Rapport Projet Pycling

Table des matières

Contexte

Depuis 2 ans, les déplacements en vélo à Paris se sont accrus et ont explosé durant la crise sanitaire. Ils représentent aujourd'hui 5.6% des déplacements vs 9% en voiture.

Ainsi, en octobre dernier, la ville de Paris a annoncé un plan d'investissement de 250 millions d'euros pour pérenniser les « coronapistes », pistes cyclables qui ont vu le jour durant le confinement sur les grands axes parisiens ; ce qui démontre une volonté de massifier cet usage.

Mais le boom de la pratique du Vélib à Paris se poursuit-elle en 2021 ?

Nous étudierons l'évolution du trafic cycliste au travers de données mise à disposition par la Mairie de Paris sur la période du 1er Septembre 2020 au 31 Octobre 2021, en 3 étapes :

I. Data Visualisation : les facteurs explicatifs du trafic

II. Cartographie : l'intensité du trafic par localité

III. Machine Learning : la prédiction du trafic grâce aux modèles prédictifs

Le dataset

1. *Source*

Notre jeu de données est issu du site de la Mairie de Paris disponible en Opendata pour évaluer le développement de la pratique cycliste : Comptage Vélo

2. *Fenêtre temporelle*

Ce jeu de données présente l'ensemble des comptages vélo horaires sur 13 mois glissants (J-13 mois), mis à jour à J-1. Afin d'apporter une analyse comparative, nous avons récupéré les données complètes d'octobre 2021. Nous disposons de 14 mois de données allant du 1er Septembre 2020 au 31 Octobre 2021 à partir de 97 compteurs.

3. *Données*

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 966708 entries, 0 to 966707
Data columns (total 10 columns):
 #   Column                                 Non-Null Count   Dtype
---  ------                                 --------------   -----
 0   Identifiant du compteur                966708 non-null  object
 1   Nom du compteur                        966708 non-null  object
 2   Identifiant du site de comptage        966708 non-null  int64
 3   Nom du site de comptage                966708 non-null  object
 4   Comptage horaire                       966708 non-null  float64
 5   Date et heure de comptage              966708 non-null  object
 6   Date d'installation du site de comptage 966708 non-null  object
 7   Lien vers photo du site de comptage    966708 non-null  object
 8   Coordonnées géographiques              966708 non-null  object
 9   Identifiant technique compteur         960241 non-null  object
dtypes: float64(1), int64(1), object(8)
memory usage: 73.8+ MB
```

Notre variable cible sera le “Comptage horaire”, la seule variable numérique qui correspond au **nombre de vélos/heure**.
Nous n’avons que deux variables explicatives à :

- Dimension Temporelle : ‘Date et heure du comptage’
- Dimension Géographique : ‘Coordonnées géographiques’.

4. Préparation des données

Langage utilisé : Python
Librairies utilisées : Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn
Module : Folium, Shap, XGBoost
Taille du DataFrame : 966 708 lignes x 9 colonnes
Doublons : aucun

Valeurs manquantes :
L'Identifiant technique compteur' est la seule variable comportant des valeurs manquantes, qui représentent 0.7% du dataset. Nous avons décidé de la supprimer car nous n'utiliserons pas cette donnée.

Suppression de variables (inutiles ou redondantes) :

- ‘Coordonnées géographiques’ a été retraitée, séparée en 'Latitude' et 'Longitude'. Nous pourrons ainsi identifier les axes avec les plus gros trafics.
- ‘Identifiant du compteur’
- ‘Identifiant du site de comptage’
- ‘Nom du site de comptage’

Ajout de variables :
Afin d’enrichir notre dataset, nous avons créé 18 nouvelles variables à partir de :

- <u>Source interne</u>

- Période : Jour, jour de la semaine, Mois, Année, Mois-année, Week-end
- Congés : Vacances scolaires, Jours fériés
- Géographique : Arrondissement

- <u>Source externe</u>

- Météorologique : Température (Minimum, maximum, moyenne), Pluie en mm
- Evènements exceptionnels : Confinement, couvre-feu.

I. Data Visualisation : les facteurs explicatifs du trafic

Tests ANOVA :

Le test Anova permet de tester si la relation entre 2 variables est statistiquement significative. C'est une analyse de variance qui nous permet d'examiner les différences entre les groupes.

Nous utiliserons cette analyse lorsque l'une des variables est quantitative et l'autre catégorique, comme c'est le cas ici avec nos différentes variables catégorielles et notre comptage horaire quantitatif.

	df	sum_sq	mean_sq	F	PR(>F)
Mois_annee	13.0	2.319608e+08	1.784313e+07	2368.766157	0.0
Residual	966694.0	7.281787e+09	7.532670e+03	NaN	NaN

	df	sum_sq	mean_sq	F	PR(>F)
Date	425.0	4.578108e+08	1.077202e+06	147.51844	0.0
Residual	966282.0	7.055937e+09	7.302151e+03	NaN	NaN

	df	sum_sq	mean_sq	F	PR(>F)
Jour_semaine	1.0	4.667309e+07	4.667309e+07	6042.413174	0.0
Residual	966706.0	7.467075e+09	7.724246e+03	NaN	NaN

	df	sum_sq	mean_sq	F	PR(>F)
Heure	23.0	1.970707e+09	8.568292e+07	14942.755727	0.0
Residual	966684.0	5.543041e+09	5.734077e+03	NaN	NaN

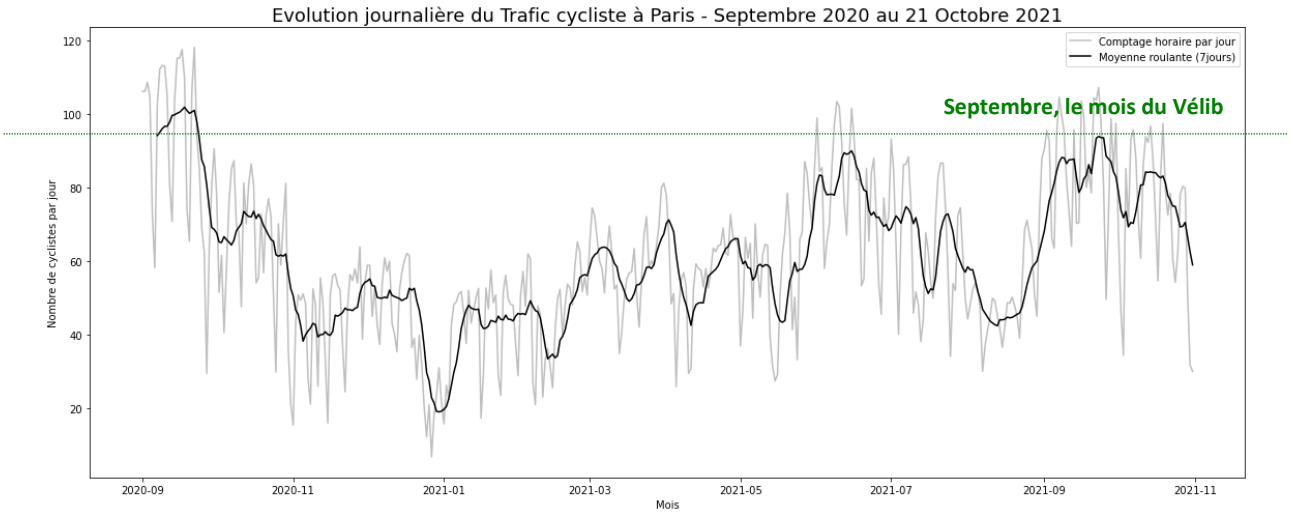
	df	sum_sq	mean_sq	F	PR(>F)
Temperature_moy	1.0	1.392219e+08	1.392219e+08	18250.214148	0.0
Residual	966706.0	7.374526e+09	7.628510e+03	NaN	NaN

Lorsque la valeur p est inférieure à 0,05, nous concluons qu'il existe une relation significative entre les variables

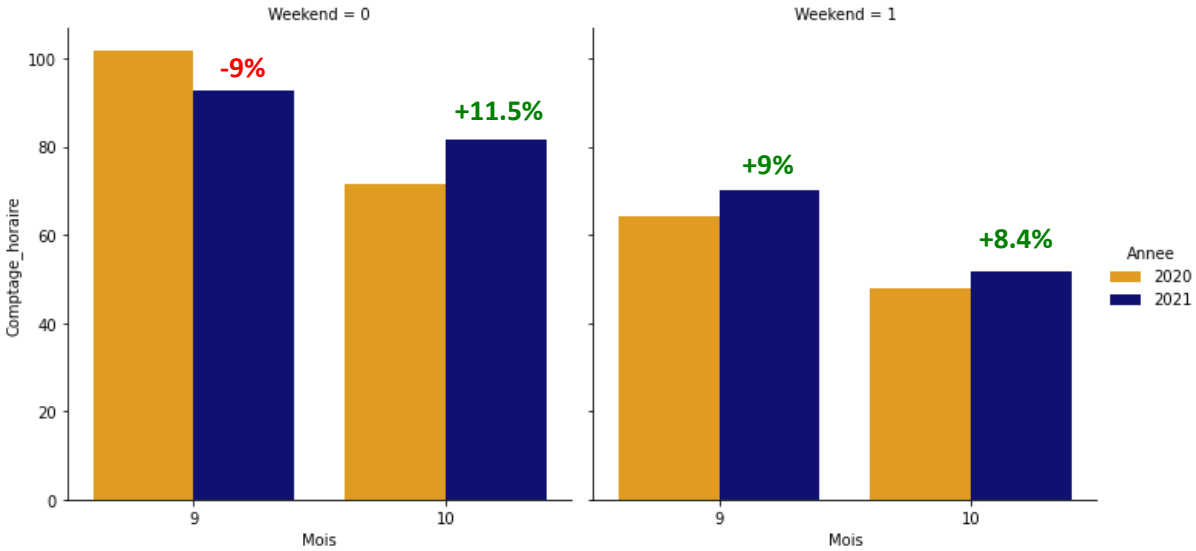
Dans les tests Anova ci-dessus, nous constatons que la p-value obtenue est nulle. Nous concluons donc qu'il existe une relation significative entre notre variable cible et ces différentes variables, que nous allons illustrer de manière visuelle cette fois dans les pages suivante.

L'usage du vélo à Paris se caractérise par :

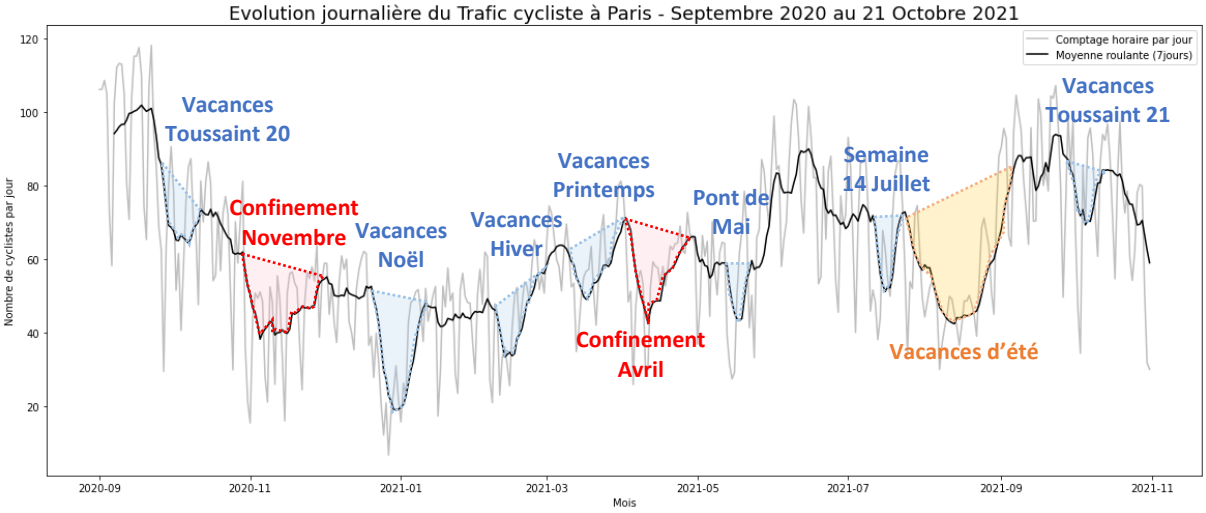
- Septembre, le mois du Vélib à Paris, avec le trafic le plus important de l'année. Septembre 2021 reste le mois à plus fort trafic, bien qu'une baisse d'utilisateurs de -5.6% soit enregistrée. C'est la baisse du trafic en semaine (-9%) qui explique ce recul, on peut supposer que le télétravail en soit responsable.



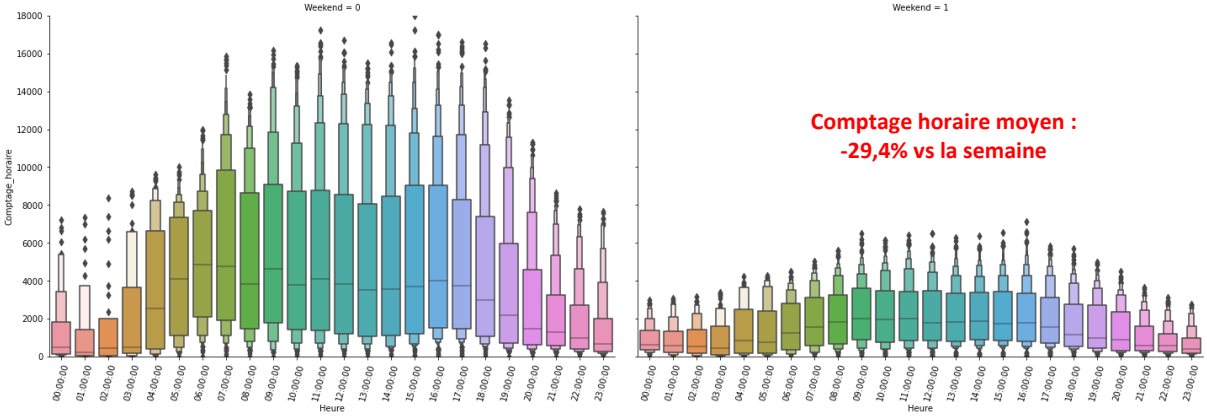
- Grâce à l'ajout de notre variable d'Octobre 2021, cela nous a permis de constater qu'octobre 2021 a connu une augmentation de +11.5% de cyclistes, liée à une hausse d'usage en semaine +14.3%, sans doute dû au retour progressif sur site.
- Une hausse du trafic le week-end est à noter sur septembre et octobre 2021, respectivement de +9% et +8.4%, ce qui laisse présager une progression continue de la pratique cycliste malgré les événements exceptionnels qu'ont connu la fin d'année 2020 et l'année 2021



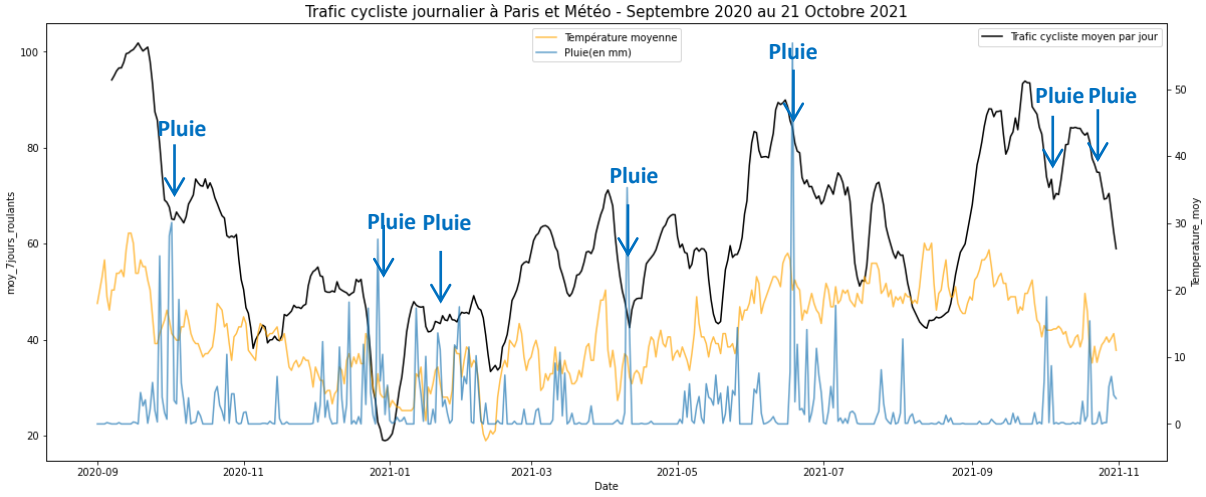
- Un trafic très sensible aux vacances scolaires (-27,4%), ponts et jours fériés (-30,4%),



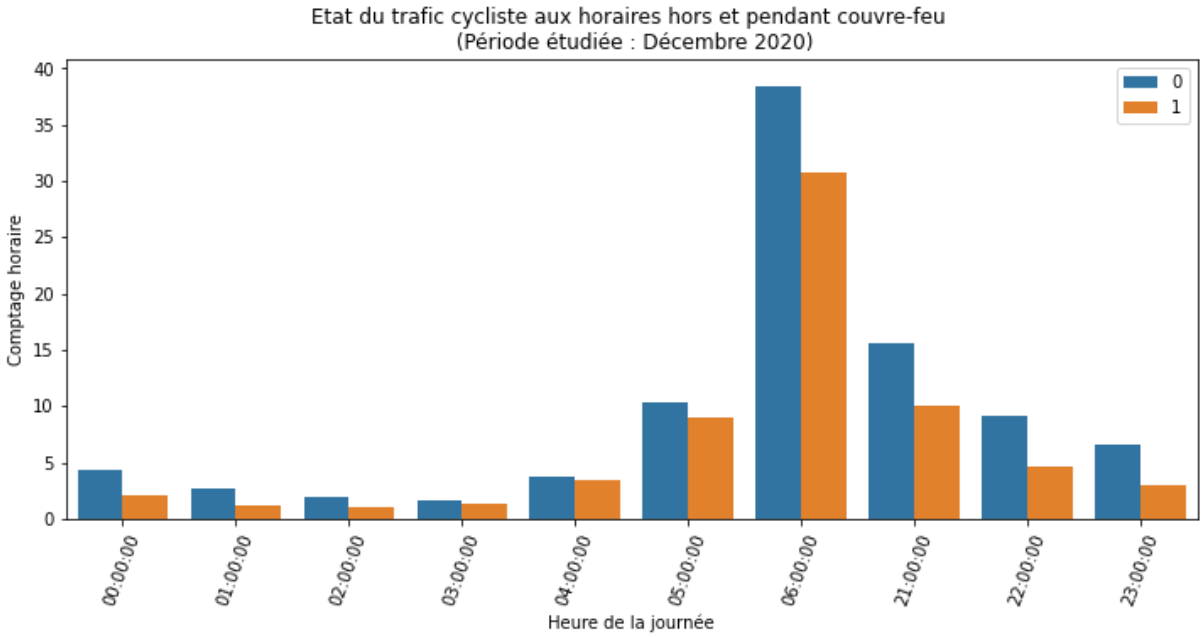
- Une forte disparité de trafic entre la semaine (des sites pouvant comptabilisés jusqu'à 18 000 cyclistes par heure) et le week-end (-29.4%). On estime que pour près de 2/3 des utilisateurs, c'est un moyen de transport utilisé pour se rendre au travail.



- Le vélib est clairement un moyen de transport saisonnier, la courbe de trafic suit globalement celle des températures. Sans surprise, la pratique du vélib est conditionnée par la météo, notamment la pluie. A chaque journée pluvieuse, le trafic connaît une légère baisse.

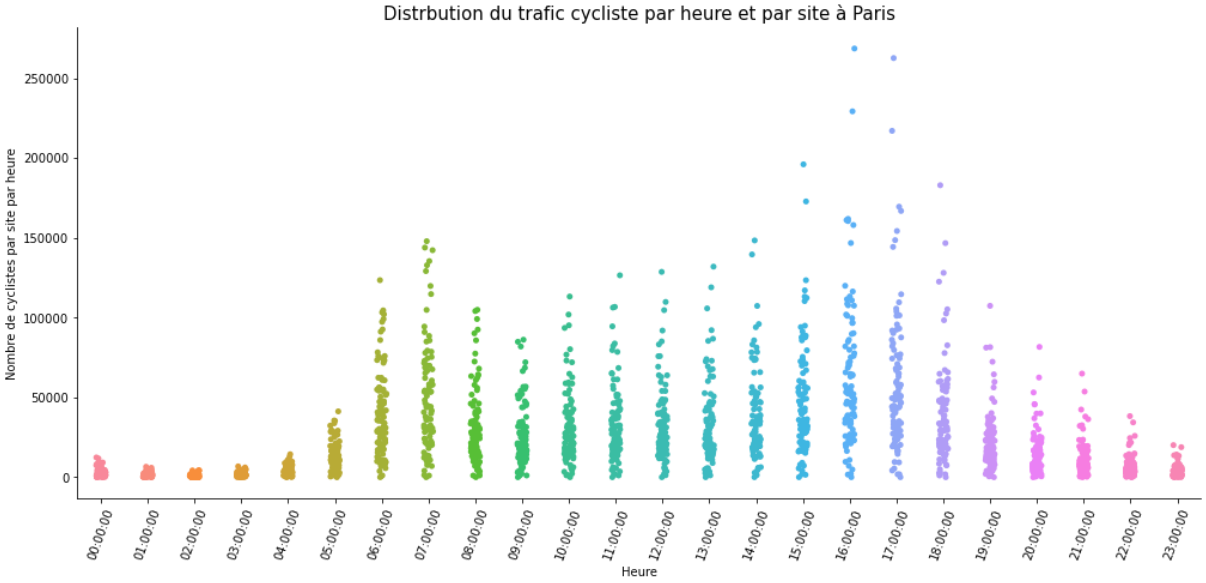


- Des événements exceptionnels sont venus ponctuer l'usage du vélo, comme le couvre-feu (-30% de trafic)

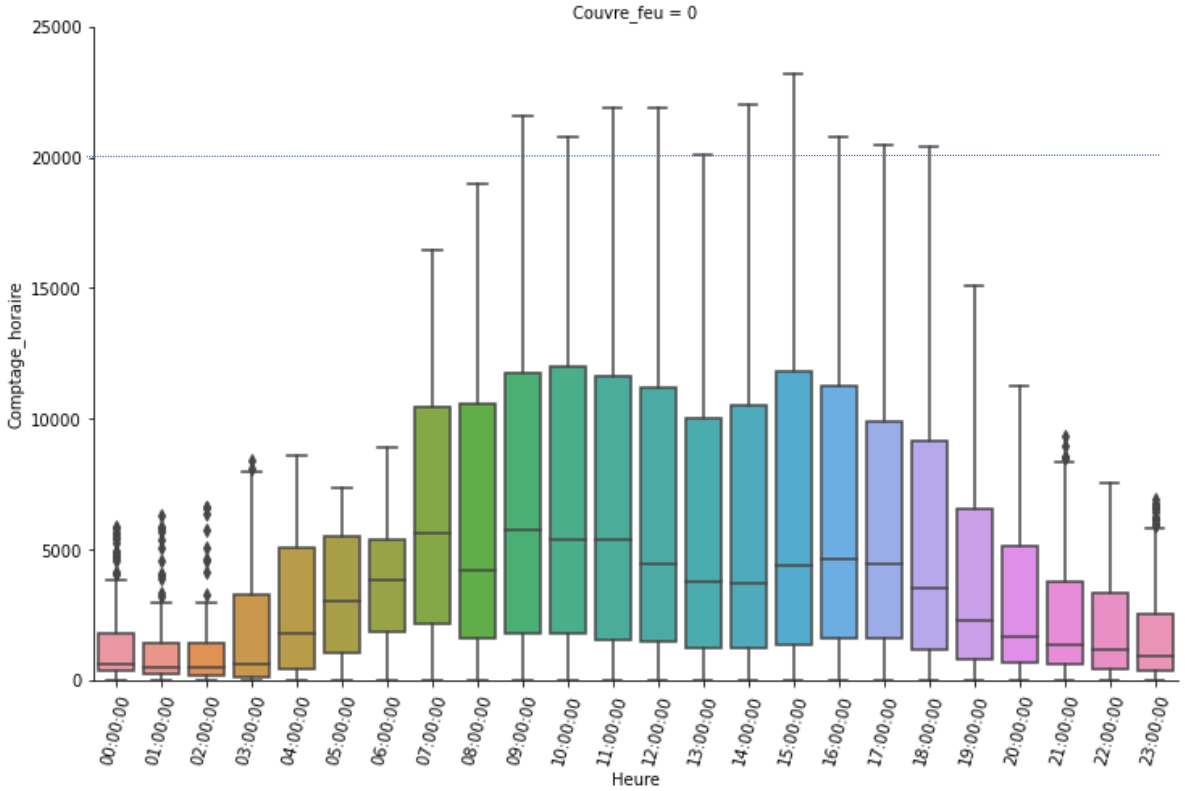


- La crise sanitaire au travers du confinement a eu pour effet, un recul de -19% du trafic.

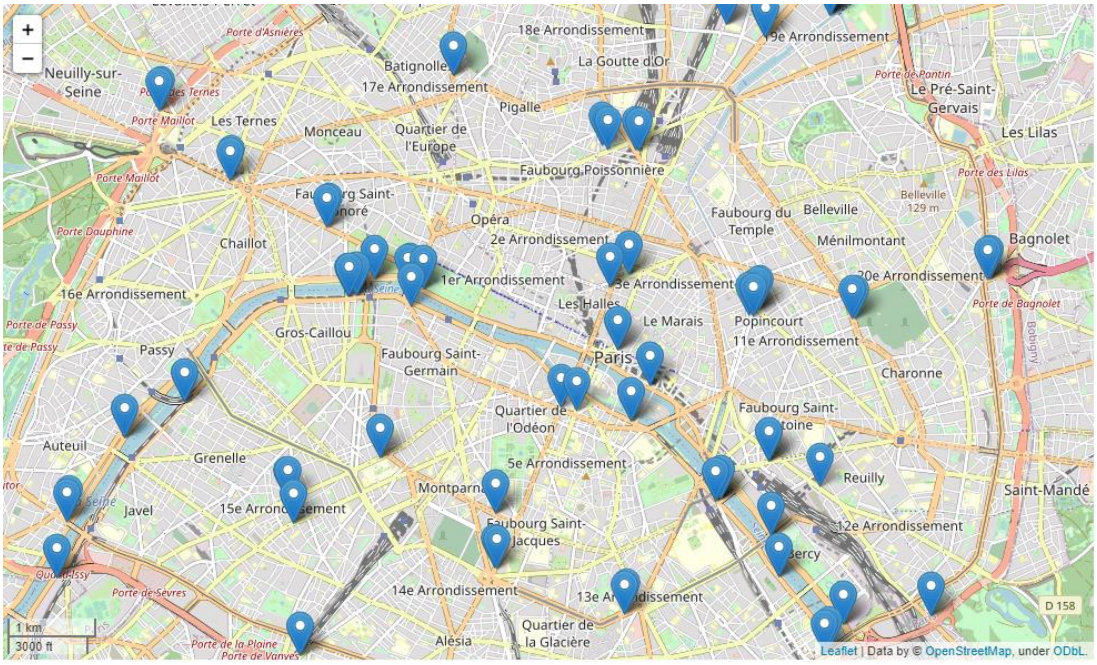
- Lorsque nous observons la distribution du comptage par site et par heure, le trafic est caractérisé par des heures de pointes entre 6h et 7h et entre 16h et 17h, pouvant enregistrer plus de 25 000 cyclistes par heure pour certains sites. Nous nous étonnons de ces horaires que nous qualifions de « tôt » : démarrer sa journée plus tôt afin de rentrer avant le couvre-feu.



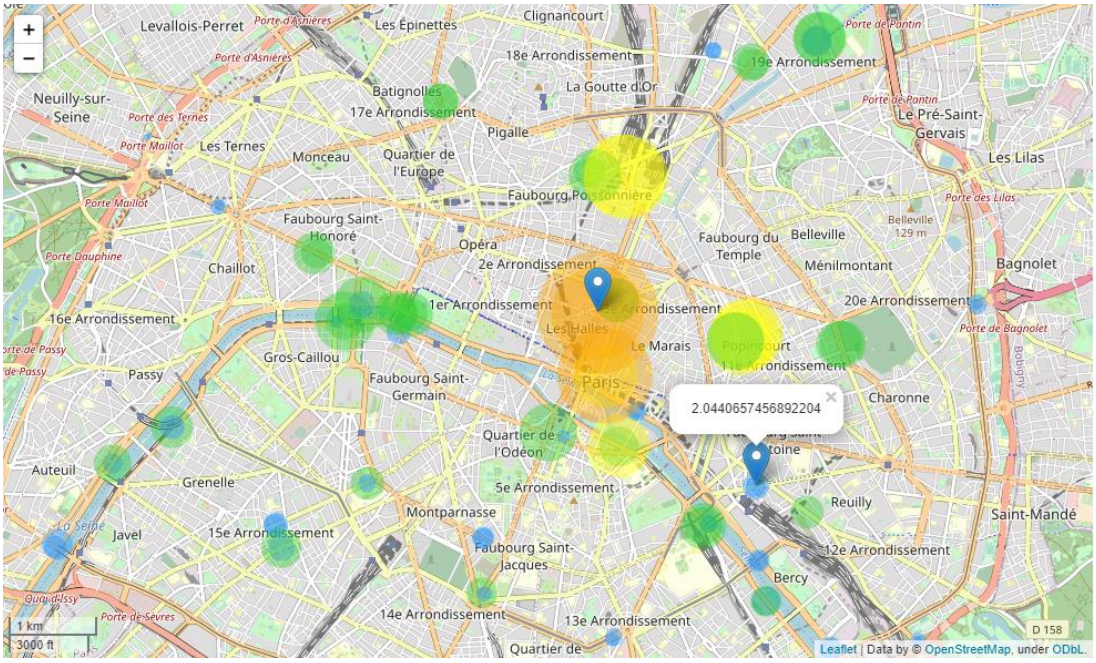
- Cette année 2021 a été marquée par une période de couvre-feux successifs du 15 décembre au 20 Juin, nous souhaitons savoir si ceux-ci ont modifié le comportement du trafic cycliste. Observons la distribution du trafic par heure hors couvre-feu : cette fois-ci, les heures de pointes correspondent davantage à un trafic « habituel », à savoir un démarrage du trafic à partir de 7h et une baisse du flux à partir de 19h.


Couvre_feu = 0
25000
20000
15000
10000
5000
0
Comptage_horaire
00:00:00
01:00:00
02:00:00
03:00:00
04:00:00
05:00:00
06:00:00
07:00:00
08:00:00
09:00:00
10:00:00
11:00:00
12:00:00
13:00:00
14:00:00
15:00:00
16:00:00
17:00:00
18:00:00
19:00:00
20:00:00
21:00:00
22:00:00
23:00:00
Heure

II. Cartographie : l'intensité du trafic par localité



En utilisant le module Folium, nous avons placé sur la carte de Paris les emplacements géographiques des compteurs de passage horaire de vélo présents dans notre dataset.

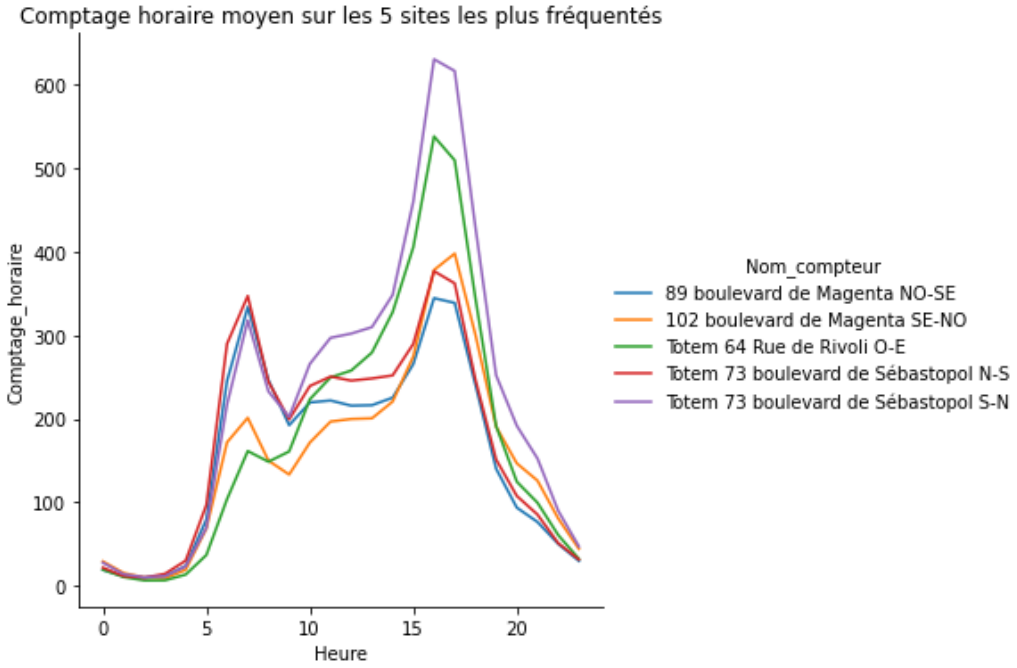


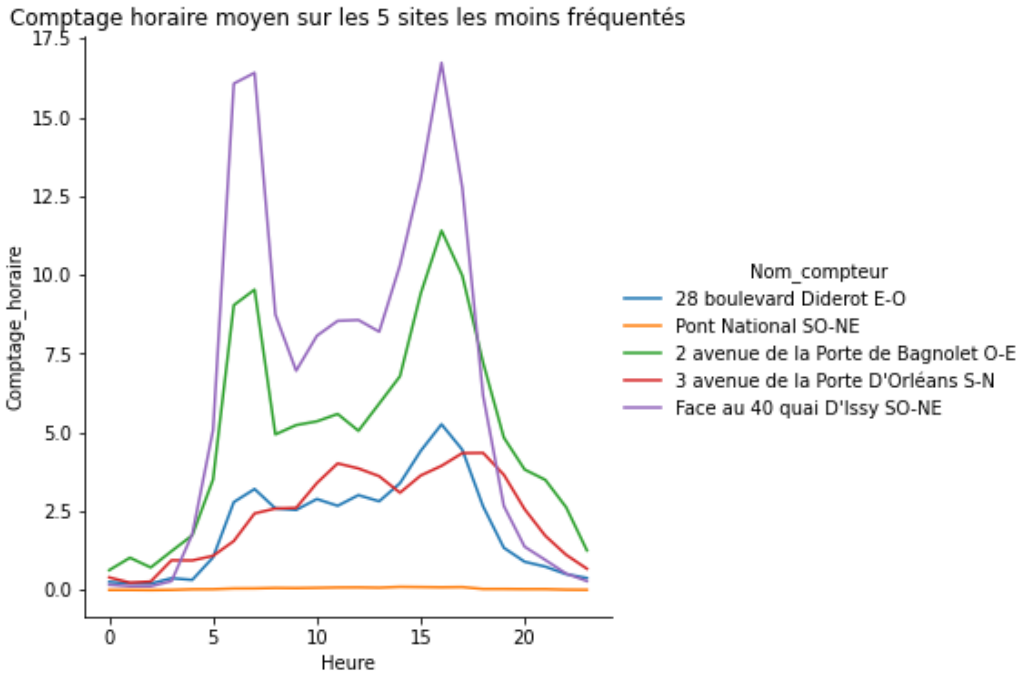
Sur ce 2ème graphique nous observons les comptages moyens sur chaque compteur, avec une couleur et une taille des halos liés à la valeur de ces comptages.

Nous observons logiquement les plus grands passages sur les grands axes parisiens, dans le plein centre aux alentours du quartier du Châtelet, des bords de Seine et des grandes gares.

Le compteur le plus fréquenté se situe au 73 Boulevard de Sébastopol, en plein centre, tandis que le moins fréquenté est au 28 Boulevard Diderot avec en moyenne 2 passages par heure.

Il est aisé de voir les grandes disparités de passages entre les 5 compteurs les plus fréquentés et les 5 compteurs qui le sont le moins sur les 2 graphiques ci-dessous, avec des moyennes journalières en heures de pointe entre 300 et 650 passages contre 2 à 17 passages :





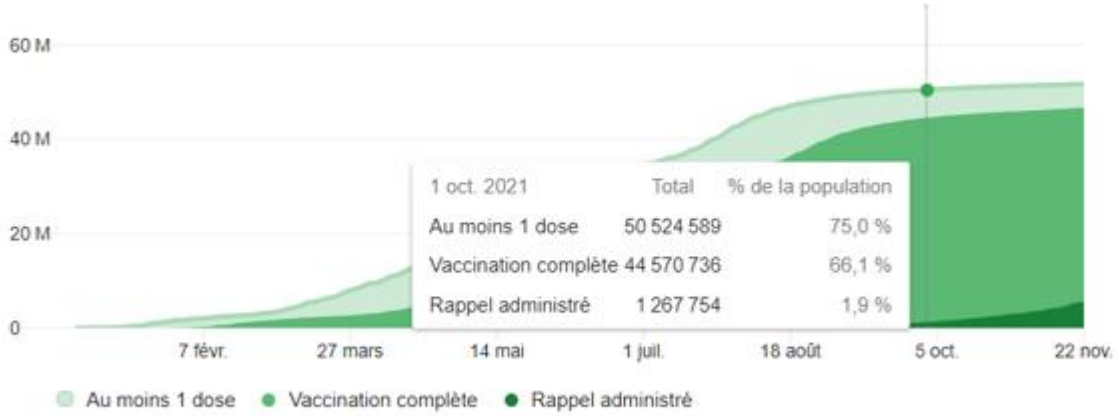
Suite à ces constats et aux nombreuses perturbations du trafic durant la période d'étude, nous décidons d'étudier leur corrélation sur le comptage horaire journalier et d'essayer de prédire ces derniers.

Nous utiliserons pour cela le Machine Learning et sélectionnerons le modèle le plus précis quant aux comptages réels observés et les comptages prédits par les algorithmes

Vaccinations

Source Our World in Data · Dernière mise à jour : il y a 3 jours

Total · France · Toute la période



Ces données indiquent le nombre de personnes ayant reçu au moins une dose de vaccin. Il est possible que les personnes vaccinées aient reçu plusieurs doses. Les vaccins de rappel sont des doses supplémentaires administrées aux personnes complètement vaccinées. · À propos de ces données

III. Machine Learning : la prédiction du trafic grâce aux modèles prédictifs

Introduction

Après avoir identifié certaines corrélations entre nos variables et le comptage horaire par heure, jour et site, nous souhaitons utiliser le Machine Learning pour entraîner un modèle de prédiction de ces comptages.

Notre variable cible est une variable quantitative. Les modèles qui correspondent à notre jeu de données sont donc de type régression linéaire.

Nous utiliserons toutes les variables de notre dataset hormis "Date_heure_comptage" et "Date" qui sont de type datetime.time.

En premier lieu, nous devrons faire un preprocessing.

1. Preprocessing

Nous commençons par créer un nouveau DataFrame qui est essentiellement une copie de notre DataFrame d'origine, sur lequel nous allons travailler.

Afin d'appliquer un modèle Machine Learning, nos variables catégorielles doivent être encodées. Pour ce faire, nous utilisons le "LabelEncoder".

	Annee	Mois_annee	Mois	Jour	Latitude	Longitude	Vac_scolaires	Feries	Confinement	Couvre_feu
147433	2020	0	9	1	48.874716	2.292439	0	0	0	0
147999	2020	0	9	1	48.874716	2.292439	0	0	0	0
148002	2020	0	9	1	48.874716	2.292439	0	0	0	0
147436	2020	0	9	1	48.874716	2.292439	0	0	0	0
148005	2020	0	9	1	48.874716	2.292439	0	0	0	0

Journee_sans_voitures	Weekend	Vac_ete	Pluie_en_mm	Temperature_mini	Temperature_maxi	Temperature_moy	Arrondissement
0	0	0	0.0	15.0	21.0	18.0	14
0	0	0	0.0	15.0	21.0	18.0	14
0	0	0	0.0	15.0	21.0	18.0	14
0	0	0	0.0	15.0	21.0	18.0	14
0	0	0	0.0	15.0	21.0	18.0	14

Ensuite nous divisons notre jeu de donnée en 2 sets, un d'entrainement et un de test. Notre jeu de donnée étant temporel, il n'est pas pertinent d'utiliser un "train_test_split" aléatoire qui ne prendrait pas en compte les tendances de chaque jour, mois,...

Nous faisons donc le split manuellement, en entraînant le modèle sur les 24 premiers jours du mois et en gardant le reste pour le set de test.

Nous définissons ensuite notre variable cible "Comptage_horaire" et séparons nos features de cette dernière.
Afin d'évaluer les performances du modèle, nous utiliserons la métrique RMSE.

La Root Mean Square Error est l'écart type des résidus (erreurs de prédiction). Elle mesure la concentration des données autour de la "line best fit". En d'autres termes, cette mesure nous présente l'erreur de prédiction obtenue à partir du modèle, et décrit les différences entre les valeurs existantes et les valeurs attendues.

2. 1ère Itération

Nous souhaitons tester 4 modèles classiques de régression linéaire :

- Régression linéaire
- Lasso
- Ridge
- ELASTIC NET

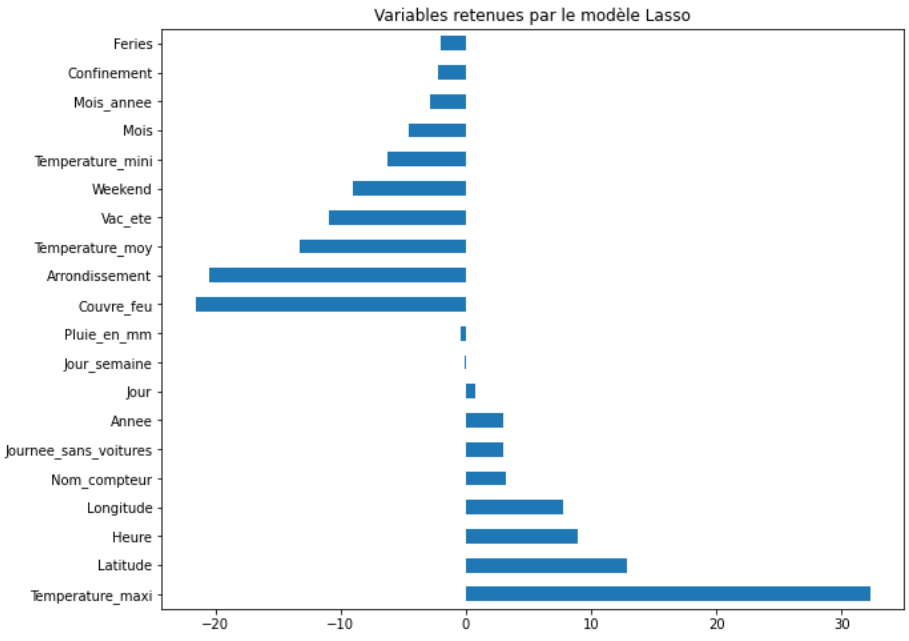
Etant donné la différence de variance entre nos variables, nous avons décidé de les normaliser à l'aide d'un StandardScaler.

Après avoir fait tourner les modèles, nous nous apercevons que les scores obtenus sont faibles. L'indice RMSE est bien trop élevé (environ 76-80) et nous obtenons des prédictions de comptage horaire négatifs, ce qui n'a aucun sens.

Scores :

Modèle	RMSE_Train	RMSE_Test
Ridge CV	80.4923	76.5572
Régression Linéaire	80.4924	76.5601
Lasso CV	80.4963	76.5302
ElasticNET	80.5471	76.6182

Toutes les variables sont bonnes à prendre selon Lasso. A priori car aucune n'est particulièrement corrélée à la target.



Conclusion :

Les modèles doivent donc être optimisés. Nous proposons de donner de la substance à notre modèle avec un effet de "mémoire" en ajoutant les valeurs du comptage des passages à h-1, 2 et 3 afin de l'entraîner plus efficacement.

3. 2ème Itération

Nous créons les 3 variables mentionnées en itération 1 : comptage horaire en h-1, h-2 et h-3.

N'ayant pas de comptages horaires h-1, 2 et 3 au 1er jour de notre jeu de données, nous remplaçons les NaN par des 0. Le nombre de NaN est dérisoire comparé au nombre de lignes de notre dataset. Nous faisons donc l'hypothèse que ces valeurs à 0 n'impacteront pas les résultats.

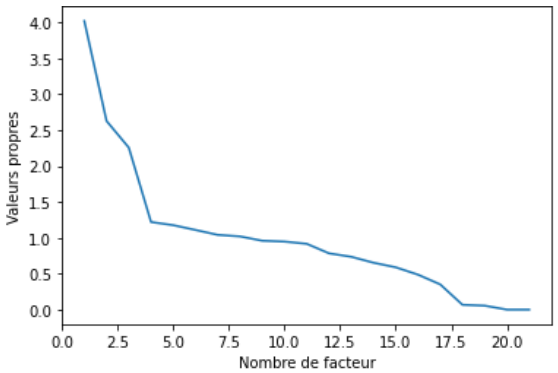
	Annee	Mois_annee	Mois	Jour	Latitude	Longitude	Vac_scolaires	Feries	Confinement	Couvre_feu	...	Weekend	Vac_ete	Pluie_en_mm
147433	2020	0	9	1	48.874716	2.292439	0	0	0	0	...	0	0	0.0
147999	2020	0	9	1	48.874716	2.292439	0	0	0	0	...	0	0	0.0
148002	2020	0	9	1	48.874716	2.292439	0	0	0	0	...	0	0	0.0
147436	2020	0	9	1	48.874716	2.292439	0	0	0	0	...	0	0	0.0
148005	2020	0	9	1	48.874716	2.292439	0	0	0	0	...	0	0	0.0

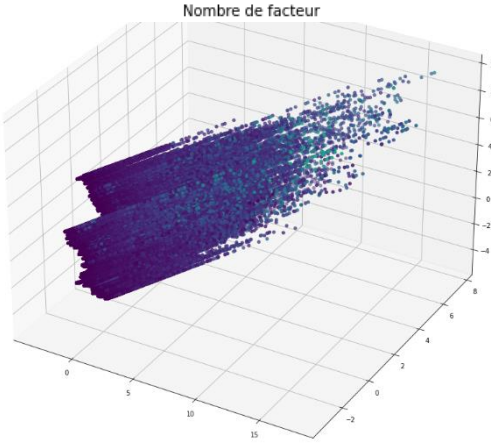
Temperature_mini	Temperature_maxi	Temperature_moy	Arrondissement	Comptage_h1	Comptage_h2	Comptage_h3
15.0	21.0	18.0	14	2.45	3.483333	7.633333
15.0	21.0	18.0	14	2.00	2.450000	3.483333
15.0	21.0	18.0	14	3.00	2.000000	2.450000
15.0	21.0	18.0	14	3.00	3.000000	2.000000
15.0	21.0	18.0	14	14.00	3.000000	3.000000

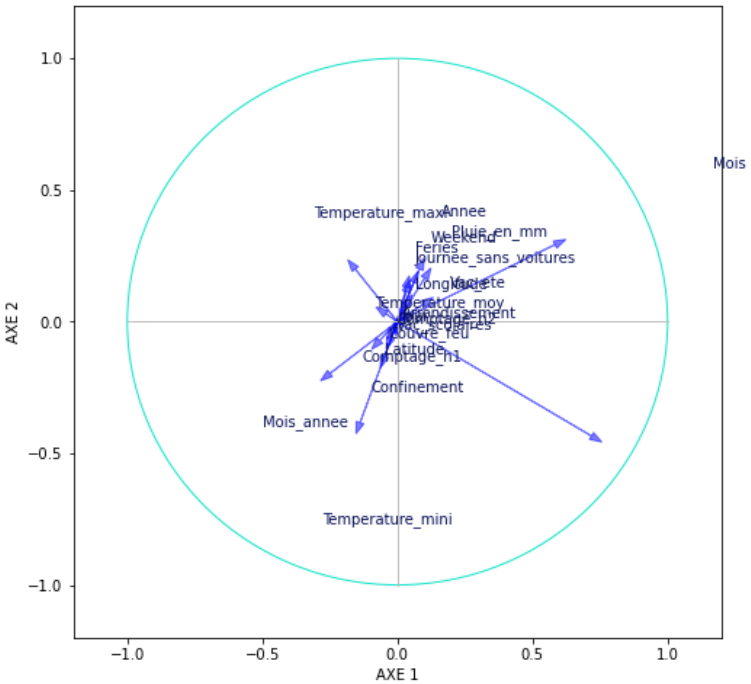
Nous nous sommes également rendus compte que les comptages négatifs prédits en itération 1 venaient du fait que nous n'avions pas normalisé les sets de notre variable cible. Nous avons donc corrigé le problème en utilisant notre StandardScaler sur ces derniers.

A ce stade, nous avons commencé à examiner s'il était nécessaire de réduire le nombre de variables explicatives.

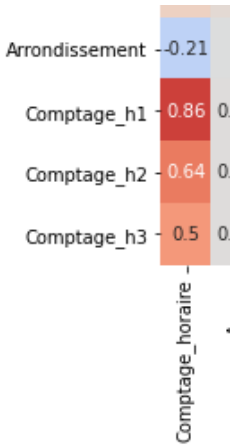
Nous avons donc créé l'ACP mais les résultats n'étaient pas concluants pour l'analyse : nous avons 4 voire 5 composantes principales identifiées avec la méthode du coude. Nous avons représenté nos variables sur un cercle de corrélation à 2 composantes mais les résultats ne sont pas exploitables, ou du moins nous ne savions pas comment faire malgré nos recherches.







Nous avons donc observé la corrélation entre les variables et notre cible dans l'ensemble de données avec une heatmap : nos nouvelles variables de comptage sont effectivement corrélées, contrairement à toutes les autres variables.



Pour cette raison, nous avons pensé qu'utiliser le SelectKBest nous permettrait de mettre en évidence les effets d'une sélection réduite de variables explicatives sur la performance du modèle : nous avons itéré nos modèles avec différentes valeurs de k, et avons constaté que nous obtenions les meilleurs résultats lorsque k = 10. Nos premières impressions suite à l'étude de la heatmap se sont vu confirmées.

En supplément de cela, nous avons testé la division des sets de test et d'entrainement automatique : nous avons utilisé la fonction "train_test_split" et un test_size = 0.2 avec l'argument shuffle = False pour conserver la temporalité de notre jeu de données.

Les résultats se sont avérés décevants, nous sommes donc revenus au split manuel.

Nous avons ensuite entraîné 3 nouveaux modèles :

- DecisionTreeRegressor
- RandomForestRegressor
- GradientBoostingRegressor

Conclusion :

Les résultats n'étaient pas particulièrement concluants. Par ailleurs nous avons involontairement créé des problèmes de fuite de données dans notre jeu de données en entraînant, ajustant et transformant plusieurs fois nos données sur chaque modèle. Le code n'était pas optimisé et les résultats non plus.

Nous décidons donc d'utiliser des pipelines pour optimiser notre code en 3ème itération. Nous pensons également ajouter d'autres variables pour faire jouer l'effet de mémoire des modèles.

4. 3ème Itération

Nous créons les variables comptage horaire j-1, 2 et 3 ainsi que s-1, 2 et 3. Ayant cette fois un nombre de NaN conséquent, nous utilisons un fillna pour remplacer ces NaN par les valeurs moyennes respectives/compteur/jour/heure.

Après ajout des nouvelles variables nous avons également réitéré des tests de SelectKBest et nous passons de k=10 à k=12 pour les meilleurs de nos modèles Ridge, Lasso et ElasticNet.

Nous utilisons cette fois l'objet PIPELINE. Le pipeline Machine Learning est un moyen d'automatiser le flux de travail d'apprentissage en permettant aux données d'être transformées et corrélées en un modèle qui peut ensuite être analysé pour obtenir des résultats.

Ces pipelines rendent le processus de saisie des données dans le modèle Machine Learning clair et facilement reproductible sur les différents modèles.

Nous intégrons dans ces pipelines nos modèles, avec le sélecteur de variables SelectKBest, le scaler StandardScaler, ainsi que les hyperparamètres à tester à l'aide d'une grille de recherche et d'une validation croisée. Cette méthode est bien plus efficace puisque les modèles itèrent sur les hyperparamètres et choisissent la meilleure combinaison. Attention cependant aux temps de chargement qui évoluent proportionnellement aux nombres de paramètres testés : nous choisissons de tester nos hyperparamètres 2 par 2 ou 3 par 3 maximum pour cette raison et nous implémentons la magic line %%time pour récupérer le wall-time de nos modèles.

Nous utilisons les pipelines sur nos modèles regressor et sur la régression, qui nous retournent les meilleurs hyperparamètres, la meilleure moyenne de RMSE en validation croisée ainsi que le RMSE de notre jeu de test.

Nous décidons d'essayer un AdaBoosting sur le DecisionTreeRegressor afin que les mauvaises prédictions soient systématiquement retravaillées à chaque niveau de branche. Cependant cela n'a pas changé notablement les résultats, nous en concluons donc que cet essai n'était pas pertinent.

Nous décidons également de remplacer le Gradient Boosting Regressor par le XGBoost Regressor qui possède plus d'hyperparamètres et qui peut donc être affiné plus facilement. Attention au réglage de ces hyperparamètres qui augmente considérablement le temps de modélisation. Nous nous sommes contentés de tenter des combinaisons de 3 hyperparamètres, pour des rendus d'environ 2h. Nous aurions certainement pu améliorer les scores mais un test de rendu avec 4 hyperparamètres nous a donné une amélioration d'environ 8% pour près de 8h de rendu. Ratio temps/amélioration que nous n'avons pas jugé admissible.

Pour les autres modèles, nous considérons que la validation croisée qui sélectionne le meilleur alpha est suffisante.

Les pipelines recommandent systématiquement de ne pas normaliser les données. Nous décidons donc d'abandonner notre StandardScaler pour nos modèles simples (nous faisons l'hypothèse que cela vient du fait que la différence max de variance de 80-90 entre nos variables identifiée avec df.describe() n'est pas contraignante sur un dataset de 700K+ lignes).

Nous obtenons nos meilleurs résultats, avec des RMSE aux alentours de 21-27 (ce qui est très loin des 79-80 de la 1ère itération).

	Modele	RMSE_train	RMSE_test	cross_validation_RMSE	wall_time_in_s
0	XGB	12.860	21.834	17.920	7010.00
1	RandomFR	17.177	22.954	20.355	7142.00
2	DecisionTR	20.801	24.011	23.220	7716.00
3	REGRESSION LINEAIRE	27.627	26.621	26.753	183.00
4	RIDGE	27.917	27.014	27.935	6.25
5	LASSO	27.953	27.059	27.969	21.70
6	ELASTIC NET	28.018	27.180	28.039	85.00

5. Etude du Meilleur Modèle

```
xgb_reg_optimized = xgb.XGBRegressor(
                    colsample_bytree=0.4,
                    eta= 0.2,
                    max_depth =9,
                    n_estimators =150,
                    n_jobs= -1,
                    subsample= 0.9,
                    )
```

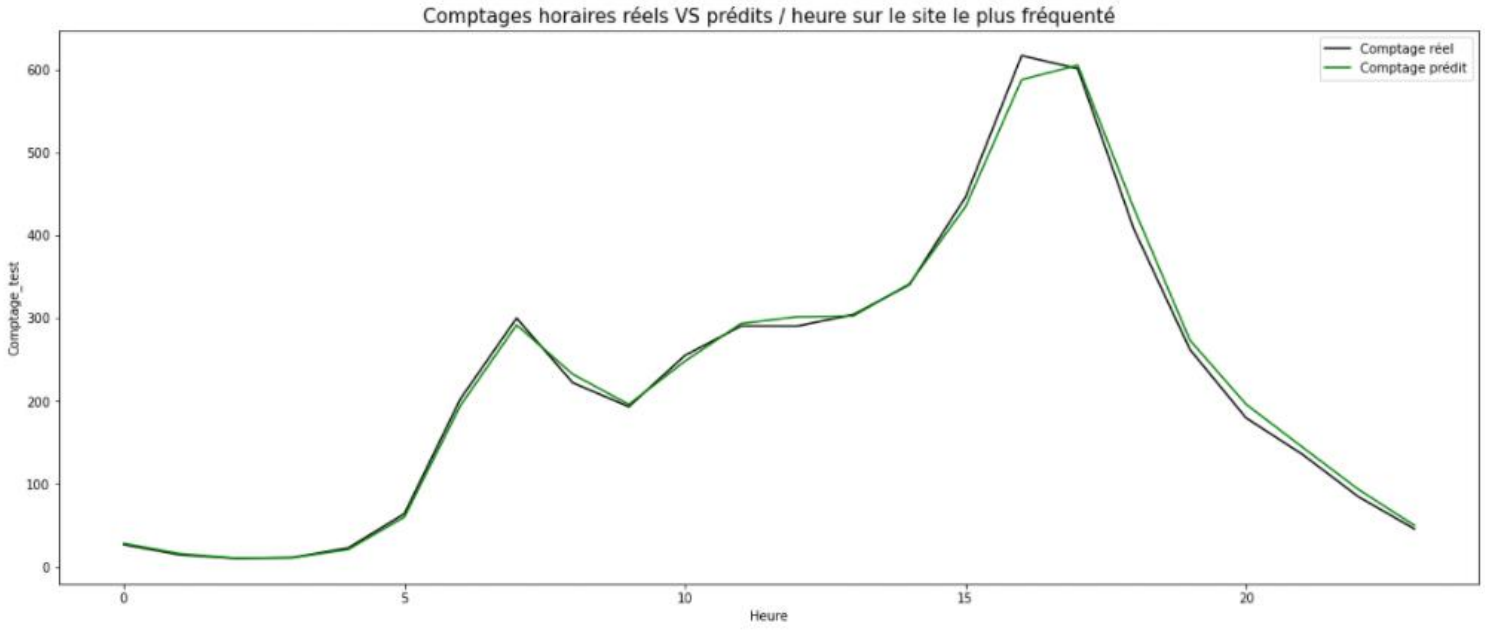
Après lancement de nos pipelines, le modèle XGBoost nous fournit les meilleurs résultats (RMSE = 21.834). Les autres modèles Regressor n'étant pas loin derrière et nos modèles simples non plus.

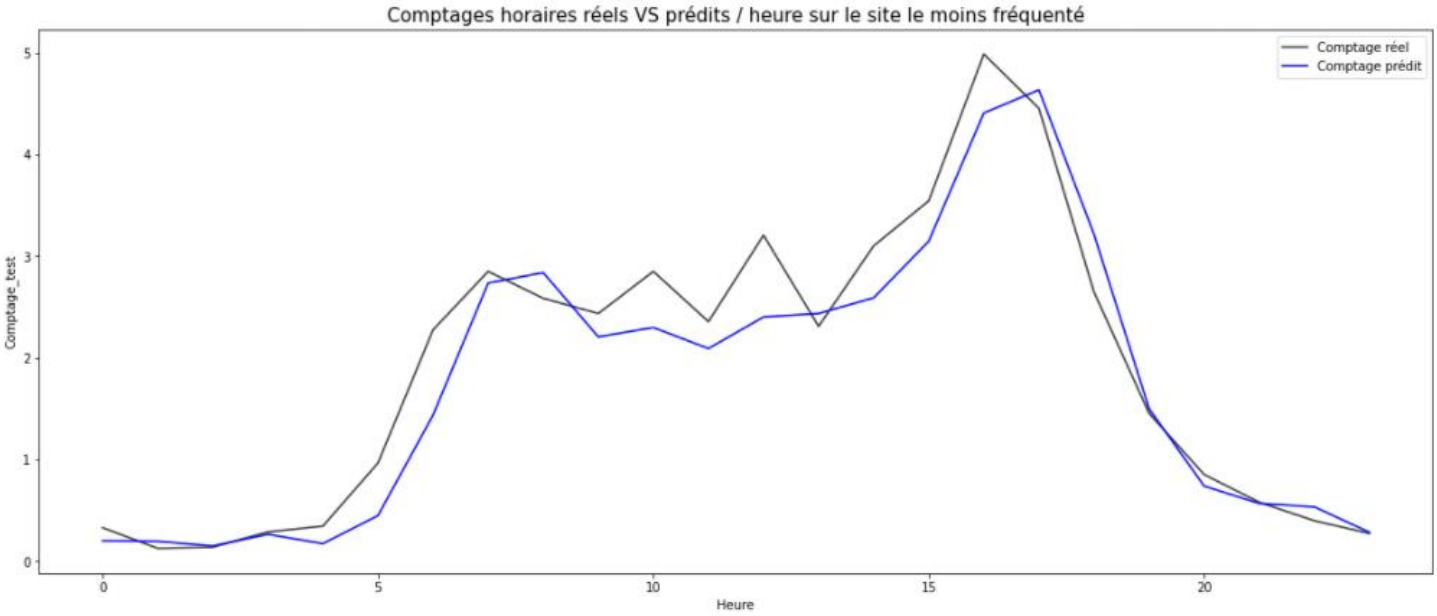
XGBoost est un modèle, basé sur des arbres de décision qui démontre la supériorité sur l'apprentissage automatique profond.

Cette direction et ce modèle nous ont fourni la plus petite erreur de prédiction pour le test et le groupe d'entrainement

De plus, l'indice R2 qui mesure la force de la relation entre le modèle et la variable dépendante qui lui est donnée : plus le R2 est élevé, plus le modèle est adapté pour prédire les observations. Dans notre modèle nous obtenons R2 = 0,933, ce qui signifie de bonnes prédictions.

Afin de confirmer notre sélection de modèle, nous avons créé un graphique montrant le comptage prédit en moyenne sur le site avec la fréquence de passage la plus élevée (Totem 73 boulevard de Sébastopol S-N). Comme nous l'avions prévu, les prévisions moyennes sont très proches des données réelles.

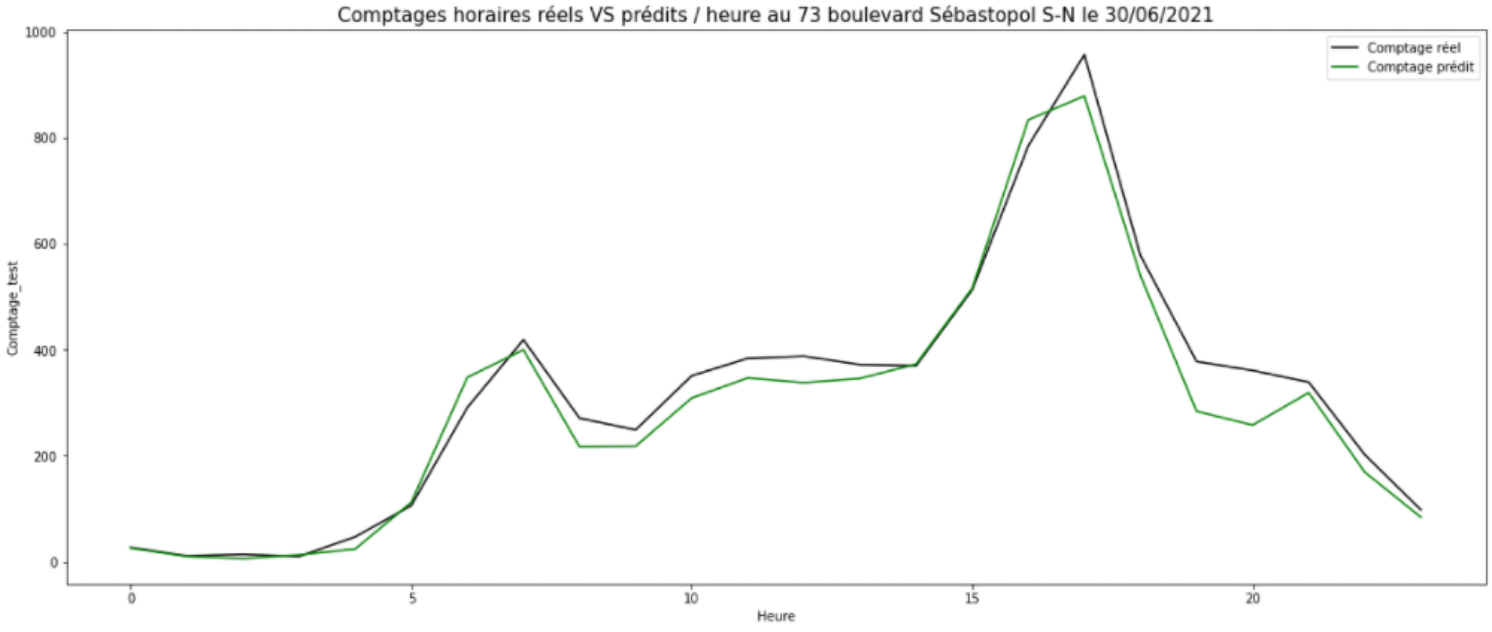




Sur le site avec l'une des fréquences de passage la plus faible (28 boulevard Diderot E-O), nous avons constaté que la prédiction est bien moins précise.

Nous avons également voulu analyser la prédiction sur une journée au hasard sur le site le plus fréquenté, afin d'obtenir une vision plus précise qu'un comptage horaire moyen sur toute la période présente sur le dataset de test.

Nous avons donc choisi une date présente dans ce dernier, à savoir le 30/06/21, de manière complètement aléatoire.



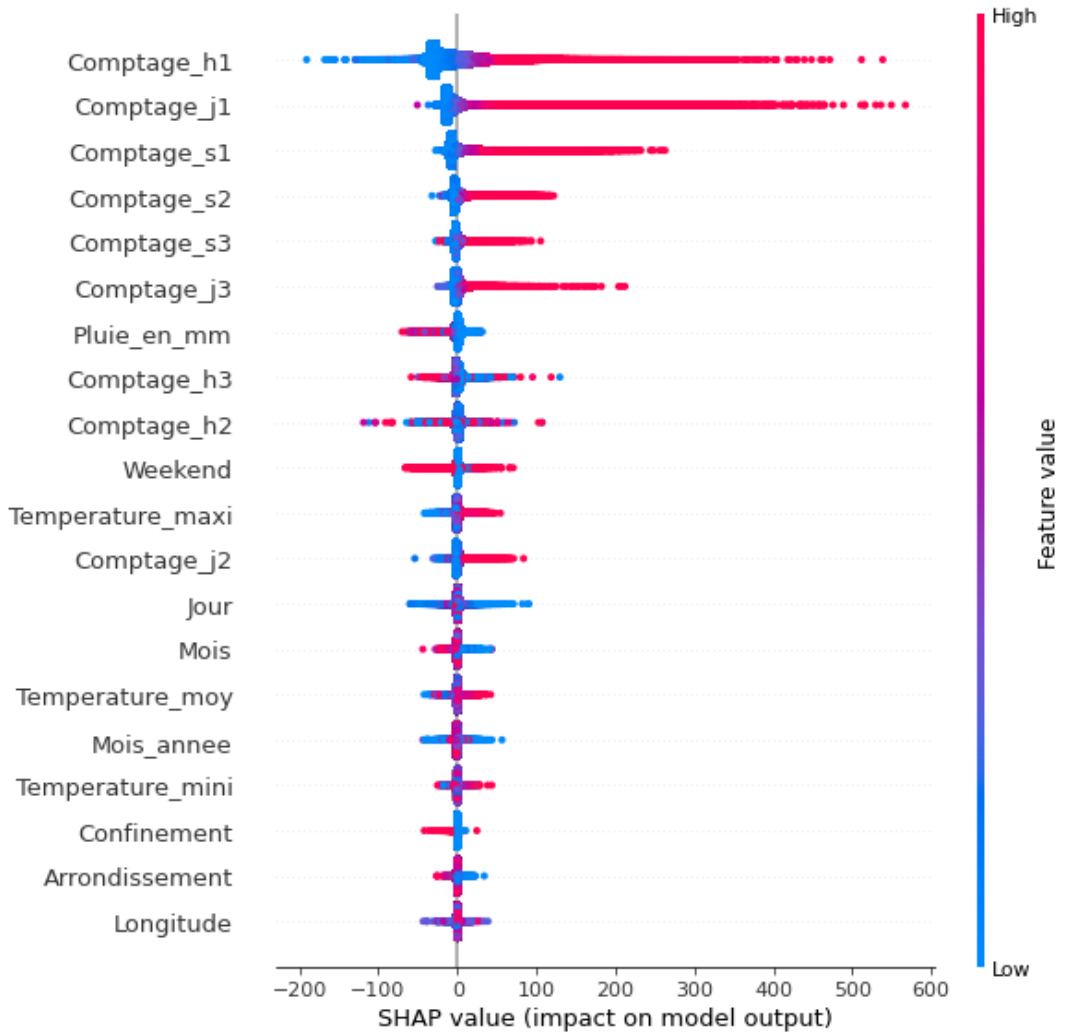
Nous observons que notre modèle XGB est fiable en période creuse. En dehors de cela, nous retrouvons des écarts qui fluctuent selon les heures de la journée et qui semblent osciller entre notre fenêtre de RMSE = 22 et une 50aine de passages.

Nous faisons l'hypothèse que les grandes variations de comptage par heure et par jour en période pleine donnent du fil à retordre au modèle.

Nous en concluons que si le modèle est en moyenne performant, il y a des disparités selon le jour, l'heure mais également la fréquentation générale du compteur qui perturbent la qualité de prédiction.

Pour comprendre cela, nous avons utilisé SHAP, qui permet d'identifier les variables qui participent le plus à l'apprentissage du modèle.

Avec l'aide de SHAP nous essaierons d'expliquer les résultats obtenus et d'utiliser l'outil, afin de localiser les caractéristiques qui ont le plus contribué à la classification et comment elles ont contribué :

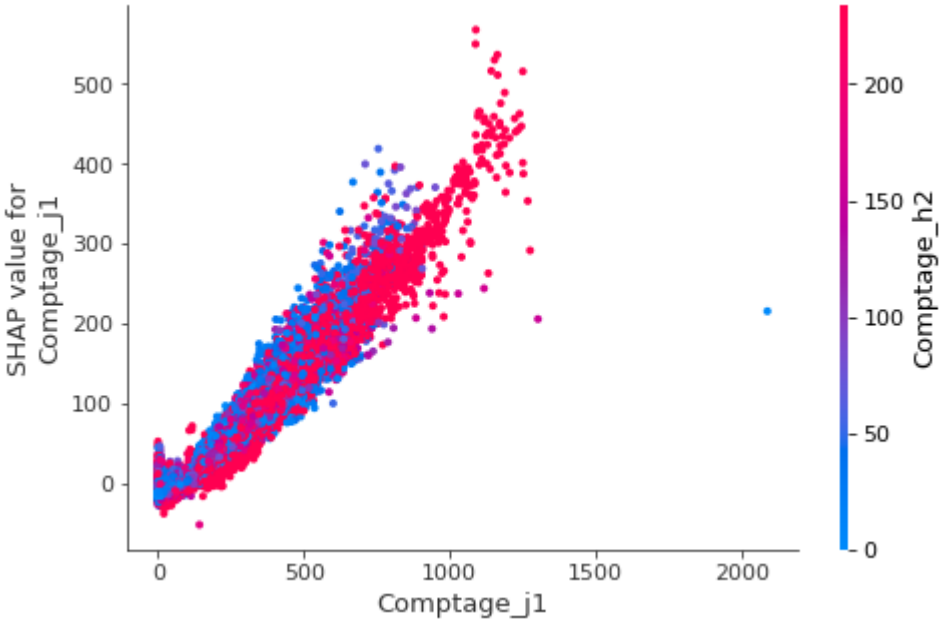


Sur ce premier graphe :

- La position sur l'axe des abscisses indique une prédiction plus petite (à gauche) et plus grande (à droite) que la variable cible réelle.

- La couleur rouge signifie que la valeur prédite concerne une valeur cible réelle (un comptage horaire réel) élevée avec un impact positif sur la prédiction, la couleur bleue signifie que la valeur prédite concerne une valeur cible réelle (un comptage horaire réel) faible avec un impact négatif sur la prédiction.

Si l'on observe les graphes de comptage prédit et réel on observe ce phénomène : les gros comptages horaires sur le site le plus fréquenté affectent positivement la prédiction, tandis que les petits comptages sur l'autre site affectent négativement cette dernière.



Sur ce 2ème graphe :

- On comprend qu'il y a une relation linéaire positive entre la variable comptage_j1 et notre comptage horaire, et que la variable comptage j1 interagit le plus avec la variable comptage h2 que les hautes et faibles valeurs impactent positivement et faiblement

Conclusion et perspectives

Suite à l'analyse sur le trafic au travers de la data visualisation, l'année 2021 a été extrêmement perturbée par le confirnament et les couvre-feux successifs, le comparatif des mois de Septembre et octobre 2021 avec 2020 nous montrent une croissance sur le trafic semaine et week-end. Il faudrait que nous puissions récupérer d'autres données afin d'élargir cette analyse comparative.