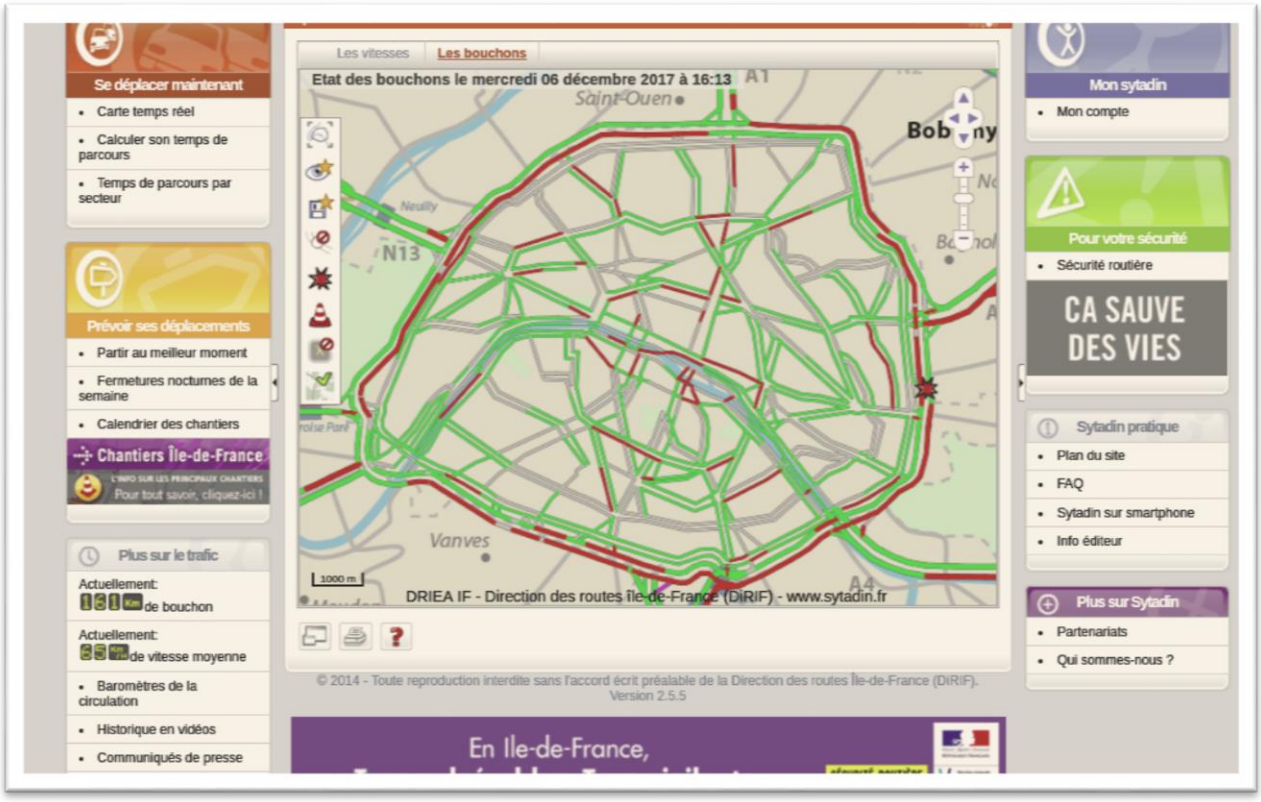
Rapport Projet
MandarinaTraffic

Un pale simulacre de Sytadin.fr



1) Premières intuitions

Nous voulions réaliser un modèle de prédiction de trafic d'Île-de-France, dans le but de :

- prévoir le trafic à court terme (amorti avec les données du temps réel)
- prévoir le trafic à long terme (attribué une couleur a un jour et une zone donné)

Pour cela nous avions envisagé d'utiliser les données :

RATP	• géolocalisation des stations et arrêts • données sur les travaux, fermeture de ligne • statistique par jour et par stations
Trafic routier	• géolocalisation des capteurs • valeur des capteurs de trafic • données sur les travaux, fermeture nocturne ...
Événements Majeurs	• Date/heure/durée • Lieux des manifestations, fête
Météo	• géolocalisation des capteurs • température, pluie, neige, forte chaleur

Nous n'avions à ce moment aucune intuition particulière sur comment résoudre ce problème et donc pas de stratégie bien définie dans nos recherche et résonnements.

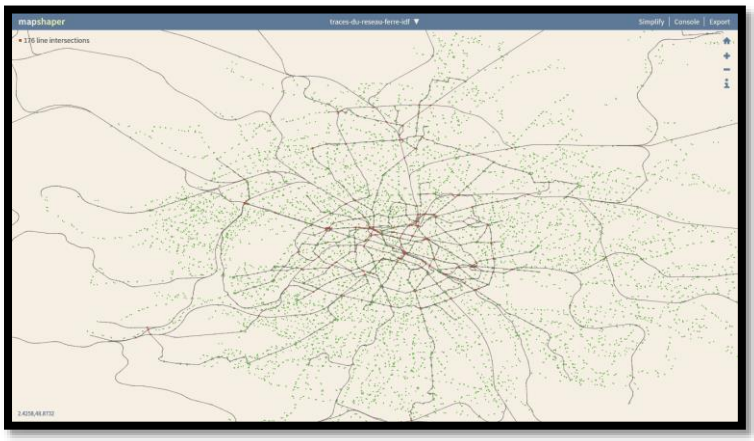
2) Récolte de données

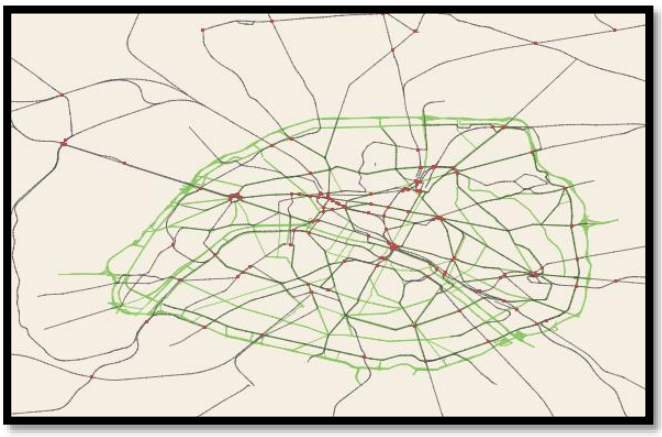
2.1) Difficultés et choix

Nous n'avons pas réussi à trouver les données du trafic de toute l'Île-de-France (seulement sont mis à dispositions celle de paris depuis 2013 dont le niveau de précision est par heure) et l'historique des fermetures de voies pour raison de travaux ou événement exceptionnel. Pour le trafic de la RATP est disponible toute les données d'arriver en gare ou en arrêt de métro, bus d'Île-de-France. Fasse a l'importance des données déjà trouvé (plus de 6Go), nous nous somme limité à cela

Premièrement nous avons dû comprendre les données que nous avions amassé, pour cela nous avons utilisé des outils de plot en ligne pour les fichiers **GeoJson** comme **mapShaper.org** nous avons pu prendre connaissance de l'importance des données :

- à gauche : Chaque point vert est un arrêt (Bus/Métro/RER) de la **RATP**. Les lignes noires : le réseau ferre
- à droite : Les capteurs de trafic de paris et du périphérique





Face à cette quantité de données nous avions fait le choix de ne conserver que les données des 57 Bus parisien (du n°20 au 96) pour ne nous préoccuper que du trafic routier des lignes de la RATP.

La compréhension et les possibilités de réutilisations des données de trafic de la RATP nous ont posé difficulté, ne trouvant pas rapidement un moyen cohérent de les exploités nous avons choisi de poursuivre avec uniquement les valeurs de trafic routier de paris.

Il existe un projet qui réutilise les valeurs de trafic routier pour d'afficher uniquement des animations du trafic parisien. Mis à part quelques indications sur les données nous n'avons pas ré-exploité son code (pas documenté et très hermétique.)

https://github.com/astyonax/heartbeat-traffic#or-the-heartbeat-of-paris

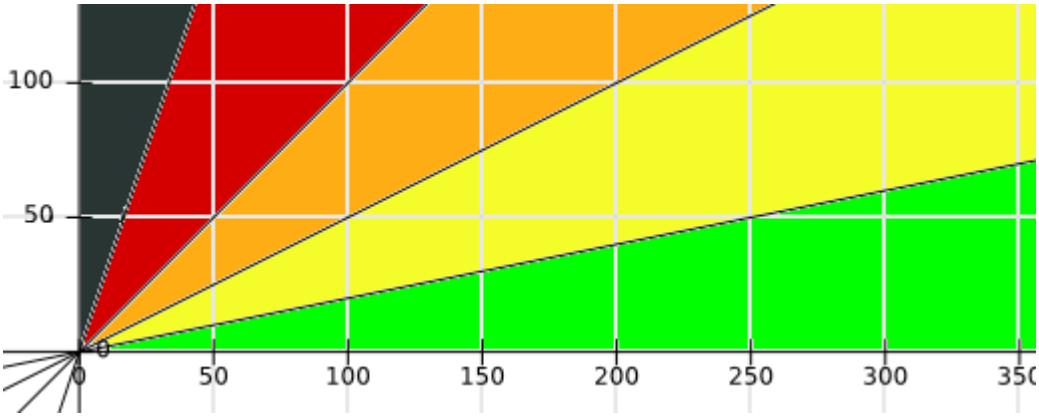
2.2) Intuitions intermédiaire et compréhension de données

Pour chaque capteur nous pouvons obtenir le débit (nombre de voiture passé en 1h) et le taux d'occupation *(en pourcentage, 25% → il y a eu au moins durant 15 minutes une voiture au-dessus du capteur)*

Pour évaluer le niveau d'embouteillage d'une zone avec un ensemble de capteur on peut classifier en k classes le niveau d'embouteillage :

par exemple k =5

[0 → Très embouteillé] [1→ embouteillé] [2→ assez fluide] [3 → fluide] [4 →très fluide]



Mais pour les zones avec peu de capteur faire une moyenne des valeurs entourantes peut présenter un problème en cas de rues au trafic indépendant (comme une voie rapide) .

Dans le cas de paris elles sont rare (il n'y a pas d'autoroute intra-muros, comme à Bangkok) donc on peut considérer que faire des moyennes de zone ne sera pas source de problème

Bien évidement ce nombre de zone jouera sur la correction du model et il faudra éviter le sur apprentissage

La DIRIF fournit des informations très limitées sur la récupération des données en Ile-de-France
http://www.dir.ile-de-france.developpement-durable.gouv.fr/la-methodologie-du-recueil-de-donnees-a168.html

3) Parsage des données et remplissage BD

Nous n'avons pas questionné le choix du langage de programmation et donc avons naturellement utilisé Python, malheureusement ses performances pour le parsage et la construction de BD sont particulièrement faible, à l' avenir nous préconiserons de séparé le code de minage des données et celui d'exploitation.

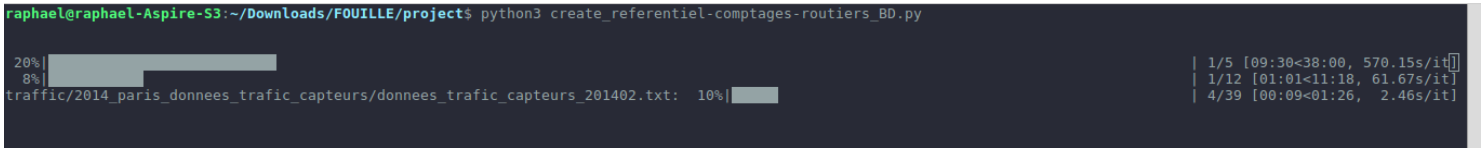
Face à la quantité importante de données ~ 4 Go nous avons choisi de stocké les informations dans une BD **SQLite** pour pouvoir effectué des requêtes SQL et ainsi en tiré des résultats facilement.

Pour le parsage des données nous avons utilisé la librairie **pandas** qui simplifie l'écriture du code et se base sur **numpy** pour l'efficacité. (bien que en comparaison avec un parse manuel et des **insertBulk** dans la BD, la différence n'est pas flagrante) . Nous avons cherché à comprendre au maximum et optimisé au mieux l'usage de la librairie mais nous avons tout de même fait face à une certaine lenteur du processus, finalement nous avons parallélisé le traitement grâce à l'API **ProcessPoolExecutor** et utilisé **tqdm** pour les bars de progressions

Ainsi il a fallu environ 50 minutes pour parser et remplir la BD **SQLite** (de taille final de 5Go) *avec un i7 – 4cœur logique cadencé à 2.4GHZ et un SSD en sata 3*

Travaillé sur toute la base de données est coûteux (nous avons donc produit de nombreuses petites BD pour différents tests)

Nous avons donc automatisé la création des bases de données. Ainsi nous pouvons travailler uniquement sur une année, ou autre sous ensemble pour accélérer les tests.



Nous avons pour chaque année de 2013 à 2017, 12 fichier contenant respectivement les données des capteurs pour chaque moi

Les colonnes de notre BD sont donc :

Id capteur	Année	Numéro de Semaine	Numéro de Jour	Heure	valeurDebit	valeurTauxOcc

Année ∈ [2013 - 2017]
Numéro de Semaine ∈ [1 - 53] (2015 a été une semaine à 53 semaines)
Numéro de Jour ∈ [0 - 6]

Car il n'y aucune corrélation entre le 24 janvier et le 24 mars, mais il peut y avoir une entre le 3 mardi du mois de Janvier et le 3 mardi du mois de Février

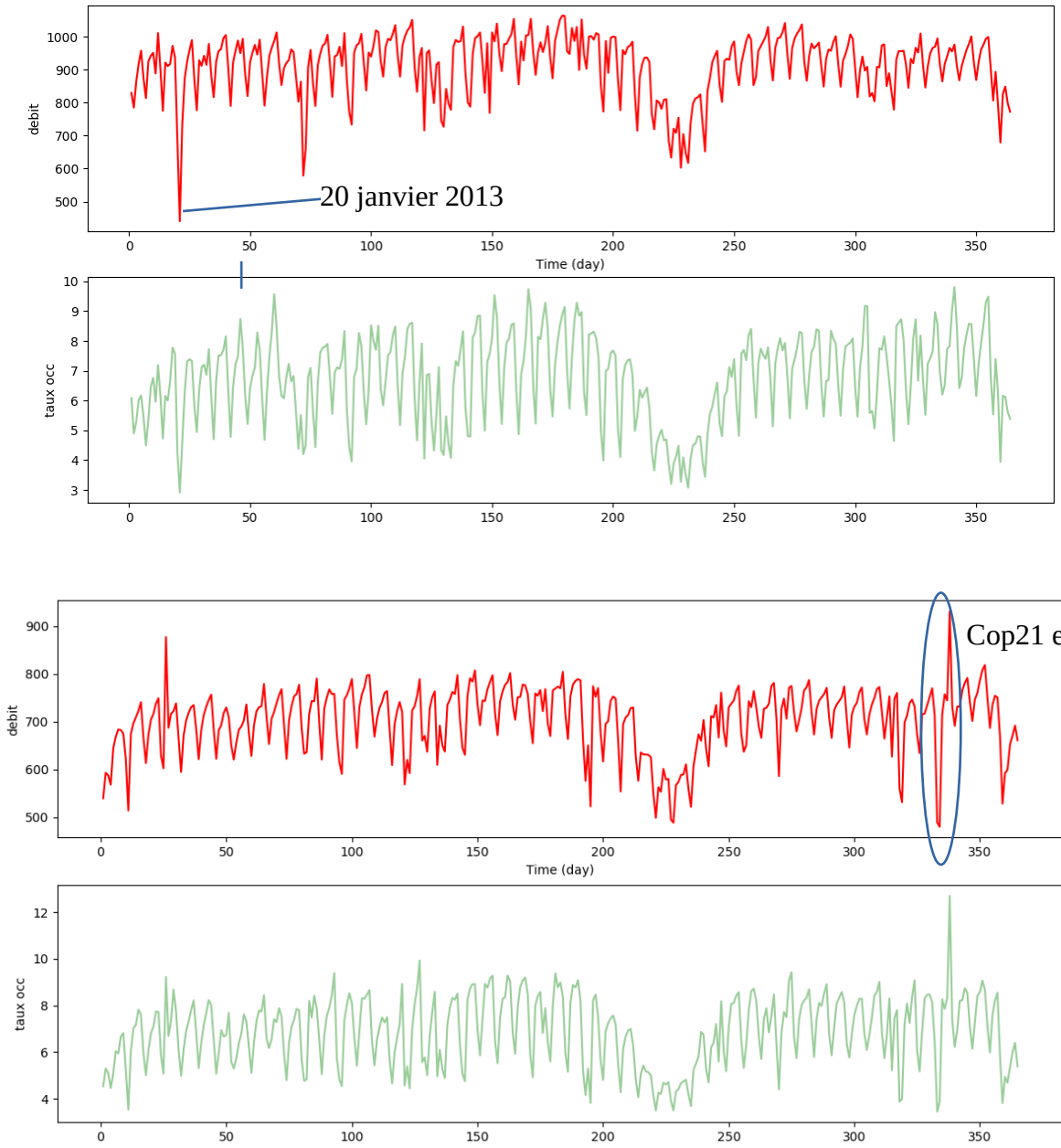
Les capteurs peuvent tomber en panne, les valeurs être aberrantes ou incomplète:

- Il y a 947 capteurs avec seulement le débit (28% des capteurs totaux) en 2013
- Beaucoup de valeur absente (Null), 48% des valeurs pour 2013 sont incomplètes (11193055 valeurs sur 23243130 sont complètes)
- Des valeurs avec un taux d'occupation supérieurs à 100% *(capteurs particulièrement présent sur le périphérique, peut-être sont-ils différent, mais la documentation n'indique rien)*
- Il y a des débits de 0 avec un taux d'occupation supérieur à 0. *(des véhicules en stationnement ?, zone de travaux ? …)*

5.2) Intuitions intermédiaire

Les moyennes par jour du débit et taux d'occurrence permettent de distinguer les fins de semaine comme le dimanche *(peu de circulation)* et le vendredi *(circulation élevée) a*insi que les périodes de vacances de Juillet-Août. Aussi le 20/01/2013 est un dimanche ou paris étais paralysé sous la neige. Que faire de ces extrêmes ?

Voici le plot des valeurs débit (rouge) et taux occupation (vert) par jour de l'année 2013 puis 2015



Les valeurs de débit et de taux d'occupation apparaissent corrélé, cela ce confirme avec le calcul de la corrélation dont le résultat est : 0,81
On pourrait donc classifier chaque jour dans une catégorie avec la moyenne de tous les capteurs et de toutes les heures.

4) Visualisation des données

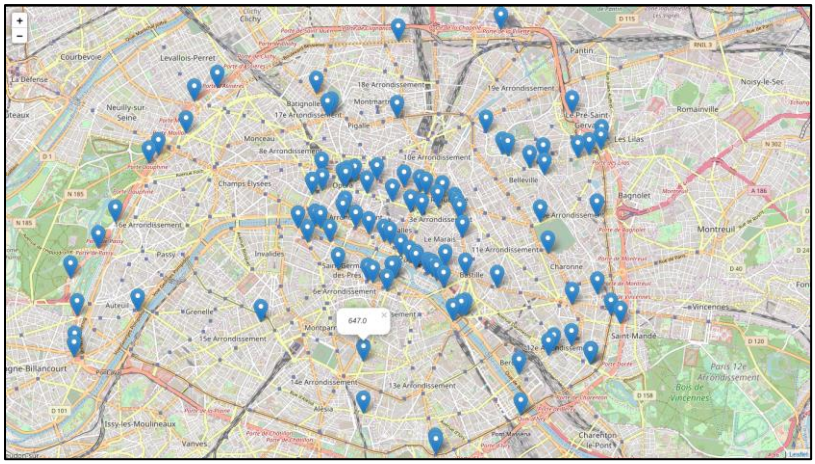
La grande quantité de données nous a poussés à écrire un grand nombre de requêtes SQL pour mieux comprendre ce qu'elle représentait, mais des moyennes et autres valeurs chiffrées ne nous donnaient que des intuitions statistiques. Donc pour mieux interprété ces résultats nous avons porté une grande attention à la visualisation des données. Nous voulions une représentation correcte des distances et donc nous passer du travail de conversion et transformation cartographique (qu'offre **pyproj**)

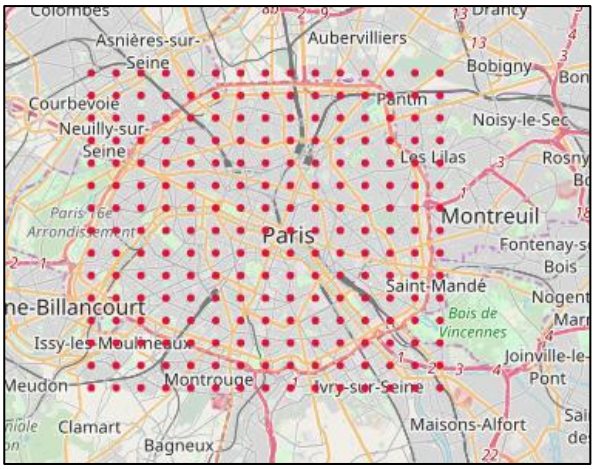
Le plot des positions de capteurs nous a posé problème, nous avons perdu du temps a essayé différents outils et librairies (**descartes** , **shapely** ...) pour finalement privilégié une libraire de génération d'HTML/JavaScript : **folium**)

(Seul défaut de cette librairie, les objets ne sont pas pickable et donc la parallélisassions n'est pas efficace, nous avons créé une issue sur le repo github de folium)
Grâce à la BD et nos fonctions d'export HTML nous pouvons visualiser automatiquement le résultat d'une requête.

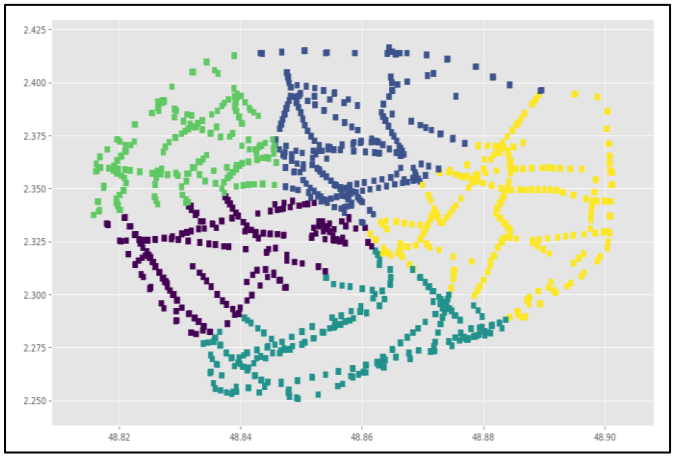
Voici différents exemples :

Une représentation de la liste des capteurs
sans valeur pour le taux occupation

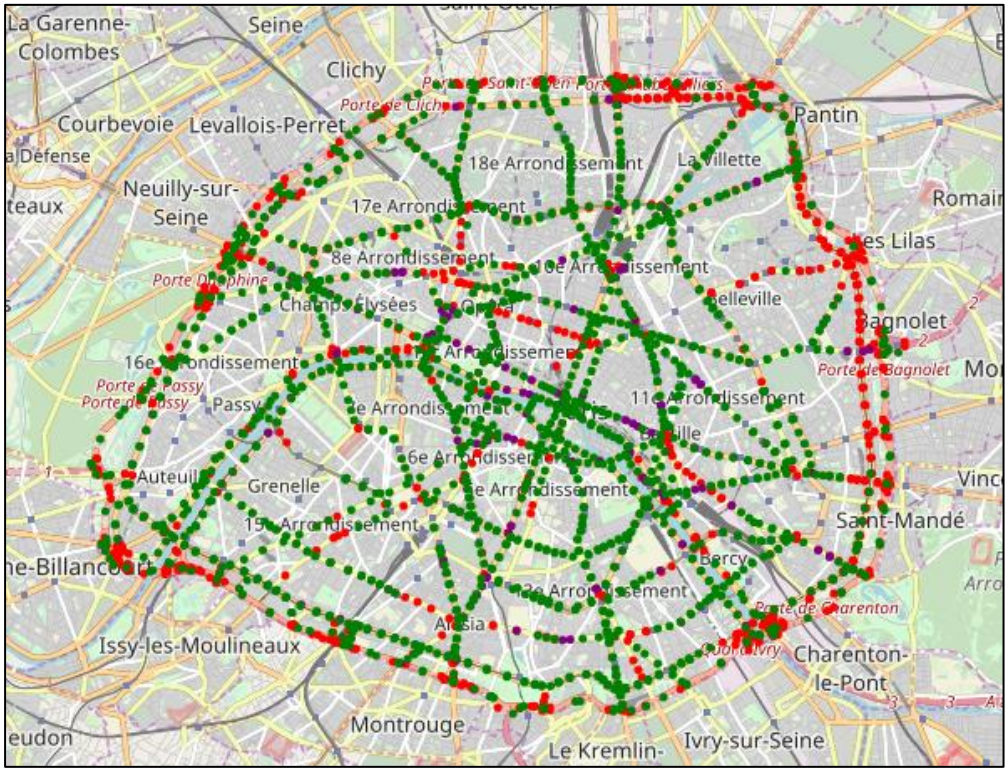




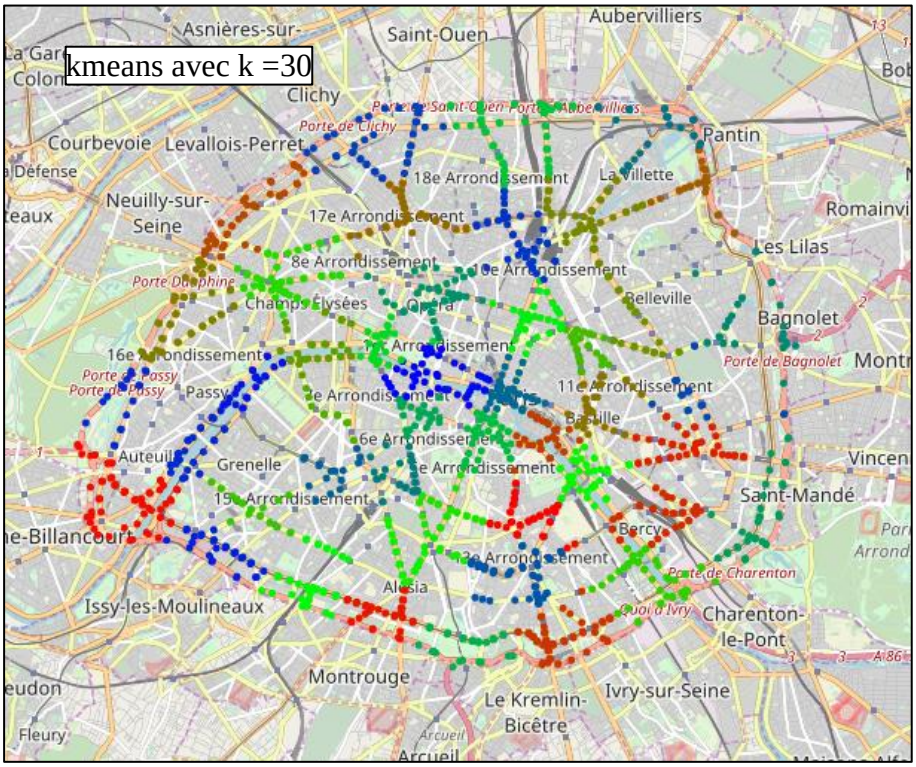
Une représentation d'une liste de zones d'influence, une matrice 15*15



Plot sans conversion des valeurs GPS au plan cartésien Avec k=5



Les capteurs ne renvoyant aucune donnée plus de 70 % de l'année peuvent être considéré défectueux (en rouge sur la carte)



Finalement crée des zones de trafic avec la méthode des kmeans apparaît comme hasardeuse car nous pourrions associer une zone extrêmement embouteillé a un autre et la moyenne n'aurait pas de sens
(Par exemple un échangeur autoroute-périphérique avec un boulevard parisien)
Ainsi il faudrait mieux grouper les capteurs par zone et ne pas groupé des capteurs du périphérique avec ceux de l'intérieure de paris

5) Jeux de Test et entraînement

La visualisation nous a permis de constaté la présence d'un grand nombre de données difficile à interpréter et à réutiliser, ainsi de nombreuses possibilités de filtrage de la BD s'offre à nous pour ignorer les valeurs ou

capteurs problématiques ainsi que les jours au trafic exceptionnel (valeur extrêmes) .Ainsi il faudrait testé le(s) modèles avec ces différentes options pour faire apparaître le(s) filtrage le(s) plus pertinent

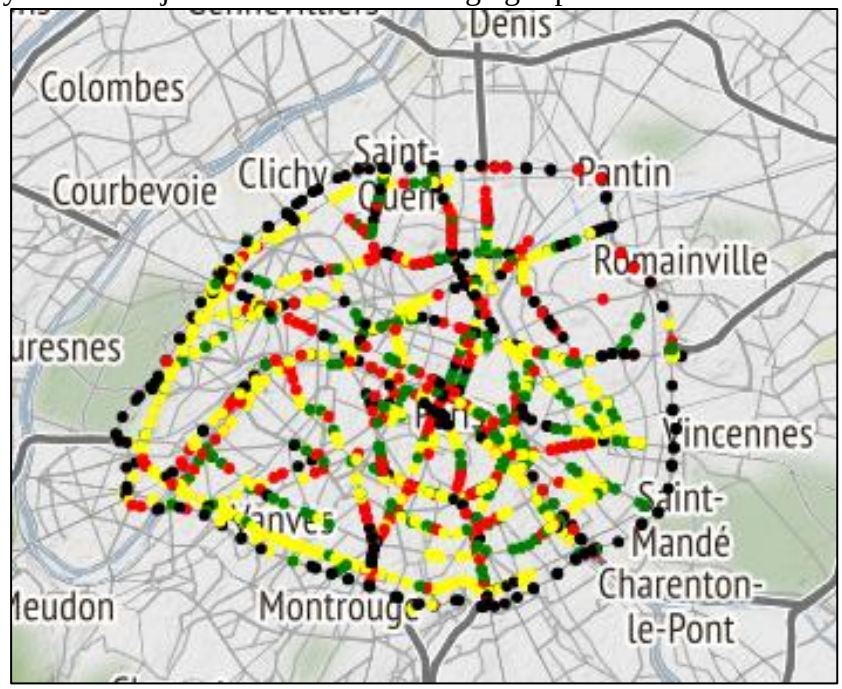
Maintenant se pose la question du niveau de précision des prédictions que nous souhaitons obetenir

Différentes possibilité s'offre à nous :

Type de mesure statistique *Moyenne simple , moyenne pondérée ,médiane ...*	Intervalle de temps par :	Nombre de cluster : *1 → nbCapteurs*	Variables: La couleur d'embouteillage : *Catégorielle/Quantitative Discrète/continue*
	Année		
	Mois		
	Semaine		
	Jour		
	K Intervalles d'heure		
-	Heure		

Nourrir le model avec des moyennes de valeur par Année ou Mois ou Semaine sur chaque capteur apparaît comme bien trop pauvre.
Nous aurons donc au maximum 5 dimension d'input et 1 dimension de sortie
Input : Année , Semaine , Jour , Heure , idCluster
Output : Couleur

Exemple :
Intervalles d'heure → [00h-7h] [7h – 9h] , [9h – 16h] , [16h – 20h] , [20h – 00h]
la moyenne pondéré avec comme coefficient par intervalle d'heure → 1 , 4 , 3 , 4 , 1
Discrétisation des valeurs de trafic en 4 classes de niveau d'embouteillage [vert , jaune ,rouge ,noir]
Clusters = nbCapteurs
Voici donc le trafic « moyen » du 10 janvier 2013 avec les réglages précédents :



Bien évidement choisir un jeu d'entraînement constitué de 2013 à 2016 et un jeu de test de 2017 est aberrant car des travaux ou fermeture de vois de circulation peuvent avoir fini en 2017 et l'inverse.

Nous considérerons donc comme bien plus importants les données récentes. Ainsi avec une méthode randomisé et pour p la probabilité globale de choisir une valeur, nous prendrons une valeur d'une année K avec la probabilité pk

$$\sum pk = p$$

2017	2016	2015	2014	2013

Ainsi nous privilégierons l'apprentissage sur les données les plus récentes. Une fois l'ensemble des valeurs choisies nous pouvons le diviser en ensemble de test et apprentissage. Nous avons choisi un ratio d'apprentissage de 77% - 33% pour éviter le sur apprentissage.

Le model suffisamment performant sur les valeurs de test (proche de celui d'entraînement), nous avons testé son niveau d'efficacité en prédiction sur le « futur ». Et pour cela nous disposions des valeurs de novembre 2017 totalement absentes des valeurs d'apprentissage fournit au model.

6) Modélisation

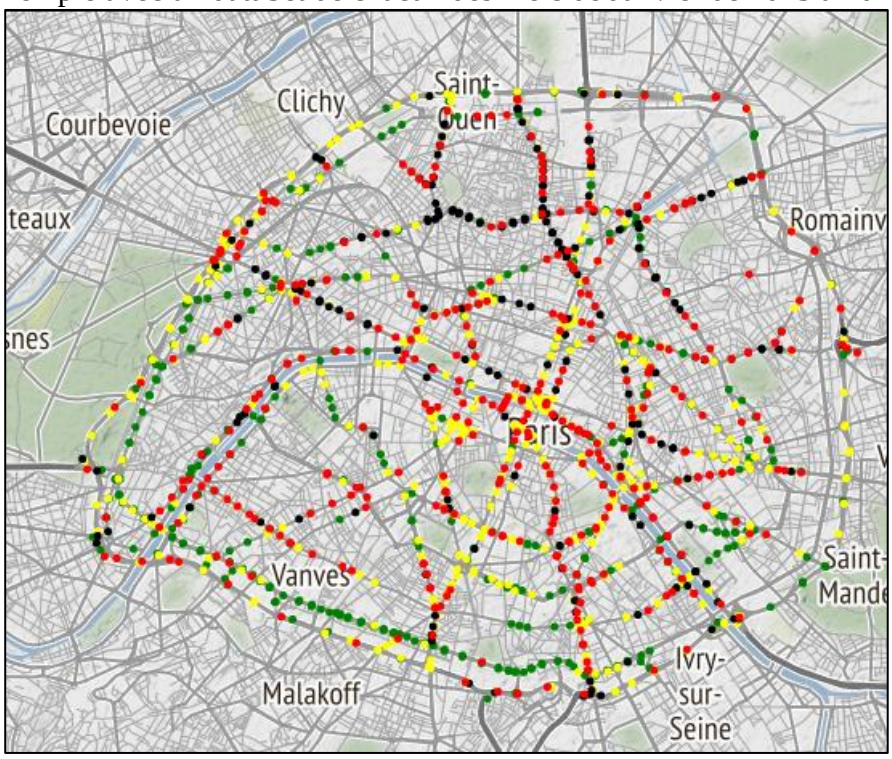
6.1) Créations

6.1.1) Premier Test

Pour commencer nous sommes partis sur l'idée d'utiliser un simple arbre de décisions, car il permet de facilement interpréter les décisions du model (pour cela nous exportons un fichier .dot consultable avec **graphvz**), et donc par la suite nous permettre de comparer les différents choix de filtrage.

Pour le premier test, nous avons fait 4 classes (Green, Orange, Red et Noir) et nous avons entraîné le model avec les valeurs filtrés de la BD .Le but est donc de prédire la class d'un capteur selon une heure un jour, une semaine et une année.
Pour éviter le sur-apprentissage, nous avons utilisé la validation croisée simple (un groupe apprentissage et un autre test). Cela nous permettra d'ajusté les paramètres de l'arbre de décision.

Nous avons obtenu une accuracy aux alentours de 60% pour ce premier test.
Exemple avec un dataSet de chacun des mois de Janvier de 2013 à 2017 inclus.

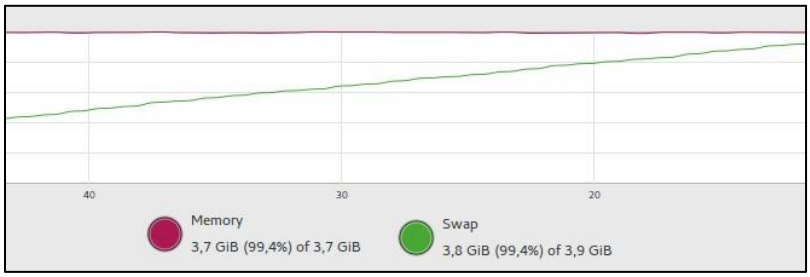


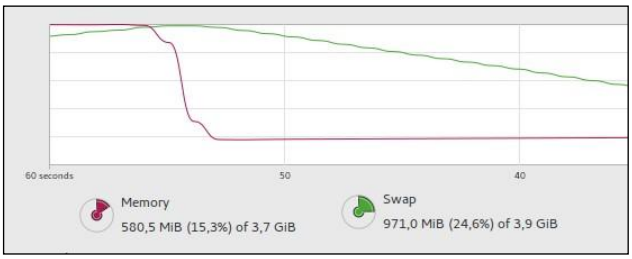
L’arbre de décision nous a permis de nous rendre compte de plusieurs chose et de vérifié quelques intuitions :

- L’heure a une influence importante sur le trafic.
- Le jour de la semaine a une influence sur le trafic.
- L’arbre ne tiens pas toujours compte des semaines (surtout quand le data Set est inférieur à celui d’un an de valeur)
- L’arbre fait des choix par rapport aux id capteur (ce que nous n’avions pas du tout envisagé). Cela peut se justifié car il existe une logique géographique dans la distribution des Id des capteurs.

Les calcule se sont avéré particulièrement lourd, surtout en mémoire. On monte à 5Go de ram avec deux ans de data ce qui nous conforte dans l’idée de filtrer et randomiser le choix des valeurs pour le modèle.

Même en ne travaillant sur une BD avec qu’une année, les requêtes prennent environ 5mn (sur nos ordinateurs personnels ayant 4 Go de RAM et un SDD). Si nous prenons plus il arrive parfois que le system KILL notre processus, due à la grande mémoire solliciter. Nous avons optimisé nos requêtes et nos calculs afin de baisser la complexité. Des améliorations sont notables, mais toujours pas suffisantes pour espérer avoir un résultat dans un temps raisonnable.

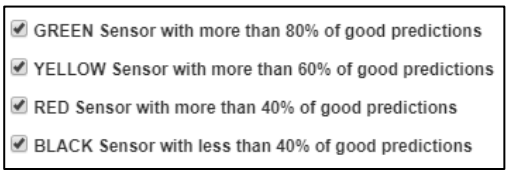




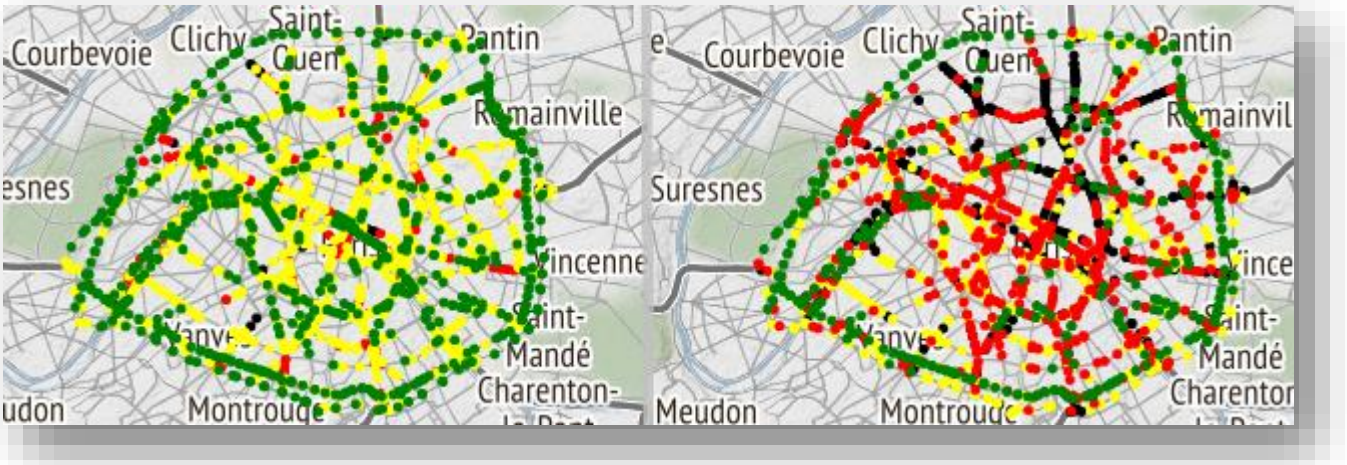
6.1.2) Deuxième test

À partir de maintenant nous n’utiliserons que les données de novembre 2016 à novembre 2017 pour des raisons de performance

Après avoir essayé l’algorithme d’arbre de décision, nous avons essayé l’algorithme de Random forest. Nous sommes passés à 80% d’accuracy (avec 10 arbres et tous les features). Ainsi une immense majorité des prédictions son correcte a plus de 60 %. Il n’y ainsi que 0,2% des capteurs avec une correction inférieure à 40%.



Aussi nous pouvons sortir une représentation visuelle des capteurs par classe de correction (de très bon a mauvaise prédiction). Ci-dessous les résultats de prédiction du mois de novembre 2017 :
Foret aléatoire (à gauche) et Arbre de décision (à droite) pour un model nourri avec les valeurs de novembre 2016 à octobre 2017



Prédiction complètes de novembre

6.2) Analyse résultats

Ayant un petit nombre de classes (différents niveau d'embouteillages) nous pouvons analyser le résultat avec l'erreur absolue moyenne. Ainsi pour la prédiction du mois de novembre 2017 nous obtenons le même niveau de correction que la fonction de score du jeu de test (80%)

7) Possibilités d'extensions :

L'ensemble du code mis a disposions est complètement réexploitable pour d'autres type de filtrage sur les données, input pour les modèles, prédictions et visualisations.
Nous pourrions par exemple :

- Faire la prédiction du trafic par jour avec une moyenne pondéré de tous les capteurs dans une journée.
- Nous aurions aimé ajouter la latitude et la longitude aux dimensions de notre modèle pour estimer les zones d'influence. Nous pensons que cet ajout de dimension aurait pu augmenter l'accuracy de nos modèles.

- Nous avons pas eu le temps d'envisager de nourrir le model avec des données récupérer en temps réel (il n'y a pas d'API public, mais le site internet du **sytadin.fr** propose de rentré en contact pour établir des partenariats :

  Pour toute demande complémentaire de renseignements,

  vous pouvez envoyer un mail à : sytadin@developpement-durable.gouv.fr

- On aurait pu utiliser les données des bus de la RATP pour enrichir notre modèle. Même si l'on aurait eu un manque de puissance de calcul.

8) Conclusion

Un temps non négligeable a été nécessaire pour comprendre les librairies et différents outils. De même nous somme tous les deux habitué à la programmation bas niveau et à travailler sur des problèmes au énoncé « clair ». Python n'est pas un langage que nous utilisons souvent, et donc le sucre syntaxique maintenant rentré dans les mœurs est une petite barrière à la compréhension et récupération de code.
Aussi le fait d'être dépendant de la bonne qualité des données amassé est nouveaux pour nous et nous n'avions jamais eu à faire à un projet aussi orienté statistique.
Comme vous l'aviez dit, beaucoup de décisions se prennent grâce au bon sens (intuitions) et ce projet nous a permis de mieux comprendre l'impact.

Finalement notre projet a été orienté data Mining et data Visualisation, ce qui nous à permit de créer un ensemble d'outils nécessaire pour développer dans le futur la partie décision et modélisation que nous n'avons pas eu le temps de formellement accomplir.