MINI PROJET « SÉRIES TEMPORELLES ET MACHINE LEARNING »

Clément BOUDOU et Baptiste BONTOUX

Prédiction de courses Uber à New York City

Sujet de l'étude

Uber est une entreprise de transport privée créée en 2009 et qui a connu une croissance phénoménale notamment en 2015 où elle a atteint une valorisation de plus de 50 milliards de dollars avec une présence dans 51 pays et 253 villes dans le monde, dont San Francisco, New York, Paris, Londres, Brasilia, Bogota, Le Caire, Tel Aviv, Johannesburg, Bombay, Pékin et Sydney. Dans le cadre de cette étude, nous nous limiterons aux données de la ville de New York. Pour cela nous disposons de données sur plus de 4,5 millions de courses Uber dans la ville de New York d'avril à septembre 2014, et 14,3 millions de courses supplémentaires de janvier à juin 2015.
Ces dernières ont été récupérées par FiveThirtyEight qui est une équipe de journalistes spécialisés dans l'analyse de données. L'ensemble des données leur ont été transmises par la NYC Taxi & Limousine Commission (TLC) avec des données sur de nombreuses autres entreprises de transports privées ainsi que les compagnies de taxi. Comme toutes les données utilisées pour leurs études, on peut y accéder à partir d'un répertoire GitHub https://github.com/fivethirtyeight/uber-tlc-foil-response.

Description des données utilisées

Plus précisément, nous n'avons utilisé que les données propres à Uber (uber-trip-data) qui sont séparées en de nombreux fichiers qu'il a fallu fusionner dans un premier temps.
Dans les fichiers bruts, on dispose des features suivantes :

- Date et heure de la course
- Position de la commande
    - 2014 : Latitude et longitude
    - 2015 : Un ID correspondant à une zone géographique
- La base liée au Uber

Cependant un pré-processing des données a été effectué afin de regrouper les courses dans des intervalles de temps plus grands (10 minutes et 1h). Les features utilisées deviennent donc :

- Date et heure de début de l'intervalle
- Pour chaque base Uber le nombre de courses ayant été effectuées pendant ce créneau de temps.

Un autre pré-processing a été effectué pour effectuer des prédictions sur l'emplacement de la course plutôt que la base d'origine. Cette transformation n'est possible que sur les données de 2015 qui possèdent un ID correspondant à une zone géographique.

- Date et heure de début de l'intervalle
- Pour chaque ID le nombre de courses ayant été demandé dans cette zone.

Ces données sont disponibles dans l'archive fournie avec ce rapport dans cinq fichiers distincts :

- 10min_main_bases_2015.csv : Les 6 bases principales, en 2015 toutes les 10 minutes (26063 lignes, 7 colonnes)
- 10min_main_locations_15.csv : Les 20 zones avec le plus de commandes, en 2015 toutes les 10 minutes (26063 lignes, 21 colonnes)
- hourly_main_bases_2015.csv : Les 6 bases principales, en 2015, toutes les heures (4344 lignes, 7 colonnes)
- hourly_main_bases_all.csv : Les 6 bases principales, sur toutes les données , toutes les heures (10944 lignes, 7 colonnes)
- hourly_main_locations_15.csv : Les 20 zones avec le plus de commandes, en 2015 toutes les heures (4344 lignes, 21 colonnes)

Voici ce à quoi ressemble par exemple le data set final des bases (hourly_main_bases_all.csv), utilisable par nos modèles :

	Time	Base_B02512	Base_B02598	Base_B02617	Base_B02682	Base_B02764	Base_B02765
0	2014-04-01 00:00:00	9.0	35.0	29.0	60.0	5.0	0.0
1	2014-04-01 01:00:00	4.0	18.0	18.0	22.0	4.0	0.0
2	2014-04-01 02:00:00	4.0	15.0	9.0	24.0	1.0	0.0
3	2014-04-01 03:00:00	4.0	24.0	18.0	46.0	1.0	0.0
4	2014-04-01 04:00:00	9.0	45.0	33.0	73.0	6.0	0.0
...	...	...	...	...	...	...	...
10939	2015-06-30 19:00:00	85.0	386.0	598.0	2221.0	1182.0	432.0
10940	2015-06-30 20:00:00	83.0	350.0	545.0	2114.0	1104.0	447.0
10941	2015-06-30 21:00:00	96.0	384.0	622.0	2309.0	1264.0	487.0
10942	2015-06-30 22:00:00	74.0	378.0	595.0	2218.0	1246.0	467.0
10943	2015-06-30 23:00:00	50.0	295.0	501.0	1758.0	1072.0	359.0

10944 rows × 7 columns

Les données des “location” (zone géographique de la commande) sont formatées de la même manière.

	Time	Location_48	Location_68	Location_79	Location_107	Location_132	Location_138	Location_144	Location_148	Location_158	...
0	2015-01-01 00:00:00	130.0	133.0	273.0	128.0	18.0	2.0	120.0	203.0	207.0	...
1	2015-01-01 01:00:00	112.0	206.0	285.0	117.0	0.0	0.0	154.0	224.0	191.0	...
2	2015-01-01 02:00:00	199.0	290.0	417.0	156.0	0.0	0.0	193.0	301.0	234.0	...
3	2015-01-01 03:00:00	132.0	226.0	308.0	96.0	1.0	0.0	154.0	230.0	184.0	...
4	2015-01-01	85.0	138.0	155.0	49.0	0.0	0.0	49.0	103.0	92.0	

Questions étudiées

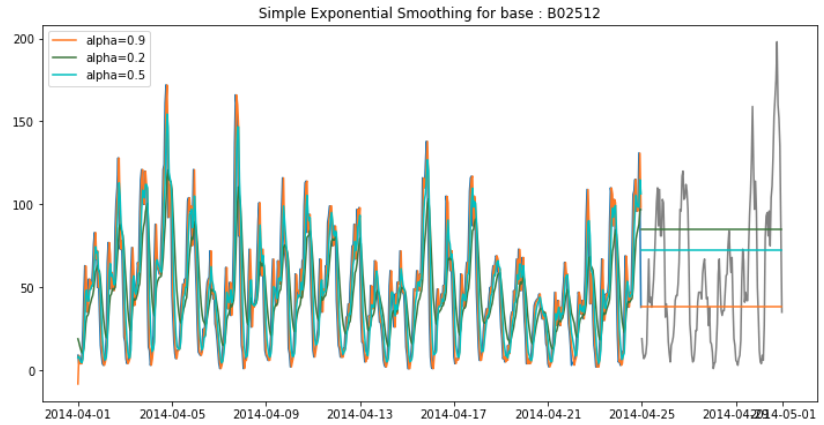
Globalement, la question que nous nous posons est : Est-ce qu’il est possible de prédire de façon fiable les prochaines courses Uber afin d’anticiper les demandes des clients ?
Il y a deux angles sous lesquels nous allons étudier cette question. D’abord en prédisant la sollicitation des différentes bases Uber. Dans un cas concret, ceci permettrait à Uber de savoir s' il faut associer plus de conducteurs à telle ou telle base si on prédit une forte affluence qui arrive.
Ensuite, comme nous disposons d’une indication sur la zone où se situe la personne ayant commandé une course, nous allons prédire le nombre de personnes qui vont avoir besoin d’une course dans cette zone. Pour Uber, cela permettrait d’envoyer des chauffeurs en avance à ces emplacements afin de réagir le plus rapidement possible à chaque commande.

Méthodes utilisées et résultats obtenus

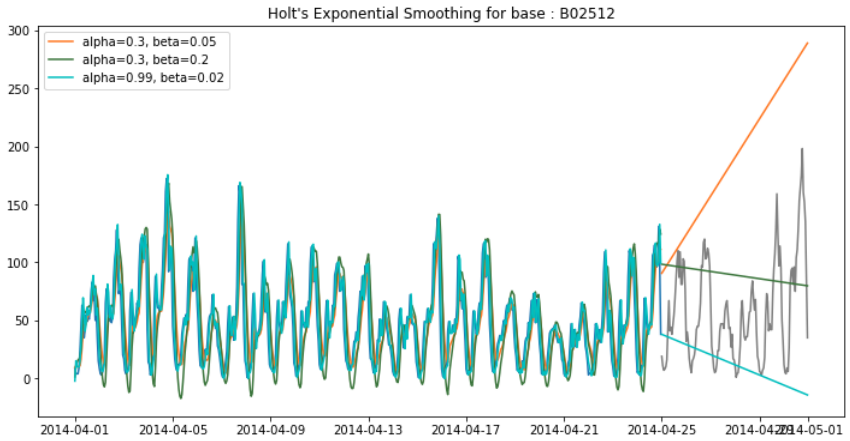
Approche naïve

Dans cette approche naïve nous voulons tester des méthodes de prévisions assez simples, nous commençons donc avec un lissage exponentiel simple et un lissage de Holt. On utilise ces méthodes pour tenter de prévoir le nombre de commandes par base et par heure avec à chaque fois 3 alpha différents. Les 2 schémas suivants montrent la base B02512 sur une période de 1 mois avec les données de test qui commence vers le 25/04.

Lissage exponentiel



Lissage de Holt

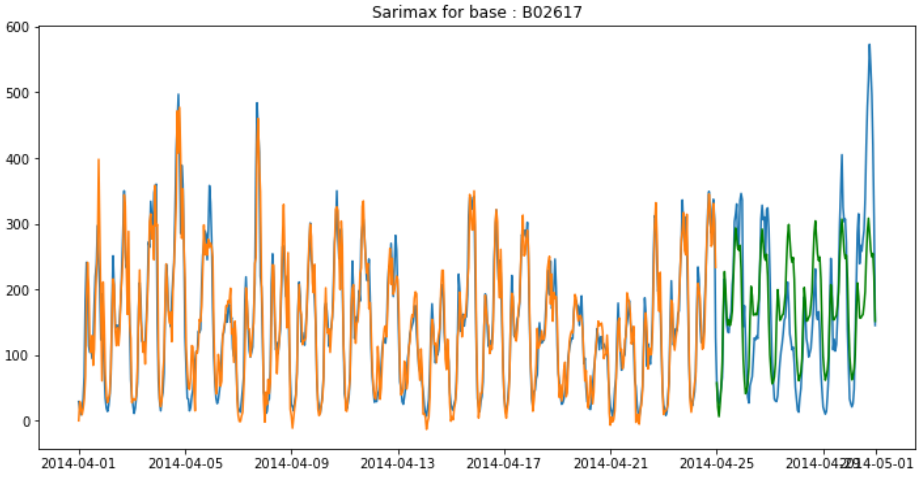


Comme on pouvait s'y attendre, ces deux techniques ne montrent pas de bons résultats. Quand on rentre dans la zone des données de test, il doivent prédire et ne montre qu'une tendance de ce qui arrive après, et encore, ce n'est pas une tendance assez précise car les tendances et les saisonnalités de nos données sont difficiles à prédire.

Approche statistique

Dans cette nouvelle approche on veut un modèle qui comprend mieux la saisonnalité de nos données, pour cela on va utiliser le modèle SARIMAX (Seasonal Autoregressive Integrated Moving Average with eXogenous factors) qui est une version améliorée du modèle SARIMA. On va essayer de prédire le nombre de commandes par heure et par base, la courbe bleue représente les vrais valeurs, la courbe orange représente les prédictions sur les données d'entraînement et la courbe verte représente les prédictions sur les données de test.

Sarimax



On comprend directement que ce modèle est bien plus précis que les lissages d'avant, on remarque que la courbe de prédiction ressemble bien plus à la vraie et la saisonnalité est bien mieux décrite. Néanmoins nos prédictions ne sont parfaites et s'adaptent mal aux gros pics, or ce sont ces pics importants qui sont déterminants pour la qualité de notre modèle.

Approche deep learning

Dans cette nouvelle approche, on veut tester les techniques de Deep Learning. Les modèles seront donc plus complexes et plus longs à entraîner mais on espère avoir de très bons résultats. On compte tester trois modèles distincts, premièrement un modèle de deep learning simple avec uniquement un réseau de neurones basique, deuxièmement un modèle récurrent appelé SimpleRNN et troisièmement un modèle à mémoire : Long Short Term Memory (LSTM). Ces deux derniers modèles sont des architectures de Deep Learning créées spécialement pour les séries temporelles. Pour chaque modèle, trois prédictions sont effectuées :

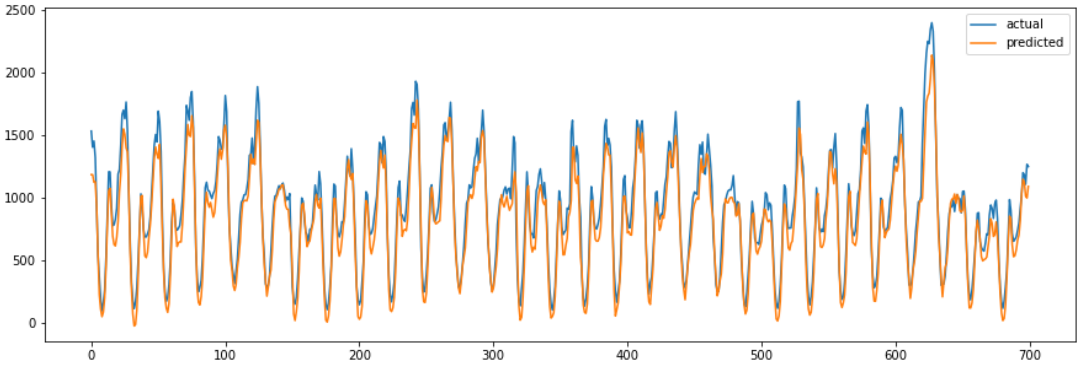
- Sollicitation d'une base Uber à t+1 (80% données d'entraînement, 20% pour le test soit environ 28 jours). On prédit chaque heure de la base de test à l'aide des vraies données qui précèdent.
- Sollicitation d'une base Uber à t+72 (72 dernières heures pour le test et le reste sert à l'entraînement). On se sert de nos prédictions pour prédire plus loin dans le temps. Autrement dit, on prédit les demandes 3 jours à l'avance.
- Demandes dans une certaine zone à t+1 (80% données d'entraînement, 20% pour le test soit environ 28 jours). On prédit chaque heure à l'aide des vraies données.

Dans les exemples ci-dessous, les prédictions sont faites pour la base B02764 ou la zone 79 mais il est tout à fait possible d'observer ce que le modèle prédit pour d'autres valeurs en modifiant le notebook *deep_learning.ipynb*. La fenêtre (loop_back) choisie (changeable dans le notebook) est d'une semaine (168h).

Modèle simple (Dense Layers)

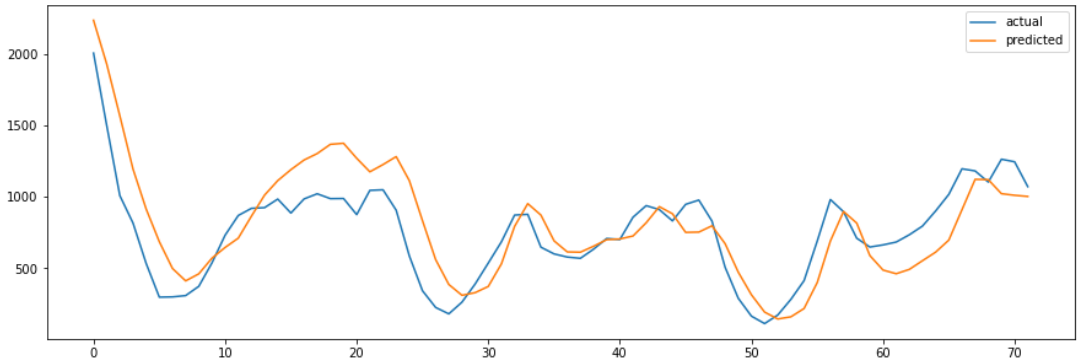
Résultats prédiction de sollicitation d'une base Uber (t+1 sur environ 28 jours).

Test Score: 136.17 RMSE
Test Score (relative): 0.15 RMSE



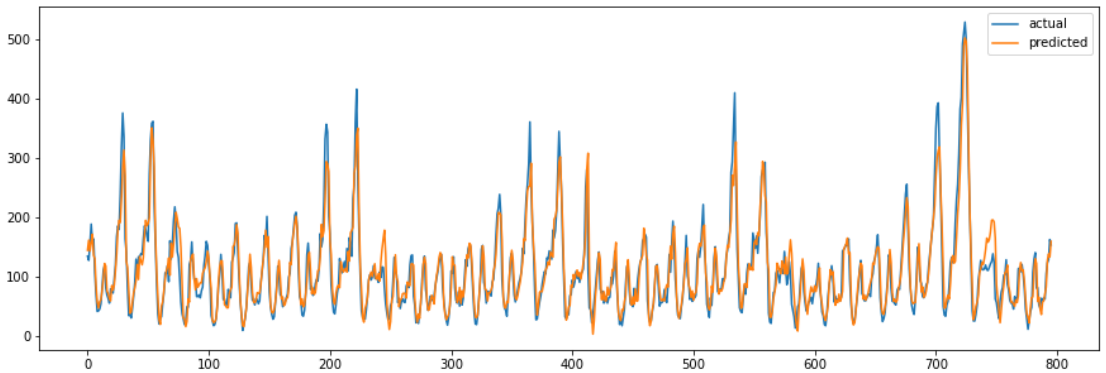
Résultats prédiction de sollicitation d'une base Uber (t+72)

Test Score: 230.82 RMSE
Test Score (relative): 0.31 RMSE



Résultats prédiction de la zone de départ de la course.

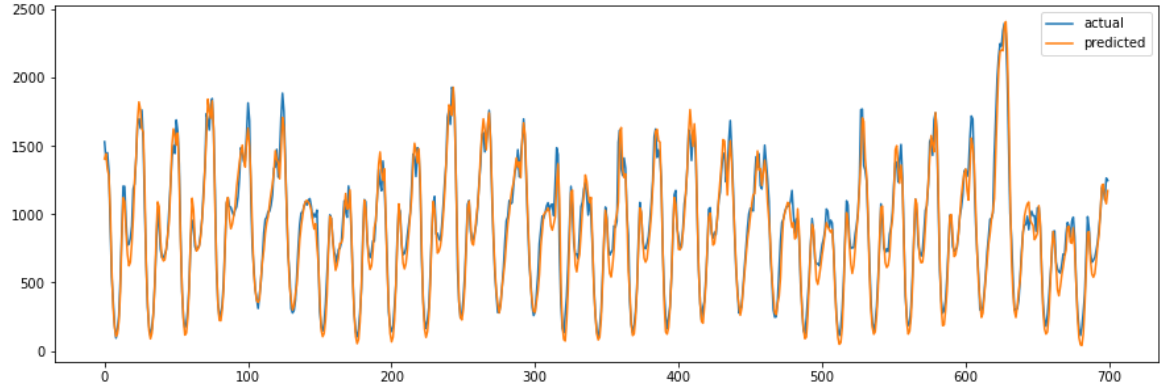
Test Score: 28.26 RMSE
Test Score (relative): 0.26 RMSE



Modèle récurrent (RNN)

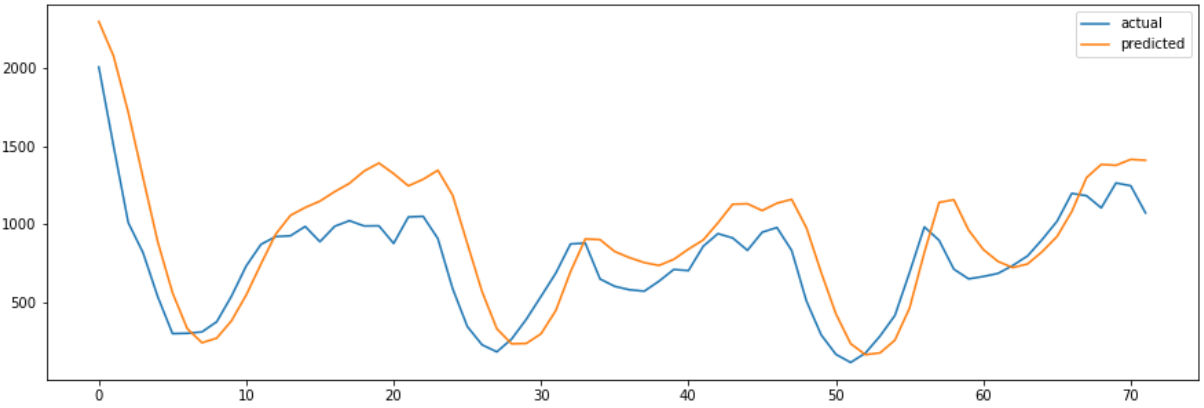
Résultats prédiction de sollicitation d'une base Uber (t+1 sur environ 28 jours).

Test Score: 94.97 RMSE
Test Score (relative): 0.11 RMSE



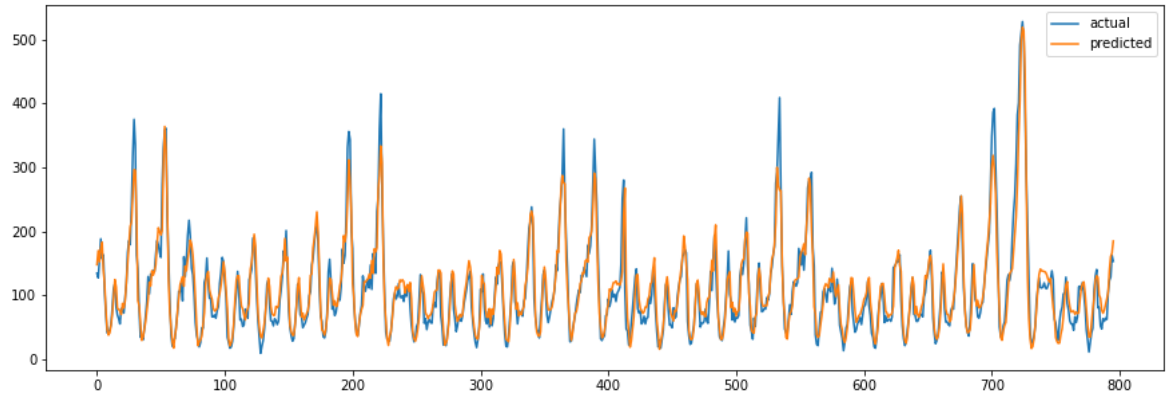
Résultats prédiction de sollicitation d'une base Uber (t+72)

Test Score: 266.95 RMSE
Test Score (relative): 0.35 RMSE



Résultats prédiction de la zone de départ de la course.

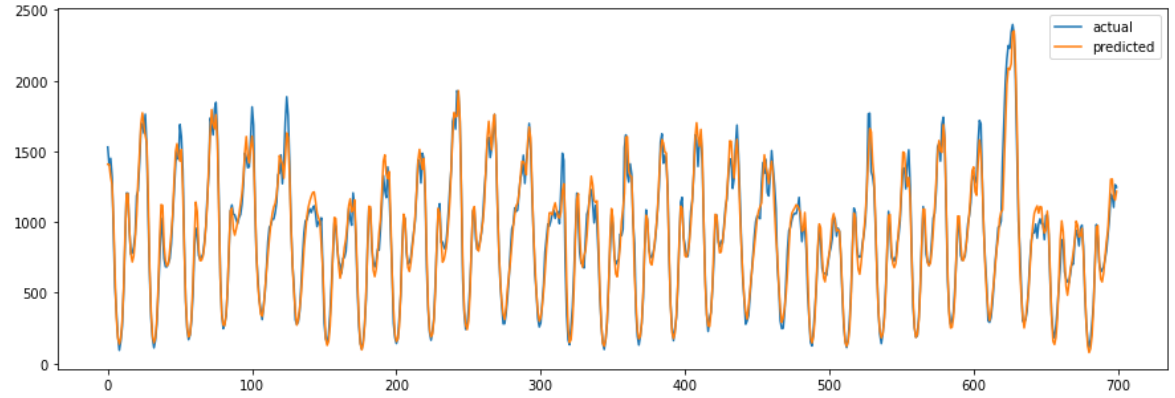
Test Score: 25.59 RMSE
Test Score (relative): 0.24 RMSE



Modèle Long Short Term Memory (LSTM)

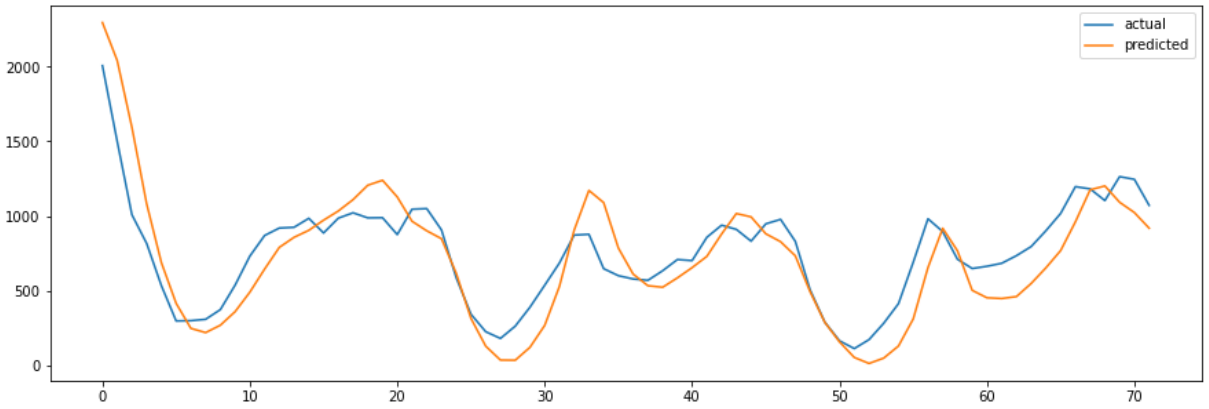
Résultats prédiction de sollicitation d'une base Uber (t+1 sur environ 28 jours).

Test Score: 82.44 RMSE
Test Score (relative): 0.09 RMSE



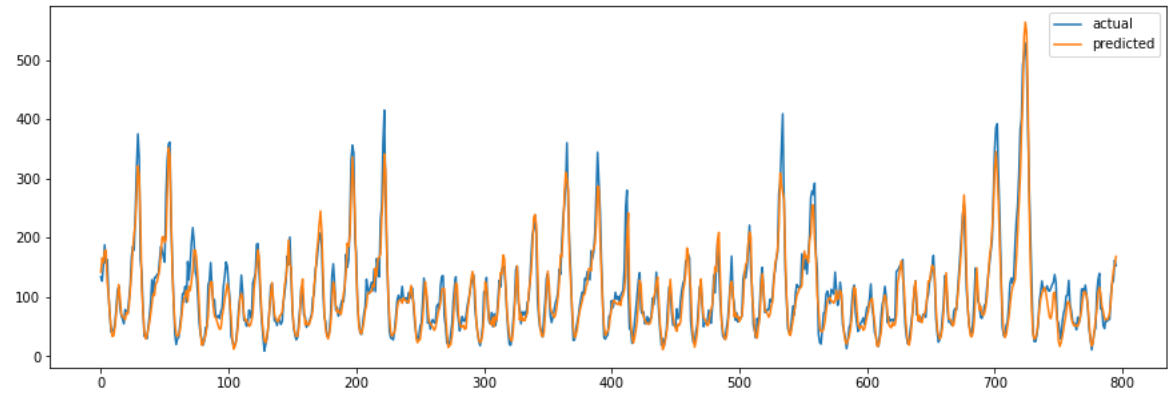
Résultats prédiction de sollicitation d'une base Uber (t+72)

Test Score: 200.69 RMSE
Test Score (relative): 0.27 RMSE

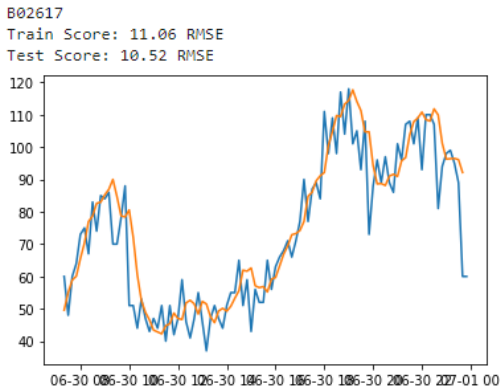


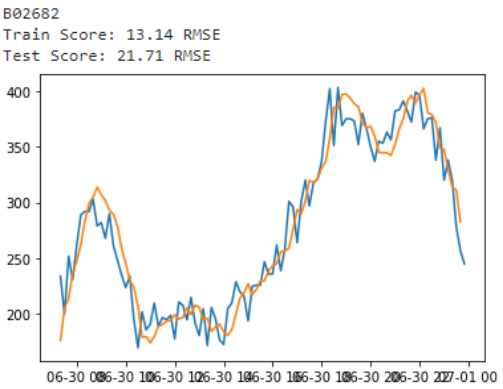
Résultats prédiction de la zone de départ de la course.

Test Score: 23.74 RMSE
Test Score (relative): 0.22 RMSE



Pour essayer de corser encore un peu plus la tâche on va essayer de faire la prédiction du nombre de commandes par 10 minutes au lieu de 1 heure. On va utiliser un LSTM avec les mêmes données que précédemment, sauf qu'on ne va afficher que les 16 dernières heures.



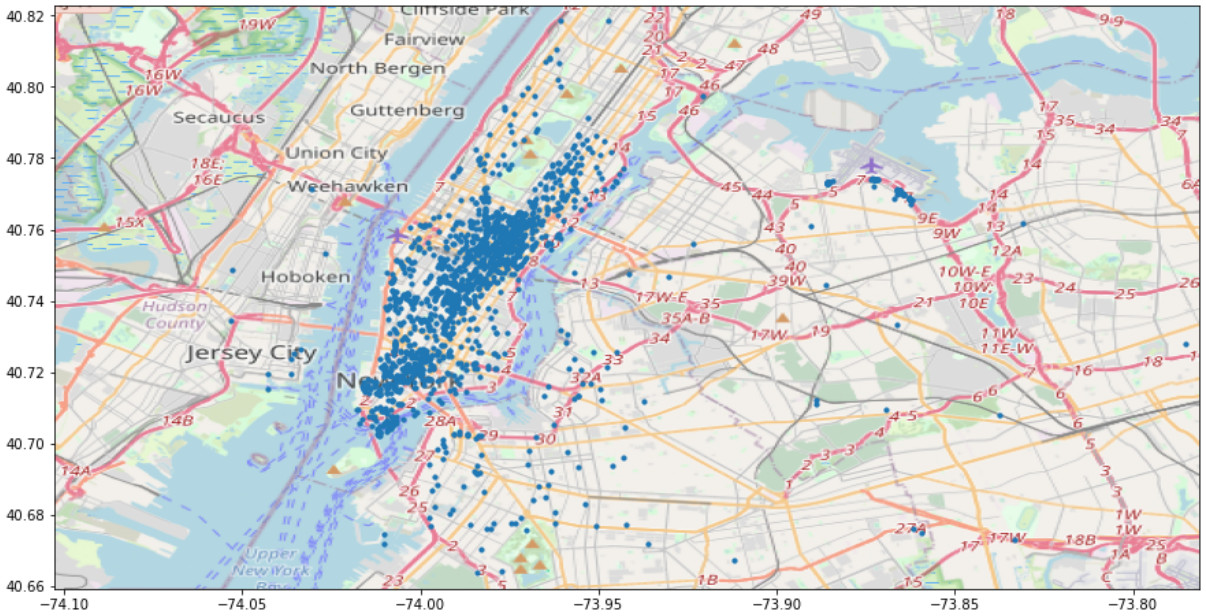


On a donc ici les prédictions sur des données de test (les 16 dernières heures de 2015) pour la base 2617 et 2682. On constate que notre modèle à une très bonne allure générale mais que l'augmentation de la fréquence d'échantillonnage augmente aussi le bruit ce qui mène à une légère perte de précision. Les prédictions sont tout de même satisfaisantes.

Data Visualization

Vu que nous avons les coordonnées de géolocalisation de chaque commande on peut visualiser toutes les commandes qui ont été passé pendant une certaine période.

Toutes les commandes entre 18h et 19h le 3 avril 2014



Toutes les commandes entre 04h et 05h le 3 avril 2014

C’est dans les zones touristiques de Manhattan que l’on trouve le plus de commandes uber.

Comparaison des résultats

En conclusion, on voulait un modèle capable de retranscrire la tendance et la saisonnalité de nos données pour prédire au mieux les pics de commande uber à New York. Le modèle de lissage exponentiel, le modèle de Holt et SARIMAX montrent des résultats intéressants mais clairement pas assez bons pour les prédictions qu'on veut.

Les modèles de Deep Learning sont de loin les plus efficaces, nos expériences montrent qu'ils surpassent grandement les performances de nos approches naïves et statistiques. Le modèle LSTM arrive même à être très précis sur des prédictions toutes les 10 minutes.

Que ça soit par base ou par position, les trois modèles de Deep Learning sont extrêmement précis. On note toutefois, en regardant les courbes et les scores RMSE, que le modèle simple avec des neurones dense est généralement moins performant que le RNN, qui est lui-même très légèrement moins performant que le LSTM. Cela parait assez logique car ces deux derniers ont été conçus pour être bons sur des données de séries temporelles.

En définitive, le modèle Long Short-Term Memory parrait être le plus adapté pour cette tâche, il arrive parfaitement à comprendre les saisonnalités et les tendances des données qu'il rencontre. Il prédit presque parfaitement à t+1 alors que nous ne disposons pas d'autres informations que le nombre de commandes des heures précedentes.

Retours et enseignements

La première difficulté a été le nombre considérable de données à traiter. Nous disposions de presque 19 millions de trajets, il a fallu dans un premier temps en réduire le nombre et rendre ces données utilisables dans nos modèles. Dans un premier temps, nous avons choisis de ne garder et d'analyser qu'une partie de ces données : celles des bases principales c'est à dire celles qui sont le plus présentes dans toutes nos données :

B02682	4661487
B02764	4616220
B02617	3405786
B02598	2680836
B02765	1038379
B02512	393785
B02729	138461
B02788	82707
B00111	80520
B00789	70530
B02774	69018

Comme on peut le constater, certaines bases sont beaucoup plus présentes que les autres. Nous avons choisi de nous limiter aux six bases les plus sollicitées et de se défaire des 278 autres bases. Idem pour les zones géographiques pour lesquelles nous avons décidé de ne garder que les 20 principales.
Dans un second temps, nous avons regroupé les données par intervalle de temps. La difficulté était de choisir le meilleur compromis entre le nombre de données à fournir à nos modèles et la performance de la prédiction. Finalement, nous avons décidé d'avoir une précision à l'heure ou par tranche de 10 minutes qui permettent une meilleure prédiction plutôt qu'une précision au jour ou à la minute par exemple.

Comme perspectives au projet nous avons envisagé d'améliorer la sélection du modèle (et des hyper paramètres). Pour ce faire, d'abord il faudrait implémenter une validation croisée pour augmenter le nombre de test sur différentes données afin d'avoir un score qui soit le plus précis possible pour juger un modèle.
Ensuite, on pourrait améliorer les modèles déjà testés ou non en cherchant les paramètres optimaux avec une Grid Search. On pourrait ainsi optimiser par exemple la fenêtre permettant d'effectuer la prédiction (look_back dans notre code) ou plus généralement l'architecture du réseau de neurones. En effet, dans notre étude, ils sont tous très simplifiés et on pourrait envisager d'en créer des beaucoup plus complexes.
Finalement, pour améliorer la prédiction, notamment à t+n, il nous faudrait d'autres données externes comme par exemple les dates/heures et lieu des événements (sportifs, culturels…) dans la ville de New York ou plus largement les jours fériés/vacances des américains.

Lien Github: https://github.com/clementB94/time_series_project