11 janvier 2022

DEFINIR LA TRAJECTOIR D'UN VELIB'

Rapport du projet CAA

Réalisé par : *Huna LARGITTE*

Madhusha SIVASELVAN

M1 OIVM – FI

Enseignant encadreur : M. Amir NAKIB

Table des matières

INTRODUCTION 2
ANALYSE DE LA PROBLEMATIQUE 2
METHODE 2
RESULTATS 3
DISCUSSIONS DES RESULTATS 4
CONCLUSION 5
Lien GitHub 6

Définir la trajectoire du vélib'

INTRODUCTION

Dans plusieurs villes à travers le monde, l'utilisation du vélo est en constante croissance. En effet, son utilisation est fulgurant comme mode de transport : le vélo n'est plus réservé qu'au loisir en fin de semaine, il dépasse dorénavant son usage récréatif pour les déplacements du quotidien.

Histoire du vélib' :
Les vélib' sont des vélos en libre-service, mis à disposition par la ville de Paris. Le principe est simple : un utilisateur retire un vélo dans une station et le dépose dans une autre station située à proximité de sa destination. Ce service fonctionne via la forte densité de stations : 1400 stations.

Le but de ce projet est de définir la moyenne trajectoire d'un vélib' ». Pour se faire il a fallu chercher parmi les connaissances acquises au cours du cursus comment définir une trajectoire de manière algorithmique.

ANALYSE DE LA PROBLEMATIQUE

Le but de ce projet est à partir des données récupérables sur l'internet de pouvoir tracer la trajectoire moyenne des cyclistes de Vélib'. La question est donc d'analyser les données récupérées afin de définir comment ces données vont permettre d'aboutir à la fin de ce projet.

METHODE

Ce projet va être fait en faisant un codage en langage python.

La création d'un compte GitHub pour ce projet permet le travail à distance entre le binôme.

Qu'est-ce que GitHub ?
GitHub est une société à but lucratif qui offre un service d'hébergement de référentiel Git basé sur le cloud. Autrement dit, GitHub est un site web et un service de cloud qui aide les développeurs à stocker et à gérer leur code, ainsi qu'à suivre et contrôler les modifications qui lui sont apportées.

LE COMMENCEMENT

Pour ce faire le but était de récupérer les données adéquates. Après plusieurs recherches le lien suivant :
http://www.xavierdupre.fr/app/manydataapi/helpsphinx/notebooks/api_velib_jcdecaux.html#apivelibjcdecauxrst a parmi d'effectuer ce projet. Ce lien permet de récupérer la data frame de JCDECAUX et également les fonctions fournies avec les fichiers. De plus, le sujet du projet a été traité par la personne à l'initiative du lien suivant ce qui a permis de partir d'une base.

DEMARCHE EXPERIEMENTALE

Avant d'obtenir les données, plusieurs méthodes avaient été évoquées :

- L'utilisation de l'algorithme de Dijkstra (à condition d'avoir tous les points intermédiaires et non pas simplement le départ et l'arrivée et de vouloir la trajectoire moyenne la plus courte)
- La possibilité d'utiliser l'algorithme des distances moyennes de clustering (KMeans)

Ces idées ont été mise de côté de peur de ne pas respecter le sujet du projet.

Par conséquent, on se base sur la définition de la trajectoire physique. Pour se faire, il faut définir la vitesse moyenne, sous forme de vecteur afin de modéliser le déplacement. Les distances sont disponibles via les données dans le cas échéant il aurait fallu faire la valeur absolue de la différence de la latitude de départ et celle d'arrivée et faire la même chose pour la longitude. La difficulté est de définir le vecteur vitesse étant donné que les positions intermédiaires ne sont pas connues.

L'ALGORITHME / LE CODE

Comme dit précédemment, un code ainsi que toutes les explications concernant ce projet était disponible en ligne. Le but n'étant pas de bêtement le recopier, il a plutôt été analysé afin de comprendre la démarche. Tout de même JCDECAUX met à notre disposition un Data frame ainsi que quelques fonctions comme l'importation des données ainsi que la fonction distance_ haversine. Après avoir réussi à importer les données des tests ont été effectués à l'aide de fonction dans le but de vérifier que les données telles que les latitudes et longitudes ou encore les distances parcourues par chaque vélo est correctement récupérées par les algorithmes. Ainsi les fonctions velocity, time, distance, distancemoy et velocitymoy ont été implémentées dans la classe Trajectories. La classe Data_ permet de simplement importer les données et pour finir la classe Main avec la méthode main permet de lancer le projet. Les fonctions permettent d'effectuer les calculs nécessaires mais n'ont pas suffi à aboutir le projet.

RESULTATS

Plusieurs fonctions ont été implémentées :

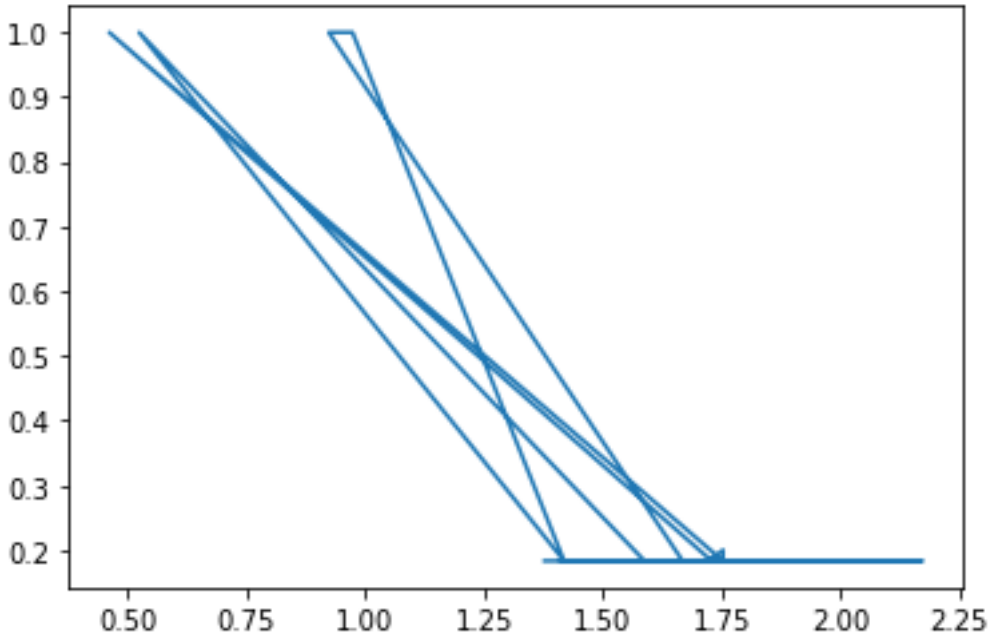
- La distance moyenne des vélib' de Besançon (fonction distancemoy dans la classe Trajectories)
- La distance parcourue par chaque vélib' (fonction distance dans la classe Trajectories)
- La vitesse pour chaque vélib' (fonction velocity dans la classe Trajectories)
- La vitesse moyenne parcourue par la liste des vélib' (fonction velocity dans la classe Trajectories)
- Le temps de parcouru par le vélib' (fonction time dans la classe Trajectories)

La classe Data_ permet l'importation des données de JDECAUX, pour finir la classe Main permet de lancer les fonctions citées précédemment ainsi que de pouvoir récupérer les fichiers à l'aide de la construction des chemins (path) en utilisant les commandes système comme Mkdir.

DISCUSSIONS DES RESULTATS

Bien que les fonctions codées permettraient d'obtenir la trajectoire d'au moins un vélib' le manque de temps et la difficulté à appliquer l'algorithmique à partir des connaissances de physique ont été le problème principal. De plus, les données fournies ne permettent pas de pouvoir retracer tout l'itinéraire d'un vélib'. C'est la raison pour laquelle, l'idée d'utiliser l'algorithme de Dijkstra a été abandonné. Concernant, les données obtenues à partir des fonctions implémentées : on note une vitesse moyenne de 6.78 km/h, une distance moyenne de 1.41 km.

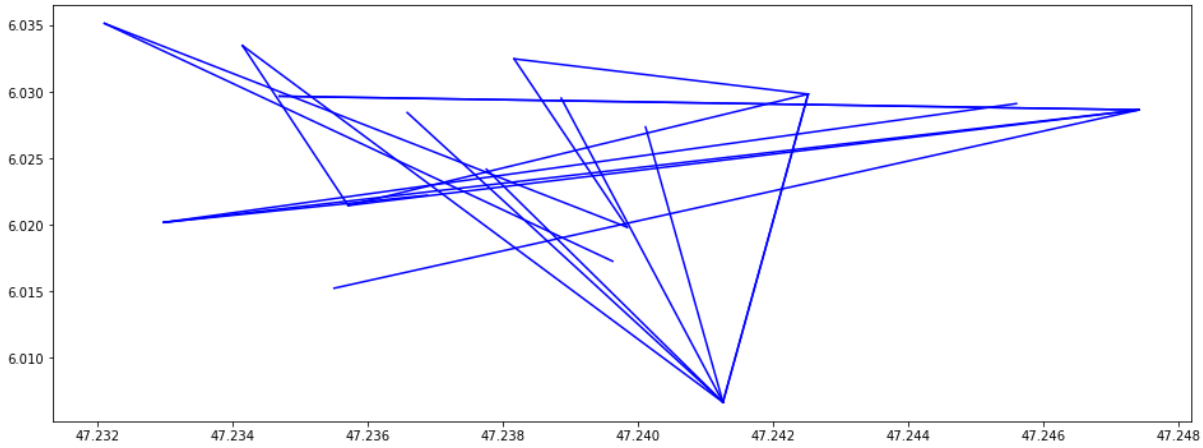
Le tracé de la vitesse des vélib' est le suivant :



Un code est disponible via le GitHub suivant :
https://github.com/sdpython/ensae_projects/blob/master/src/ensae_projects/challenge/velib_trajectories.py#L7

Selon les explications disponibles via le lien suivant :
http://www.xavierdupre.fr/app/ensae_projects/helpsphinx/challenges/velib_trajectories.html

l'application des programmes implémentés par cette personne ne serait pas un problème pour un vélib' d'ailleurs voici le graphe obtenu par son code :



Le graphe correspond aux latitudes et longitudes présent dans le fichier de données. On note un graphe qui commence à être légèrement illisible par conséquent, il le serait complétement s'il avait été appliqué aux données parisiennes qui compte 1000 stations de vélib'.

CONCLUSION

En prenant ce projet, nous avons voulu a tous cœur aller jusqu'au bout, c'est-à-dire, comme s'intitule le projet « définir la trajectoire d'un vélib' », nous n'avons pas réussi à le faire, car nous avons eu des difficultés qui sont :

- La compréhension du sujet (pour éviter le hors-sujet)
- Adapter l'algorithme aux données
- Faire le lien entre l'algorithme et nos connaissances en physique
- Appliquer l'algorithme sur un grand nombre de données
- Calculer la trajectoire d'un seul vélib' ainsi que de définir les points dans le but de tracer la trajectoire
- Par conséquent définir la trajectoire moyenne pour les vélib'

En somme, ce projet nous a permis de découvrir le principe d'étude des data, c'est-à-dire, des données clés, qui sont ici les différentes stations des vélos qui sont disponible, les vélos utilisés et leurs trajectoires.

INFORMATION

https://fr.wikipedia.org/wiki/V%C3%A9lib'_M%C3%A9tropole

Par ailleurs, à des fins statistiques, la Somupi (Société des mobiliers urbains pour la publicité et l'information), filiale de JCDecaux, dispose de ces données relatives aux déplacements pour une période de 24 mois. La CNIL s'est fermement opposée à ce délai, estimant que tout trajet inférieur à trente minutes ne devait pas faire l'objet d'une mémorisation informatique nominative,

et que ceux supérieurs à 30 minutes ne le soient que pour une durée maximale de cinq jours (en vue de permettre une éventuelle contestation). Pour ce qui est des informations nominatives des abonnés courte durée, la Somupi s'est engagée auprès de la CNIL à n'en conserver aucune.

Lien GitHub

https://github.com/Hupec/project_velib_trajectories

https://github.com/Madhusha-SIVASELVAN/project_CAA_velib_trajectoire