CentraleSupélec

OPENCLASSROOMS

Anticiper le retard des vols

Contenu

Synthèse

Contexte :

Air Data, une nouvelle compagnie aérienne désire optimiser sa logistique et anticiper les retards possibles de sa flotte. A partir de données existantes sur d'autres compagnies aériennes, celle-ci nous demande de mettre en place un modèle de régression afin de prédire les possibles retards/avances.

Problème :

Ce problème est un problème de régressions. A l'aide de différents modèles, une évaluation de la prédiction sera faite. Par la suite, une optimisation sera faite sur les hyperparamètres des modèles ainsi qu'avec du Boosting. L'objectif majeur étant de prédire les courts retards fréquents.

Données :

Les données fournies sont des données issues d'une base publique gouvernementale (www.transtats.bts.gov). Ce site regroupe les données de chaque vol intérieur aux USA par mois sur les 30 dernières années. Dans notre étude, nous avons à notre disposition les données de l'année 2016.

Approche :

Après un nettoyage des données inutiles dans les différents datasets, différents modèles vont être testés sur différentes configurations des features. Les modèles seront ensuite évalués sur le MAE et MSE en cas d'égalité. Une recherche des meilleurs hyperparamètres sera faite à l'aide de Grid Search pour chacun d'entre eux.

Performances des modèles :

Lors de ce projet, un des problèmes majeurs a été la performance. Beaucoup de modèles ne passent pas en mémoire sur les Notebook et ont donc été découpés dans des scripts. Concernant l'évaluation, le MAE a été le critère principal. L'objectif étant de prévoir majoritairement les petits retards facilement anticipables plutôt que les gros retards potentiellement dus à des problèmes imprévisibles (sécurité, météo, panne, …).

Résultats :

Le 1er modèle souffre fortement du bruit et ne permet pas de prédire une tendance particulière. Quant au modèle 2, l'agrégation par heure du retard moyen permet de prédire tout de même une tendance globale correcte. Cependant les 2 modèles ne sont peut-être pas les plus pertinents pour ce type de prédiction car il y a une souffre d'underfitting. Des pistes d'évolutions, sont donc présentées à la fin du rapport.

Introduction

A partir des données des vols locaux aux USA de l'année 2016, l'objectif de ce projet est de mettre en place un modèle de régression afin de prédire les retards possible des avions de la compagnie AirData.

De ce fait, nous explorerons différentes possibilités de sélections de features, différents modèles ainsi qu'une optimisation sur le Mean Absolute Error (MAE) en jouant sur les hyper-paramètres.

Afin de rendre notre modèle utilisable, une API va être mise en place pour prédire les possibles retards. De ce fait, une des contraintes principales est que l'utilisateur doit avoir accès à toutes les données pour faire la prédiction.

Nettoyage

Les données que l'on possède se décomposent en 12 datasets avec les données par mois pour l'année 2016. Celui-ci regroupe tous les vols aux USA sur l'année complète et représente 5.6 millions de vols avec 65 features. Ce dataset a pour avantage de ne pas avoir de données manquantes.

Dans un premier temps, un rapide nettoyage a été effectué sur les datasets mensuels. Lors de celui-ci, une suppression des features inutilisables (tailNumber, wheelsON/OFF) ou en double (par exemple UniqueCarrier, AirlineID et Carrier) vont être réduites à une seule donnée unique (dans ce cas : UniqueCarrier). La sélection a été faite grâce aux descriptions des features présentes à cette adresse.

Hormis pour le dataset d'avril 2016, aucun autre nettoyage sera effectué. Cependant le dataset d'avril est erroné. Il comporte des lignes du mois de mars et qui ont parfois plus de features. Celles-ci sont donc supprimées.

Ce nettoyage a permis de réduire le dataset de 12 fichiers de 173 mo à 12 fichiers de 35mo. Par la suite, un second script a été mis en place pour grouper les 12 mois et générer le dataset d'exploration et le dataset utilisé pour les modèles (le second ayant moins de features et possiblement une autre structure). Ces 2 derniers datasets sont donc plus légers et utilisable assez facilement (245 mo et 131 mo).

Pour la mise en place de certains modèles que nous verrons par la suite, divers nettoyage additionnels ont été faits que l'on va voir dans les parties suivantes.

Nettoyage particulier – Modèle 1:

Pour le modèle 1, l'objectif était de garder l'ensemble des données sans agrégations. Afin d'éviter certains grands retards pénalisant fortement le modèle, 2 filtrages de données ont été mis en place :

Suppression des retards imprévisibles

J'ai considéré comme "retards imprévisibles", les retards liés aux vols supprimés ou détournés. De ce fait, toutes les lignes ayant subis un de ces cas ont été supprimées. Celles-ci sont

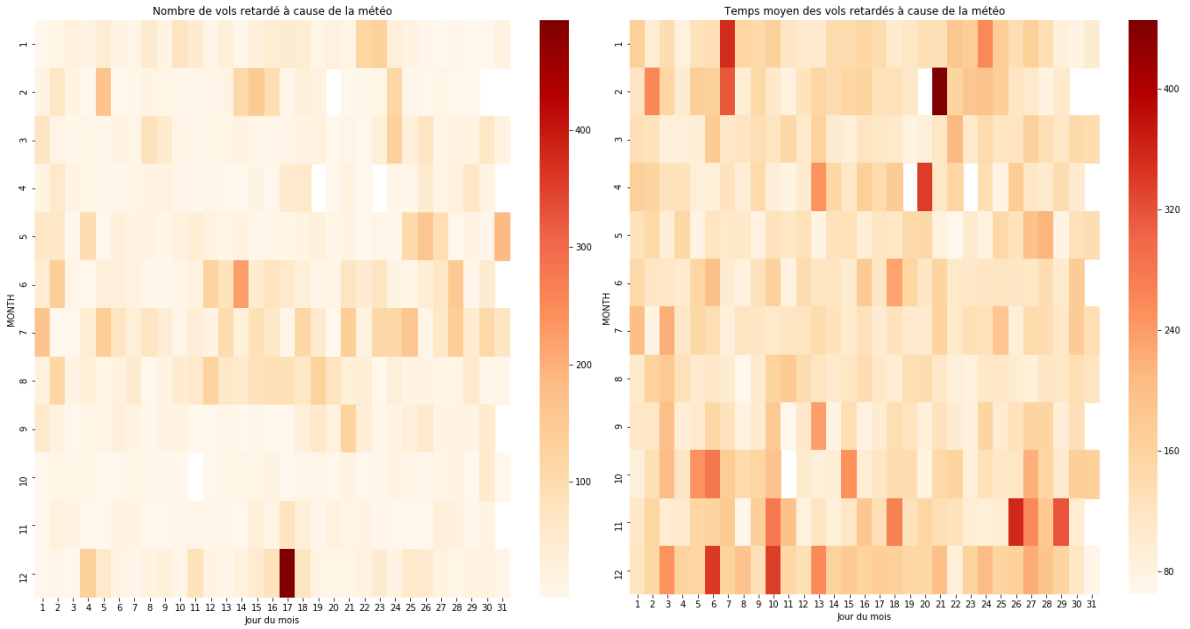
très minoritaires et n'ont donc pas beaucoup impacté le dataset (suppression de 65 219 lignes pour les vols annulés et 13489 pour les détournés sur 5.52 millions de vols enregistrés).

Suppression des retards semi-prévisibles

Les retards semi-prévisibles sont les retards qui ne sont pas imprévisibles mais qui restent tout de même exceptionnels. De ce fait, j'ai considéré comme features semi-prévisible le délai dû à la météo (par exemple le retard lié à la neige ou un ouragan est peu probable mais tout de même légèrement anticipables), à la sécurité (en renforçant la sécurité en amont des vols, ceux-ci peuvent possiblement être mieux anticipés afin de moins pénaliser les vols).

Pour ces features, j'ai décidé de ne supprimer que les vols ayant ce type de retard impactant de plus d'une heure le décollage. Cela ne regroupe que 273 retards pour la sécurité, 120 000 pour les délais dus à un avion en retard, 50 000 pour le NAS Delay et 12 300 à cause de la météo soit environ 3% des vols.

Si on regarde la répartition des vols supprimés dans le temps à cause de la météo par exemple on a :



On remarque que sur les 12300 vols ayant plus d'une heure de délai à cause de la météo, le nombre de vols est à peu près réparti uniformément durant l'année sauf le 17 décembre 2016. Par contre, si l'on regarde du côté du temps moyen perdu à cause de ces retards, la majorité à lieu en hiver (Novembre – Avril).

Nettoyage particulier – Modèle 2:

Pour le second modèle, l'objectif était d'agréger certaines données. Pour ce faire, je suis parti de 2 hypothèses.

- Le retard est lié au nombre de vols au décollage par aéroport
    - En effet, si un aéroport a une petite capacité, il ne sera pas en mesure de faire décoller 1 avion par minute par exemple. Par contre pour des aéroports internationaux, leur capacité de décollage/atterrissage est beaucoup plus élevée. De ce fait, si un aéroport est en mesure de faire décoller 12 avions par heure (1 toute les

5 minutes), s'il doit en faire décoller 13, le retard sera donc augmenté de 5 minutes par heure.
    - Cette façon de faire n'est pas exploitable tel quelle, cependant une matrice du nombre de vols moyen par heure et par aéroport va être utilisé pour la prédiction.
- Le retard ne dépend pas de l'aéroport d'arrivé
    - Dans ce cas, c'est plus logique. En effet, un vol ne sera pas décalé car l'autre aéroport a du retard. Dans ce cas, le vol sera mis en attente d'atterrissage au-dessus de l'aéroport d'arrivé mais en aucun cas ne sera décalé au décollage.

Ces 2 hypothèses m'ont permis de faire une agrégation par jour de la semaine, semaine de l'année, heure de décollage, aéroport de décollage et compagnie. Pour chaque features, le nombre de vols sera agrégé et le retard sera moyenné. De plus, cela évitera les gros pics à cause d'un seul et unique retard important par exemple. De ce fait, les retards liés aux retards non ou semi-prévisibles ont été gardés.

Exploration

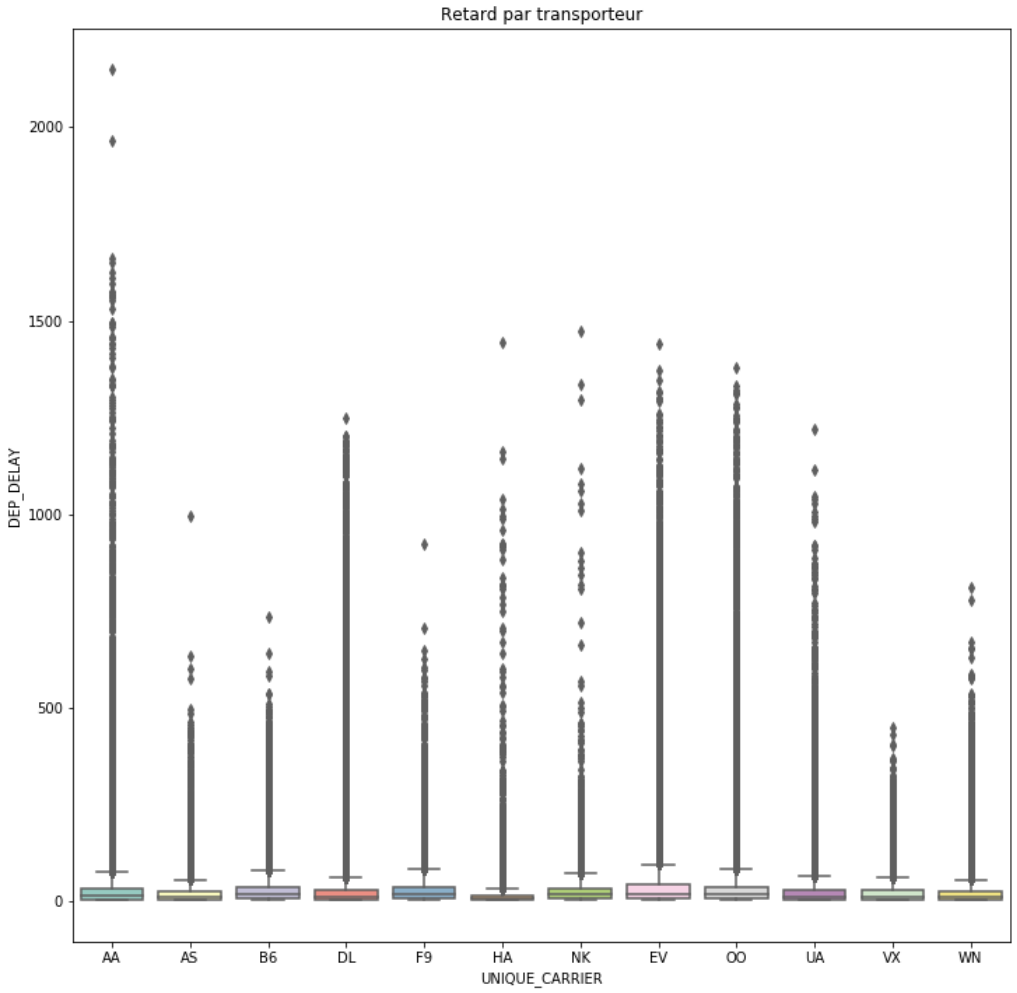
Etant donné que la prédiction doit se faire sur des paramètres connus, le choix des features a été fait basé sur les données sur un billet de vol, c'est-à-dire :

- Date et Heure
- Compagnie
- Aéroport de départ et d'arrivé
- Le numéro de vol/Immatriculation de l'avion n'est pas utilisé car peu pertinent dans notre cas. Certes, un avion précis peut-être plus souvent en panne qu'un autre et retarder le décollage, mais cela reste exceptionnel.

De ce fait, l'exploration a été basée sur ses divers paramètres:

Impacte de la compagnie

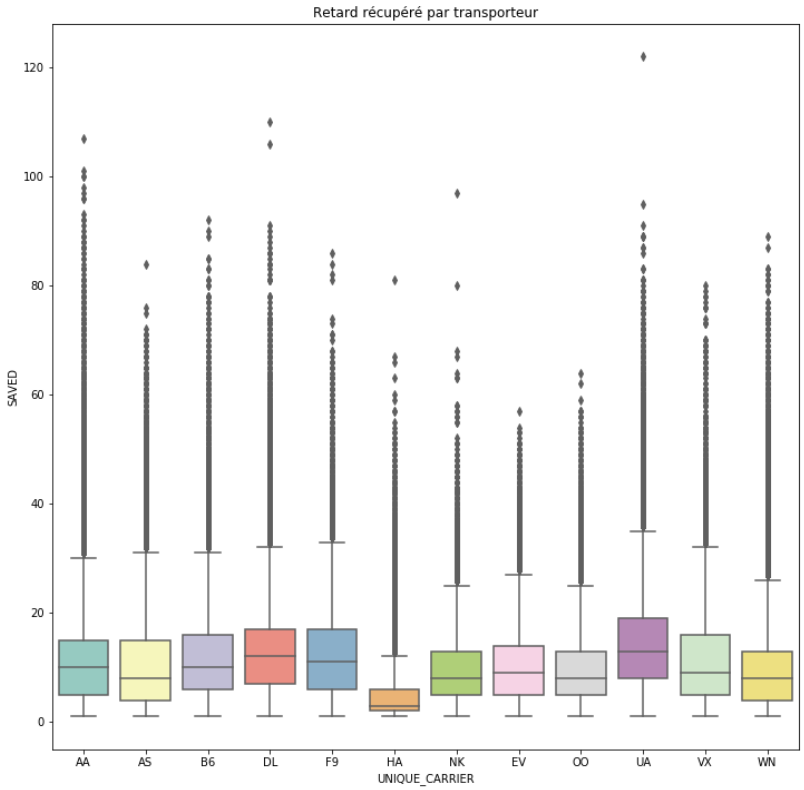
Dans un premier temps, une analyse a été faite sur les différentes compagnies afin de voir s'il y a des disparités importantes. Pour ce faire, un boxplot par compagnie a été fait :



De cette analyse, on remarque que globalement toutes les compagnies cumulent du retard d'un même ordre de grandeur. Par contre, parmi ceux-ci certains ont plus ou moins d'outliers très fort. Par exemple, la compagnie "B6" a eu comme retard maximum environ 10h (> 600 min) alors que "AA" a cumulé plusieurs vols à plus d'un jour de retard (> 1440 minutes).

Comme on n'a pas beaucoup de compagnies, celles-ci vont être OneHotEncodées pour garder des facteurs pénalisants par compagnie.

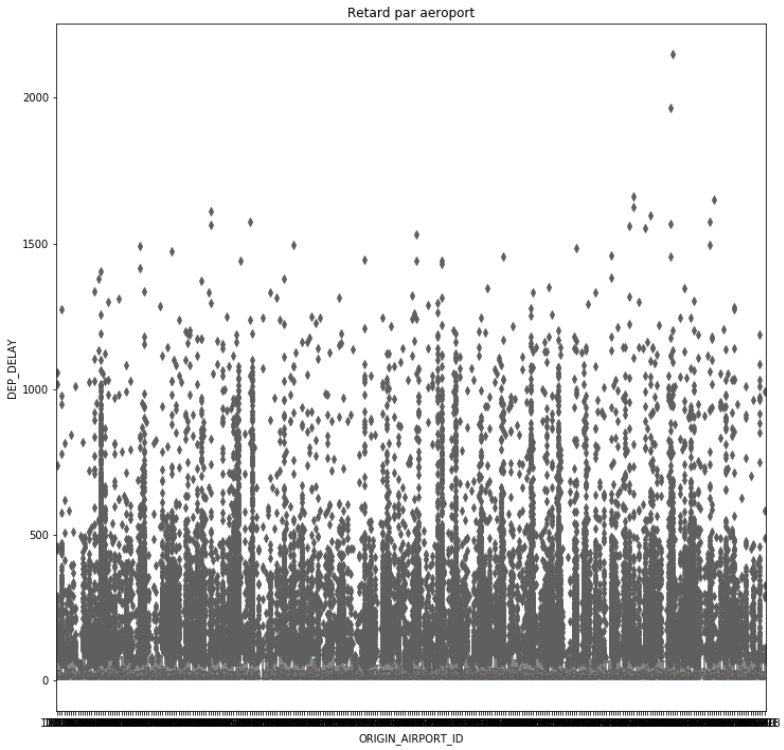
De la même manière, si on regarde au niveau du retard récupéré durant le vol (delay departure – delay arrival), on remarque que certaines compagnies sont plus efficaces que d'autres.



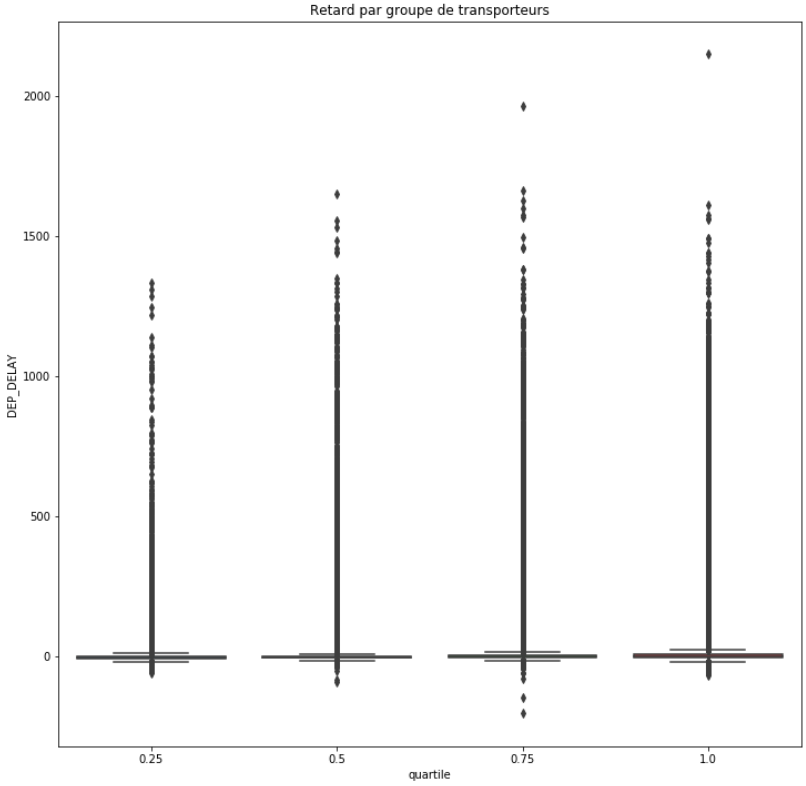
Bien que cette features ne soit pas utile en soit, cela met en relief le fait que la gestion des compagnies peut impacter leur retard à l'atterrissage.

Impacte des aéroports

Notre dataset contient 311 aéroports distincts. Si l'on regarde comme pour les compagnies le boxplot des délais au décollage et à l'atterrissage par aéroport on a une représentation comme suit:



On remarque majoritairement un bruit global. Certains aéroports sont moins en retard que d'autres mais il va être difficile d'utiliser un OHE dans ce cas. De ce fait, une autre approche a été prise. Le regroupement par rang des aéroports. En fonction des quartiles basés sur la moyenne des retards, les aéroports auront un rang assigné. Dans ce cas on trouve :



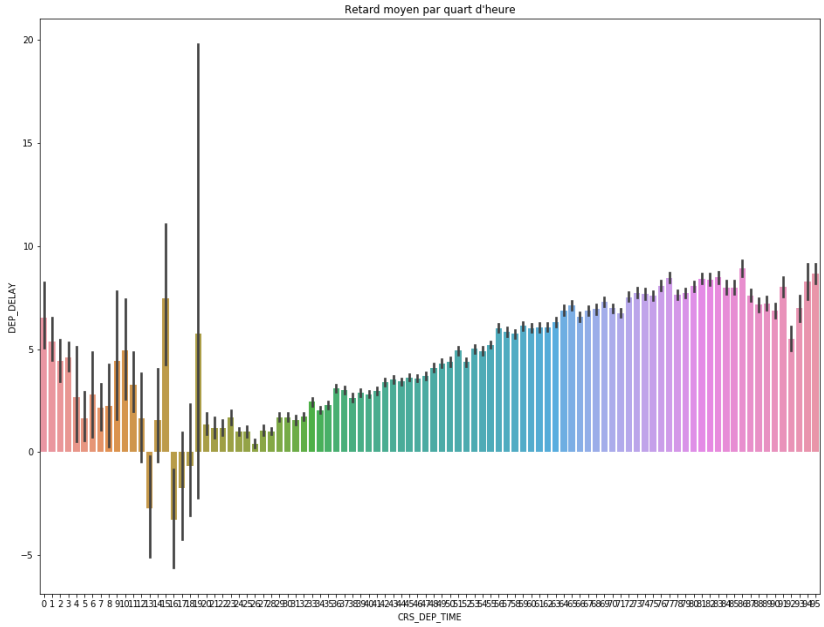
Cette solution est assez justifiée car on remarque un écart-type qui augmente avec le groupe, une moyenne qui monte ainsi que les outliers. Quant à la modélisation, il y a eu 2 cas:

- Test avec cette features "linéaire"
- Test avec cette feature "OneHotEncodée"

Impacte de l'heure

Modèle 1

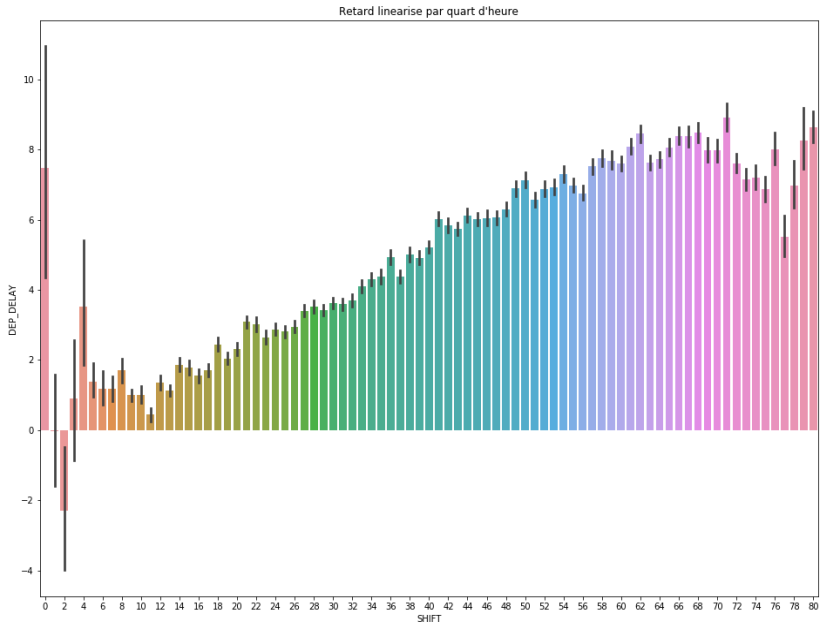
Dans le cas du modèle 1, le retard moyen a été observé par quart d'heure et on trouve :



On remarque une certaine linéarité que l'on peut améliorer avec la transformation suivante:

$$n = abs(n - 15)$$

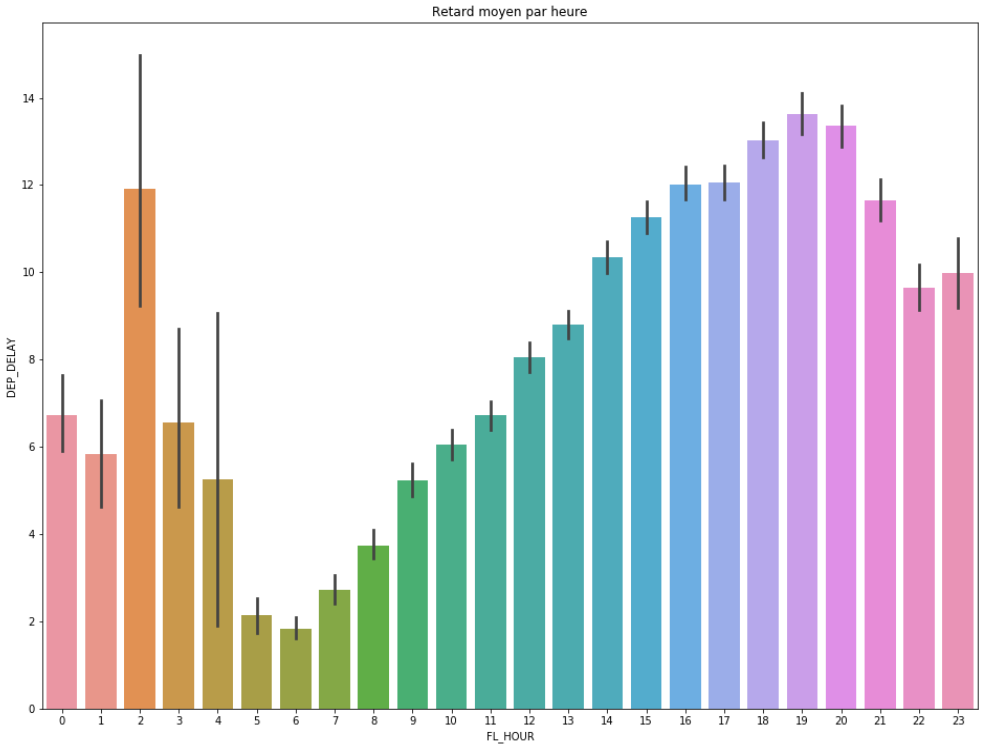
Celle-ci décale les retards 00h00 - 3h15 entre 3h15 - 6h30. De ce fait on trouve :



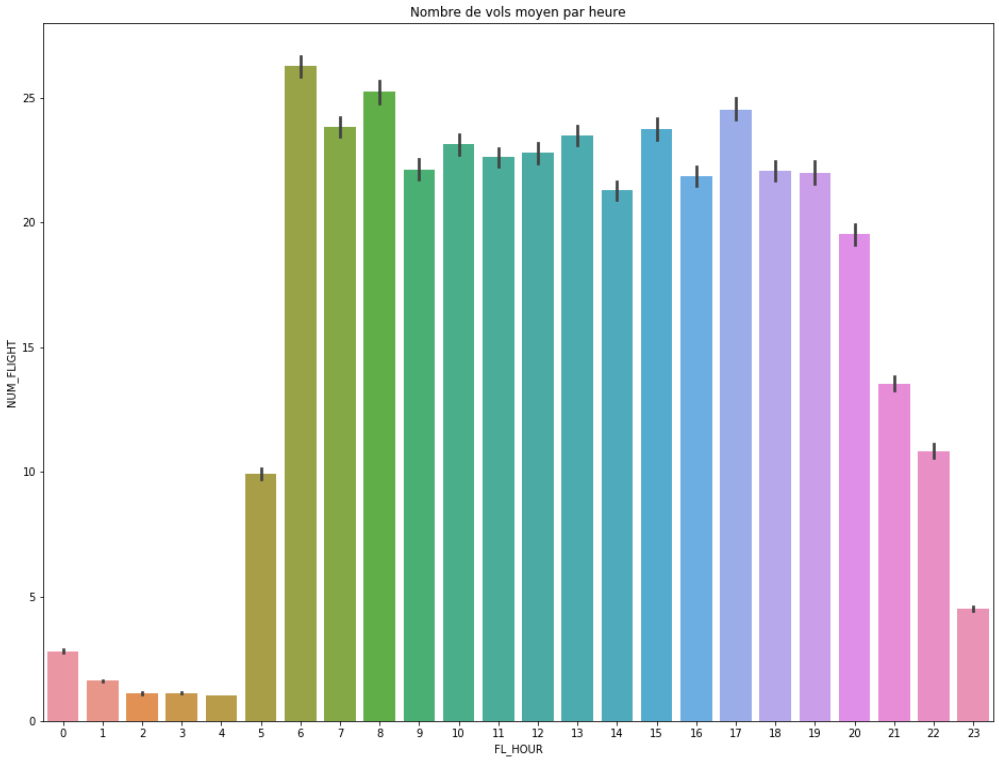
Avec cette linéarité couplée au peu de variance sauf à très faible horaire, on peut garder cette feature telle quelle.

Modèle 2

Concernant le modèle 2, souhaitant avoir le nombre de vol par aéroports et par jour, l'agrégation ne s'est faite que sur l'heure et non le quart d'heure. On trouve donc la répartition suivante:



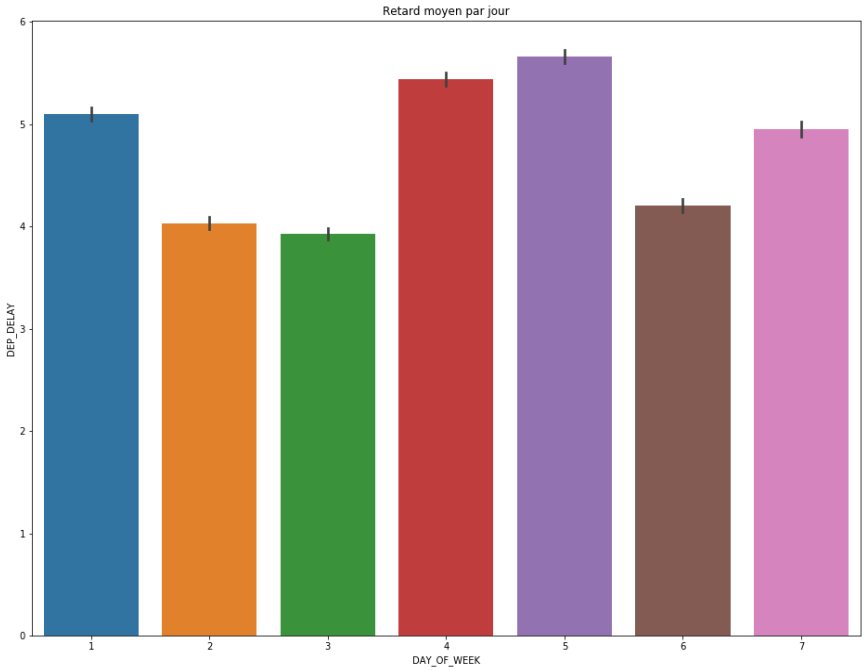
On retrouve le même phénomène. Cependant la transformation n'a pas été effectuée car on a le nombre de vols comme nouvelle features qui va compenser la non-linéarité.



Impacte du jour de la semaine

Cette feature est plutôt simple et judicieuse. En effet, la majorité des sociétés ont tendance à faire des voyages durant 1 semaine donc on peut s'attendre à des pics de vols le Lundi et Vendredi.

C'est le même cas pour les départs en vacance. Si on regarde le retard moyen par jour de la semaine on trouve donc :



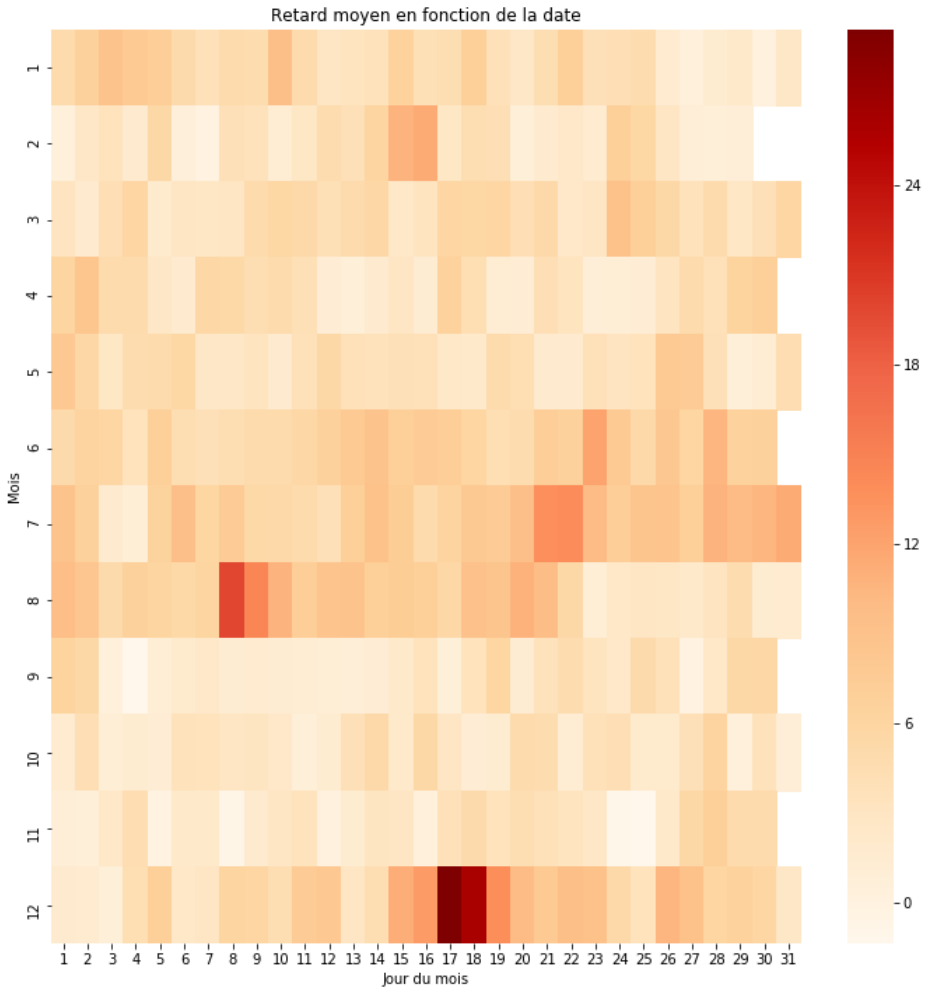
On remarque en effet ce phénomène avec un pic le Lundi et Vendredi mais aussi le Dimanche (surement des vols anticipés pour le travail). Le pic un peu moins prévisible est celui du Jeudi. Celui-ci est peut-être le retour des vols pris le dimanche.

Impacte de la date

Dans le cas de la date, il y avait 2 possibilités. Le premier cas était de travailler en terme de date du mois et le mois concerné (31 + 12 features) ou alors en parlant de semaines (52 features)

Cas Jour/mois

Pour le cas Jour/Mois, on peut regarder le retard moyen comme une matrice et on trouve :



Le problème est que l'on ne voit pas de relations particulières sauf des lignes légèrement plus sombres en été ou Décembre. De ce fait, il se difficile de prédire un retard tel que :

$$Retard\ moyen = \alpha_{mois} * mois + \beta_{jour} * jour$$

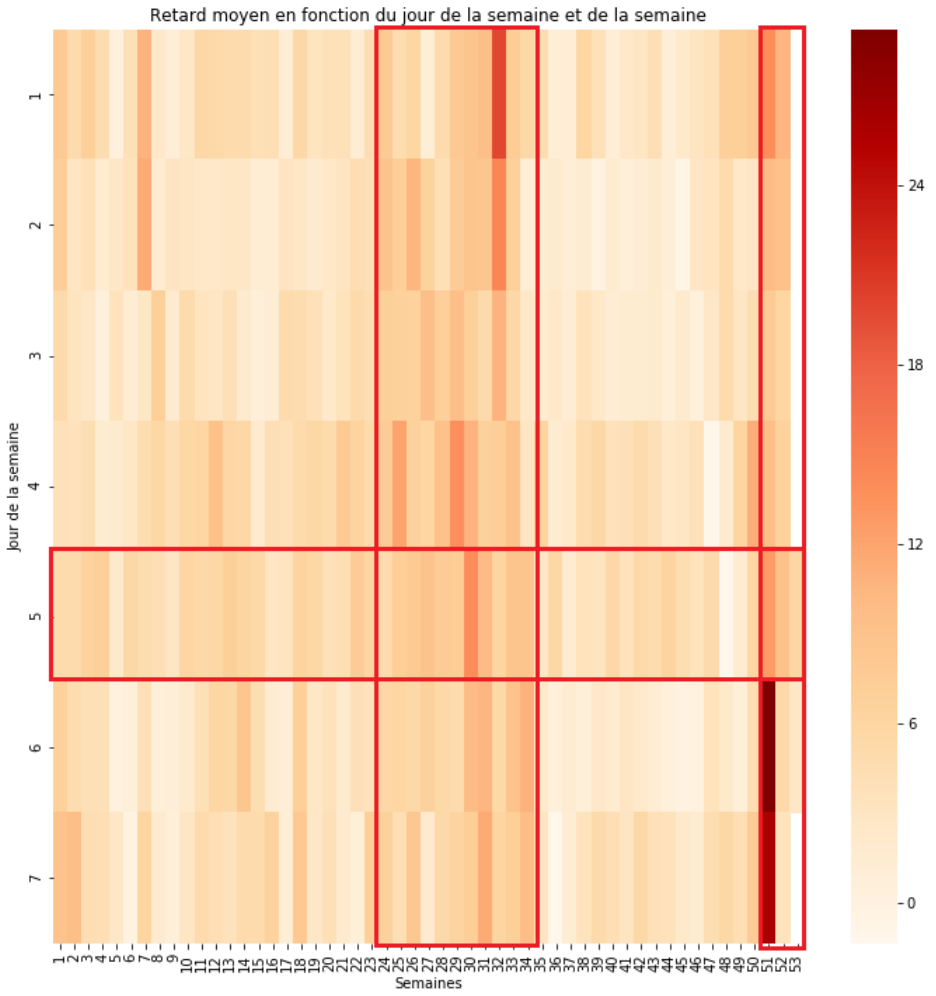
Il faudrait donc OneHotEncoder les 365 jours pour être plus précis.

Cas Semaines

Une autre façon de voir est de regarder en termes de semaines. En effet, jusque-là on regarde le retard par **heure dans la journée**, par **journée dans la semaine**. Il serait donc logique de continuer le "dezoomer" en regardant par **semaine dans l'année.** De plus, réfléchir par semaine est plus logique. Que ce soit les entreprises ou les particuliers :

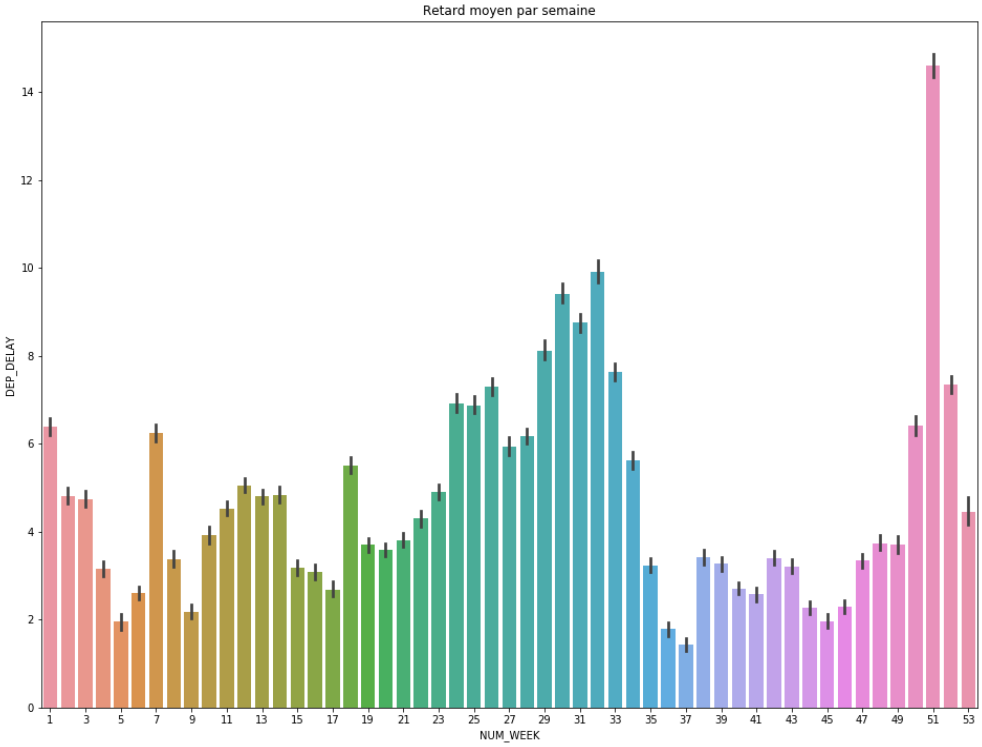
- les voyages d'affaires sont prévus par semaine et pas par date.
- les vacances sont prévues par semaines. Les gens partent par exemple la 1ere semaine d'août plutôt que le 1er août.

Maintenant, si on regarde le retard en fonction du jour de la semaine et de la semaine on trouve :



On remarque déjà plus de lignes et colonnes plus foncés (cf. les rectangles rouges). Plus l'intersection sera sombre. On a donc une relation un peu plus linéaire. Cependant cette solution n'est pas parfaite non plus. On va donc évaluer initialement les 2 modélisations sur les mêmes modèles pour choisir une solution.

Si on regarde du côté du retard moyen par semaine, on a :



On remarque que la variation est faible mais pas linéaire, de ce fait un OHE est nécessaire pour modéliser ce phénomène.

Modélisation

Au vu de la taille des datasets, il n'y avait pas beaucoup de modèles disponibles. Le KNN, SVM, Régression linéaire simple n'était pas envisageable à cause du nombre de données. Le choix s'est donc tourné sur le Stochastique Gradient Descent Regressor.

Pour ce qui est du Boosting/Bagging, le choix était aussi très restreint. Le Bagging ne peut être utilisé car il nécessite l'entrainement de plusieurs modèles en parallèles. Le Boosting peut être utilisé sous condition qu'il accepte le SGDR comme modèle. De ce fait, seul Adaboost est utilisable (les autres utilisant majoritairement les arbres décisionnels).

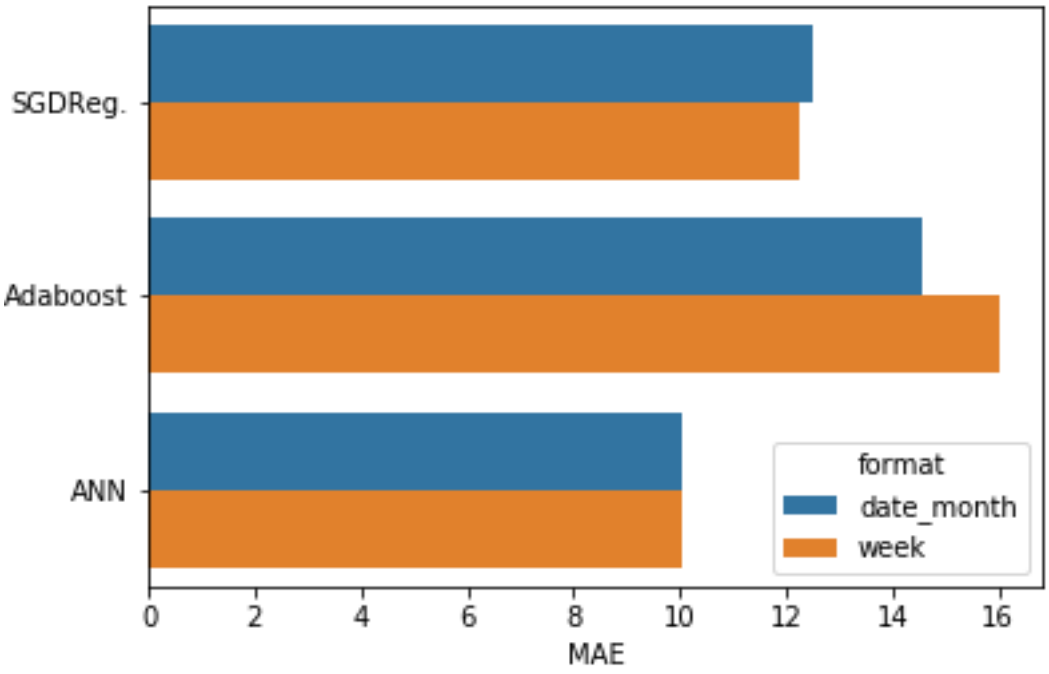
Pour la partie modélisation, elle a été faite en 2 principales parties :

Modèle 1:

Celui-ci est basé sur le dataset sans agrégations comme expliqué dans la partie Nettoyage. Dans un 1er temps, il a fallu valider le format de dataset à utiliser (jour/mois ou par semaine). Pour ce faire, un SGDR et adaboost ont été évalués sur chaque datasets et le modèle avec le plus faible MAE sera retenu.

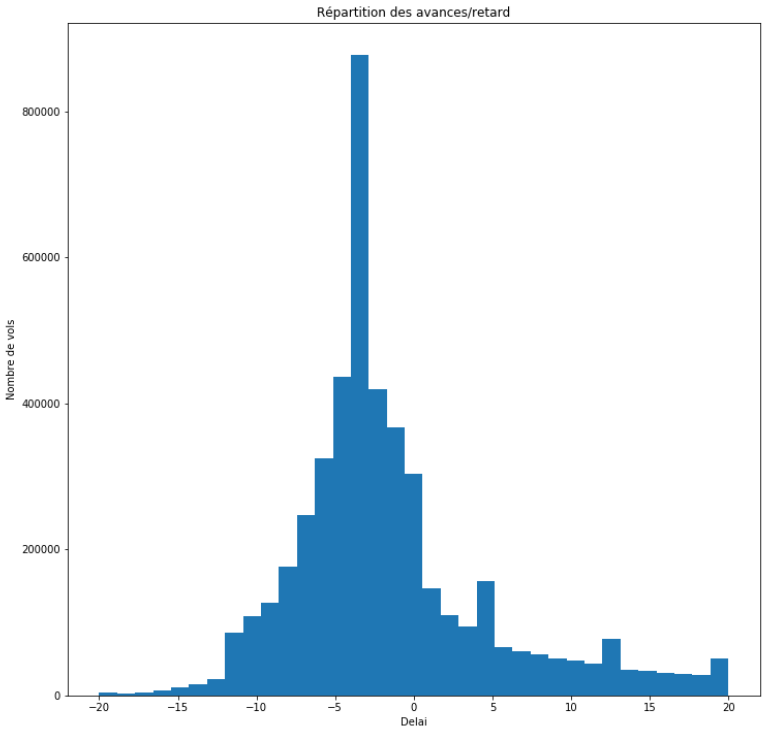
Sélection du dataset (week format / jour&mois)

Ces 2 modèles étant linéaires, un petit ANN a été mis aussi en place pour comparer les résultats entre les modèles linéaires et un modèle non linéaire. L'objectif étant juste de valider le fait qu'un modèle linéaire est acceptable pour cette tâche. Si les MAE sont très différentes, cela mettra en relief le fait que le modèle linéaire n'est pas judicieux. Dans le cas de l'évaluation on trouve :



Les paramètres ont tous été gardés par défaut. On remarque notamment que l'ANN performe aussi bien avec les 2 datasets (il est très légèrement meilleur avec le modèle semaine mais c'est minime). Concernant le SGDR, il performe mieux sur le modèle par semaine. Par contre, Adaboost performe moins bien avec le format semaine. Cela est dû au fait que le modèle n'étant pas optimiser via un Grid Search, les erreurs ont été amplifiées avec la profondeur de Adaboost. Du coup, le choix a été fait d'optimiser pour les 2 datasets les modèles. Les résultats sont très légèrement meilleurs avec le format semaine.

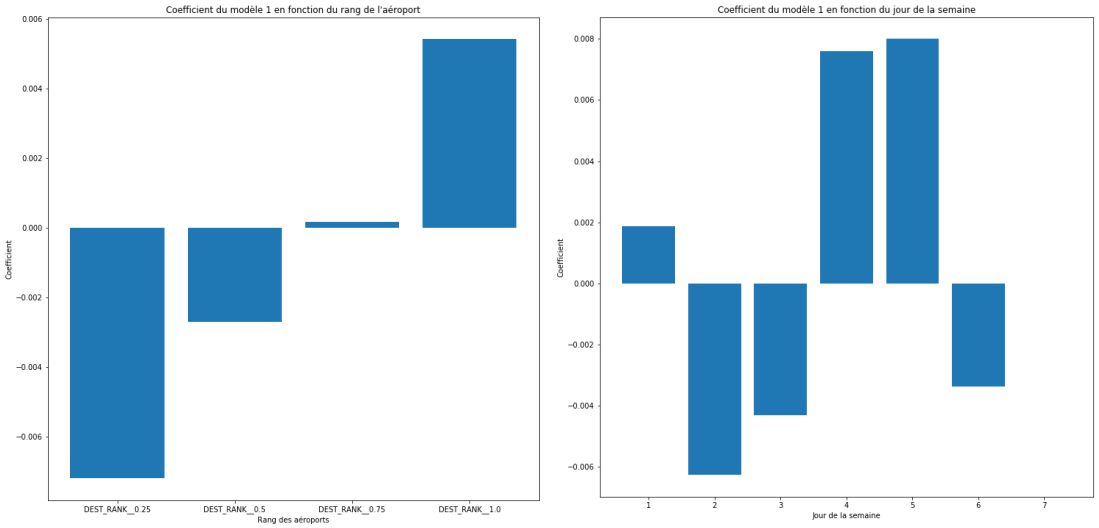
Une fois le modèle fixé et optimisé, j'ai voulu regarder les prédictions. Celles-ci étaient toutes négatives (entre -8 et -3 minutes). Partant de cela, on peut rechercher la cause de cette mauvaise prédiction. Celle-ci se trouve au niveau de la répartition des retards. N'ayant rien agrégé, le bruit rend le modèle "instable". En effet au même moment, il peut y avoir un retard important pour un vol et très négative pour un autre. Du coup le modèle se place au milieu pour minimiser les erreurs.

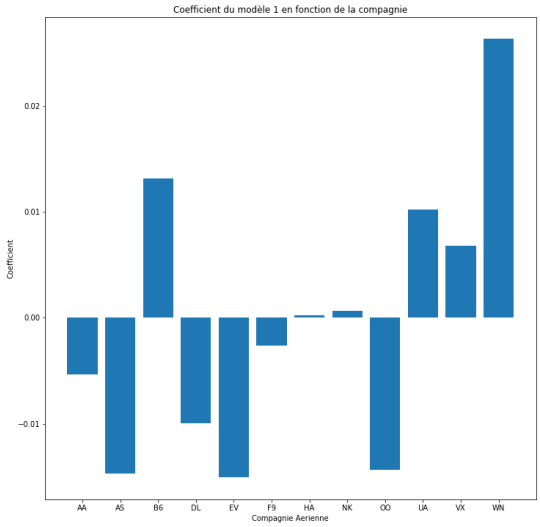


Une des solutions dans ce cas serait de considérer que tous les vols en avances, ne sont donc pas en retard. Dans ce cas, on peut placer le délai à 0.

Après la même optimisation de ce modèle, les délais étaient tous entre 0s et 1 min dans le pire des cas (semaine 52 avec la pire compagnie et aéroport de type 4).

L'analyse du modèle m'a permis de me rendre compte que sans agrégation, le "bruit" est tellement grand (il peut y avoir à la même heure des vols en avance ou en retard) que l'ensemble des facteurs étaient proche de 0:





On remarque que les facteurs évoluent et pénalisent plus ou moins les jours, compagnies ou aéroports mais les facteurs sont tous de l'ordre du centième. A partir de cette conclusion, l'idée de l'agrégation par heure avec l'ajout du nombre de vol a été mise en place.

Modèle 2:

Comme expliqué dans le Nettoyage, ce second dataset utilise la moyenne de retard par semaine/jour/compagnie et aéroport de départ. L'aéroport d'arrivé n'est plus pris en compte car sinon il n'y a qu'un seul vol utilisant cette ensemble de clé. Du coup cela revient au modèle précédent. De plus, il semble logique que le retard à un aéroport ne soit pas dû à l'aéroport d'arrivé.

Un second dataset a donc été refait et les mêmes modèles entrainés dessus avec aussi un Grid Search.

A première vue, ce modèle ne devrait pas mieux fonctionner car le MAE est passé de 10.2 min à près de 11.2 minutes. Cependant le dataset étant différent, ceux-ci ne sont en fait pas vraiment comparables. Dans ce cas, nous verrons dans la partie interprétation que le modèle n'est pas si mauvais.

Split du dataset

A première vue, le modèle étant temporel, l'évaluation du modèle ne peut pas se faire comme d'habitude. Le retard à un instant T étant lié au retard à l'instant T-1, l'évaluation devrait se faire sur une suite continue de points (par exemple sur tout le mois de décembre).

N'ayant qu'une année de données et les mois étant très différents en termes de retard, on ne peut pas se permettre d'entrainer le modèle sur 11 mois et l'évaluer sur décembre. La prédiction serait complètement fausse.

Cependant comme le modèle est linéaire, on peut se permettre d'utiliser des points pris au hasard dans le dataset car on n'utilise pas les points précédents pour prédire le retard à l'instant T.

Du coup le split est fait comme d'habitude avec le split_train_test de Scikit-Learn mais il faut garder en tête que cela n'est pas toujours possible (nous verrons cela dans la partie "Pistes d'évolutions".

Interprétation

Visualisation de la prédiction

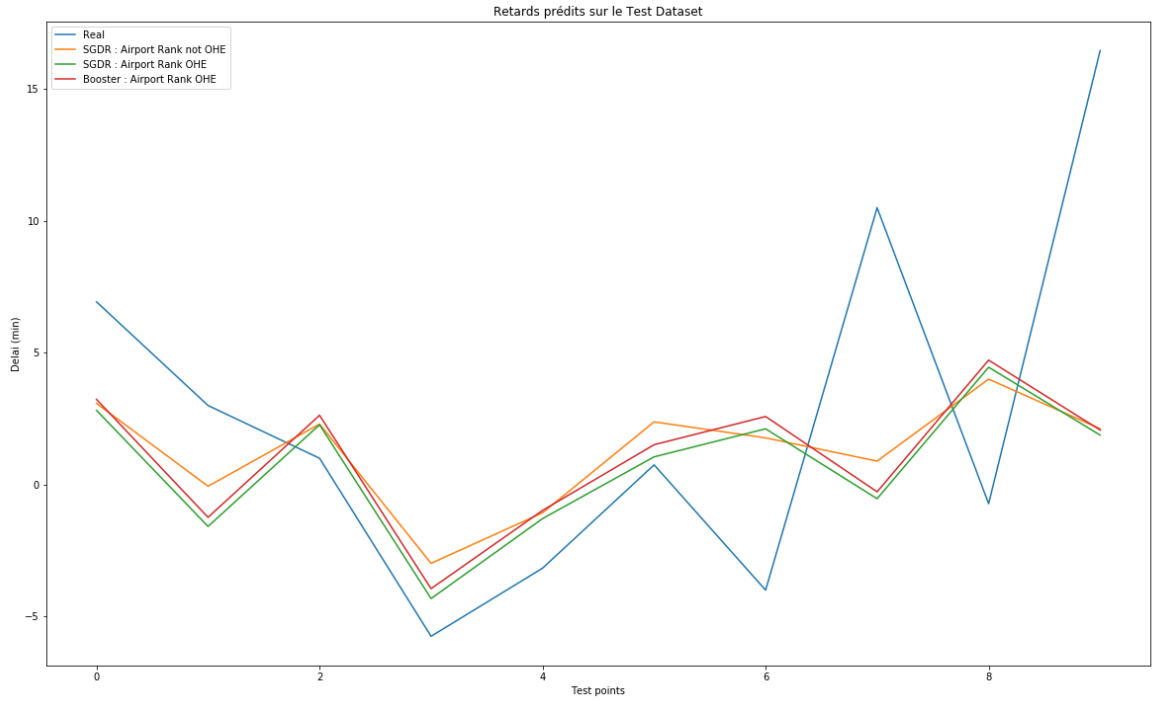
Pour l'interprétation, nous allons donc nous focaliser sur le modèle 2. Sur celui-ci, un test a été fait pour voir s'il faut mieux garder les rangs des aéroports dans 1 seule colonne ou passer sur un OHE.

Si on regarde la MAE/MSE pour ces différents modèles on trouve :

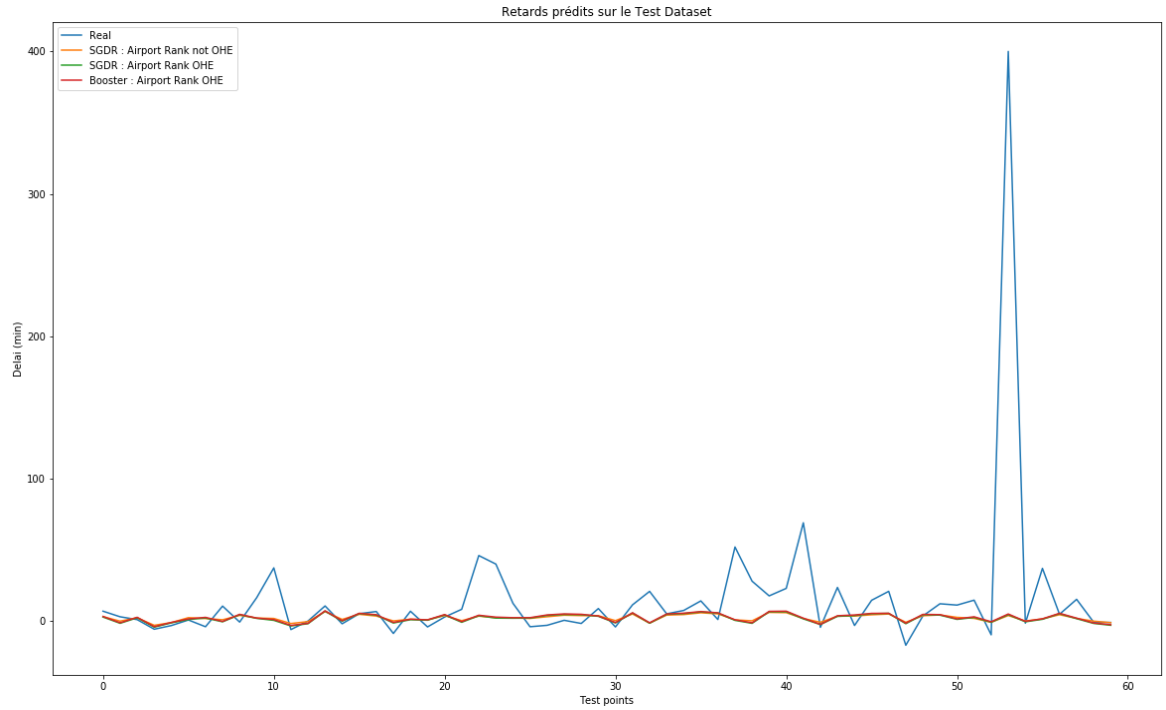
- SGDR avec rang non OHE :
  - MSE 652.6672 - MAE 11.3115
- SGDR avec rang OHE :
  - MSE 652.2052 - MAE 11.2355
- Boosting avec rang OHE
  - MSE 648.2626 - MAE 11.2342

On remarque que le Boosting n'a pas particulièrement d'impact sur le MAE mais impact un peu la MSE. Ce modèle sera donc utilisé sur l'API pour prédire les retards.

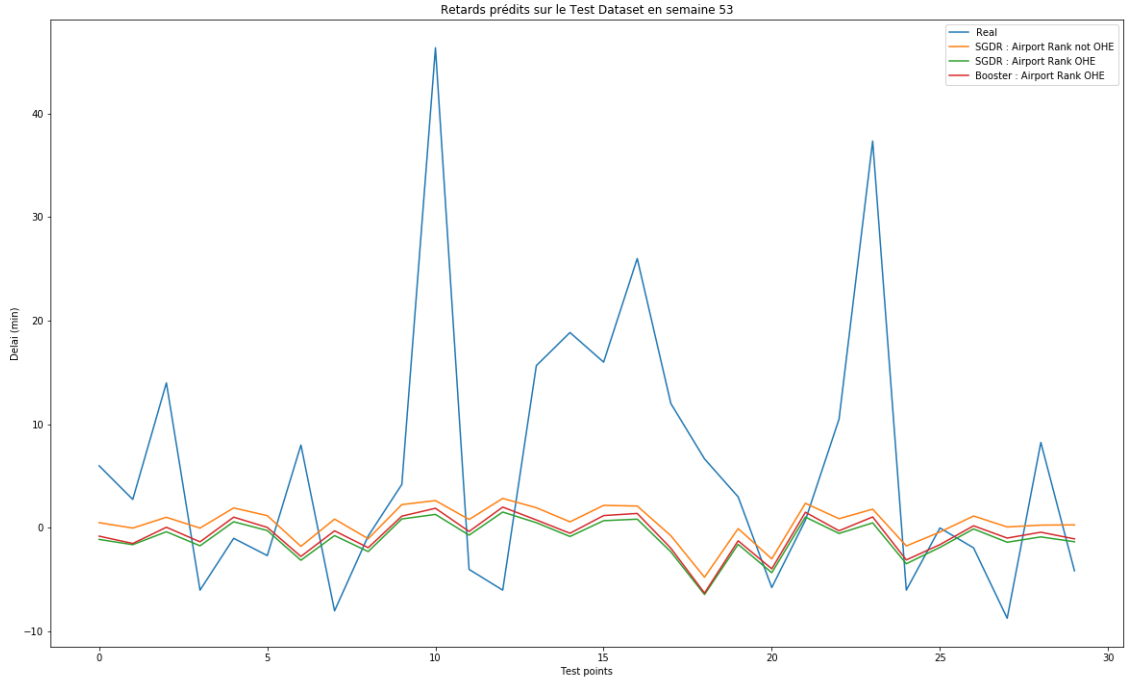
Cependant, afin d'éviter des prédictions erronées comme le modèle 1, on peut regarder la prédiction de chaque modèle par rapport au test dataset. On trouve :



On remarque que la tendance est assez bien suivi "à court terme" On retrouve les tendances montantes/descendantes. Par contre si on dézoome par exemple sur 60 points on a:



On remarque toujours que la "tendance" est suivie mais par contre les gros retard sont mal prédits (celle-ci est écrasée). On a vu aussi qu'en semaine 52, on a de grosses variations. Si on affiche la prédiction lors de cette semaine on a:

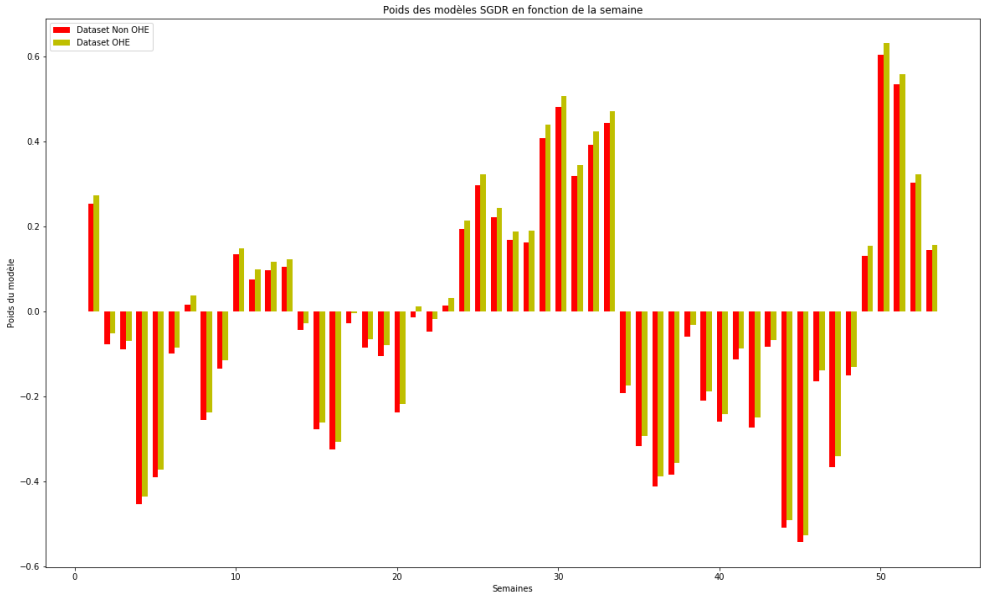


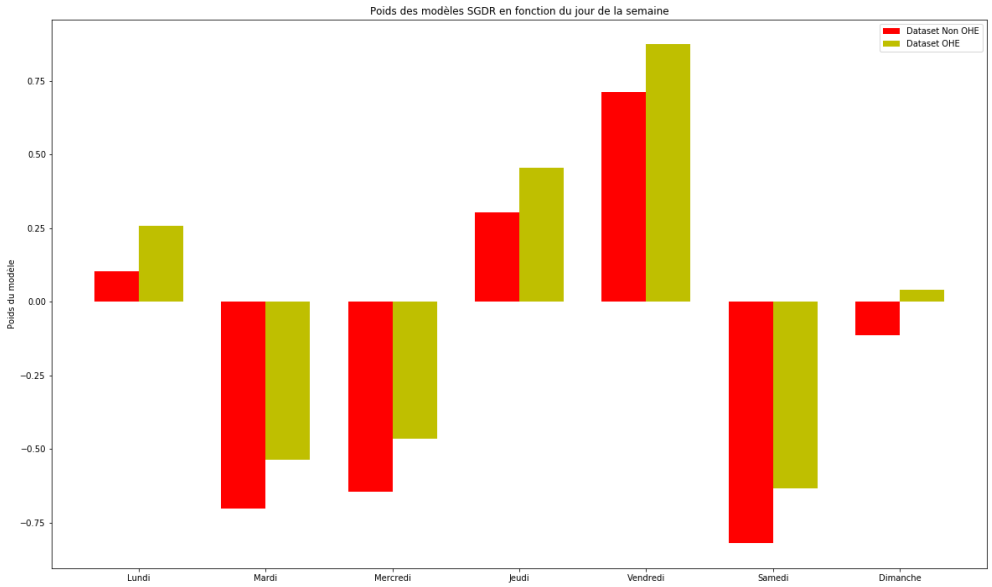
Une fois de plus la tendance est suivi mais de manière écrasée.

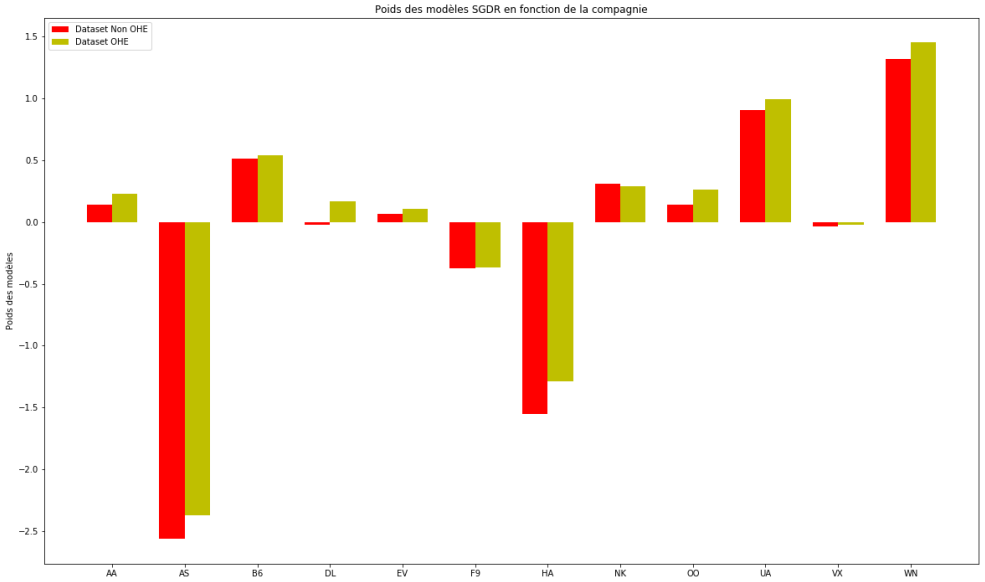
Visualisation des coefficients

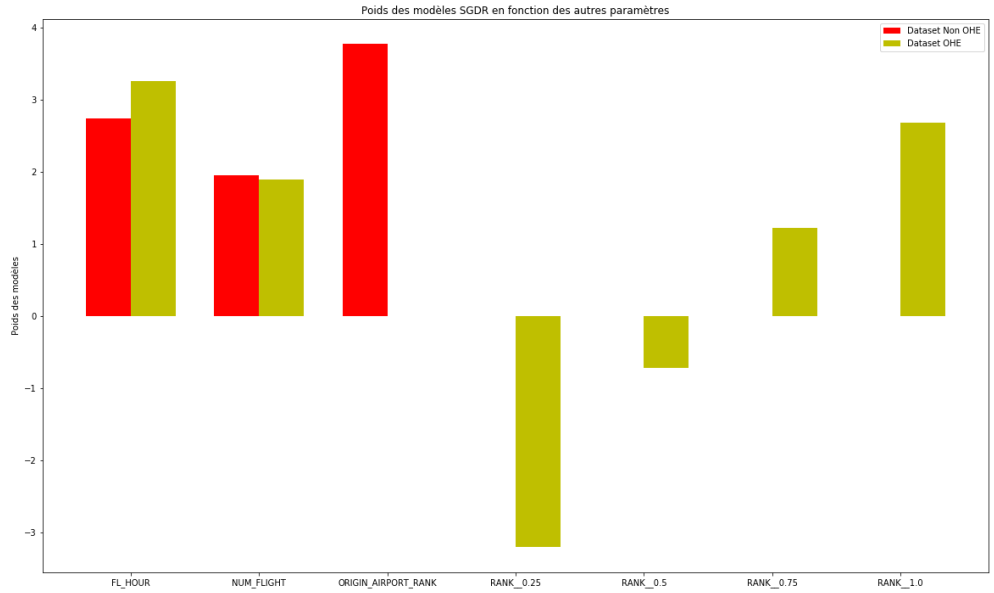
SGDR avec et sans OHE

De la même manière que l'on a regardé les coefficients sur le modèle 1, on peut faire de même et regarder les poids des 2 modèles SGDR avec et sans OHE et on trouve :









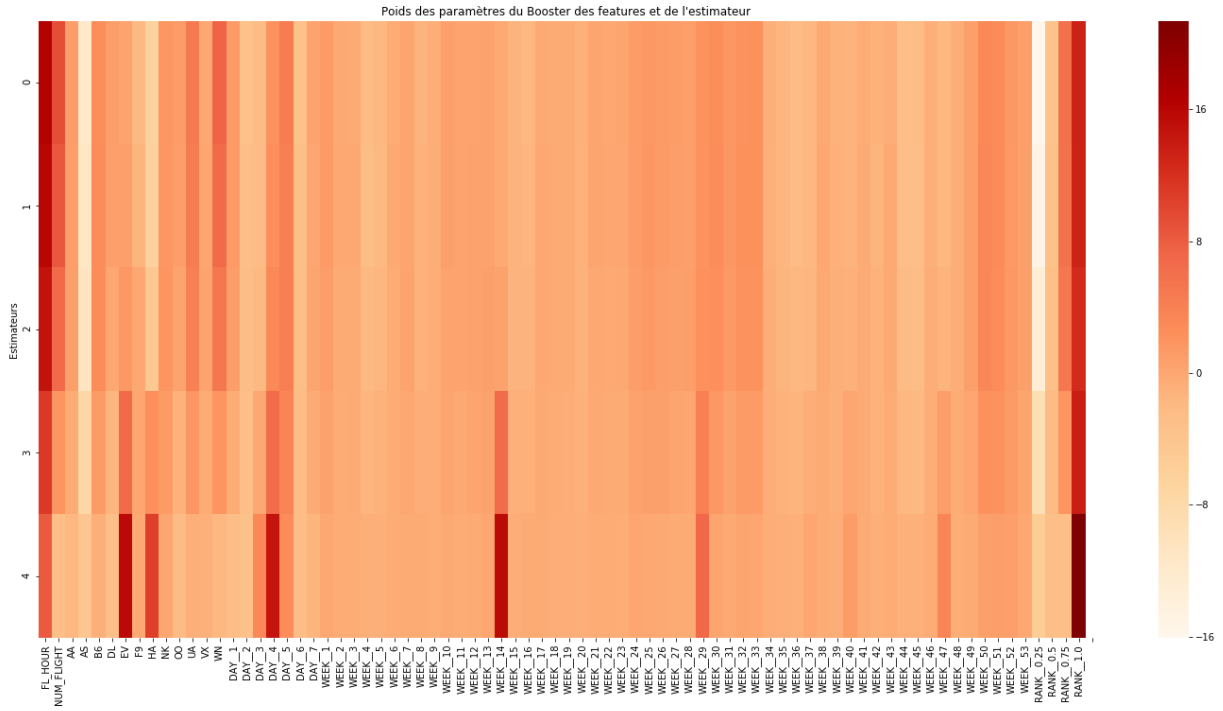
On remarque déjà que les coefficients sont beaucoup plus grands (de l'ordre de l'unité). De ce fait, par exemple un vendredi, le retard sera pénalisé de 0.75 minutes et le mercredi avancés de 0.5 minutes.

On remarque aussi que les facteurs sont très assez proches pour le modèle OHE ou non. Le OHE à un offset positif par rapport au non OHE (les facteurs négatifs sont moins négatifs et les positifs sont plus positifs).

Concernant les rangs, le modèle non OHE applique une pénalité de 1 par rang (4 rangs par unité et le facteur est 4). Pour la version OHE, les facteurs sont plus grands (environ 2 minutes de pénalité par rang)

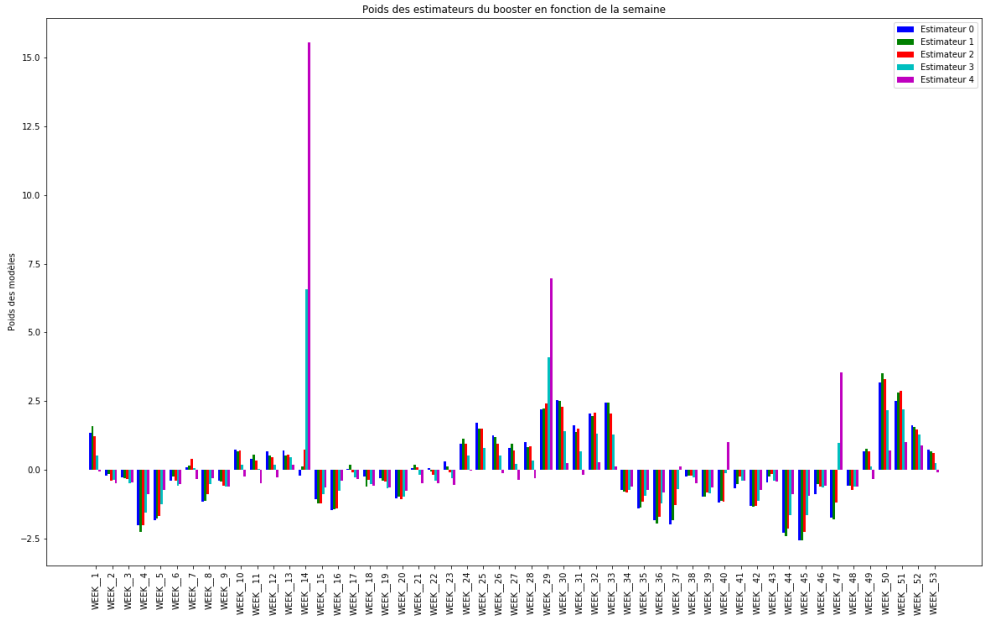
Modèle Adaboost

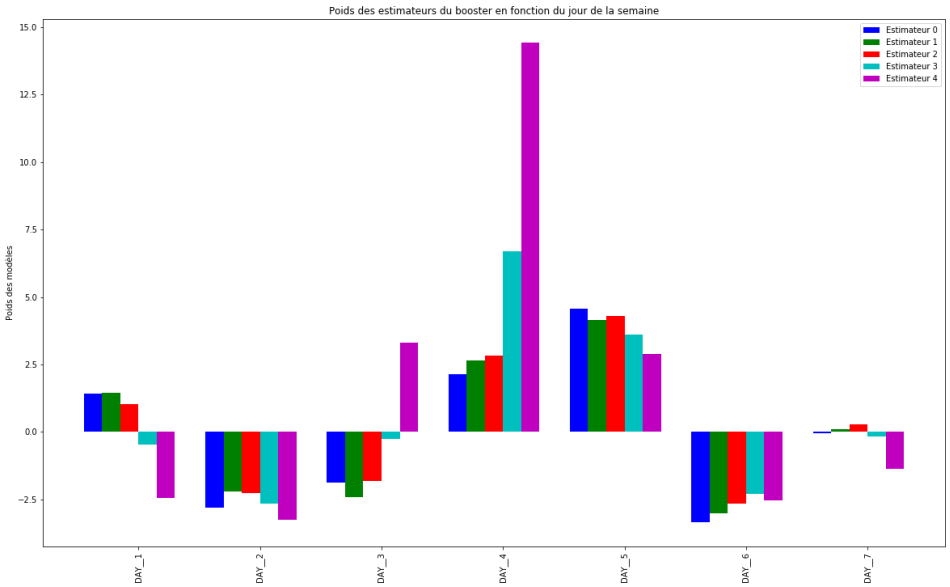
Cette analyse a été faite sur le SGDR, cependant notre meilleur modèle est basé sur le Boosting qui va entrainer 5 modèles de régression sur l'erreur du précédent. Si on regarde les coefficients des modèles on trouve:

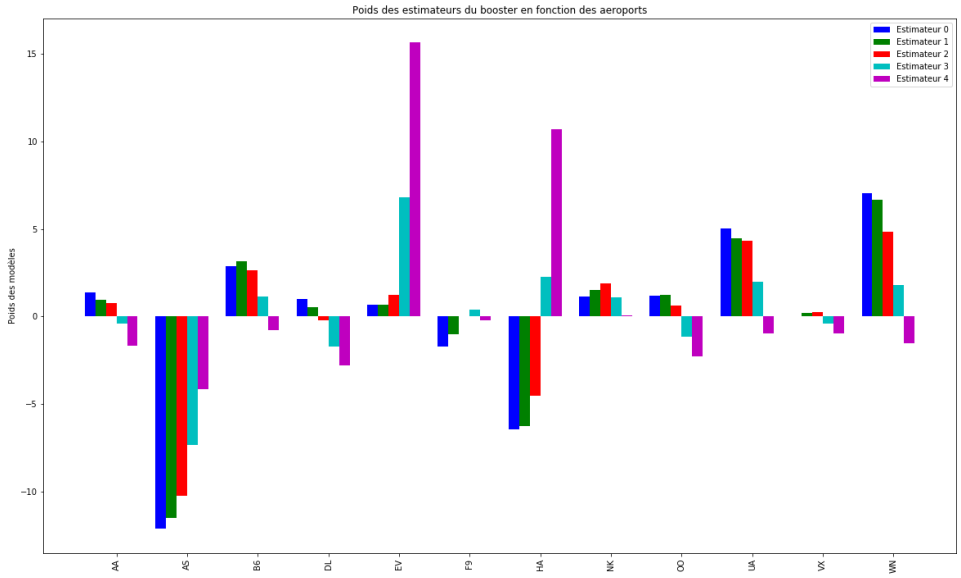


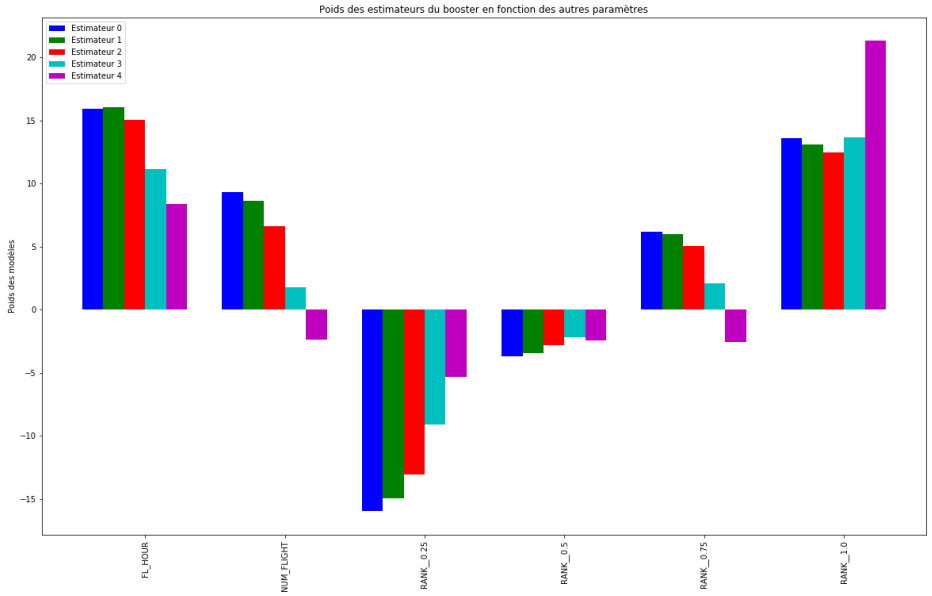
On peut de cette façon de visualiser ou le dataset est bruité (les gros coefficients pour des estimateurs "profonds") et les features importantes (heure, compagnie, rang). Bizarrement, le coefficient est faible au niveau de la semaine 53…

On peut aussi visualiser les coefficients pour chaque estimateur avec les barplots et on trouve:




Poids des estimateurs du booster en fonction du jour de la semaine
Estimateur 0
Estimateur 1
Estimateur 2
Estimateur 3
Estimateur 4
Poids des modèles
DAY_1
DAY_2
DAY_3
DAY_4
DAY_5
DAY_6
DAY_7


Poids des estimateurs du booster en fonction des aeroports
Estimateur 0
Estimateur 1
Estimateur 2
Estimateur 3
Estimateur 4
Poids des modèles
AA
AS
B6
DL
EV
F9
HA
NK
OO
UA
VX
WN


Poids des estimateurs du booster en fonction des autres paramètres
Estimateur 0
Estimateur 1
Estimateur 2
Estimateur 3
Estimateur 4
Poids des modèles
FL_HOUR
NUM_FLIGHT
RANK_0.25
RANK_0.5
RANK_0.75
RANK_1.0

Sur l'estimateur 0 (bleu), on retrouve les mêmes tendances:

- Facteur plus important en été et vers noël
- Facteur positifs le Lundi/jeudi et vendredi
- Pseudo linéarité entre les rangs

Quant aux autres, ils sont généralement plus faibles sauf pour certaines features bruitées (Compagnie EV – Date du Jeudi, Semaine 14/29/47).

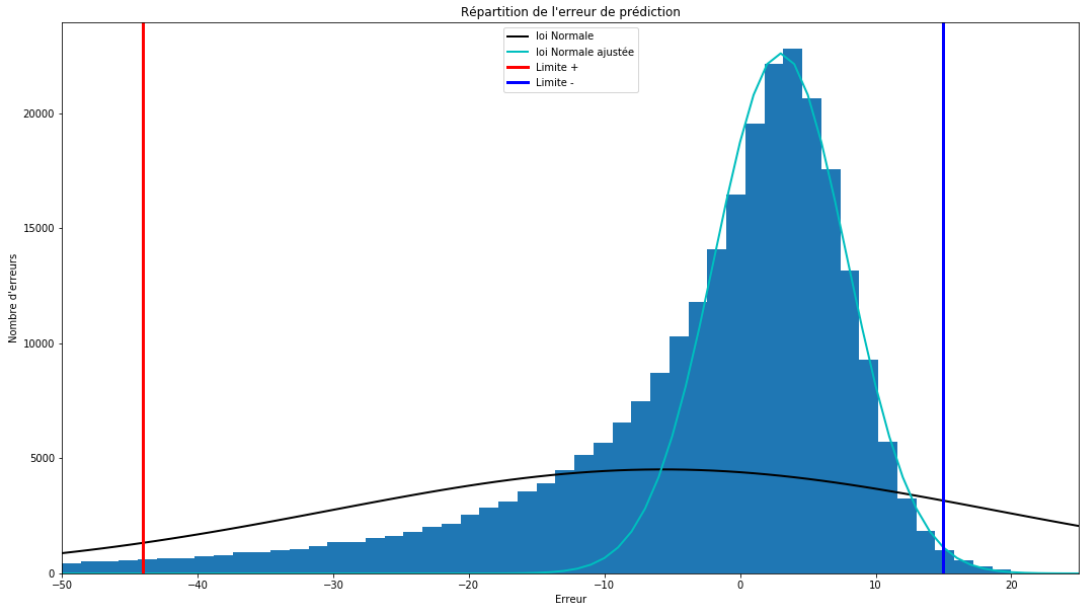
Basé sur les métriques sur le test dataset, le Boosting va être utilisé sur l'API afin d'estimer le retard.

Analyse des résidus

Une des analyses que l'on peut faire est sur l'analyse de l'erreur de prédiction. Sur un histogramme de la répartition de l'erreur, on peut voir qu'une grande partie de l'erreur négative. On prédit donc très souvent en dessous du retard prévue et parfois de beaucoup. De plus on remarque que l'on prédit très peu de fois beaucoup au-dessus. Ce qui est logique car un avion peut partir 10h en retard mais rarement 1h en avance.

De ce fait on peut difficilement faire fitter la loi normale sur cette erreur. En utilisant la Standard Déviation et la moyenne de l'erreur sur la loi Normale, on trouve la courbe en noir ci-dessous. Si on ajuste au plus juste la loi Normale sur le pic de retard (courbe cyan), on se rend compte que l'on ignorera beaucoup de points. Du coup, afin de prédire un intervalle de confiance, il faut regarde le nombre de points dans un certains range par rapport au nombre de point total. L'objectif étant d'avoir le range le plus petit permettant d'avoir n% des points inclus.

Les limites sont en rouge et bleu ci-dessous.



On a donc un range très grand pour avoir une prédiction à 95%. Celui-ci va de -44 à +15 minutes. De ce fait, ce modèle n'est pas extrêmement précis pour donner une prédiction. On peut aussi voir ça sur la valeur du coefficient de détermination r^2 qui n'est que de 0.06 (uniquement 6% de la variance est expliquée)

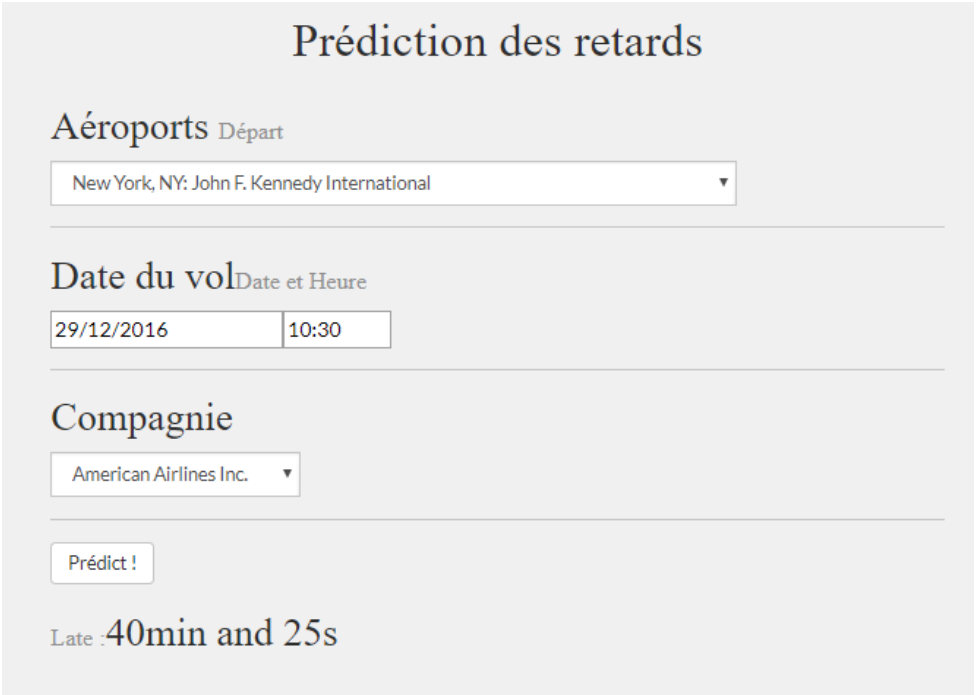
API

L'API a été mise en place sur Flask. L'utilisateur doit fournir l'aéroport de départ, la date et l'heure du vol ainsi que la compagnie. Ces informations sont envoyées via une requête POST.

Celui-ci encode ensuite cet input au format du dataset de training OHE du modèle 2. Les données sont scalées de la même manière. Le modèle sauvegardé prédit ensuite le retard qui est retourné sur la page de l'utilisateur.

Comme vu précédemment, le modèle utilisé est Adaboost avec les paramètres ayant minimisé la MAE. Le serveur possède aussi quelques fichiers pickle contenant le nom des aéroports et compagnies afin d'avoir une interface conviviale.

Pour ce qui est du nombre de vols, une moyenne par aéroport et par heure est utilisée car celui-ci est très corrélé lié à l'aéroport.



L'API est disponible à cette adresse :

http://coni57.pythonanywhere.com/p4/

Pistes d'évolutions

Modèle non-linéaire

On a remarqué lors de l'interprétation que le modèle écrase un peu le retard mais garde la "tendance". Cela est dû au fait que le modèle linéaire est en underfitting. Certains paramètres ne sont pas linéairement dépendants mais ont parfois avec des "features interactions" plus complexes. Par exemple, le retard actuel peut-être prédit comme suit (à cause du OHE):

$$Retard = \alpha * heure + \beta_{jour} + \gamma_{semaine} + \delta_{compagnie} + \varepsilon_{aeroport} + \theta * nb_{vols}$$

Or on peut imaginer que le retard est plutôt basé sur un produit de "pénalité". Par exemple le retard du mercredi de la semaine 52 n'est pas égal à la moyenne des retards de la semaine 52 plus la moyenne des retards du mercredi mais plutôt un facteur pénalisant du mercredi multiplié par un facteur pénalisant de la semaine 52. Cependant, ce type de modèle n'est pas possible de manière linéaire car le nombre de dimensions exploserait. C'est ce que permet un ANN avec tout leur ensemble de perceptrons.

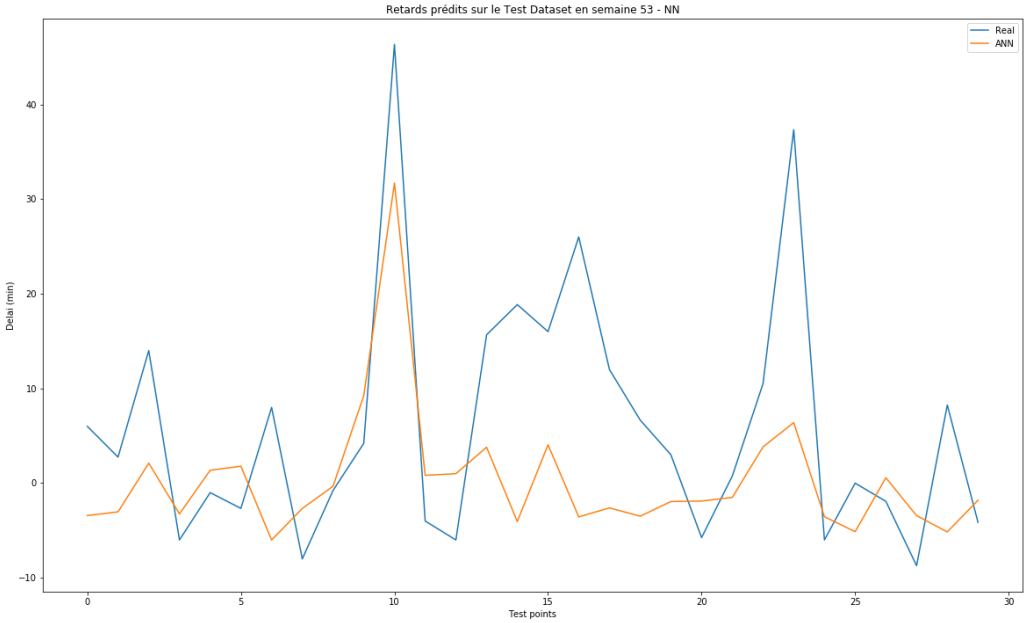
Pour évaluer l'intérêt de la non-linéarité dans la prédiction, on peut utiliser ce même dataset avec un simple ANN (sans recherches particulières d'optimisations) et on trouve en résultat :

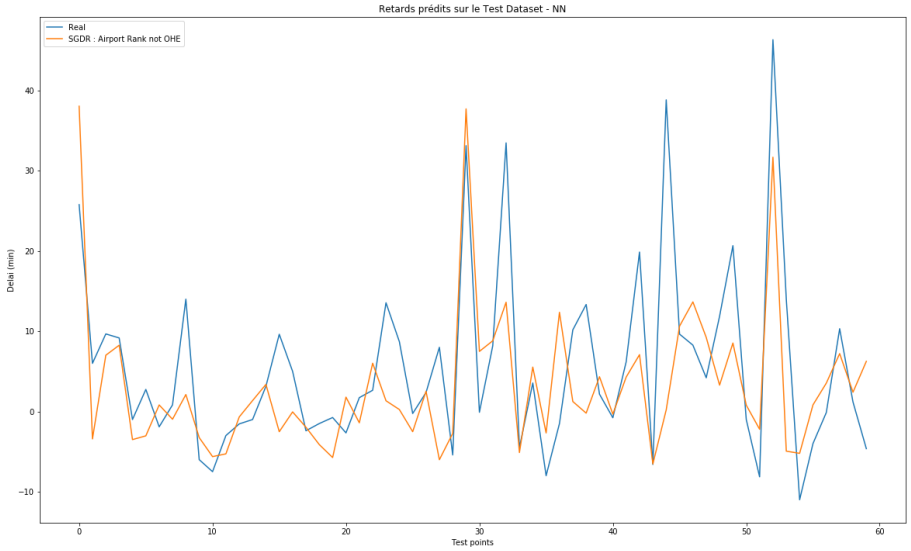
- MAE: 8.2088 - MSE: 467.2681

Si on le compare au Boosting qui était :

- MSE 648.2626 - MAE 11.2342

De la même manière, on peut aussi afficher la prédiction par rapport au test set et on trouve :



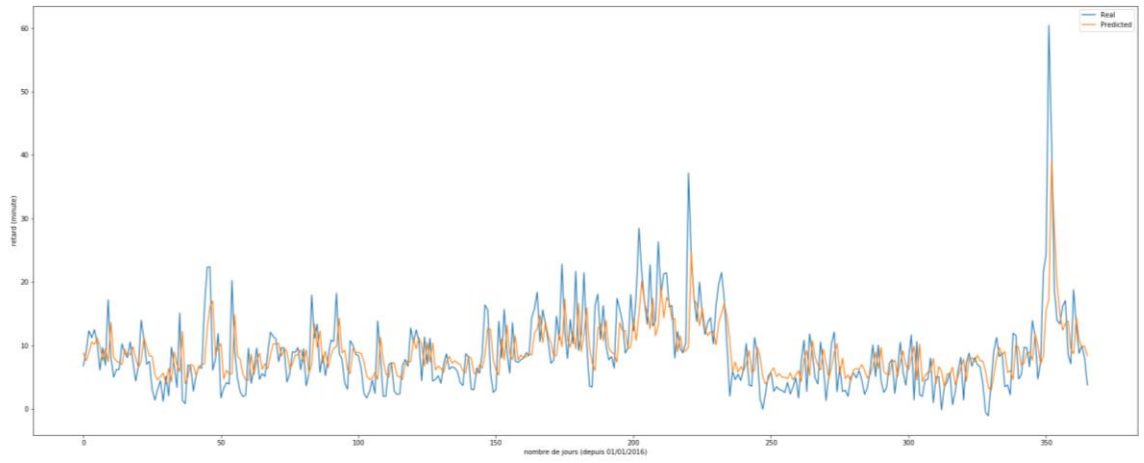


On remarque que la tendance est bien mieux suivie ainsi que l'échelle. Cela prouve que le modèle linéaire underfitting.

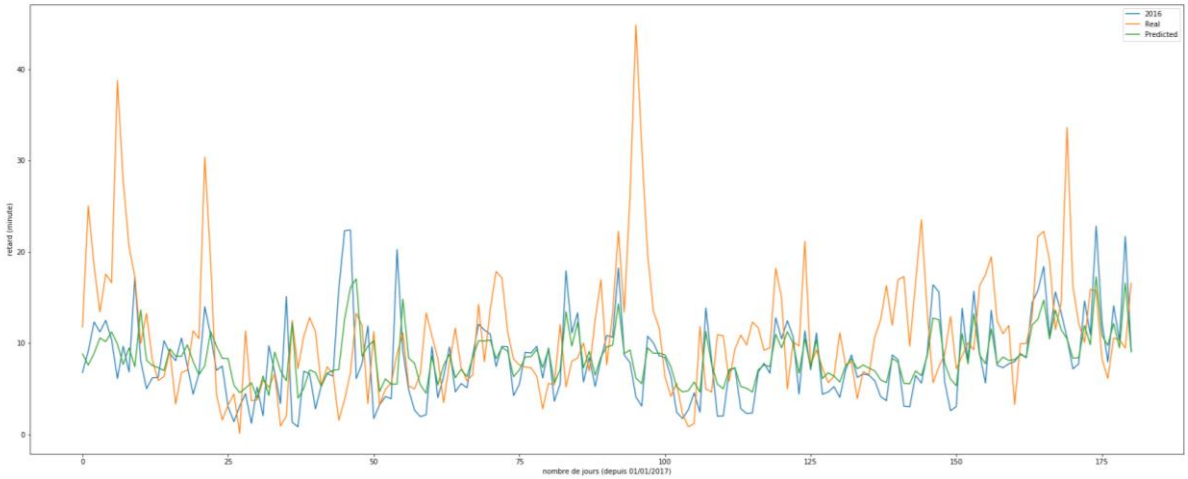
Modèles temporels

Une autre façon de prédire la tendance est d'utiliser un modèle utilisant les données temporelles avec pour objectif d'extraire des tendances par un phénomène similaire au Boosting. On pense notamment au ARIMA ou fbProphet qui utilisent une moyenne glissante afin d'extraire une certaines tendances. A l'instar du Boosting, cette tendance est soustraite pour rendre visible une seconde sous-tendance et ce jusqu'à qu'il ne reste qu'un bruit ambiant. Le problème de ce modèle est qu'il n'est basé que sur le temps.

Par exemple, si on utilise ARIMA avec un Grid Search sur ses 3 paramètres pour réduire la MAE, on trouve la prédiction suivante :



Le problème de ce modèle, en plus de ne pas utiliser d'autres données que le temps/retard, est qu'il ne permet pas de prédire une tendance sur plus de 7j. De plus, on ne peut pas utiliser des points aléatoires du dataset comme fait avec le modèle linéaire car celui-ci utilise des données récurrentes. Par contre à titre de comparaison on peut superposer la prédiction de 2016 sur le dataset 2017 (bien que hors sujet) pour regarder si la tendance se répète.



On peut voir que certains pics se retrouvent encore (par exemple vers les jours 80 ou 170). Par contre à d'autres moments, cela colle moins bien (par exemple les 10 premiers jours ou le pic autour du jour 95).

RNN

Un des derniers modèles possibles serait un Récurrent Neural Network (type LSTM/GRU). Ce réseau aurait en entrée la date, le nombre de vols, la compagnie etc… En fournissant lors du training l'année 2016, il serait peut-être possible d'avoir un modèle un peu plus précis pour l'année 2017. Tout comme le modèle précédent, on ne peut pas utiliser des données prises aléatoirement dans le dataset car le retard des timesteps précédentes impacte le retard suivant prédit. De ce fait l'évaluation devrait se faire sur le dataset 2017. De par l'ensemble de données utilisées, celui-ci serait plus précis que la régression linéaire car comme tout ANN, celui-ci est capable d'apprendre à partir de données très non linéaires et il serait plus précis qu'ARIMA car il prendrait en considérations de multiples inputs au lieu d'uniquement le temps.

Conclusion

En conclusion, nous avons vu lors de ce projet comme mettre en place et optimiser un modèle linéaire sur des données temporelles. La difficulté majeure de ce dataset était la mémoire parfois limitante et le problème d'underfitting d'un modèle linéaire sur ce cas plutôt non linéaire.

Au final, nous avons pu remarquer que le modèle 2 permet de suivre la tendance globale mais écrase les retards. Certaines pistes d'évolutions existent ainsi que d'autres modèles plus souhaitables pour ce type de données temporelles.

Au niveau des résultats, on arrive à avoir un MAE d'environ 10 minutes. Ceci, n'est pas particulièrement mauvais pas très bon non plus car le retard moyen sur le dataset est de 8 min 20 avec une standard déviation de 24 min.