



Auteurs :

**Gaëtan LE FLOCH**

**Mehdi FERHAT**

**Alexis VIGNARD**

# Etude et modélisation des retards de TGV entre 2015 et 2020

## Manuel d'utilisation du code

Afin de pouvoir exécuter le code de manière optimale, il faut veiller à avoir tous les fichiers **dans le même dossier** (que ça soit le fichier main ou les modules). Les utilisateurs sous iOS doivent faire attention au fait que les modules ***SNCFToolbox*** et ***OutilsStatistiques*** font tous deux appel au package OS, et les chemins spécifiés dans les fonctions concernées contiennent des double slash (puisque le codage a été réalisé sous Windows). S'il y a un souci lors d'un export, n'hésitez pas à modifier ces double-slash (les fonctions concernées sont spécifiées à la fin de chaque module). Enfin, il est important de noter que ces exports se trouveront dans un dossier "*Exports*" qui sera créé là où les fichiers se situeront.

# Sommaire

<b>Sommaire.....</b>	<b>3</b>
<b>Présentation.....</b>	<b>5</b>
I-Partie 1, Recherche de données et création du premier module	<b>7</b>
a) Sélection et visualisation de la base.....	<b>7</b>
b) Le module SNCFToolbox.....	<b>9</b>
c) Exécution du module et observations.....	<b>10</b>
II- Partie 2, Premières analyses, création de nouveaux modules et préparation pour le modèle.....	<b>10</b>
a) Etude de la base, créations des graphiques.....	<b>11</b>
1. Etude temporelle.....	<b>12</b>
2. Données coupe transversale.....	<b>13</b>
b) Module OutilsStatistiques.....	<b>16</b>
III- Partie 3, Modèle et résultats.....	<b>17</b>
a) Régression : étape préliminaire.....	<b>17</b>
b) Choix des variables explicatives.....	<b>18</b>
c) Création et exécution du modèle.....	<b>19</b>
<b>Témoignages, commentaires d'expérience.....</b>	<b>24</b>





# Présentation

Depuis sa création en 1938, la SNCF est un acteur majeur de la société et de nos vies quotidiennes. Elle permet grâce à ses milliers de voies de chemins de fers et trains de desservir les individus entre leurs domiciles, lieux de travail et toutes autres destinations.

En 2019, on comptait près de 73 % des français utilisant les transports en commun pour leurs déplacements quotidiens, contre 63 % en 2014. Cette forte hausse témoignant d'un succès inhérent au coût mais également à la simplicité de déplacement dans des agglomérations submergées par le trafic routier et autoroutier notamment en Ile-de-France font de la SCNF, aujourd'hui, un maillon structurant des mouvements humains.

De ce fait, l'efficacité des réseaux de transport est un objectif majeur tant la moindre défaillance peut paralyser les circuits essentiels, plus particulièrement ceux des individus se rendant sur leurs lieux de travail, impactant de manière quasi-directe l'économie du pays. Les grèves survenues à l'hiver 2019-2020 sont un exemple concret des conséquences d'un fonctionnement en dent-de-scies des trains et métros.

C'est ainsi que nous avons décidé de consacrer notre projet **à l'analyse des dysfonctionnements des réseaux de transports**. Séduits par l'application d'un algorithme de prédiction sur les crimes aux Etats-Unis et en cohérence avec notre champ d'études, nous avons tenté de créer un modèle le plus solide possible pour expliquer les raisons des différences de temps de trajet entre plusieurs itinéraires.

## *Bibliothèques utilisées :*

- Seaborn
- Pandas
- Numpy
- Matplotlib
- SciPy
- Statmodels

# I – Partie 1, Recherche de données et création du premier module

## a) Sélection et visualisation de la base



Grâce aux jeux de données disponible en libre-service sur le site de la SNCF (<https://ressources.data.sncf.com>) nous avons parcouru une large liste de bases à la recherche d'un fichier correspondant à nos attentes.



### Régularité mensuelle TGV par liaisons

Découvrez la régularité mensuelle TGV par liaisons (AQST).

Régularité

TGV

Retards

Horaires

Données 7 560 éléments  
Producteur SNCF Voyage

Nous nous sommes ainsi tournés vers un jeu de données modélisant les régularités mensuelles des TGV.

Nous avons à l'origine 3 bases de disponible mais à un nombre d'observations  $N < 500$ , respectivement 64 et 373 rows.

Un nombre que nous estimions insuffisant d'autant plus que l'on s'attendait logiquement dans cet exercice à un nettoyage de la base pour d'éventuelles données manquantes.

On dispose du fichier .csv sélectionné ci-dessus et on l’affiche pour voir ce qu’il en est et procéder aux éventuels correctifs.

```
In [2]: dataSNCF = pd.read_csv(".\\regularite-mensuelle-tgv-aqst.csv", sep=";")
...: print(len(dataSNCF.columns))
34
```

Nous disposons ici de 34 variables d’une liste non exhaustive que l’on affichera par la fonction « **print(data.SNCF.columns)** » tel que le « Retard moyen des trains au départ (min) », le « Nombre de trains en retard au départ/arrivée » mais également le pourcentage de trains en retard pour x raisons.

Assez vite on s’aperçoit que de nombreuses variables ne peuvent être utilisées en tant que telles :

- Parce qu’elles sont présentes sous la forme de chaîne de caractères (par exemple la Gare de départ/arrivée).
- Parce que la colonne est tout simplement vide, de manière entière ou partielle.

Si le premier point exige des modifications pour certaines variables qui peuvent-être interprétées graphiquement/mathématiquement. D’autres vont nécessairement devoir être éjectées de notre étude. En effet, la gare de départ/arrivée bien que présente sous forme de chaîne de caractère peut parfaitement être interprétée comme une variable qualitative à condition d’effectuer une transformation. On optera pour la création d’indicateurs, sera assigné alors un nombre pour chaque gare. L’objectif ici est donc de transformer des variables d’un type **str** à un type **int**.

Pour ce qui est des commentaires, ces derniers seront tout simplement supprimés. Nous faisons également en sorte que les observations ne disposant pas de valeurs pour les variables non éjectées soient supprimées dans un souci d’uniformité.

\*) Méthodologie des retards

La notion de retard est relative, c’est pour cela qu’il convient d’y apporter une définition stricte.

Dans cette étude, il sera considéré qu’un train est en retard si :



- Le retard à l'arrivée au terminus est supérieur à 5 minutes pour une correspondance d'une durée inférieure à 1h30
- Le retard à l'arrivée au terminus est supérieur à 10 minutes pour une correspondance d'une durée comprise entre 1h30 et 3h
- Le retard à l'arrivée au terminus est supérieur à 15 minutes pour une correspondance d'une durée supérieure à 3h.

Ce sont de ces besoins de nettoyage de données (ou Data Cleansing) que nous créons notre premier module : ***SNCFToolbox.py***

## b) Le module SNCFToolbox

Ce module qui a pour but d'effectuer la première partie du travail d'analyse de données est fondamental tant il va définir la fiabilité de la suite de notre étude.

Celui-ci est composé de 4 fonctions :

- La fonction **CreateIndicator()** qui prend en argument le dataframe, la colonne d'intérêt ainsi que l'indicatrice qui garde ou retire la colonne. Elle permet donc de transformer une colonne texte en une colonne numérique : Chaque colonne d'intérêt aura un identifiant numérique, et sera manipulable à souhait (contrairement aux colonnes textes).
- La fonction **CreateDummy()** qui prend en argument le dataframe, la variable d'intérêt, son nouveau nom, ses modalités, ainsi que la valeur de l'indicatrice. Elle permet donc de créer une indicatrice (et par définition attribuera soit la valeur 0, soit la valeur 1).
- La fonction **ToLog()** applique une transformation logarithmique à la colonne sélectionnée dans le dataframe. Elle nous est utile pour le modèle de régression mis en place plus tard.

- La fonction **ExportDB()** permet d'exporter le dataframe sélectionné en tant que fichier CSV. Ce fichier est situé dans un dossier nommé Exports au même emplacement que le fichier main.py.

## c) Exécution du module et observation

Il est maintenant temps de mettre en œuvre notre premier module.

- La fonction **CreateIndicator()** est utilisée pour transformer les colonnes des gares de départs et d'arrivée en variables en variables numériques : il s'agit d'identifiants..
- La fonction **CreateDummy()**, quant à elle, permet d'ajouter une indicatrice renvoyant si le trajet est national ou international.

On purge les données inutiles grâce à « **pandas.DataFrame.drop** » (comme spécifié précédemment, les données de type **str** qui n'ont aucune interprétation économique mais également celles qui ont été transformées en **float**, n'étant plus utiles, nous nous concentrerons sur les nouvelles variables portant les mentions **revised**).

Nous nous retrouvons ainsi avec une dataframe avec le même nombre observations mais plus que 32 variables. On consulte les statistiques descriptives pour se faire une idée de la pertinence du jeu de données une fois nettoyé (procédure « **.describe()** »).

## II – Partie 2, premières analyses, création de nouveaux modules et préparation pour le modèle

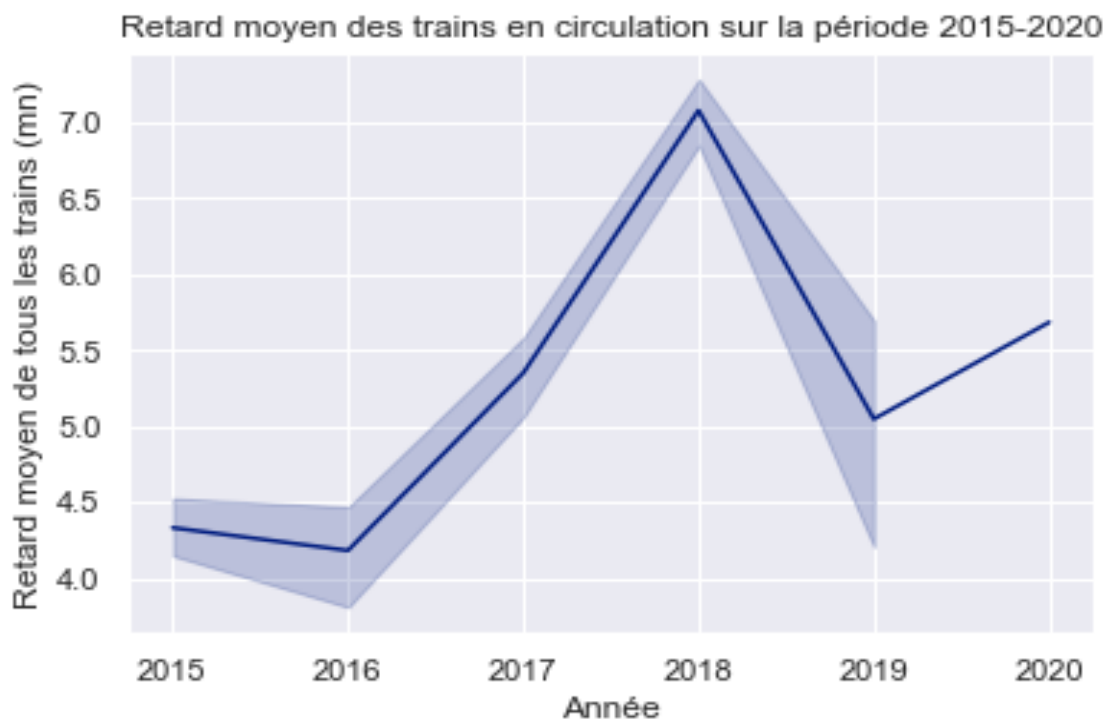
## a) Etude de la base, création de graphiques

\*par souci de clarté, l'entièreté du code des graphiques a été réalisée à l'intérieur du module *Graphes\_f*, ces derniers seront appelés par la fonction **Graphiques()**

La base de données nous renseigne sur la moyenne de retard des trains en retard, mais également sur celui de tous les trains (comprenant ainsi ceux à l'heure, et en avance).

Il nous a semblé plus pertinent de nous concentrer sur les retards totaux, car les retards des trains déjà en retard ne décrivent pas la réalité des retards totaux, mais uniquement celle des trains déjà en retard.

De plus, nous choisissons de regarder uniquement les retards à l'arrivée, car un retard au départ peut se corriger au fil du trajet.



Le spectre bleu correspond à un intervalle de confiance fixée au seuil de confiance 95%. On remarque que l'intervalle n'est pas présent sur la dernière année car cette dernière est toujours en cours.

Nous constatons un pic des retards en début de 2018, allant jusqu'à 7 minutes. Ce retard moyen peut sembler énorme, cependant il faut garder à l'esprit qu'un train n'arrive presque jamais en avance afin de maintenir une bonne fluidité de trafic (un train en avance peut gêner un autre train).

Ainsi, les retards ne sont jamais compensés par des avances, ce qui explique bien cette tendance d'en moyenne 5 minutes au fil des années.

Mais à quoi sont dus ces retards ?

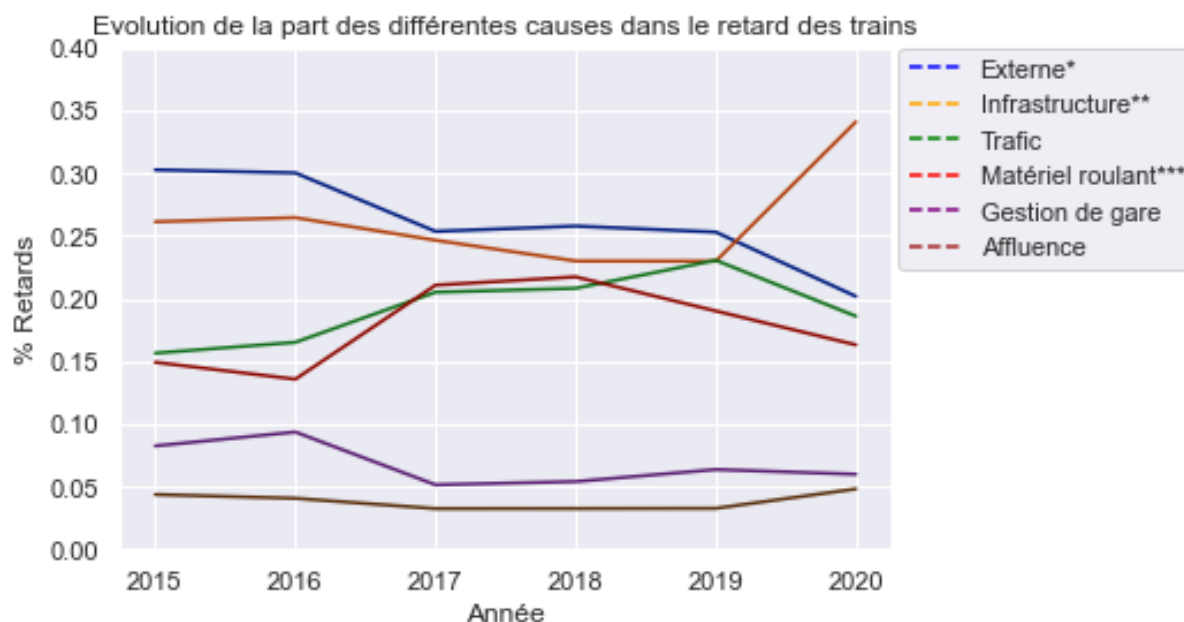
### 1. Etude temporelle

Dans ce dataframe, chaque retard est expliqué par l'un des six motifs suivants :

Motifs	Description
Retards pour causes externes	Les intempéries, les dépôts d'obstacles sur la voie, les signaux d'alarme tirés à tort...
Retards pour causes d'infrastructure	Maintenances (défaillance des feux par ex)
Retards pour causes de gestion de trafic	Problèmes d'organisation
Retards pour cause de matériel roulant	Présence sur les voies de véhicules ferroviaires gênants
Retards pour causes de gestion en gare	-
Retards pour causes voyageurs/affluence	-

Nous nous intéressons ici à leur taux d'occurrence au fil des années\*.

(\*Il est cependant important de noter que la démarche ici n'est qu'indicative étant donné que la nature temporelle des données n'a qu'une portée annuelle puisqu'elle a été effectuée par moyennisation.)



Ce graphique compile les différentes causes de retard entre 2015 et 2020.

On remarque que les causes externes (ici en bleu), qui comprennent les manifestations sociales et les grèves, ne prennent pas une place aussi conséquente qu'attendue dans l'explication des retards.

Nous aurions en effet pu nous attendre à des pics de retards en 2016 lors du passage de la loi El Khomri, en 2018 par les mouvements gilets jaunes, puis en 2019 avec la réforme des retraites : il semblerait que ces mouvements se soient plus traduit par une annulation massive des correspondances, que par des retards.

Les retards causés par des problèmes d'infrastructures (en orange) ont une tendance à la hausse (10% des retards supplémentaires entre 2015 et 2020), cela peut indiquer un manque d'entretien des voies.

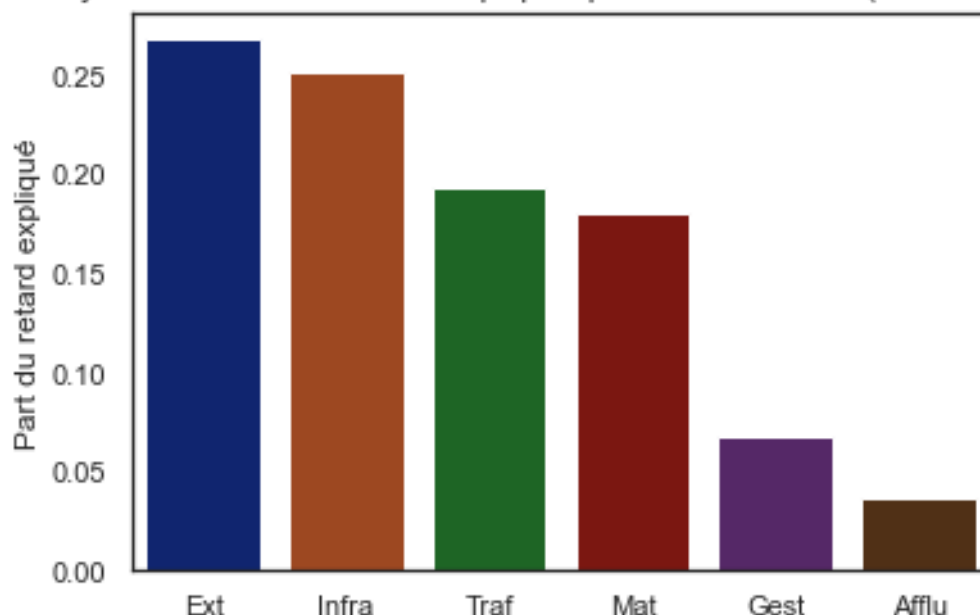
## 2. Données Coupe transversale

Après avoir étudié l'évolution de ces taux de retard dans le temps, nous avons décidé de nous intéresser à l'étude en coupe des différents motifs de retard.

Cependant, les variables du dataframe n'étant pas discrètes, dresser un barplot des causes ne semblait à priori pas possible.

Afin de pallier ce problème, nous avons calculé la moyenne de chacune des causes de retard, et les avons compilées dans une liste : nous pouvons ainsi dresser un graphique représentant la part d'importance de chaque type de retard.

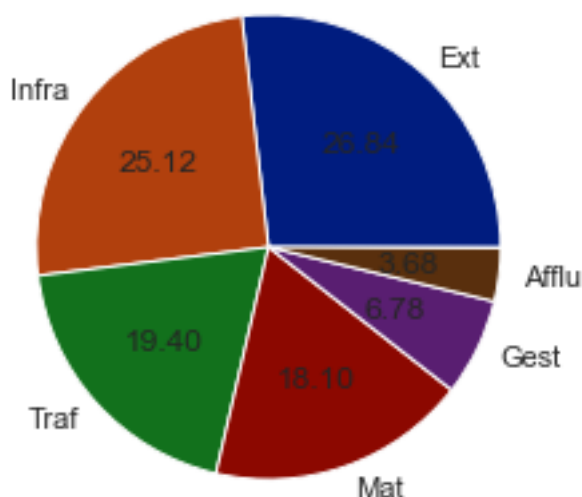
Part moyenne du retard des trains expliquée par une des causes (Période 2015-2020)



Comme attendu, les principaux retards des TGV sont dus à des causes externes (météo, mouvements sociaux), des problèmes d'infrastructures (problèmes de signalisations, de voies ...), de trafic, et enfin de matériel roulant (pannes).

Un diagramme circulaire retranscrit bien l'importance de ces 4 causes majeures :

Répartition moyenne des causes de retard (en %)



Notons tout de même que les causes externes sont des causes sur lesquelles la SNCF ne peut agir.

Une première intuition graphique dans l'étude des retards de TGV serait de concentrer le budget de la SNCF dans ces 3 autres causes majeures de retard.

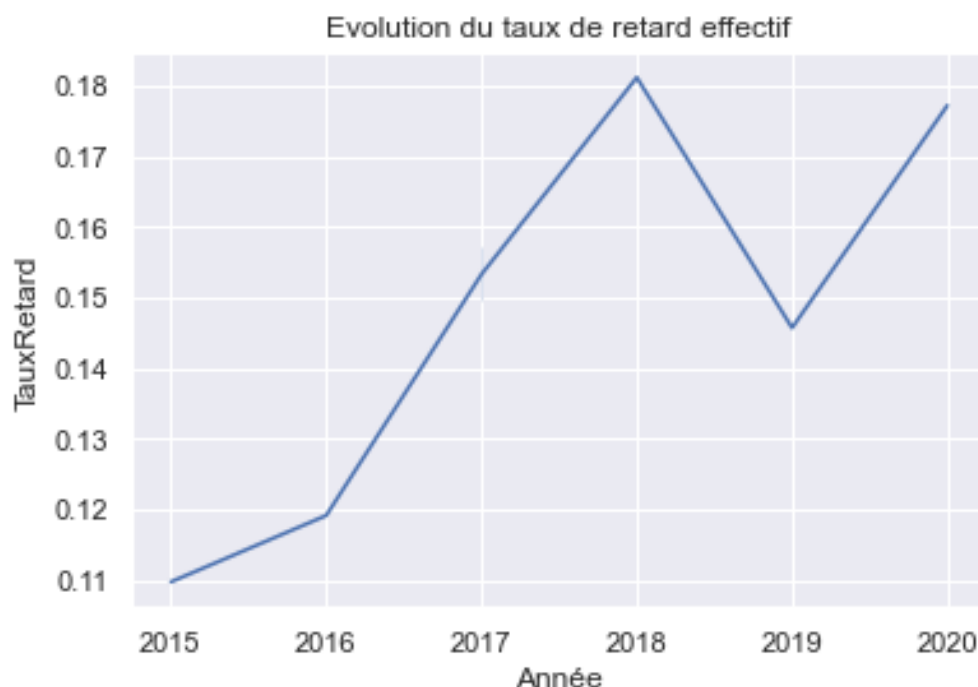
### 3. La variable Taux de retard

Comme dit précédemment avec les causes externes, de nombreux trains prévus (représentés par la colonne “Nombre de circulations prévues”) sont annulés.

Afin de ne plus prendre en compte ces annulations dans notre analyse, nous avons choisi de créer deux nouvelles variables :

- La première, “**NbTrainE**” représente le nombre de trains effectifs, qui se calcule en retranchant au nombre de circulation prévues (que nous appellerons désormais “**NbTrainT**” pour théorique), le nombre de trains annulés.

La seconde, “**TauxRetard**” représente le taux de retard des correspondances ayant réellement eu lieu : on divise le nombre de trains en retard par le nombre de trains effectivement en circulation.



Ce graphique nous informe sur le taux de retard global des TGV entre 2015 et 2020. Le pic des retards est atteint en 2018, où près d’un TGV sur cinq aurait été retardé. La tendance semble nettement croissante, en effet, en 2015 seul un trajet sur 10 était concerné

## b) Le module OutilsStatistiques

Afin d'appliquer nos cours d'économétrie et dans un souci d'obtenir des résultats plus précis qu'avec les autres packages (qui en général renvoient des chiffres arrondis), nous avons décidé de programmer plusieurs éléments d'économétrie (essentiellement appliqués à la régression par la méthode des moindres carrés ordinaires) dans un module appelé ***OutilsStatistiques***. Ce module contient une fonction :

- La fonction **HeatMap()** permet de dresser un graphique qui est une représentation visuelle de la matrice de corrélation des variables sélectionnées.

Ce module contient également de la programmation orienté objet. Ainsi, nous définissons la classe *LROLS* comme étant une régression linéaire réalisée par les moindres carrés ordinaires. Ses attributs sont le dataframe des variables explicatives (la matrice *self.X*), la série de la variable dépendante (le vecteur *self.Y*) ainsi qu'un nom à donner à la régression qui apparaîtra lors de plusieurs sorties (*self.Label*). Les méthodes sont les suivantes :

- La méthode **VarGraph()** dessine le graphique de toutes les variables du modèle. Le but est ici de constater visuellement la constance des moments d'ordre 1 et 2 pour prévoir l'apparition potentielle d'outliers.
- La méthode **CorrGraph()** dessine les scatter plots de toutes les explicatives par rapport à la variable expliquée. Ces scatter plots permettent de confirmer la présence d'outliers, mais aussi de constater si à priori il y a bien une potentielle relation linéaire entre les deux variables.
- La méthode **GetCoeffs()** donne les coefficients des MCO.
- La méthode **GetResiduals()** donne les résidus de la régression.
- La méthode **GetSD()** donne les écarts-types des coefficients estimés.
- La méthode **GetTStats()** donne les t-statistiques des coefficients afin de tester leur significativité.
- La méthode **PlotResiduals()** dessine le graphique des résidus. Ce graphique renseigne visuellement sur une potentielle hétéroscédasticité ou une autocorrélation des résidus qui viendrait invalider le modèle MCO.
- La méthode **GetR2()** renvoie un tuple contenant le coefficient de détermination ainsi que le R2 ajusté.
- La méthode **WhiteS()** réalise un test de White simplifié. Dans ce test, une régression auxiliaire est estimée où les carrés des résidus sont expliqués par la prédiction de la variable expliquée ainsi que le carré de cette prédiction. La



statistique  $NR^{\{2\}}$  est à comparer à un chi-2 à deux degrés de liberté. L'hypothèse nulle suggère un modèle homoscédastique.

- La méthode **DurbinWatson()** réalise un test de Durbin-Watson. La statistique de Durbin-Watson est le rapport de la somme des différences des résidus (avec un lag d'une période) sur la somme des carrés des résidus et elle est comparée aux valeurs de la table de Durbin-Watson. Ce test peut conclure à une autocorrélation (positive ou négative), à une absence d'autocorrélation voire à une situation de doute.
- La méthode **ExportResults()** exporte les résultats de la régression et les paramètres clés dans un fichier "*OLS.txt*" situé dans un dossier "*Exports*" qui est placé dans le même répertoire que le fichier *main.py*. Plusieurs régressions peuvent être stockées dans ce fichier.

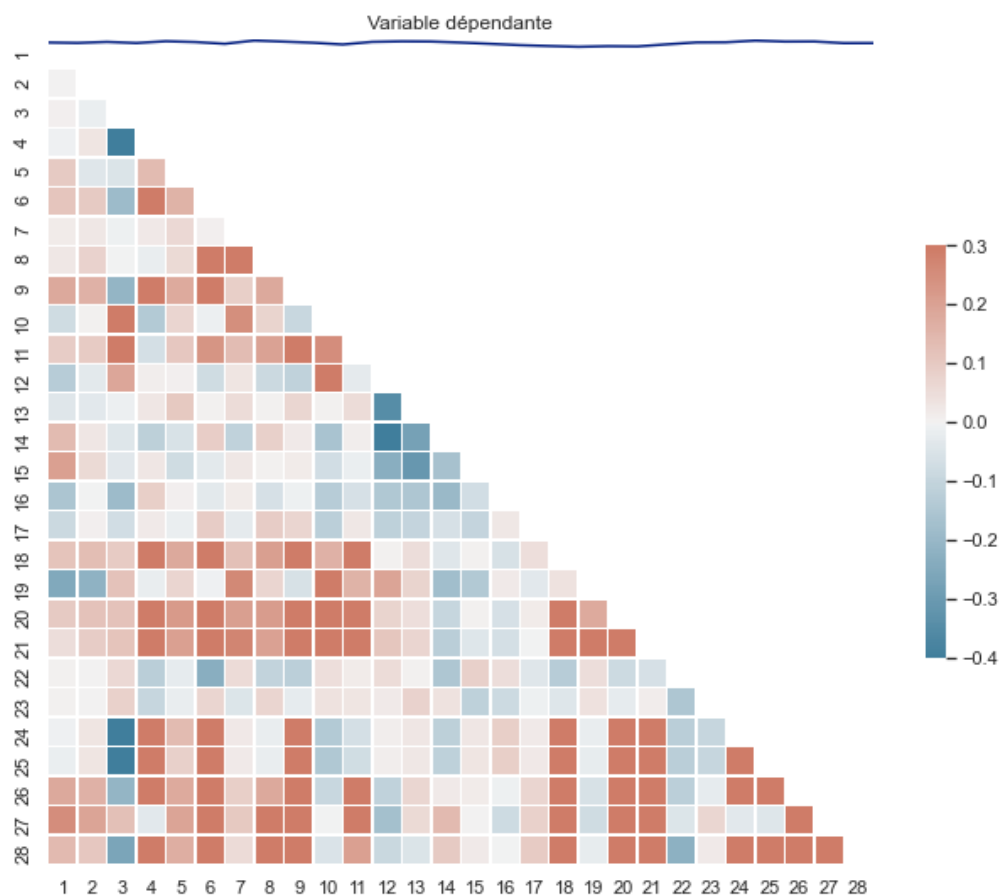
### III- Partie 3, Modèle et résultats.

#### a) Régression : étape préliminaire

##### Corrélation des variables

Avant d'entamer la régression de notre modèle, il convient d'étudier la corrélation de nos variables à l'aide d'une Heatmap. Cette étape de vérification est indispensable à la "bonne santé" d'un modèle : en effet, avec des variables trop corrélées, il est impossible de procéder à l'estimation de notre modèle.

La Heatmap est une représentation graphique de la matrice de corrélation linéaire, qui s'intéresse aux corrélations deux à deux des variables. Le rouge montre une corrélation positive entre deux variables, le bleu représente une corrélation négative, et les variables opaques seront les moins corrélées.



Nous constatons que les variables de notre dataframe sont peu corrélées, avec des corrélations comprises entre -0.4 et 0.3, avec une majorité de variables opaques.

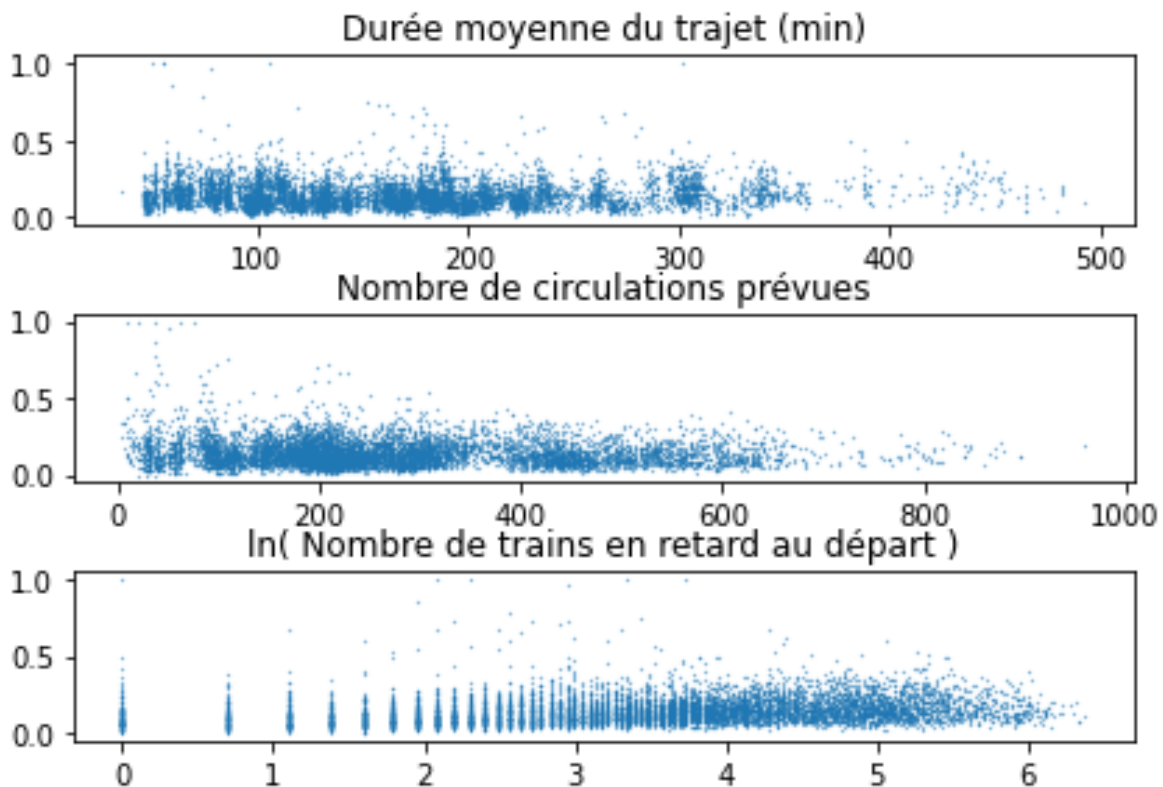
Une corrélation trop élevée nous aurait forcé à supprimer les variables concernées.

## b) Choix des variables

Le choix des variables s'est orienté sur :

- La durée moyenne du trajet en minute, car un trajet plus long entrainera une plus grosse probabilité de retard
- Le nombre de circulation prévues : en effet, une ligne plus empruntée est mieux entretenue et est plus soumise à des objectifs de performances
- Le nombre de trains en retard au départ : un train ayant un retard au départ le sera probablement à l'arrivée.

En effet, en traçant les scatter plots de ces variables par rapport au taux de retard construit précédemment, nous pouvons remarquer une vague tendance linéaire :



## c) Création et exécution du modèle

Alors, nous réalisons une estimation MCO de :

$$TauxRetard_i = \beta_0 + \beta_1 DuréeMoyenne_i + \beta_2 NombreCircu_i + \beta_3 ln(RetardDepart)_i + \varepsilon_i$$

Où les résultats donnent :

```

=====
                        OLS Regression Results
=====
Dep. Variable:          TauxRetard      R-squared:                0.134
Model:                  OLS             Adj. R-squared:           0.133
Method:                 Least Squares   F-statistic:             384.8
Date:                   Sun, 22 Nov 2020 Prob (F-statistic):       2.05e-232
Time:                   16:14:42         Log-Likelihood:          8384.1
No. Observations:       7482            AIC:                    -1.676e+04
Df Residuals:           7478            BIC:                    -1.673e+04
Df Model:                3
Covariance Type:        nonrobust
=====

```

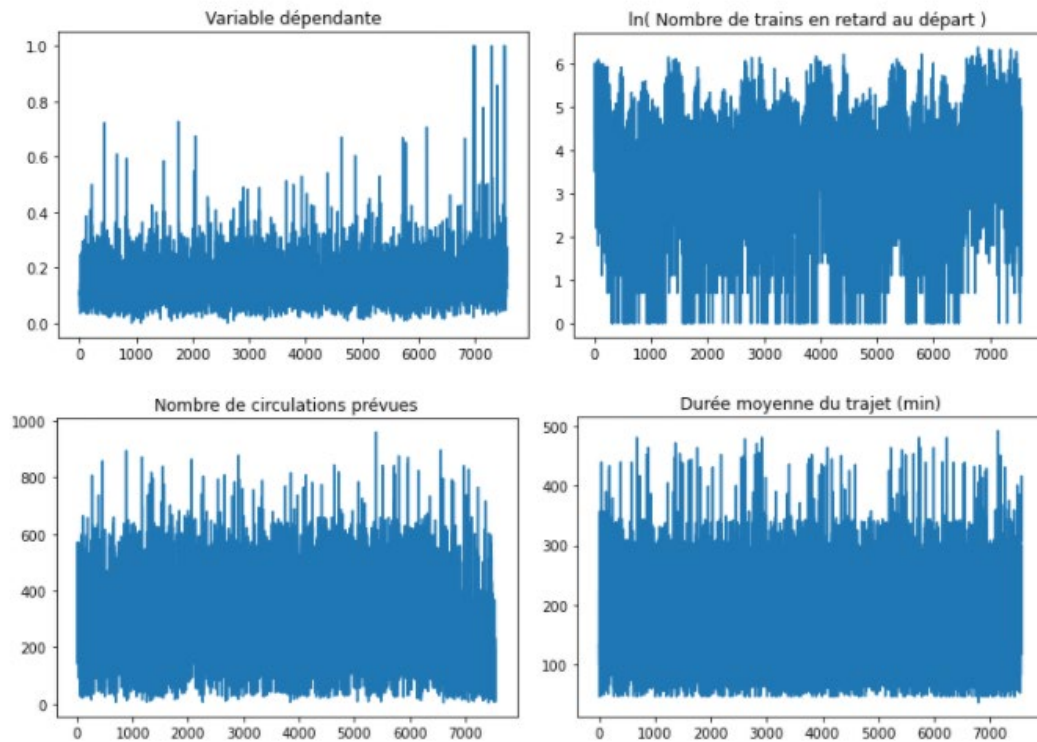
	coef	std err	t	P> t	[0.025	0.975]
const	0.0868	0.004	22.205	0.000	0.079	0.095
Durée moyenne du trajet (min)	4.838e-05	1.26e-05	3.850	0.000	2.37e-05	7.3e-05
Nombre de circulations prévues	-0.0001	7.14e-06	-20.351	0.000	-0.000	-0.000
ln( Nombre de trains en retard au départ )	0.0262	0.001	32.963	0.000	0.025	0.028

```

=====
Omnibus:                 4528.640      Durbin-Watson:             1.498
Prob(Omnibus):            0.000        Jarque-Bera (JB):          94229.710
Skew:                     2.515        Prob(JB):                  0.00
Kurtosis:                 19.642        Cond. No.:                  1.48e+03
=====

```

Toutefois, en bons économètres, nous avons décidé de nous fier à nos propres calculs et d'étudier les limites du modèle précédemment construit, d'où l'existence du module ***OutilsStatistiques*** qui est codé en orienté objet. Nous faisons d'abord appel à la fonction **VarGraph()** qui donne un graphique par variable dans le but de constater la constance (ou non) de leurs moments d'ordre 1 et 2 :



Alors, nous souhaitons savoir si nos variables explicatives pourraient avoir une relation linéaire avec la variable d'intérêt. Nous faisons appel à la fonction **CorrGraph()** qui donne le graphique situé plus haut. Ainsi, nous calculons les différents éléments de la régression MCO par les opérations matricielles basiques avec **GetCoeffs()**, **GetResiduals()**, **GetSD()**, **GetTStats()**. Nous obtenons respectivement les résultats suivants :

**Coefficients :**

[0.08684212962434112, 4.837688506202512e-05, -0.00014531957516381304,  
0.02616590789964096]

**5 premier résidus :**

[-0.08383311444547584, -0.07825515690838969, -0.09234472964466964,  
-0.05033157620111693, -0.1340225804694833]

**Écarts-types des coefficients :**

[0.003910933296280071, 1.2564848952510154e-05, 7.140695724727445e-06,  
0.0007937925958531252]

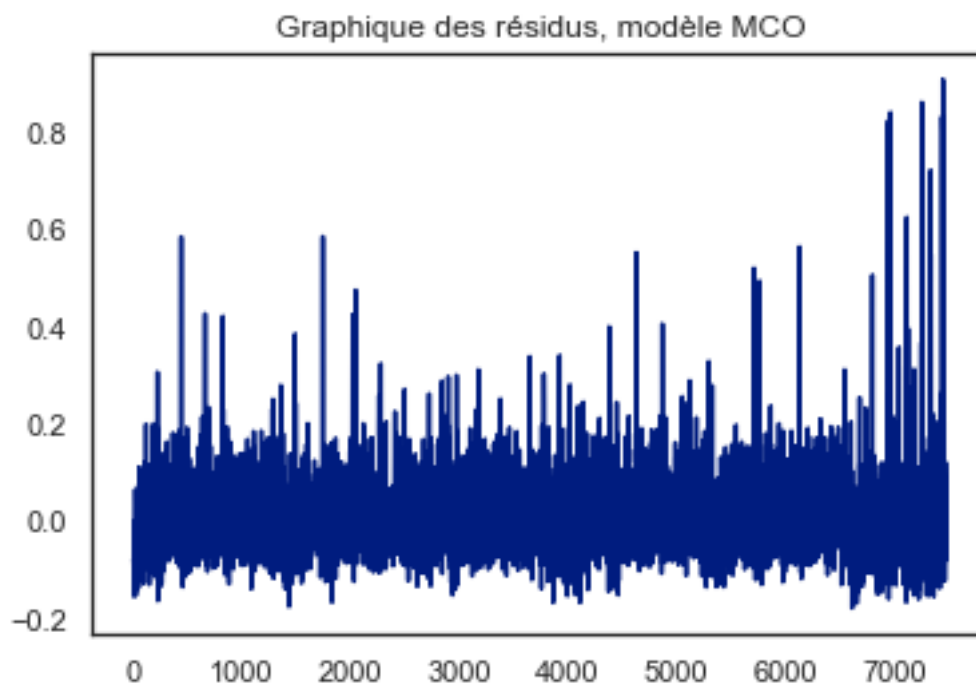
**T-statistiques :**

[22.2049631240047, 3.8501764123762574, -20.350898675123116, 32.96315440120635]

**R2 et R2 ajusté :**

(0.1337175628099564, 0.13337003040669748)

Nous constatons alors que les calculs MCO sont bien identiques à ceux fournis par la librairie *StatModels*. Toutefois, conscients des limites très rapidement atteintes par cette méthode de régression, nous souhaitons observer le graphique des résidus. La fonction **PlotResiduals()** nous donne :



La variance des résidus n'apparaît pas constante, mais nous ne savons pas spécifier la forme de cette potentielle hétéroscédasticité. Afin de ne pas perdre beaucoup de degrés de liberté, nous avons mis en place un test de White réduit où les carrés des résidus de la régression sont eux-mêmes expliqués par la prédiction et le carré de cette prédiction.

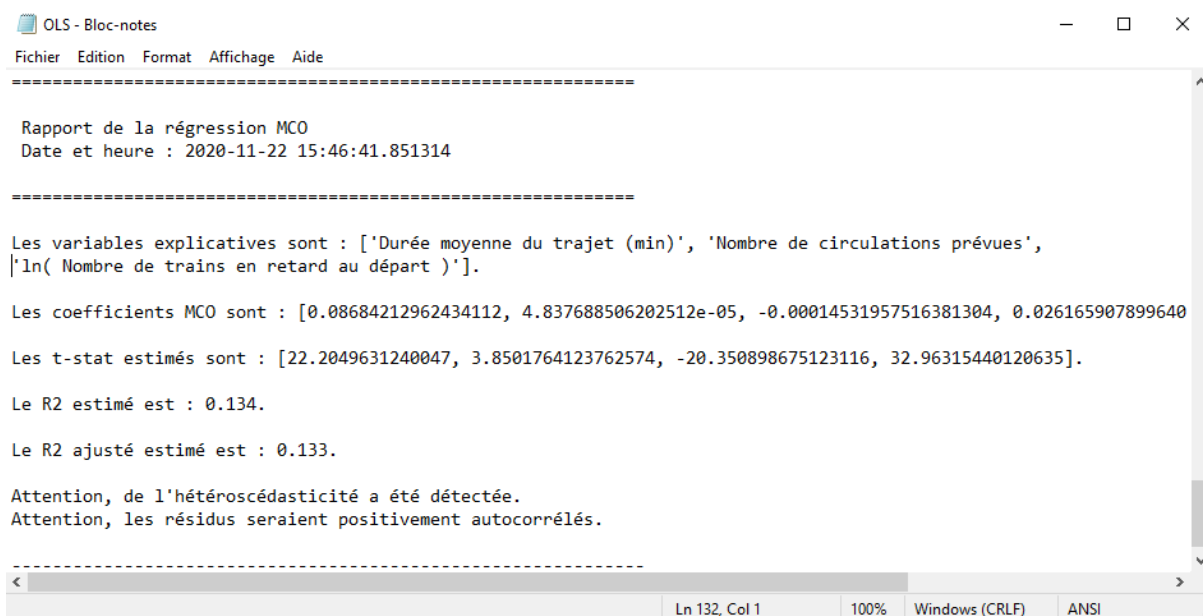
Dans notre cas, la statistique de test suit un  $\chi^2$  à deux degrés de liberté. La fonction *WhiteS()* nous renvoie une statistique égale à 39.55 et au risque de type 1 de 5%, la valeur critique est de 5.99 et nous affirmons qu'il y a hétéroscédasticité :

```
In [13]: Regression.WhiteS() # Il y aurait hétéroscédasticité d'après le test de White simplifié.
La statistique calculée 39.546 est supérieure à la valeur critique 5.99 observée au seuil de 5%
et à deux degrés de liberté. Au risque de première espèce de 5%, il y a hétéroscédasticité.
```

De plus, la fonction **DurbinWatson()** réalise un test de Durbin-Watson et nous pouvons également affirmer au risque de première espèce de 5% que les résidus sont autocorrélés de manière positive :

```
In [14]: Regression.DurbinWatson() # Il y aurait également autocorrélation des résidus, cf.
Durbin-Watson.
DW = 1.498, il y a une autocorrélation positive.
Out[14]: 1
```

Alors, notre modèle devra prendre en compte ces spécificités afin de prédire les taux de retard de manière convenable. Pour garder une trace de ce modèle, nous utilisons la fonction **ExportResults()** qui exporte toutes ces données dans un fichier texte lui-même rangé dans un dossier “Exports” :



```
OLS - Bloc-notes
Fichier Edition Format Affichage Aide

=====

Rapport de la régression MCO
Date et heure : 2020-11-22 15:46:41.851314

=====

Les variables explicatives sont : ['Durée moyenne du trajet (min)', 'Nombre de circulations prévues',
'ln( Nombre de trains en retard au départ )'].

Les coefficients MCO sont : [0.08684212962434112, 4.837688506202512e-05, -0.00014531957516381304, 0.026165907899640]

Les t-stat estimés sont : [22.2049631240047, 3.8501764123762574, -20.350898675123116, 32.96315440120635].

Le R2 estimé est : 0.134.

Le R2 ajusté estimé est : 0.133.

Attention, de l'hétéroscédasticité a été détectée.
Attention, les résidus seraient positivement autocorrélés.

=====
Ln 132, Col 1 100% Windows (CRLF) ANSI
```

Le fichier “OLS.txt” peut contenir plusieurs rapports de régressions à la fois, car la fonction **ExportResults()** l’ouvre avec le paramètre “a+”.

# Témoignages, commentaires d'expérience

- **Gaëtan** : « Avant de suivre ce cours, j'avais des connaissances extrêmement basiques en Python du fait du suivi de tutoriels. Ces enseignements m'ont permis de consolider mes bases, mais surtout de découvrir des librairies importantes en data science (je ne connaissais que *Numpy*, *Pandas*, *Matplotlib* ect. de nom) ainsi que la programmation orienté objet (qui à mon avis est une chose extrêmement intéressante et utile). Enfin, j'ai apprécié le fait de pouvoir appliquer toutes ces connaissances dans une analyse de données "réelle" où j'ai pu conforter mon savoir de la meilleure manière qui soit en programmation : en tombant sur des bugs, en les maudissant pour 2 secondes à 30 minutes, puis en les résolvant. Au delà de Python, ce projet m'a permis d'appliquer mes cours d'économétrie à la fois pour du calcul théorique (typiquement, le codage du module *OutilsStatistiques*) mais aussi pour l'interprétation (comprendre les erreurs du modèle, qui ici sont en très grande partie dues aux outliers). Ayant toujours voulu avoir des bases solides dans ce langage avant de me lancer dans des compétitions sur Kaggle, je pense que ce cours m'aura donné les clés nécessaires pour tenter ces concours. »
- **Alexis** : « Au cours du projet, j'ai surtout appris à représenter graphiquement une base de données. En effet, il n'est pas toujours intuitif de faire parler des données, et j'y ai rencontré de nombreux problèmes : Par exemple, au départ je pensais impossible la création du barplot représentant chacune des causes de retard avec notre jeu de donnée (du fait de la disposition originelle du dataframe et des variables non-discrètes) : c'est après de nombreuses tentatives infructueuse qu'il m'est venu à l'idée de créer une liste comportant les moyennes de chaque cause pour en dresser un barplot. C'est ainsi que chaque partie du cours m'a été utile pour résoudre ce problème : il ne suffit pas de connaître les commandes d'exécution d'un graphique pour parvenir à des visualisations pertinentes. Cela peut certes paraître basique, mais l'exercice de visualisation graphique d'une base de données était loin de m'être intuitif.

D'une manière plus globale, je me suis rendu compte à quel point Python était polyvalent et pouvait concurrencer d'autres logiciels d'analyse de données (tant



en traitement de données qu'en visualisation graphique) tels que R et SAS. Cependant, ce que l'on peut faire en quelques commandes sur SAS se fera en bien plus d'instructions sur python (ce qu'on voit bien avec la proc reg par exemple).

Au-delà du python, j'ai également pu pour la première fois travailler en groupe de 3, et cette expérience fut très enrichissante. Bien que le contexte soit particulier du fait du confinement, nous avons parfaitement pu échanger via Discord et partager nos avancements afin de compléter le projet.

Dès que nous rencontrions un problème, le fait d'être plusieurs avec différentes approches, nous a permis de rapidement les résoudre et d'avancer bien plus efficacement.

Enfin, bien que je n'ai pas participé à la partie régression du dataframe, le fait d'avoir été témoin de l'avancement de cette partie du projet m'a permis de mieux la comprendre et je me sentirais aujourd'hui capable d'analyser d'autres bases de données sous python à l'aide d'outils économétriques basiques.

- **Mehdi** : « Bien que les dernières journées furent fastidieuses avec des cycles du sommeil ne dépassant pas en heures le chiffre 5, je peux très certainement affirmer que cet exercice m'a permis à moi et mes camarades d'énormément progresser. Outre les connaissances acquises en cours, c'est bien le fait d'avoir été livrés à nous-mêmes à la recherche de nos propres réponses qui a rendu cela possible. Du trial & error à la lecture de guide utilisateur/forums aux réponses parfois floues, les bibliothèques que je n'appliquais sans trop comprendre le pourquoi du comment en TD m'apparaissent maintenant comme intuitives notamment *Pandas*, *Matplotlib*, *Statsmodels* et même des modèles de prédictions sur *Scikit-learn* ! (Qu'on avait d'ailleurs commencé à construire pour prédire les retards mais ça faisait un peu beaucoup et on a malheureusement dû se débarrasser de cela). Au niveau du groupe, je pense que nous avons eu une très bonne organisation, parce que personne ne se limitait strictement à l'élaboration d'une partie, nous avons tous pu progresser ensemble et sur les mêmes outils. Enfin je pense que l'application à une situation concrète, encore plus pour la SNCF, de nos modèles est une introduction et même une avance prise pour ce que nous ferons au second-semestre et peut-être dans le futur en entreprise en tant que Data-Scientists.