**Groupe SDIN**

Nicolas HERVOUIN

Arthur DIAZ

Josselin HONORE

Adhika KHAN

***Rapport Data Challenge***

Le sujet que nous avons choisi est le numéro 1. La problématique est de trouver un moyen de définir l'emplacement idéal des bases opérationnelles d’Enedis, c’est-à-dire, les villes sur lesquelles elles seraient implantées, ainsi que leur champ d’action, qui représente les communes qui leur seraient rattachées.

Pour traiter ce sujet, nous nous sommes appuyés sur quatre bases de données. D’abord, nous nous sommes servis sur les trois fichiers au format csv qui ont été fournis par Enedis.

La première base comporte le temps de trajet en voiture en secondes, entre une commune bretonne et une autre.

La deuxième base contient la quantité d’interventions au niveau de la clientèle et aussi au niveau du réseau dans les différentes communes bretonnes. Pour ces deux variables, il y a quatre niveaux d’intervention : très bas, bas, moyen, haut et très haut.

Ensuite, le troisième jeu de données présente le nom, le code postal, ainsi que les coordonnées géographiques de chaque commune. Il y avait également d’autres variables, mais nous ne les avons pas jugé utiles pour répondre à la problématique.

Enfin, la quatrième et dernière base de données, était fournie pour traiter le sujet numéro 2. Il s’agit de la densité de la population de chaque commune d’Ille-et-Vilaine. Nous avons complété ces données en prenant la densité de la population pour chaque commune en Bretagne, sur le site internet de l’INSEE. Nous avons fait ce choix afin d’affiner notre analyse.

Nous avons utilisé l’outil de programmation Python afin de pouvoir lire et analyser les quatre bases de données sous forme de dataframe.

Un problème a été soulevé par Josselin et Arthur concernant le premier jeu de données sur le temps de trajet, lors de l’analyse : la base était censée présenter uniquement les « villes de destination se situant à moins de 30 minutes de route de la commune de départ », donc 1 800 secondes maximum. Cependant, on retrouve dans cette base, des destinations à plus de 45 000 secondes ; il a donc fallu les retirer.

Ensuite, nous avons fusionné les quatre fichiers avec le nom et le code postal de la commune qui nous ont permis de faire le lien entre eux.

Sur internet, nous sommes allés récolter un code GeoJSON, en utilisant la méthode de webscraping avec le module Requests. Cela nous a permis de tracer la carte de la Bretagne, que nous utiliserons par la suite pour placer différents points et réaliser des cartes grâce aux modules Geopandas et Matplotlib.

Pour les différentes cartographies, un binôme s’est chargé de placer les points concernant les interventions de réseau de niveaux « haut » et « très haut » et l’autre binôme avait comme rôle de placer ceux concernant les niveaux d’activité clientèle « haut » et « très haut », pour ensuite pouvoir comparer les deux cartes côte à côte. Le nombre d’intervention élevé est présenté en rouge sur les deux graphiques.

GRAPH

L’idée derrière est de placer les bases opérationnelles proches des lieux où les clients nécessitent souvent des interventions de réparation, mais en même temps, il faut que ce soit dans un lieu où il y a peu de problèmes de réseau.

Nous avons donc émis l’hypothèse suivante : les zones d’action d’Enedis doivent couvrir toute la Bretagne, il est donc inévitable que certaines zones d’action se chevauchent. Il doit y avoir une corrélation entre le nombre d'interventions (clientèle et réseau) et le nombre de collaborateurs disponibles dans cette zone. Les bases opérationnelles doivent se situer à 1h de distance en moyenne des unes des autres.

La solution que nous avons tenté de suivre aux premiers abords est la suivante : nous voulions repérer toutes les villes à 1h de distance chacune, en moyenne, ayant des niveaux d’activité clientèle et réseau classés « haut » et « très haut », en partant de Rennes car c’est la capitale de la Bretagne.

Nous avons, en premier lieu, mis en place des rayons d’action du groupe Enedis de 30 minutes maximum, sans que les cercles ne se rencontrent trop. Cependant, nous sommes allés chercher sur internet s’il n’existait pas déjà un algorithme pouvant le faire. Nous sommes tombés sur l’algorithme de Dijkstra, mais en l’analysant, nous nous sommes rendus compte que ce n’était pas exactement ce que nous recherchions car cet algorithme calcule en fait le chemin le plus court.

Ensuite, nous sommes tombés sur l’algorithme de Breadth-First Search (BFS), qui est appelé algorithme de parcours en largeur en français. Il permet de créer un chemin entre chaque ville sans repasser par une ville où on est déjà passé. Sur Python, cela se caractérise par une matrice : si la ville de destination se situe à moins de 30 minutes par rapport à la ville de départ, cela affiche un 1 ou 0 sinon. Toutefois, un problème s’est présenté : nous n’arrivions pas à enregistrer les villes, où le chiffre 1 s’affichait, sous forme de dictionnaire. Un binôme s’est donc chargé de trouver la solution.

À côté, on s’est projeté en se disant qu’il y aurait très probablement une base opérationnelle dans les villes suivantes : Rennes, Saint-Brieuc, Vannes, Quimper, là où la densité de population est élevée. Nous avons donc réalisé un graphique supplémentaire ou les villes en bleu sont les villes situées à moins de 30 minutes de ces quatre villes. Cependant, il y a quelques erreurs car le code compte plus de villes (1250) que la base de données (1208) ; c’est pour cela qu’il y a des villes en bleu au milieu de la région.

DIAPO

1. Qualité analyse de données : expliquer comment l’exploiter au préalable
   1. Expliquer les bases de données
   2. Expliquer les problèmes rencontrés avec les iles, les durées
   3. Rajout d’une base de donnée du sujet 2 sur la densité de la population pour affiner notre analyse. Modification du dossier de base pour prendre les infos que nous voulons enlever période non exploitable, les communes hors de bretagne
2. Explication de la solution, difficultés rencontrées.
   1. Fusion des bases en fonction du code postal de chaque ville et du nom
   2. Mettre en place des rayons d’action du groupe Enedis de 30 min max. Sans que les cercles ne se rencontrent trop.
   3. Mise en place de l’algorithme de BFS, mais difficulté de le mettre à cause de la création des dictionnaires. Explication de l’algorithme : permet de créer un chemin entre chaque sans repasser par une ville où il est déjà passé. Algo complexe utilisation d’internet pour comprendre l’algorithme. On a pensé d’abord à l’algo de Dijkstra mais pas le bon car chemin le plus court.
3. Performance des programmes

Hypothèse : les zones d’action d’Enedis doivent couvrir toute la Bretagne, il est obligatoire que certaines zones d’actions se chevauchent. Il doit y avoir une corrélation entre le nombre de problème (client et réseau) et le nombre de collaborateur disponible dans cette zone. Montrer zone de problème

Robustesse : mise en place de fonction, code clean,

Qualité de documentation dans le code (mettre hashtag pour expliquer code, mais faire demain avec nico)

1. Les résultats, ambition à long terme.
   1. Faire fonctionner l’algorithme, et pouvoir l’adapter à n’importe quelle région de France.
   2. Opérabilité du programme, prendre en compte plus de paramètre, l’intégrer à Enedis.

On veut le prix de l’innovation. Réflexion sur la mise en place de drone afin d’être plus rapide pour la réparation mais aussi étendre la zone d’action d’Enedis. Mise en place de modèle prédictif afin d’anticiper les pannes, les problèmes (dus à la météo par exemple). Se servir de la donnée pour Identifier les zones précaires et en identifiant ces zones installé les bases à coté pour agir plus rapidement.

En fonction de la demande d’énergie par ville, département, adapter l’offre pour économie d’énergie mais aussi éviter les problèmes de tensions.