

# Data mining

Morgan Fassier, Fabrice Pont

11 décembre 2020

Le code de cette application est accessible via le github suivant : <https://github.com/fapont/gps-ecolo>. Vous y trouverez une documentation et un mode d'emploi.

## 1 Introduction

Nous avons cherché à étudier des problématiques de chemins optimaux dans les réseaux de transports en commun en France métropolitaine. Nous avons choisi d'étudier le principe d'optimalité selon 4 axes :

- **Temps de trajet** : correspond au temps mis pour faire le trajet d'un point A à un point B
- **Prix** : correspond au prix du trajet (prix au kilomètre)
- **Distance parcourue** : distance parcourue pour faire le trajet
- **Empreinte carbone** : correspond aux émissions de CO<sub>2</sub> pour une personne sur le trajet (dépend du mode de transport et de la distance parcourue)

Nous avons visualisé l'impact de ces quatre critères sur le choix du trajet pour aller d'un point A à un point B. Pour se faire il nous a fallu récupérer des données génériques sur les différents trajets possibles en France, le mode de transport et toutes les caractéristiques ci-dessus. Enfin, nous avons cherché à visualiser les résultats de nos calculs sur une carte interactive.

## 2 Récupération et traitement des données

Nous avons cherché les données de transport en commun en France métropolitaine terrestres et aériens et avons trouvé cinq sources de données regroupant une bonne base de transports : **avion**, **TGV**, **bus**, **TER** et **intercites**.

Avec cela nous avons pu créer cinq DataFrames. La figure qui suit montre celui pour les TGV ce qui permet d'illustrer le format des données de sortie (*voir annexe A pour les quatre autres DataFrames*).

Les DataFrames suivent un schéma de graphe non dirigé (pas de doublons et les trajets de A vers B et de B vers A sont réduits à un unique trajet de A vers B).

	point_1		point_2		duree (min)	distance (km)	empreinte carbone (gCO2)	prix (euros)
	nom	latitude	longitude	nom	latitude	longitude		
0	Gare de Agde	43.31757	3.46602	Gare de Béziers	43.33623	3.21922	14.50	20.077
1	Gare de Agde	43.31757	3.46602	Gare de Sète	43.41281	3.69640	15.50	21.430
2	Gare de Agen	44.20797	0.62090	Gare de Bordeaux-St-Jean	44.82654	-0.55619	75.60	115.970
3	Gare de Agen	44.20797	0.62090	Gare de Montauban-Ville-Bourbon	44.01464	1.34197	40.00	61.470
4	Gare de Agen	44.20797	0.62090	Gare de Toulouse-Matabiau	43.61146	1.45356	62.00	94.096
...	...	...	...	...	...	...	...	...
715	Gare de Versailles-Chantiers	48.79557	2.13546	Gare de Rouen-Rive-Droite	49.44903	1.09415	101.00	105.017
716	Gare de Vitry-le-François	48.71773	4.58723	Gare de Bar-le-Duc	48.77363	5.16702	27.33	42.976
717	Gare de Vitry-le-François	48.71773	4.58723	Gare de Châlons-en-Champagne	48.95551	4.34895	20.33	31.683
718	Gare de Vitré	48.12242	-1.21197	Gare de Laval	48.07625	-0.76094	20.50	33.895
719	Gare de Vitré	48.12242	-1.21197	Gare de Rennes	48.10352	-1.67233	22.75	34.253

FIGURE 1 – DataFrame des données de TGV nettoyées

## 2.1 Avion

Nous avons récupéré un fraction des données de vol d'Air France KLM provenant de data.gouv pour l'année 2019 [1] qui contiennent les positions des arrêts ainsi que les temps de trajets. Cependant ces vols n'étant pas que en France métropolitaine nous avons donc également récupéré les noms des aéroports en France métropolitaine et fait une jointure (sur les code des aéroports = IATA).

Les données de distance sont évalués avec la formule de Haversine [2]. Celles de l'empreinte carbone sont calculées à l'aide d'une étude de l'émission de CO2 par mode de transport [3]. Et enfin celles du prix sont évaluées en ce basant sur les données d'un article de Les Échos [4].

## 2.2 TGV

Nous avons récupéré les données de la SNCF provenant de leur site d'open data [5]. Ces données correspondent à tous les trajets de TGV en France ainsi que tous les arrêts pour chacun des trajets. Nous avons donc pu récupérer facilement tous les trajets ainsi que les temps de routes de chacun. Les données de distance, d'empreinte carbone et du prix sont obtenues de la même manière que pour les avions.

## 2.3 Bus

Nous avons récupéré les données de Flixbus provenant de data.gouv pour les trajets du 2020-12-07 au 2021-12-31 [6] qui contiennent les positions des arrêts ainsi que les temps de trajets. On a rencontré le même problème que pour les avions. En effet, ce dataset comprend les données des trajets de toute l'Europe. Nous avons donc utilisé la position des points extrêmes de France métropolitaine pour exclure une grande partie des points qui sont au dela de ces points [7]. Ensuite on a utilisé la liste des communes de France pour faire un deuxième tri [8] et pour finir on a vérifié à la main pour les exceptions (ville en Allemagne proche France avec le même nom qu'une commune de France, Cologne, par exemple ...)

Les données de distance sont évalués à l'aide d'une API permettant 15 000 requêtes par mois gratuitement [9]. Celles de l'empreinte carbone et du prix sont obtenues de la même manière que pour les avions.

## 2.4 TER et intercités

Nous avons récupéré les données de la SNCF provenant de data.gouv pour les trajets du 2020-12-07 au 2021-12-31 [10, 11] qui contiennent les positions des arrêts ainsi que les temps de trajets. Même format que pour les bus sans le problème des arrêts hors France.

Les données de distance, d'empreinte carbone et du prix sont obtenues de la même manière que pour les avions.

## 3 Modélisation du problème

Après avoir acquis et nettoyé les données la prochaine étape a été de générer un graphe à partir de ces données englobant les informations qui nous intéressent et ensuite développer un algorithme de plus cours chemin répondant à nos attentes.

### 3.1 Passage des données au graphe

Nous avons utilisé le package NetworkX pour générer les graphes à partir des DataFrames. On a défini :

- les noeuds comme étant les arrêts avec comme nom le nom de l'arrêt et comme attribut la position géographique
- les arêtes comme étant la liaison entre deux arrêts et ayant comme attributs les différentes caractéristiques qui nous intéressent (durée, distance, empreinte carbone, prix) ainsi que le nom du type de transport et le poids

Le poids a été défini comme suit :

$$poids = \frac{1}{c} \sum_{i=1}^4 c_i \frac{x_i - x_{i,min}}{x_{i,max} - x_{i,min}}$$

Où  $c = \sum_{i=1}^4 c_i$  et les  $c_i$  sont les coefficients (durée, distance, empreinte écologique et prix) entre 1 et 10 (1 étant une importance faible et 10 l'inverse).  $x_i$  représente un des quatre attributs (durée, distance, empreinte carbone, prix) de l'arête associé et  $x_{i,min}, x_{i,max}$  sont les valeurs minimales et maximales sur toutes les arêtes du graphe. On ramène donc chacune des caractéristiques entre 0 et 1 et on évalue son importance.

Ensuite on a fait une union disjointe de ces cinq graphes pour avoir un graphe englobant tous les trajets possibles. Pour pouvoir bien connecter les graphes entre eux on a, pour chaque noeud, fait une connexion à pied (vitesse de 3 km/h et trajet à vol d'oiseau) de tous les noeuds à moins de 5 km ou alors, si il n'y en a pas, du seul après. Cette union de graphes ayant remplacer les noms des arrêts par des numéros uniques (fonction `disjoint_union` de NetworkX), on a créé un dictionnaire avec les numéros comme clé et les noms des arrêts comme valeur en utilisant l'attribut (la position géographique) en supposant qu'a une position géographique, nous avons un nom d'arrêt unique.

Sur la page qui suit on retrouve, tout d'abord, une illustration de l'ensemble des graphes et ensuite l'illustration du graphe des bus (*voir annexe B pour le graphe de l'union et les graphes des autres moyens de transport*) :

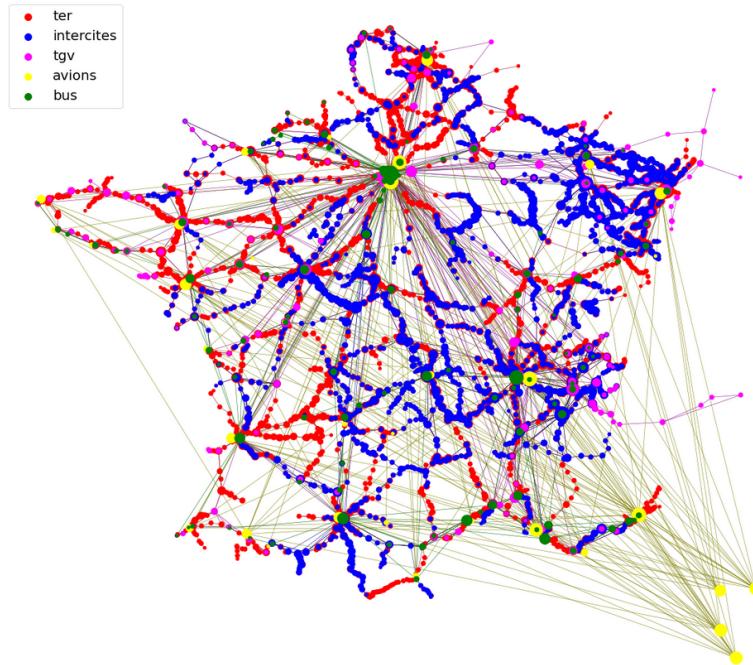


FIGURE 2 – Illustration des graphes pour les cinq moyens de transport

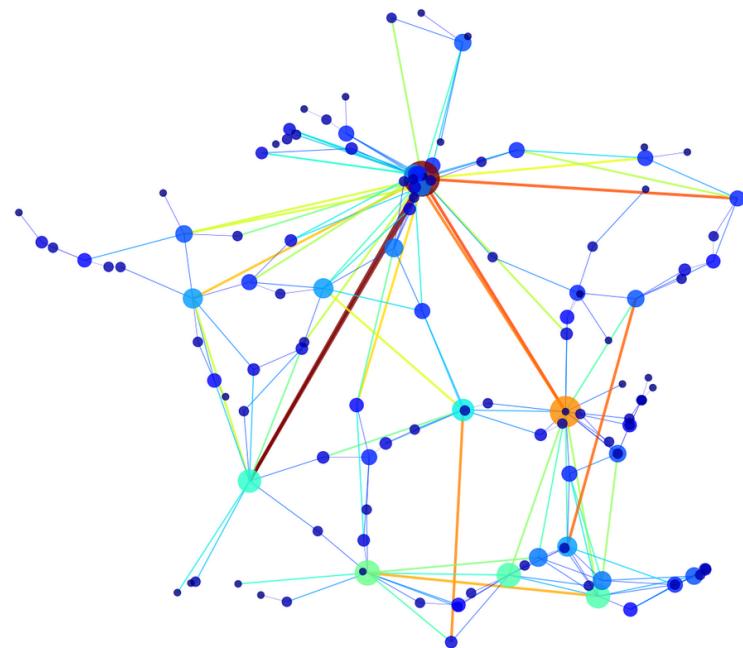


FIGURE 3 – Illustration du graphe des bus : taille du noeud en fonction du degré et épaisseur de l'arête en fonction du poids

### 3.2 Algorithme du plus court chemin

Après avoir bien défini les graphes, l'algorithme du plus court chemin se fait relativement bien en utilisant l'algorithme de Dijkstra avec la fonction `dijkstra_path` déjà implémentée sous NetworkX. Il faut lui donner le graphe total (avec tout les arrêts, leur connexions par transport et les connexions additionnelles à pied définies plus haut), le point de départ, le point d'arrivée et le nom de l'attribut utilisé comme poids (défini au paragraphe 3.1).

Comme les points de départ et d'arrivée ne sont que très rarement exactement ceux d'un noeuds du graphe, il faut créer les noeuds départ et arrivée puis les connecter aux noeuds des arrêts existants (dans notre algorithme nous connectons ces noeuds au 10 noeuds les plus proches en distance avec la formule de Haversine).

### 3.3 Améliorations

On voit assez vite les points d'amélioration :

- On a peu de connexions entre les moyens de transports à l'arrivée dans une ville et les données de bus sont assez pauvre. Il faudrait avoir les données des transports en communs dans chaque ville et d'autres compagnies de bus
- Les trajets à pieds, en tgv, en intercités et en ter se font à vol d'oiseau (comme pour les avions en utilisant la formule de Haversine) ce qui n'est pas réaliste et amène à des visualisation incohérentes (traverser une rivière, etc..)
- Les temps d'attente entre les moyens de transport sont nuls, ce qui n'est pas du tout réaliste. Il faudrait pouvoir estimer l'attente moyenne par moyen de transport pour rendre ça plus réaliste
- Ajouter des nouvelles données est très simple, il suffit de générer un DataFrame au même format que les autres et en l'ajoutant à la liste des DataFrames disponibles on peut utiliser ces nouvelles données (algorithme facilement applicable à d'autres données de transport). Il serait intéressant de rendre cela automatique

## 4 Exploitation des données

Après avoir étudié les données sous forme de graphe nous avons voulu visualiser plus facilement les données et les rendre manipulable par n'importe qui. Pour ce faire nous avons réalisé une application WEB permettant de visualiser les trajets sur une carte interactive.

### 4.1 Présentation de l'application WEB

Notre application a été développé grâce au framework Flask (codé en Python). Nous avons pu facilement intégrer nos calculs de graphes au backend de cette application et afficher les résultats de manière interactive. Notre application contient deux parties : une carte interactive permettant de visualiser les trajets et une section de configuration permettant de renseigner nos préférences en matière d'éologie, temps, prix et distance (les quatre premiers attributs des arêtes présents dans le graphe global qui correspondent à leur poids). Cette section nous permet aussi de choisir notre trajet et de montrer des informations relatives à notre trajet (différentes étapes, prix total, distance totale, etc..). De manière plus visuelle, notre application se présente sous la forme ci-dessous :

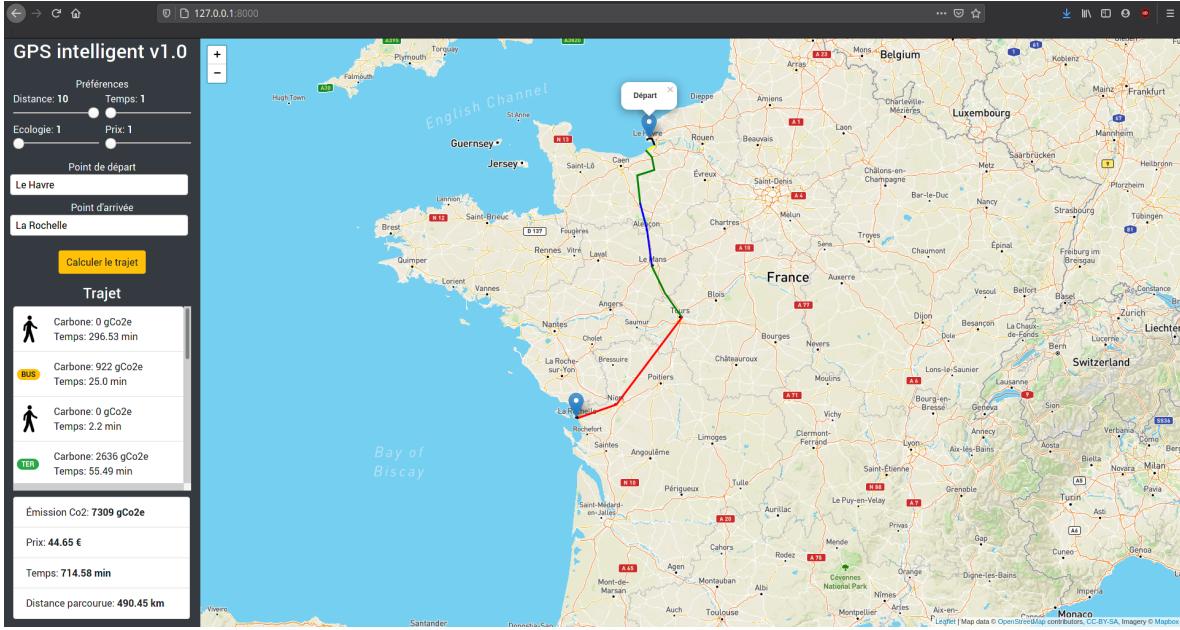


FIGURE 4 – Interface de l’application web

On retrouve bien sur la carte, le trajet optimal en fonction des préférences de l’utilisateur. Chaque moyen de transport est représenté par une couleur unique ce qui permet de facilement comprendre le trajet.

## 4.2 Fonctionnalités

Pour faciliter l’utilisation de notre application nous avons rajouté quelques options qui rendent l’exploration plus agréable. Les fonctionnalités sont les suivantes :

- Recherche dynamique des adresses (villes, rues, etc..) et récupération des coordonnées géographiques grâce à une API proposée par le gouvernement [12].
- Conservation du trajet précédent dans les champs d’écriture afin de pouvoir facilement comparer ce même trajet pour plusieurs configurations différentes (ex : trajet le plus court, le plus écologique, etc..)
- Navigation sur la carte de manière interactive afin de pouvoir visualiser les différents points d’arrêts (ex : gare, aéroport, etc..)

## 4.3 Exemple d’utilisation

Afin d’illustrer le fonctionnement de notre application nous avons choisi un trajet assez complexe car beaucoup de chemins existent entre eux. Nous avons choisi le trajet **Rennes - Cannes** et avons différencié 4 cas d’utilisation (vous êtes invité à effectuer les exemples qui suivent en parallèle avec l’application WEB lancée en suivant les instructions présentes dans le README du github) :

- **Le trajet le plus rapide** : ce trajet nous indique que l'avion est le moyen le plus rapide pour se rendre à Cannes depuis Rennes (ce trajet est aussi le trajet permettant de parcourir le moins de distance). Le trajet se termine par un TER permettant de relier l'aéroport à la ville.

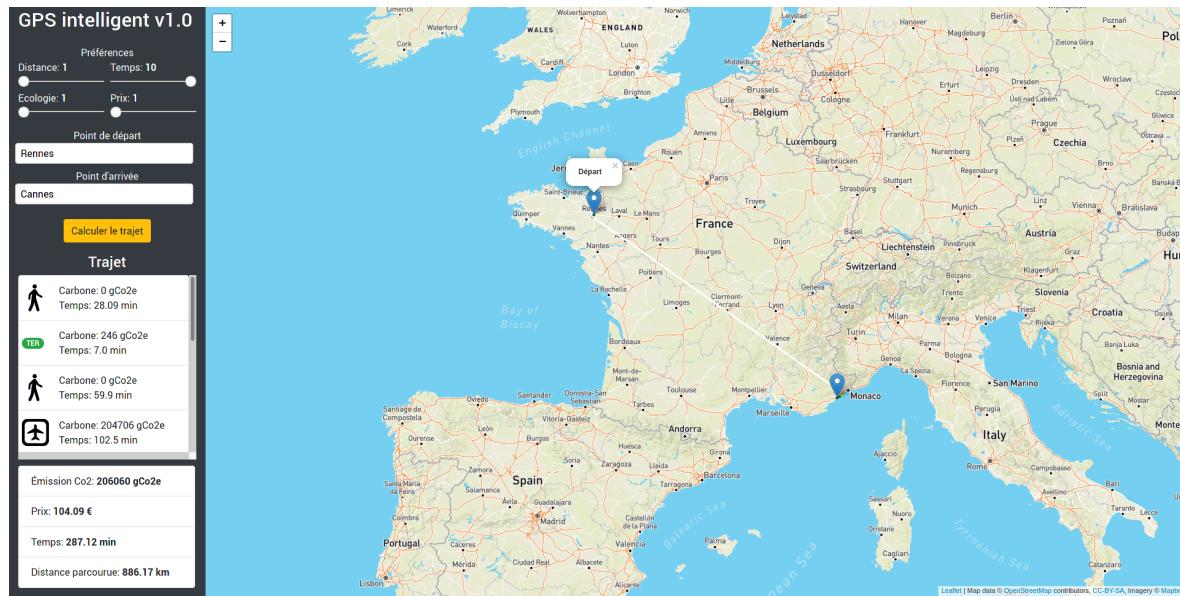


FIGURE 5 – Trajet le plus rapide

- **Le trajet le moins cher** : ce trajet nous propose une alternance entre TER, intercités et bus mais le temps de transport est très important.

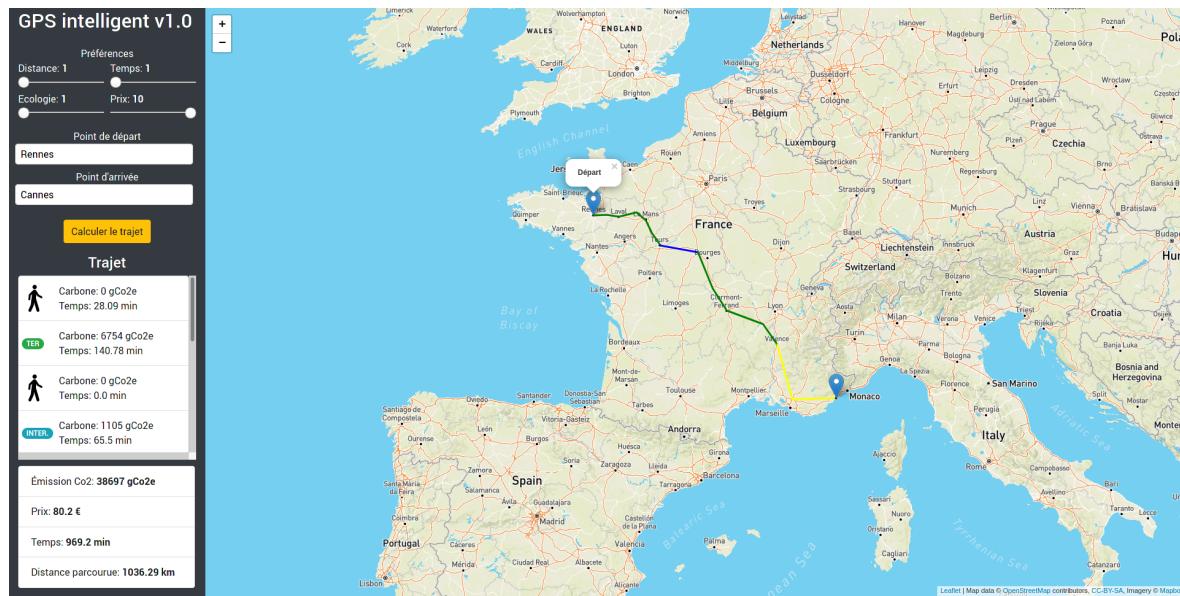


FIGURE 6 – Trajet le moins cher

- **Le trajet le plus écologique** : ce trajet correspond à l'utilisation exclusive des lignes de TGV, il est très écologique comparé à tous les autres et le temps est raisonnable.

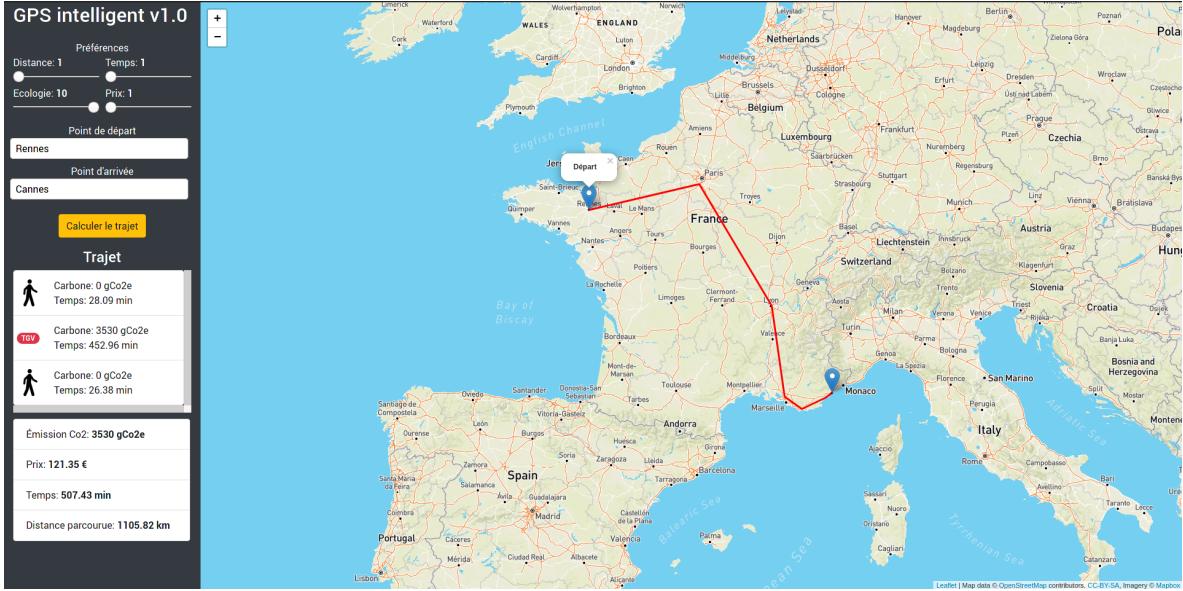


FIGURE 7 – Trajet le plus écologique

Il est tout à fait possible grâce à notre application de choisir n'importe quel trajet en France métropolitaine et de choisir un compromis entre les différentes caractéristiques énoncées ci-dessus.

## 5 Conclusion

Nous avons réalisé diverses étapes de conception d'un projet d'exploration des données. Tout d'abord, nous avons choisi un sujet qui nous semblait intéressant et qui nous permettait de manipuler des concepts vus en cours. Ensuite, nous avons cherché et nettoyé des données pour pouvoir explorer le sujet que nous avions choisi. Nous avons appliqué des algorithmes de fouille des données (notamment des algorithmes de plus court chemin) sur notre dataset préparé au préalable. Enfin, afin de pouvoir exploiter ces résultats et tester la robustesse de nos modèles nous avons réalisé une interface graphique permettant d'utiliser toutes ces notions de manière simple et rapide.

Finalement, nous avons réussi à valoriser des données brutes récupérées sur internet et à fournir une application utilisable.

## Références

- [1] *Données Air France KLM.* <https://transport.data.gouv.fr/datasets/programme-simplifie-par-saison-iata/#dataset-resources>.
- [2] *Formule de Haversine.* [https://fr.wikipedia.org/wiki/Formule\\_de\\_haversine](https://fr.wikipedia.org/wiki/Formule_de_haversine).
- [3] *Étude sur les émissions de CO<sub>2</sub> par moyen de transport.* <http://www.chair-energy-prosperity.org/wp-content/uploads/2019/01/emissions-de-co2-par-mode-de-transport.pdf>.
- [4] *Article sur les prix moyen par km par moyen de transport.* <https://www.lesechos.fr/2012/12/les-trajets-en-voiture-deux-fois-plus-couteux-quen-transport-en-commun-383882>.
- [5] *Données SNCF.* <https://ressources.data.sncf.com/explore/dataset/horaires-des-train-voyages-tgvinouiuigo/information>.
- [6] *Données de Flixbus.* <https://transport.data.gouv.fr/resources/11681>.
- [7] *Points extrêmes en France métropolitaine.* [https://fr.wikipedia.org/wiki/Liste\\_de\\_points\\_extr%C3%AAmes\\_de\\_la\\_France](https://fr.wikipedia.org/wiki/Liste_de_points_extr%C3%AAmes_de_la_France).
- [8] *Données sur les communes de France métropolitaine.* <https://www.data.gouv.fr/fr/datasets/communes-france-metropole/>.
- [9] *API pour les routes (pour les bus dans notre cas).* <https://business.mapquest.com/>.
- [10] *Données SNCF TER.* <https://transport.data.gouv.fr/resources/16387>.
- [11] *Données SNCF intercités.* <https://transport.data.gouv.fr/resources/16499>.
- [12] *API gourvernement.* <https://geo.api.gouv.fr/adresse>.

## A DataFrames

	point_1		point_2			duree (min)	distance (km)	empreinte carbone (gCO2)	prix (euros)
	nom	latitude	longitude	nom	latitude	longitude			
0	PARIS-CHARLES-DE-GAULLE	49.00972	2.55000	NANTES-ATLANTIQUE	47.15972	-1.61000	65.83	371.545	89728.1 44.59
1	PARIS-ORLY	48.71972	2.37972	MARSEILLE-PROVENCE	43.43972	5.20972	80.43	626.944	151407.0 75.23
2	PARIS-ORLY	48.71972	2.37972	TOULOUSE-BLAGNAC	43.63000	1.37000	78.12	571.879	138108.8 68.63
3	NICE-COTE-D'AZUR	43.67000	7.20972	PARIS-ORLY	48.71972	2.37972	87.49	673.845	162733.6 80.86
4	PARIS-ORLY	48.71972	2.37972	BORDEAUX-MERIGNAC	44.82972	-0.71972	71.06	493.196	119106.8 59.18
...	...	...	...	...	...	...	...	...	...
149	RENNES-ST-JACQUES	48.07000	-1.72972	STRASBOURG-ENTZHEIM	48.53972	7.62972	77.50	694.546	167732.9 83.35
150	LILLE-LESQUIN	50.56000	3.08972	PAU-PYRENEES	43.38000	-0.42000	95.00	842.296	203414.5 101.08
151	METZ-NANCY-LORRAINE	48.97972	6.25000	PAU-PYRENEES	43.38000	-0.42000	100.00	807.341	194972.9 96.88
152	STRASBOURG-ENTZHEIM	48.53972	7.62972	PAU-PYRENEES	43.38000	-0.42000	95.00	846.454	204418.6 101.57
153	STRASBOURG-ENTZHEIM	48.53972	7.62972	BIARRITZ-BAYONNE-ANGLET	43.46972	-1.53000	100.00	904.537	218445.7 108.54

FIGURE 8 – DataFrame des données d'avions nettoyées

	point_1		point_2			duree (min)	distance (km)	empreinte carbone (gCO2)	prix (euros)
	nom	latitude	longitude	nom	latitude	longitude			
0	Paris Charles de Gaulle (Central Bus Station ...)	49.01067	2.55926	Paris (Bercy Seine)	48.83532	2.38052	42.69	29.749	1740.3 2.08
1	Lyon Perrache (Central Bus Station)	45.74875	4.82602	Grenoble (central bus station)	45.19283	5.71394	95.50	104.401	6107.5 7.31
2	Paris (Orly Airport)	48.73156	2.37358	Paris (Bercy Seine)	48.83532	2.38052	31.25	15.046	880.2 1.05
3	Lyon Perrache (Central Bus Station)	45.74875	4.82602	Clermont-Ferrand (bus terminal Les Salins)	45.77076	3.08225	128.12	166.360	9732.1 11.65
4	Paris Charles de Gaulle (Central Bus Station ...)	49.01067	2.55926	Lille	50.63920	3.07620	138.93	198.490	11611.7 13.89
...	...	...	...	...	...	...	...	...	...
224	Saint-Étienne	45.44250	4.40286	Givors (South Lyon)	45.59105	4.77193	35.00	37.148	2173.2 2.60
225	Givors (South Lyon)	45.59105	4.77193	Lyon Perrache (Central Bus Station)	45.74875	4.82602	20.00	23.717	1387.4 1.66
226	Dunkerque	51.03119	2.36896	Lille	50.63920	3.07620	60.00	82.963	4853.3 5.81
227	Calais (Bus Stop on Terminal Ferry)	50.96679	1.86345	Lille	50.63920	3.07620	90.00	116.484	6814.3 8.15
228	Marseille (central bus station St Charles)	43.30408	5.37999	Avignon (Le Pontel)	43.96063	4.85594	77.50	98.585	5767.2 6.90

FIGURE 9 – DataFrame des données de bus nettoyées

	point_1			point_2			duree (min)	distance (km)	empreinte carbone (gCO2)	prix (euros)
	nom	latitude	longitude	nom	latitude	longitude				
0	Gare de Cormery	47.26235	0.83382	Cormery-Pl-du-Croissant	47.26861	0.83527	1.95	0.705	7.6	0.06
1	Cormery-Pl-du-Croissant	47.26861	0.83527	Truyes-Chapelle	47.27528	0.83055	1.96	0.824	8.9	0.07
2	Gare de Loches	47.13040	1.00097	Loches-St-Jacques	47.13755	0.99982	2.11	0.801	8.7	0.07
3	Chambourg-Chopin	47.18221	0.96805	Azay-Rivieres	47.20921	0.94616	3.86	3.431	37.1	0.31
4	Chambray-CHR-Trousseau	47.34777	0.71000	Gare de Tours	47.38981	0.69351	15.28	4.842	52.3	0.44
...	...	...	...	...	...	...	...	...	...	...
3293	Gare de Albens	45.78613	5.94855	Grésy-S/Aix-Ondea	45.72324	5.91845	7.00	7.381	79.7	0.66
3294	Grésy-S/Aix-Ondea	45.72324	5.91845	Gare de Aix-les-Bains-le-Revard	45.68786	5.90935	14.00	4.001	43.2	0.36
3295	Gare de Allevilliers	47.91387	6.33739	St-Loup-sur-Semouse	47.88563	6.27575	10.00	5.572	60.2	0.50
3296	St-Loup-sur-Semouse	47.88563	6.27575	Gare de Luxeuil-les-Bains	47.81492	6.37268	17.00	10.695	115.5	0.96
3297	Gare de Champagney	47.70295	6.70125	Gare de Belfort-Ville	47.63245	6.85392	27.00	13.877	149.9	1.25

FIGURE 10 – DataFrame des données d’intercités nettoyées

	point_1			point_2			duree (min)	distance (km)	empreinte carbone (gCO2)	prix (euros)
	nom	latitude	longitude	nom	latitude	longitude				
0	Gare de Bordeaux-St-Jean	44.82654	-0.55619	Gare de Cenon	44.85677	-0.53360	4.46	3.808	113.1	0.30
1	Gare de Mouchard	46.97681	5.79972	Gare de Arc-et-Senans	47.03046	5.77693	5.28	6.218	184.7	0.50
2	Gare de Laroche-Migennes	47.96085	3.51294	Gare de Joigny	47.97383	3.39296	7.93	9.058	269.0	0.72
3	Gare de Entzheim-Aéroport	48.54700	7.62795	Gare de Molsheim	48.53752	7.50021	7.57	9.473	281.3	0.76
4	Gare de Montmélian	45.50307	6.04312	Gare de Chambéry-Challes-Eaux	45.57104	5.91980	12.36	12.235	363.4	0.98
...	...	...	...	...	...	...	...	...	...	...
5750	Gare de Champagney	47.70295	6.70125	Gare de Belfort-Ville	47.63245	6.85392	27.00	13.877	412.1	1.11
5751	Gare de Mulhouse	47.74180	7.34285	Gare de Belfort-Ville	47.63245	6.85392	26.50	38.608	1146.7	3.09
5752	Gare de Valence-Ville	44.92807	4.89329	Gare de Veynes-Dévoluy	44.53200	5.81583	122.00	85.244	2531.7	6.82
5753	Gare de Châteaubourg	48.10615	-1.40530	Gare de Les Lacs	48.10903	-1.32181	5.00	6.214	184.6	0.50
5754	Gare de Les Lacs	48.10903	-1.32181	Gare de Vitre	48.12242	-1.21197	8.00	8.298	246.5	0.66

FIGURE 11 – DataFrame des données de TER nettoyées

## B Graphes

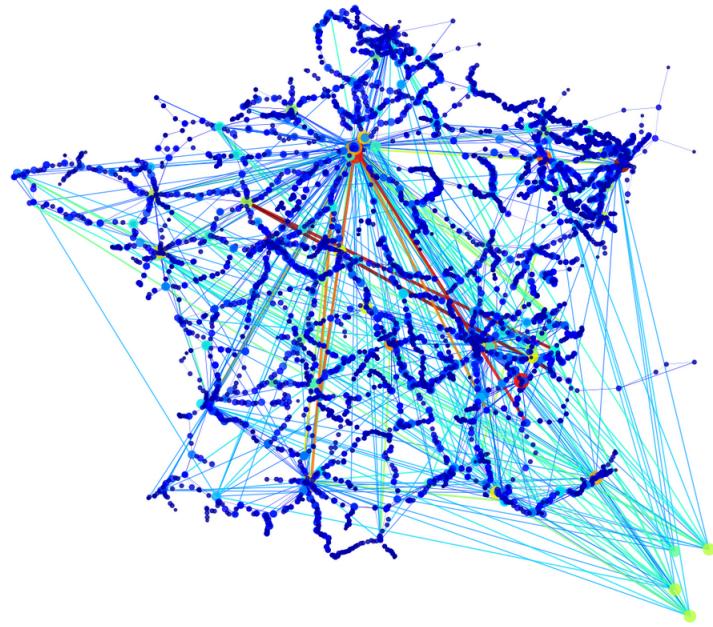


FIGURE 12 – Illustration du graphe de tout les moyens de transport : taille du noeud en fonction du degré et épaisseur de l'arête en fonction du poids

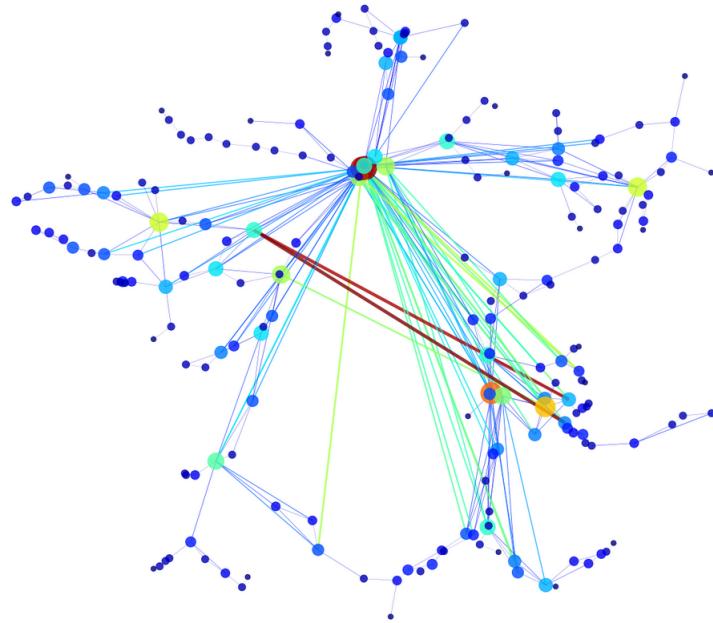


FIGURE 13 – Illustration du graphe des TGV : taille du noeud en fonction du degré et épaisseur de l'arête en fonction du poids

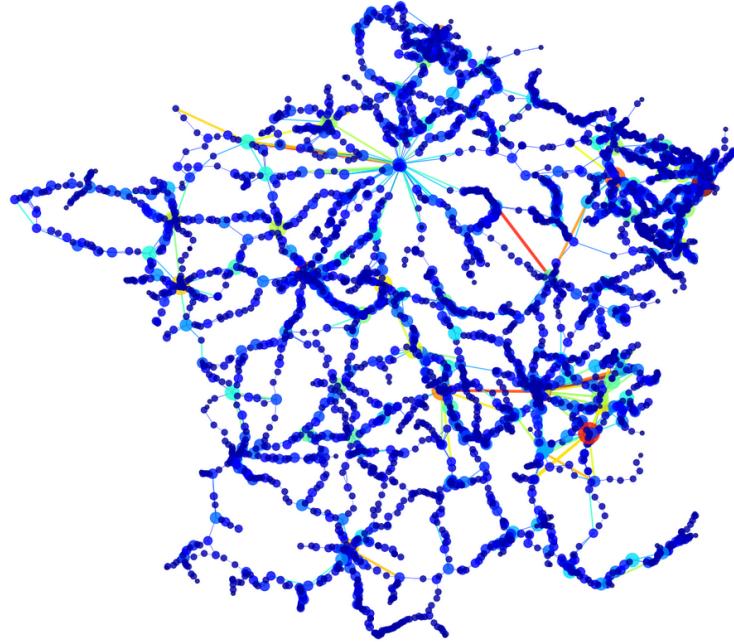


FIGURE 14 – Illustration du graphe des TER : taille du noeud en fonction du degré et épaisseur de l'arête en fonction du poids

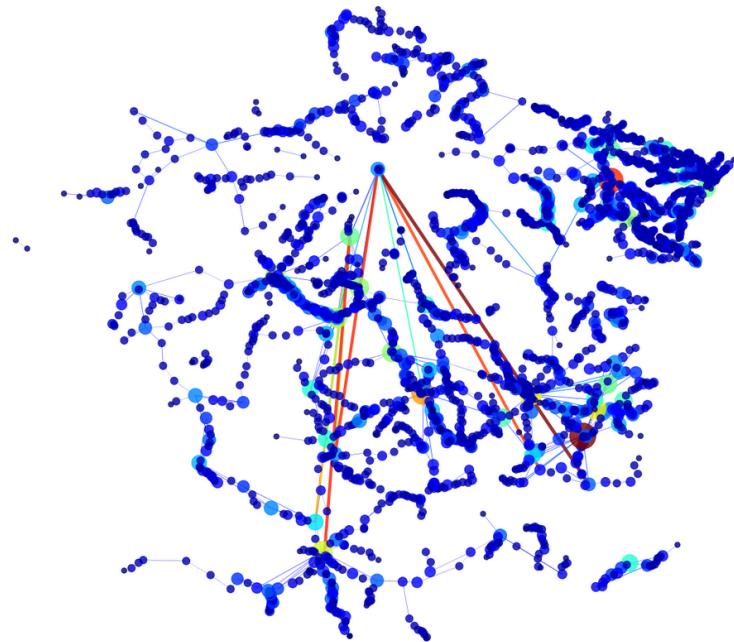


FIGURE 15 – Illustration du graphe des intercités : taille du noeud en fonction du degré et épaisseur de l'arête en fonction du poids

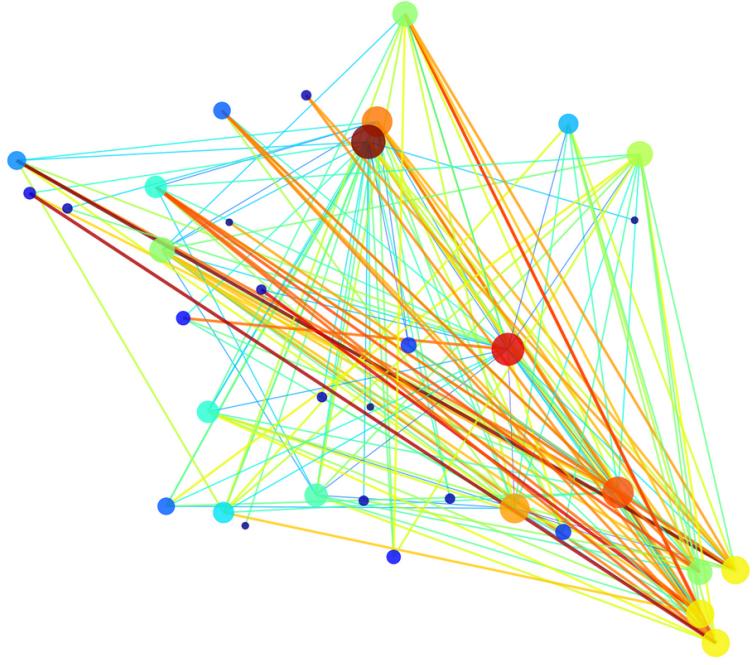


FIGURE 16 – Illustration du graphe des avions : taille du noeud en fonction du degré et épaisseur de l'arête en fonction du poids