'application démarrée, les utilisateurs peuvent explorer les données en naviguant vers l'URL fournie. Ils auront accès à un aperçu des prix de location par région sur la page d'aperçu. En cliquant sur une région spécifique, ils pourront consulter des informations détaillées sur les prix de location, y compris les prix moyens, minimums et maximums.

La structure du projet est organisée de manière à faciliter la gestion et le développement de l'application. Le fichier `app.py` est le fichier principal de l'application Flask, tandis que le dossier `/templates` contient les modèles Jinja2 pour les vues de l'application. Le fichier `requirements.txt` répertorie toutes les dépendances requises pour le projet, facilitant ainsi l'installation des paquets Python nécessaires.

Enfin, le projet encourage la contribution via des pull requests et est distribué sous licence MIT, offrant ainsi aux contributeurs la possibilité de participer au développement de l'application tout en respectant les termes de la licence.