Test Technique Data Analyst

Partie 2 : Utilisation de données open source pour calculer un indicateur

Ce document me permet d'expliquer mon angle d'attaque et mes décisions en effectuant la seconde partie du test technique.

Les premières étapes que je ne détaillerai pas en profondeur, concerne la lecture en précision des consignes et la compréhension des sources données. Particulièrement, les détails des données de registre de preuve de covoiturage et l'utilisation que je dois en faire pour intégrer mon indicateur.

Le nom donné dans la consigne étant « Voyages covoiturés via plateformes », cela semble en opposition dans la description faite par la suite « incluant à la fois les trajets intermédiés par des plateformes et ceux réalisés de manière informelle ».

La subtilité entre voyages covoiturés via plateformes et nombre total de trajets réalisés en covoiturage est donnée comme un facteur 25 d'après Comprendre le covoiturage en France.

La démarche sera donc la même, que cela soit pour calculer uniquement les voyages covoiturés via plateforme, ou l'ensemble des trajets réalisés en covoiturage.

1. Choix des outils et construction du plan

J'ai commencé par créer un répertoire GitHub voyages_covoit_fr, j'effectuerai tous mes scripts en Python. Lors de nouveaux projets, je suis habitué à Python car il me permet de jongler rapidement entre des notebooks Jupyter pour l'exploration et la visualisation et une efficacité de production avec des scripts Python.

Avec ma compréhension du sujet et ma connaissance des données à ce moment, voici le premier plan construit :

- Récupération des données jusqu'à juin 2022 (le format n'est plus le même avant)
- Exploration des données.
- Construire un script Python pour nettoyer les données et garder uniquement les quelques colonnes qui m'intéressent.
- Construire des tests confirmant le bon fonctionnement du script
- Construire un script Python d'output du nouvel indicateur (nombre de trajets par jour, en milliers)
- Construire des tests confirmant le bon fonctionnement du script

Il sera légèrement modifié en avançant dans le projet.

J'utilise Visual Studio Code et Ubuntu sur ma machine.

2. Récupération des données et exploration

Je ne trouve aucune façon de récupérer les données automatiquement depuis data.gouv.fr, le nombre de fichier étant relativement bas, je décide de télécharger manuellement les csv jusqu'à juin 2022.

Je travaille ensuite sur un notebook Jupyter sur un fichier individuel, nommé historic_explo.ipynb. J'y retranscrit notamment quelques informations sur les données me

semblant essentielles et décide des colonnes que je garderai ou non. L'objectif se limitant à un nombre de trajet par jour, la plupart ne sont pas nécessaire pour mon projet.

Voici les quelques points que je remarque lors de cette étape :

- La très grande majorité des réservations concernent uniquement un seul passager (>98%)
- Il manque certaines données des colonnes « journey_start_town » et « journey_end_town ». Elles sont nulles lorsque le départ ou l'arrivée sont dans des pays étrangers.
- Les données sont de très bonne qualité pour les colonnes qui m'intéressent

3. Script Python

Mon plan était dans un premier temps de séparer mes scripts en deux programmes distincts. Un premier fusionnant et nettoyant les données, un second regroupant les données et calculant certaines métriques supplémentaires. Je me suis rapidement rendu compte que l'ensemble des données ne tenait pas en mémoire sur mon ordinateur.

Ainsi j'ai décidé de tout faire d'une traite : lire les fichiers un par un en les regroupant à chaque fois par date, pour limiter l'utilisation de la mémoire. Malgré une légère perte de segmentation du code, cette technique me permettra de venir à bout de l'ensemble de l'historique sans problème.

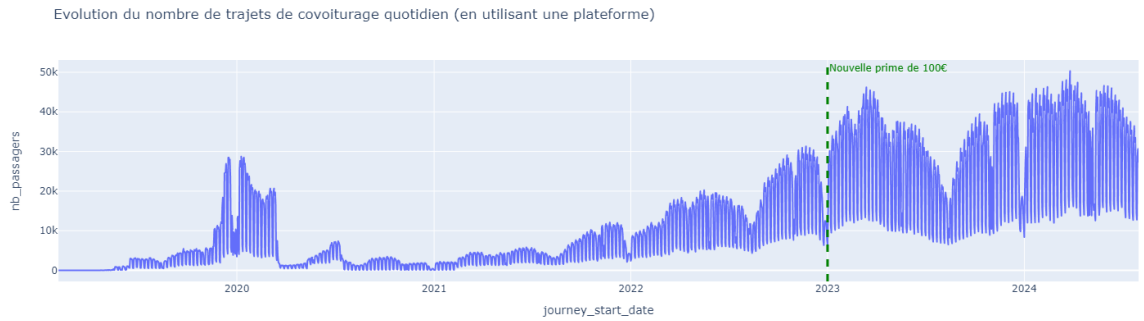
Le script se sépare en 4 fonctions.

- *concat_clean_data* : Fichier par fichier, nettoie et groupe par jour certaines colonnes en faisant appel à la fonction *clean_group_df* puis concat tous les fichiers sources historiques. Le résultat est écrit en faisant appel à la fonction *write_data*.
- *clean_group_df* : Check les données en appelant *data_quality_checks,* ensuite calcule une nouvelle métrique calculant la distance totale parcourue par chaque passager indépendamment d'un trajet. Métrique non demandée dans la consigne mais elle me semblait intéressante. Les données sont ensuite regroupées par jour et trois métriques sont calculées : la somme des passagers, le nombre de trajet (objectif du projet) et la distance totale parcourue
- *data_quality_checks* : Vérifie deux conditions, si la distance et le temps est bien supérieur à 0 et si le nombre de passager(s) est bien compris entre 1 et 6. Supprime les lignes si elles ne sont pas vérifiées. Cette fonction est testée dans test/test_concat_clean.py avec trois cas de figure, un dataframe avec des données propres, un dataframe avec des soucis de temps et de distance puis un dataframe avec des incohérences dans les nombres de passagers.
- *write_data* : Ecrit les données dans un fichier csv. Si nécessaire, créé le dossier d'outputs pour éviter une erreur lors de la première exécution.

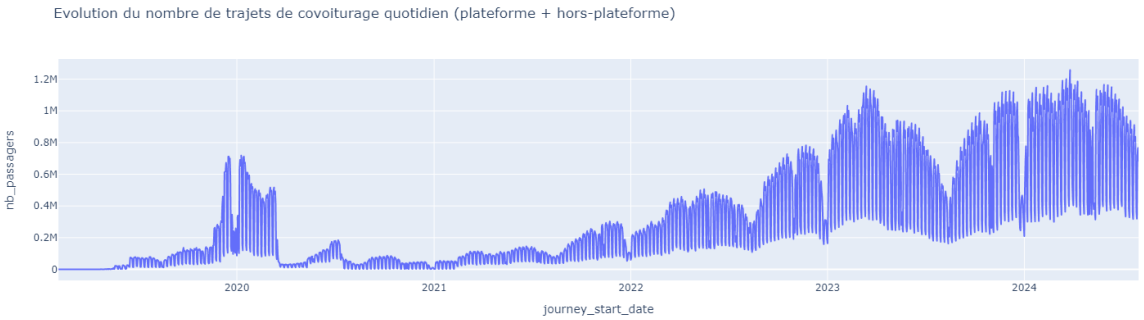
4. Visualisations

Après avoir exécuté sur l'ensemble des données (le script fonctionnant aussi sur les données antérieures à juin 2022), voici les résultats obtenus :

a) Uniquement sur les plateformes



b) Plateformes et hors-plateformes compris



La différence entre les deux graphiques étant simplement un facteur de 25, les conclusions sont les mêmes. On remarque un pic chaque semaine le mardi et une baisse significative le week-end (-66%). On peut aussi noter une augmentation du nombre de covoiturage à partir du premier janvier 2023, date à laquelle une nouvelle prime de 100€ a été mise en place pour les covoitureurs.

5. Exécutions du pipeline

Les étapes à suivre sont décrites sur le Readme du github directement.